IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Mihail A Ana
27/10/2025

Outline

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

Executive Summary

- Steps:
  - Data Collection (SpaceX REST API, WebScraping)
  - Data Wrangling
  - Exploratory Data Analysis (SQL)
  - Interactive Visual Analytics (Folium and Dash)
  - Predictive Analysis(Linear Regression, SVM, Tree, KNN)
- Summary of all results
  - SSO, HEO, ES-L1 Orbits have a 100% success rate
  - Success Rate improves with time passing
  - VAFB-SLC was not used for heavier payload launches
  - Based on predictive analysis results, the Decision Tree model performs best on the reduced data set (18 samples)

Lin. Reg.

	precision	recall	f1-score	support
0	1.00	0.50	0.67	6
1	0.80	1.00	0.89	12
accuracy			0.83	18
macro avg	0.90	0.75	0.78	18
weighted avg	0.87	0.83	0.81	18

Dec. Tree

	precision	recall	f1-score	support
0	0.80	0.67	0.73	6
1	0.85	0.92	0.88	12
accuracy			0.83	18
macro avg	0.82	0.79	0.80	18
weighted avg	0.83	0.83	0.83	18

SVM

	precision	recall	f1-score	support
0	1.00	0.50	0.67	6
1	0.80	1.00	0.89	12
accuracy			0.83	18
macro avg	0.90	0.75	0.78	18
weighted avg	0.87	0.83	0.81	18

KNN

	precision	recall	f1-score	support
0	1.00	0.50	0.67	6
1	0.80	1.00	0.89	12
accuracy			0.83	18
macro avg	0.90	0.75	0.78	18
weighted avg	0.87	0.83	0.81	18

Introduction

- Due to the high cost of replacing a booster SapceX can outperform competitors through reusing the boosters. The purpose of this project is to predict whether if Falcon 9 First Stage will land successfully.
- We are trying to observe if there is a direct correlation between payload mass, orbit, launch site, flight number and mission success. Important to note that some of the failures have sometimes been planned by SpaceX due to various reasons.

Section 1

Methodology

Data Collection

- Data collected by using Space X Rest API:
  - Payload Information

```python
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
       if load:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

  - LaunchSite Information

```python
# Takes the dataset and uses the launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['launchpad']:
       if x:
         response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
         Longitude.append(response['longitude'])
         Latitude.append(response['latitude'])
         LaunchSite.append(response['name'])
```

```python
# Takes the dataset and uses the cores column to call the API and append the data to the lists
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
                Block.append(response['block'])
                ReusedCount.append(response['reuse_count'])
                Serial.append(response['serial'])
            else:
                Block.append(None)
                ReusedCount.append(None)
                Serial.append(None)
            Outcome.append(str(core['landing_success'])+' '+str(core['landing_type']))
            Flights.append(core['flight'])
            GridFins.append(core['gridfins'])
            Reused.append(core['reused'])
            Legs.append(core['legs'])
            LandingPad.append(core['landpad'])
```

  - Core Information (Booster Version, serial, reused count, etc)

```python
# 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'])
```

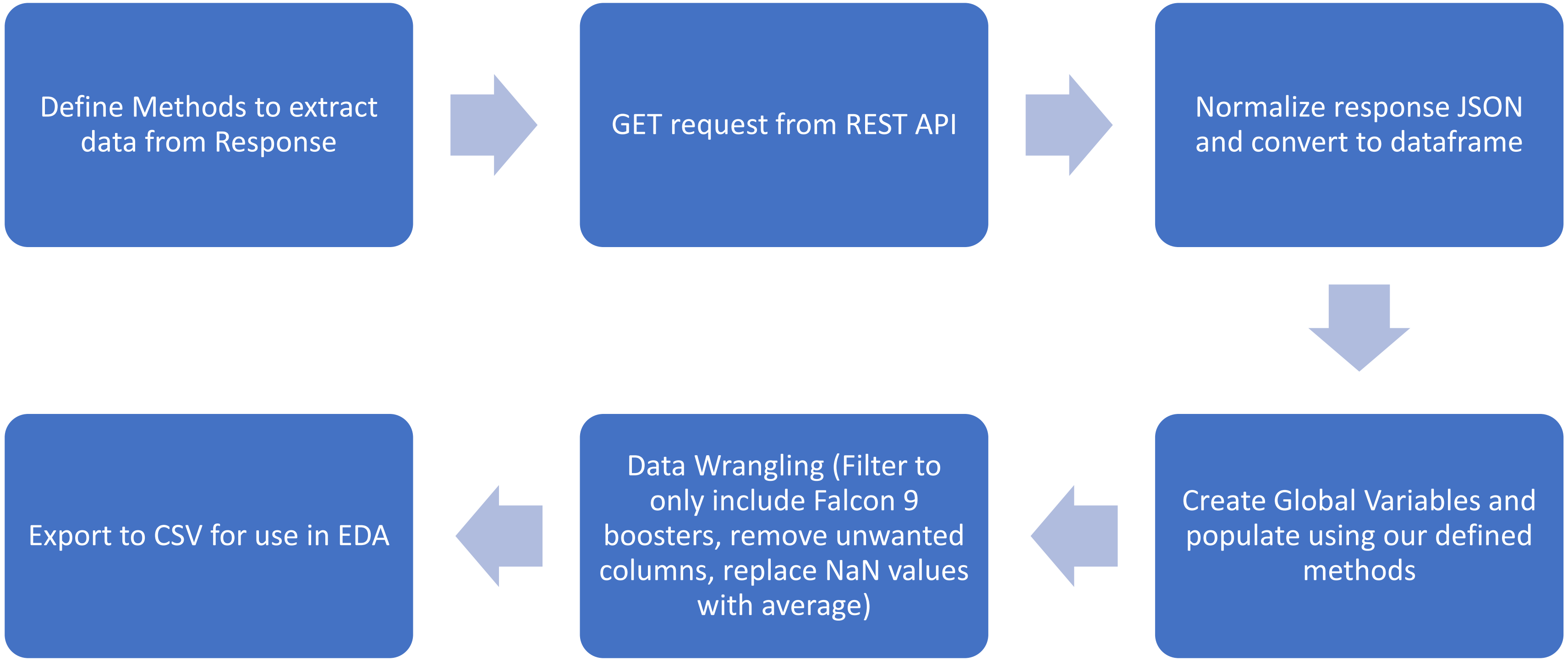
Data Collection – SpaceX API

Notebook here!

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
response = requests.get(spacex_url)
```

```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```




```
#Global variables 
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

```
# Create a data from launch_dict
df = pd.DataFrame.from_dict(launch_dict)
```

```
# Calculate the mean value of PayloadMass column
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, data_falcon9['PayloadMass'].mean())
# Replace the np.nan values with its mean value
```

NoteBook here!

Data Collection - Scraping

Request the Falcon9 Launch Wiki page from its URL → Extract all column/variable names from the HTML table header → Create a data frame by parsing the launch HTML tables

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

headers = {
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) "
                  "AppleWebKit/537.36 (KHTML, like Gecko) "
                  "Chrome/91.0.4472.124 Safari/537.36"
}
```

```
# use requests.get() method with the provided static_url and headers
# assign the response to a object
response = requests.get(static_url, headers=headers)
```

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

```
# 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

```
# Use soup.title attribute
soup.title
```

```
# 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', class_='wikitable')
print(len(html_tables))
```

```
column_names = []

# Apply find_all() function with `th` element on first_launc
# Iterate each th element and apply the provided extract_colu
# Append the Non-empty column name (`if name is not None and

# Find all <th> elements in the first table
for th in first_launch_table.find_all('th'):
    # Apply the provided function
    name = extract_column_from_header(th)

    # Append if the name is valid (not None and not empty)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

NoteBook here!

Data Collection - Scraping

Request the Falcon9 Launch Wiki page from its URL

Extract all column/variable names from the HTML table header

Create a data frame by parsing the launch HTML tables

```python
extracted_row = 0

# Extract each table
for table_number, table in enumerate(soup.find_all('table', "wikitable plainrowheaders collapsible")):
    # get table row
    for rows in table.find_all("tr"):
        # check to see if first table heading is a number corresponding to launch number
        if rows.th:
            if rows.th.string:
                flight_number = rows.th.string.strip()
                flag = flight_number.isdigit()
        else:
            flag = False

        # get table element
        row = rows.find_all('td')

        # if it's a valid number, save cells in the dictionary
        if flag:
            extracted_row += 1

            # Flight Number
            launch_dict['Flight No.'].append(flight_number)

            # Date and Time
            datatimelist = date_time(row[0])
            date = datatimelist[0].strip(',')
            time = datatimelist[1]
            launch_dict['Date'].append(date)
            launch_dict['Time'].append(time)
```

```python
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

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

Data Wrangling

NoteBook Here!

- The step consisted of these steps:

  1. Filter the dataframe to only include Falcon 9 launches

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

Now that we have removed some values we should reset the FlgihtNumber column

```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

  2. Calculate average PayloadMass to replace NaN values within Respective column

```
# Calculate the mean value of PayloadMass column
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, data_falcon9['PayloadMass'].mean())
# Replace the np.nan values with its mean value
```

  3. Consolidate Landing outcomes(as response Outcomes vary based on mission objectives):

```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

```
0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 False Ocean
6 None ASDS
7 False RTLS
```

We create a set of outcomes where the second stage did not land successfully:

```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

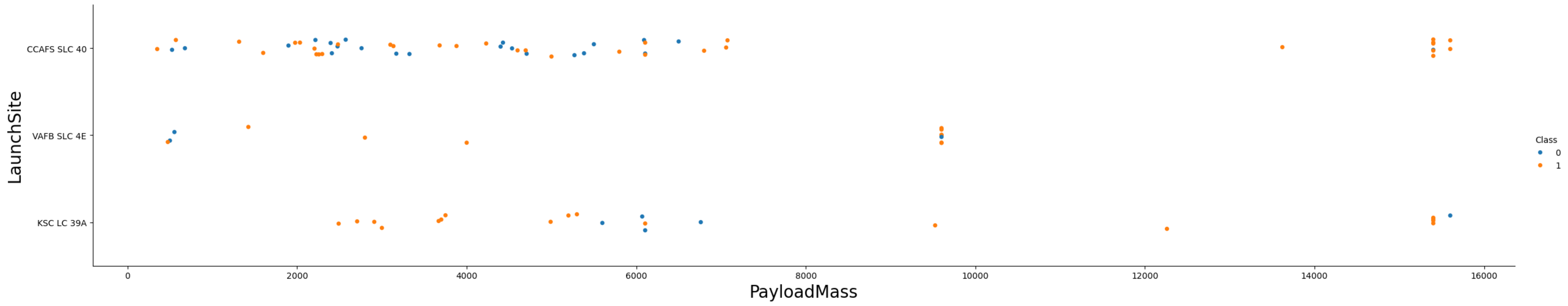
  4. Create a landing outcome label from Outcome column

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = df['Outcome'].apply(lambda x: 0 if x in bad_outcomes else 1)
```

EDA with Data Visualization – LaunchSite/PayloadMass

- Relevant in order to identify any corelaction between chosen site and payload.
- We can identify that VAFB-SLC does not operate Payloads above 10000kg

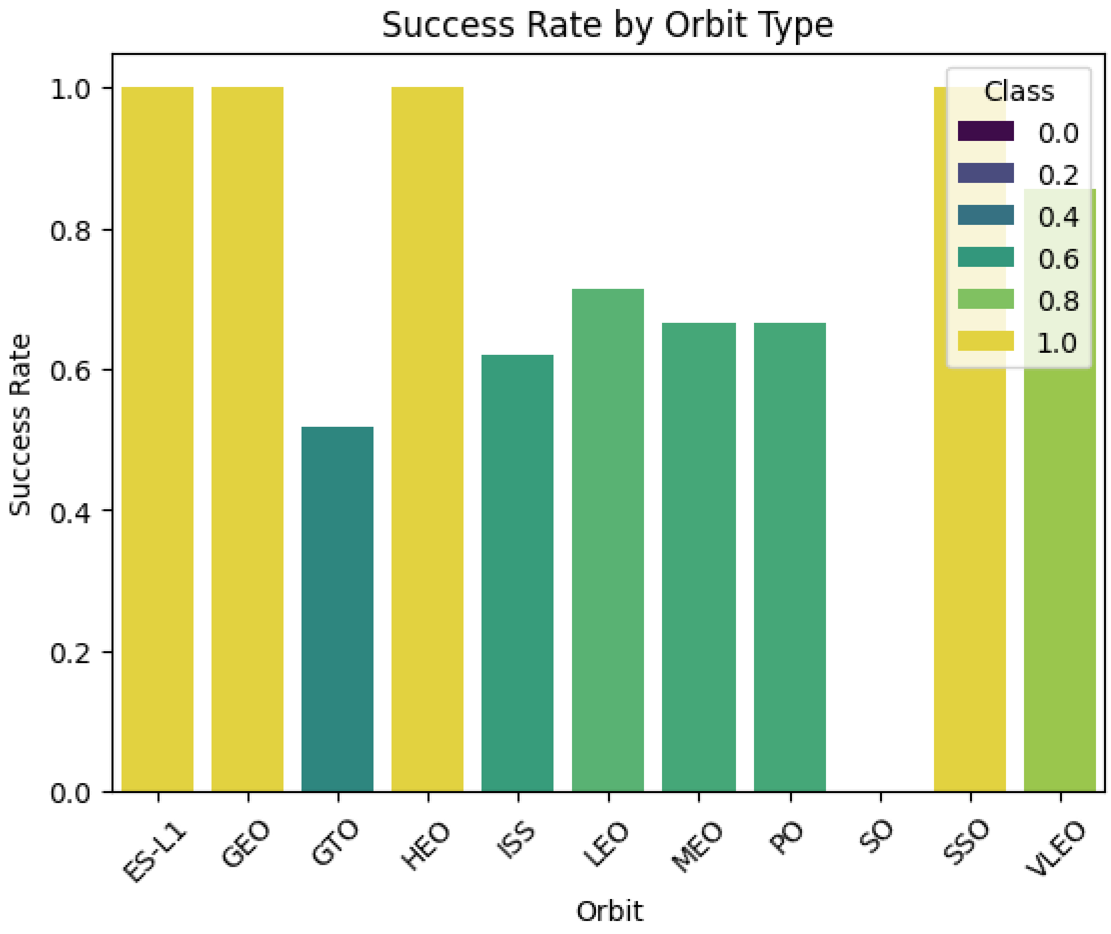
Notebook [Here!]



EDA with Data Visualization –SuccessRate/OrbitType

- Relevant in order to identify if mission destination could affect success rate
- We can identify that ES-l1, GEO, HEO and SSO orbits have a success rate of 100%

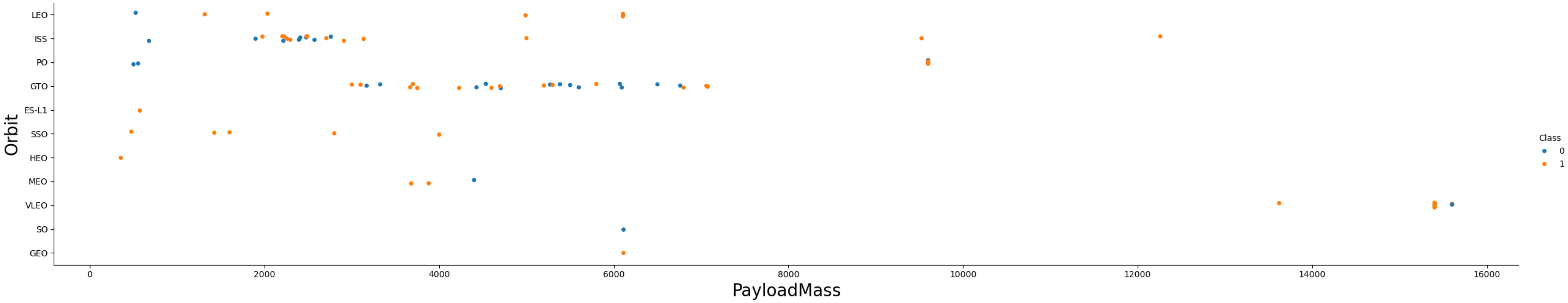
Notebook [Here!]()



EDA with Data Visualization –OrbitType/Payload

- With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.
- However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.

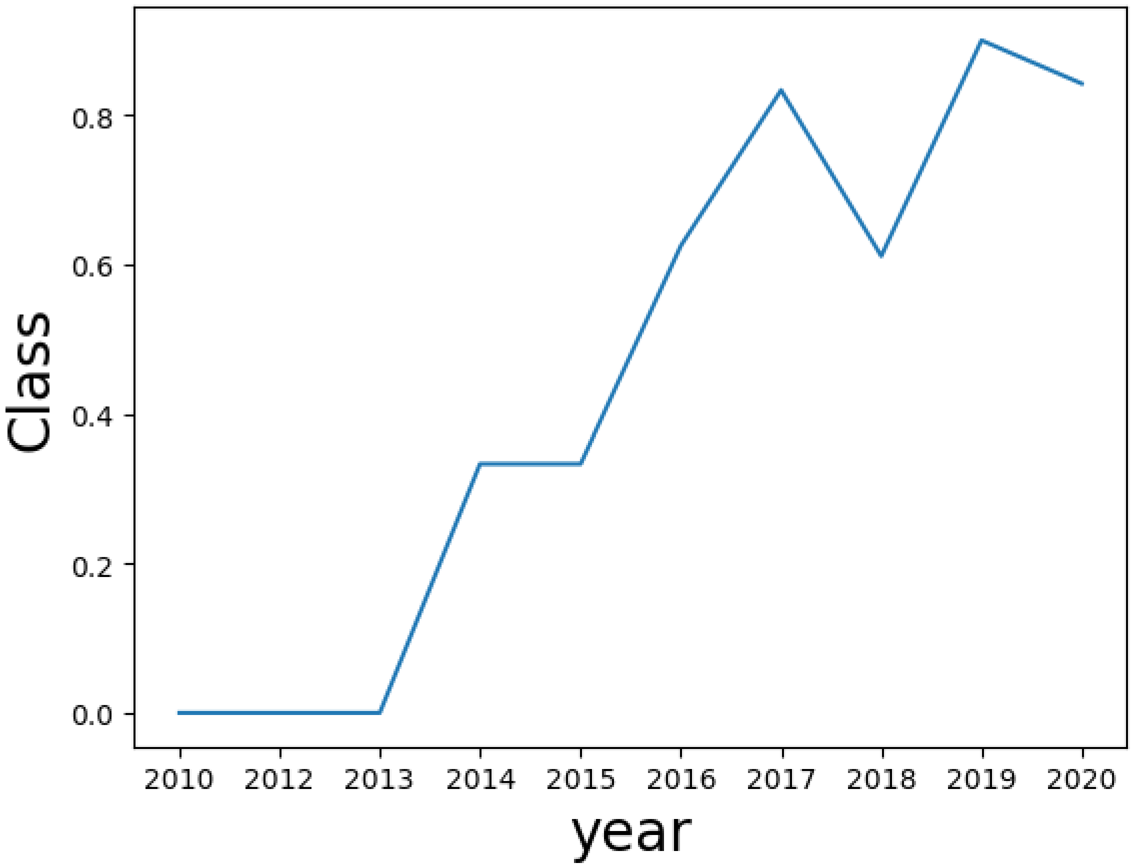
Notebook [Here!]



EDA with Data Visualization – Success/Year

- Sucees Rate vs year is important in order to maintain trust and operability
- We can see there is a positive trend as success rate has increased over 10 years

Notebook [Here!]



EDA with SQL

Notebook Here!

- SQL Queries performed:

1. Display total mass launched by NASA (CRS)
2. Display average mass carried by an F9 V1.1 booster
3. List date of first successful ground landing pad outcome
4. List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
5. List the total number of successful and failure mission outcomes
6. List all the booster_versions that have carried the maximum payload mass
7. 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
8. 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.

Build an Interactive Map with Folium

- Added:
  - NASA CRS
  - Launch Site Locations
  - Marker group for tracking launches with outcomes (green/red)
  - Mouse tracker for determining key distances to locations of interest from sites
- Notebook Here!

Build a Dashboard with Plotly Dash

- Site Drop down input component added with Pie Plot depicting success rates
- Range Slider for Different Payload Ranges with Scatter Plot for Booster Type

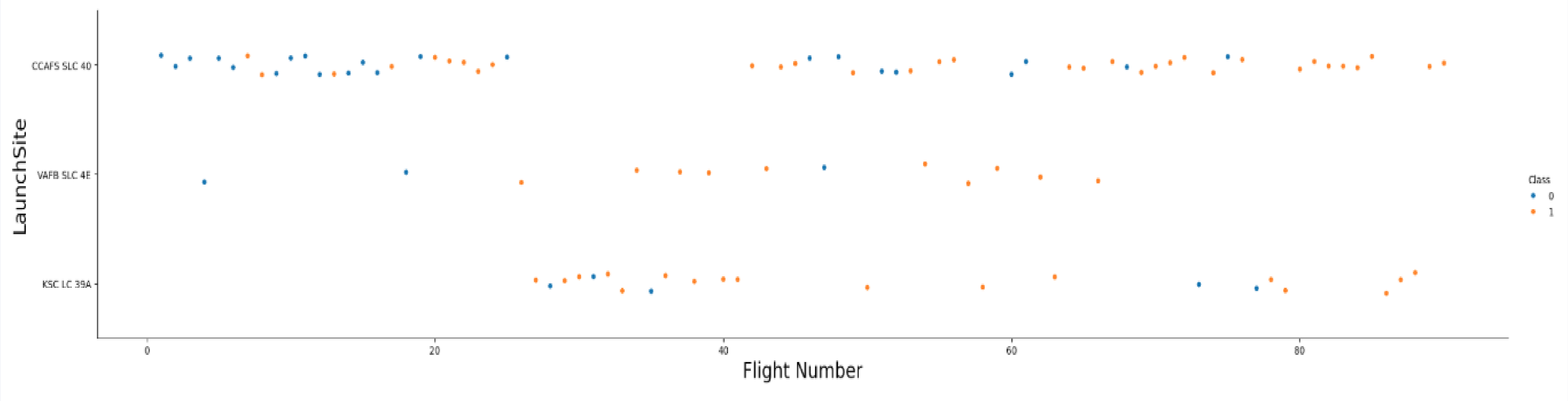
Predictive Analysis (Classification)

- 4 Models were built and tested
  - Linear Regression
  - Support Vector Machine (SVM)
  - Decision Tree
  - K-Nearest Neighbours (KNN)
- Used Test – Train function with the following parameters:
  - Set the parameter test_size to 0.2 and random_state to 2, producing 18 samples.
  - Cross Validation was set to 10 for each model

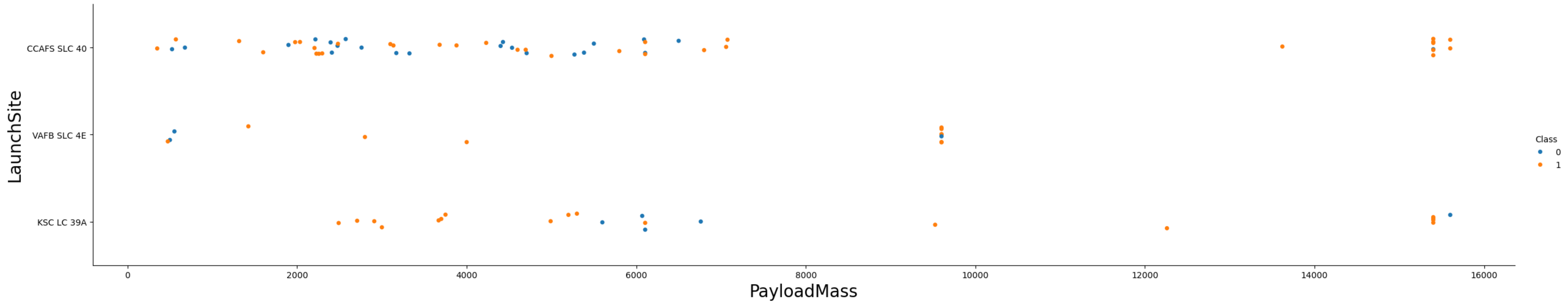
Section 2

Insights drawn from EDA

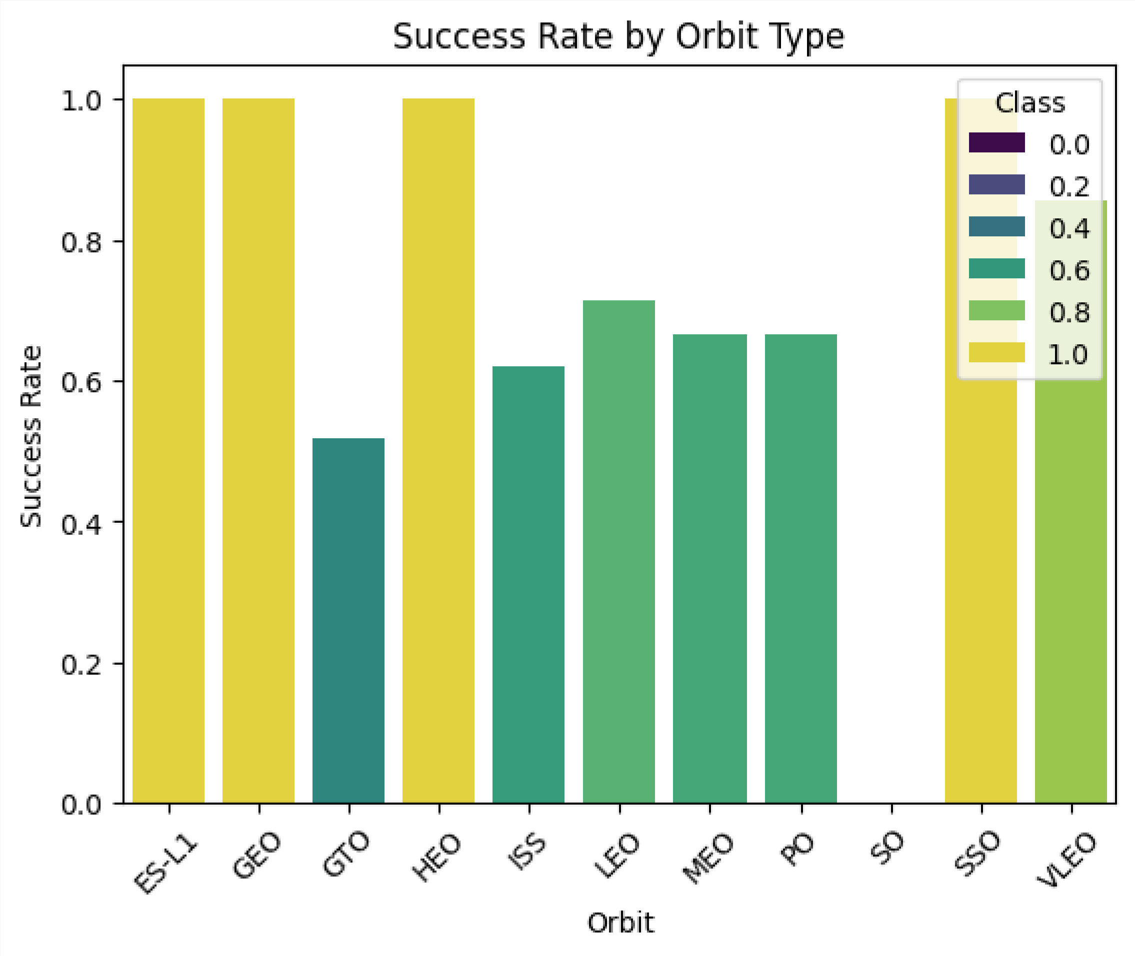
Flight Number vs. Launch Site



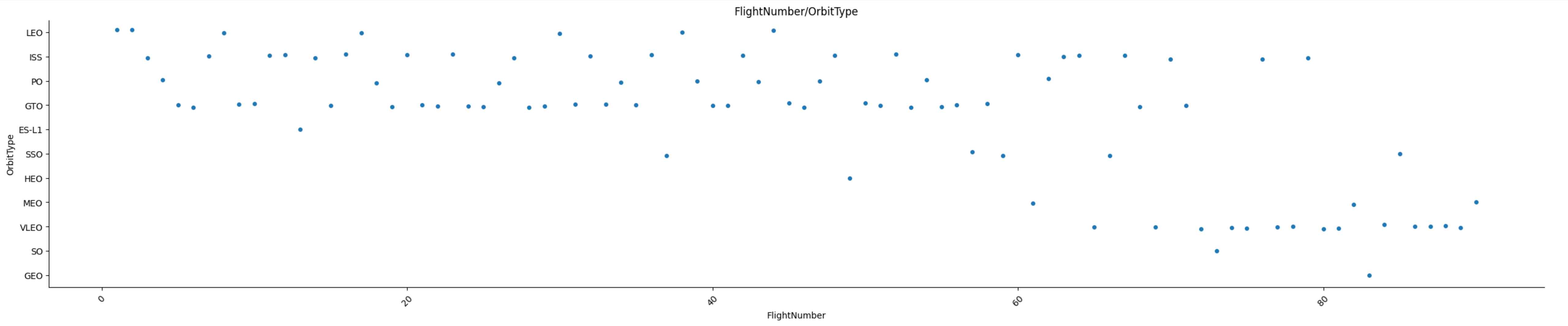
Payload vs. Launch Site



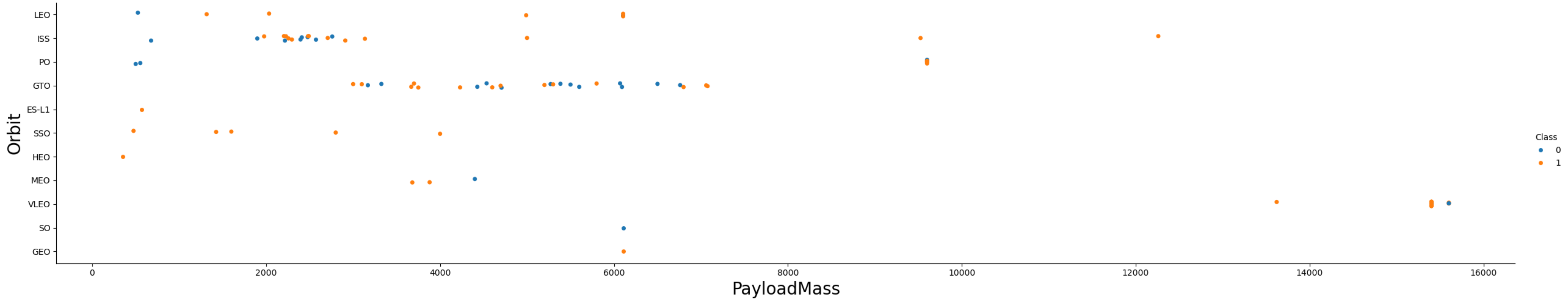
Success Rate vs. Orbit Type



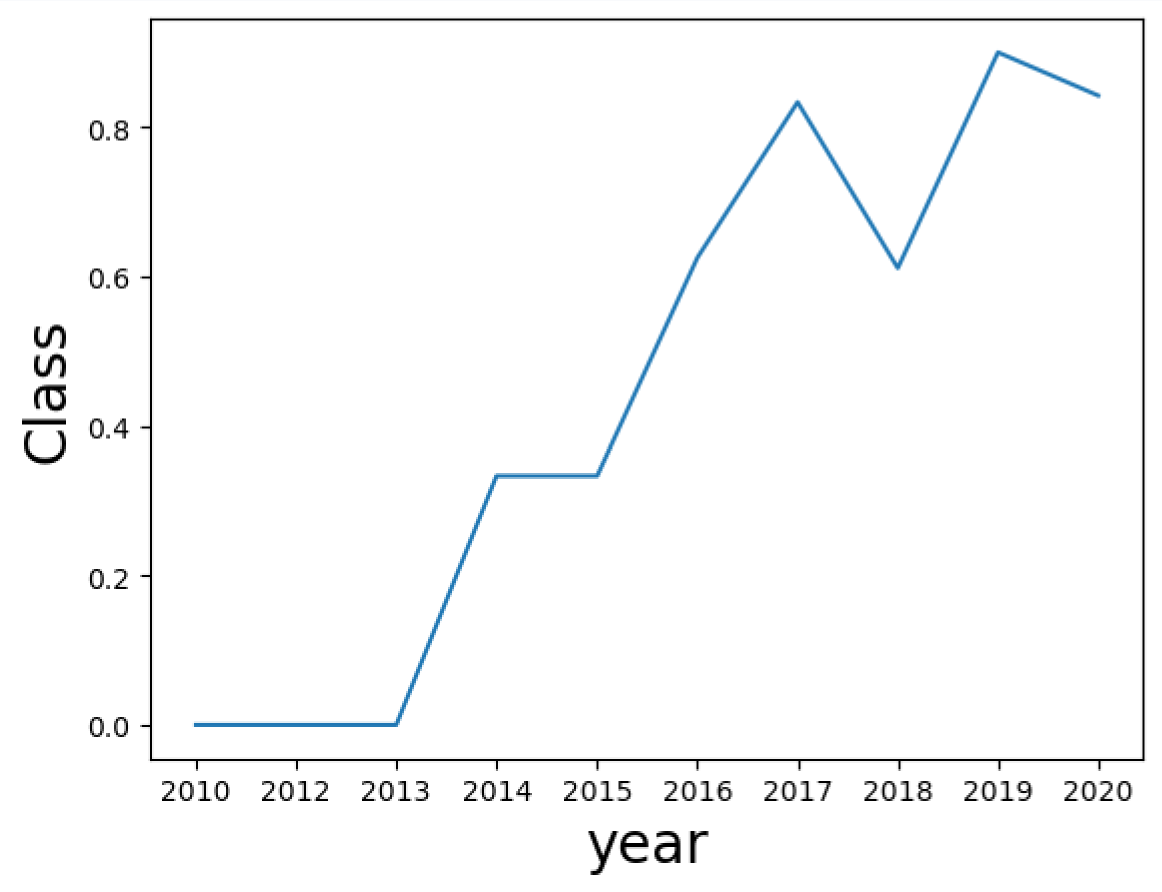
Flight Number vs. Orbit Type



Payload vs. Orbit Type



Launch Success Yearly Trend



All Launch Site Names

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

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

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

There are 4 distinct Launch Sites as seen above

Launch Site Names Begin with '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

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

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

total_mass
48213

Average Payload Mass by F9 v1.1

```
%sql SELECT AVG("PAYLOAD_MASS__KG_") as avg_mass_f9v1dot1 FROM SPACEXTABLE WHERE Booster_Version LIKE '%F9 v1.1%';
```

'9 v1.1

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

avg_mass_f9v1dot1
2534.6666666666665

First Successful Ground Landing Date

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

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2015-12-22	1:29:00	F9 FT B1019	CCAFS LC-40	OG2 Mission 2 11 Orbcomm-OG2 satellites	2034	LEO	Orbcomm	Success	Success (ground pad)

Successful Drone Ship Landing with Payload between 4000 and 6000

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 * FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000 AND Landing_Outcome LIKE '%Success (drone ship)%';
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2016-05-06	5:21:00	F9 FT B1022	CCAFS LC-40	JCSAT-14	4696	GTO	SKY Perfect JSAT Group	Success	Success (drone ship)
2016-08-14	5:26:00	F9 FT B1026	CCAFS LC-40	JCSAT-16	4600	GTO	SKY Perfect JSAT Group	Success	Success (drone ship)
2017-03-30	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
2017-10-11	22:53:00	F9 FT B1031.2	KSC LC-39A	SES-11 / EchoStar 105	5200	GTO	SES EchoStar	Success	Success (drone ship)

Total Number of Successful and Failure Mission Outcomes

```
%sql SELECT COUNT(*) FROM SPACEXTABLE WHERE Mission_Outcome LIKE '%Success%';
```

* sqlite:///my_data1.db
Done.

COUNT(*)
100

```
%sql SELECT COUNT(*) FROM SPACEXTABLE WHERE Mission_Outcome LIKE '%Failure%';
```

* sqlite:///my_data1.db
Done.

COUNT(*)
1

failure mission outcomes

ation here

Boosters Carried Maximum Payload

```
%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

yload mass

2015 Launch Records

```
%sql SELECT substr(Date, 6, 2) AS Month, Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTABLE WHERE substr(Date, 1, 4) = '2015' AND Landing_Outcome LIKE '%Failure (drone ship)%';
```

```
 * 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

- Present your query result with a short explanation here

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

```
%sql SELECT Landing_Outcome, COUNT(*) AS Outcome_Count FROM SPACEXTABLE WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY Landing_Outcome ORDER BY Outcome_Count DESC;
```

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

Landing_Outcome	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

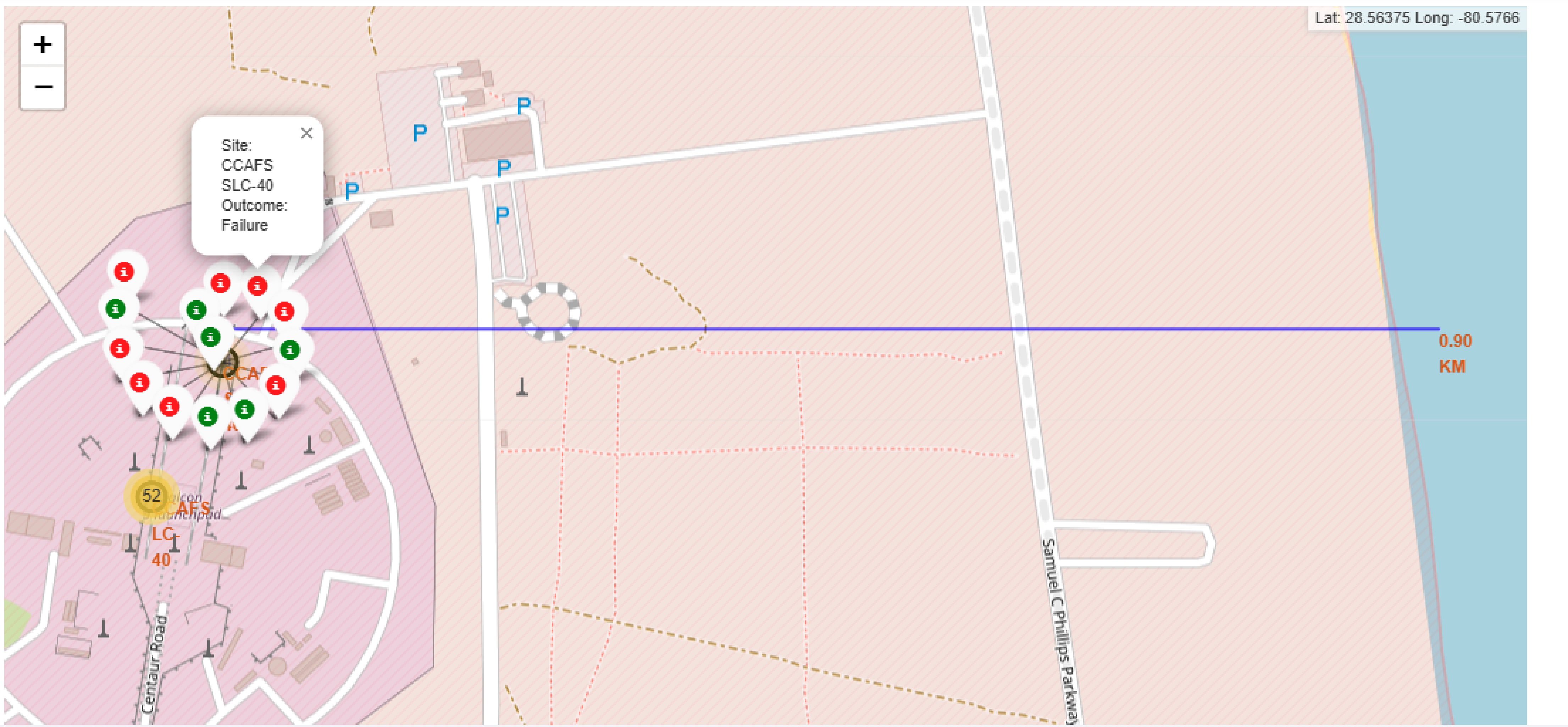
Folium – Launch Sites

Folium – Color Labled launch outcomes

VAFB-SLC

CCAFS LC-40

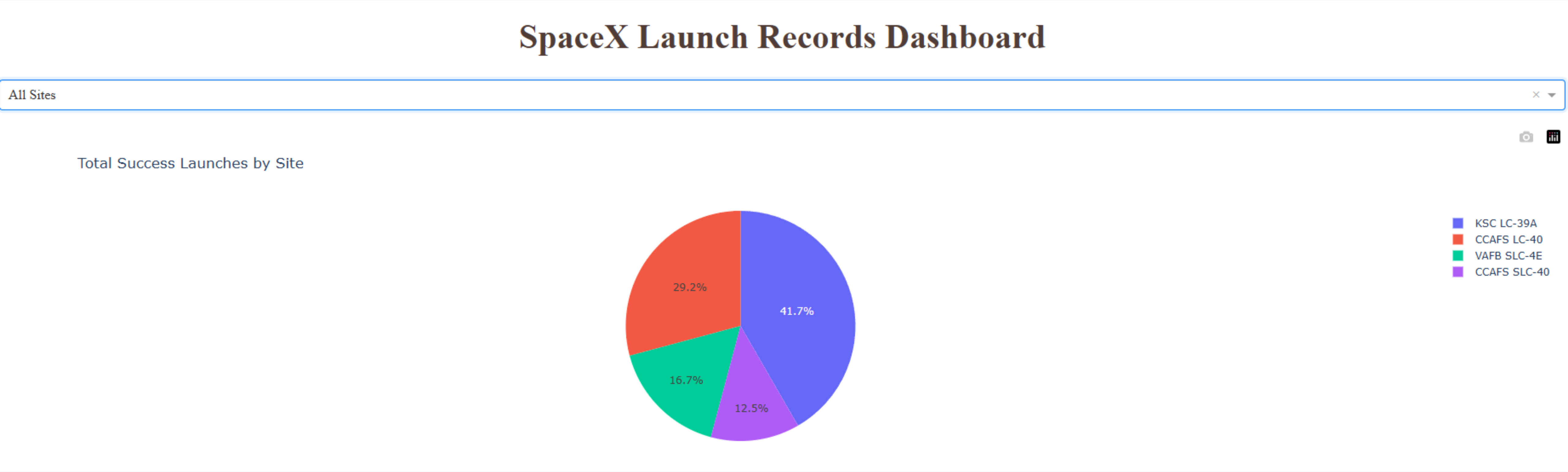
Folium – Marker between CCAFS-SLC to CoastLine



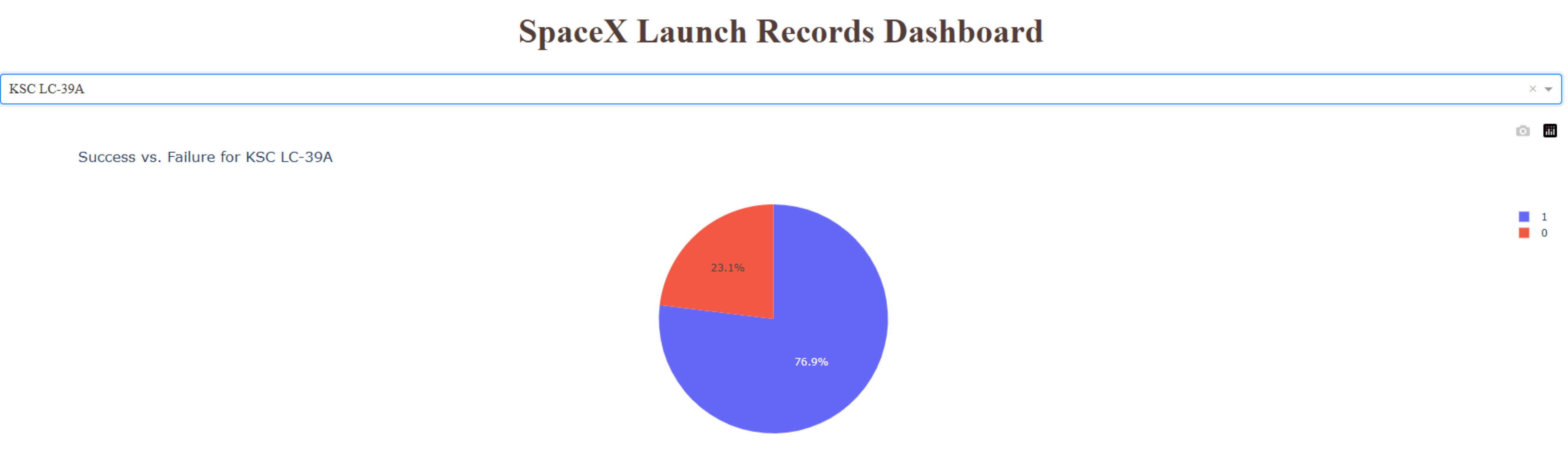
Section 4

Build a Dashboard with Plotly Dash

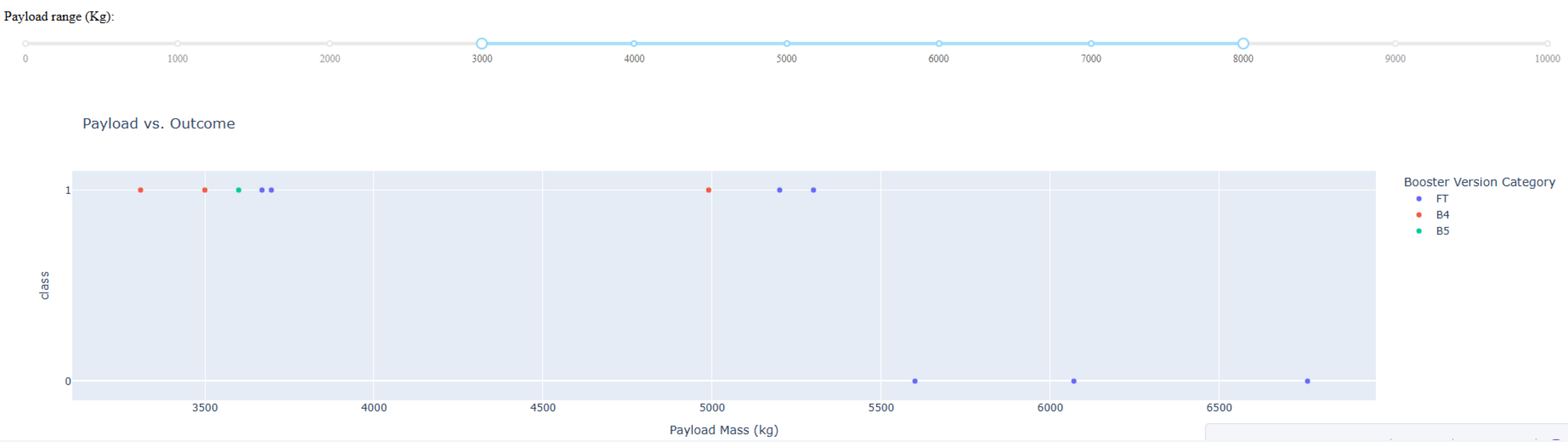
<Dashboard Screenshot 1>



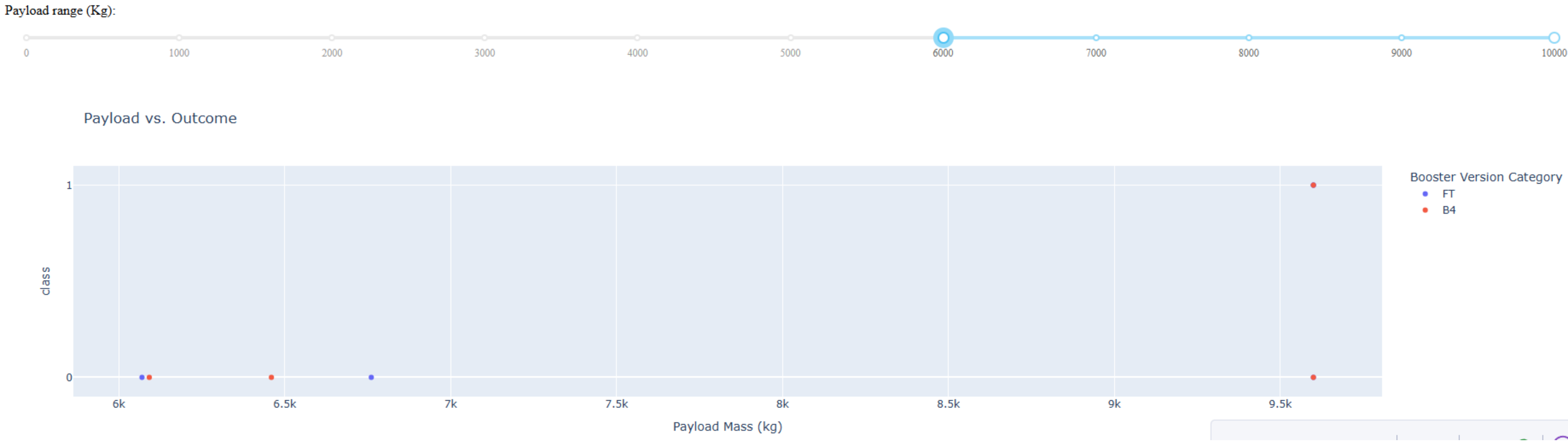
<Dashboard Screenshot 2>



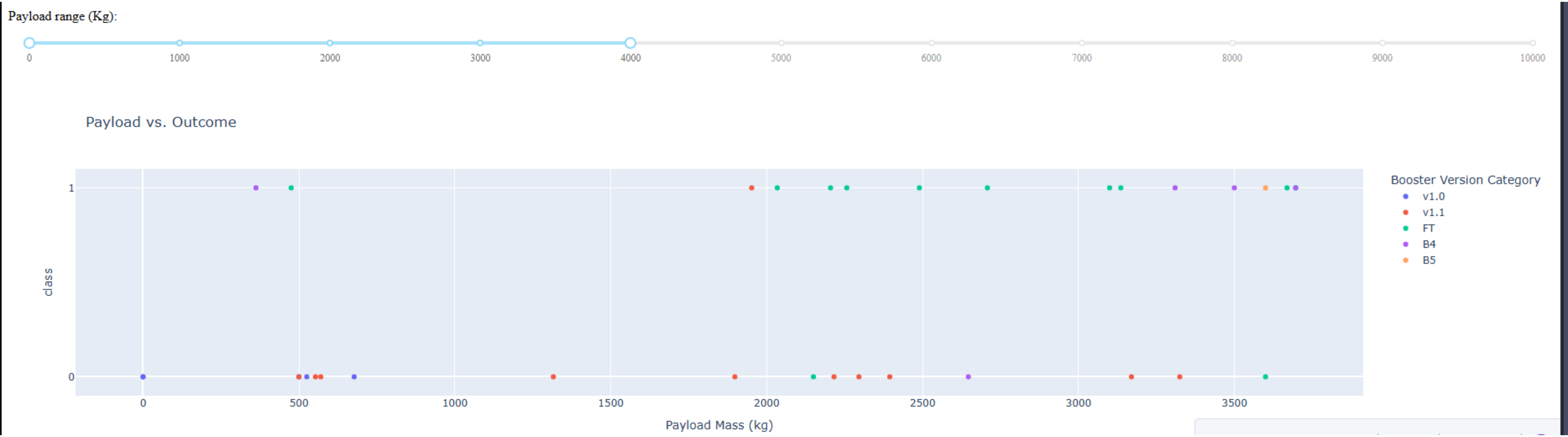
<Dashboard Screenshot 3>



<Dashboard Screenshot 4>



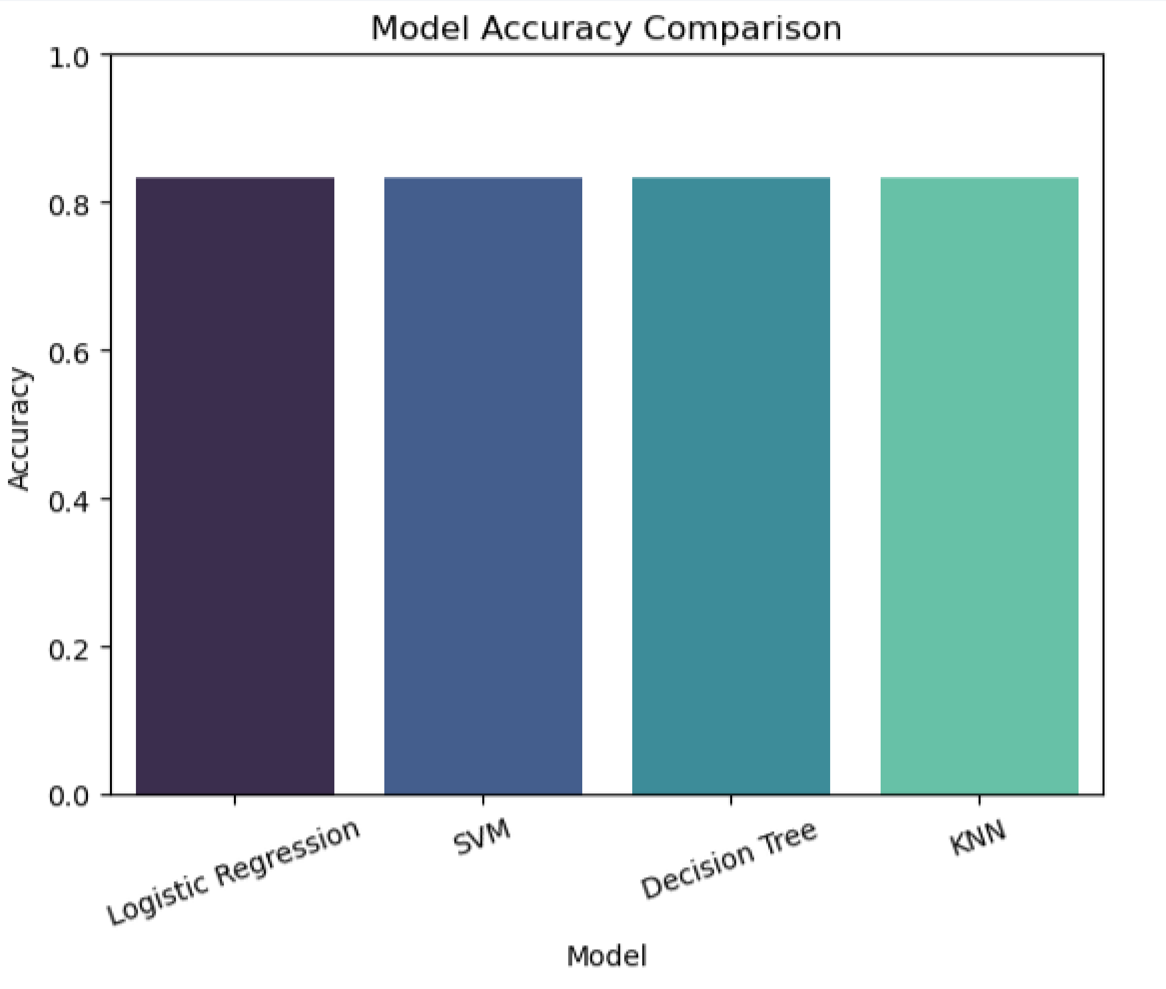
<Dashboard Screenshot 5>

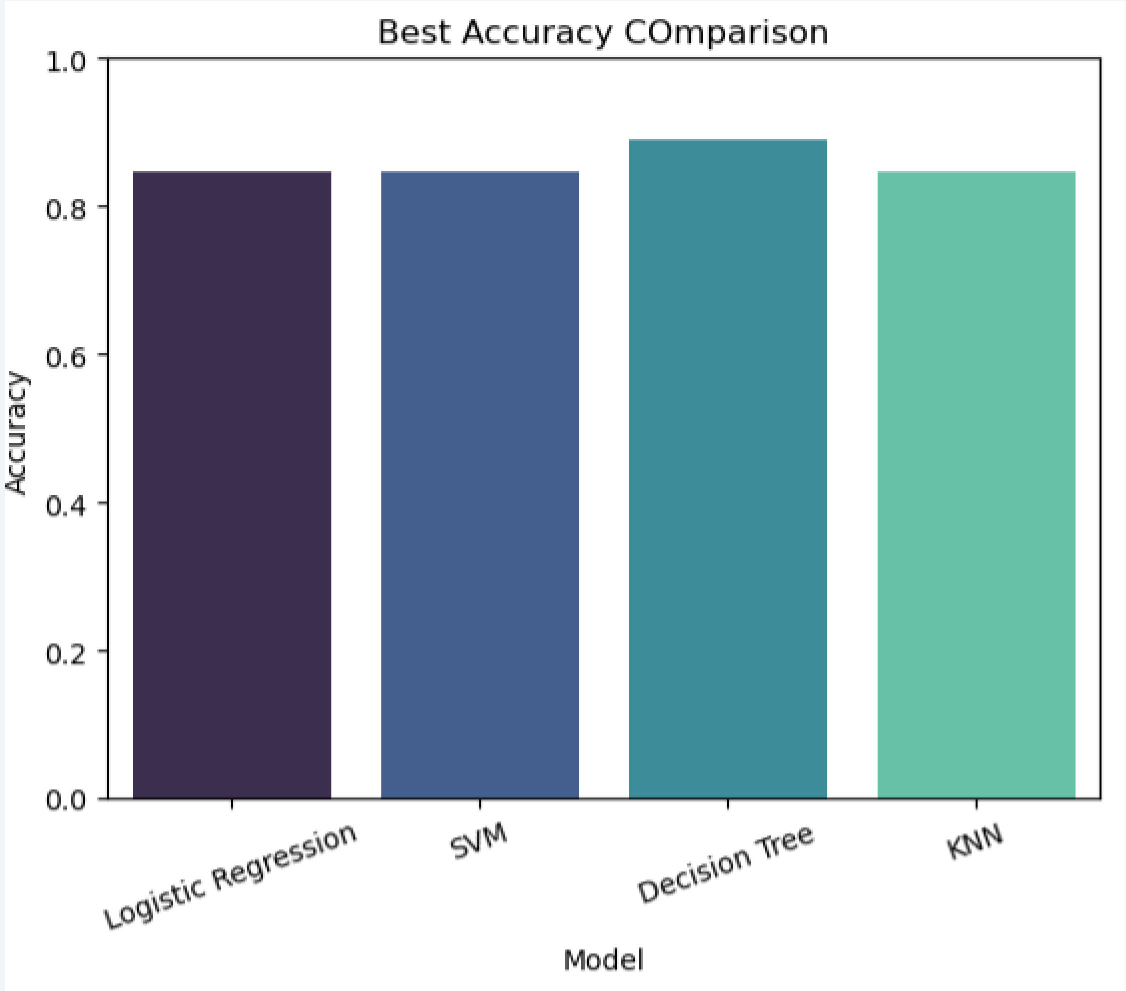


Section 5

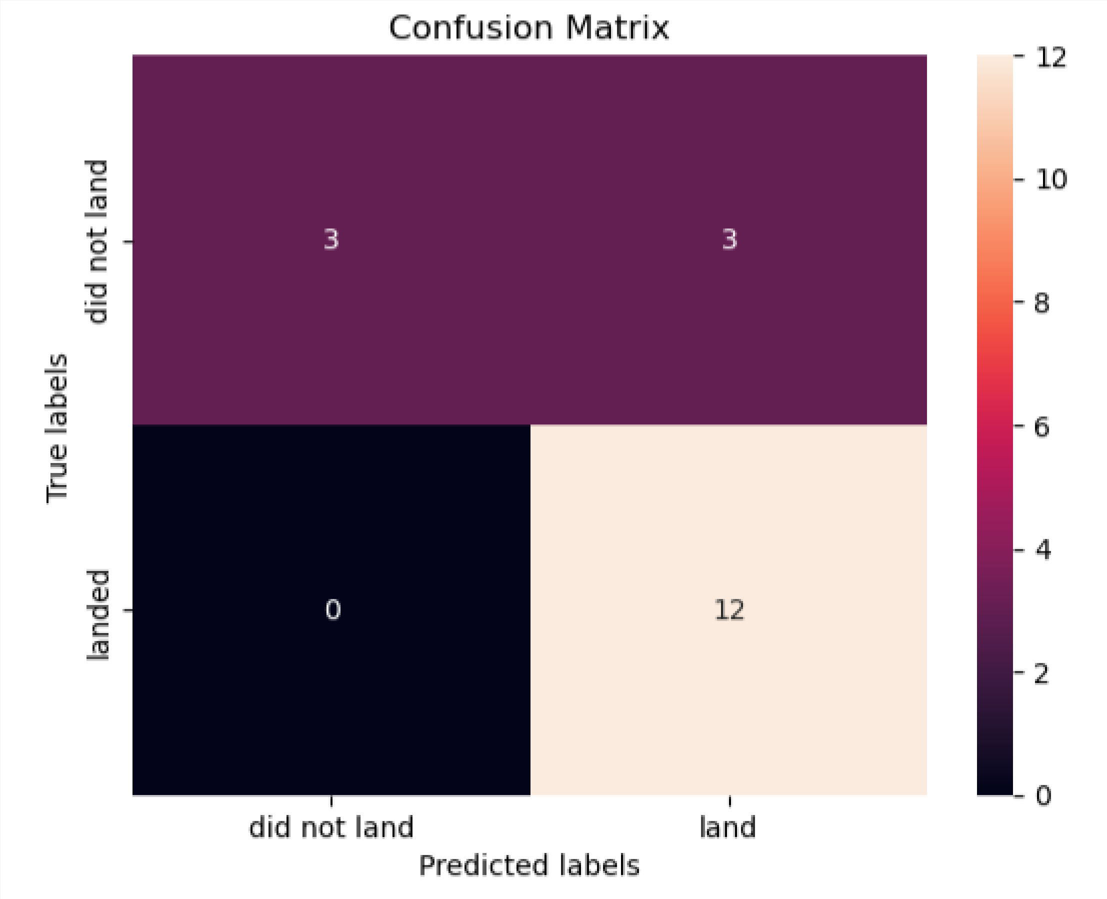
Predictive Analysis (Classification)

Classification Accuracy





Confusion Matrix



There are 3 false positives, leading one to Assume the Model presents a lower Recall. Due to the Imbalanced Data, it would be best to Proceed with the F-1Score: 0.67

Data Balance Overview: 66/33% 1-0

```
data['Class'].value_counts()

Class
1    60
0    30
Name: count, dtype: int64
```

Conclusions

- When using test set, all models produce the same accuracy, therefore the GridSearchCV best score parameter will be used, resulting in Decision Tree being the best choice. A larger test sample would produce a more conclusive result.
- Multiple factors affect mission outcome such as Payload, Orbit type, and Site Location. There are additional factors not covered such as atmospheric conditions on launch day.
- Under Heavier Payloads and Orbit type of Polar, LEO and ISS produce a positive landing rate while in general ES-L1, SSO, and HEO produce a higher success rate.

Appendix

- Full Git hub: https://github.com/GenericParadox/IBM-Applied-Data-Science-Capstone

Thank you!