IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Adem SONUVAR
Aug 15, 2023

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion

Executive Summary

- In the rapidly evolving domain of commercial space travel, SpaceX stands as a beacon of innovation, particularly with its Falcon 9 rockets. The core objective of our study is to anticipate SpaceX's decisions regarding the reuse of the Falcon 9's first stage. This initiative, unique to SpaceX, presents significant financial, competitive, and environmental implications. A successful prediction of such decisions can profoundly influence budgetary planning, offering insights into the competitive dynamics of the space industry and promoting sustainable practices by reducing space debris. Through our analysis, we aim to provide a multifaceted perspective on space exploration, encompassing its economic, environmental, and competitive dimensions.

Introduction

- **Project Background and Context** In this project, we aimed to predict the successful landing of Falcon 9's first stage. SpaceX promotes its Falcon 9 rocket launches on its website at a cost of $62 million. In contrast, other providers charge upwards of $165 million. A significant portion of these savings for SpaceX comes from its ability to reuse the first stage of the rocket. By predicting the success of the first stage landing, we can estimate the cost of a launch. This data is invaluable for any competitors aiming to bid against SpaceX for rocket launch contracts.
- **Key Challenges Addressed:**
- Determining factors influencing the successful landing of the rocket.
- Understanding how specific rocket variables impact the landing success rate.
- Identifying the conditions SpaceX must meet to optimize their chances of a successful rocket landing.

Section 1

Methodology

Methodology

- **Data Collection Methodology:**
- Utilized the **SpaceX Rest API** for primary data extraction.
- Employed **Web Scraping** techniques to gather additional information from Wikipedia.
- **Data Wrangling:**
- Transformed data to make it suitable for Machine Learning applications.
- Applied **One Hot Encoding** to relevant data fields.
- Dropped columns that were deemed irrelevant to the analysis.
- **Exploratory Data Analysis (EDA):**
- Used both visualization tools and SQL for in-depth data analysis.
- Created visual representations such as **Scatter Graphs** and **Bar Graphs** to elucidate relationships and patterns within the data.
- **Interactive Visual Analytics:**
- Employed tools like **Folium** and **Plotly Dash** to create dynamic, interactive visual representations of the data.
- **Predictive Analysis:**
- Utilized various classification models for predictive tasks.
- Detailed the process on how to build, optimize (tune), and evaluate these classification models for best performance

Data Collection – SpaceX API

In [22]:

```
from js import fetch
import io

URL = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_
resp = await fetch(URL)
dataset_part_1_csv = io.BytesIO((await resp.arrayBuffer()).to_py())
```

Load Space X dataset, from last section.

In [23]:

```
df=pd.read_csv(dataset_part_1_csv)
df.head(10)
```

Out[23]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0
5	6	2014-01-06	Falcon 9	3325.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0
6	7	2014-04-18	Falcon 9	2296.000000	ISS	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0
7	8	2014-07-14	Falcon 9	1316.000000	LEO	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0
8	9	2014-08-05	Falcon 9	4535.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0
9	10	2014-09-07	Falcon 9	4428.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0

[Github URL to Notebook]

Data Collection - Scraping

1

```
In [3]:
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_
```

executed in 13ms, finished 23:12:03 2023-08-20

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

1.2.1 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 H

```
In [4]:
# use requests.get() method with the provided static_url
# assign the response to a object
page = requests.get(static_url)
page.status_code
```

executed in 553ms, finished 23:12:04 2023-08-20

```
Out[4]: 200
```

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

```
In [5]:
# Use BeautifulSoup() to create a BeautifulSoup object from a res
soup = BeautifulSoup(page.text, 'html.parser')
```

executed in 781ms, finished 23:12:04 2023-08-20

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

```
In [6]:
# Use soup.title attribute
soup.title
```

executed in 14ms, finished 23:12:04 2023-08-20

```
Out[6]: <title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

2

end of this lab

```
In [7]:
# Use the find_all function in the BeautifulSoup object, with element type `table
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

executed in 19ms, finished 23:12:04 2023-08-20

Starting from the third table is our target table contains the actual launch records.

```
In [8]:
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

executed in 15ms, finished 23:12:04 2023-08-20

```
>landing</a>
</th></tr>
<tr>
<th rowspan="2" scope="row" style="text-align:center;">1
</th>
<td>4 June 2010,<br/>18:45
</td>
<td><a href="/wiki/Falcon_9_v1.0" title="Falcon 9 v1.0">F9 v1.0</a><sup class="referenc
<a href="#cite_note-MuskMay2012-13">[7]</a></sup><br/>B0003.1<sup class="reference" id=
ref="#cite_note-block_numbers-14">[8]</a></sup>
</td>
<td><a href="/wiki/Cape_Canaveral_Space_Force_Station" title="Cape Canaveral Space Forc
="/wiki/Cape_Canaveral_Space_Launch_Complex_40" title="Cape Canaveral Space Launch Comp
</td>
<td><a href="/wiki/Dragon_Spacecraft_Qualification_Unit" title="Dragon Spacecraft Quali
ualification Unit</a>
</td>
<td>
</td>
```

3

```
column_names = []
temp = soup.find_all('th')
for x in range(len(temp)):
    try:
     name = extract_column_from_header(temp[x])
     if (name is not None and len(name) > 0):
        column_names.append(name)
    except:
     pass
```

executed in 30ms, finished 23:12:04 2023-08-20

heck the extracted column names

```
print(column_names)
```

executed in 11ms, finished 23:12:04 2023-08-20

```
['Flight No.', 'Date and time ( )', 'Launch site', 'Payload',
t No.', 'Date and time ( )', 'Launch site', 'Payload', 'Paylo
'Date and time ( )', 'Launch site', 'Payload', 'Payload mass
nd time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orb
nd time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orb
( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Cu
'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer
me ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit',
e', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch
t', 'Customer', 'Date and time ( )', 'Launch site', 'Payload
yload', 'Orbit', 'Customer', 'Date and time ( )', 'Launch si
s', 'Crewed missions', 'Commercial satellites', 'Scientific
missions', 'Current', 'In development', 'Retired', 'Cancelle
t', 'Retired', 'Unflown', 'Orbital', 'Atmospheric', 'Landing
ams', 'Key people', 'Related', 'General', 'General', 'People
t', 'Agencies, companies and facilities', 'Other mission list
```

.3 TASK 3: Create a data frame by parsing the

e will create an empty dictionary with keys from the extracted column names i
ataframe

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']


launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

executed in 15ms, finished 23:12:04 2023-08-20

4

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

executed in 0ms, finished 23:12:05 2023-08-20

[Github URL to Notebook]()

Data Wrangling

Perform Exploratory Data Analysis (EDA) on the dataset

Calculate the number of launches at each site

Calculate the number and occurrence of mission outcome per orbit type

Export dataset as .CSV

Calculate the number and occurrence of each orbit

Create a landing outcome label from the "Outcome" column

Work out the success rate for every landing in the dataset

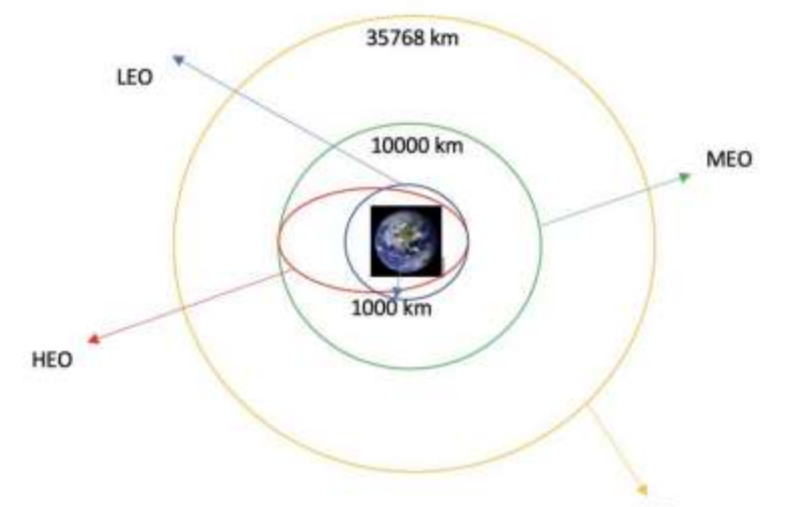


Diagram showing common orbit types SpaceX uses

[Github URL to Notebook](#)

EDA with Data Visualization

- **Flight Number vs Launch Site**
- **Payload vs Launch Site**
- **Success rate of each orbit type**
- **FlightNumber vs Orbit type**
- **Payload vs Orbit type**
- **Launch success yearly trend**

[Github URL to Notebook]

EDA with SQL

- Using bullet point format, summarize the SQL queries you performed
- Add the GitHub URL of your completed EDA with SQL notebook, as an external reference and peer-review purpose
- Unique launch sites in the space mission
- 5 records where launch sites begin with the string 'CCA'
- Total payload mass carried by boosters launched by NASA (CRS)
- Average payload mass carried by booster version F9 v1.1
- Date when the first successful landing outcome in ground pad was achieved.
- Names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- Total number of successful and failure mission outcomes
- Names of the booster versions which have carried the maximum payload mass.
- Records which will display the month names, failure landing outcomes in drone ship ,booster versions, launch site for the months in year 2015.
- 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

GitHub URL of completed interactive map with Folium map, as an external reference and peer-review purpose

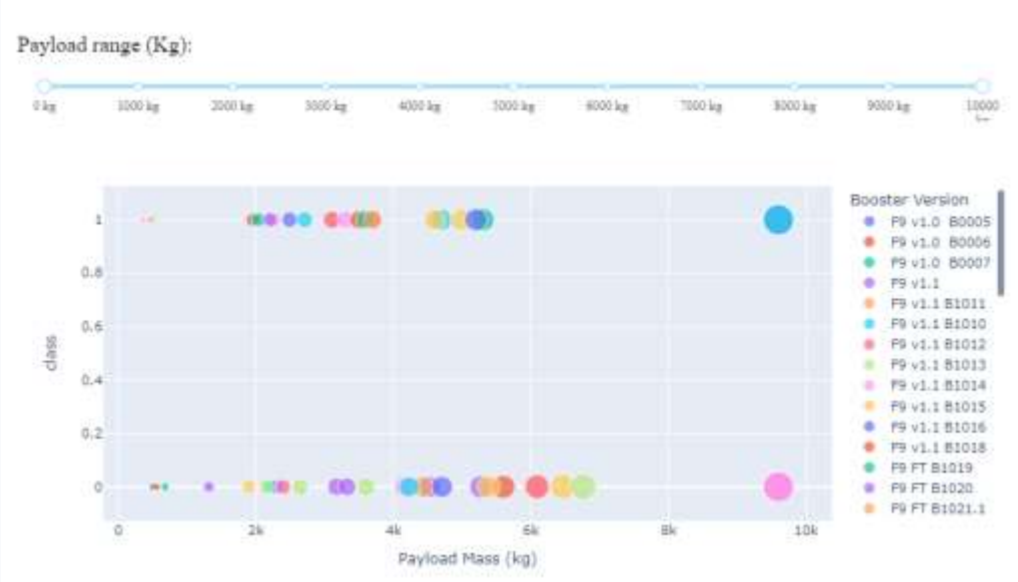
Build an Interactive Map with Folium

- Green Marker shows successful Launches
- Red Marker shows Failures
- Circles shows launch sites and scape centers.
- Lines shows distance between a launch site to its closest city, railway, highway, etc.

GitHub URL of completed interactive map with Folium map, as an external reference and peer-review purpose

Build a Dashboard with Plotly Dash

- Pie chart for the launch site with highest launch success ratio
- Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider



GitHub URL of completed interactive map with Folium map, as an external reference and peer-review purpose

Predictive Analysis (Classification)

1.9 TASK 5

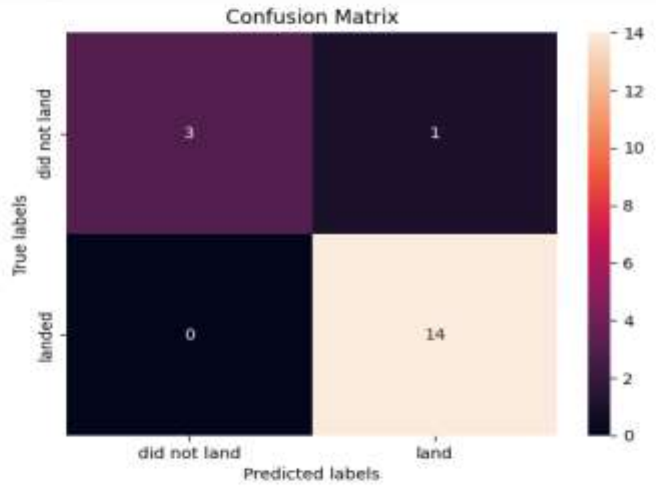
Calculate the accuracy on the test data using the method `score` :

n [47]:
```
accuracy = logreg_cv.score(X_test, Y_test)
print(f"Accuracy on the test data: {accuracy:.2f}")
```

```
Accuracy on the test data: 0.94
```

Lets look at the confusion matrix:

n [48]:
```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



1.15 TASK 11

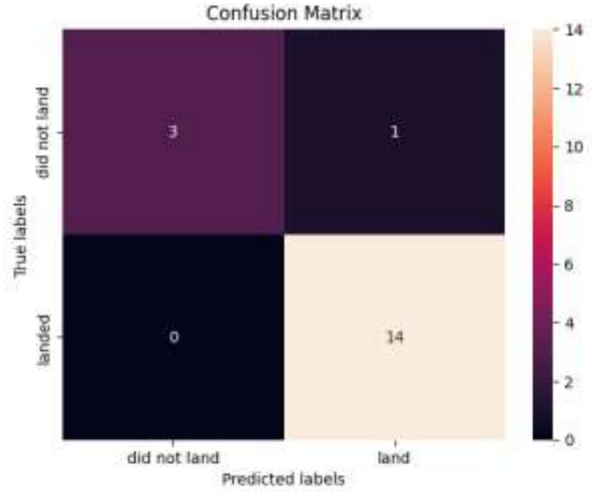
Calculate the accuracy of knn_cv on the test data using the method `score` :

In [26]:
```
accuracy = knn_cv.score(X_test, Y_test)
print(f"Accuracy on the test data: {accuracy:.2f}")
```

```
Accuracy on the test data: 0.94
```

We can plot the confusion matrix

In [27]:
```
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



1.16 TASK 12

Find the method performs best:

In [ ]:
```
logreg_cv and knn_cv
```


[GitHub URL of completed interactive map with Folium map, as an external reference and peer-review purpose]()

Results

Exploratory Data Analysis (EDA):

•Successful Falcon 9 first stage landings were visualized.
•Some unsuccessful landings are planned and controlled.

Feature Engineering:

•Dataset has 12 features and 90 samples.
•OneHotEncoding was applied to convert categorical data to numerical.
•Data types for numerical columns were standardized.

Overview of the DataSet

•SpaceX's capability to return spacecraft from low-earth orbit is highlighted.
•Reusability of Falcon 9 first stage offers cost advantages.
•Determining the success of first stage landing can help estimate launch cost.

Section 2

Insights drawn from EDA

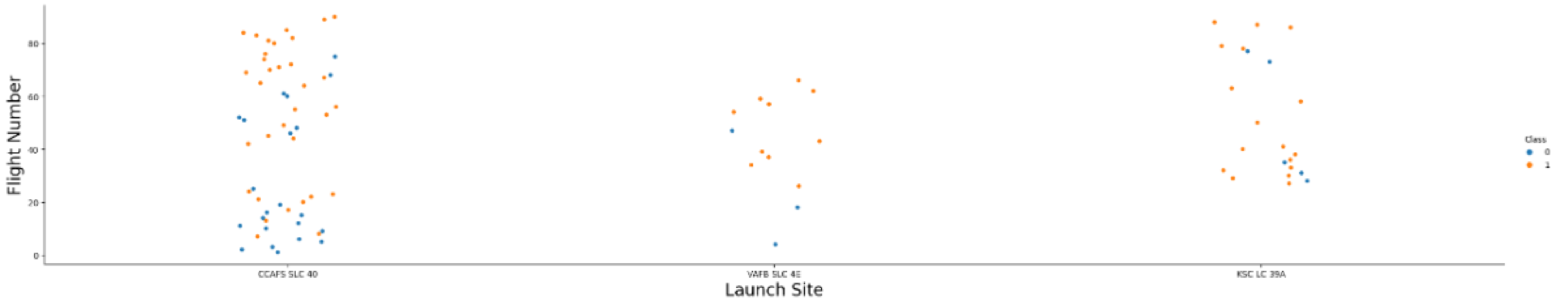
Flight Number vs Launch Site

In [5]:

```python
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class

sns.catplot(x="LaunchSite", y="FlightNumber", hue="Class", data=df, aspect = 5)
plt.ylabel("Flight Number",fontsize=20)
plt.xlabel("Launch Site",fontsize=20)
plt.show()
```

executed in 297ms, finished 21:52:34 2023-08-27

```
C:\Users\adems\anaconda3\envs\notebook-6.4.5\lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has c
hanged to tight
  self._figure.tight_layout(*args, **kwargs)
```

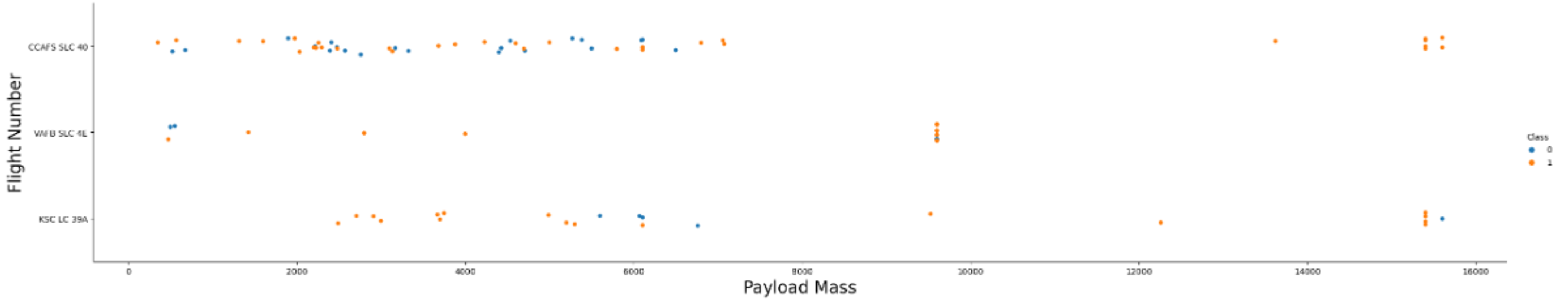


Payload Mass vs Launch Site

```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the c

sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.ylabel("Flight Number",fontsize=20)
plt.xlabel("Payload Mass",fontsize=20)
plt.show()
```

executed in 247ms, finished 21:55:23 2023-08-27

```
C:\Users\adems\anaconda3\envs\notebook-6.4.5\lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has c
hanged to tight
  self._figure.tight_layout(*args, **kwargs)
```



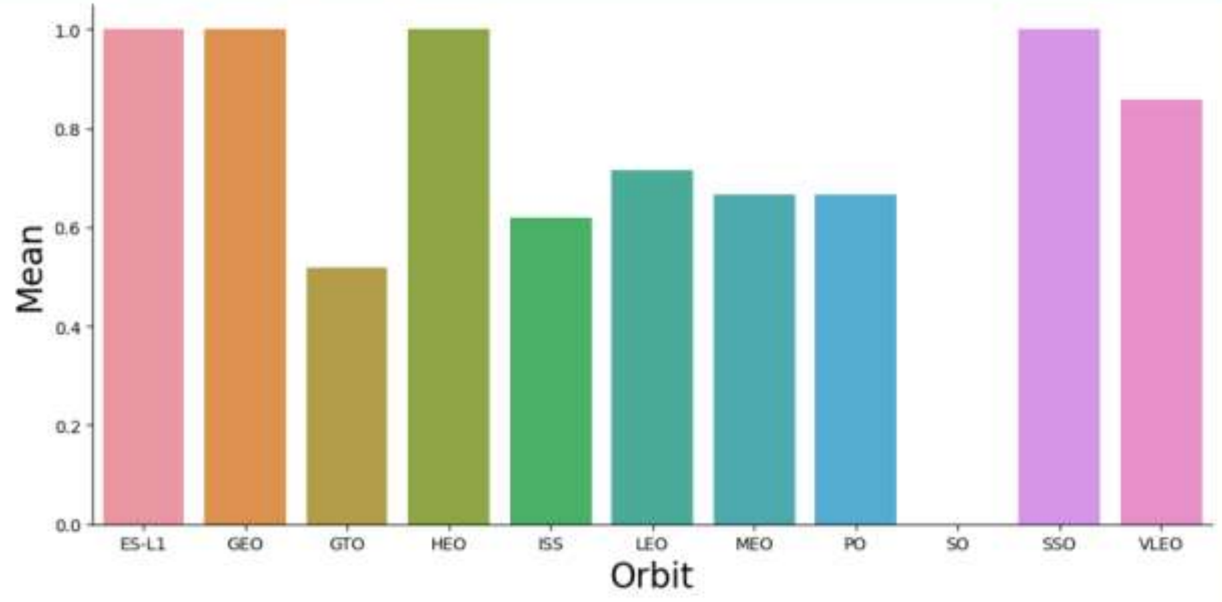
Success Rate vs Orbit Type

[15]:

```python
# HINT use groupby method on Orbit column and get the mean of Class column
# Group by 'Orbit' and compute the mean of only the 'Class' column

orbit_success_rate = df.groupby('Orbit')['Class'].mean().reset_index()

sns.catplot(x="Orbit", y="Class", kind="bar", data=orbit_success_rate, aspect=2)
plt.xlabel("Orbit", fontsize=20)
plt.ylabel("Mean", fontsize=20)
plt.show()
```

executed in 134ms, finished 22:06:19 2023-08-27

```
C:\Users\adems\anaconda3\envs\notebook-6.4.5\lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has c
hanged to tight
  self._figure.tight_layout(*args, **kwargs)
```



Flight Number vs Orbit Type

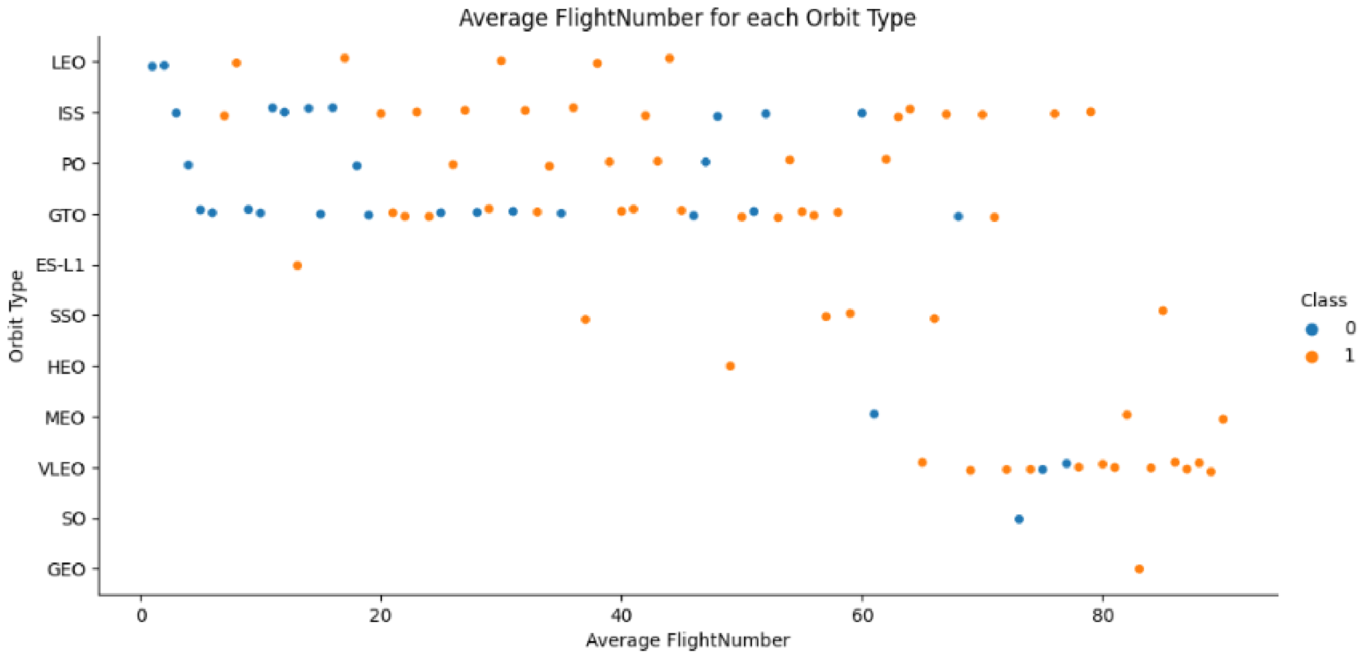
In [12]:
```
grouped_data = df.groupby('Orbit')['Class'].value_counts().unstack().fillna(0)
```

In [18]:
```
# 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
plt.figure(figsize=(10, 6))
sns.catplot(y='Orbit', x='FlightNumber', data=df, hue="Class", aspect=2)

plt.title('Average FlightNumber for each Orbit Type')
plt.xlabel('Average FlightNumber')
plt.ylabel('Orbit Type')
plt.show();
```

executed in 271ms, finished 22:08:33 2023-08-27

```
C:\Users\adems\anaconda3\envs\notebook-6.4.5\lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has c
hanged to tight
  self._figure.tight_layout(*args, **kwargs)
```

```
<Figure size 1000x600 with 0 Axes>
```



Payload vs Orbit Type

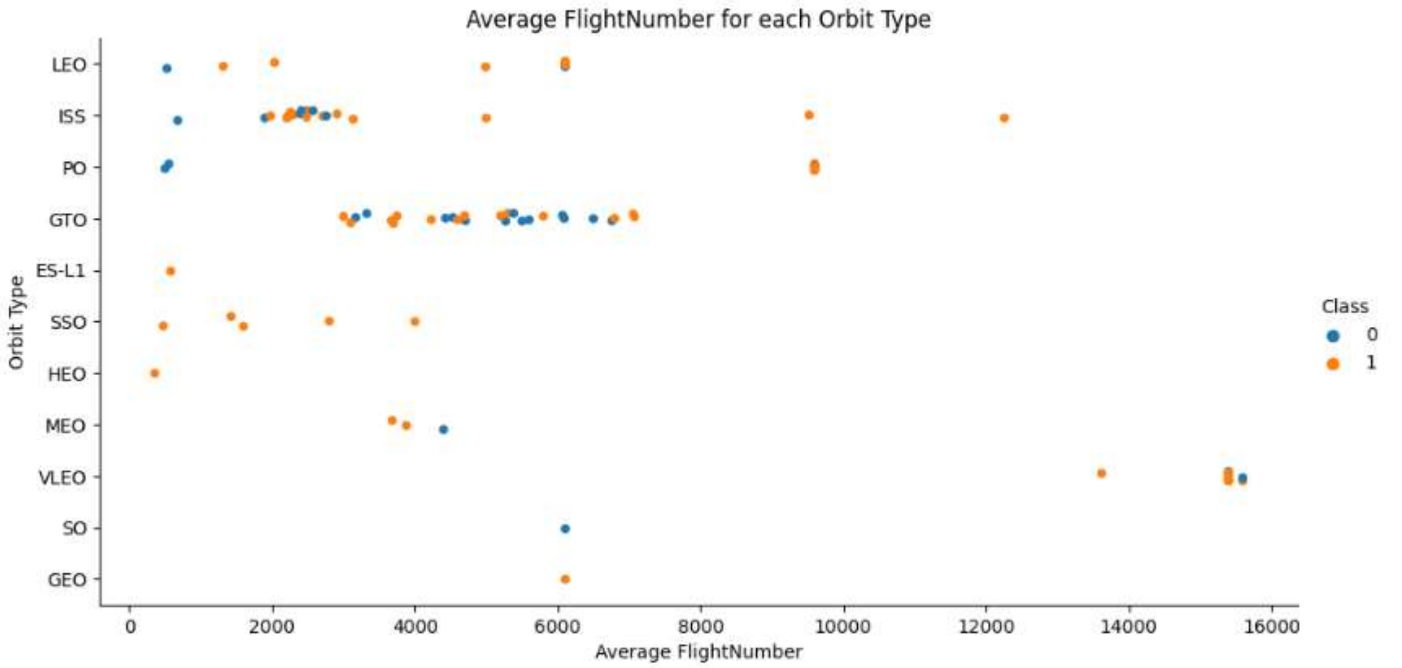
In [19]:

```
# 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

plt.figure(figsize=(10, 6))
sns.catplot(y='Orbit', x='PayloadMass', data=df, hue="Class", aspect=2)

plt.title('Average FlightNumber for each Orbit Type')
plt.xlabel('Average FlightNumber')
plt.ylabel('Orbit Type')
plt.show()
```

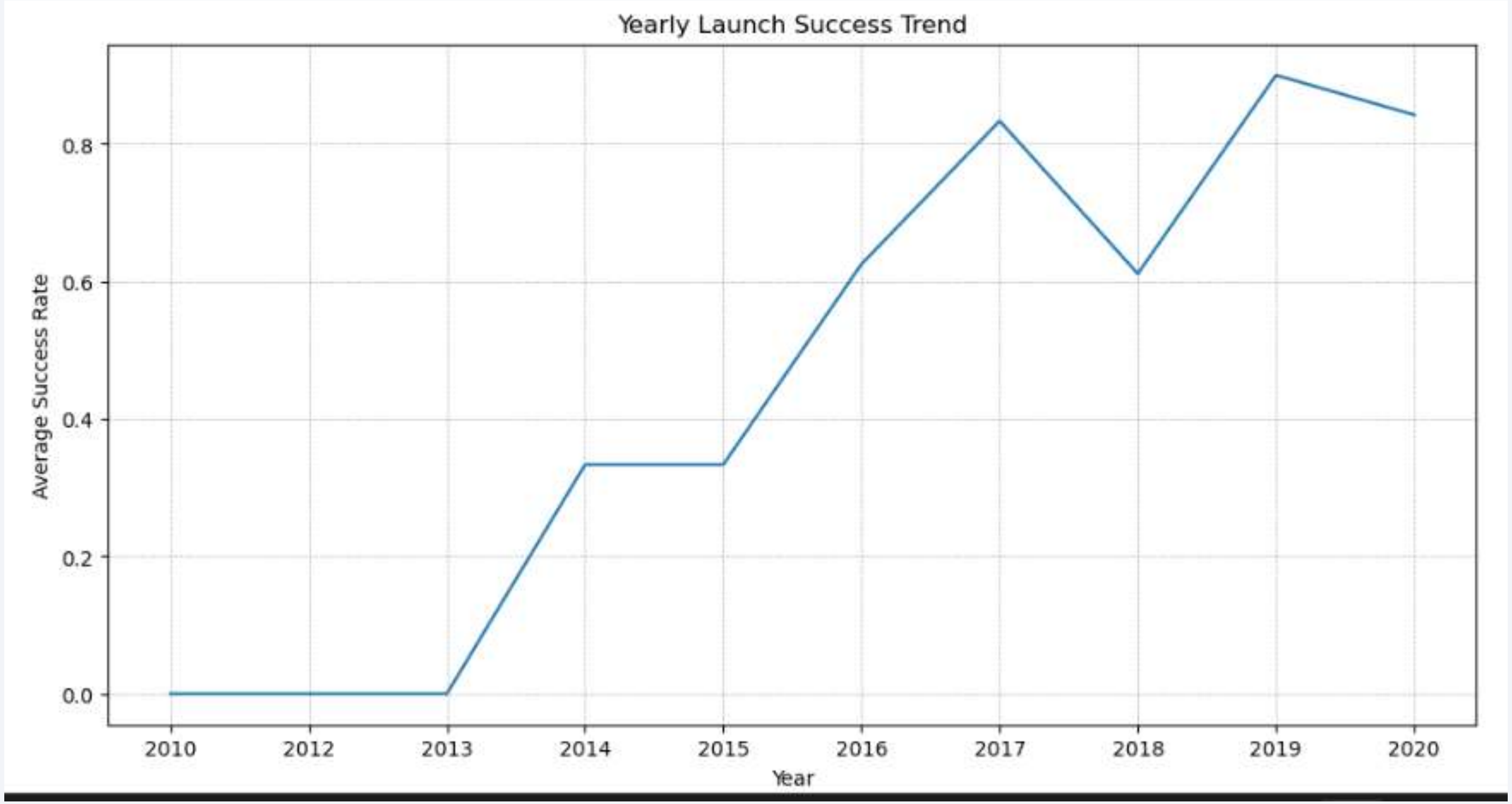
executed in 281ms, finished 22:10:37 2023-08-27

```
C:\Users\adems\anaconda3\envs\notebook-6.4.5\lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has c
hanged to tight
  self._figure.tight_layout(*args, **kwargs)
```

```
<Figure size 1000x600 with 0 Axes>
```



Launch Success Yearly Trend



All Launch Site Names

1.3.1 Task 1

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

In [7]:
```
%sql select distinct Launch_Site from SPACEXTABLE
```

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

Out[7]:

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

The code retrieves a list of unique Launch Site values from the SPACEXTABLE in the database.

Launch Site Names Begin with 'CCA'

1.3.2 Task 2

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

In [8]:
```
%sql select Launch_Site from SPACEXTABLE where Launch_Site like "CCA%" limit 5
```

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

Out[8]:

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

This query fetches the first 5 Launch_Site values from the SPACEXTABLE table that start with "CCA".

Total Payload Mass

1.3.3 Task 3

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

In [9]:
```
%sql select distinct Landing_outcome  from SPACEXTABLE limit 20
```
```
 * sqlite:///my_data1.db
Done.
```

Out[9]:

Landing_Outcome
Failure (parachute)
No attempt
Uncontrolled (ocean)
Controlled (ocean)
Failure (drone ship)
Precluded (drone ship)
Success (ground pad)
Success (drone ship)
Success
Failure
No attempt

In [10]:
```
%sql select sum(PAYLOAD_MASS__KG_) from SPACEXTABLE where Customer = "NASA (CRS)"
```
```
 * sqlite:///my_data1.db
Done.
```

Out[10]:

sum(PAYLOAD_MASS__KG_)
45596

The first query calculates the total payload mass (in kilograms) for all entries in the **SPACEXTABLE** table where the customer is "NASA (CRS)".

The second this query retrieves the first 20 unique landing outcomes from the **SPACEXTABLE** table.

Average Payload Mass by F9 v1.1

1.3.4 Task 4

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

In [11]:
```
%sql select avg(PAYLOAD_MASS__KG_) from SPACEXTABLE where Booster_Version = "F9 v1.1"
```

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

Out[11]:

avg(PAYLOAD_MASS__KG_)
2928.4

this query calculates the average payload mass (in kilograms) for all entries in the SPACEXTABLE table where the booster version is "F9 v1.1".

First Successful Ground Landing Date

1.3.5 Task 5

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

Hint:Use min function

[12]:
```
%sql select min(Date) from SPACEXTABLE where Landing_Outcome = "Success (ground pad)"
```

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

Out[12]:

min(Date)
2015-12-22

This query determines the earliest date on which a successful landing on a ground pad was recorded in the SPACEXTABLE table.

Successful Drone Ship Landing with Payload between 4000 and 6000

1.3.6 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 [13]:
```
%sql select min(Date) from SPACEXTABLE where Landing_Outcome = "Success (ground pad)"
```

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

Out[13]:

min(Date)
2015-12-22

This query seeks the earliest (minimum) date from the SPACEXTABLE where the Landing_Outcome column has the value "Success (ground pad)".

In essence, it's trying to find out the date of the first successful landing on a ground pad recorded in the SPACEXTABLE.

Total Number of Successful and Failure Mission Outcomes

1.3.7 Task 7

List the total number of successful and failure mission outcomes

In [16]:
```
%sql select distinct Mission_Outcome from SPACEXTABLE
```

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

Out[16]:

Mission_Outcome
Success
Failure (in flight)
Success (payload status unclear)
Success

In [20]:
```
%sql SELECT COUNT(*) FROM SPACEXTABLE WHERE Mission_Outcome IN ('Success', 'Success (payload status unclear)')
```

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

Out[20]:

COUNT(*)
99

In [40]:
```
%%sql SELECT COUNT(*)
FROM SPACEXTABLE
WHERE Mission_Outcome = 'Failure (in flight)';
```

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

Out[40]:

COUNT(*)
1

The first query retrieves all unique values of the Mission_Outcome column from the SPACEXTABLE. It gives you an idea of the different mission outcomes that have been recorded in the table.

The second query counts the number of records in **SPACEXTABLE** where the **Mission_Outcome** is either 'Success' or 'Success (payload status unclear)'. It tells you how many missions had these particular outcomes.

The third query counts the number of records in **SPACEXTABLE** where the **Mission_Outcome** is 'Failure (in flight)'. It gives you the number of missions that failed in flight.

Boosters Carried Maximum Payload

In [30]:

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

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

Out[30]:

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

This subquery finds the maximum value of the PAYLOAD_MASS__KG_ column from the SPACEXTABLE. It identifies the heaviest payload ever launched as per the records in the table.

This query then uses the result of the subquery to filter the SPACEXTABLE and retrieve the Booster_Version for the record(s) with that maximum payload mass.

2015 Launch Records

In [59]:
```
%%sql SELECT
    substr(Date, 6, 2) as Month,
    Landing_Outcome,
    Booster_Version,
    Launch_Site
FROM SPACEXTABLE
WHERE
    Landing_Outcome = "Failure (drone ship)"
    AND substr(Date, 1, 4) = '2015';
```

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

Out[59]:

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

substr(Date, 6, 2) as Month:
This extracts the month from the Date column, assuming the Date is in a format like YYYY-MM-DD. The result will be assigned the alias Month.

Landing_Outcome, Booster_Version, Launch_Site:
These select the respective columns from the table, retrieving the landing outcome, booster version, and launch site details.

WHERE Landing_Outcome = "Failure (drone ship)":
This filters the records to only those where the landing outcome was a failure on a drone ship.

AND substr(Date, 1, 4) = '2015':
This further filters the results to only include records from the year 2015. It uses the substr function to extract the year portion of the Date column and compares it to '2015'.

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

1.3.9 Task 10

Rank the count of landing outcomes (such as Fa
descending order.

In [60]:

```
%%sql SELECT
    Landing_Outcome,
    COUNT(*) as Count
FROM SPACEXTABLE
WHERE
    Date >= '2010-06-04'
    AND Date <= '2017-03-20'
GROUP BY Landing_Outcome
ORDER BY Count DESC;
```

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

Out[60]:

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

Landing_Outcome, COUNT(*) as Count:
This selects the Landing_Outcome column and counts the number of occurrences of each unique landing outcome. The count is given an alias Count.

WHERE Date >= '2010-06-04' AND Date <= '2017-03-20':
This filters the records to only those that have a Date between '2010-06-04' and '2017-03-20' inclusive.

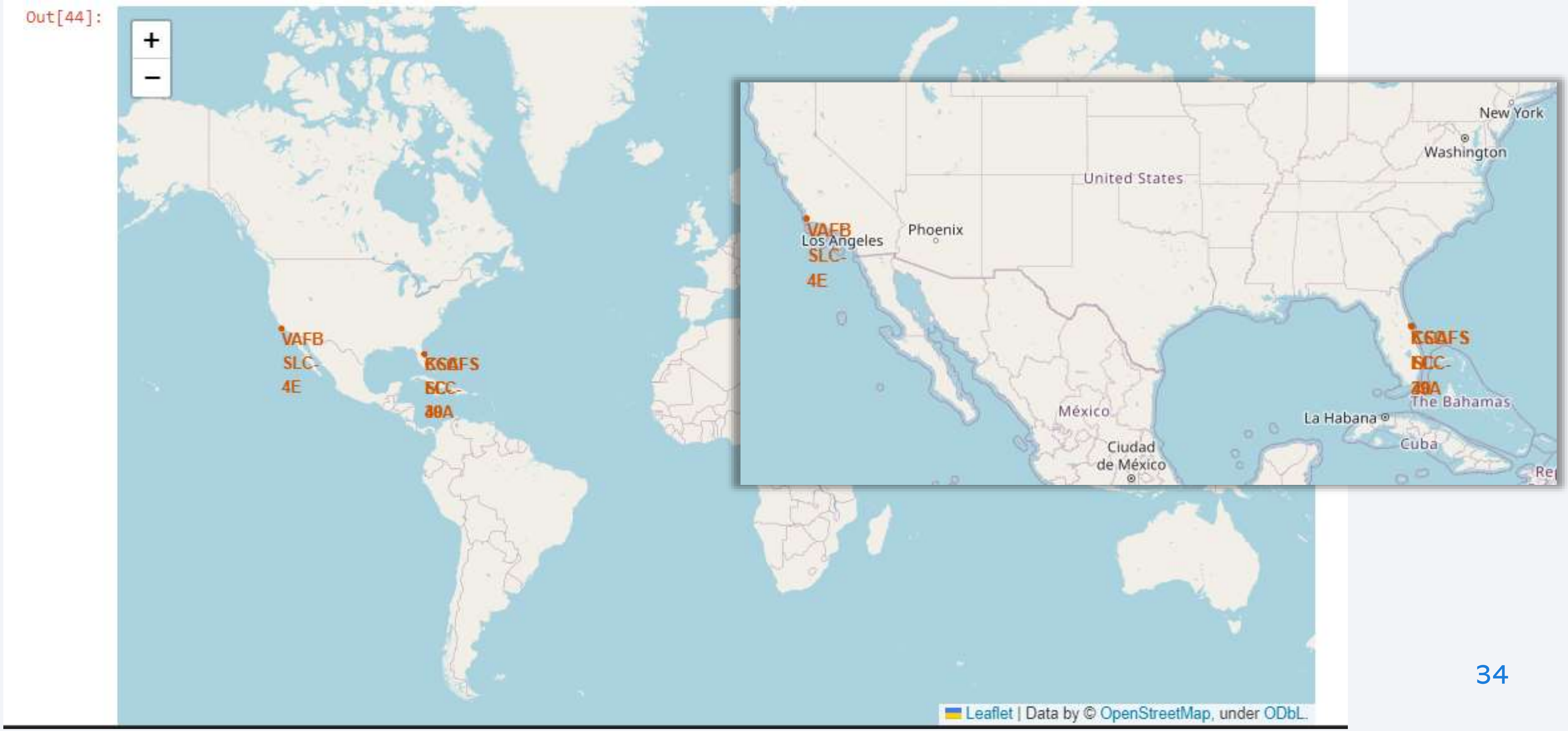
GROUP BY Landing_Outcome:
This groups the results by unique values in the Landing_Outcome column, which allows the COUNT(*) function to count the occurrences of each landing outcome.

ORDER BY Count DESC:
This orders the results in descending order based on the count of each landing outcome, so the most frequent outcomes will appear first.

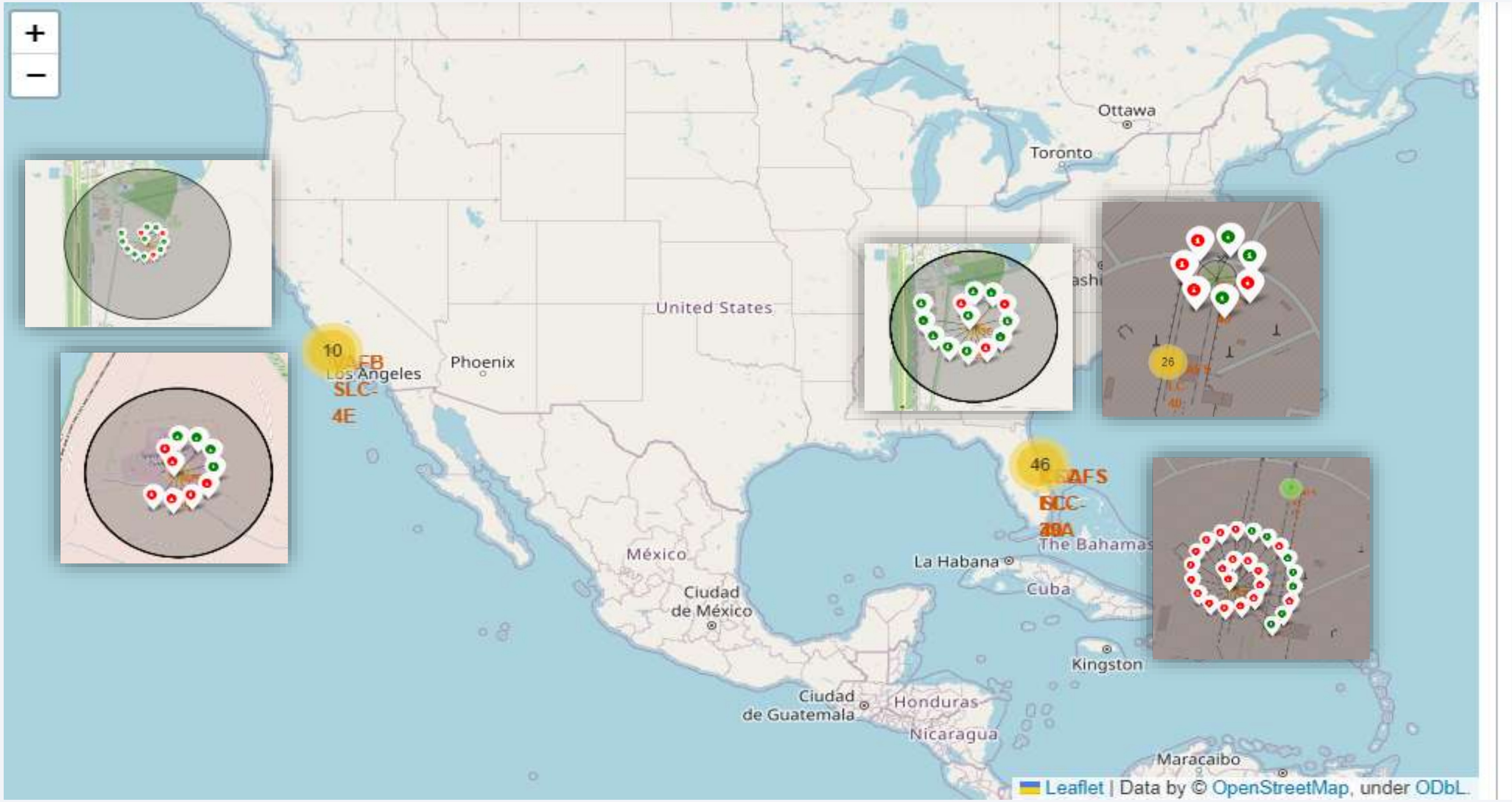
Section 3

Launch Sites Proximities Analysis

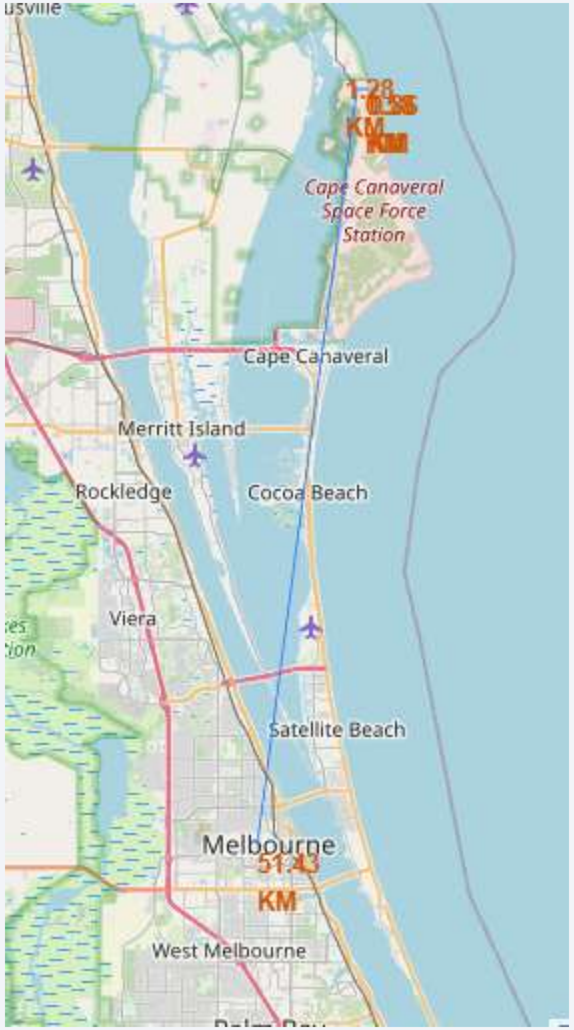
Launch Sites



Launch Sites' Success



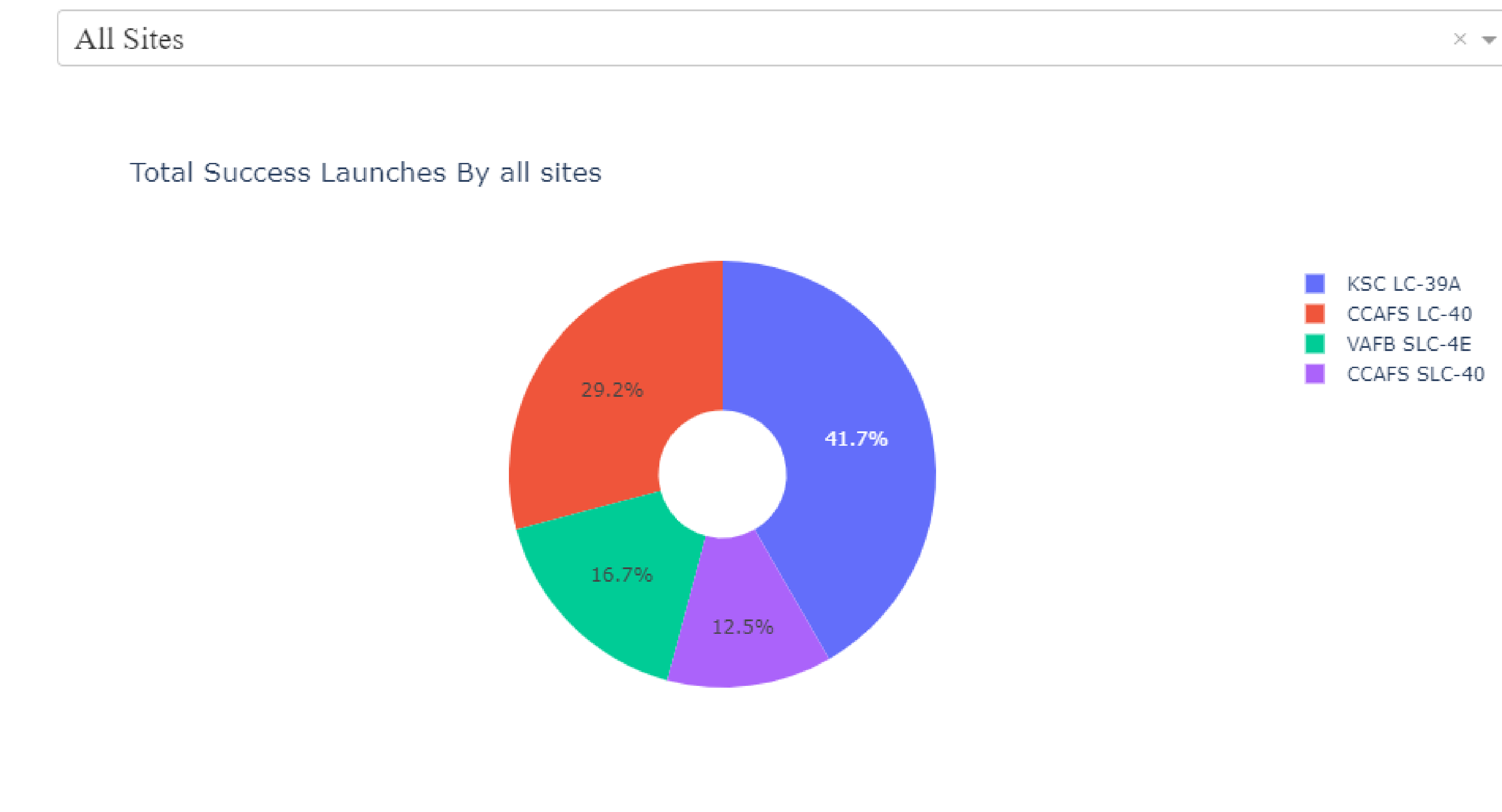
Proximities to railway, highway, closest city,



Section 4

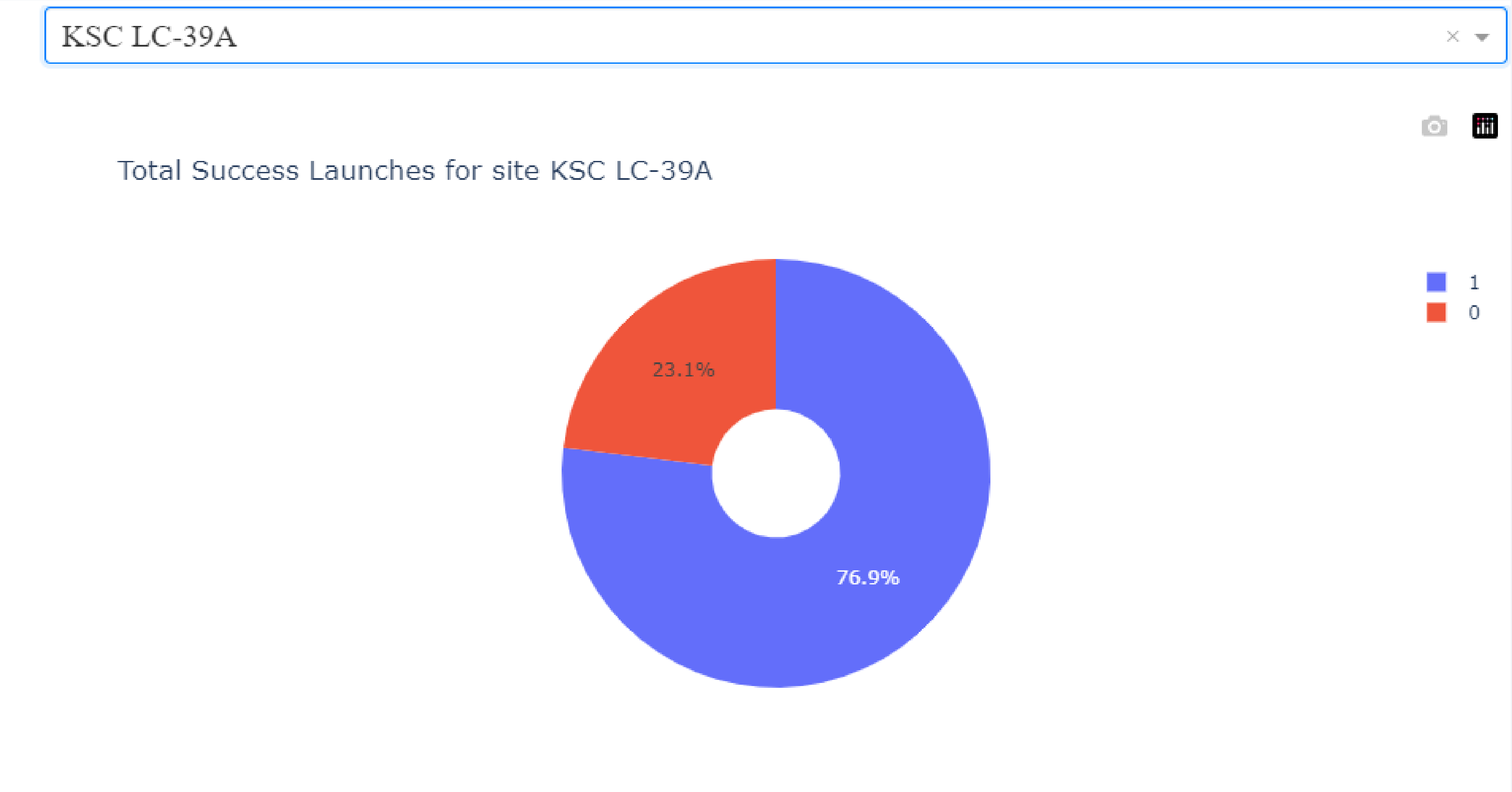
Build a Dashboard with Plotly Dash

Total Success Launches By all sites



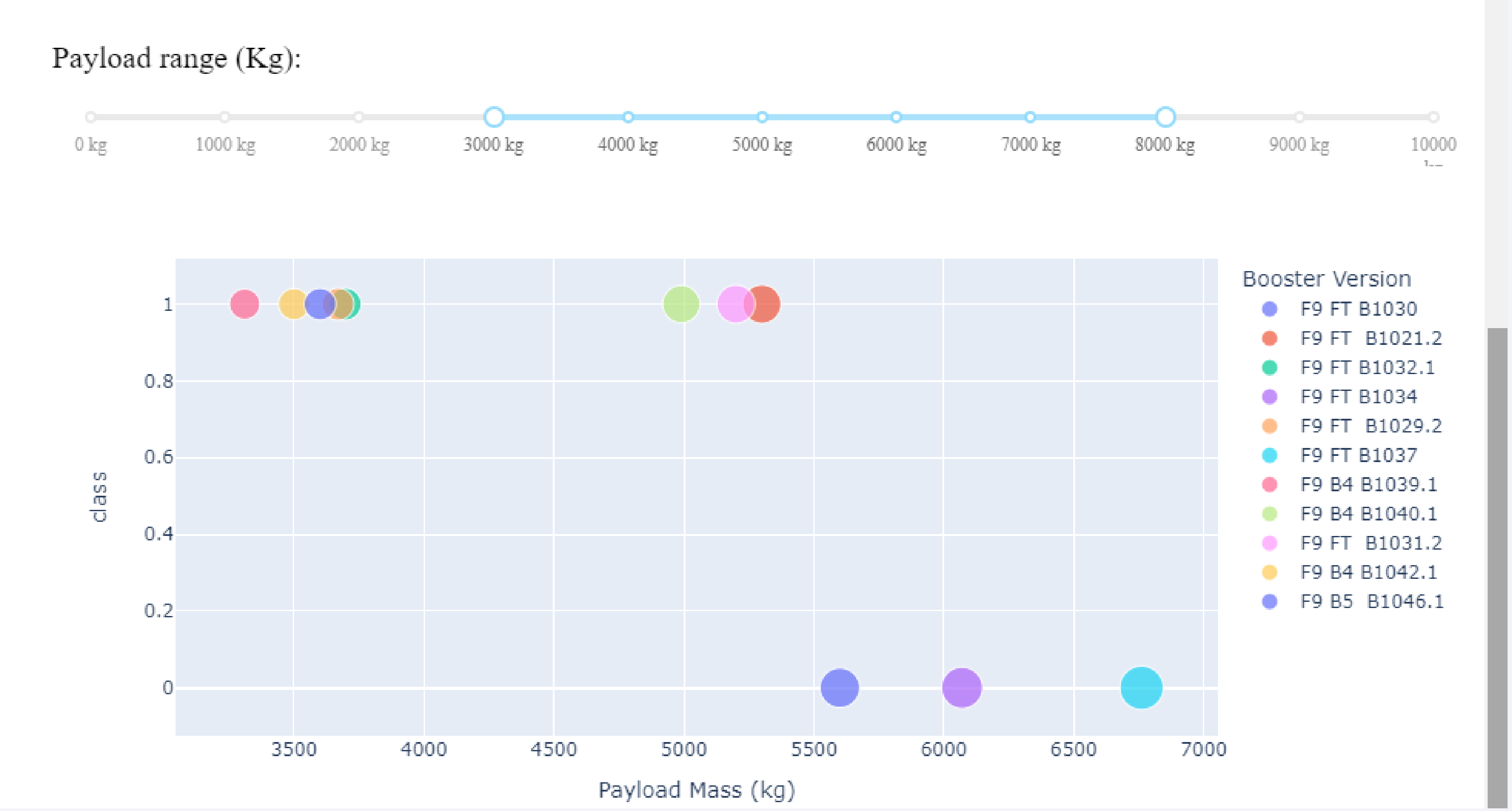
- Highest success among all sites belongs to KSC LC-39A with 41.7%

<Dashboard Screenshot 2>



- KSC LC-39A's success is 76.9%.

Payload Mass vs Success

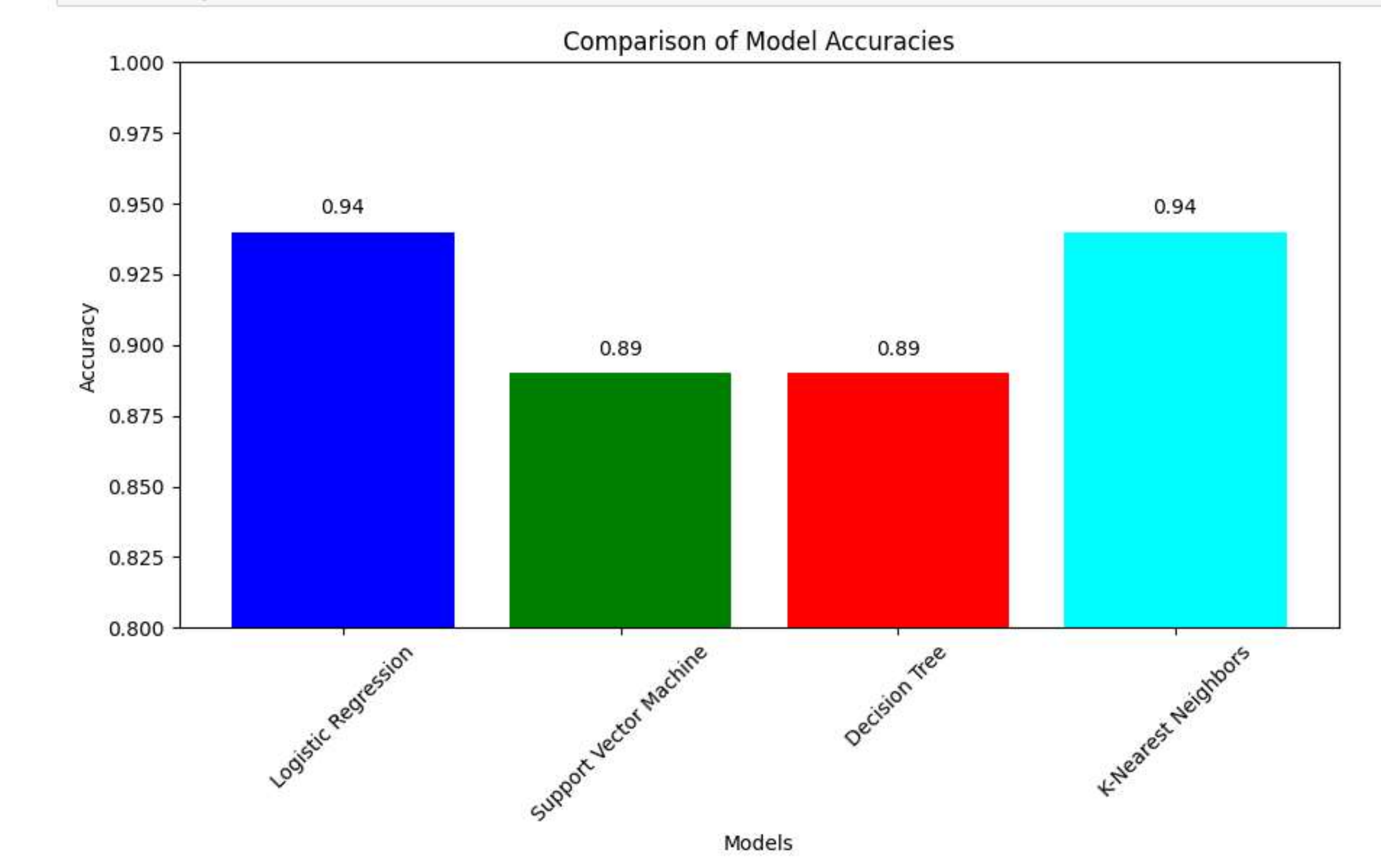


- Distribution of Booster versions regarding to payload mass between 3000-8000 kg.

Section 5

Predictive Analysis (Classification)

Classification Accuracy



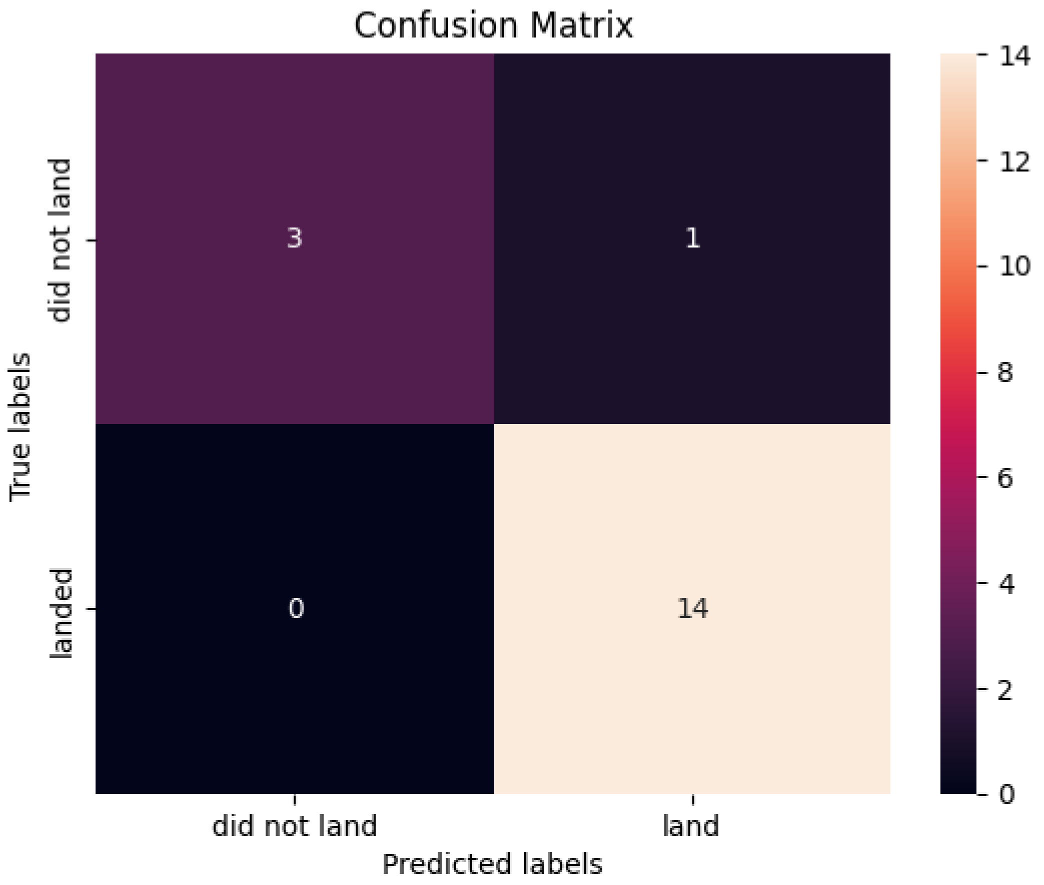
- **While both Logistic Regression and KNN achieved the same top accuracy on the test data, Logistic Regression is selected based on its advantages in interpretability, model complexity, prediction speed, and potential for better generalization.**

Confusion Matrix

Lets look at the confusion matrix:

In [48]:

```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Conclusions

- Logistic Regression algorithm appears to be the most suitable for this dataset even it has the same accuracy score (0.94) with KNN.
- Lighter payloads tend to have better performance compared to heavier ones.
- As time progresses, SpaceX's launch success rates improve, suggesting they are on a path to perfecting their launch capabilities.
- Among all launch sites, KSC LC-39A has the highest number of successful launches.
- The orbits GEO, HEO, SSO, and ES-L1 demonstrate the highest success rates.
- The total payload mass (in kilograms) for NASA (CRS) is 45596 kg.

Thank you!