Presentation On Space X With Data Science

Amrik Ghosh

02-04-2023

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 Results
  - EDA Results
  - Interactive Analysis
  - Predictive Analysis

Introduction

- Project Background and Context
  - Falcon 9 rocket launches of Space X advertises on its website, with a cost of 62 million dollars while other providers cost upward of 165 million dollars each. Space X saves more since it can reuse the first stage.
- Problems of this Project
  - This project task is for predicting the first stage of Falcon 9 rocket will land Successfully.

Part 1

Methodology

Methodology

- Executive Summary
- Data Collection Methodology:
  - SpaceX Rest API
  - Web scrapping from Wikipedia
- Perform data wrangling
  - One hot Encoding data fields for Machine Learning and data cleaning of null values and irrelevant columns.
- Perform EDA using SQL & Visualization
- Using Folium and Plotly Dash performing visual analytics.
- Predictive Analysis using Classification Models
  - LR, KNN, SVM, DT models

Data Collection

- The Following datasets was collected:
  - SpaceX launch data from REST API
  - Rocket Used, payload delivered, launch specifications, landing specifications & landing outcome.
  - api.spacexdata.com/v4/ URL used.
  - Web Scrapping Wikipedia using BeautifulSoup.

Data Collection – SpaceX API

Now let's start requesting rocket launch data from SpaceX API with the following URL:

In [6]:
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

In [7]:
```
response = requests.get(spacex_url)
```

Task 2: Filter the dataframe to only include `Falcon 9` launches

Finally we will remove the Falcon 1 launches keeping only the Falcon 9 launches. Filter the data dataframe using the `BoosterVersion` column to only keep the Falcon 9 launches. Save the filtered data to a new dataframe called `data_falcon9`.

In [23]:
```
# Hint data['BoosterVersion']!='Falcon 1'
filt =  df['BoosterVersion']!=  'Falcon 1'
data_falcon9 = df.loc[filt]
data_falcon9.head()
```

Out[23]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitu
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003	-80.577366	28.561
5	8	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0005	-80.577366	28.561
6	10	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0007	-80.577366	28.561
7	11	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None	1.0	0	B1003	-120.610829	34.632
8	12	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B1004	-80.577366	28.561

https://github.com/amrik-mark42/Final-Presentation/blob/main/1.%20jupyter-labs-spacex-data-collection-api.ipynb

Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

In [8]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

We should see that the request was successfull with the 200 status response code

In [9]:
```
response.status_code
```

Out[9]: 200

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

In [10]:
```
# Use json_normalize meethod to convert the json result into a dataframe
respjson = response.json()
data = pd.json_normalize(respjson)
```

Using the dataframe `data` print the first 5 rows

1]:
```
# Get the head of the dataframe
data.head()
```

1]:

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships	capsules	payloads	
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 33, 'altitude': None, 'reason': 'merlin	Engine failure at 33 seconds and loss of	[]	[]	[]	[5eb0e4b5b6c3bb0006eeb1e1]	5e9e45

Task 3: Dealing with Missing Values

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace `np.nan` values in the data with the mean you calculated.

In [26]:
```
# Calculate the mean value of PayloadMass column
plm_mean = data_falcon9['PayloadMass'].mean()

# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].replace(np.nan, plm_mean, inplace=True)
```

```
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/pandas/core/generic.py:6619: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return self._update_inplace(result)
```

You should see the number of missing values of the `PayLoadMass` change to zero.

Now we should have no missing values in our dataset except for in `LandingPad`

Data Collection - Scraping

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

In [22]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
import requests
import pandas as pd

# Define the URL of the Falcon 9 Launch Wikipedia page
url = 'https://en.wikipedia.org/wiki/Falcon_9'

# Send an HTTP GET request to the URL and store the response in a variable
response = requests.get(url)

# Check the status code of the response to ensure the request was successful
if response.status_code == 200:
    print('Request successful')
else:
    print('Request failed')

# Convert the SpaceX API response to a DataFrame using pd.json_normalize()
df = pd.json_normalize(response.json())

# Check the year in the first row of the static_fire_date_utc column
first_row_year = pd.to_datetime(df['static_fire_date_utc'].iloc[0]).year
print(first_row_year)
```

```
Request successful
```

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

soup = BeautifulSoup(response.content, "html.parser")

print(soup.title.string)
```

```
List of Falcon 9 and Falcon Heavy launches - Wikipedia
```

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

In [11]:
```
# 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")

print(len(html_tables))
```

```
18
```

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

In [12]:
```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

TASK 3: Create a data frame by parsing the launch HTML tables

We will create an empty dictionary with keys from the extracted column names in the previous task. Later, this dictionary will be converted into a Pandas dataframe

In [16]:
```
launch_dict= dict.fromkeys(column_names)

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

# Let's initial the launch_dict with each value to be an empty list
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']=[]
```

https://github.com/amrik-mark42/Final-Presentation/blob/main/2.%20jupyter-labs-webscraping.ipynb

Data Wrangling

TASK 1: Calculate the number of launches on each site

The data contains several Space X launch facilities: Cape Canaveral Space Launch Complex 40 **VAFB SLC 4E** , Vandenberg Air Force Base Space Launch Complex 4E **(SLC-4E)**, Kennedy Space Center Launch Complex 39A **KSC LC 39A** .The location of each Launch Is placed in the column `LaunchSite`

Next, let's see the number of launches for each site.

Use the method `value_counts()` on the column `LaunchSite` to determine the number of launches on each site:

In [7]:

```python
# Apply value_counts() on column LaunchSite
launch_counts = df["LaunchSite"].value_counts()

# Print the launch counts
print(launch_counts)
```

```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

Each launch aims to an dedicated orbit, and here are some common orbit types:

TASK 3: Calculate the number and occurence of mission outcome per orbit type

Use the method `.value_counts()` on the column `Outcome` to determine the number of `landing_outcomes` .Then assign it to a variable landing_outcomes.

In [9]:

```python
# landing_outcomes = values on Outcome column
orbit_counts = df["Orbit"].value_counts()
landing_outcomes = df.groupby("Orbit")["Outcome"].value_counts()

# Print the orbit and mission outcome counts
print("Orbit and Mission Outcome Counts:")
print("--------------------")
for orbit in orbit_counts.index:
    print(f"Orbit: {orbit}")
    print(landing_outcomes[orbit])
    print("--------------------")
```

```
Orbit and Mission Outcome Counts:
--------------------
Orbit: GTO
Outcome
True ASDS     13
None None     11
False ASDS     1
None ASDS      1
True Ocean     1
Name: Outcome, dtype: int64
--------------------
Orbit: ISS
Outcome
True RTLS      7
True ASDS      5
None None      3
False ASDS     2
False Ocean    1
False RTLS     1
None ASDS      1
True Ocean     1
Name: Outcome, dtype: int64
--------------------
Orbit: VLEO
Outcome
```

TASK 2: Calculate the number and occurrence of each orbit

Use the method `.value_counts()` to determine the number and occurrence of each orbit in the column `Orbit`

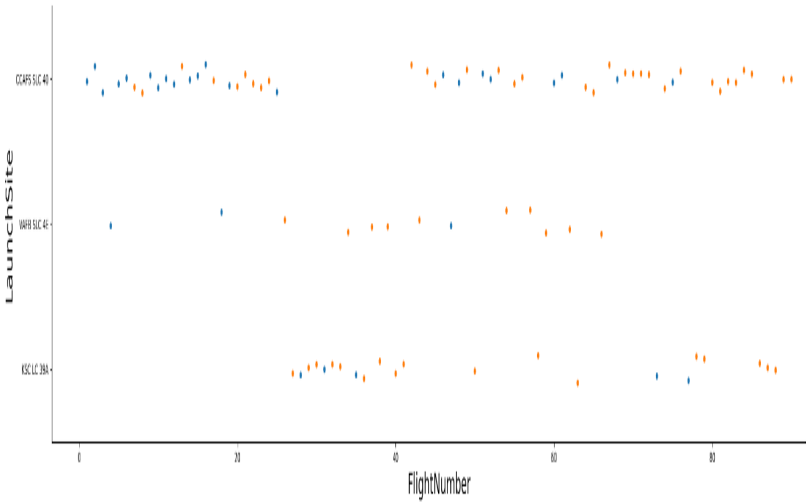
In [8]:

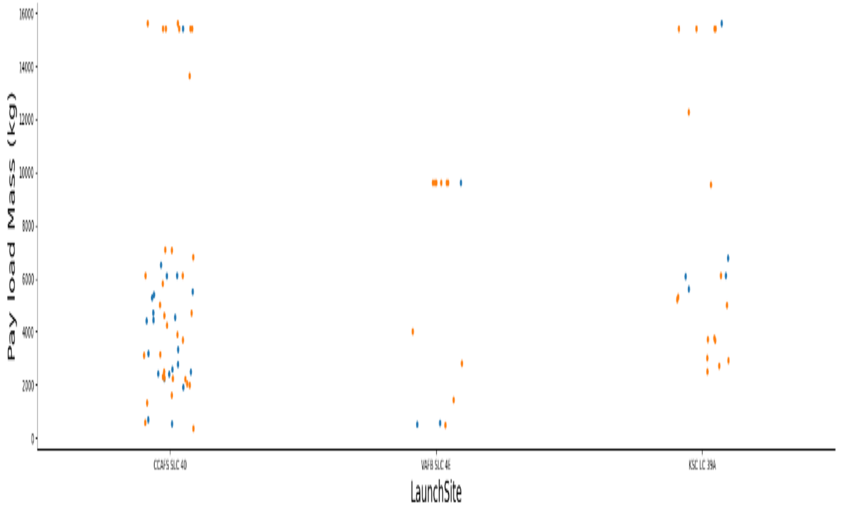
```python
# Apply value_counts on Orbit column
orbit_counts = df["Orbit"].value_counts()

# Print the orbit counts
print(orbit_counts)
```

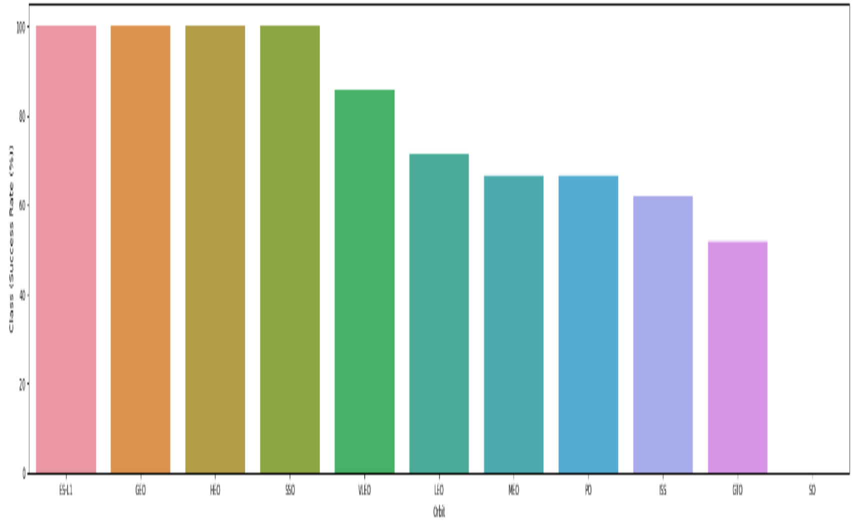
```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
Name: Orbit, dtype: int64
```

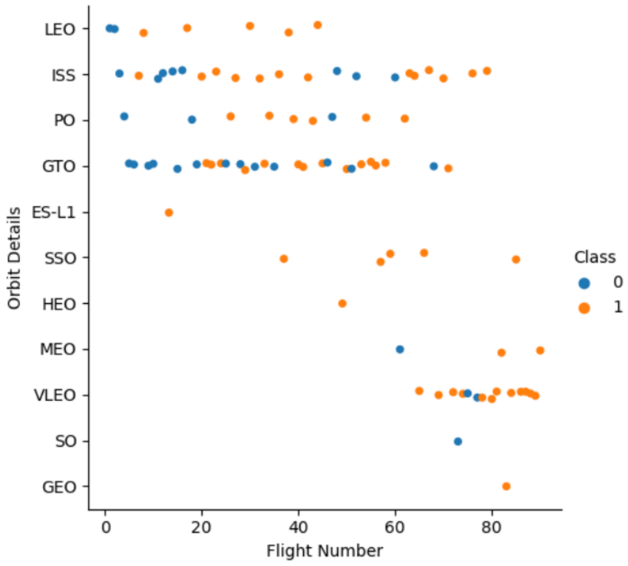
https://github.com/amrik-mark42/Final-Presentation/blob/main/3.%20jupyter-spacex-data_wrangling_jupyterlite.jupyterlite.ipynb

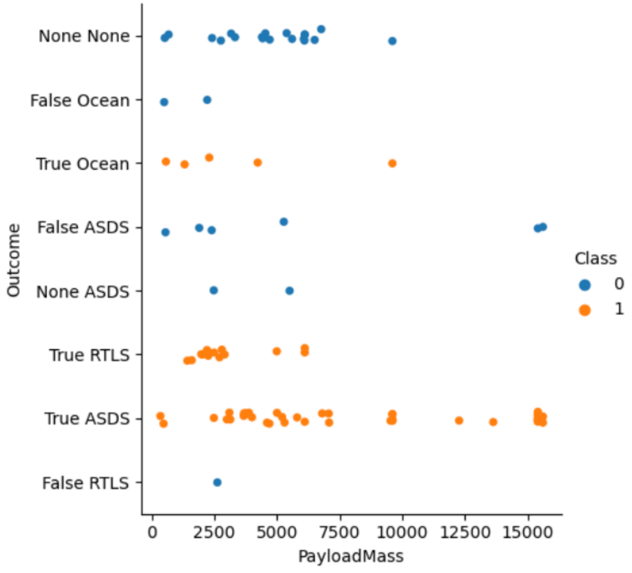
EDA with Data Visualization

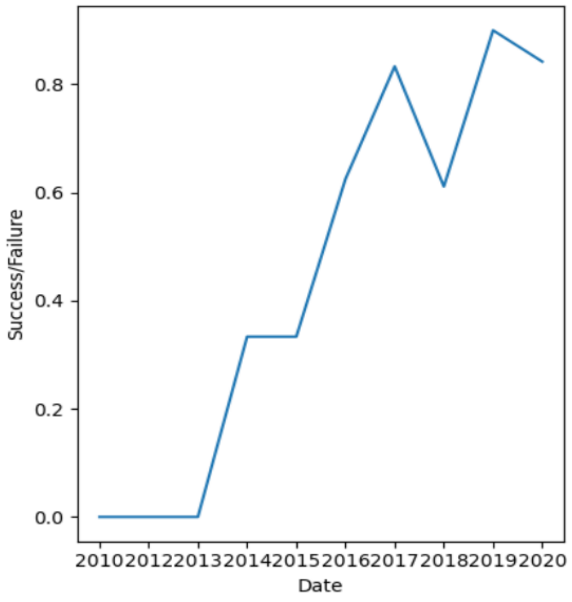












https://github.com/amrik-mark42/Final-Presentation/blob/main/5.%20jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

- SQL queries performed include:
    - Displaying the names of the unique launch sites in the space mission
    - Displaying 5 records where launch sites begin with the string ‘ KSC ‘
    - Displaying the total payload mass carried by boosters launched by NASA.
    - Displaying average payload mass carried by booster version F9 v1.1
    - Listing the date where the successful landing outcome in drone ship was achieved.
    - Listing the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000.
    - Listing the total number of successful and failure mission outcomes.
    - Listing the names of the booster_versions which have carried maximum payload mass.
    - Listing the records which will display the month names, successful landing outcomes in ground pad, booster versions, launch_site for the months in year 2017
    - Ranking the count of successful landing outcomes in descending order.

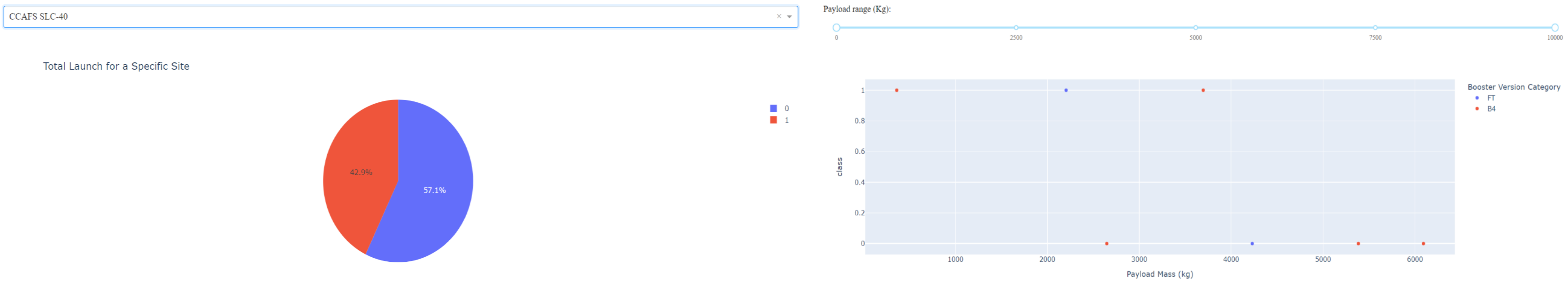
https://github.com/amrik-mark42/Final-Presentation/blob/main/4.%20jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

Map Markers have been added to the map with aim to finding an optimal location for building a launch site

https://github.com/amrik-mark42/Final-Presentation/blob/main/6.%20lab_jupyter_launch_site_location.jupyterlite.ipynb

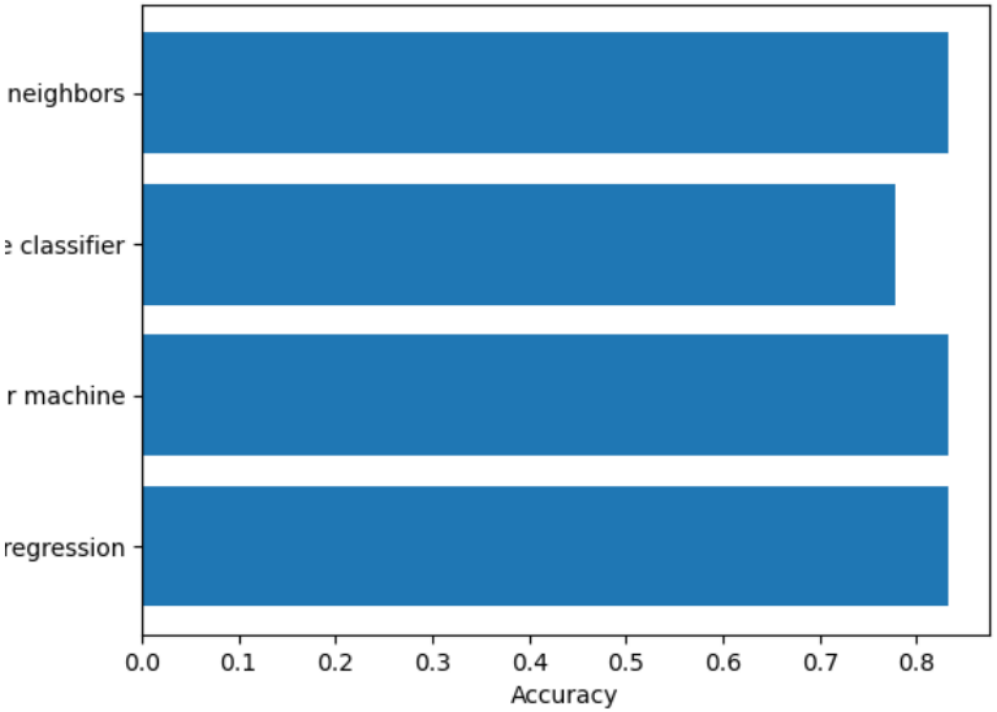
Build a Dashboard with Plotly Dash

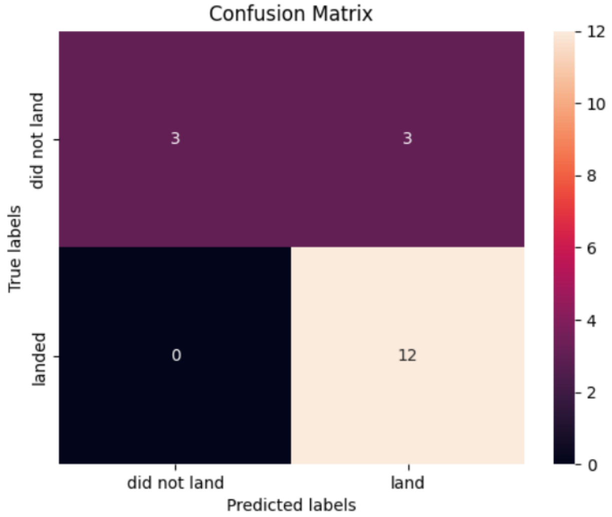


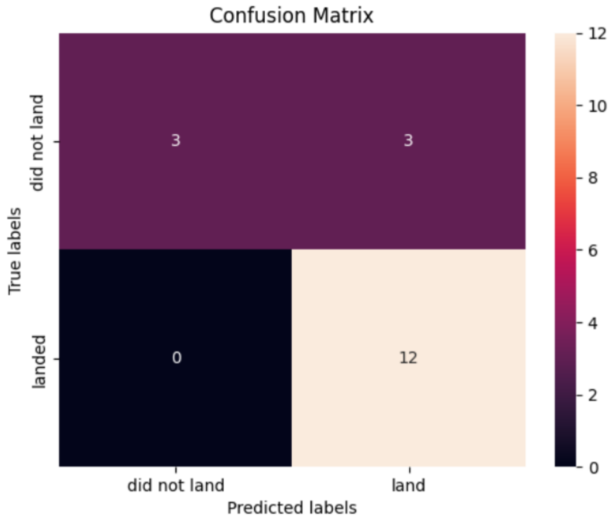
https://github.com/amrik-mark42/Final-Presentation/blob/main/7.%20spacex_dash_app.py

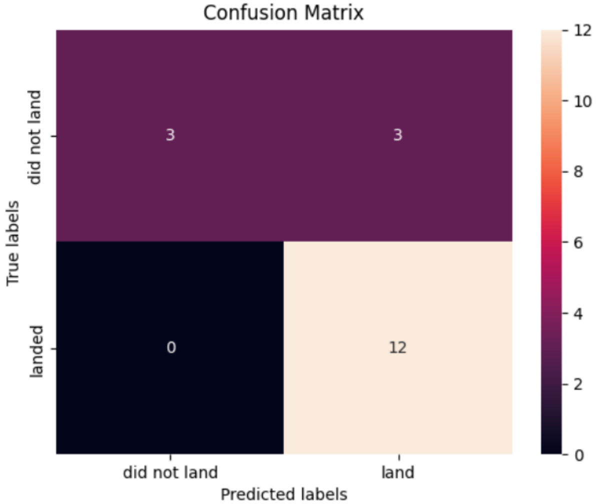
Predictive Analysis (Classification)

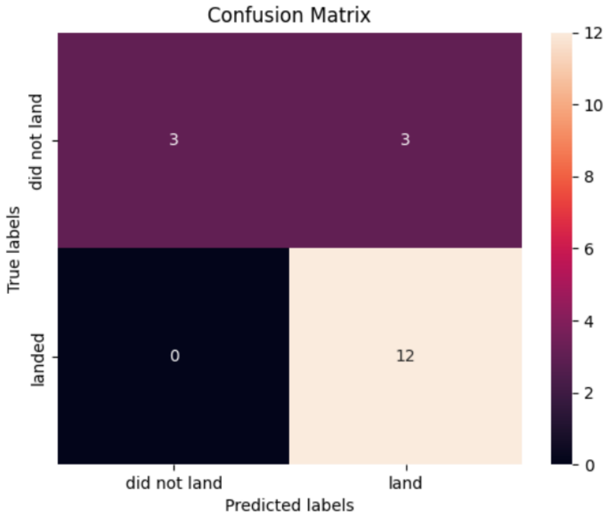
- The SVM, KNN and Logistic Regression model achieved the highest accuracy at 83.3%, while the SVM performs the best in terms of Area Under the Curve at 0.958.











https://github.com/amrik-mark42/Final-Presentation/blob/main/8.%20SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

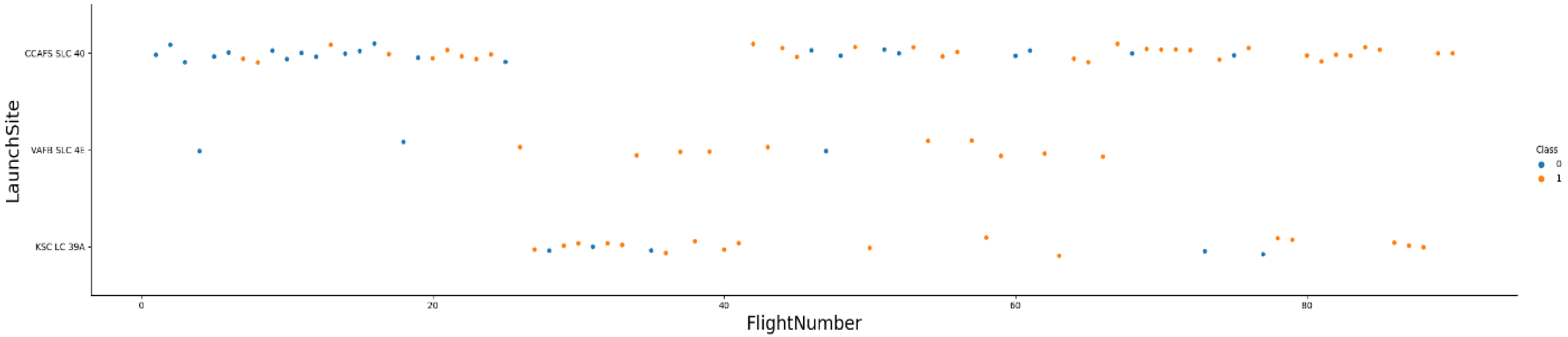
Results

- The SVM, KNN and Logistic Regression models are the best in terms of prediction accuracy for this dataset.
- Low weighted payloads perform better than the heavier payloads.
- The success rates for spaceX launches is directly proportional time in years they will eventually perfect the launches.
- KSC LC 39A had the most successful launches from all the sites.
- Orbit GEO, HEO, SSO, ES L1 has the best Success Rate.

Part 2

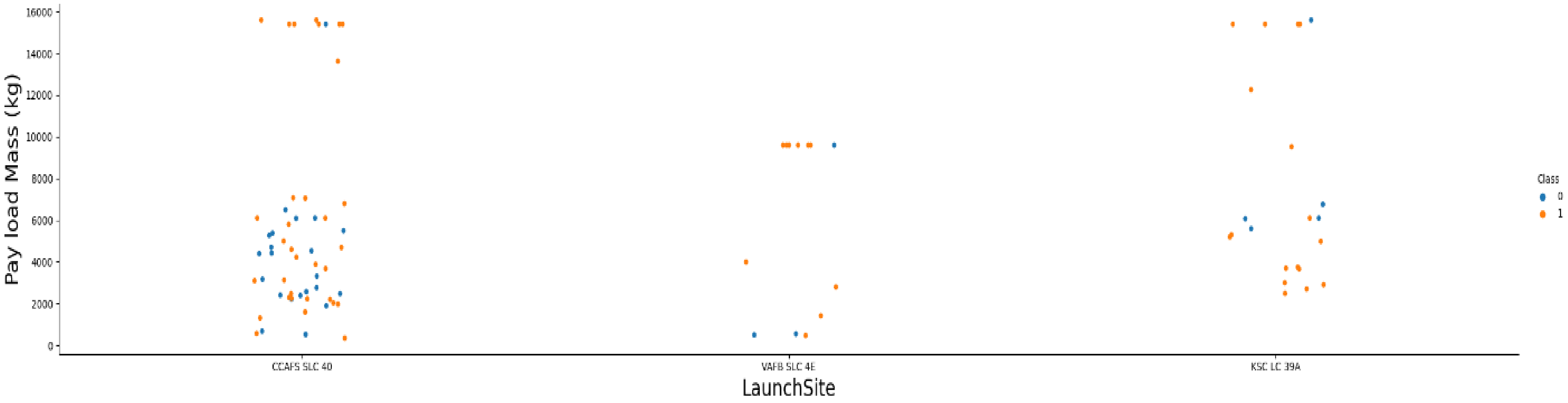
Results From EDA

Flight Number vs. Launch Site



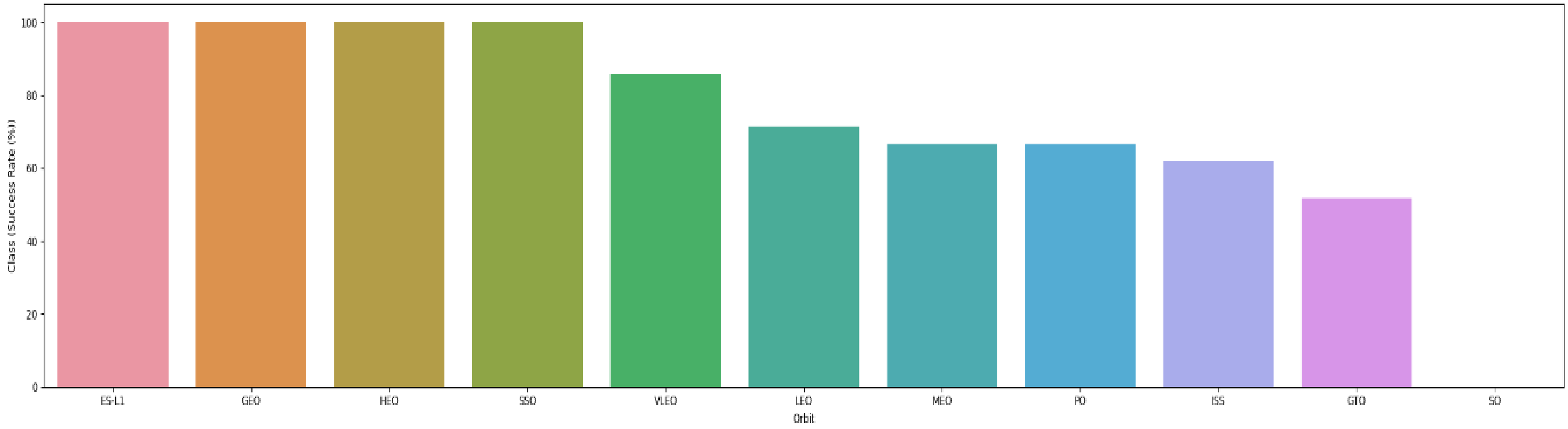
- Launches from the site of CCAFS SLC 40 are significantly higher than launches from other sites.

Payload vs. Launch Site



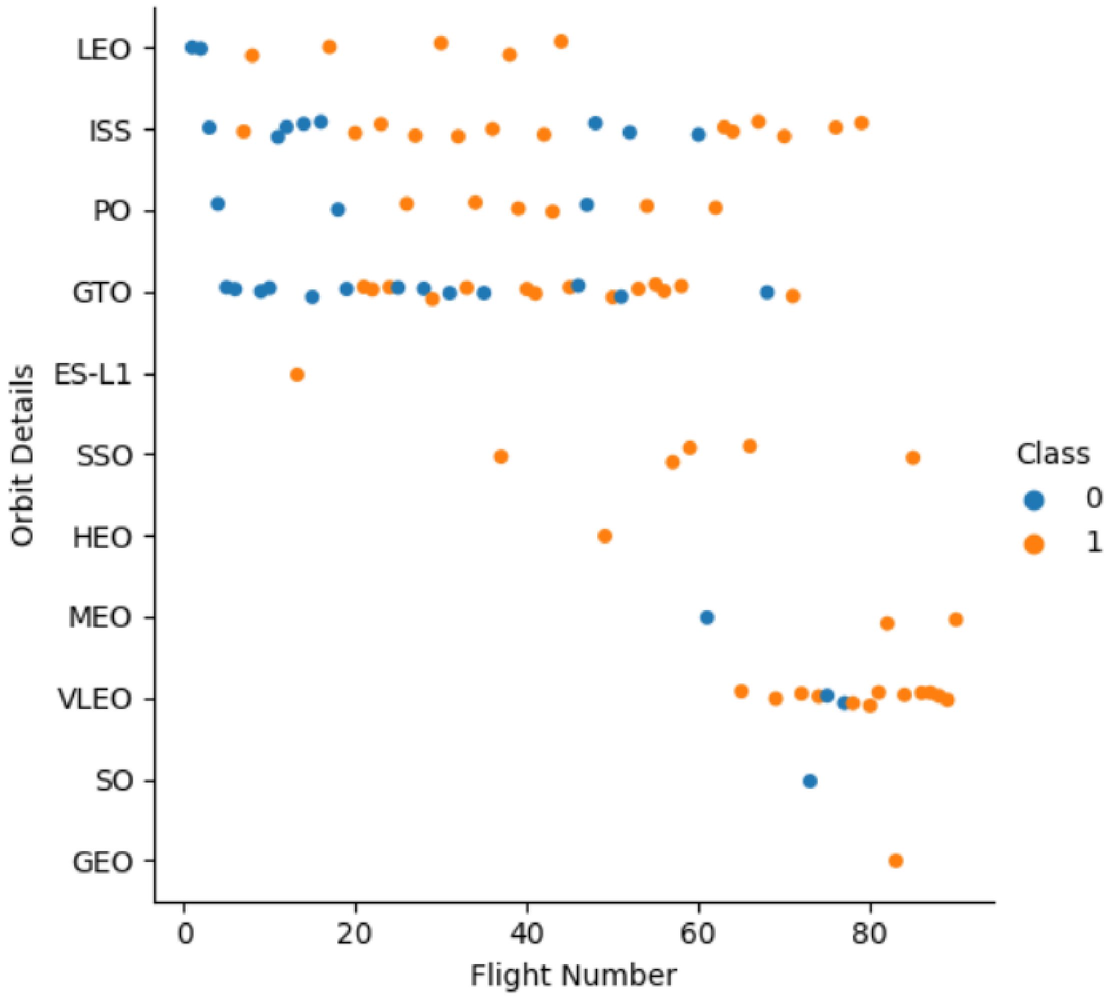
- The majority of Pay Loads with lower Mass have been launched from CCAFS SLC 40.

Success Rate vs. Orbit Type



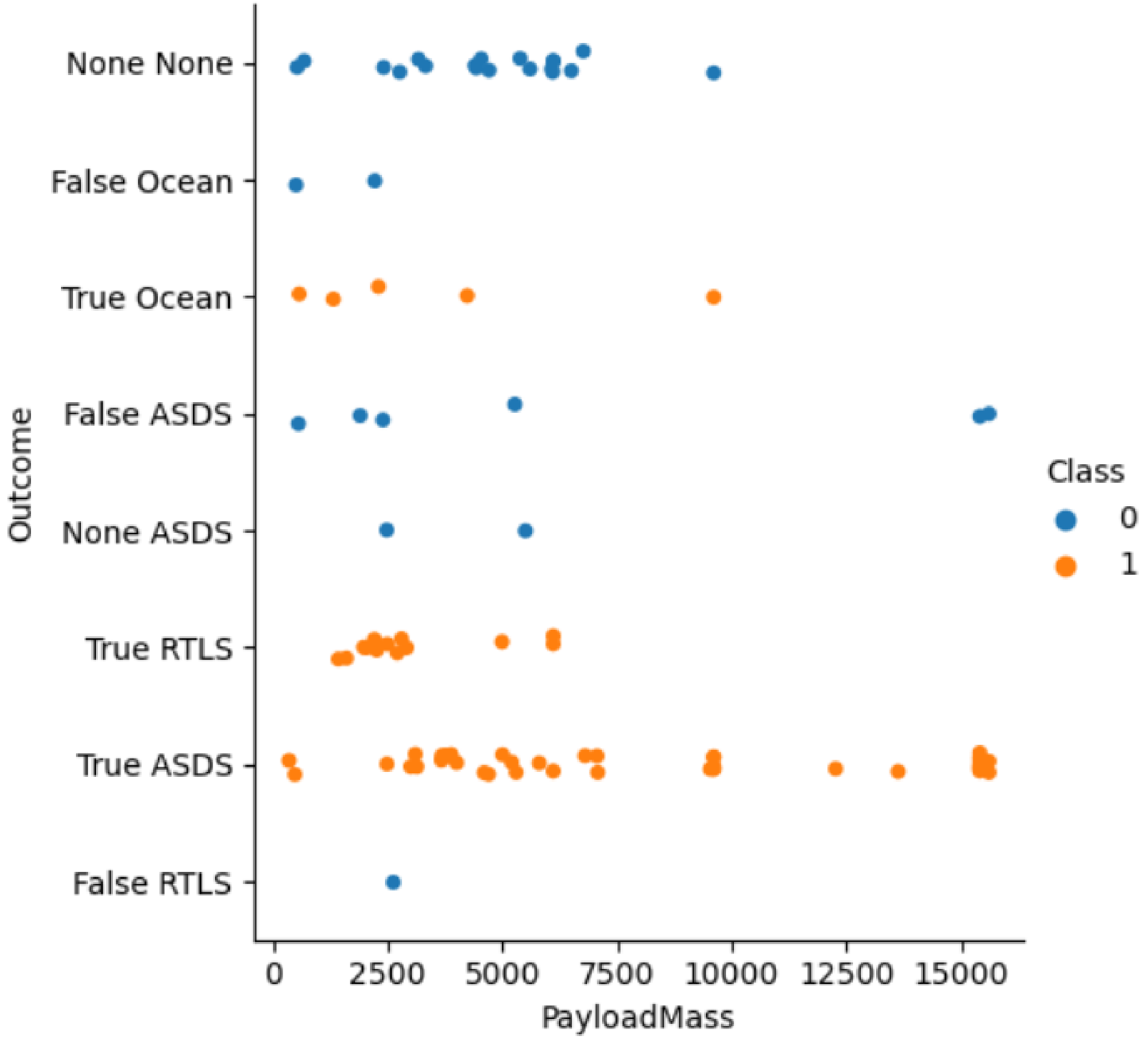
- The orbit types of ES-L1, GEO, HEO, SSO are among the highest success rate.

Flight Number vs. Orbit Type



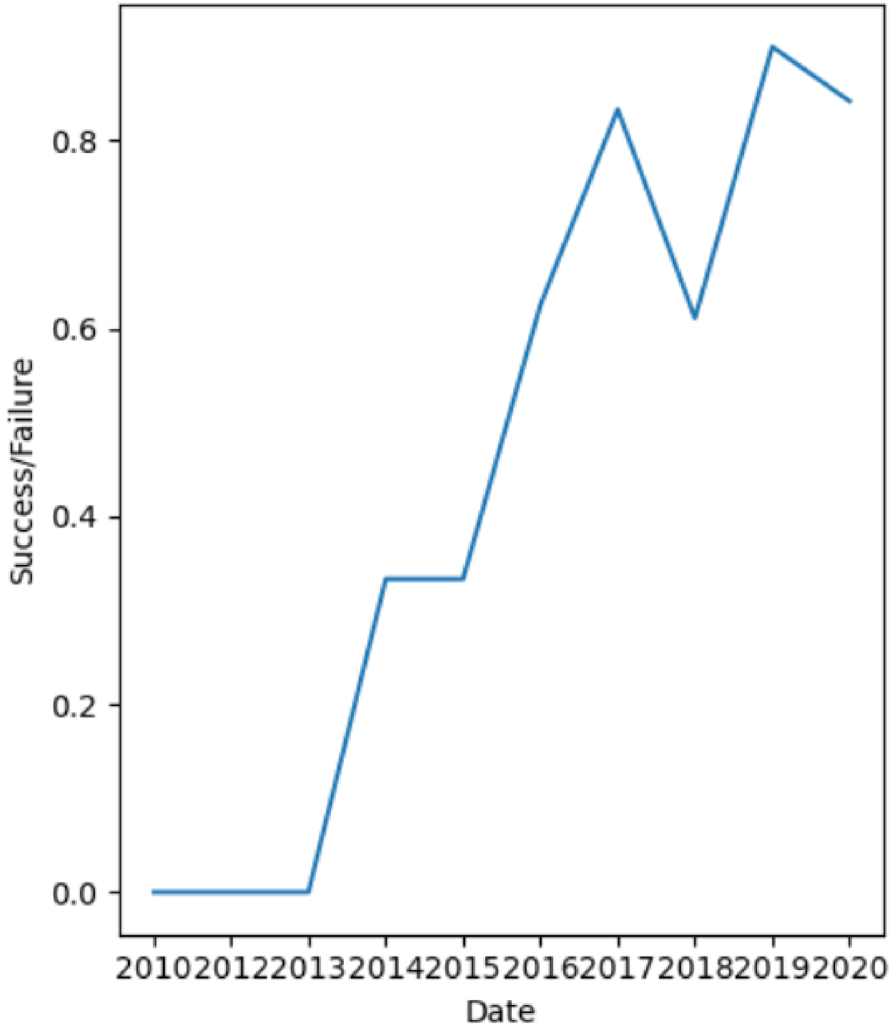
- A trend can be observed of shifting to VLEO launches in recent years.

Payload vs. Outcome



- There are strong relation between Payload Mass and Outcome.

Launch Success Yearly Trend



- Launch Success Rate has increased significantly since 2013 and stabilized since 2019.

Launch Site Names Begin With 'CCA'

In [8]:
```
%sql SELECT LAUNCH_SITE from SPACEXTBL 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

Total Payload Mass

In [9]:
```
%sql select sum(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL;
```

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

Out[9]:

payloadmass
619967

Average Payload Mass by F9 V1.1

In [9]:
```
%sql select sum(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL;
```

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

Out[9]:

payloadmass
619967

First Successful Ground Landing Date

In [11]:
```
%sql select min(DATE) from SPACEXTBL;
```

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

Out[11]:

min(DATE)
01-03-2013

Total Number of Successful & Failure Mission Outcomes

In [13]:

```
%sql select count(MISSION_OUTCOME) as missionoutcomes from SPACEXTBL GROUP BY MISSION_OUTCOME;
```

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

Out[13]:

missionoutcomes
1
98
1
1

Boosters Carried Maximum Payload

In [14]:

```
%sql select BOOSTER_VERSION as boosterversion from SPACEXTBL where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACEXTBL);
```

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

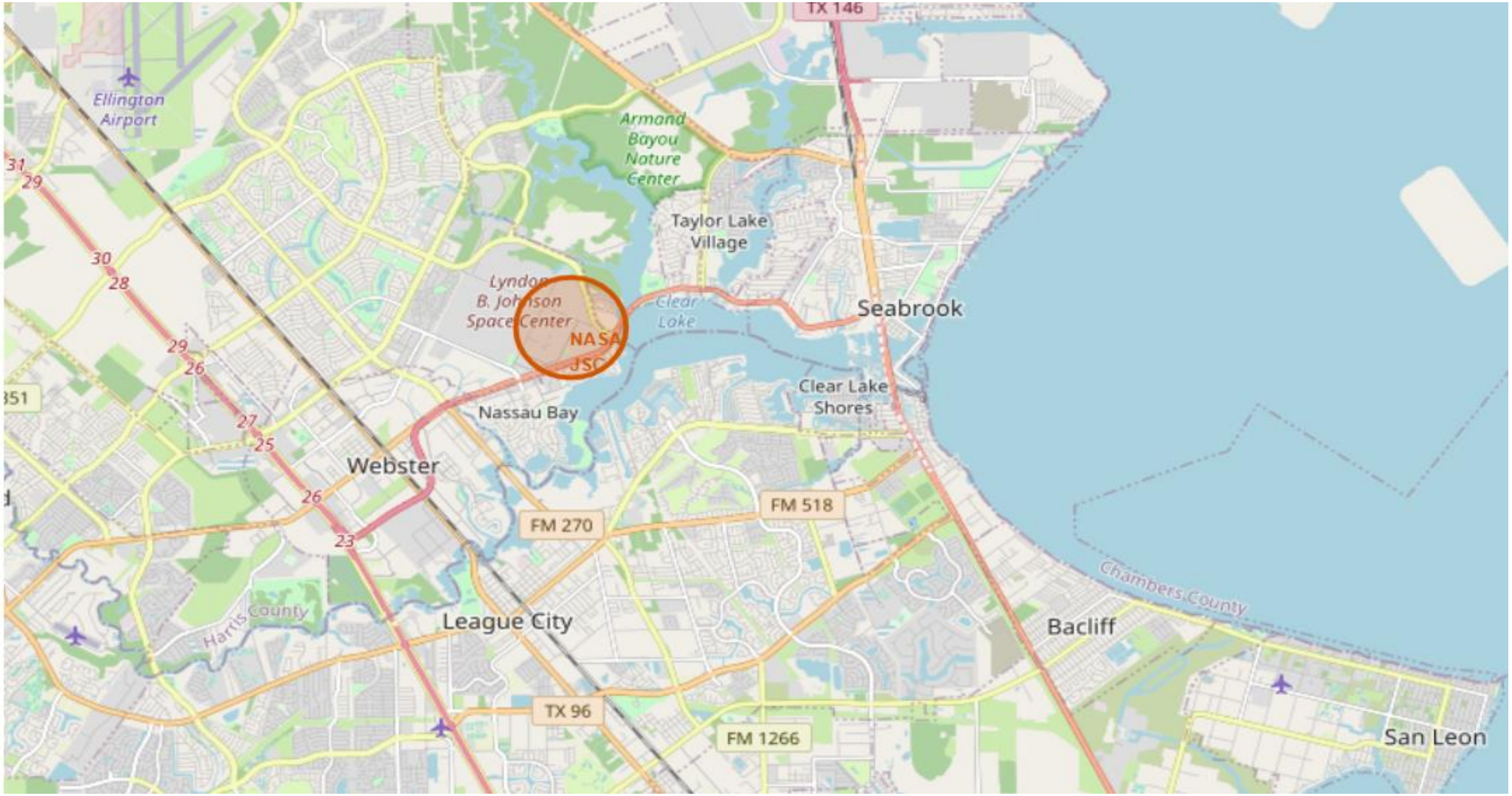
Out[14]:

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

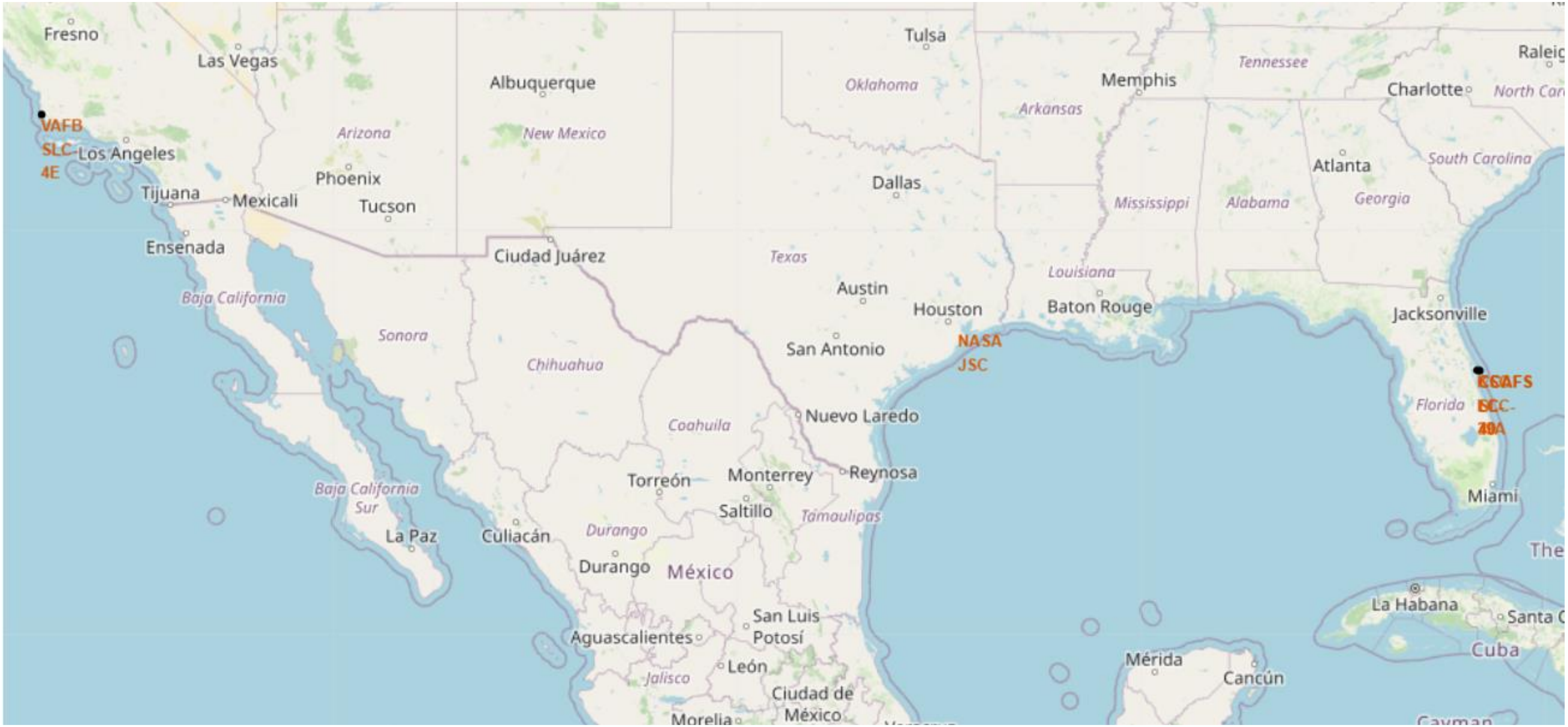
Part 3

Launch Sites Proximities Analysis

All Launch Sites Marked on a Map



Success/ Failed Launches Marked on the Map

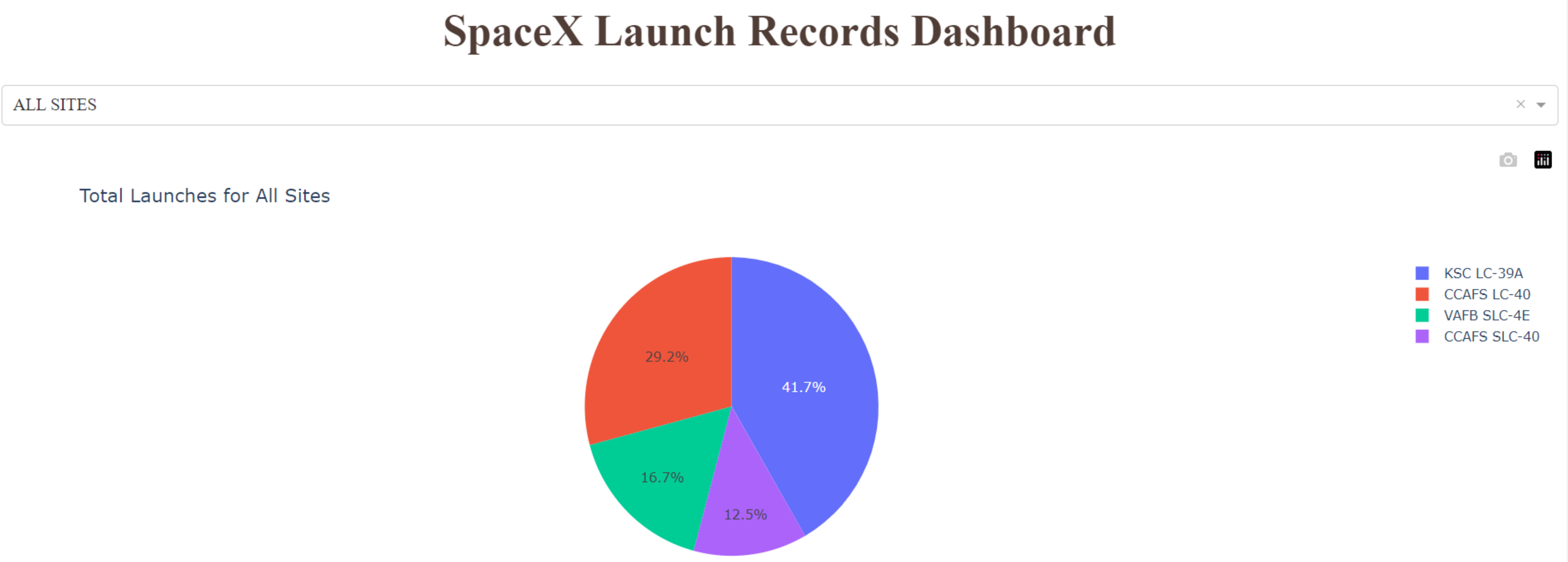


Distances between a Launch Site to its Proximities

Part 4

Build a Dashboard with Plotly Dash

Total Success Launches by all sites



Success Rate by Site

SpaceX Launch Records Dashboard

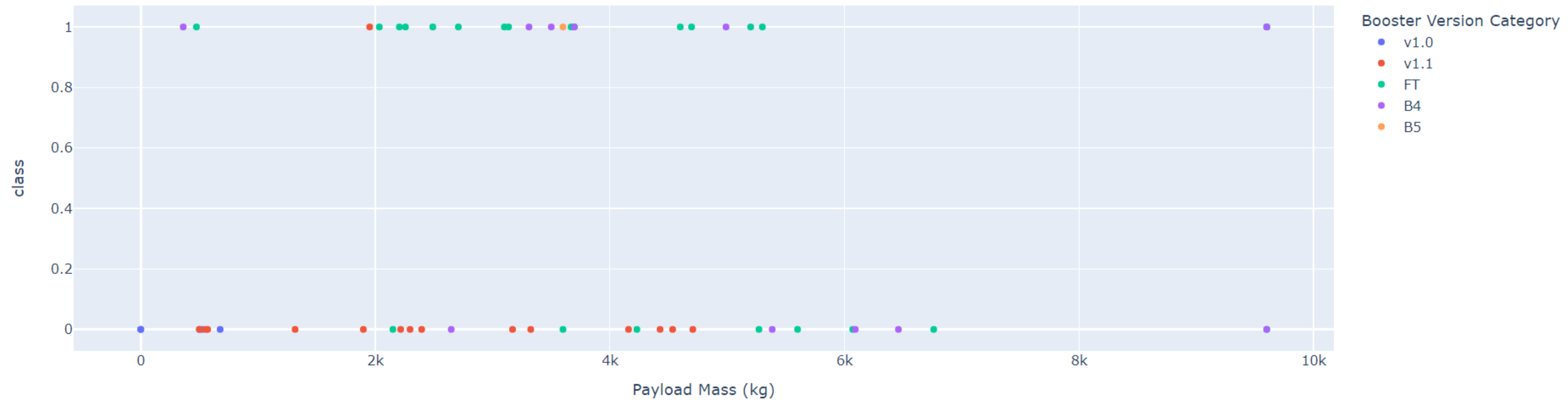
CCAFS LC-40

Total Launch for a Specific Site

Payload vs Launch Outcome

Payload range (Kg):

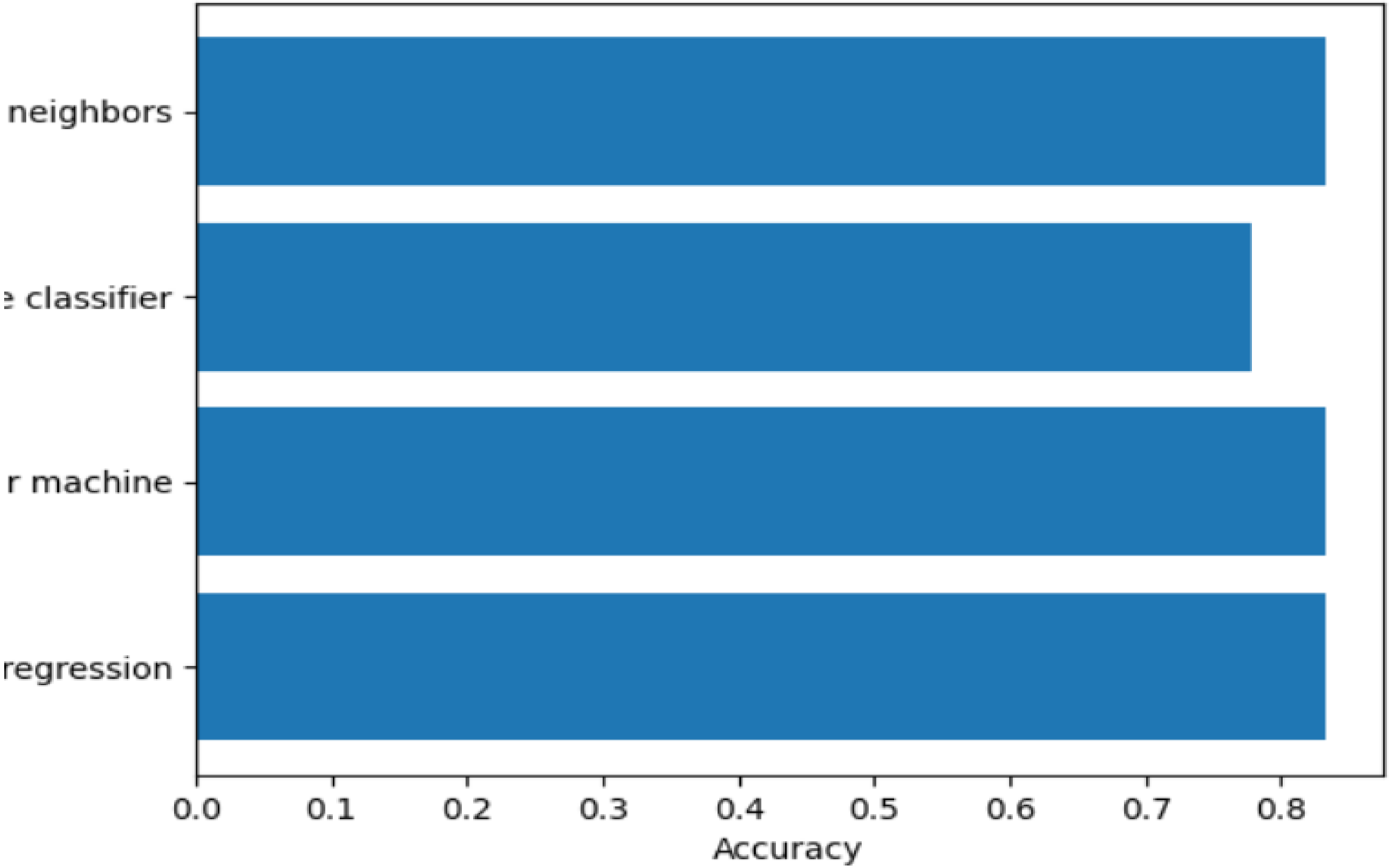
0 2500 5000 7500 10000



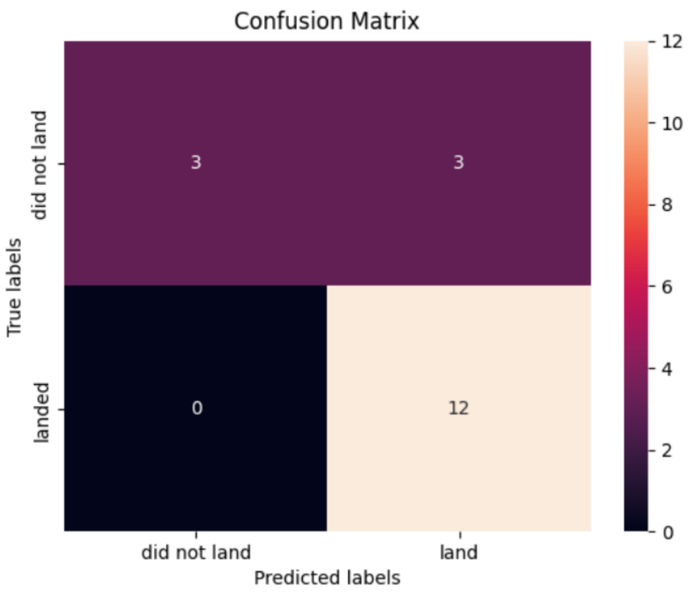
Part 5

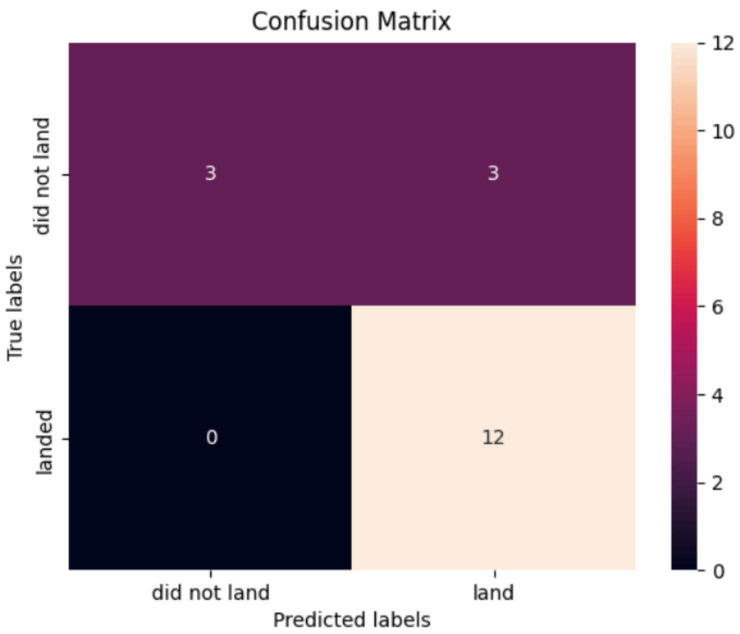
Predictive Analysis (Classification)

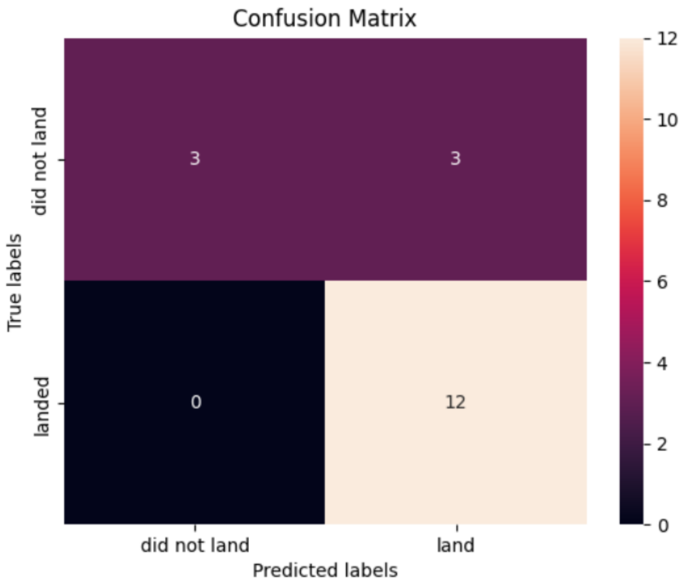
Classification Accuracy

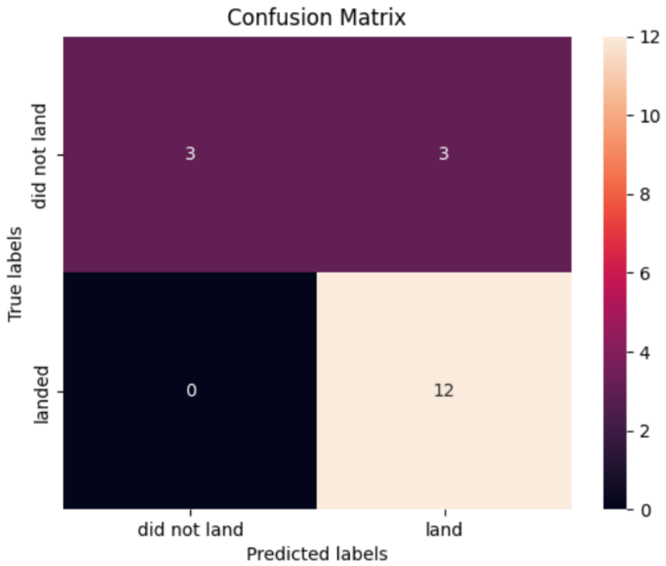


Confusion Matrix









Conclusions

- The SVM, KNN and Logistic Regression Models are the best in terms of prediction accuracy for this dataset.
- Low weighted payloads perform better than the heavier payloads.
- The Success rates for SpaceX launches is directly proportional time in years they will eventually perfect the launches.
- KSC LC 39A had the most successful launches from all the sites.
- Orbit GEO, HEO, SSO, ES L1 has the best Success Rate.


Thank You!