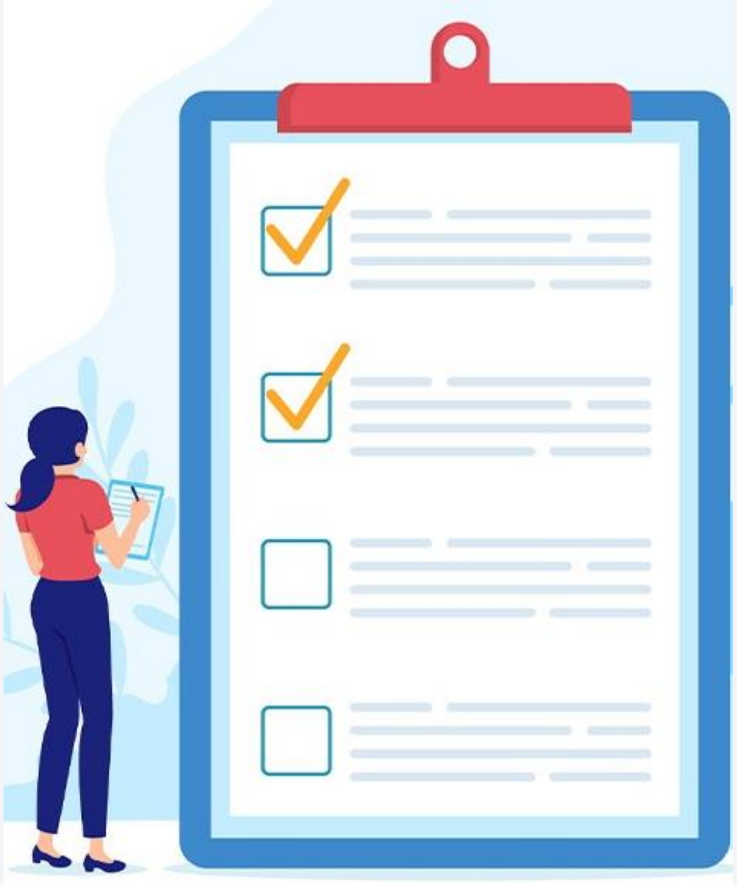
IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

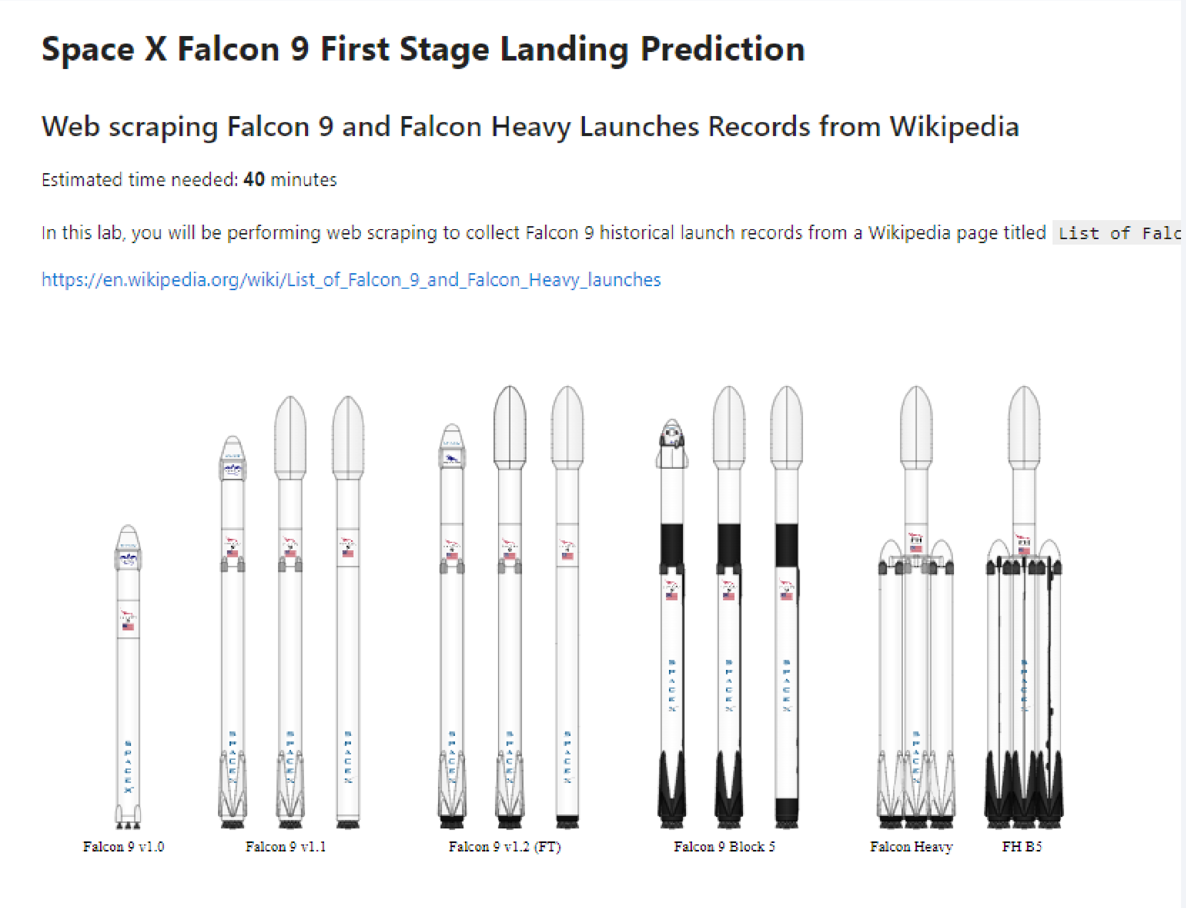
Happy Nkanta Monday
July 2, 2024

Outline



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

Executive Summary



- SpaceY, a new contender in the commercial rocket launch market, aims to compete with SpaceX.
- SpaceX offers launch services starting at $62 million for missions where some fuel is reserved to land the first stage rocket booster, allowing for its reuse. Publicly available information from SpaceX suggests that the cost to build a first stage Falcon 9 booster is over $15 million, excluding R&D costs and profit margins.
- This report's models predict the successful landing of the first stage rocket booster with an accuracy of 83.3%, based on mission parameters such as payload mass and target orbit.
- Consequently, SpaceY can leverage these landing predictions to make more informed and competitive bids against SpaceX by using them as a proxy for estimating launch costs.

Introduction

- This report has been created as part of the Applied Data Science Capstone course. In this project, I assume the role of a data scientist working for SpaceY, a new entrant in the rocket launch industry.

- The data science findings and models presented in this report will enable SpaceY to make more strategic and competitive bids against SpaceX for rocket launch services.

Business Problem

- SpaceX offers Falcon 9 rocket launches at a cost of $62 million when the first stage of the rocket is reusable. The construction cost for this first stage is estimated to be over $15 million, excluding R&D costs and profit margins.
- However, there are instances where SpaceX opts to forgo the reuse of the first stage due to specific mission parameters, such as payload requirements, target orbit, and customer preferences.
- This report seeks to accurately predict the likelihood of the first stage rocket landing successfully, serving as a proxy to estimate the overall cost of a launch.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

Data Collection

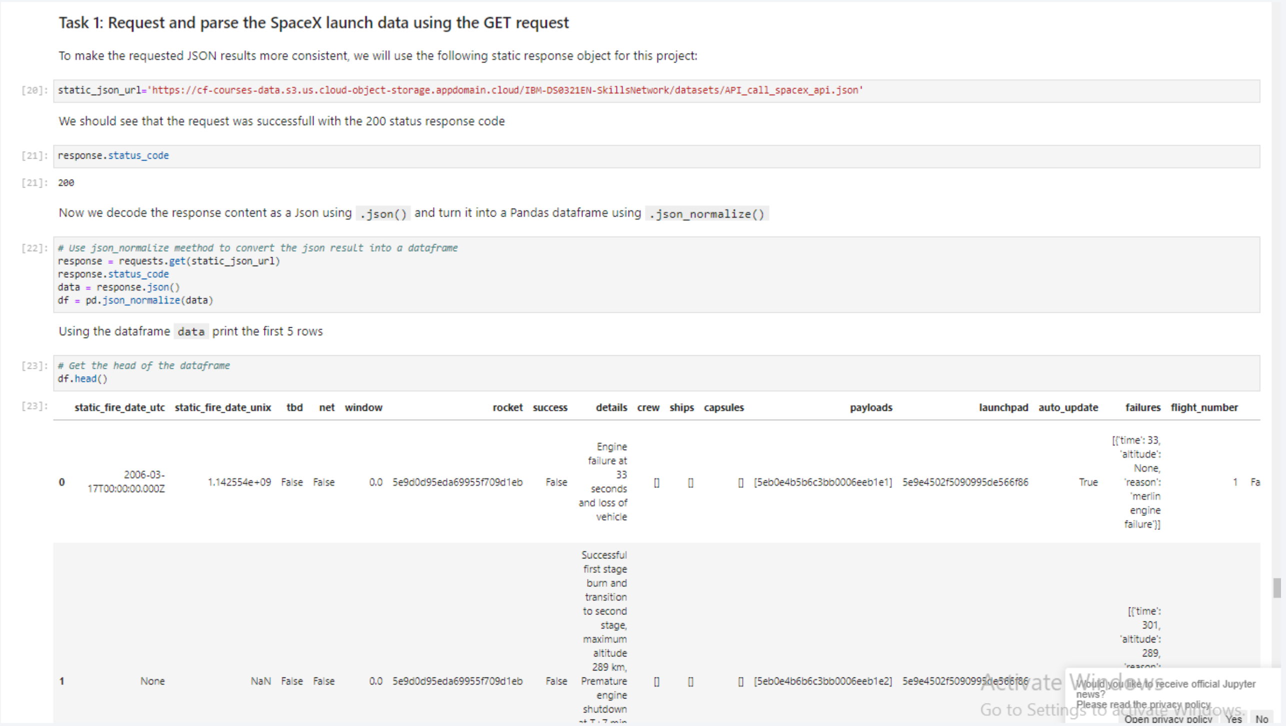
- API
  - Collected historical launch data from an Open Source REST API for SpaceX.
  - Utilized GET requests to retrieve and parse the SpaceX launch data.
  - Filtered the dataset to include only Falcon 9 launches.
  - Imputed missing payload mass values for classified missions with the mean.
- Web Scraping
  - Gathered historical launch data from the Wikipedia page titled "List of Falcon 9 and Falcon Heavy Launches."
  - Accessed the Wikipedia URL for the Falcon 9 launch data.
  - Extracted all column and variable names from the HTML table headers.
  - Parsed the table and converted it into a Pandas DataFrame.

Data Collection – SpaceX API

Space X API

- Collected historical launch data from an Open Source REST API for SpaceX.
- Utilized GET requests to retrieve and parse the SpaceX launch data.
- Filtered the dataset to include only Falcon 9 launches.
- Imputed missing payload mass values for classified missions with the mean.

- Add the GitHub URL of the completed SpaceX API calls notebook (must include completed code cell and outcome cell), as an external reference and peer-review purpose



https://github.com/happymondaynkanta/Data-Science-Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Data Collection - Scraping

- Gathered historical launch data from the Wikipedia page titled "List of Falcon 9 and Falcon Heavy Launches."
- Accessed the Wikipedia URL for the Falcon 9 launch data.
- Extracted all column and variable names from the HTML table headers.
- Parsed the table and converted it into a Pandas DataFrame.
- Add the GitHub URL of the completed web scraping notebook, as an external reference and peer-review purpose

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.

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

response = requests.get(static_url)
```

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

[6]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.text, 'html.parser')
```

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

[7]:
```
# Use soup.title attribute
print(soup.title)
```

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

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

[11]:
```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`

# Find all tables on the wiki page
html_tables = soup.find_all('table')

# Print the third table to check its content
target_table = html_tables[2]
print(target_table.prettify())


# Extract all column/variable names from the HTML table header
# Extract the header row from the target table
headers = target_table.find_all('th')
column_names = [extract_column_from_header(header) for header in headers]

# Print the extracted column names
print(column_names)
```

```
<table class="wikitable plainrowheaders collapsible" style="width: 100%;">
 <tbody>
```


https://github.com/happymondaynkanta/Data-Science-Capstone/blob/main/jupyter-labs-webscraping_new.ipynb

Data Wrangling

Conducted an exploratory analysis to determine the appropriate labels for training supervised models.

Calculated the number of launches at each site and the frequency of each orbit type.

Analyzed mission outcomes based on orbit type to identify patterns.

Derived a landing outcome training label from the 'Outcome' column, creating the 'Class' label.

- Training Label: 'Class'

Class = 0: First stage booster did not land successfully

None None: Landing not attempted

None ASDS: Launch failure, unable to attempt landing

False ASDS: Drone ship landing failed

False Ocean: Ocean landing failed

False RTLS: Ground pad landing failed

Class = 1: First stage booster landed successfully

True ASDS: Drone ship landing succeeded

True RTLS: Ground pad landing succeeded

True Ocean: Ocean landing succeeded

Import Libraries and Define Auxiliary Functions

We will import the following libraries.

[1]:
```
# Pandas is a software library written for the Python programming language for data manipulation and analysis.
import pandas as pd
#NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays
import numpy as np
```

Data Analysis

Load Space X dataset, from last section.

[2]:
```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv")
df.head(10)
```

[2]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632093
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857
5	6	2014-01-06	Falcon 9	3325.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1005	-80.577366	28.561857
6	7	2014-04-18	Falcon 9	2296.000000	ISS	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1006	-80.577366	28.561857
7	8	2014-07-14	Falcon 9	1316.000000	LEO	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1007	-80.577366	28.561857
8	9	2014-08-05	Falcon 9	4535.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1008	-80.577366	28.561857
9	10	2014-09-07	Falcon 9	4428.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1011	-80.577366	28.561857

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

EDA with Data Visualization

Loaded the dataset into a Pandas DataFrame.

Utilized Matplotlib and Seaborn libraries to create various plots for insightful analysis.

Plotted key relationships with Class (1st stage booster landing outcome) overlaid:

Flight Number vs. Payload Mass

Flight Number vs. Launch Site

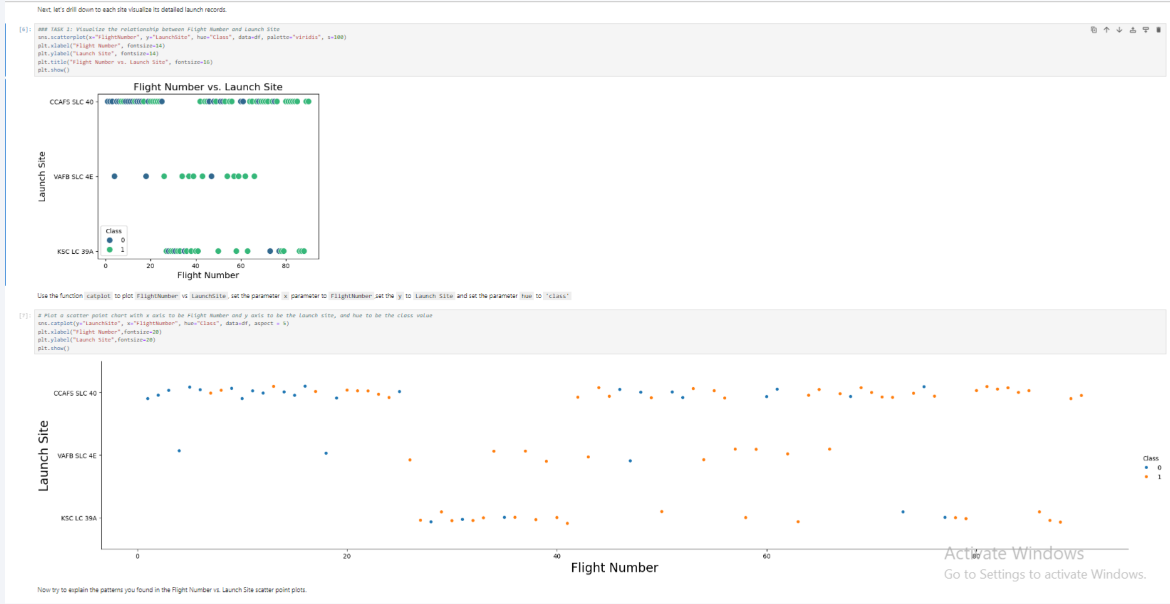
Payload vs. Launch Site

Orbit Type vs. Success Rate

Flight Number vs. Orbit Type

Payload vs. Orbit Type

Year vs. Success Rate



https://github.com/happymondaynkanta/Data-Science-Capstone/blob/main/module_2_edadataviz.ipynb

EDA with SQL

Imported the dataset into an IBM DB2 instance for advanced analysis.

Executed SQL queries to extract and display key information:

- Launch sites
- Payload masses
- Booster versions
- Mission outcomes
- Booster landings

https://github.com/happymondaynkanta/Data-Science-Capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Task 2

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

```
[16]: %sql SELECT *FROM SPACEXTABLE WHERE "Launch_Site" LIKE 'CCA%' LIMIT 5;
```

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

[16]:

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	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	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

Task 3

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

```
[17]: %sql SELECT SUM("PAYLOAD_MASS__KG_") AS Total_Payload_Mass FROM SPACEXTABLE WHERE "Customer" LIKE '%NASA (CRS)%';
```

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

[17]:

Total_Payload_Mass
48213

Task 4

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

```
[18]: %sql SELECT AVG("PAYLOAD_MASS__KG_") AS Average_Payload_Mass FROM SPACEXTABLE WHERE "Booster_Version" = 'F9 v1.1';
```

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

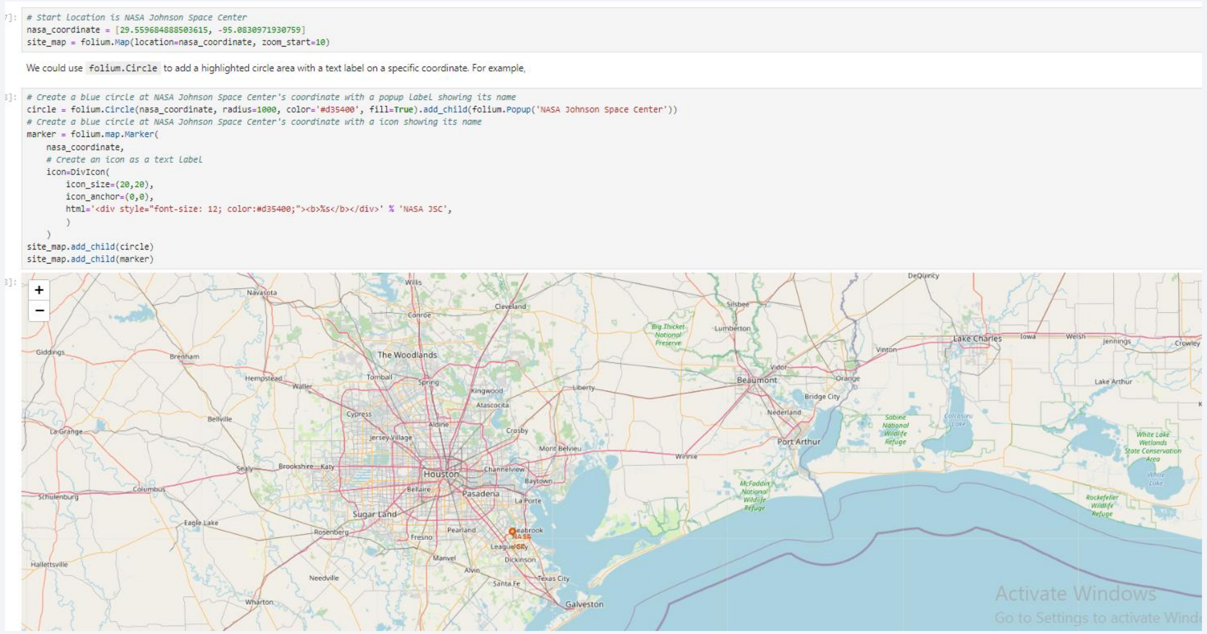
[18]:

Average_Payload_Mass
2928.4

Build an Interactive Map with Folium

Launch Sites Location Analysis:

- Utilized the interactive mapping library Folium in Python.
- Plotted all launch sites on an interactive map.
- Indicated successful and failed launches for each site.



Computed the distances from each launch site to nearby features:

- Railways
- Highways
- Coastlines
- Cities

This spatial analysis provided insights into the geographical factors influencing launch site selection and success rates.

https://github.com/happymondaynkanta/Data-Science-Capstone/blob/main/module_3_lab_jupyter_launch_site_location.ipynb

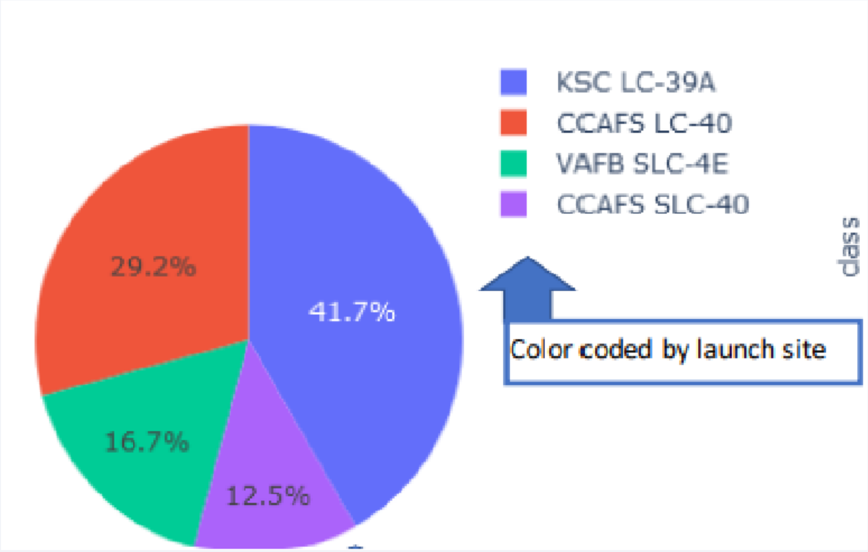
Build a Dashboard with Plotly Dash

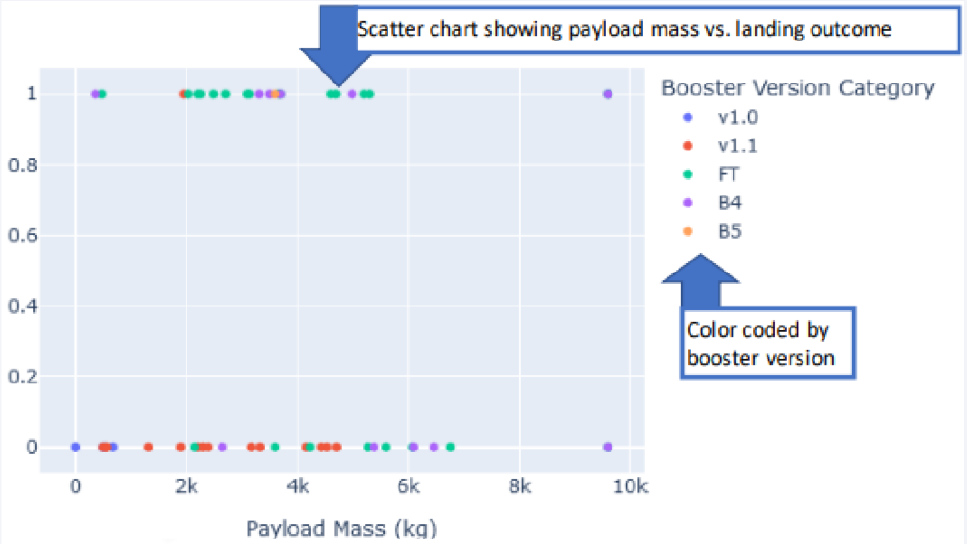
Launch Records Dashboard:

Leveraged Plotly Dash, a Python interactive dashboarding library, to provide stakeholders with real-time data exploration and manipulation capabilities.

Features included:

- A pie chart displaying the success rate, color-coded by launch site.
- A scatter chart illustrating payload mass versus landing outcome, color-coded by booster version, with an adjustable range slider to filter payload amounts.
- A dropdown menu allowing users to switch between viewing data for all launch sites or individual launch sites.
- The dashboard was deployed on IBM's static web app hosting service for accessibility.





Predictive Analysis (Classification)

- Imported essential libraries:

  Pandas, Numpy, Matplotlib, Seaborn, Sklearn

- Loaded the DataFrame created during the data collection phase.
- Added a 'Class' column, which was generated during data wrangling, as our training label.
- Standardized the dataset.
- Split the data into training and testing sets.
- Applied various machine learning models to the training data:

  Logistic Regression, Support Vector Machine, Decision Tree Classifier, K-Nearest Neighbors Classifier

- Employed cross-validated grid search (using Sklearn's GridSearchCV) to optimize hyperparameters for each model.

  Assessed the accuracy of each model on the test data to determine the most effective model.

  Defined and utilized a function to create a confusion matrix for model evaluation.

Results

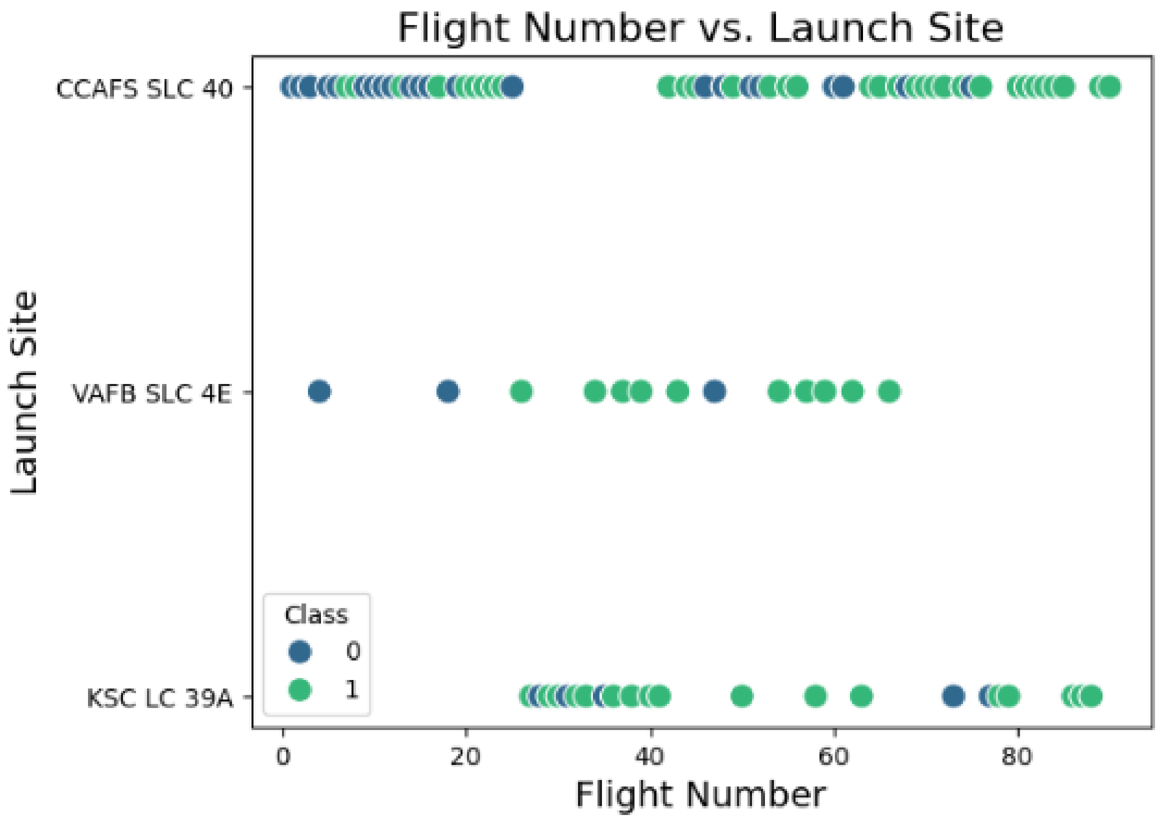
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

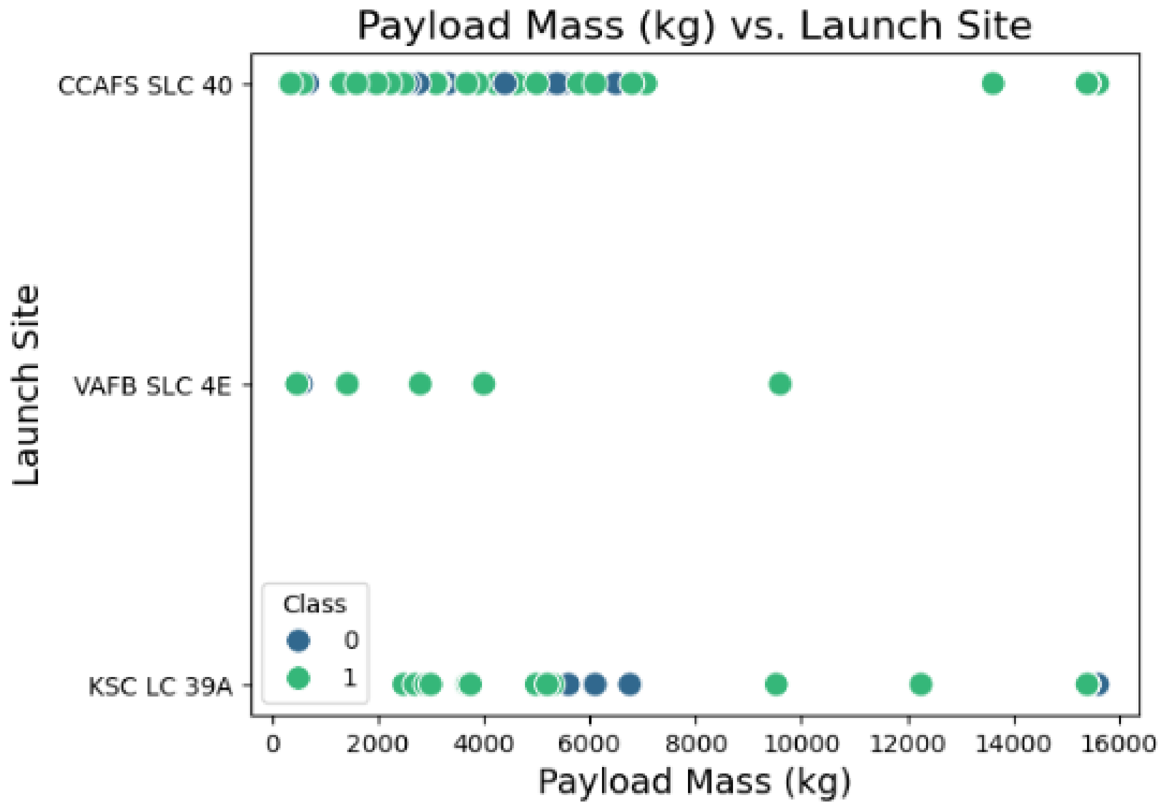
Flight Number vs. Launch Site

[6]:
```
### TASK 1: Visualize the relationship between Flight Number and Launch Site
sns.scatterplot(x="FlightNumber", y="LaunchSite", hue="Class", data=df, palette="viridis", s=100)
plt.xlabel("Flight Number", fontsize=14)
plt.ylabel("Launch Site", fontsize=14)
plt.title("Flight Number vs. Launch Site", fontsize=16)
plt.show()
```



Payload vs. Launch Site

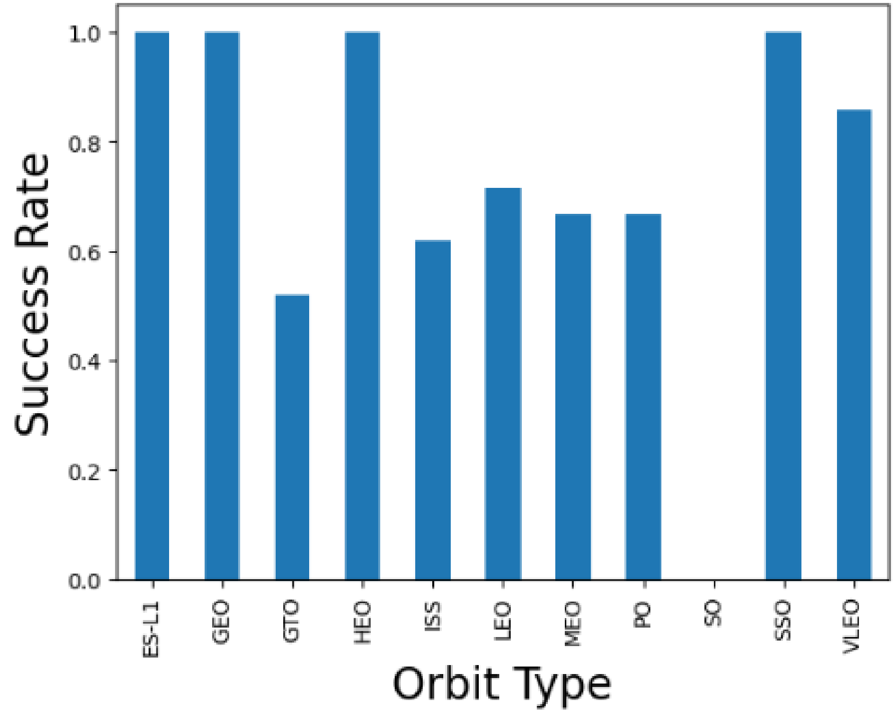
```python
# Visualizing the relationship between Payload Mass (kg) and Launch Site
sns.scatterplot(x="PayloadMass", y="LaunchSite", hue="Class", data=df, palette="viridis", s=100)
plt.xlabel("Payload Mass (kg)", fontsize=14)
plt.ylabel("Launch Site", fontsize=14)
plt.title("Payload Mass (kg) vs. Launch Site", fontsize=16)
plt.show()
```



Success Rate vs. Orbit Type

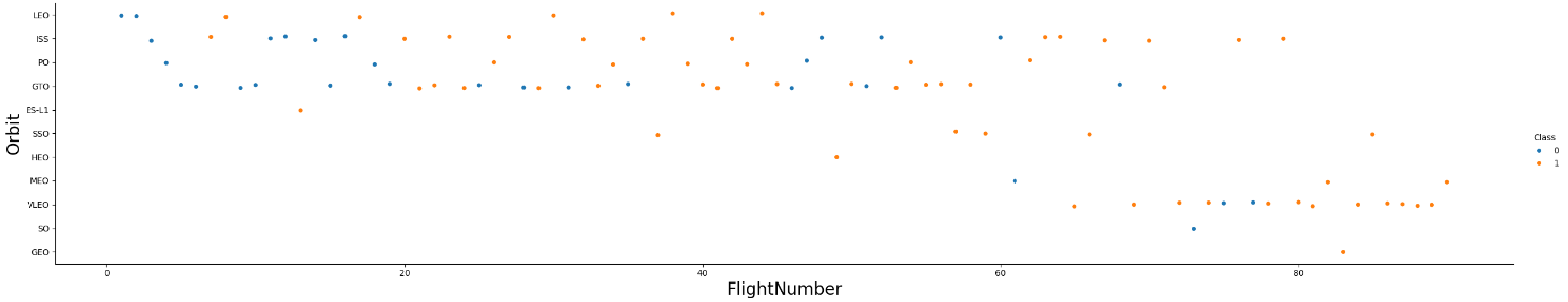
```
10]: ### TASK  3: Visualize the relationship between success rate of each orbit type
     # HINT use groupby method on Orbit column and get the mean of Class column
     df.groupby("Orbit").mean()['Class'].plot(kind='bar')
     plt.xlabel("Orbit Type",fontsize=20)
     plt.ylabel("Success Rate",fontsize=20)
     plt.show()
```

```
<ipython-input-10-2757f3c6c491>:3: FutureWarning: The default value of numeric_only in DataFrameGroup
valid for the function.
  df.groupby("Orbit").mean()['Class'].plot(kind='bar')
```



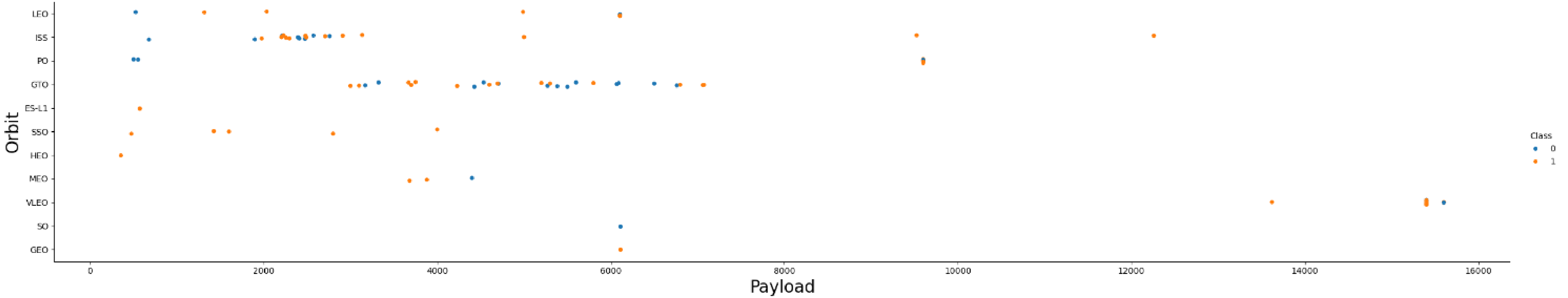
Flight Number vs. Orbit Type

```
[13]: # Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("FlightNumber",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



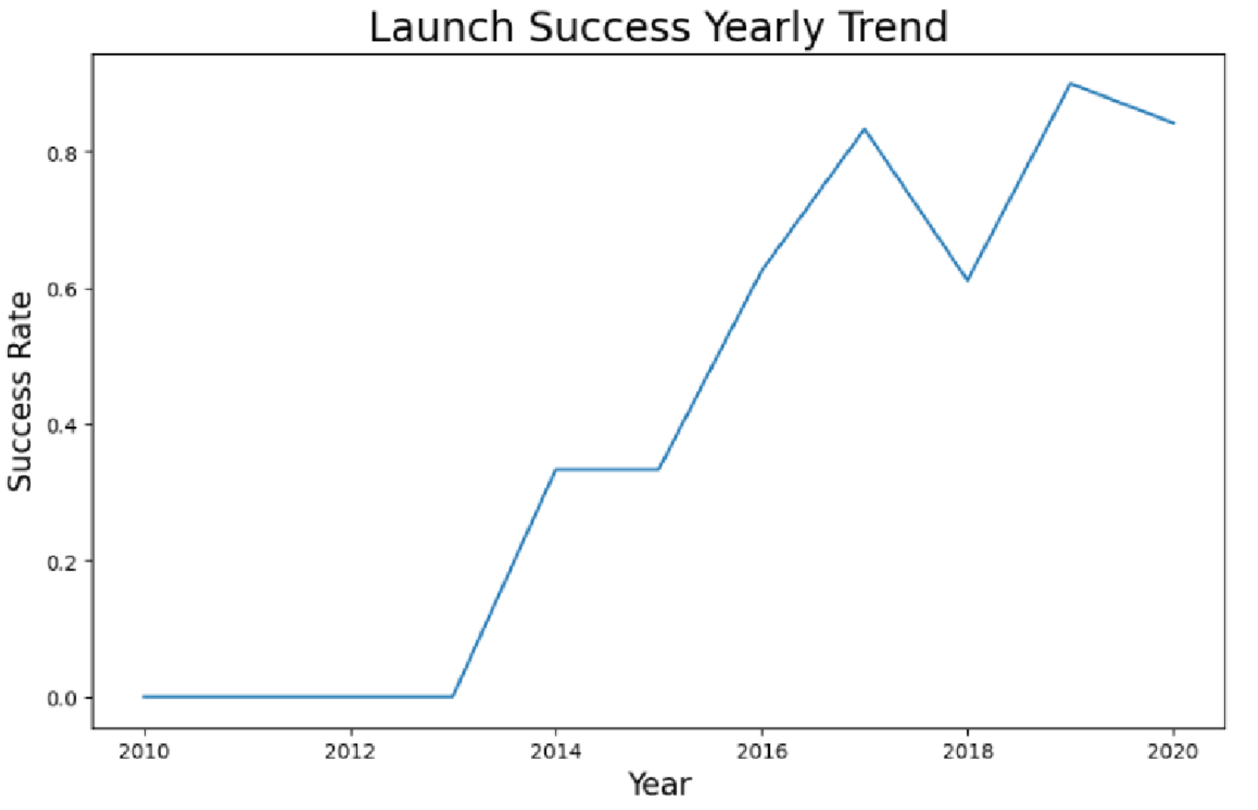
Payload vs. Orbit Type

[15]:
```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Payload",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



Launch Success Yearly Trend

```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
yearly_success = df.groupby('Year')['Class'].mean().reset_index()
plt.figure(figsize=(10, 6))
sns.lineplot(x='Year', y='Class', data=yearly_success)
plt.xlabel("Year", fontsize=15)
plt.ylabel("Success Rate", fontsize=15)
plt.title("Launch Success Yearly Trend", fontsize=20)
plt.show()
```



All Launch Site Names

- The names of the unique launch sites are:
- CCAFS LC-40
- VAFB SLC-4E
- KSC LC-39A
- CCAFS SLC-40

Task 1

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

```
%sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE;
```

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

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

Launch Site Names Begin with 'CCA'

- launch sites begin with `CCA`

CCAFS LC-40

CCAFS LC-40

CCAFS LC-40

CCAFS LC-40

CCAFS LC-40

Task 2

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

```
%sql SELECT *FROM SPACEXTABLE WHERE "Launch_Site" LIKE 'CCA%' LIMIT 5;
```

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

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	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	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

Total Payload Mass

- Total payload carried by boosters from NASA is 48213

Task 3

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

```
%sql SELECT SUM("PAYLOAD_MASS__KG_") AS Total_Payload_Mass FROM SPACEXTABLE WHERE "Customer" LIKE '%NASA (CRS)%';
```

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

Total_Payload_Mass
48213

Average Payload Mass by F9 v1.1

- The average payload mass carried by booster version F9 v1.1 is 2928.4

Task 4

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

```
%sql SELECT AVG("PAYLOAD_MASS__KG_") AS Average_Payload_Mass FROM SPACEXTABLE WHERE "Booster_Version" = 'F9 v1.1';
```

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

Average_Payload_Mass
2928.4

First Successful Ground Landing Date

- The dates of the first successful landing outcome on ground pad is 2015-12-22

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

Hint:Use min function

```
%sql SELECT MIN(Date) AS First_Successful_Landing FROM SPACEXTABLE WHERE Landing_Outcome='Success (ground pad)';
```

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

First_Successful_Landing
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- List the names of boosters are F9 FT B1022, F9 FT B1026, F9 FT B1021.2, F9 FT B1032.2

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

```
%sql SELECT "Booster_Version" FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Success (drone ship)' AND "PAYLOAD_MASS__KG_" > 40
```

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

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 total number of successful mission outcomes is 100 and failure mission outcomes is 1

Task 7

List the total number of successful and failure mission outcomes

```
%sql SELECT "Mission_Outcome", COUNT(*) AS Count FROM SPACEXTABLE GROUP BY "Mission_Outcome";
```

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

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

Boosters Carried Maximum Payload

The names of the booster which have carried the maximum payload mass are: 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

Task 8

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

```
%sql SELECT "Booster_Version" FROM SPACEXTABLE WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTABLE
```

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

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

- Failure (drone ship) F9 v1.1 B1012 CCAFS LC-40
- Failure (drone ship) F9 v1.1 B1015 CCAFS LC-40

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.

```
%sql SELECT strftime('%m', "Date") AS Month, "Landing_Outcome", "Booster_Version", "Launch_Site" FROM SPACEXTABLE WHERE "Lar
```

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

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

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

- 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

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

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.

25]:

```
%sql SELECT "Landing_Outcome", COUNT(*) AS Count FROM SPACEXTABLE WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20' GROUP E
```

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

25]:

Landing_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

<Folium Map Screenshot 1>

```
folium.map.Marker(coordinate, icon=DivIcon(icon_size=(20,20),icon_anchor=(0,0), html='<div style="font-size: 12;
color:#d35400;"><b>%s</b></div>' % 'label', ))
```

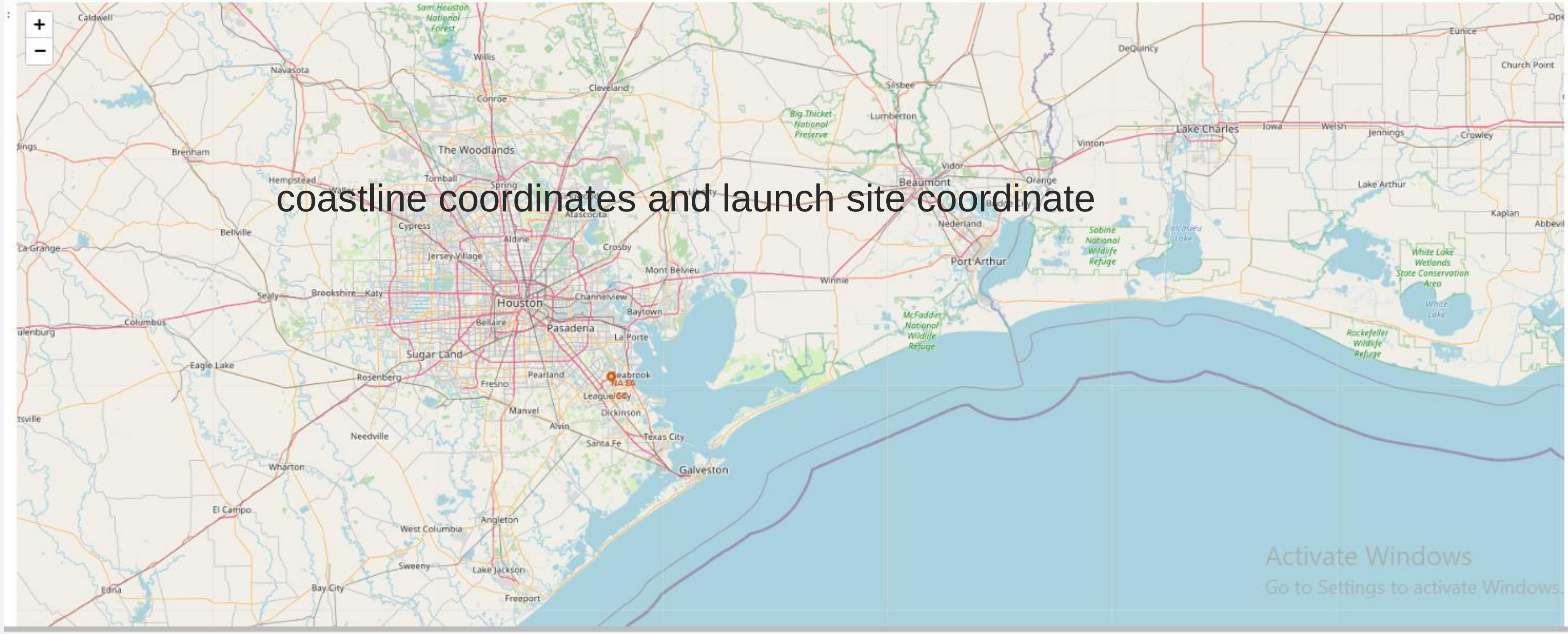
```
# Initial the map
site_map = folium.Map(location=nasa_coordinate, zoom_start=5)
# For each launch site, add a Circle object based on its coordinate (Lat, Long) values. In addition, add Launch site name as
```

The generated map with marked launch sites should look similar to the following:

Launch sites visualization in proximity to coast and equator

<Folium Map Screenshot 2>

```
# Create a blue circle at NASA Johnson Space Center's coordinate with a popup label showing its name
circle = folium.Circle(nasa_coordinate, radius=1000, color='#d35400', fill=True).add_child(folium.Popup('NASA Johnson Space Center'))
# Create a blue circle at NASA Johnson Space Center's coordinate with a icon showing its name
marker = folium.map.Marker(
    nasa_coordinate,
    # Create an icon as a text label
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % 'NASA JSC',
        )
    )
site_map.add_child(circle)
site_map.add_child(marker)
```



coastline coordinates and launch site coordinate

<Folium Map Screenshot 3>

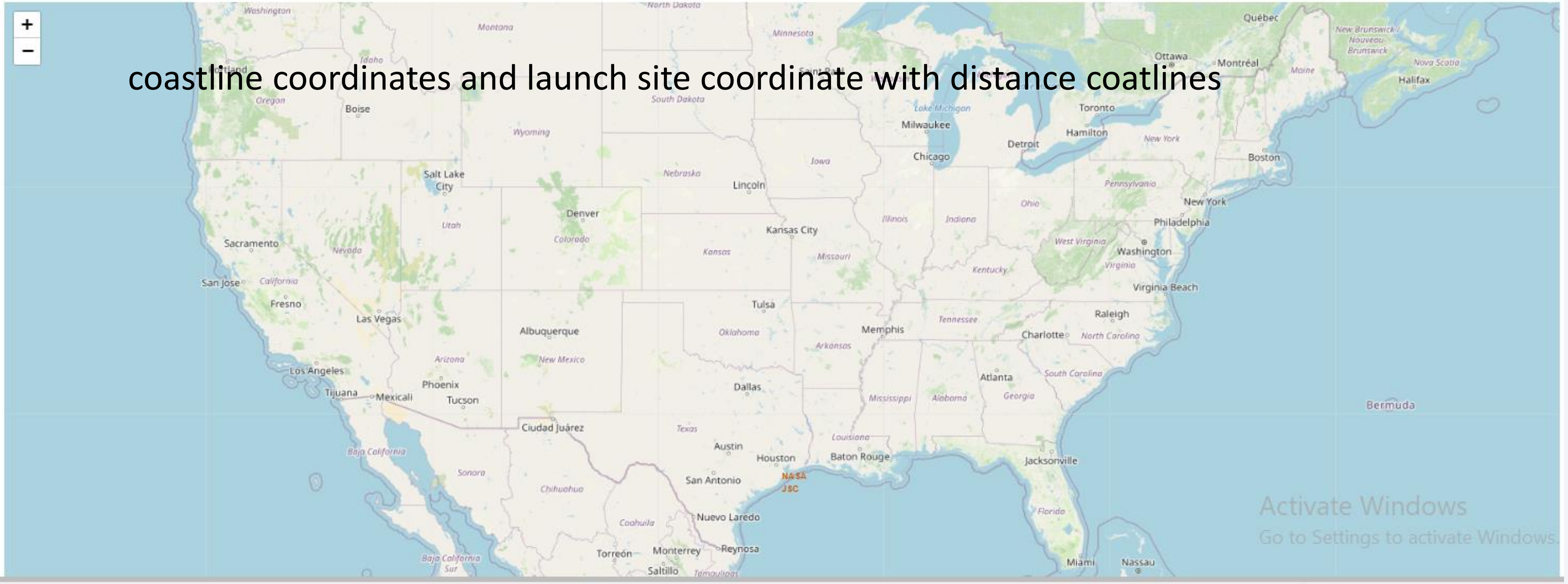
TODO: For each launch result in spacex_df data frame, add a folium.Marker to marker_cluster

```
[14]: # Add marker_cluster to current site_map
site_map.add_child(marker_cluster)

# for each row in spacex_df data frame
# create a Marker object with its coordinate
# and customize the Marker's icon property to indicate if this launch was successed or failed,
# e.g., icon=folium.Icon(color='white', icon_color=row['marker_color']
for index, record in spacex_df.iterrows():
    # TODO: Create and add a Marker cluster to the site map
    # marker = folium.Marker(...)
    marker_cluster.add_child(marker)

site_map
```

[14]:

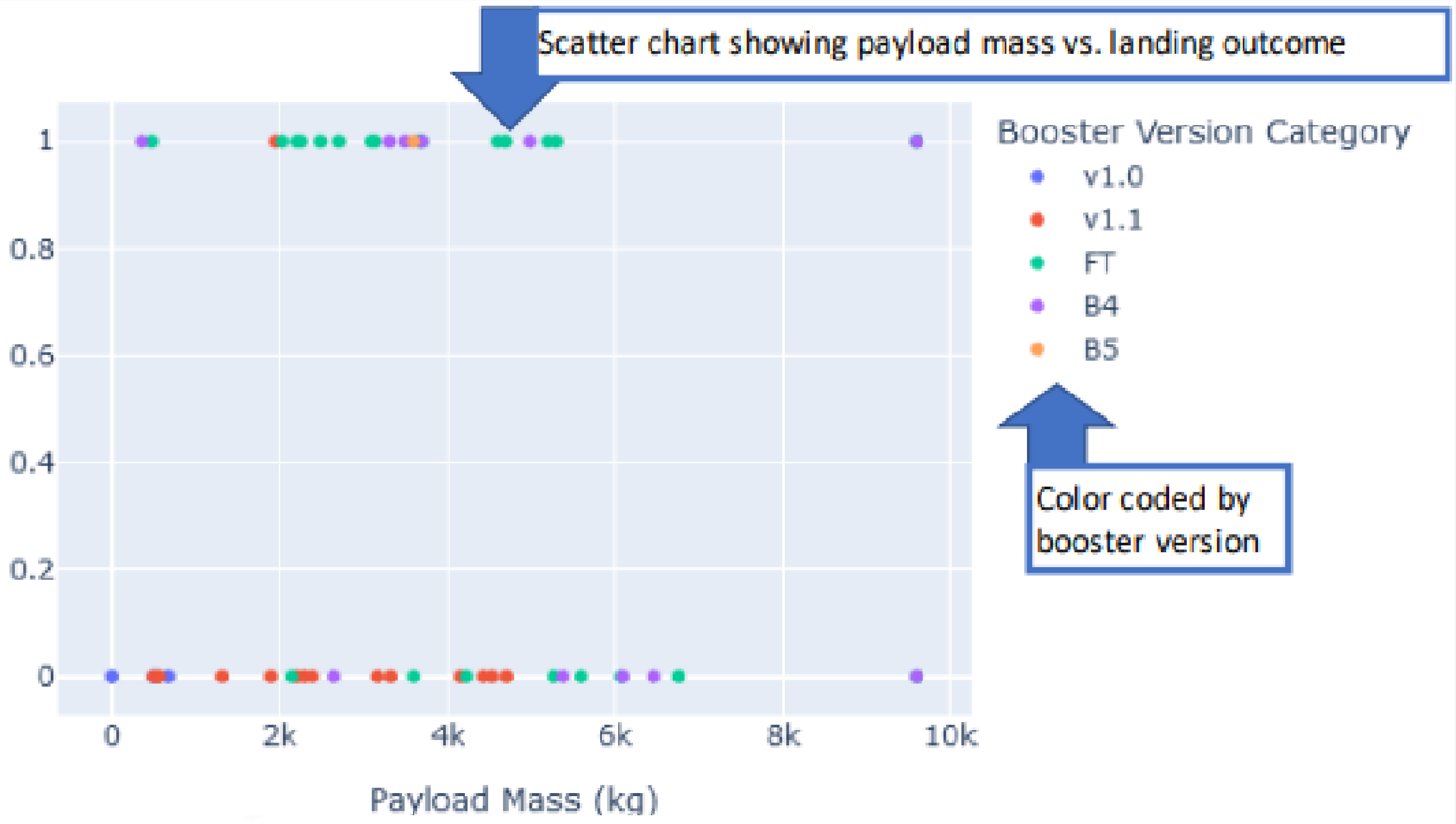


coastline coordinates and launch site coordinate with distance coatlines

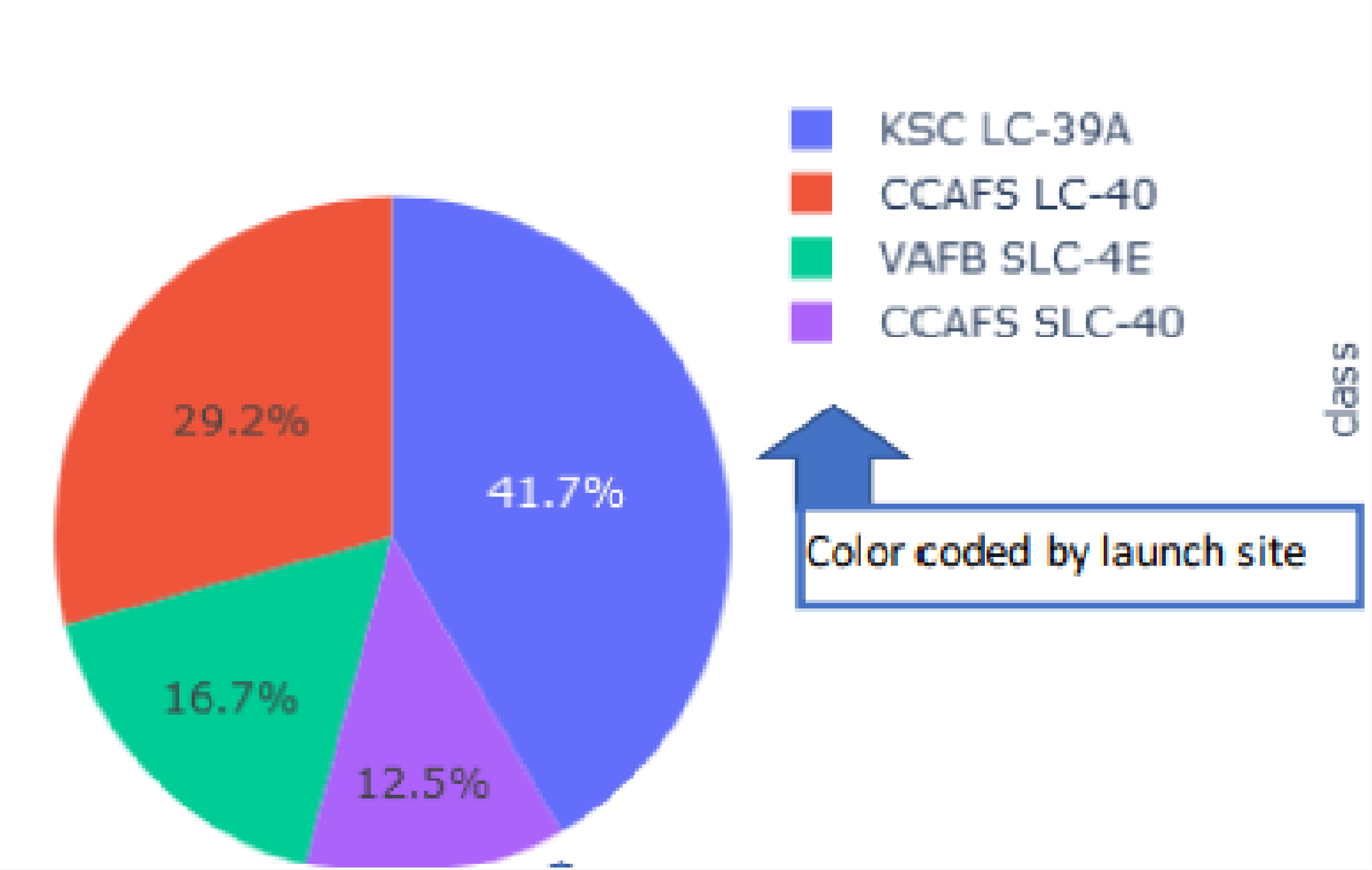
Section 4

Build a Dashboard with Plotly Dash

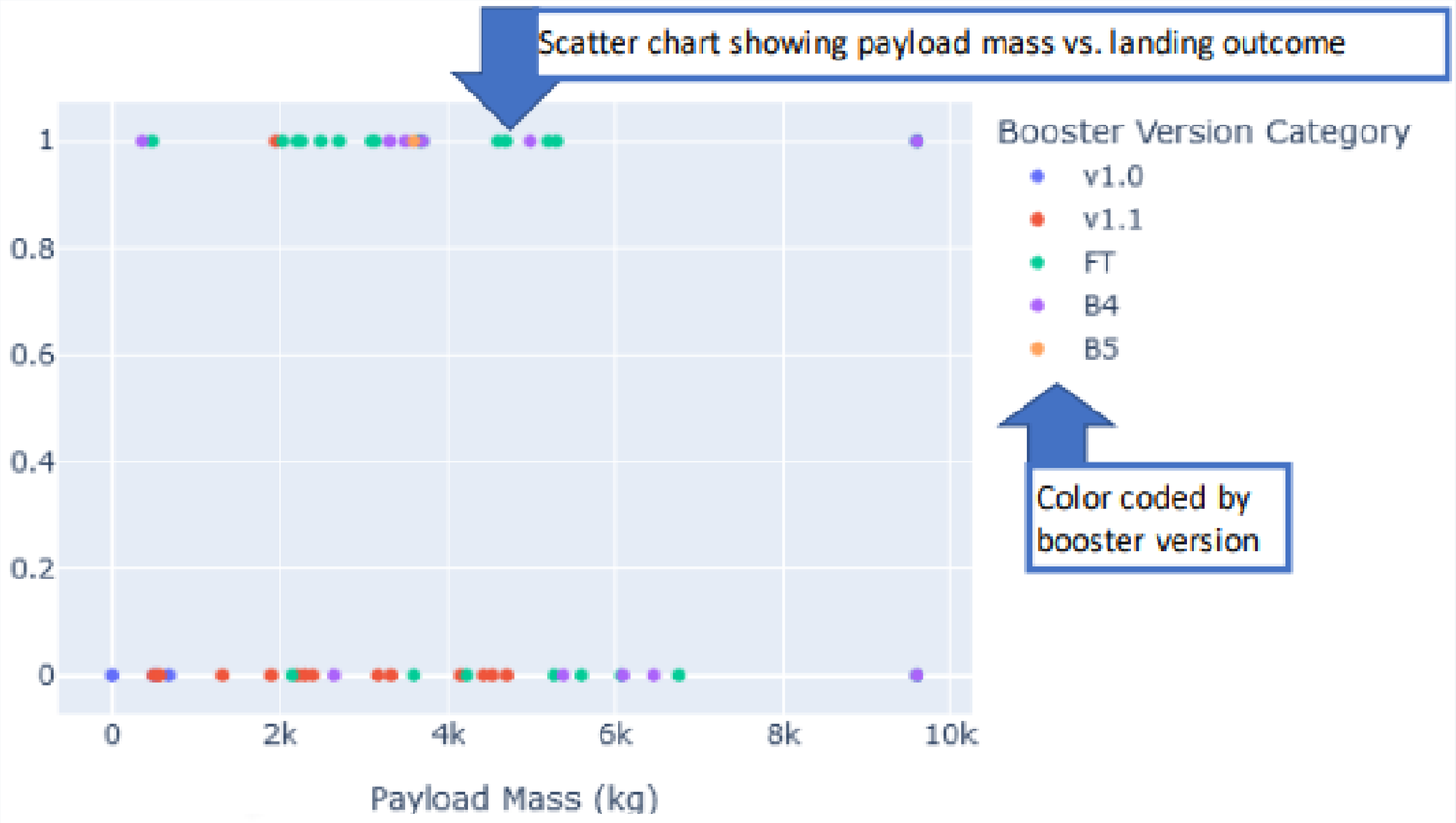
<Dashboard Screenshot 1>



<Dashboard Screenshot 2>



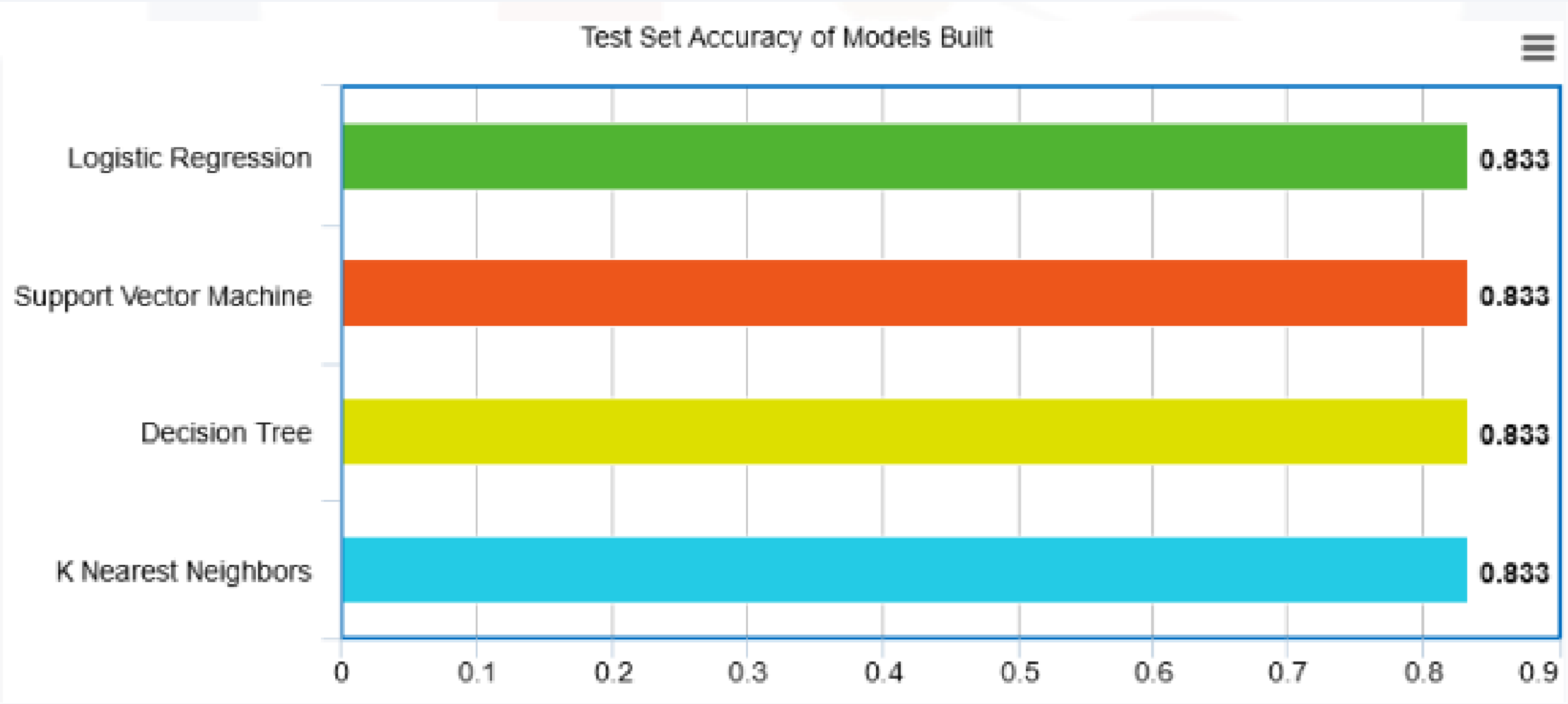
<Dashboard Screenshot 3>



Section 5

Predictive Analysis (Classification)

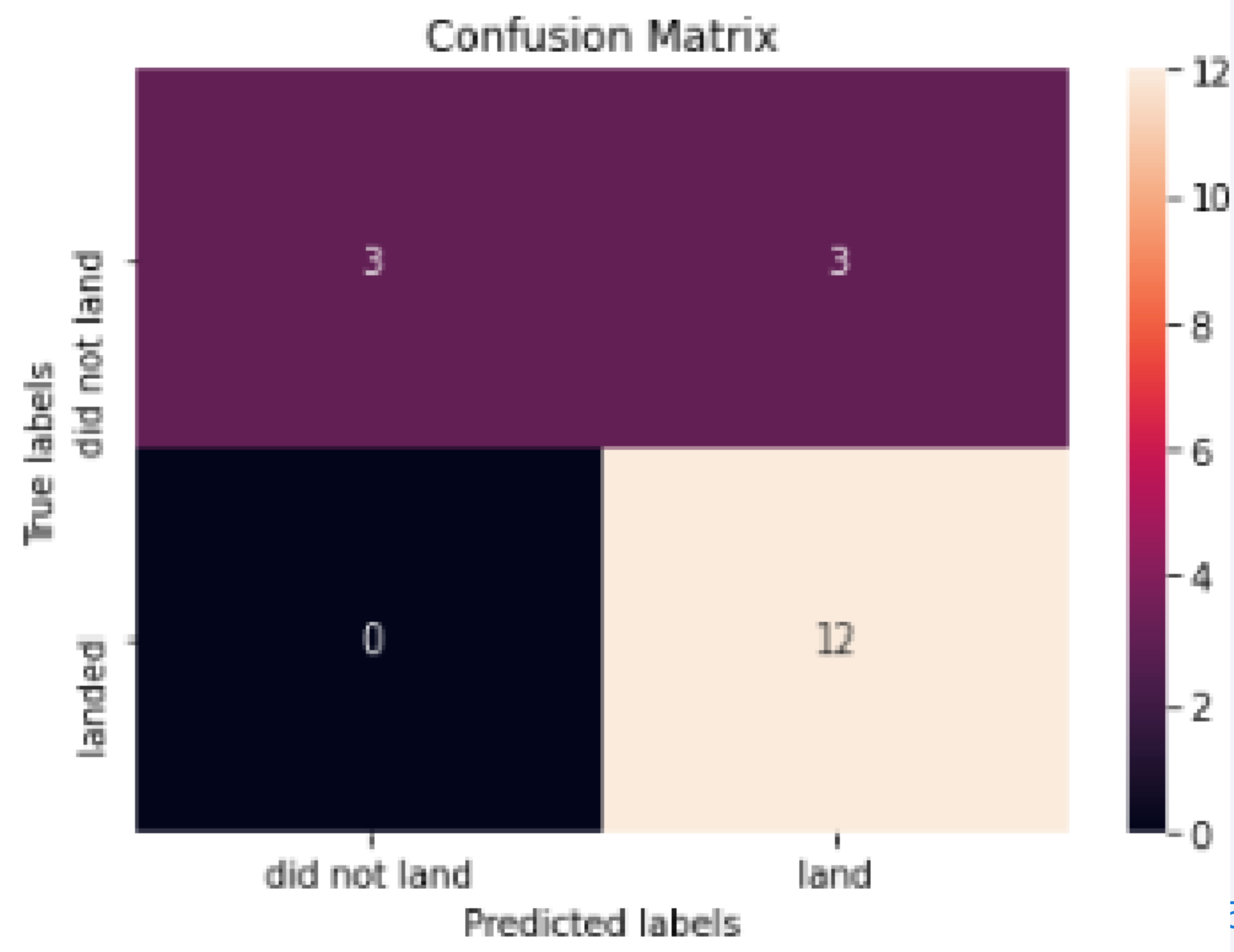
Classification Accuracy



All model has the same highest classification accuracy of 83.33%

Confusion Matrix

The confusion matrix for the best performing model is identical across all four models (logistic regression, KNN, SVM, and decision tree). It indicates 15 true classifications and 3 misclassifications along the diagonal axis.



Conclusions

With the models developed in this report, SpaceY can predict SpaceX's success in landing the first stage booster with 83.3% accuracy. SpaceX has publicly stated that constructing the first stage booster costs over $15 million. This insight allows SpaceY to make more strategic bids, as they can estimate when SpaceX's bids might include the cost of a lost first stage booster. Given SpaceX's launch price of $62 million, the inclusion of a $15+ million booster sacrifice would raise their bid to approximately $77 million.

Future Opportunities for SpaceY:

- Optimize Model Performance: Freeze the optimal combination of model and hyperparameters and retrain using the entire dataset rather than just the training subset. This approach could potentially improve the model's accuracy, but it would eliminate the ability to measure its predictive accuracy.
- Expand Dataset: Continuously update the model with additional launch data as it becomes available to enhance predictive capabilities.

Model Refinement:

- Subdivide Current Model: Develop two separate models: one to predict if SpaceX will attempt to land the first stage, and another to predict the success of such attempts.
- Predict Reuse: Create a model to forecast whether SpaceX will use a previously flown first stage booster in their launches. This would help SpaceY anticipate when SpaceX might offer a discounted bid.

- These steps will enable SpaceY to gain a competitive edge by anticipating the components of SpaceX's bids more accurately.

Appendix

- Notebooks:
  - https://github.com/happymondaynkanta/Data-Science-Capstone/blob/main/lab_jupyter_launch_site_location.ipynb
  - https://github.com/happymondaynkanta/Data-Science-Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb
- Acknowledgments
  - Thanks to Joseph Santarcangelo at IBM for the creating the materials
- References:
  - https://www.coursera.org/learn/applied-data-science-capstone/ungradedLti/EUhln/hands-on-lab-complete-the-machine-learning-prediction-lab

Thank you!