IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Nathaniel anak Ulah
31/1/2025

Outline

Executive Summary

- In this capstone project, we will predict if the SpaceX Falcon 9 first stage will land successfully using several machine learning classification algorithms.
- The main steps in this project include:
- Data collection, wrangling, and formatting
- Exploratory data analysis
- Interactive data visualization
- Machine learning prediction
- Our graphs show that some features of the rocket launches have a correlation with the outcome of the launches, i.e., success or failure.
- It is also concluded that decision tree may be the best machine learning algorithm to predict if the Falcon 9 first stage will land successfully.

Introduction

- **In this capstone, we will predict if the Falcon 9 first stage will land successfully.**
- SpaceX 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 SpaceX 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 SpaceX for a rocket launch.

- **Most unsuccessful landings are planned.**
- Sometimes, SpaceX will perform a controlled landing in the ocean.

- **The main question we are trying to answer is:**
- For a given set of features about a Falcon 9 rocket launch—which include its payload mass, orbit type, launch site, and so on—will the first stage of the rocket land successfully?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - CSV Data: The primary dataset for SpaceX launch records is provided in a CSV file named spacex_launch_dash.csv. This file contains information about various SpaceX launches, including launch site, payload mass, booster version, and success/failure of each launch.
  - API: Additional data was collected through APIs. Specifically, jupyter-labs-spacex-data-collection-api.ipynb is used to gather real-time data about SpaceX launches from an API endpoint, integrating dynamic updates into the analysis.
  - Web Scraping: In jupyter-labs-webscraping.ipynb, web scraping techniques were used to gather supplementary data, potentially related to launch statistics, site information, or other relevant metrics from online sources.

Methodology

Executive Summary

- Perform Data Wrangling:
  - Handling Missing Values: Missing data was detected and appropriately handled, using imputation for numerical columns or dropping rows with too many missing values.
  - Data Transformation: Column names were standardized, and data types were converted (e.g., converting categorical variables to numerical representations).
  - Filtering and Aggregation: Specific filters were applied, such as selecting only the records with successful or failed launches. The payload mass and launch site were also used as key variables to segment and analyze the data.
  - Outlier Detection: Outliers in the data, especially in columns like payload mass, were identified and dealt with appropriately to ensure the accuracy of subsequent analyses.

Methodology

Executive Summary

- Perform Exploratory Data Analysis (EDA) Using Visualization and SQL:
  - Visualization: In the notebooks like edadataviz.ipynb, a variety of charts were created using Plotly, Dash, and Folium. These visualizations helped reveal trends, such as the relationship between payload mass and launch success. The interactive nature of the dashboards allowed for insights like:
  - Pie Charts: Created using plotly.express to show the percentage of successful vs. failed launches at different sites (using the dropdown selection).
  - Scatter Plots: Displaying the correlation between payload mass and the success of the launches, including a breakdown by booster version.
  - SQL Queries: The jupyter-labs-eda-sql-coursera_sqllite.ipynb notebook integrates SQL queries to filter and aggregate data based on different criteria like launch site, payload mass, and launch success. These queries were vital in helping with the aggregation and detailed examination of the dataset to draw meaningful conclusions.

Methodology

Executive Summary

- Perform Interactive Visual Analytics Using Folium and Plotly Dash:
  - Select Launch Sites: A dropdown menu enabled the selection of different SpaceX launch sites, showing success/failure metrics specific to each site.
  - Visualize Success vs. Failure: A pie chart was updated dynamically based on the launch site selection to show success vs. failure counts.
  - Payload Mass Slider: An interactive slider allowed users to filter launches based on payload mass, adjusting the scatter plot to show the relationship between payload and success for the selected range.
  - Geospatial Mapping: Folium was potentially used to visualize the launch sites on a map, helping to understand spatial patterns related to launch site success.

Methodology

Executive Summary

- Perform Predictive Analysis Using Classification Models:
    - To predict the outcome of SpaceX launches, various classification models were applied in the SpaceX_Machine Learning Prediction_Part_5.ipynb notebook:
    - Model Selection: The classification algorithms tested included:Logistic RegressionSupport Vector Machine (SVM)Decision Trees
    - K-Nearest Neighbors (KNN)Model Training: The models were trained using key features such as payload mass, booster version, and launch site. The training process involved splitting the data into training and test sets, followed by fitting the models to the training data.
    - Hyperparameter Tuning: Models were tuned by adjusting hyperparameters (e.g., depth of the decision tree or kernel in SVM) to improve accuracy. Techniques like grid search were used to optimize these parameters.
    - Evaluation: The models were evaluated using metrics such as accuracy, precision, and recall. The best model was selected based on the highest accuracy score. As seen in the notebook, the Decision Tree model outperformed others with an accuracy of approximately 0.889.Prediction: After model evaluation, the best-performing model was used to predict the success or failure of upcoming SpaceX launches.

Data Collection

- Sources of Data:
- SpaceX Launch Data: The main dataset is sourced from the spacex_launch_dash.csv file. This dataset contains critical information about SpaceX launches, including: Launch sitePayload mass (kg)Booster version categoryLaunch success or failureAPI
- Integration:Data was also collected from the SpaceX API to gather real-time launch information. This API fetches data dynamically, providing up-to-date information on launches that isn't available in the static CSV.
- Web Scraping:To complement the CSV data and API, web scraping techniques were utilized to gather additional details about SpaceX launches. This was implemented in the jupyter-labs-webscraping.ipynb notebook, which extracts data from online sources.

Data Collection – SpaceX API

- Github url: https://github.com/matkool-ai/IBM-Applied-Data-Science-Capstone-Project/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)
```

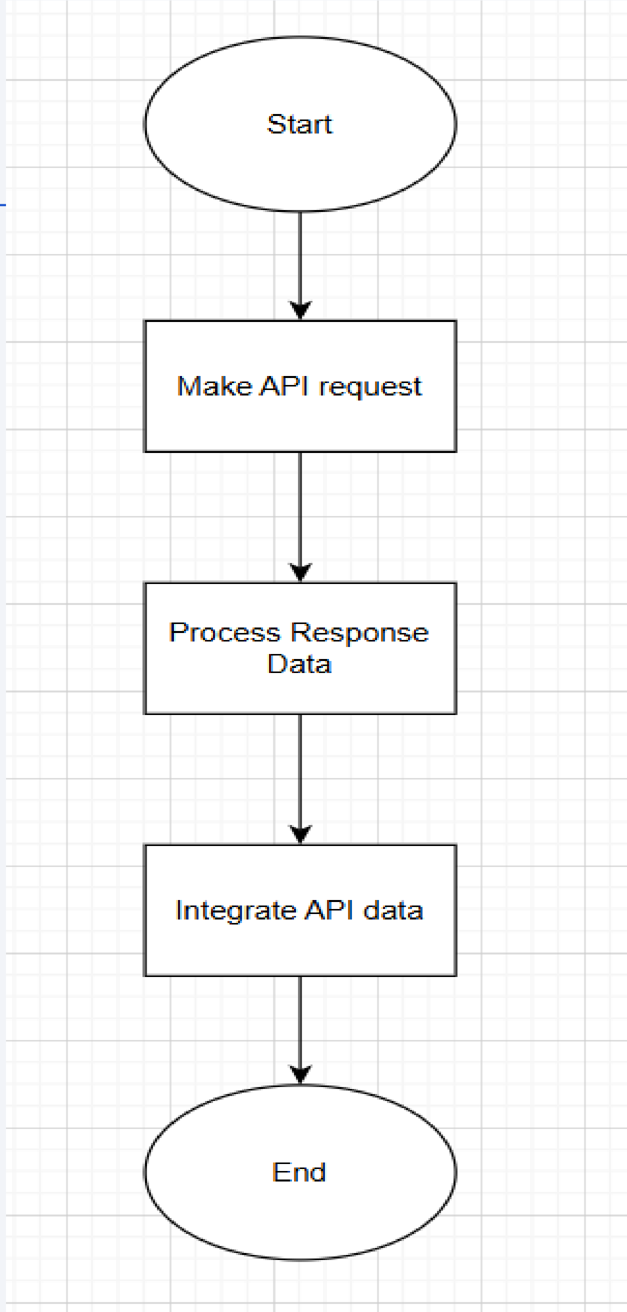
2. Use json_normalize method to convert json result to dataframe

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

- GitHub URL: https://github.com/matkool-ai/IBM-Applied-Data-Science-Capstone-Project/blob/main/jupyter-labs-webscraping.ipynb

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

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

In [5]:
```python
# 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]:
```python
# 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

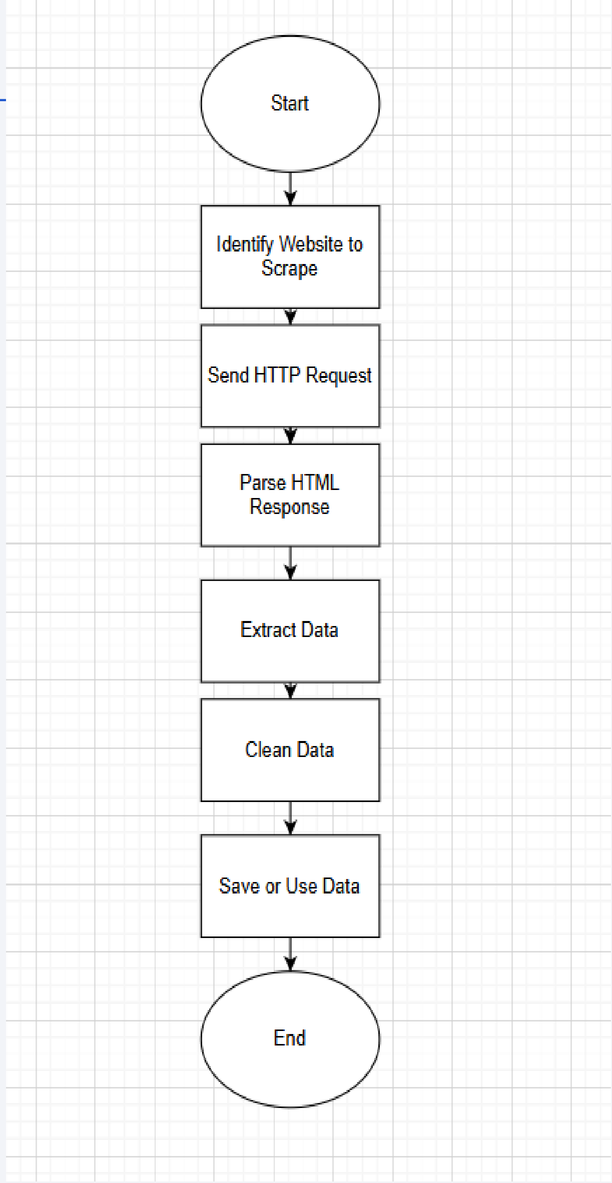
In [7]:
```python
# Use soup.title attribute
soup.title
```

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

3. Extract all column names from the HTML table header

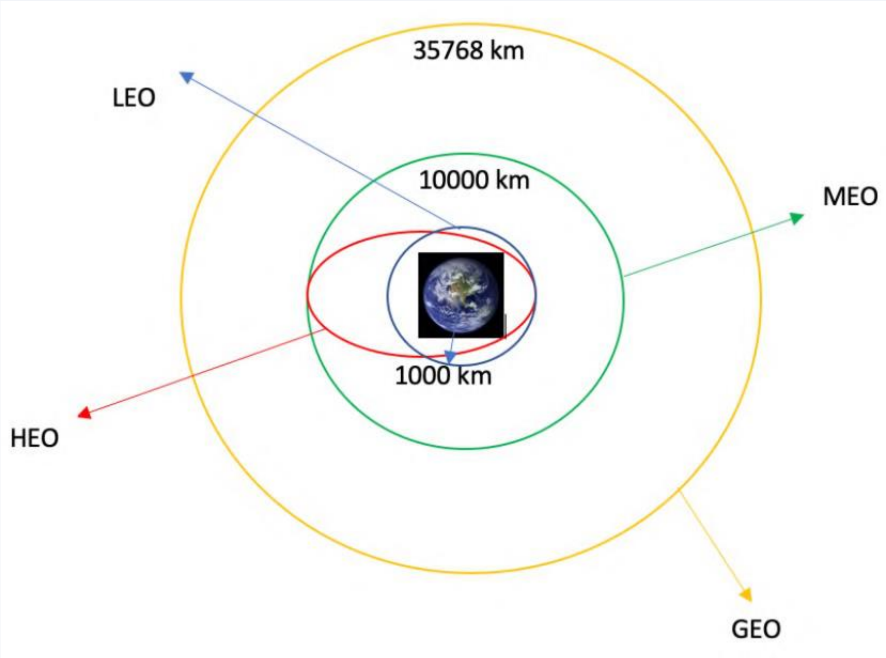
In [10]:
```python
column_names = []

# 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

- Data Wrangling is the process of cleaning and unifying messy and complex data sets for easy access and Exploratory Data Analysis (EDA).
- We will first calculate the number of launches on each site, then calculate the number and occurrence of mission outcome per orbit type.
- We then create a landing outcome label from the outcome column. This will make it easier for further analysis, visualization, and ML.
- Lastly, we will export he result to a CSV.

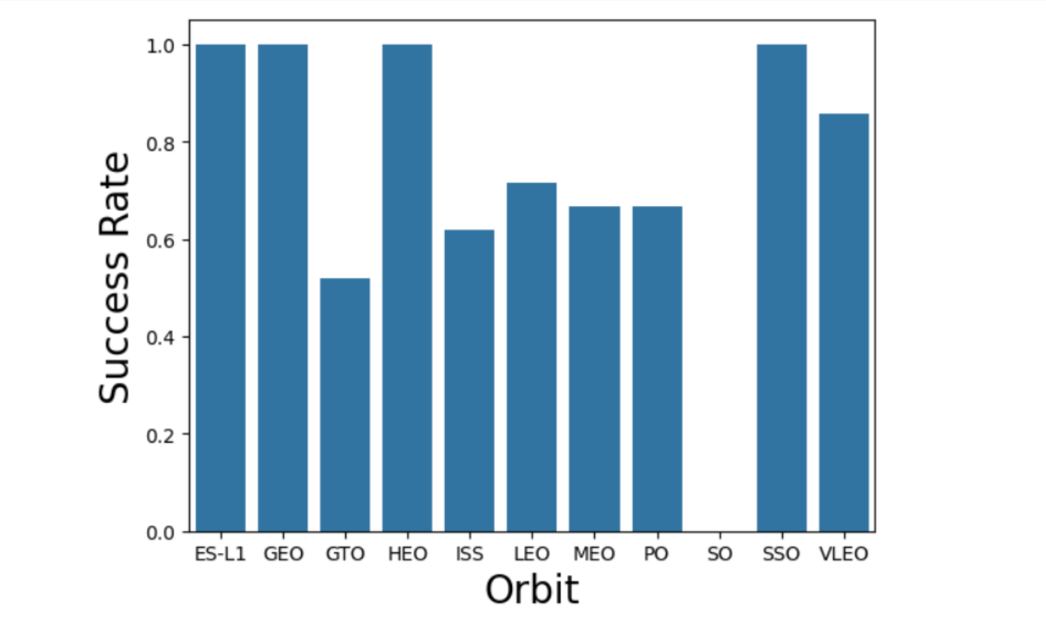


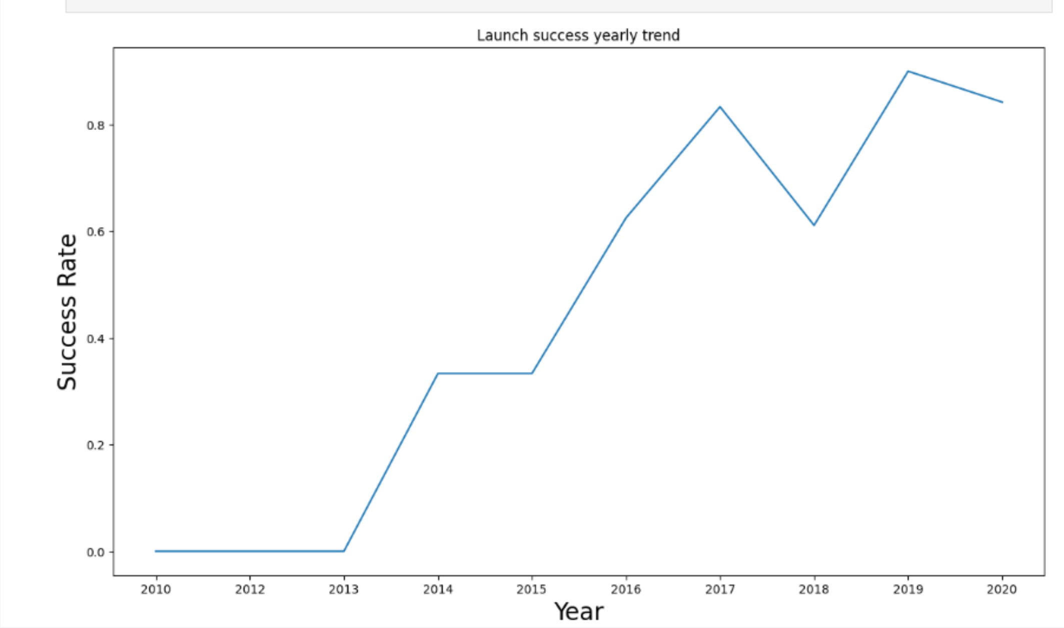
https://github.com/matkool-ai/IBM-Applied-Data-Science-Capstone-Project/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

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.

https://github.com/matkool-ai/IBM-Applied-Data-Science-Capstone-Project/blob/main/edadataviz.ipynb





EDA with SQL

- We loaded the SpaceX dataset into a PostgreSQL database without leaving the jupyternotebook.
- 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 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.

- Github URL: https://github.com/matkool-ai/IBM-Applied-Data-Science-Capstone-Project/blob/main/jupyter-labs-eda-sql-coursera_sqllite.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.

Github URL: https://github.com/matkool-ai/IBM-Applied-Data-Science-Capstone-Project/blob/main/lab_jupyter_launch_site_location.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.

GITHUB URL: https://github.com/matkool-ai/IBM-Applied-Data-Science-Capstone-Project/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- We loaded the data using numpyand 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.

GITHUB URL: https://github.com/matkool-ai/IBM-Applied-Data-Science-Capstone-Project/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

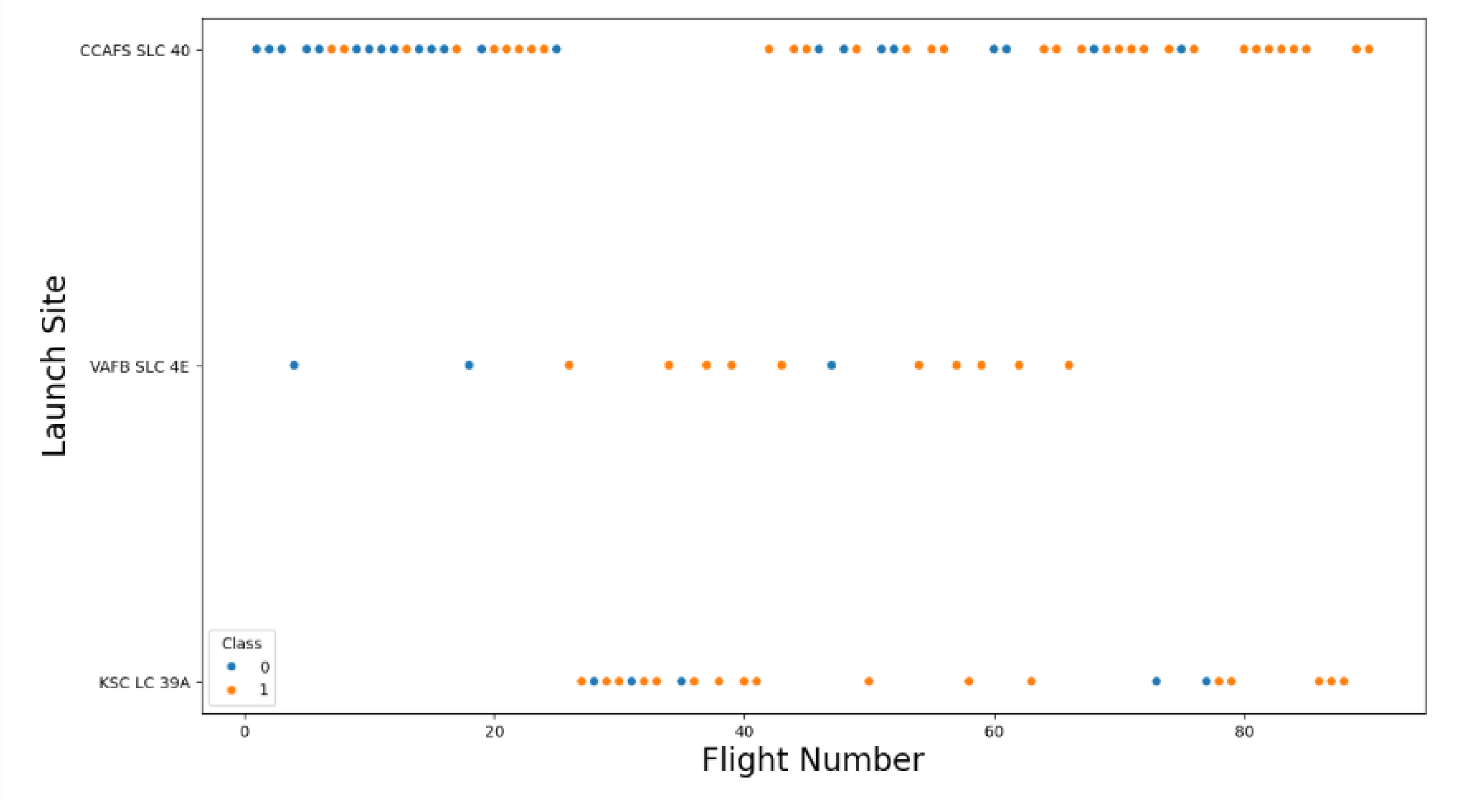
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

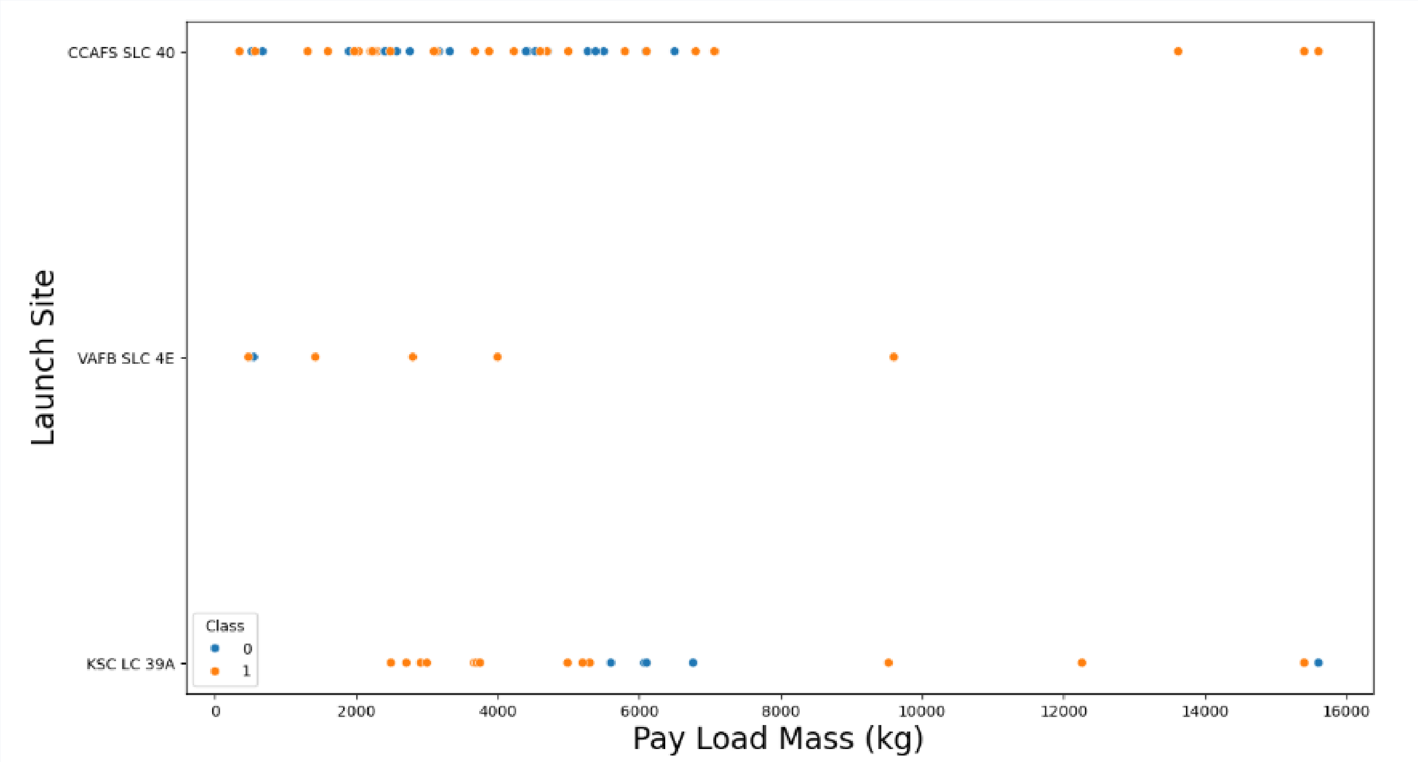
Flight Number vs. Launch Site

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



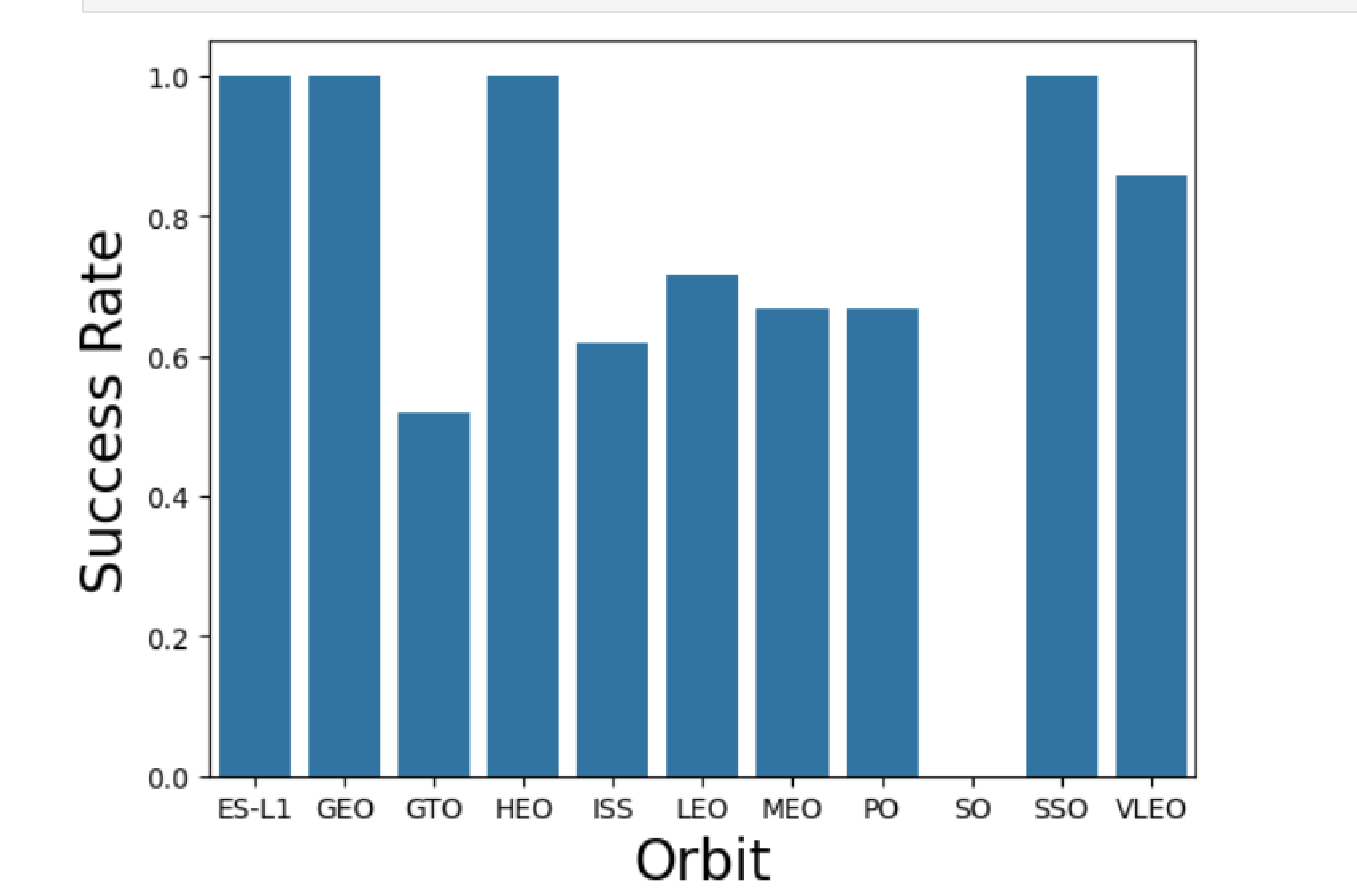
Payload vs. Launch Site

- The greater the payload mass for launch site CCAFS SLC 40 the higher the success rate for the rocket



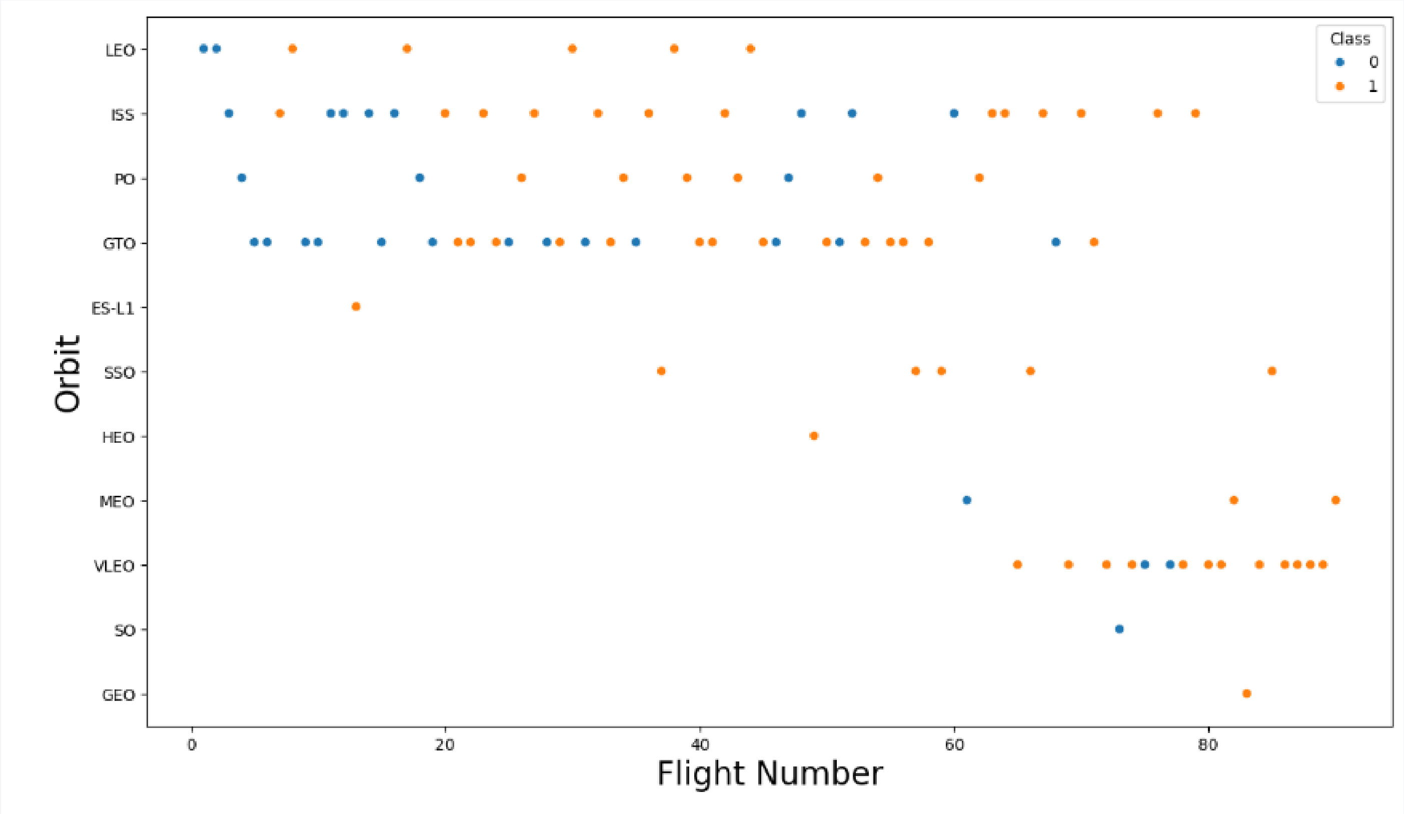
Success Rate vs. Orbit Type

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



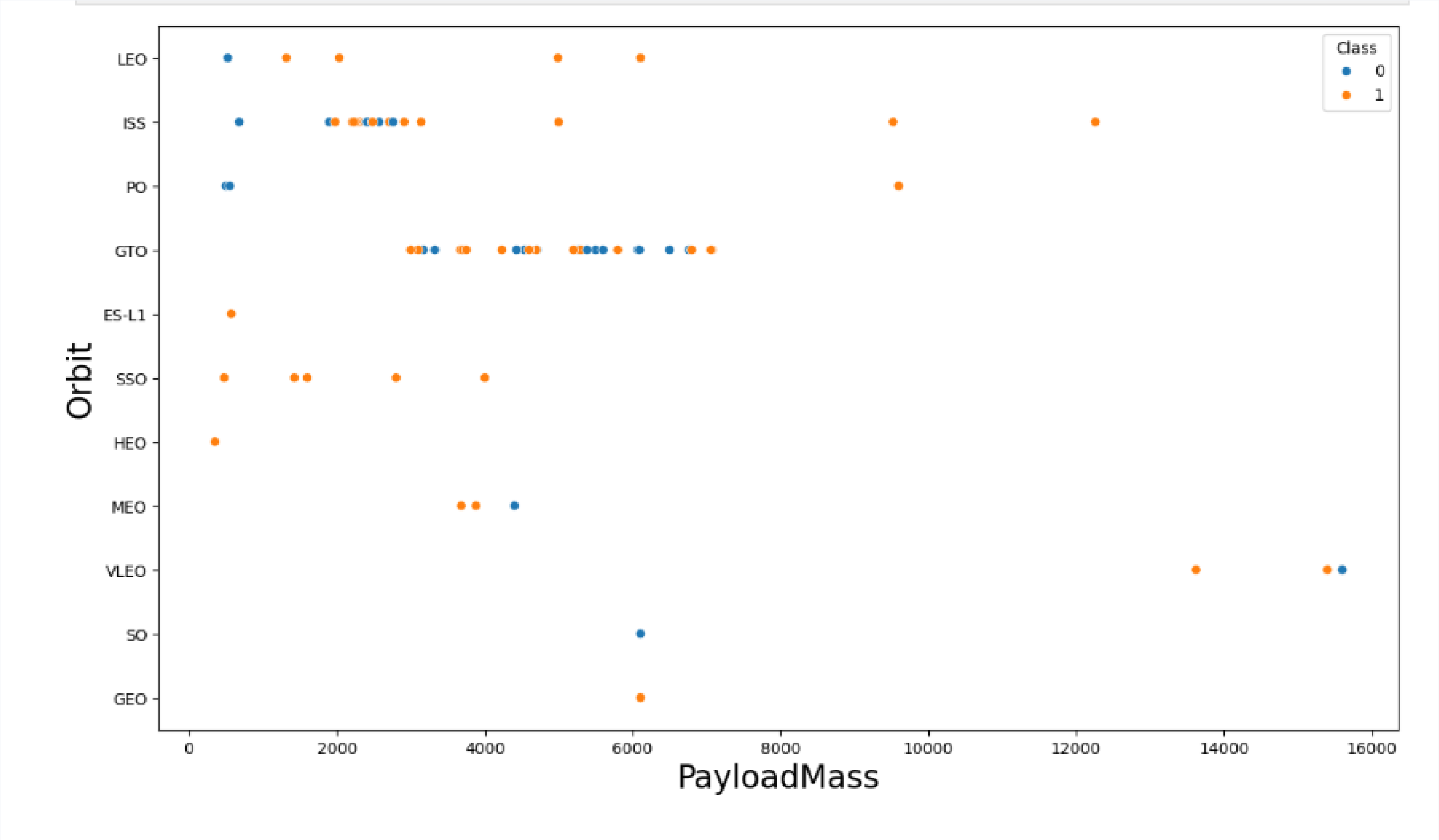
Flight Number vs. Orbit Type

- The plot below shows the Flight Number vs. Orbit type. We observe 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.



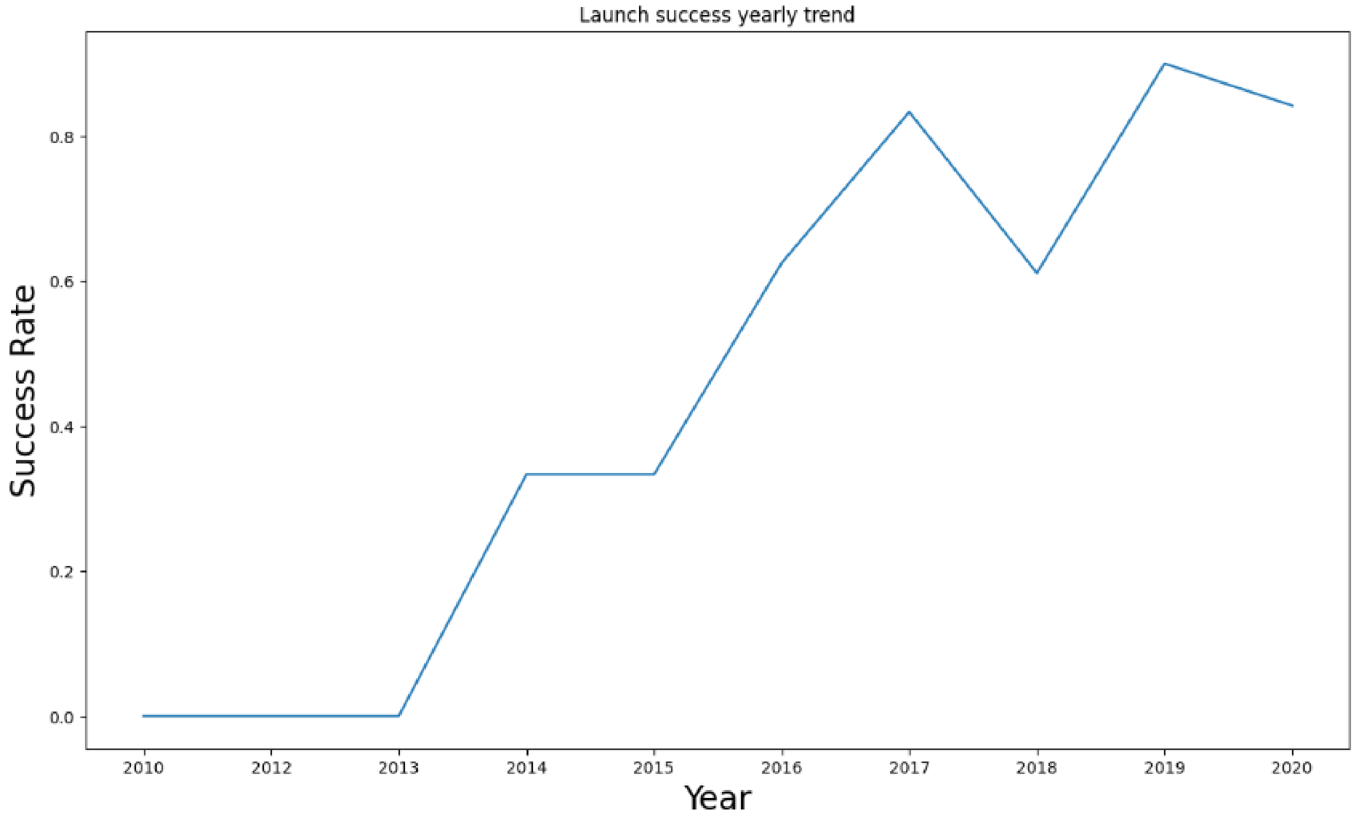
Payload vs. Orbit Type

- We can 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.

Task 1

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

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

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

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

Launch Site Names Begin with 'CCA'

- We used the query BESIDE to display 5 records where launch sites begin with `CCA`

Task 2

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

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

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

7]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Missi
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	
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	
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	

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 [18]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS "Total payload mass by NASA (CRS)" FROM SPACEXTBL WHERE CUSTOMER = 'NASA (CRS)';
```

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

Out[18]:

Total payload mass by NASA (CRS)
45596

Average Payload Mass by F9 v1.1

- We calculated the average payload mass carried by booster version F9 v1.1 as 2928.4

45596

Task 4

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

In [19]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS "Average payload mass by Booster Version F9 v1.1" FROM SPACEXTBL WHERE BOOSTER_VERSION
```

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

Out[19]:

Average payload mass by Booster Version F9 v1.1
2928.4

First Successful Ground Landing Date

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

Task 5

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

Hint:Use min function

```
22]: %sql SELECT MIN(DATE) AS "Date of first successful landing outcome in ground pad" FROM SPACEXTBL WHERE landing_outcome = 'Su
```

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

22]:

Date of first successful landing outcome in ground pad
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- We used the WHEREclause to filter for boosters which have successfully landed on drone ship and applied the ANDcondition 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 [24]:
```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE landing_outcome = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ BETWEEN 400(
```

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

Out[24]:

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 MissionOutcome was a success or a failure.

Task 7

List the total number of successful and failure mission outcomes

In [25]:
```
%sql SELECT number_of_success_outcomes, number_of_failure_outcomes FROM (SELECT COUNT(*) AS number_of_success_outcomes FROM
```

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

Out[25]:

number_of_success_outcomes	number_of_failure_outcomes
100	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.

Task 8

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

In [26]:
```
%sql SELECT DISTINCT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ =(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

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

Out[26]:

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 WHEREclause, LIKE, AND, and BETWEENconditions to filter for failedlanding outcomes in drone ship, their booster versions, andlaunch site names for year 2015

Task 9

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

[28]:

```
%sql SELECT substr(Date, 6, 2) AS Month, BOOSTER_VERSION, LAUNCH_SITE, LANDING_OUTCOME FROM SPACEXTBL WHERE substr(Date, 1,
```

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

t[28]:

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

- We selected Landing outcomes and the COUNTof landing outcomes from the data and used the WHEREclause to filter for landing outcomes BETWEEN2010-06-04 to 2010-03-20.
- 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.

Task 10

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

In [32]:

```
%sql SELECT landing_outcome, COUNT(landing_outcome) AS Landing_Count FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '20
```

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

Out[32]:

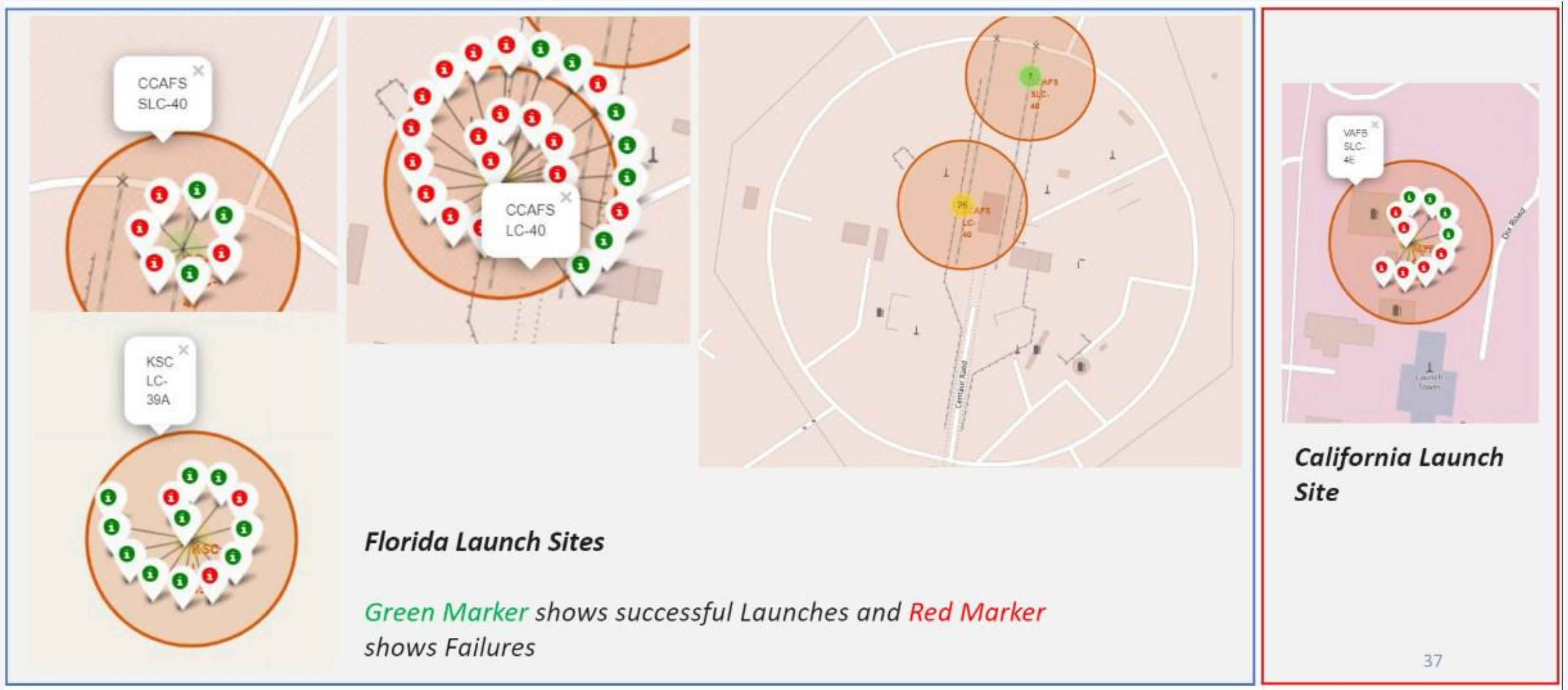
Landing_Outcome	Landing_Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

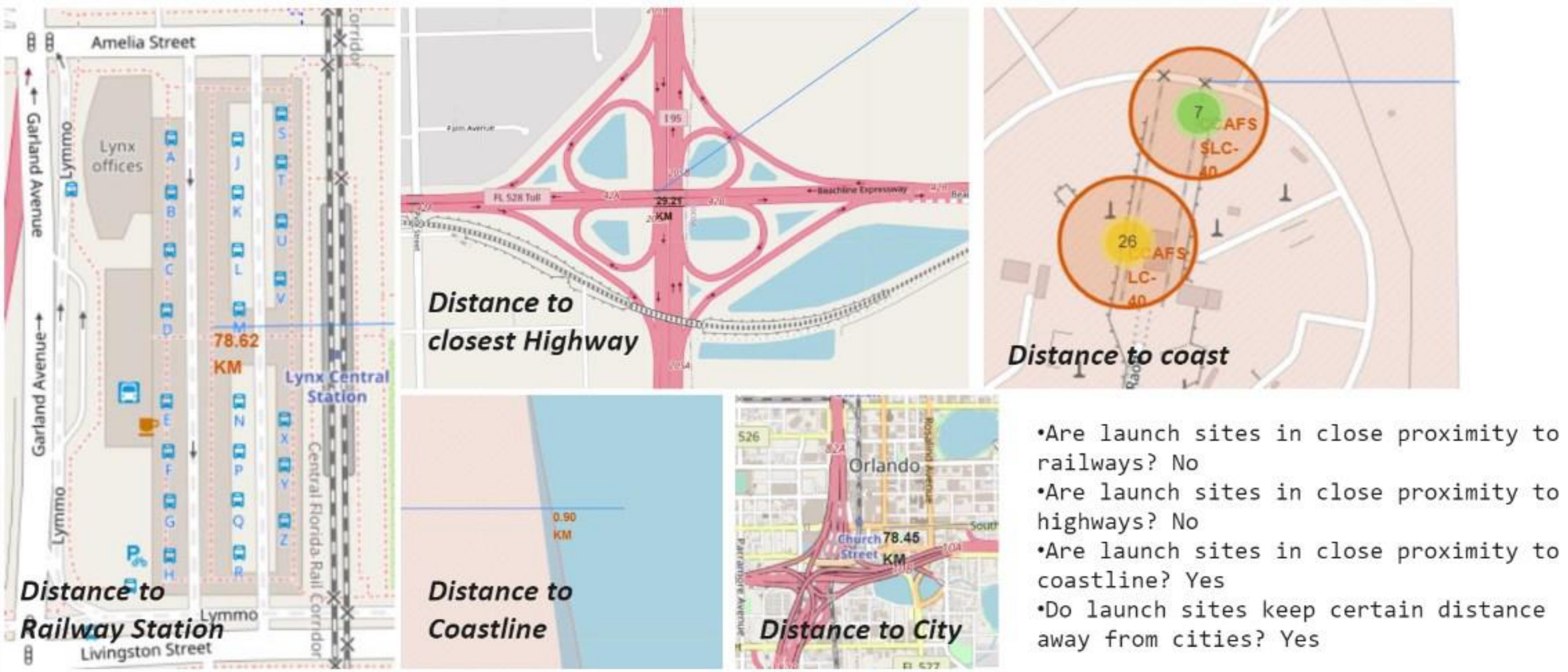
Launch Sites Proximities Analysis

All launch sites global map markers

Markers showing launch sites with color labels



Launch Site distance to landmarks

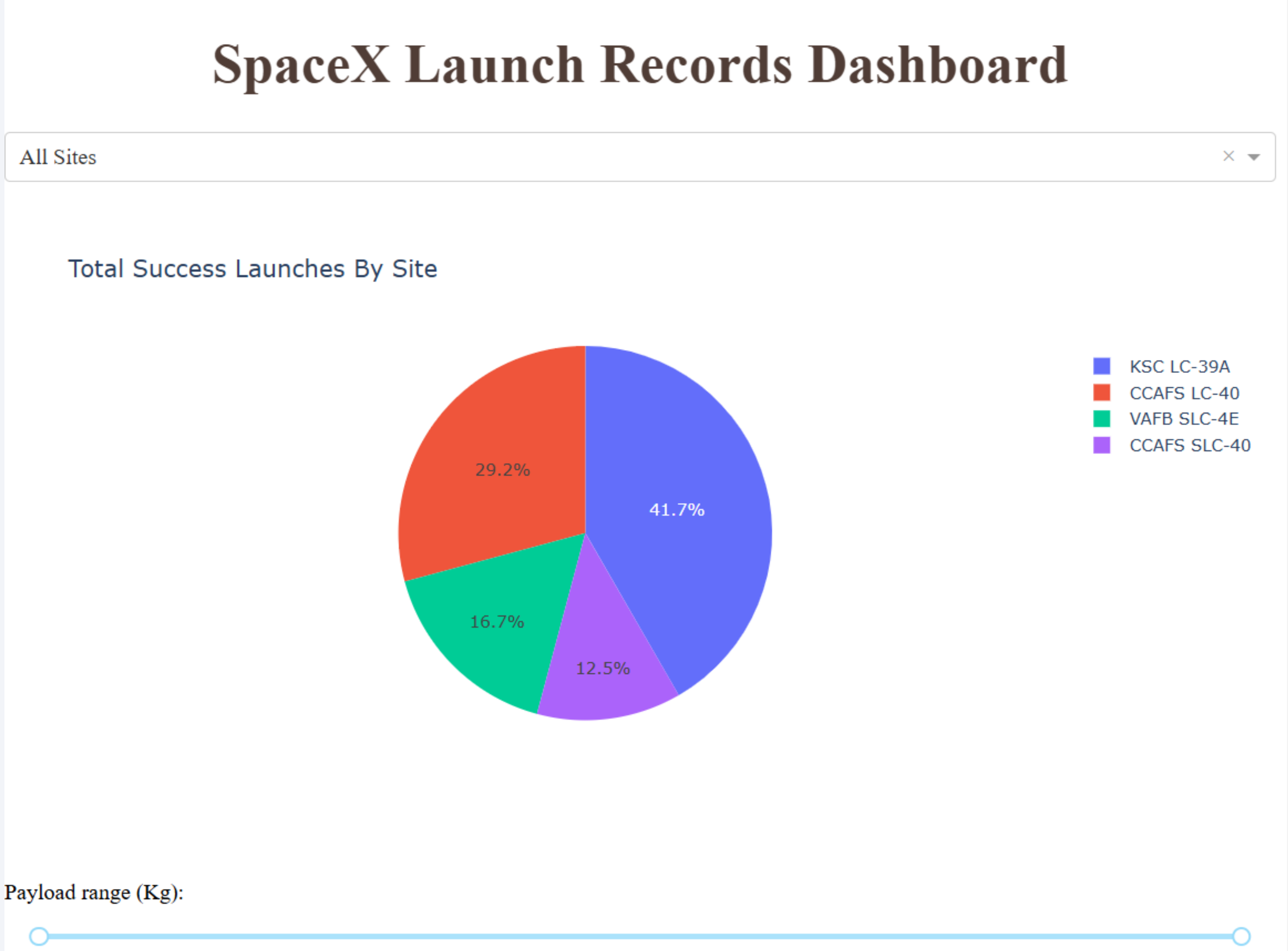


•Are launch sites in close proximity to railways? No
•Are launch sites in close proximity to highways? No
•Are launch sites in close proximity to coastline? Yes
•Do launch sites keep certain distance away from cities? Yes

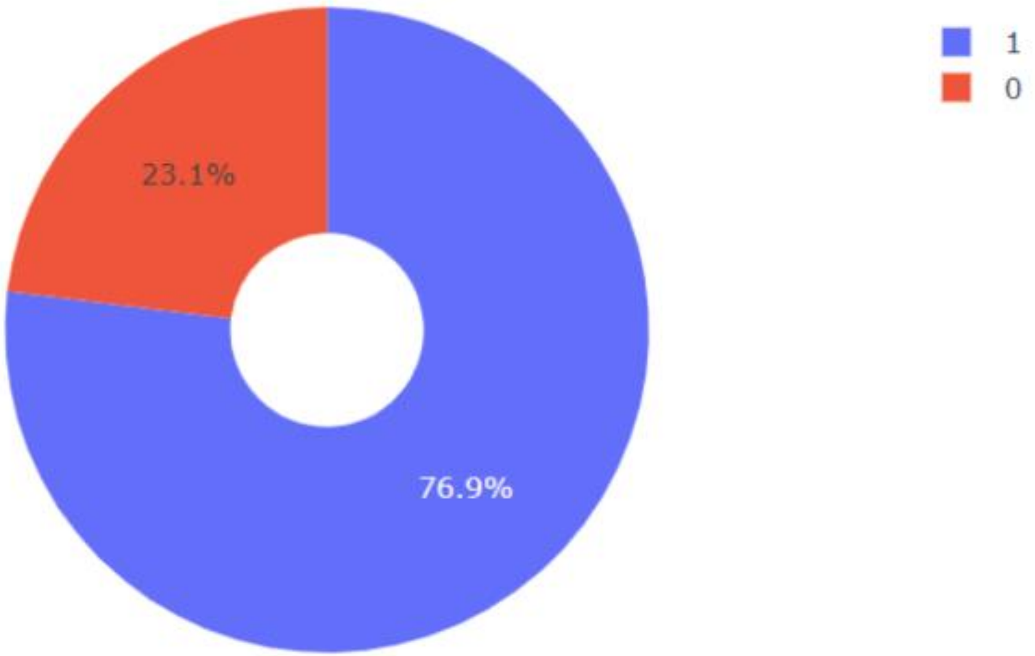
Section 4

Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site

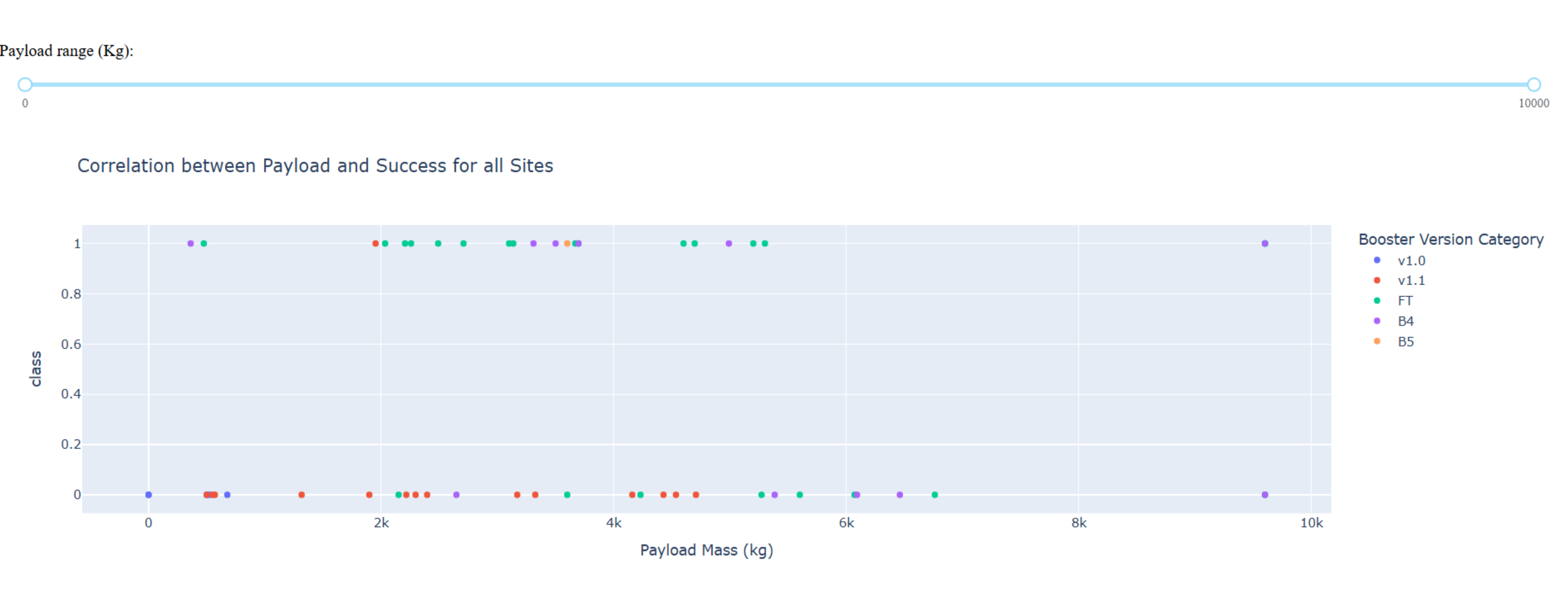


Pie chart showing the Launch site with the highest launch success ratio



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



Section 5

Predictive Analysis (Classification)

Classification Accuracy

- As we can see, by using the code as below: we could identify that the best algorithm to be
- the Tree Algorithm which have the highest classification accuracy

TASK 8

Create a decision tree classifier object then create a `GridSearchCV` object `tree_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

In [26]:
```
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()
```

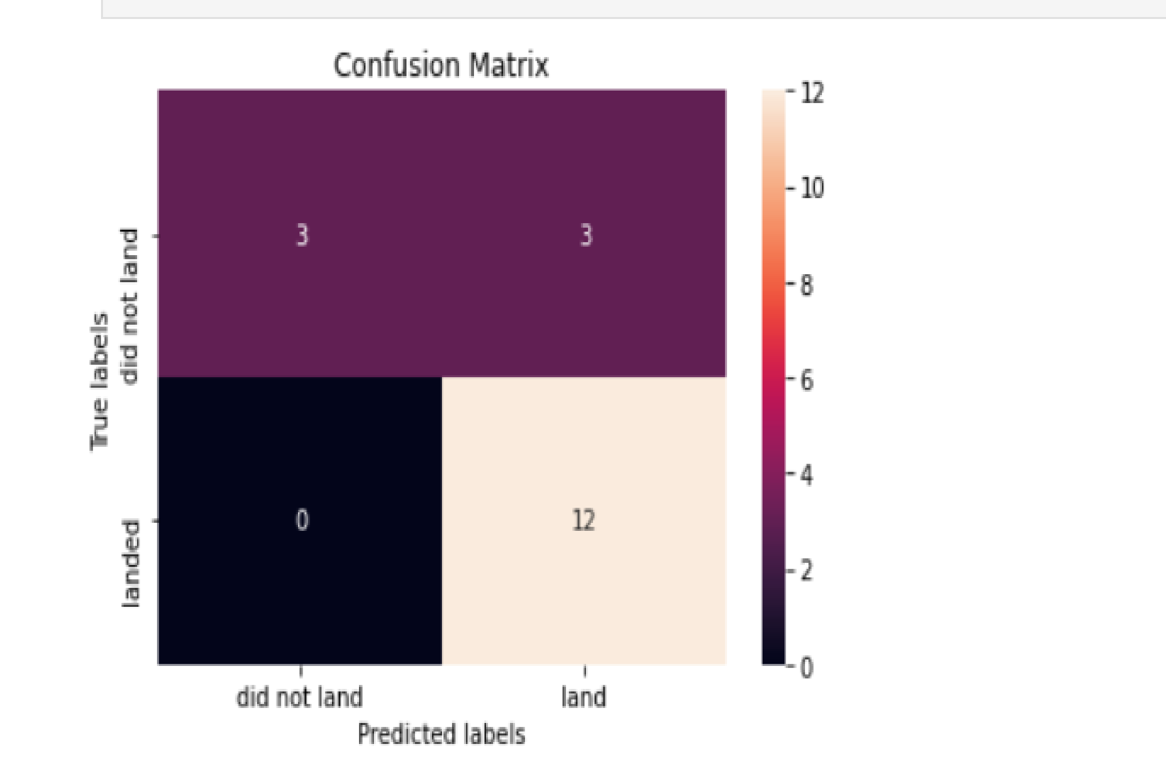
In [29]:
```
tree_cv = GridSearchCV(tree, parameters, scoring='accuracy', cv=10)
tree_cv = tree_cv.fit(X_train, Y_train)
```

In [30]:
```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 4, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'best'}
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:
- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- 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 of any sites.
- The Decision tree classifier is the best machine learning algorithm for this task.

Thank you!