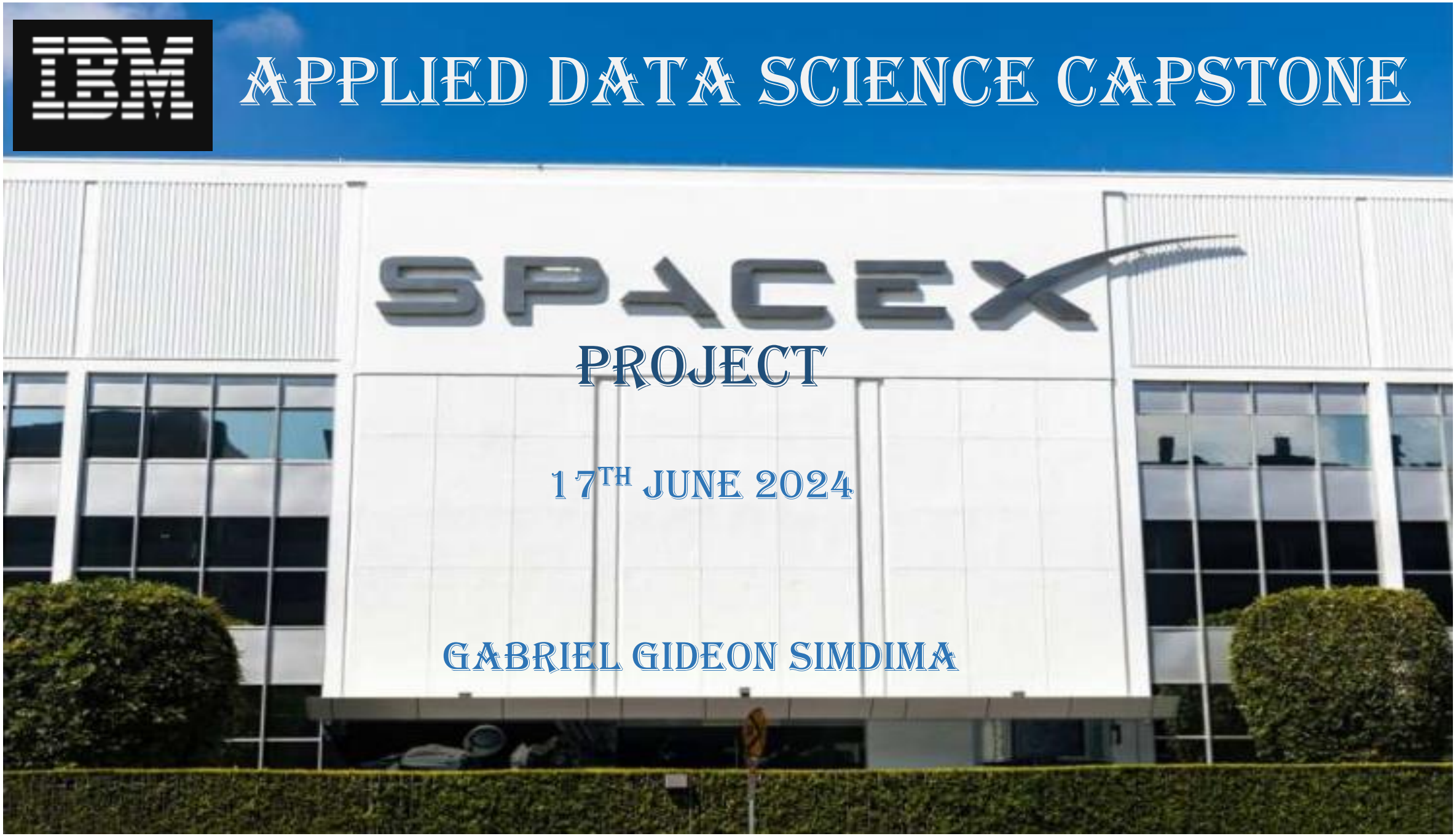
APPLIED DATA SCIENCE CAPSTONE



PROJECT

17TH JUNE 2024

GABRIEL GIDEON SIMDIMA

Outline

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

Executive Summary

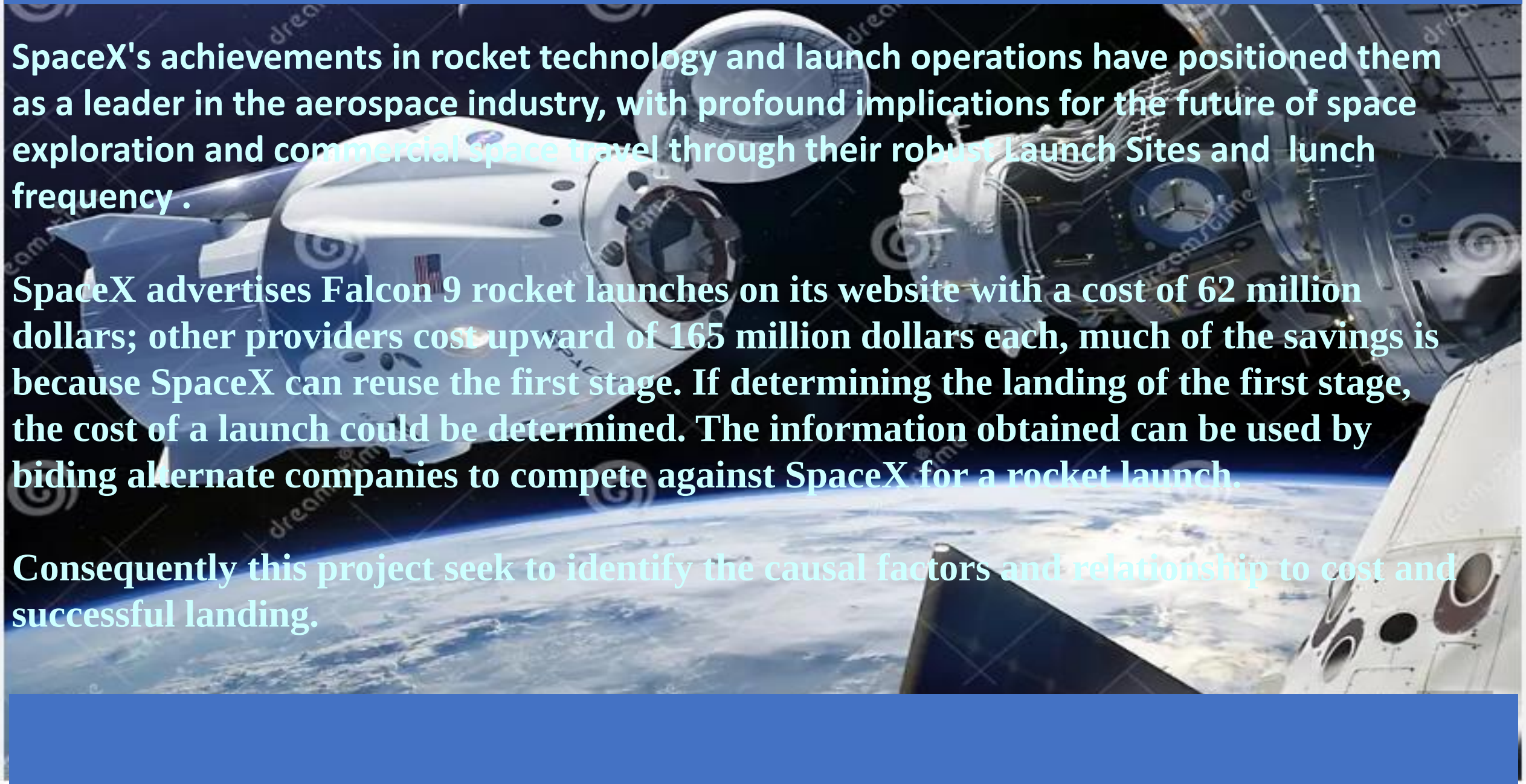
Summary of methodologies

- Data collection
- Data wrangling
- EDA with data visualization
- EDA with SQL
- Building an interactive map with Folium
- Building a Dashboard with Plotly Dash
- Predictive analysis (Classification)

Summary of all

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

Introduction



SpaceX's achievements in rocket technology and launch operations have positioned them as a leader in the aerospace industry, with profound implications for the future of space exploration and commercial space travel through their robust Launch Sites and lunch frequency .

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. If determining the landing of the first stage, the cost of a launch could be determined. The information obtained can be used by biding alternate companies to compete against SpaceX for a rocket launch.

Consequently this project seek to identify the causal factors and relationship to cost and successful landing.

SECTION 1

Methodology

Methodology *Executive Summary*

Data collection from SpaceX Rest API, Web Scraping from page of Falcon and Falcon Heavy Langes

Data Wrangling performance through One Hot Encoding and removing empty dataset

Performing exploratory data analysis EDA and visualization using SQL

Performing interactive visual analytics using Folium and Plotly Dash

Perform predictive analysis using classification methods such as logistic regression, K-Nearest Neighbors, Support Vector Machines and Decision Tree

Data collection

In this project data collection was carried out through request SpaceX API and web scraping

1. The Request to SpaceX API consist of:

Gathering past data of SpaceX through source API after which it is retrieved, processed and ensuring that he data is the Falcon 9 launch. Also the missing payloads weight data from secret mission with average available

2. Web scraping was carried out requesting Falcon 9 and Falcon Heavy launch data from Wikipedia page. Then accessed the Falcon 9 page, extracted all the column names from HTML Tables, parsed and transform into pandas frame ready for the analysis.

Data collection – spaceX API

- SpaceX API to collect data
- clean the requested data and data wrangling and formatting.
- Return spacex data to JSON
- Data into .csv

1 After response fron API conver to a.JSON file

```
In [2]:
# Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
        if x:
            response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
            BoosterVersion.append(response['name'])
```

2 Custom function to clean data

```
In [15]:
BoosterVersion

Out[15]:
[]
```

Now, let's apply getBoosterVersion function method to get the booster version

```
In [16]:
# Call getBoosterVersion
getBoosterVersion(data)
```

the list has now been update

```
In [17]:
BoosterVersion[0:5]

Out[17]:
['Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 9']

In [18]:
# Call getLaunchSite
getLaunchSite(data)

In [19]:
# Call getPayloadData
getPayloadData(data)

In [20]:
# Call getCoreData
getCoreData(data)
```

3 List to Dict to dataframe

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}

```

4 Filter dataframe and export

```
# Hint data['BoosterVersion']!='Falcon 1'
filt =  df['BoosterVersion']!=  'Falcon 1'
data_falcon9 = df.loc[filt]
data_falcon9.head()
```

Out[24]:

Data Collection - Scraping

- We webscrap from Falcon 9 launch records using BeautifulSoup
- Then parsed the table and converted it into a pandas dataframe.
- Save dataframe to .csv

1 Creating beautifulSoup object

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

2 Creating beautifulSoup object

```
# use requests.get() method with the provide
# assign the response to a object
response = requests.get(static_url)
```

3 Finding Table

```
# Use the find_all function in the BeautifulSoup object, with elemen
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

4 Getting Columns Name

```
column_names = []

# Apply find_all() function with `th` element
# Iterate each th element and apply the provi
# Append the Non-empty column name (`if name

table = first_launch_table.find_all('th')
for row in table:
    name = extract_column_from_header(row)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

5 Getting Columns Name

```
launch_dict= dict.fromkeys(column_names)

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

# Let's initial the launch_dict with each
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'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

6 Append data to keys

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(s
    # get table row
    for rows in table.find_all("tr"):
```

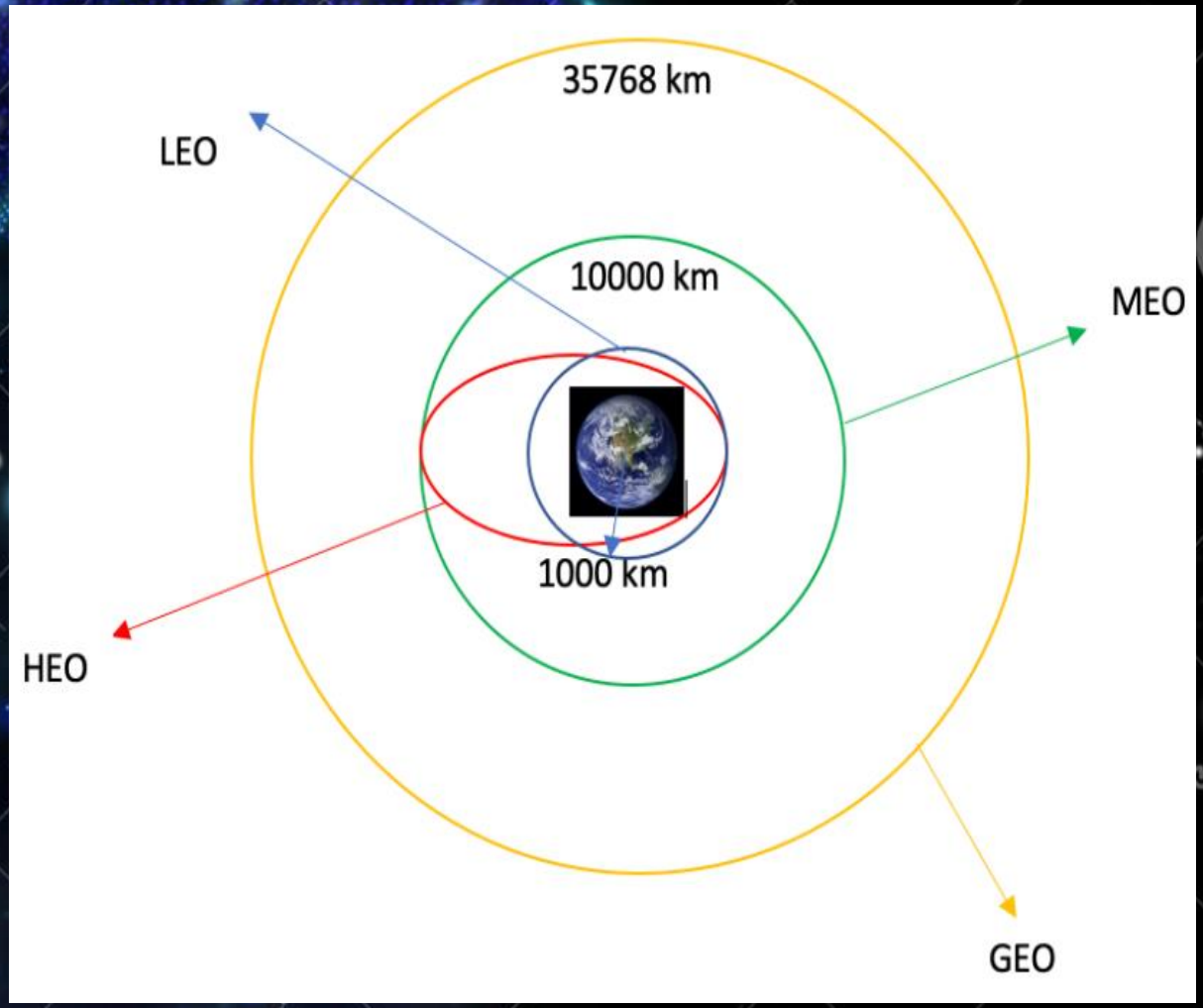
7 Covert dict to dataframe and to .csv

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

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

Data Wrangling

- The number of launches at each site and number of occurrences of each orbit were determined
- Landing outcome label frame outcome table and the result in .csv was exported.



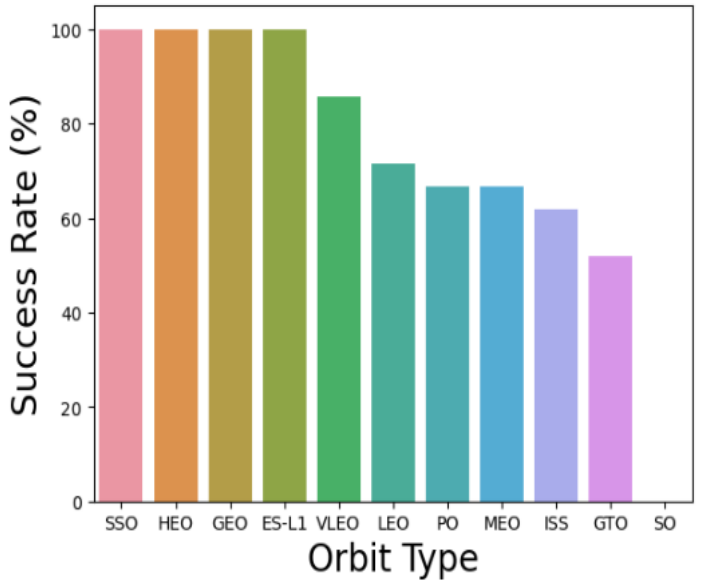
Eda with Data Visualization

Visualized the relationship between

- flight number and Payload mass,
- Payload and lunch site
- Orbit and flight number
- Fight number and Launch Site
- Orbit and Payload Mass
- Payload and Orbit Type

Mean vs Orbit

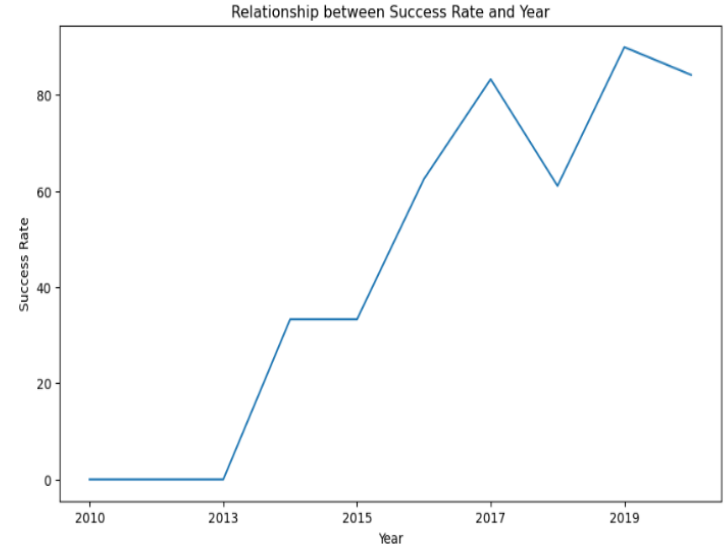
```
[39]:
# HINT use groupby method on Orbit column and get the mean of Class column
orbit_success = df.groupby(['Orbit'])['Class'].aggregate(np.average).reset_
orbit_success['Class'] = np.round(orbit_success['Class']*100, 2)
sns.barplot(x='Orbit', y='Class', data=orbit_success)
plt.ylabel("Success Rate (%)",fontsize=20)
plt.xlabel("Orbit Type",fontsize=20)
plt.show()
```



Success rate vs Year

```
[16]:
# Plotting a line chart with x axis to be the extracted year and y axis to be the success rate
Extract_year(df)
df['Year']=year
fig,ax=plt.subplots()
df_success1=df.groupby('Year')['Class'].mean()*100
df_success1.plot(kind='line', figsize=(10,6))
plt.xlabel('Year') # add to x-label to the plot
plt.ylabel('Success Rate') # add y-label to the plot
plt.title('Relationship between Success Rate and Year') # add title to the plot

plt.show()
```



EDA with SQL

1 SpaceX dataset

We loaded the SpaceX dataset into a PostgreSQL database without leaving the jupyter notebook.

2 We applied EDA

EDA with SQL were applied to get insight from the data. We wrote queries to find out:

- 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

Interactive Map with Folium

The activities carried out are as follow

- 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

Dashboard with Plotly Dash

The activities carried out are

- 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.

Predictive Analysis (Classification)

Here we

- loaded the data using numpy and pandas, then transformed the data, split the data into training and testing.
- built different machine learning models and tune different hyperparameters using GridSearchCV.
- used accuracy as the metric for out the model, improved the model using feature engineering and algorithm tuning.
- lastlydetermined the best performing classification model.

Results

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

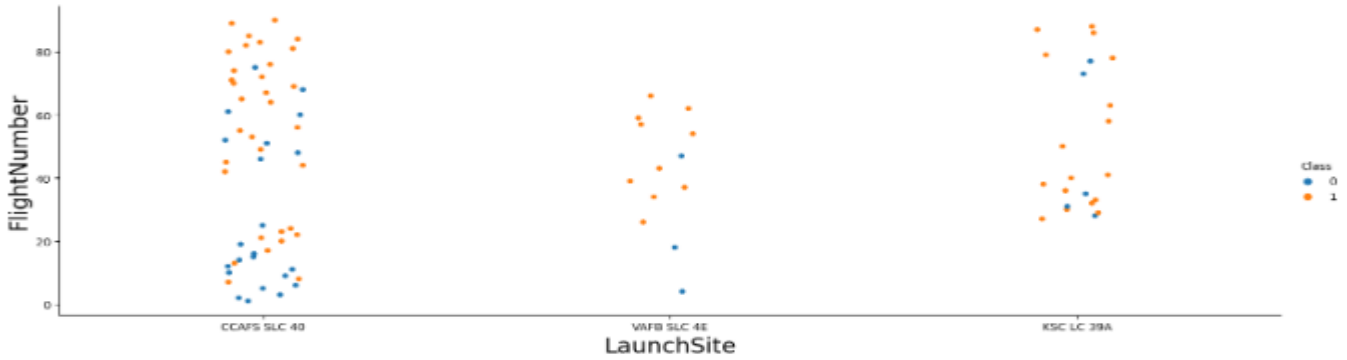
SECTION 2

Insight Drawn from EDA

Flight Number Vs Lunch Site

```
# Plot a scatter point chart with x axis to be Flight Number and y axis to b
sns.catplot(y="FlightNumber",x="LaunchSite",hue='Class',data=df, aspect=3)
plt.xlabel("LaunchSite",fontsize=20)
plt.ylabel("FlightNumber",fontsize=20)
plt.show()
```

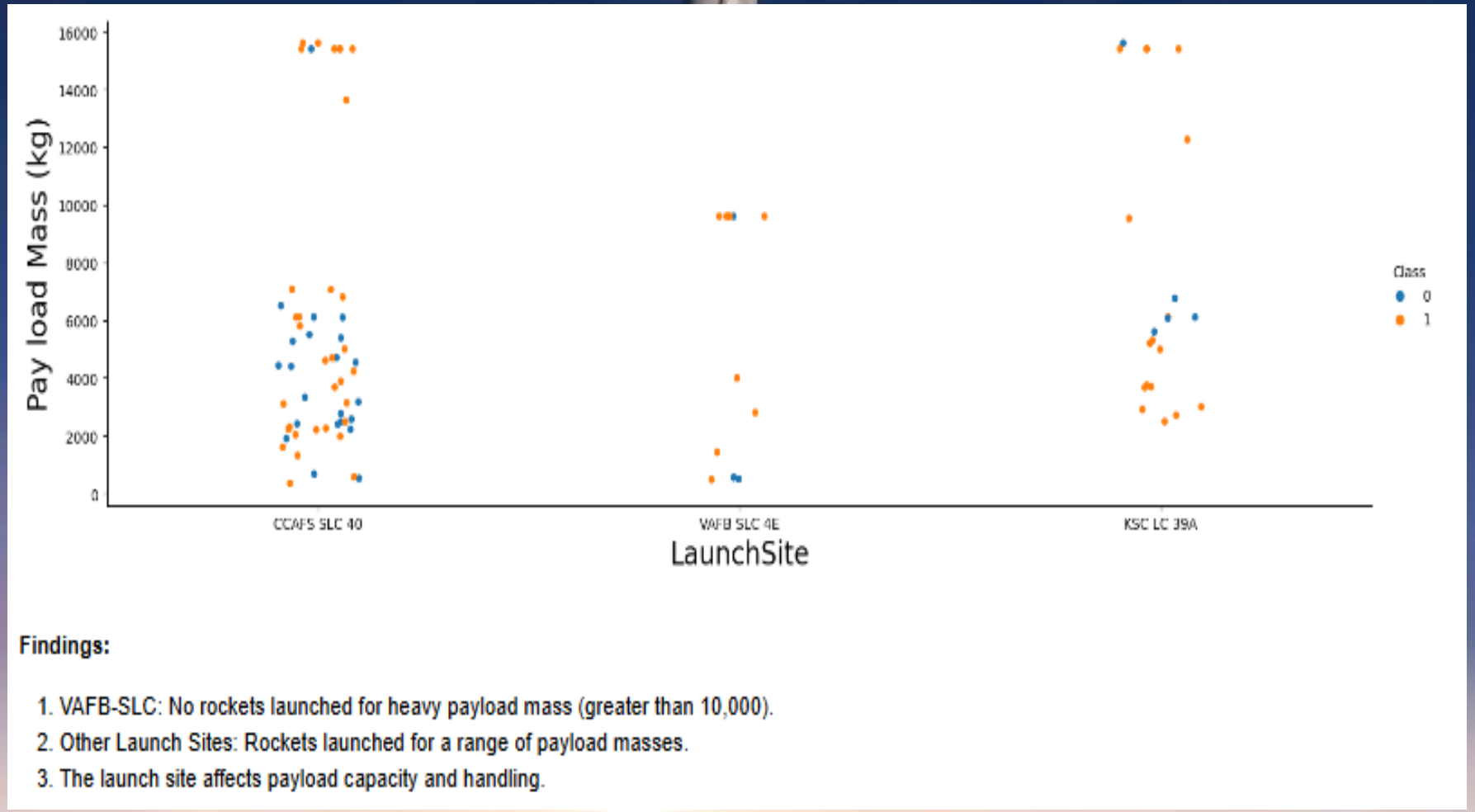
```
C:\Users\GABRIEL SIMDIMA\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: U
serWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



Findings:

1. CCAFS LC-40: No strong relationship between flight number and success rate. Success appears consistent across various flight numbers.
2. KSC LC-39A: Similarly, no strong correlation between flight number and success rate. Success rates vary across flight numbers.
3. VAFB SLC 4E: Success rates appear consistent, regardless of flight number.

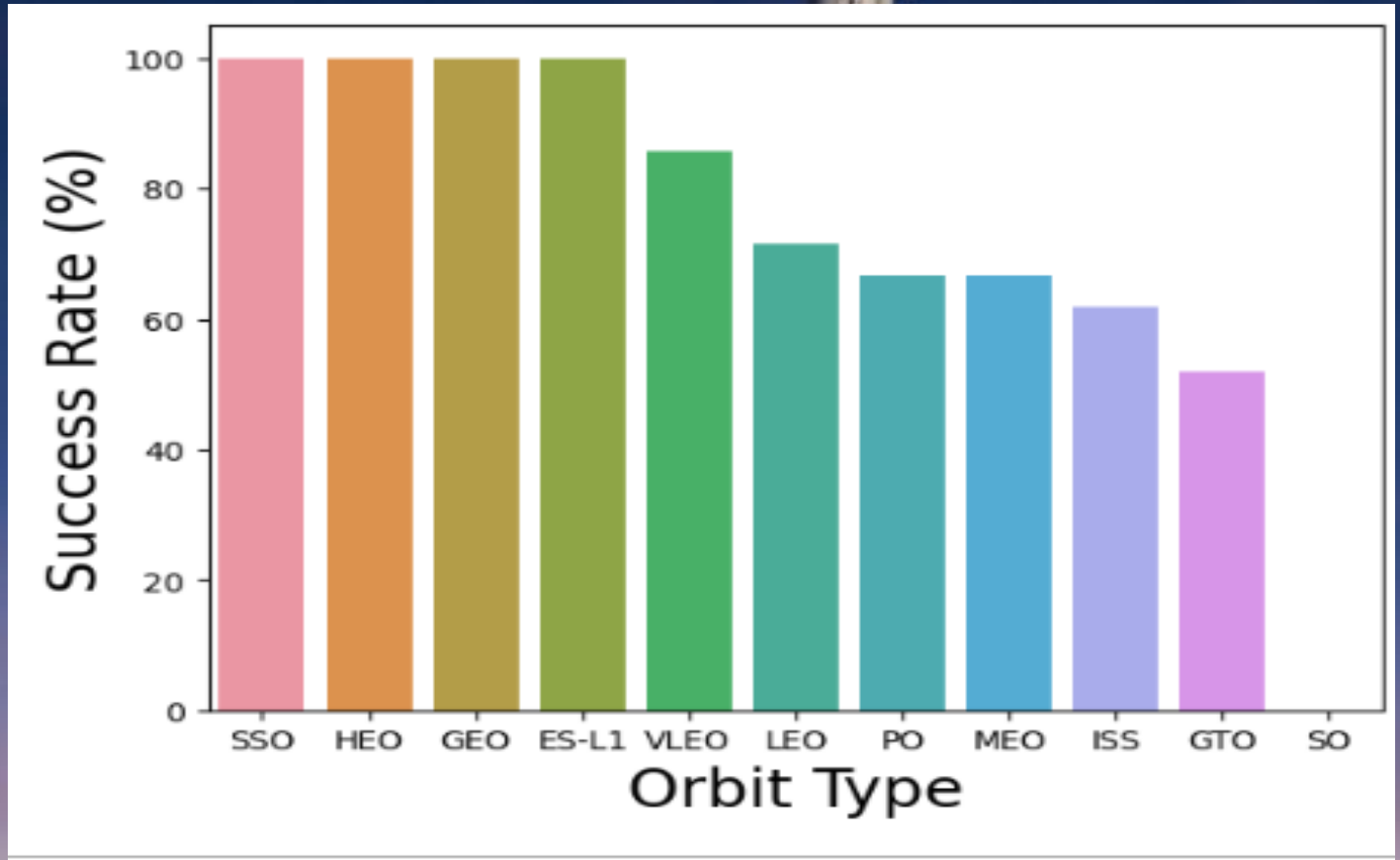
Payload Mass Vs Lunch Site



Findings:

1. VAFB-SLC: No rockets launched for heavy payload mass (greater than 10,000).
2. Other Launch Sites: Rockets launched for a range of payload masses.
3. The launch site affects payload capacity and handling.

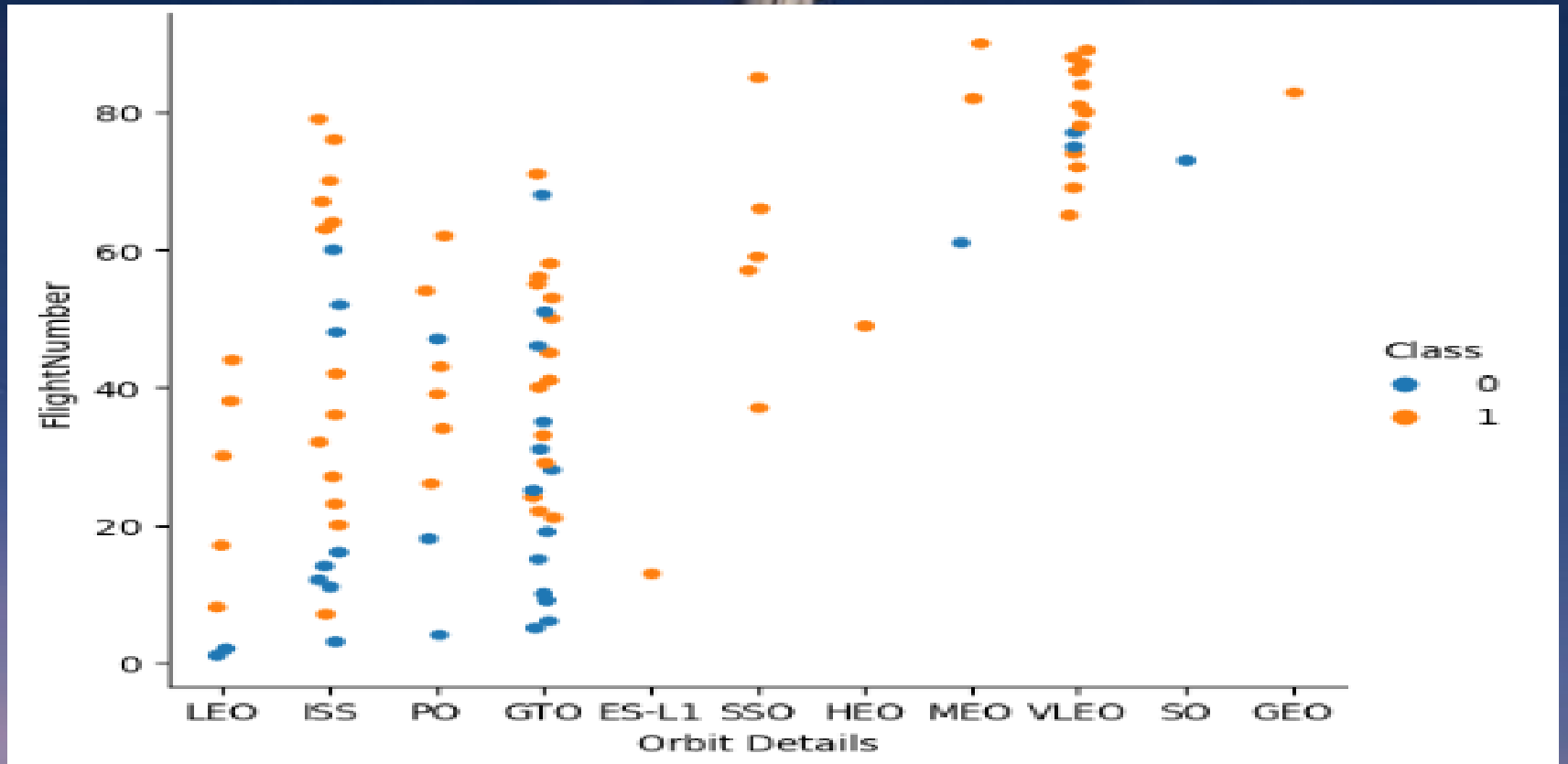
Success Rate Vs Orbit Type



Findings:

1. SSE, HEO, GEO, ES-L1 Orbits: 100% success rates, indicating high reliability.
2. Other Orbit Types: Varying success rates, suggesting mission complexities.

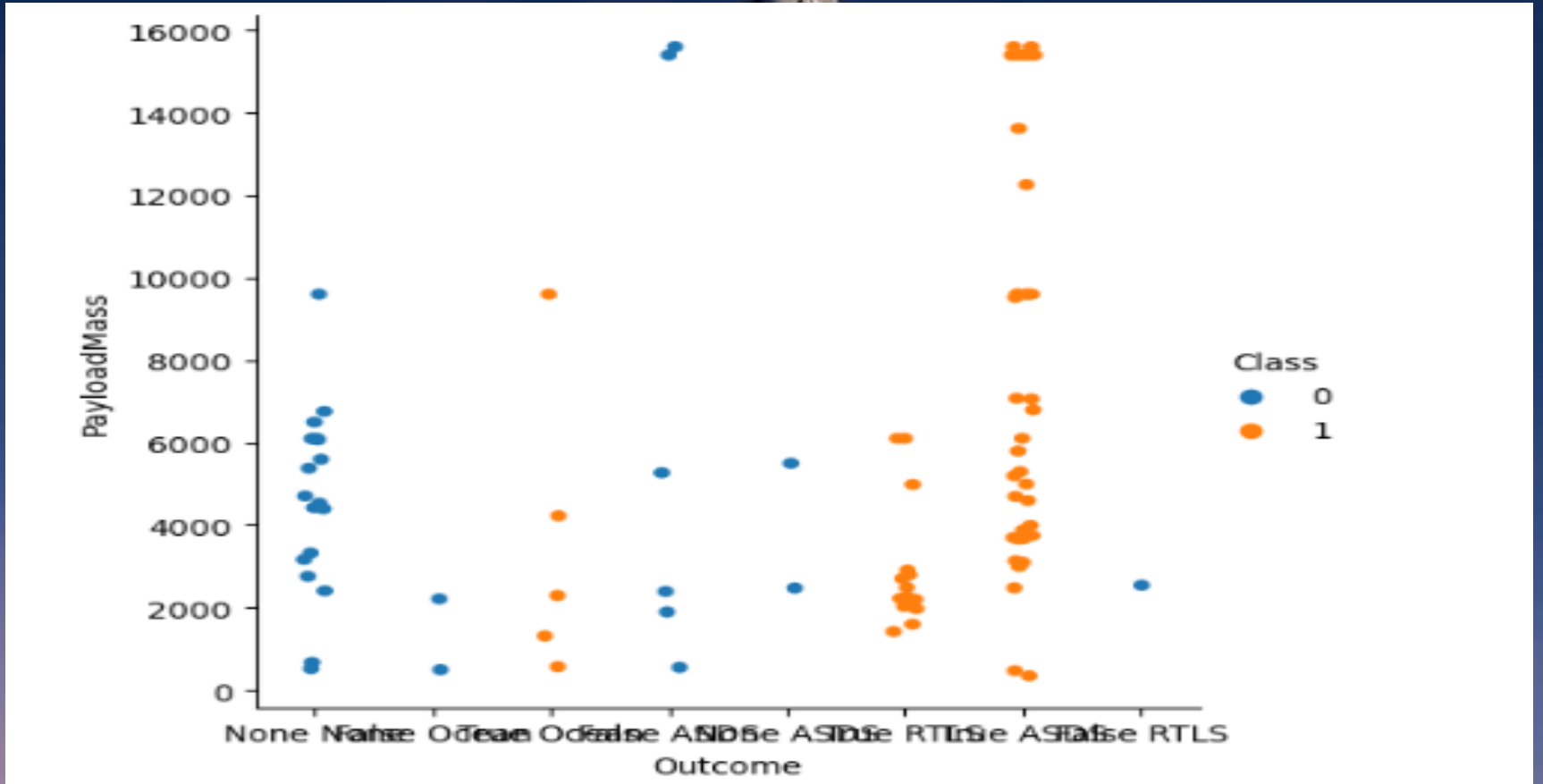
Flight Number Vs Outcome



Findings:

1. LEO Orbit: Success appears correlated with the number of flights.
2. GTO Orbit: No clear relationship between flight number and success.

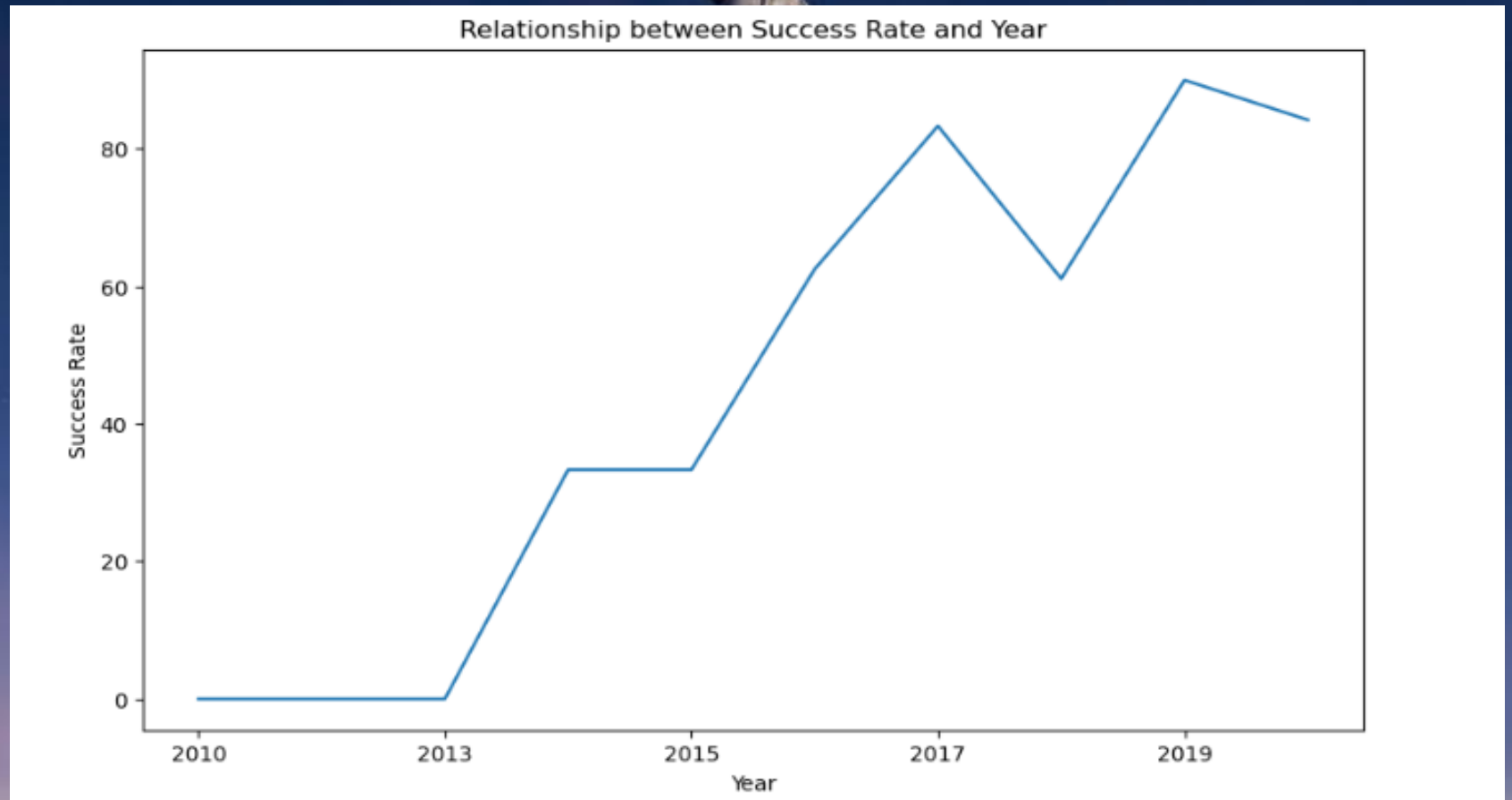
Payload Mass Vs Outcome



Findings:

1. PO, LEO, ISS Orbits: Higher payload mass corresponds to a more successful landing.
2. GTO Orbit: Payload mass does not clearly affect landing success.

Success Rate Vs Year



Findings:

1. 2013 to 2020: There is consistent increase in success rates.
2. Highlight: SpaceX's continual improvement in launch reliability.

Unique Lunch Sites

This shows the unique Lunch Sites from the spaceX data obtained

In [11]:

```
count=0
for site in df["Launch_Site"]:
    if "CCA" in site and count < 5:
        print(site)
        count=count+1

```

```
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
```

SECTION 3

EDA with .SQL

Unique Lunch Sites

This shows the unique Lunch Sites from the spaceX data obtained

In [11]:

```
count=0
for site in df["Launch_Site"]:
    if "CCA" in site and count < 5:
        print(site)
        count=count+1

```

```
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
```

Lunch Sites names beginning with CCA

In [48]:
```
%sql SELECT * \
    FROM SPACEXTBL \
    WHERE LAUNCH_SITE LIKE'CCA%' LIMIT 5;
```

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

Out[48]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
04/06/2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08/12/2010	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)
22/05/2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08/10/2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01/03/2013	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 as 45596 AS shown

In [49]:

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) \
    FROM SPACEXTBL \
    WHERE CUSTOMER = 'NASA (CRS)';
```

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

Out[49]:

SUM(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

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

```
In [20]:  1 df_F9_1_1=df[df["Booster_Version"] == "F9 v1.1"]
          2 df_F9_1_1["PAYLOAD_MASS__KG_"].mean()

Out[20]: 2928.4
```

First successful landing outcome Date

```
In [21]:  1 df_ground_pad=df[df['Landing_Outcome'] == 'Success (ground pad)' ]
          2 df_ground_pad.iloc[0,0]

Out[21]: '22/12/2015'
```

Successful Drone Ship Landing with Payload between 4000 and 6000

In [22]:
```
%sql select BOOSTER_VERSION from SPACEXTBL where Landing_Outcome='Success (drone ship)' and PAYLOAD_MASS__KG_ BETWEEN 4000 a
```

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

Out[22]:

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

Total Number of Successful and Failure Mission Outcomes

```
In [24]: 1 %sql select count(Mission_Outcome) as missionoutcomes from SPACEXTBL GROUP BY MISSION_OUTCOME;

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

Out[24]:

missionoutcomes
1
98
1
1

Boosters Carried Maximum Payload

```
n [26]:  1 %sql select BOOSTER_VERSION as boosterversion from SPACEXTBL where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from
```

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

ut[26]:

boosterversion
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

In [29]:
```
%sql SELECT substr(Date,4,2) as month, Date,Booster_Version, Launch_Site, [Landing_Outcome] \
FROM SPACEXTBL \
where [Landing_Outcome] = 'Failure (drone ship)' and substr(Date,7,4)='2015';
```

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

Out[29]:

month	Date	Booster_Version	Launch_Site	Landing_Outcome
01	10/01/2015	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
04	14/04/2015	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

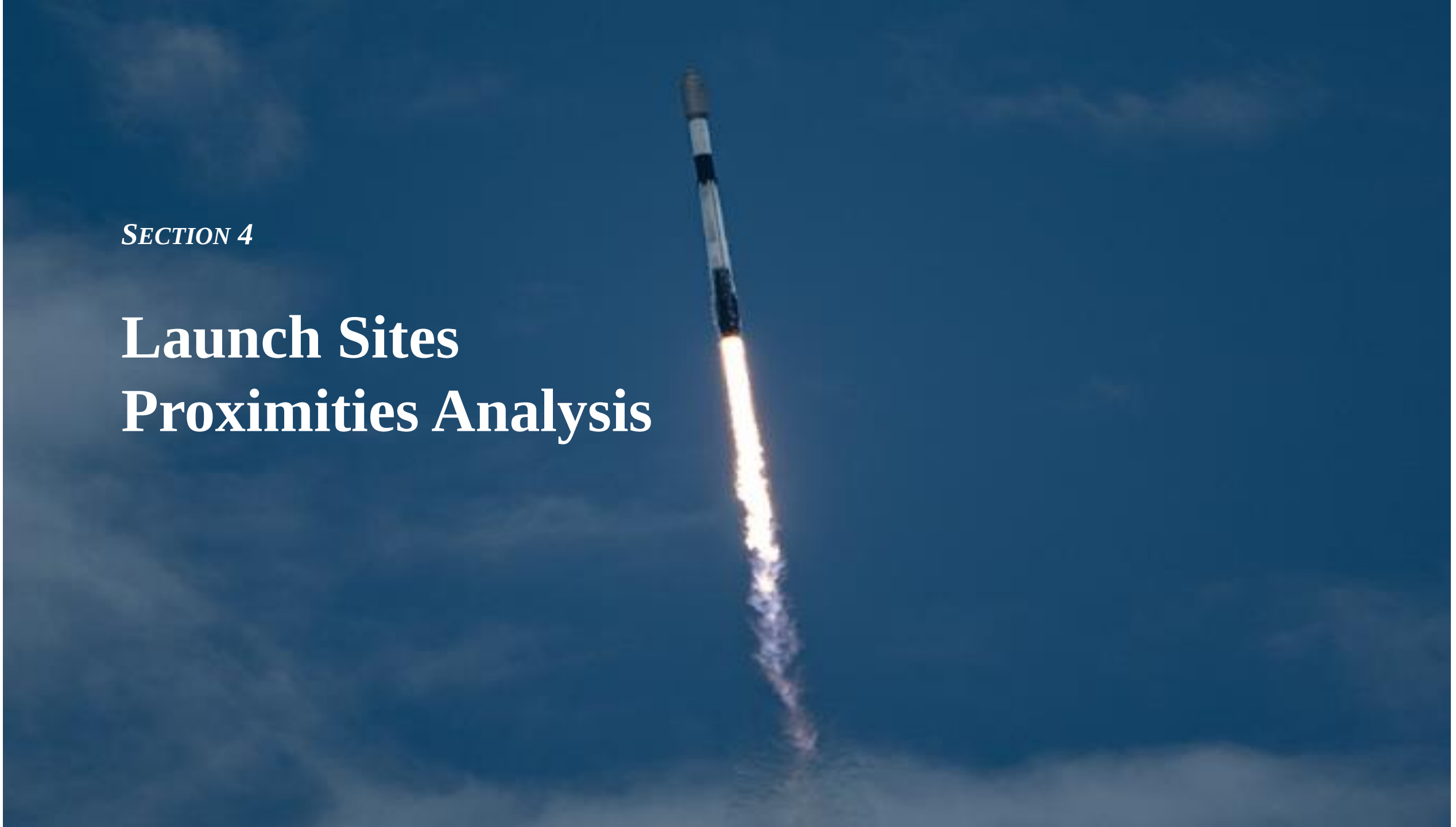
Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

[31]:
```
%sql SELECT [Landing_Outcome], count(*) as count_outcomes \
FROM SPACEXTBL \
WHERE DATE between '04-06-2010' and '20-03-2017' group by [Landing_Outcome] order by count_outcomes DES
```

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

[31]:

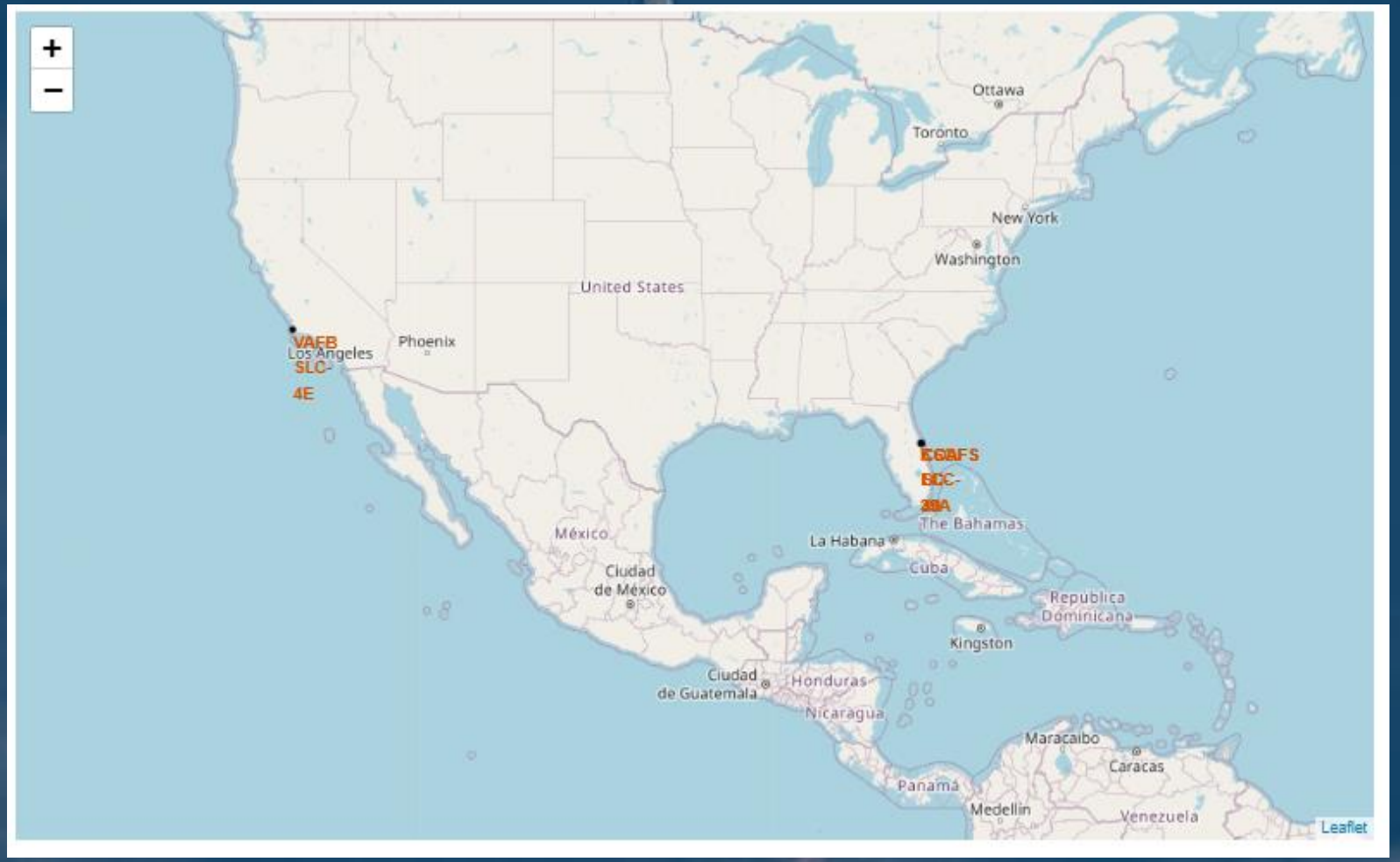
Landing_Outcome	count_outcomes
Success	21
No attempt	10
Success (drone ship)	8
Success (ground pad)	6
Failure (drone ship)	5
Failure	3
Controlled (ocean)	3
Failure (parachute)	2
No attempt	1



SECTION 4

Launch Sites Proximities Analysis

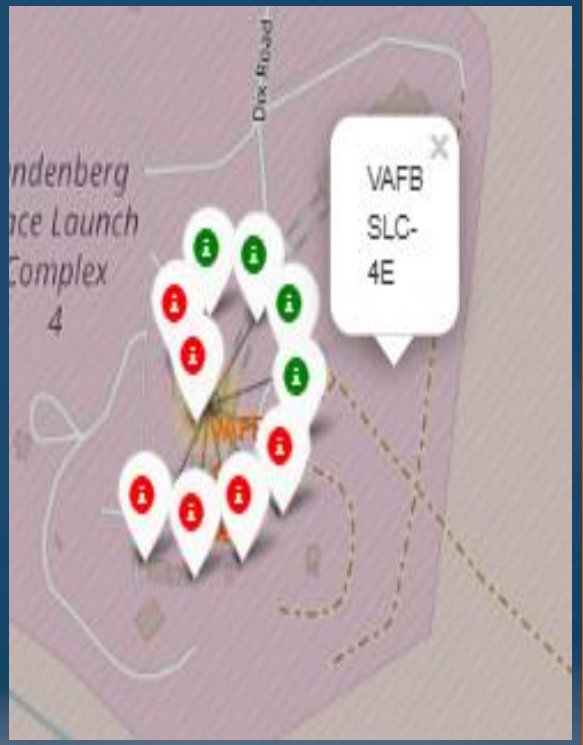
All launch sites global map markers



All launch sites global map markers

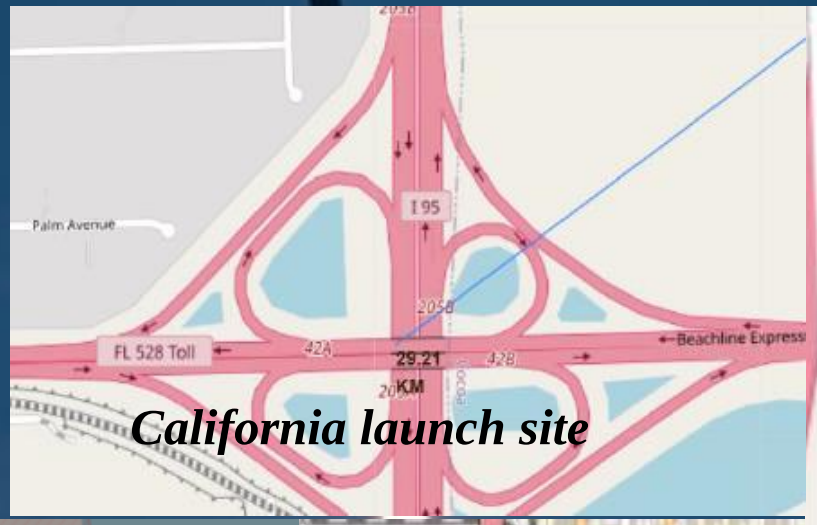
Florida launch site with green indicating successful while red indicating unsuccessful launch

California launch site

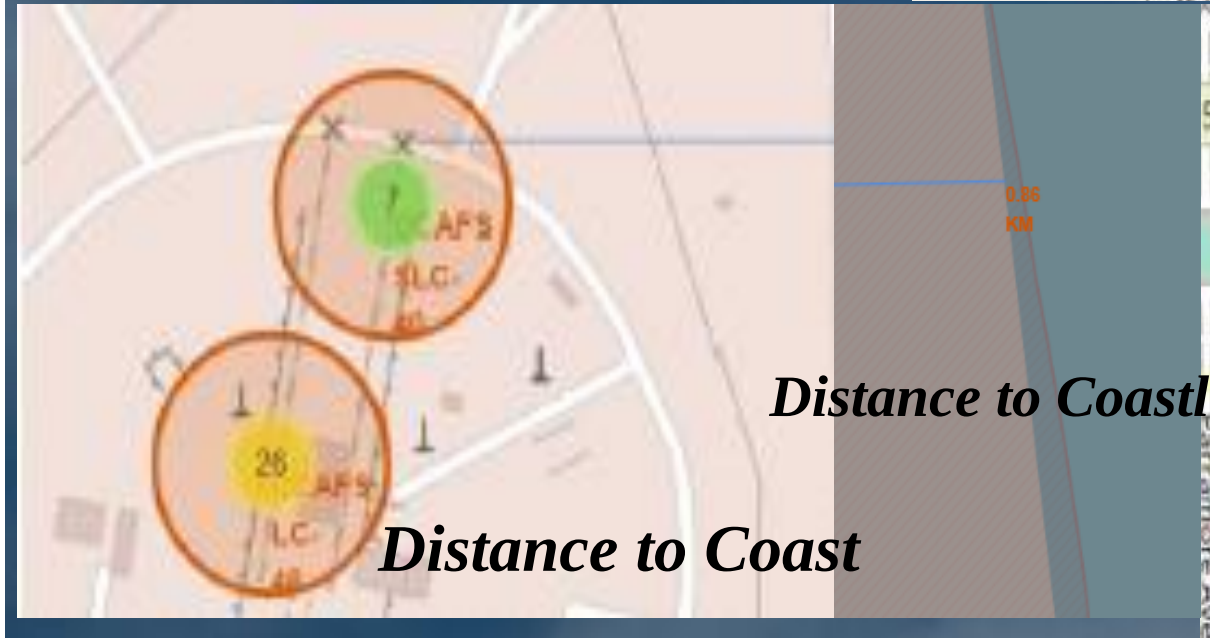


Launch Site distance to landmarks

- *Are the launch sites in close proximity to railway? NO*
- *Are the launch sites in close proximity to highway? NO*
- *Are the launch sites in close proximity to Coaster line? YES*
- *Do launch sites keep certain distance away from the city? YES*



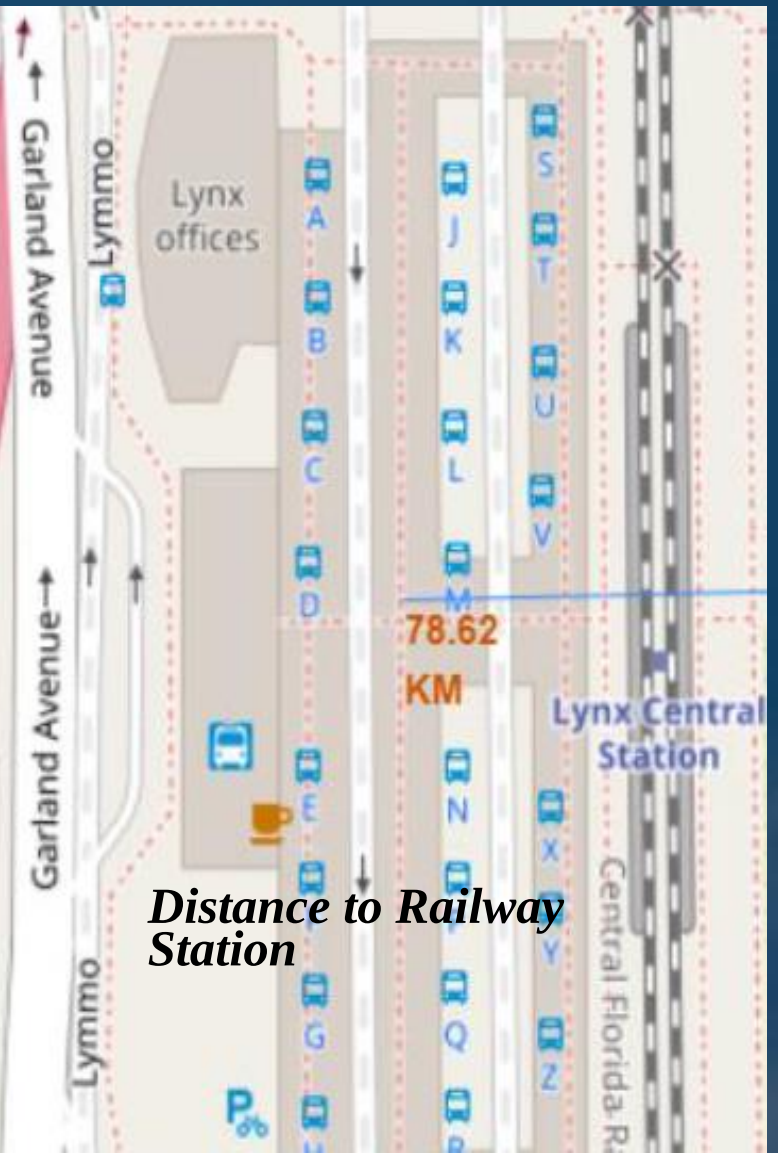
California launch site



Distance to Coast

Distance to Coastline

Distance to City

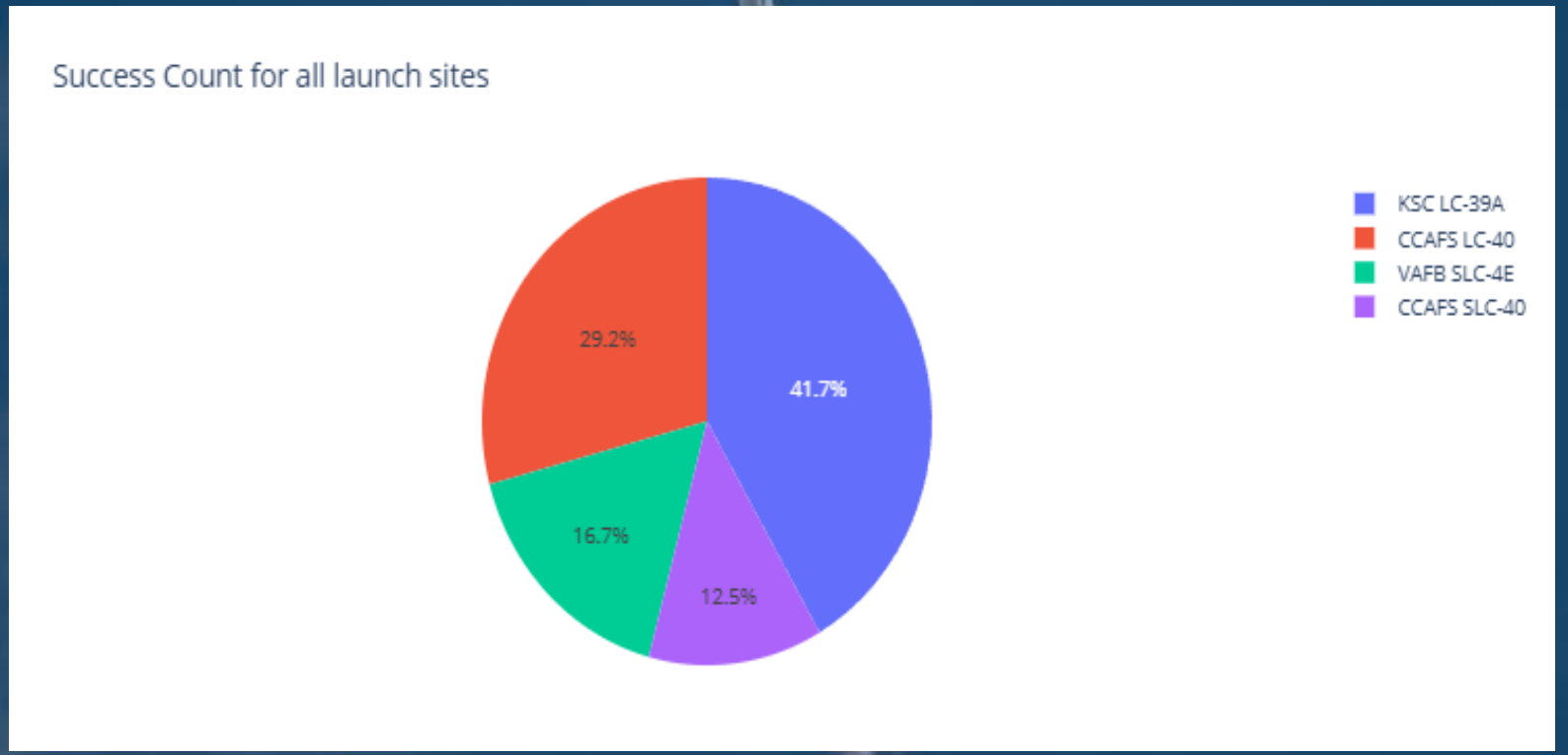


Distance to Railway Station

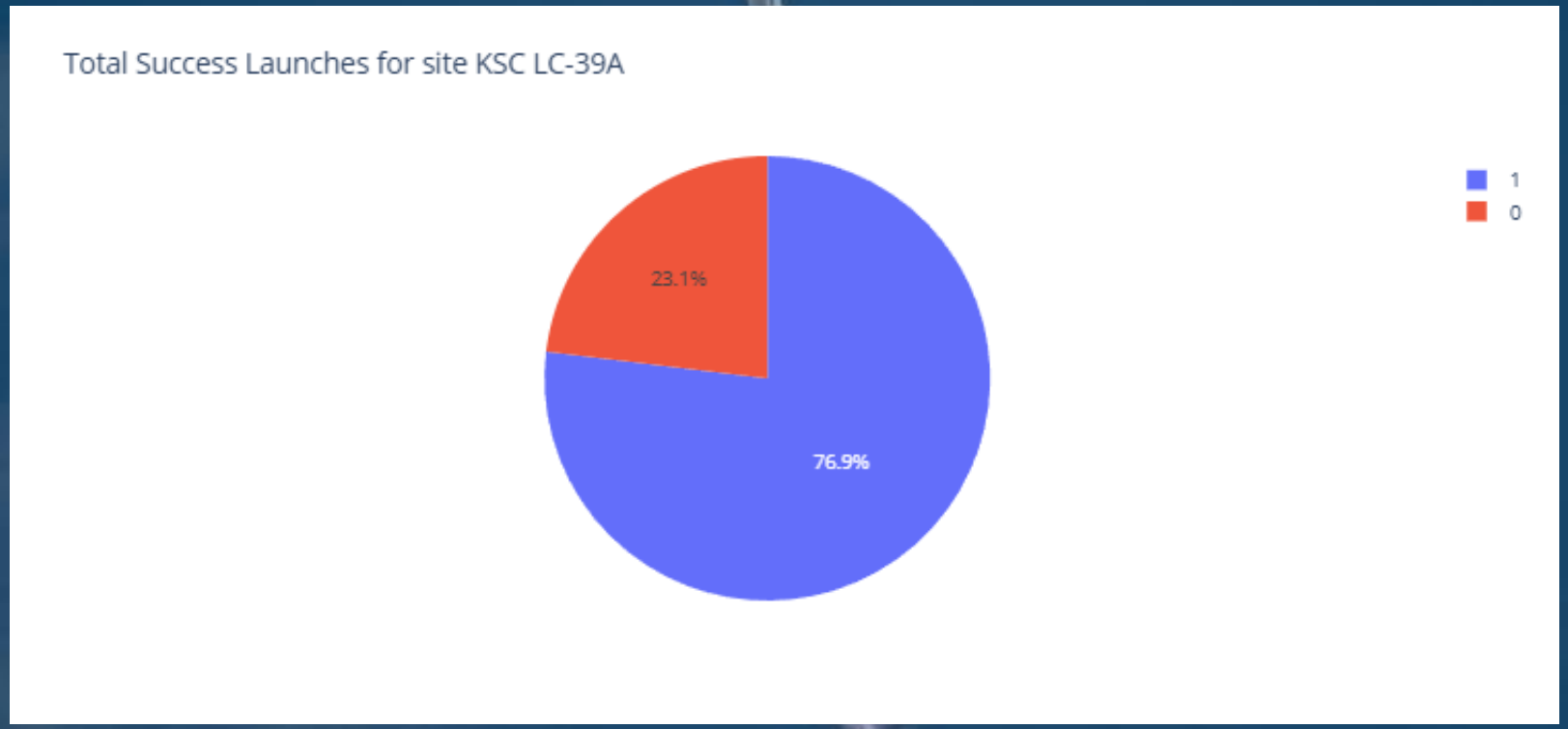
SECTION 5

Build a Dashboad with Plotly Dash

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

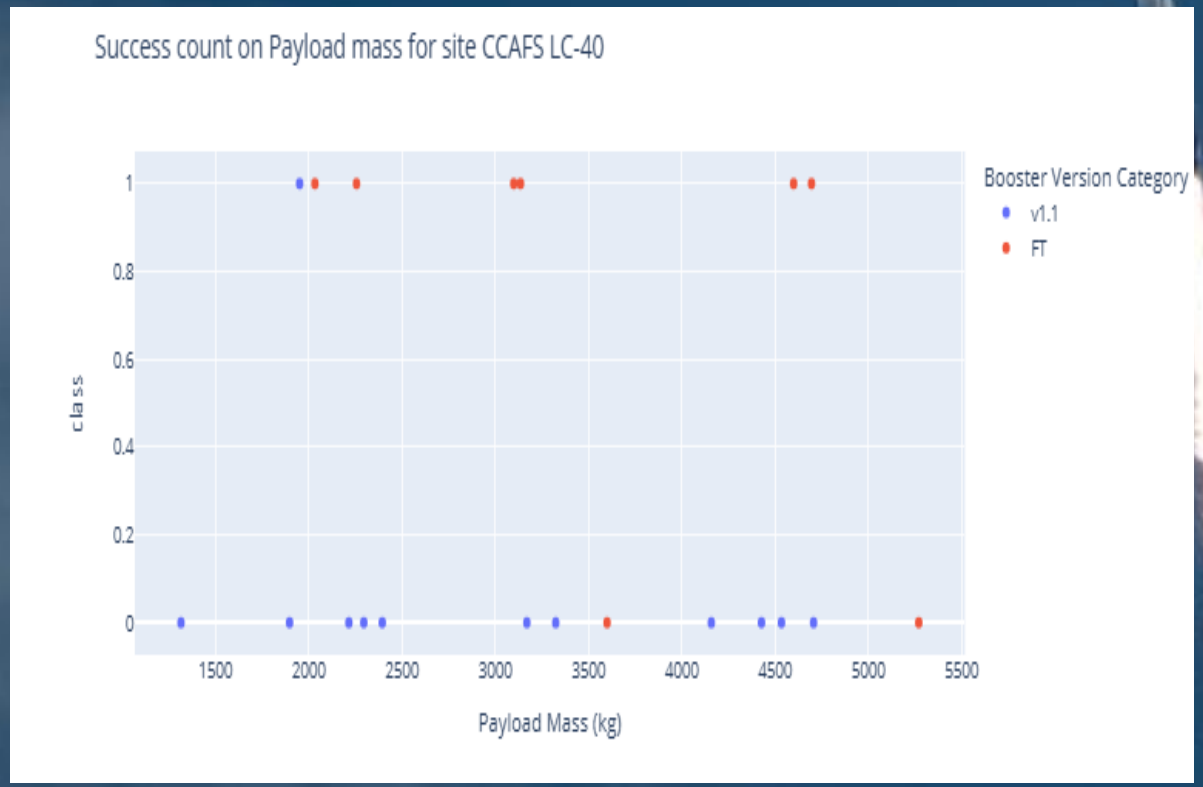


Pie chart showing the Launch site with the highest launch success ratio



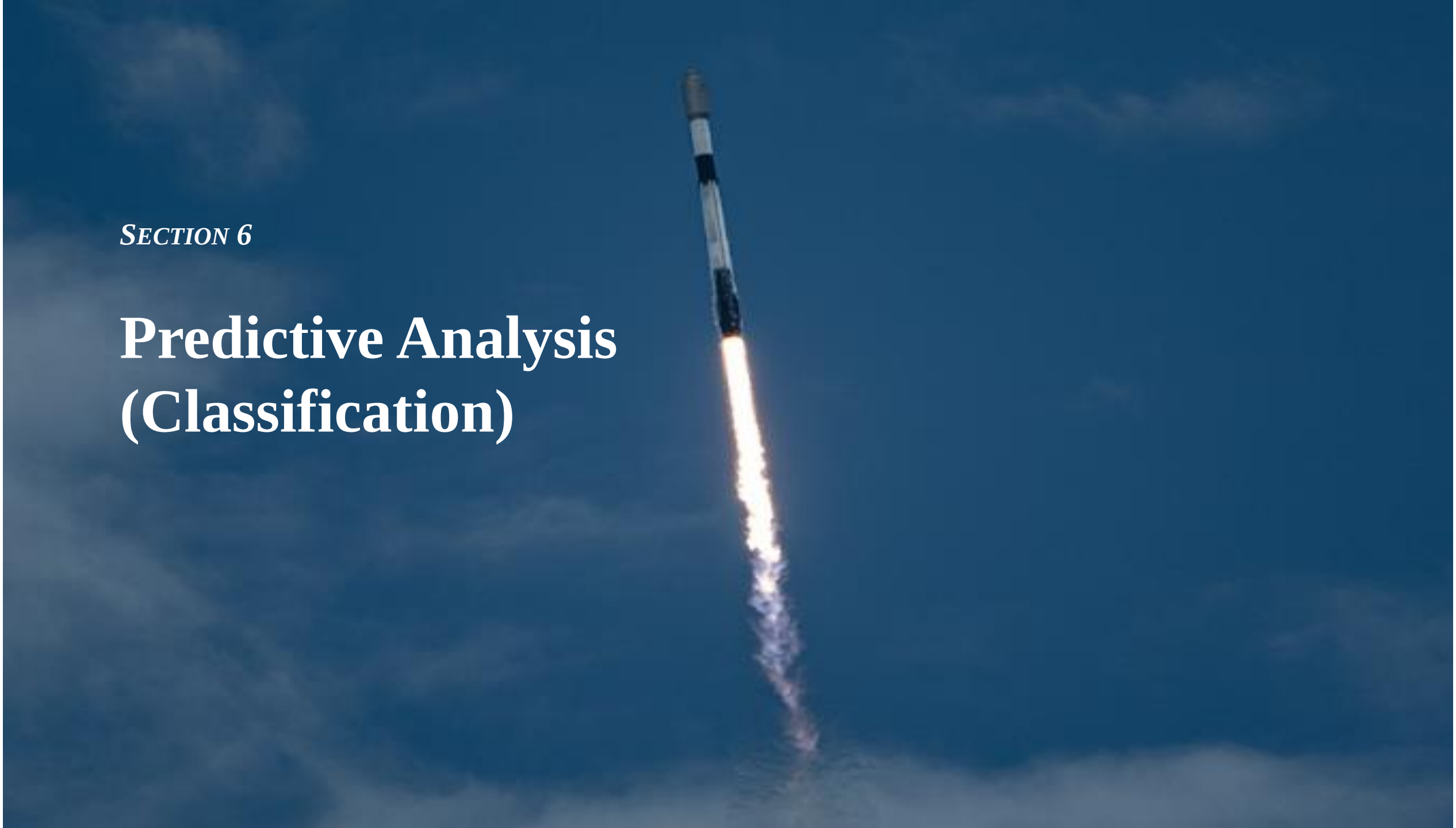
Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider

Low weighted payload 0-4000kg



Heavy weighted payload 4000-1000kg

We can also conclude therefore that the success of the low weighted payloads is higher than the heavy weighted payloads

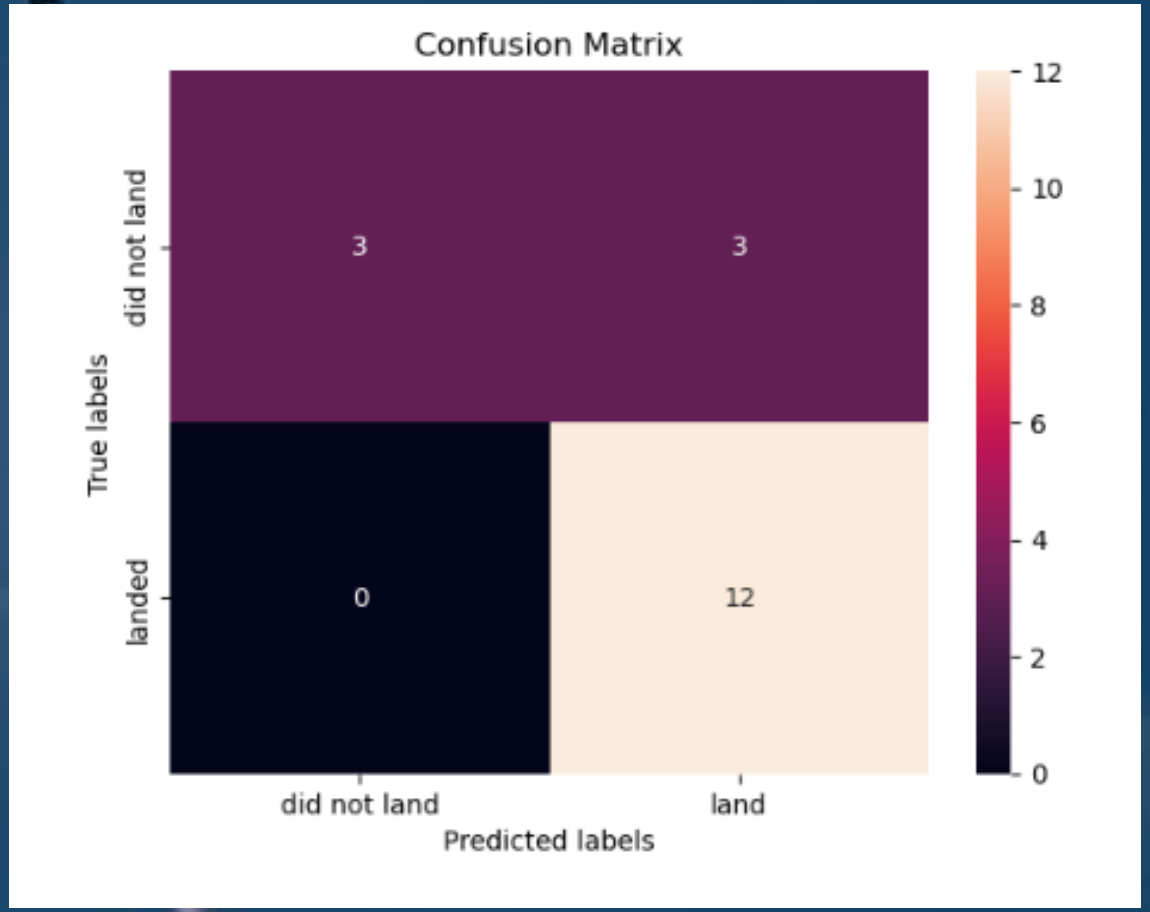


SECTION 6

Predictive Analysis (Classification)

Confusion Matrix

From confusion matrix results of the prediction using KNN, Logistic Regression, SVM and Decision Tree model, the major problem is the false positives .i.e., unsuccessful landing marked as successful landing by the classifier.



Classification Accuracy using Training Data

From the result shown the prediction accuracy

However Decision Tree is the least with Accuracy of 72.2%

```
1  Report = pd.DataFrame({'Method' : ['Test Data Accuracy']})
2
3  knn_accuracy=knn_cv.score(X_test, Y_test)
4  Decision_tree_accuracy=tree_cv.score(X_test, Y_test)
5  SVM_accuracy=svm_cv.score(X_test, Y_test)
6  Logistic_Regression=logreg_cv.score(X_test, Y_test)
7
8  Report['Logistic_Reg'] = [Logistic_Regression]
9  Report['SVM'] = [SVM_accuracy]
10 Report['Decision Tree'] = [Decision_tree_accuracy]
11 Report['KNN'] = [knn_accuracy]
12
13 Report.transpose()
```

	0
Method	Test Data Accuracy
Logistic_Reg	0.833333
SVM	0.833333
Decision Tree	0.722222
KNN	0.833333

Conclusion

By way of conclusion we can conclude that:

1. The larger the flight amount at a launch site, the greater the success rate at a launch site.
2. Launch success rate started to increase in 2013 till 2020.
3. The Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
4. The KSC LC-39A had the most successful launches of any sites.
5. The Decision tree classifier seems to be the least of the machine learning algorithm for this task, though the result show it is a good method for this task.


Thank you very much.