IBM Developer
SKILLS NETWORK

# IBM DATA SCIENCE CAPSTONE PROJECT

LAMIA AL-HUSSAINAN

1-6-2023

# OUTLINES:

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

# EXECUTIVE SUMMARY:

## Summary of methodologies

- Data Collection
- Data Wrangling
- Exploratory Data Analysis
- Interactive Visual Analysis

## Summary of all results

•Exploratory Data Analysis
•Geospatial Analysis
•Interactive Dashboard
•Predictive Analysis of Classification Models

# INTRODUCTION:

Sending spacecraft to the International Space Station, Starlink, a satellite internet constellation providing satellite Internet access, Sending manned missions to Space, are all accomplishments of the SpaceX.
SpaceX is the most successful commercial space company, that is making space travel affordable for everyone. Unlike other rocket providers, SpaceX's Falcon 9 can recover the first stage, and that is one of the reason why it's relatively inexpensive, as SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upwards of 165 million dollars each.

Therefore, we will predict whether the Falcon 9 first stage will land. If this was predicted, the cost of a launch will be determined. This information can be used if an alternate company would bid against SpaceX for a rocket launch.

This project will ultimately predict if the Space X Falcon 9 first stage will land successfully.

# METHODOLOGY:

## Executive Summary

**Data Collection Methodology:**

- SpaceX API
- Web Scraping
- List of Falcon 9 and Falcon Heavy launches

**Perform Data Wrangling**

- Remove missing values
- Count the values
- Generate a landing outcome label showing the booster landing successfully or unsuccessfully

**Perform Exploratory Data Analysis (EDA)**

- SQL
- Visualization

**Perform Interactive Visual Analytics**

- Folium
- Plotly Dash

**Perform Predictive Analysis**

- Using Classification models including Logistic Regression - SVM - Decision Tree - KNN

# DATA COLLECTION:

**The Dataset was collected from a Wikipedia page (List of Falcon 9 and Falcon Heavy launches).**

**The process of Data Collection has been completed by using the SpaceX records to create a SpaceX API and Web Scraping data collection.**

SpaceX API

1. Request and parse the SpaceX launch data using the GET request
2. Filter the dataframe
3. Dealing with missing values

Web Scraping

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

# DATA COLLECTION - SPACEX API:

GITHUB LINK

Creating SpaceX API to retrieve information such as: launches, payloads, landing outcomes, etc.

## Requesting SpaceX data using <GET RESPONSE>

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

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

## Converting to a data frame using <JSON_normalize>

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

Data will be stored in lists and will be used to create a new dataframe. Call the function to retrieve the data.

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

```
[20]: # Call getLaunchSite
getLaunchSite(data)

[21]: # Call getPayloadData
getPayloadData(data)

[22]: # Call getCoreData
getCoreData(data)
```

## Using these lists we will create a Dictionary, and a pandas datafarme.

```
[23]: 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}
```

```
[24]: # Create a data from launch_dict
data2 = pd.DataFrame(launch_dict)
```

## Filter the dataframe, and dealing with missing values using <.mean() and .replace()>

```
[27]: # Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = data2[data2['BoosterVersion']=='Falcon 9']
data_falcon9
```

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

# DATA COLLECTION - WEB SCRAPING:

GITHUB LINK

**Performing Web Scraping to collect Falcon 9 historical launch records from a Wikipedia page titled List of Falcon 9 and Falcon Heavy launches**

Request the Falcon9 Launch HTML page using GET method, Creating beautifulsoup object.

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

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

Finding all tables using <.find_all>

```
[8]: # Use the find_all function in the BeautifulSoup object
     # Assign the result to a list called `html_tables`
     html_tables=soup.find_all('table')
```

Extracting columns.

```
10]: column_names = []

     # Apply find_all() function with `th` element on first_launch_table
     # Iterate each th element and apply the provided extract_column_from_header(
     # Append the Non-empty column name (`if name is not None and len(name) > 0`)
     for i in first_launch_table.find_all('th'):
         if extract_column_from_header(i)!=None:
             if len(extract_column_from_header(i))>0:
                 column_names.append(extract_column_from_header(i))
```

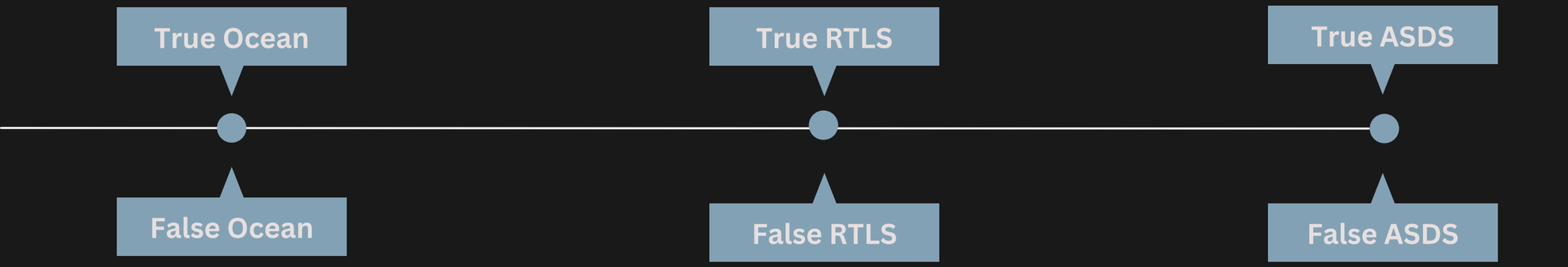
create an empty Dictionary with keys from the extracted column names, And create a dataframe.

```
[12]: 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
      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']=[]
```

```
[18]: df=pd.DataFrame(launch_dict)
```

# DATA WRANGLING:

There are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident.




We will mainly convert those outcomes into Training Labels with 1 means the booster successfully landed, 0 means it was unsuccessful

# DATA WRANGLING:

GITHUB LINK

`True Ocean` means the mission outcome was successfully landed to a specific region of the ocean while `False Ocean` means the mission outcome was unsuccessfully landed to a specific region of the ocean. `True RTLS` means the mission outcome was successfully landed to a ground pad `False RTLS` means the mission outcome was unsuccessfully landed to a ground pad. `True ASDS` means the mission outcome was successfully landed to a drone ship `False ASDS` means the mission outcome was unsuccessfully landed to a drone ship. `None ASDS` and `None None` these represent a failure to land.

[11]:
```
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
```

Create a set of outcomes where the second stage did not land successfully.

[12]:
```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

[12]:
```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

Create a list where the element is zero if the corresponding row in Outcome is in the set bad_outcome; otherwise, it's one. Then assign it to the variable landing_class:

[15]:
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
def theoutcome(item):
    if item in bad_outcomes:
        return 0
    else:
        return 1
landing_class = df['Outcome'].apply(theoutcome)
landing_class
```

[15]:
```
0     0
1     0
2     0
3     0
4     0
     ..
85    1
86    1
87    1
88    1
89    1
Name: Outcome, Length: 90, dtype: int64
```

# EDA - DATA VISUALIZATION:

GITHUB LINK

## SCATTER

- Flight Number & Launch Site
- Payload & Launch Site  Flight Number & Orbit Type
- Payload & Orbit Type

Scatter charts are useful to show relationships, or correlations, between variables.

## BAR

- Orbit Type & Success Rate

Bar charts are used to compare a numerical value to a categorical variable.

## LINE

- Year & Success Rate

Line charts are used to show the change of a variable over time.

# EDA - SQL:

GITHUB LINK

Summarization of all SQL queries performed:

- Display the names of the unique launch sites  in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome in ground pad was achieved
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster_versions which have carried the maximum payload mass, using a subquery
- 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
- Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order

# INTERACTIVE MAP - FOLIUM:

GITHUB LINK

**Mark all launch sites on a map**

- Create a folium Map object, with an initial center location using folium.Map
- Create and add folium.Circle and folium.Marker for each launch site on the site map

**Mark the success/failed launches for each site on the map**

- Assign a marker color of successful (class = 1) as green, and failed (class = 0) as red
- Add a folium.Marker to the MarkerCluster object, to put the launches into clusters

**Calculate the distances between a launch site to its proximities**

- Create a folium.Marker object to show the distance
- Draw a folium.PolyLine and add this to the map to display the distance line between two points

# DASHBOARD - PLOTLY DASH:

DAHSBOARD

**Summarization of all plots/graphs:**

## PIE CHART:

- Showing total success launches of all sites using < px.pie() >
- Indicate the success rate of individual launch per site

## SCATTER CHART:

- Showing the relationship between outcomes & payload mass using < px.scatter() >
- Filtering by the payload mass and booster version

# PRIDICTIVE ANALYSIS - CLASSIFICATION:

GITHUB LINK

## Building Model

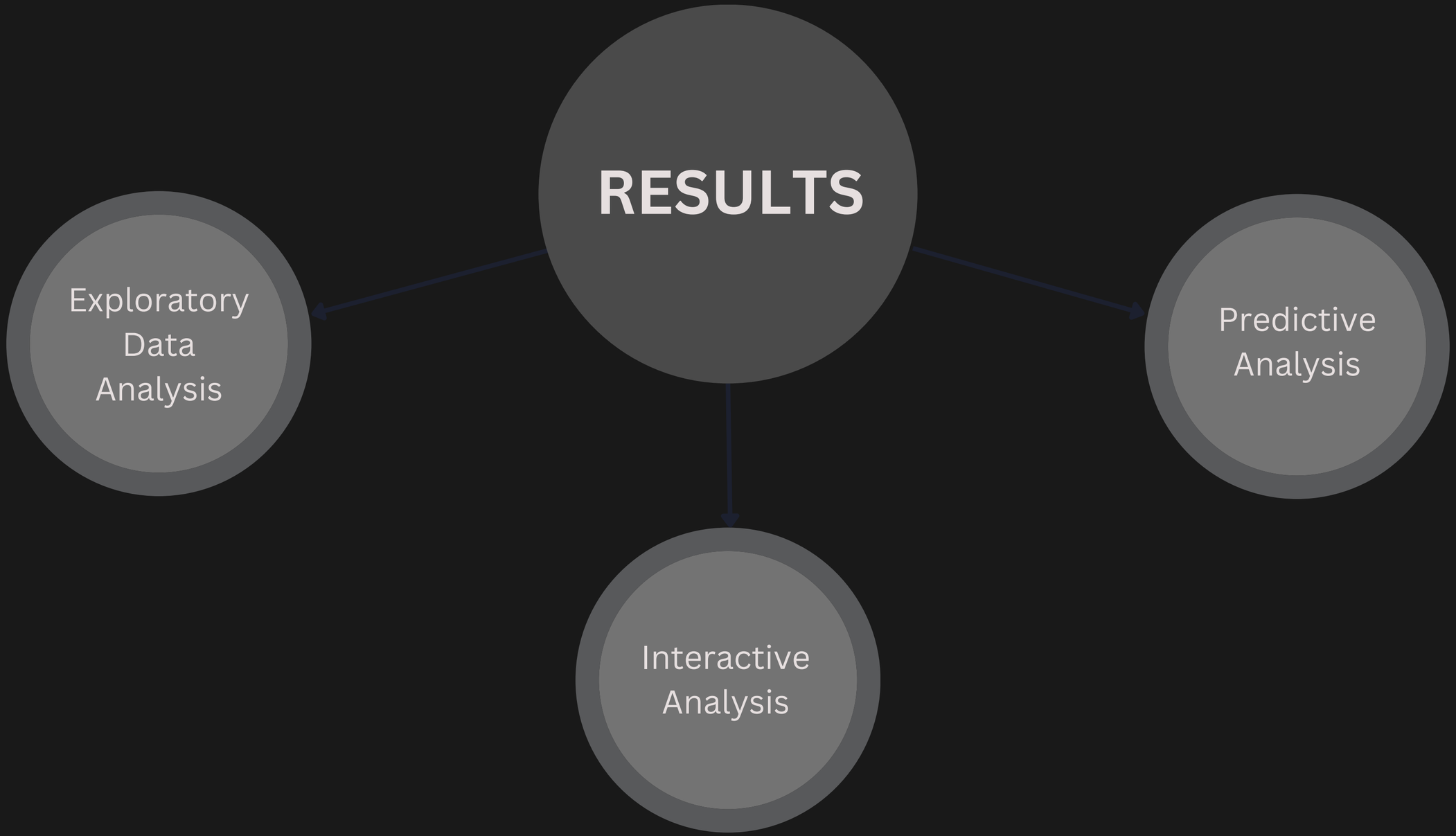
- Create a column for the class
- Standardize the data
- Split into training data and test data
- Create Logistic Regression, SVM, Decision Tree, KNN classification objects

## Evaluating Model

- Evaluate the tuned Hyperparameter (Best Parameter)
- Evaluate the accuracy (Best Score)

## Best Performance

- Finding the best accuracy score out of all algorithms to determine the best performing model


RESULTS
Exploratory Data Analysis
Predictive Analysis
Interactive Analysis

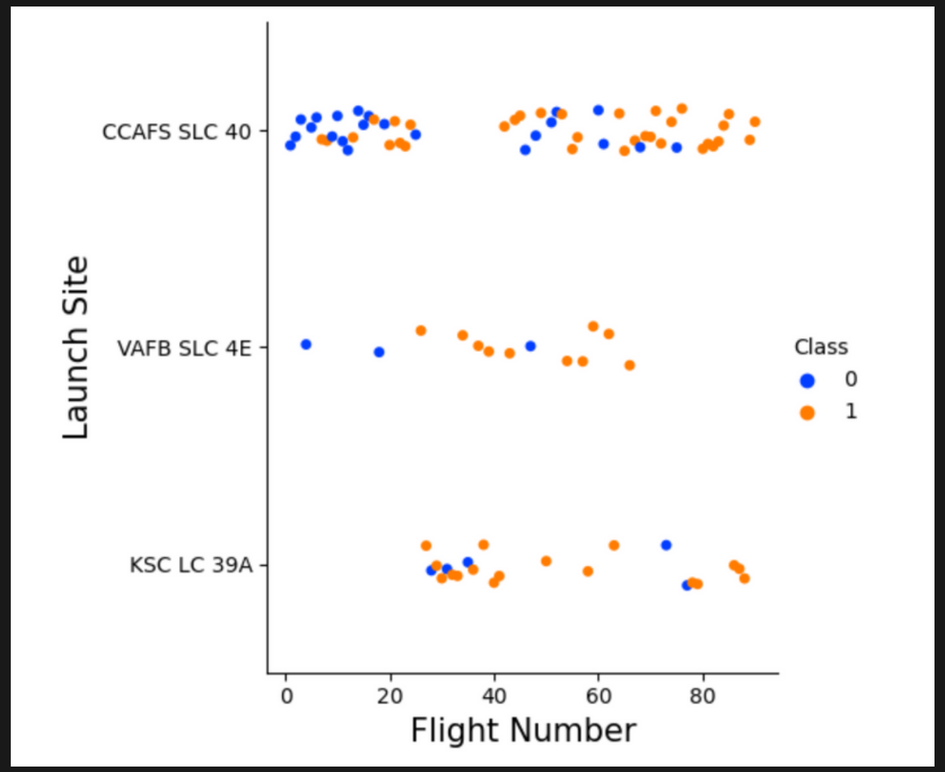
# Insights Drawn From EDA

# EDA - VISIUALIZATION

# Flight Number vs. Launch Site

**This figure shows that the success rate increases as the number of flights increases.**

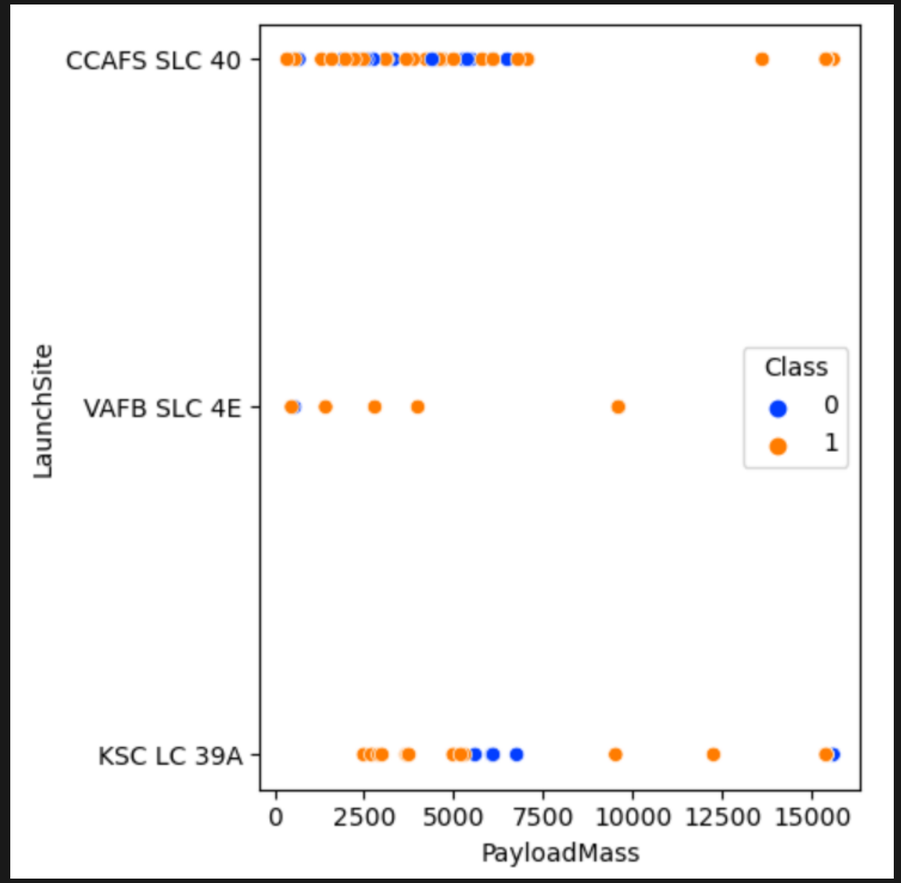
- Class 0 (blue) represents unsuccessful launches
- Class 1 (orange) represents successful launches
- The success rate has increased significantly since the 20th flight



# Payload vs. Launch Site

**This figure shows no clear correlation between payload mass and success rate of launch site.**

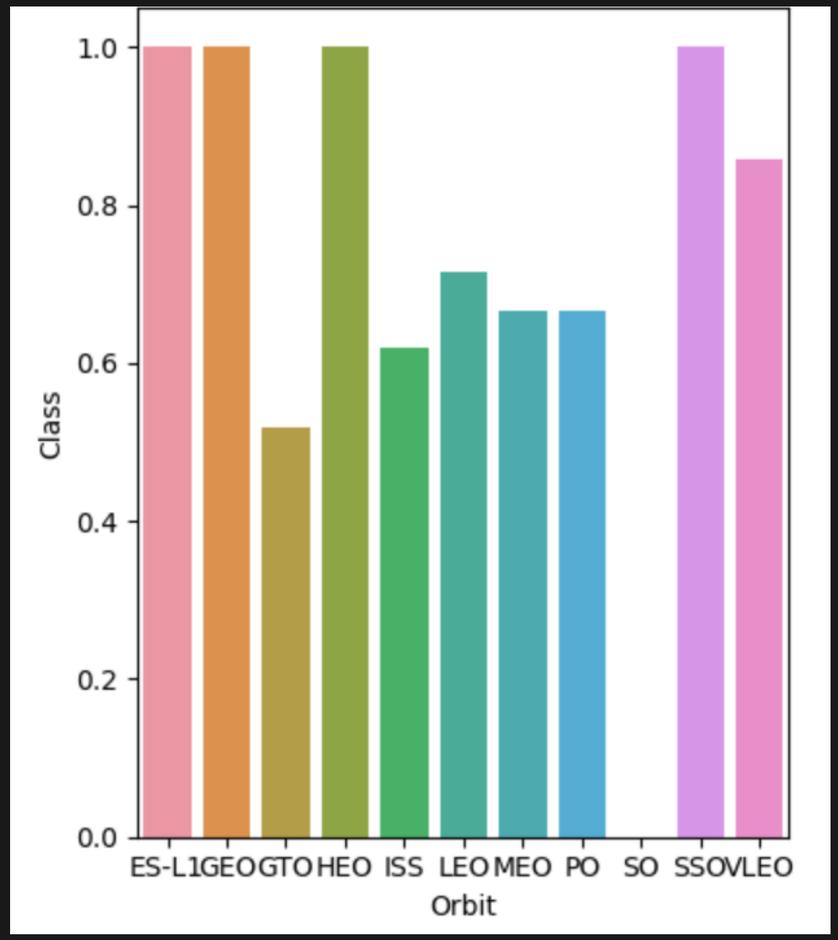
- Class 0 (blue) represents unsuccessful launches
- Class 1 (orange) represents successful launches
- As we notice, larger payload mass seems to have higher success rate, but it's difficult to determine since there's no clear correlation



# Success Rate vs. Orbit Type

**The following bar chart shows 4 of orbit types have 100% success rate**

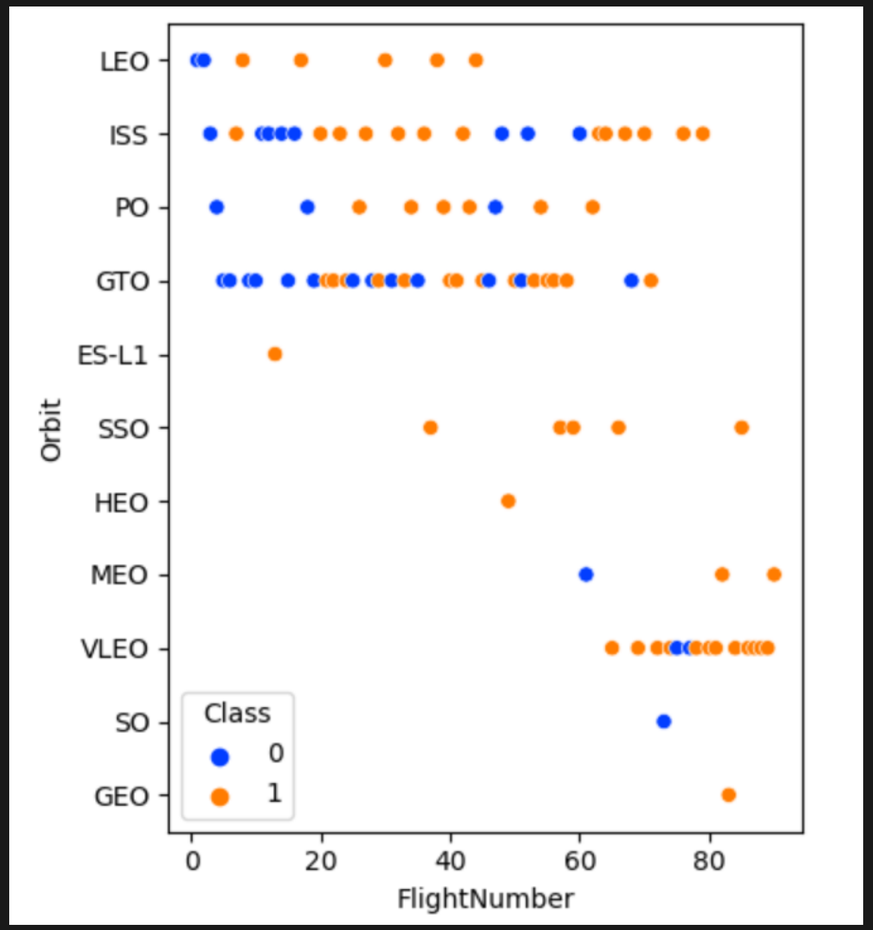
- Orbit types ES-L1, GEO, HEO, SSO have the highest success rates 100%
- As it shows, the success rate varies between different orbit types
- The orbit SO shows a failure since the success rate is 0%



# Flight Number vs. Orbit Type

**This figure shows significantly high success rates**

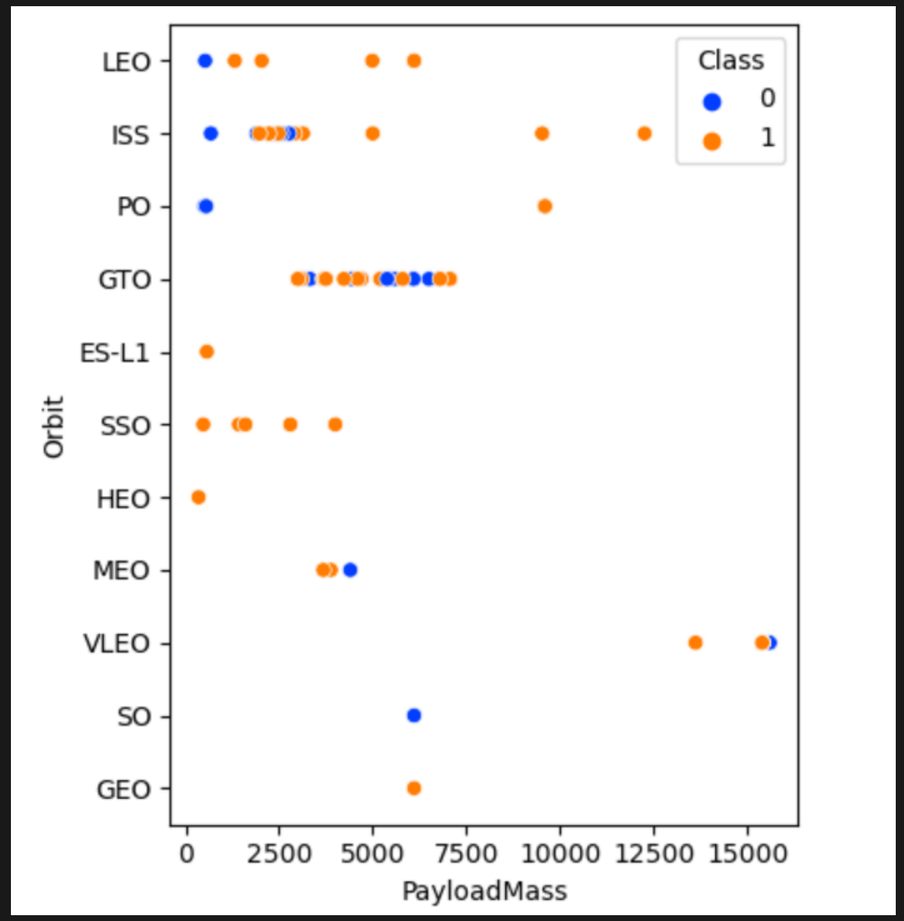
- Class 0 (blue) represents unsuccessful launches
- Class 1 (orange) represents successful launches
- ES -L1, HEO, GEO, show 100% success rate , and SSO shows the highest flight number with 100% success rate
- As others vary in success rate, GTO seems to have no relationship between flight number and success rate



# Payload vs. Orbit Type

**This scatter plot  shows the relationship of Orbit Type vs. Payload Mass**

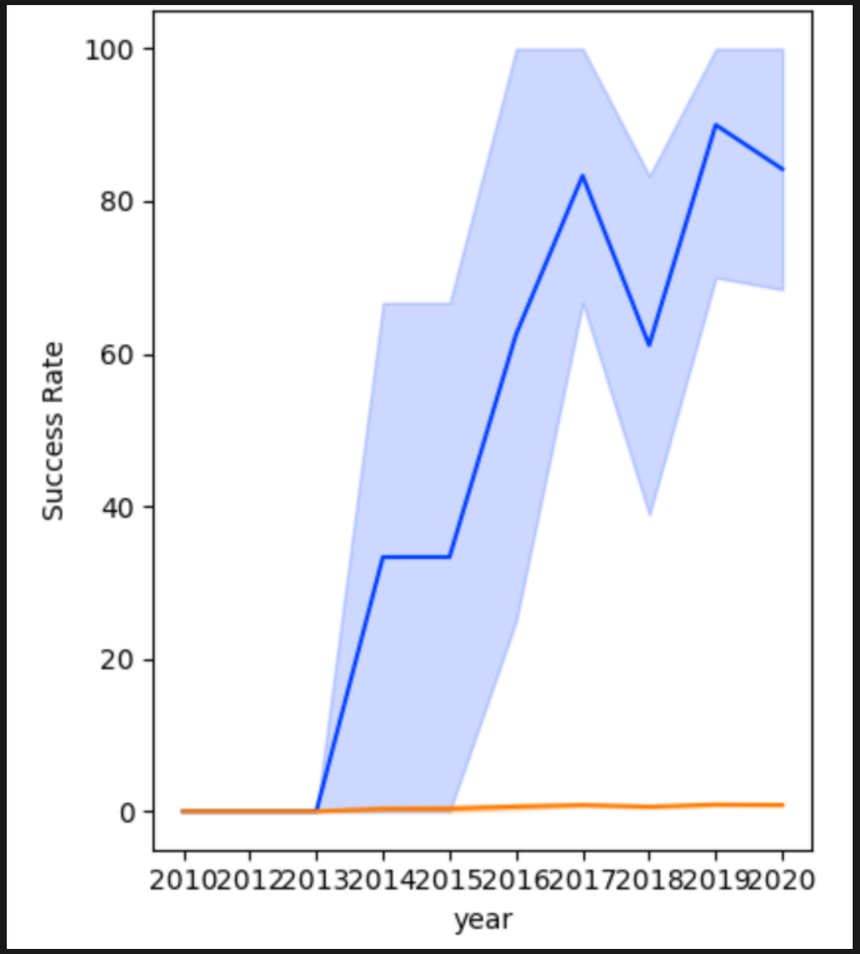
- Class 0 (blue) represents unsuccessful launches
- Class 1 (orange) represents successful launches
- With larger payloads, LEO, ISS, PO, VLEO are showing higher success rates
- As others vary in success rate, GTO on the other hand seems to have no clear relationship



# Launch Success Yearly Trend

**This figure shows the line chart of yearly average success rate**

- Between 2010 and 2013, all landings were unsuccessful as the success rate is 0
- After 2013, the success rate has notably increased
- There's a slight dip in the year of 2018
- Latest years show a huge success rates



# EDA - SQL

# All Launch Site Names

Using the keyword <DISTINCT>, only unique values should be displayed from the LAUNCH_SITE column of the SPACEXTBL table

**Query:**

```
%sql SELECT DISTINCT
LAUNCH_SITE FROM SPACEXTBL
```

**Output:**

[9]:

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

# Launch Site Names Begin with 'CCA'

**<LIMIT 5> fetches only 5 records, and the <LIKE> keyword is used to retrieve string values starting with ‘CCA’.**

**Query:**

```
%sql SELECT LAUNCH_SITE FROM
SPACEXTBL WHERE LAUNCH_SITE
LIKE 'CCA%' LIMIT 5
```

**Output:**

[10]:

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

# Total Payload Mass

Using the <SUM> keyword to calculate the total of the PAYLOAD_MASS__KG column, and the <WHERE> keyword filters the results to the customer NASA (CRS)

**Query:**

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

**Output:**

[11]:

| TOTAL_PAYLOAD_MASS |
| --- |
| 45596 |

# Average Payload Mass by F9 v1.1

The <AVG> keyword is used to calculate the average of the PAYLOAD_MASS__KG column, and the <WHERE> keyword filters the results to only the F9 v1.1 booster version

**Query:**

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS
AVG_PAYLOAD_MASS FROM SPACEXTBL
WHERE BOOSTER_VERSION = 'F9 v1.1'
```

**Output:**

[12]:

| AVG_PAYLOAD_MASS |
| --- |
| 2928.4 |

# First Successful Ground Landing Date

Using the <MIN> function to find out the earliest date in the column DATE, the <WHERE> clause filters the result to only if Landing__outcome is Success (ground pad)

**Query:**

```
%sql SELECT MIN(DATE) AS FIRST_SUCCESSFUL_LANDING FROM SPACEXTBL WHERE LANDING__OUTCOME = 'Success (ground pad)'
```

**Output:**

[13]:

| FIRST_SUCCESSFUL_LANDING |
| --- |
| 01-05-2017 |

# Successful Drone Ship Landing with Payload between 4000 and 6000

The <WHERE> clause filters the result to only if Landing__outcome is Success (drone ship), <AND> is used to perform additional function according to the <WHERE> clause, <BETWEEN> keyword allows 4000 < x < 6000 values to be selected

**Query:**

```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL
WHERE LANDING__OUTCOME = 'Success (drone ship)'
AND (PAYLOAD_MASS__KG_ BETWEEN 4000 AND
6000)
```

**Output:**

[14]:

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

# Total Number of Successful and Failure Mission Outcomes

**The <COUNT> keyword is used to calculate the total number of mission outcomes, and the <GROUP BY> keyword is used to group these results by the type of mission outcome**

**Query:**

```
%sql SELECT MISSION_OUTCOME, COUNT(*) AS total_number FROM SPACEXTBL GROUP BY MISSION_OUTCOME
```

**Output:**

[15]:

| Mission_Outcome | total_number |
|---|---|
| Failure (in flight) | 1 |
| Success | 98 |
| Success | 1 |
| Success (payload status unclear) | 1 |

# Boosters Carried Maximum Payload

The <DISTINCT> keyword is used to only show unique values, and the <MAX> keyword is used as a subquery to display the maximum value of payload

**Query:**

```
%sql SELECT DISTINCT(BOOSTER_VERSION) FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
```

**Output:**

[16]:

| 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 |

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

**Query:**

```
%sql SELECT LANDING__OUTCOME,
COUNT(LANDING__OUTCOME) AS TOTAL_NUMBER FROM
SPACEXTBL
WHERE julianday(substr(date,7)||'-'||substr(date,4,2)||'-
'||substr(date,1,2)) BETWEEN julianday('2010-06-04') AND
julianday('2017-03-20')
GROUP BY LANDING__OUTCOME ORDER BY TOTAL_NUMBER
DESC;
```

**Output:**

[18]:

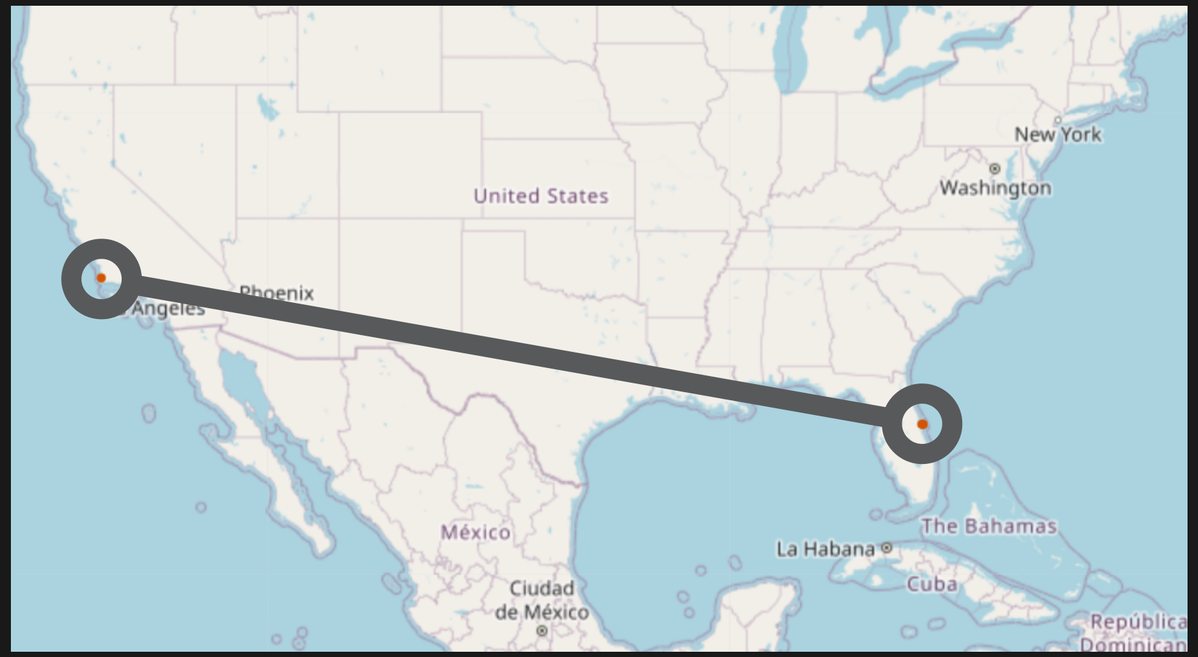
| Landing__Outcome | TOTAL_NUMBER |
|---|---|
| No attempt | 10 |
| Success (drone ship) | 5 |
| Failure (drone ship) | 5 |
| Success (ground pad) | 3 |
| Controlled (ocean) | 3 |
| Uncontrolled (ocean) | 2 |
| Failure (parachute) | 2 |
| Precluded (drone ship) | 1 |

The <WHERE> keyword is used to filter the results, <JULIANDAY(SUBSTR(DATE))> is used as sqlite function to return the date, <BETWEEN> is used to select the date from 04-06-2010 <x> 20-03-2017, Using the <ORDER BY, GROUP BY> keyword to sort the records by total number of landing, and <DESC> keyword to sort the records in descending order

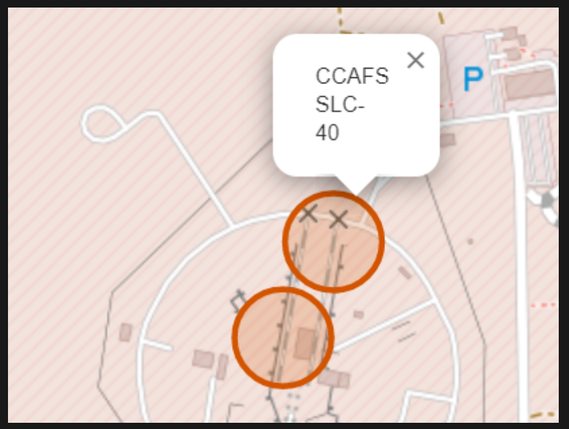
# Launches Sites Proximities Analysis

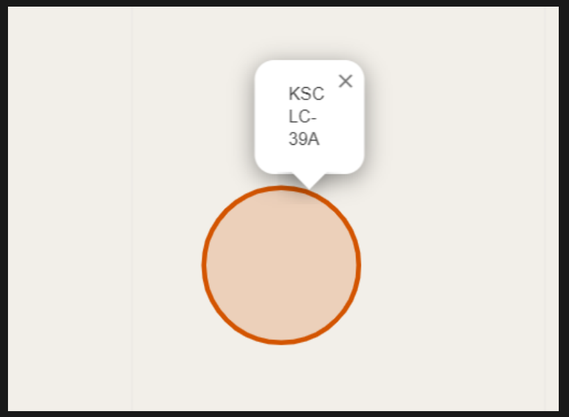
# All Launch Site On A Map

SpaceX Launches Sites all took place in The United States, we can observe that all launches are near the coast



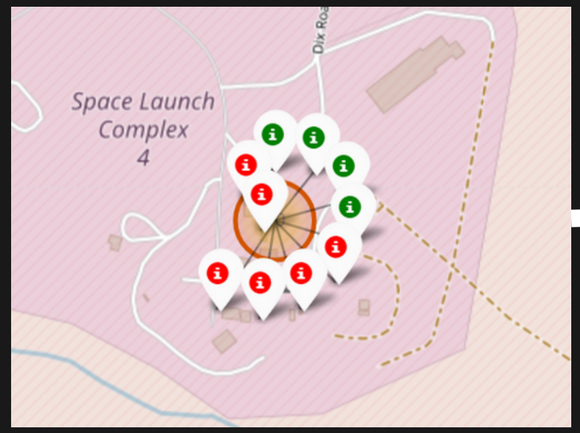




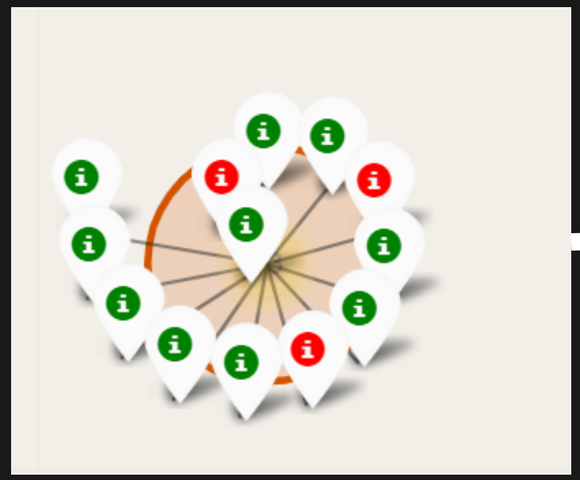



# The success/failed Launches

Launches have been grouped into clusters, Green icons represent successful launches, and red icons for failed launches

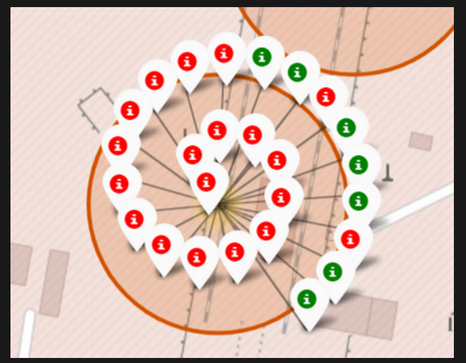


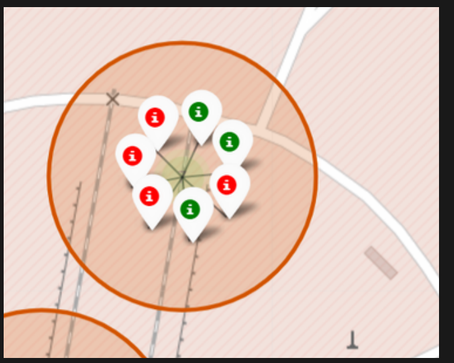

VAFB-SLC-4E



KSC-LC-39A

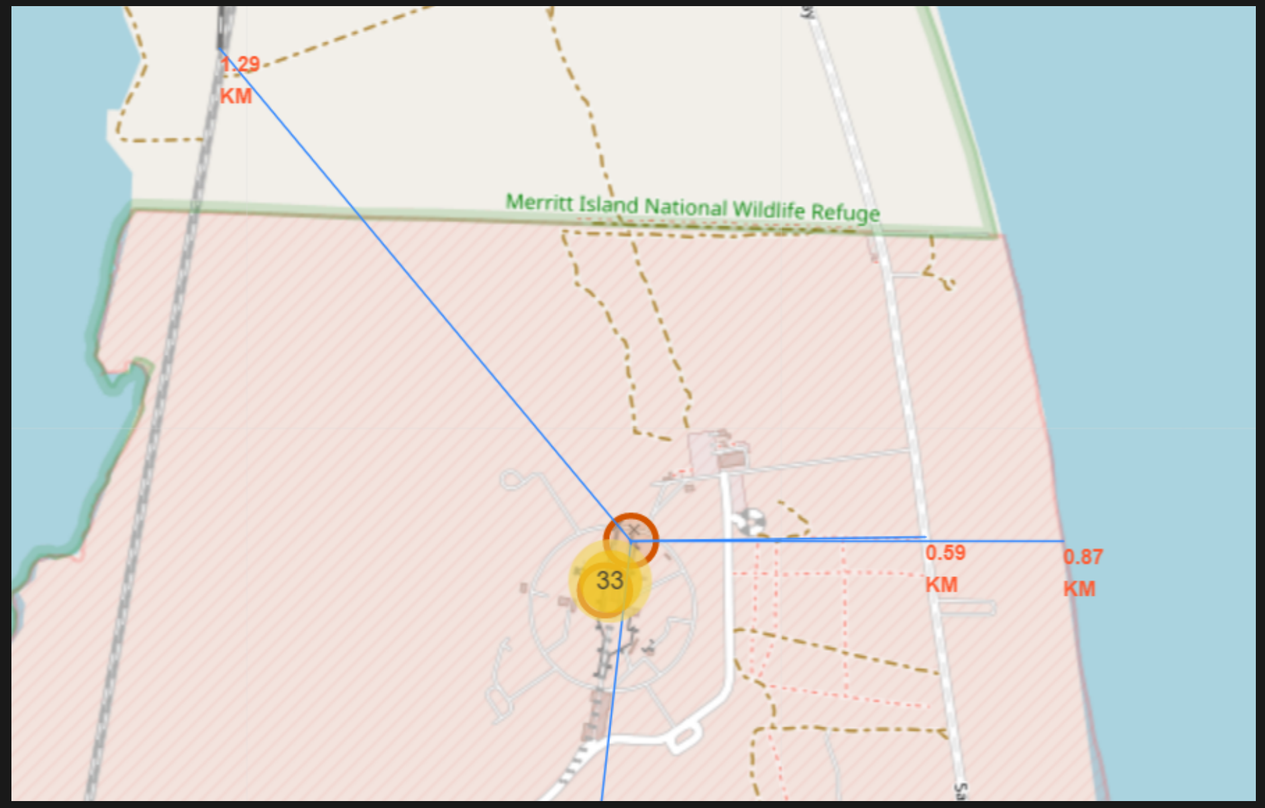
CCAFS-SLC-40

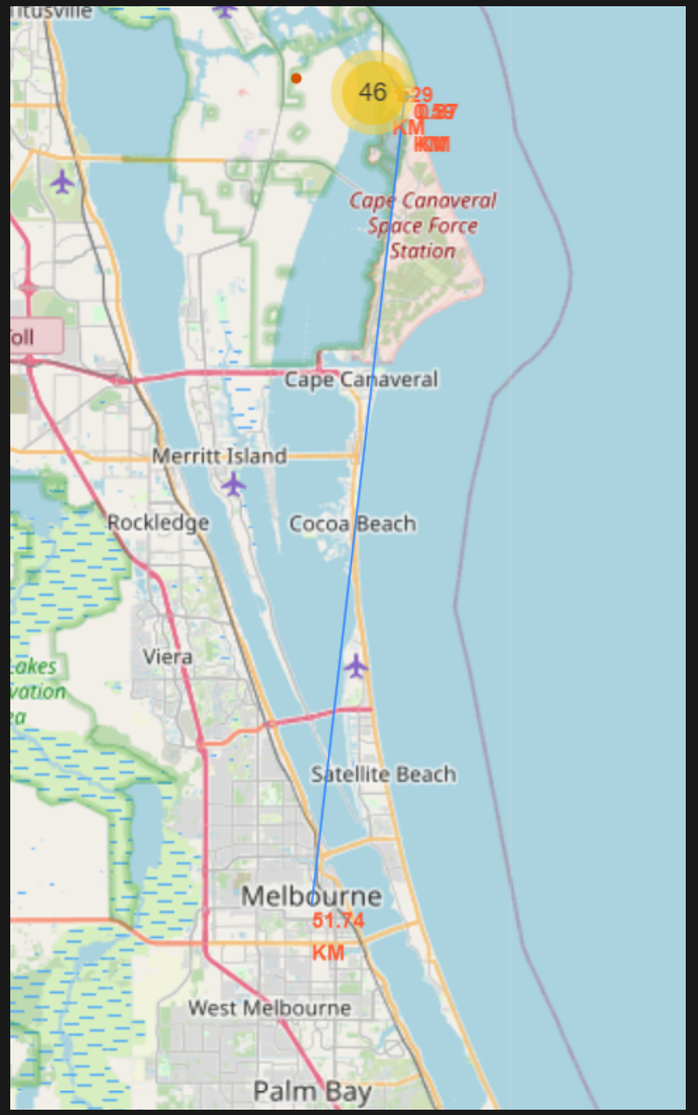





# The Distances Between A Launch Site To Its Proximities

Having CCAFS-SLC-40 Launch Site, we observe that the launch site is close to railways and highways , and is near coast line

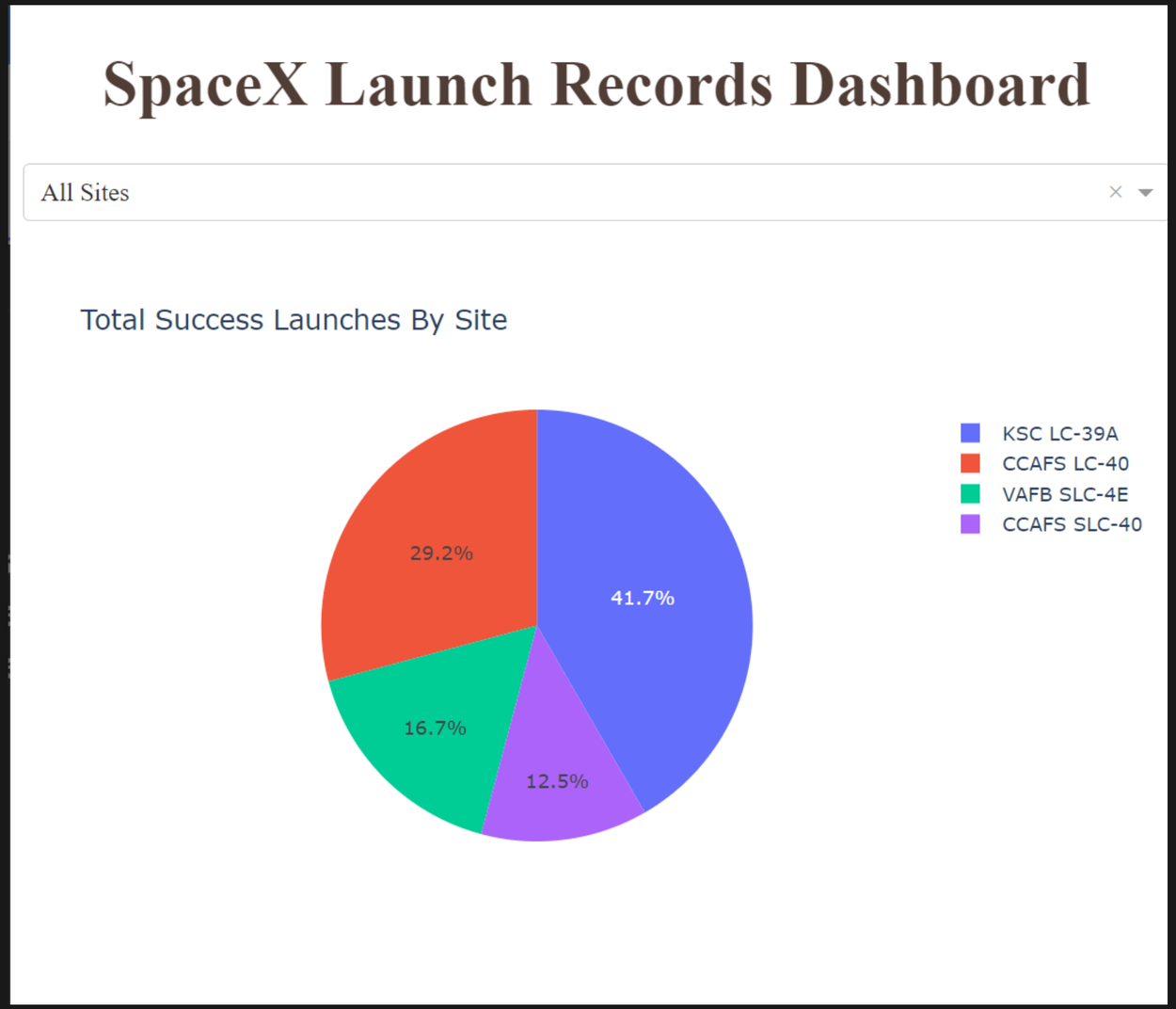


# Building A Dashboard - Plotly Dash

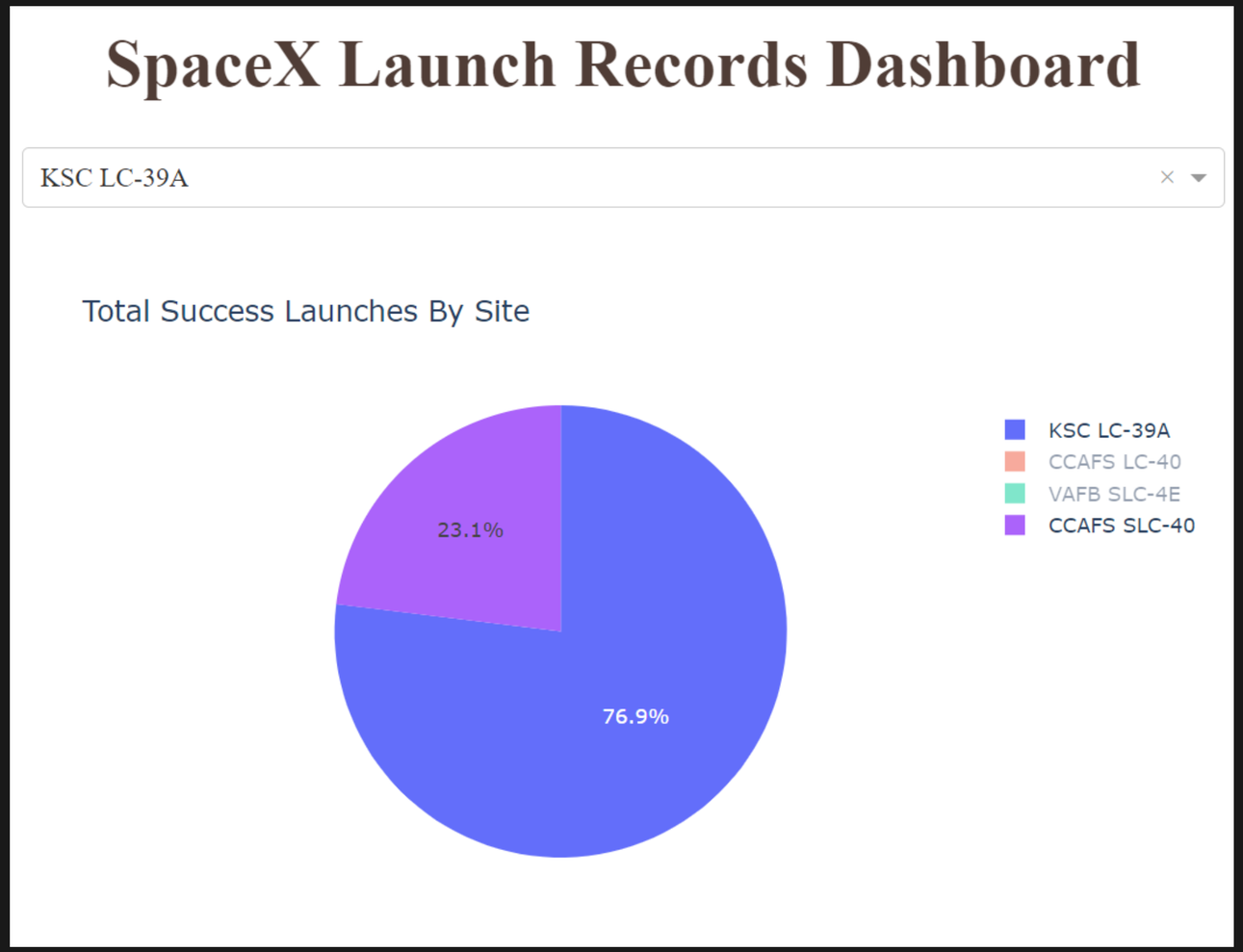
# Total Success Launches By All Sites

The launch site KSC LC-39 has the most successful launches among all sites, with 41.7% of the total successful launches



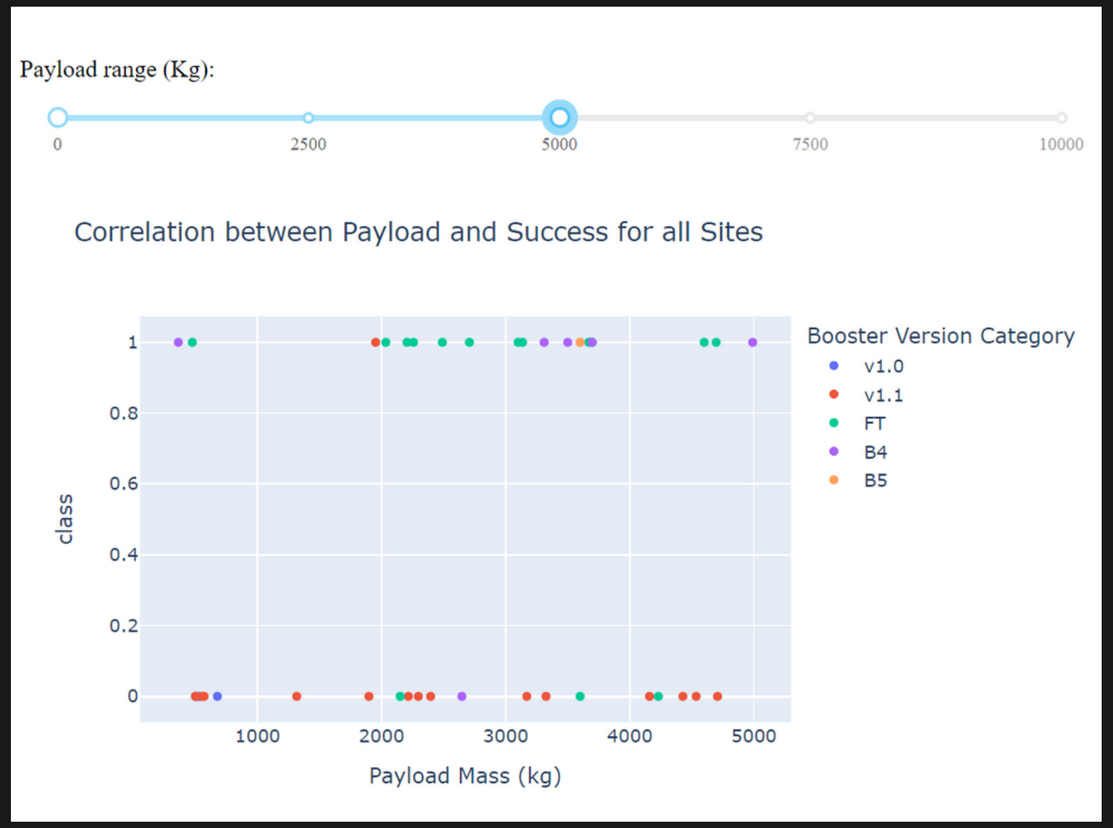
# Launch Site With The Highest Launch Success Rate

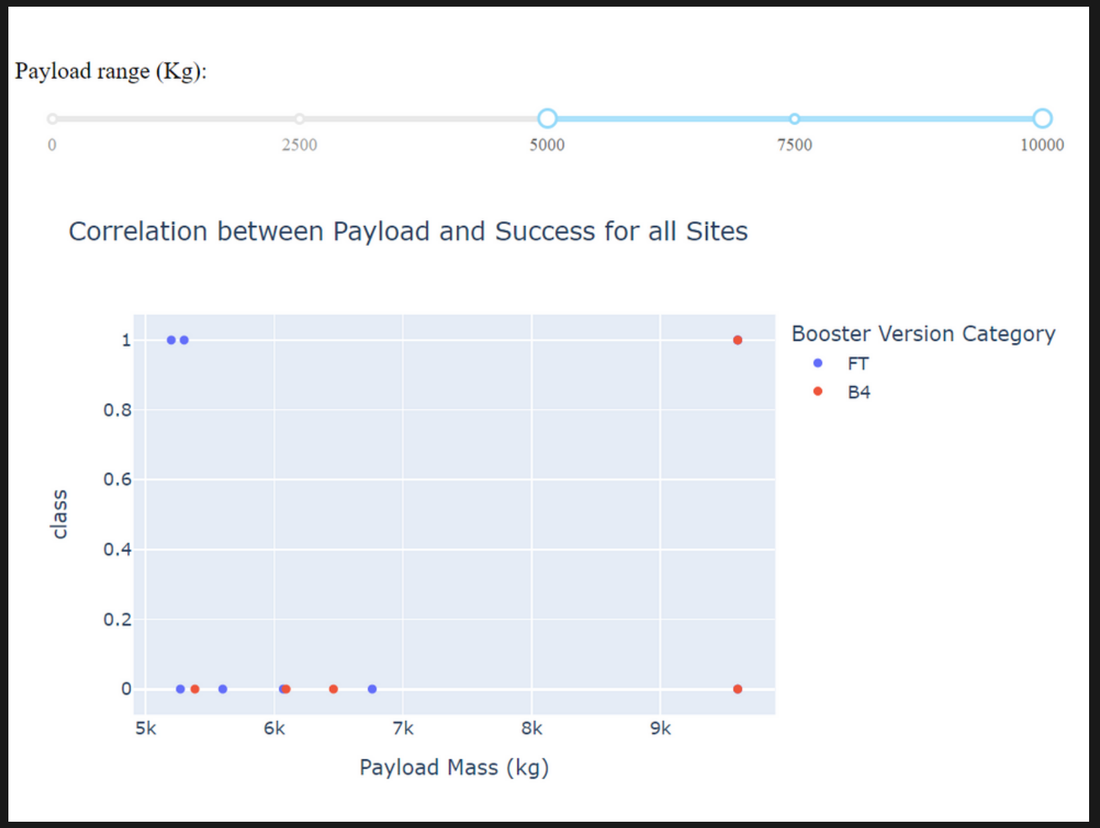
**KSLC-39A** also has the highest success rate, with (76.9%) success rate



# Payload vs. Launch Outcome Scatter Plot for All Sites

**Figures below show that the launch success rate for Low Weighted Payloads (0-5000 kg) is higher than that of Heavy Weighted Payloads (5000-10000 kg)**

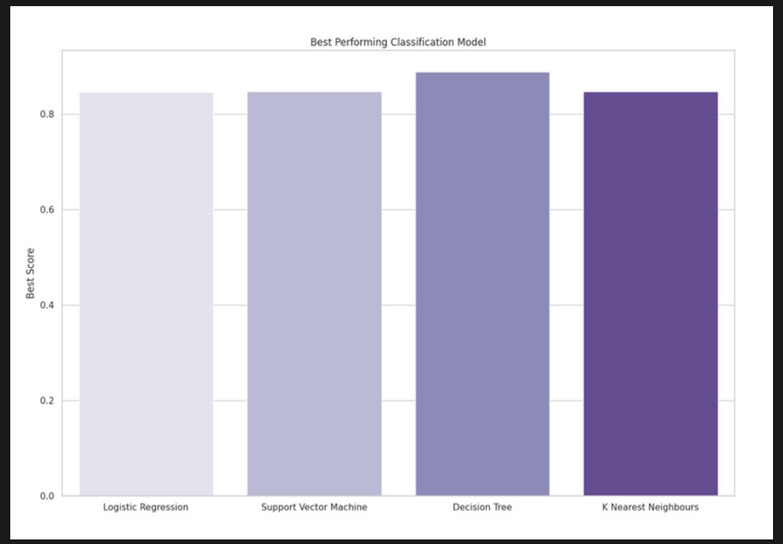


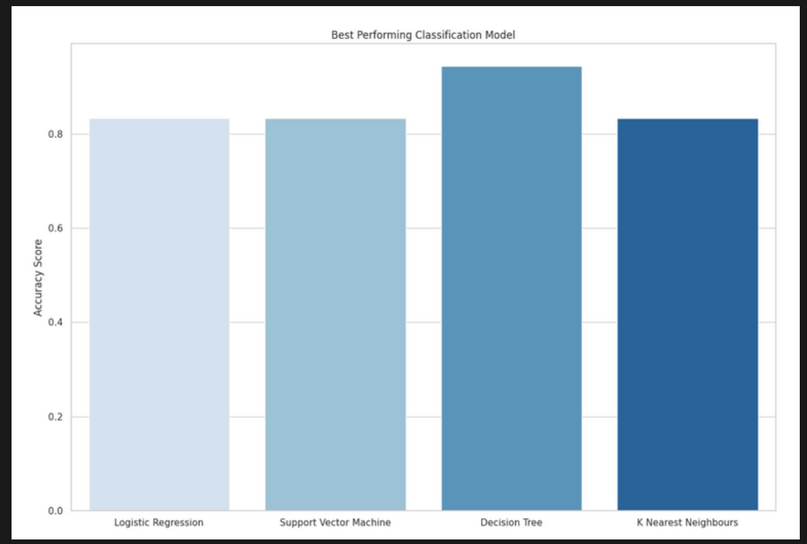



# Predictive Analysis - Classification

# Classification Accuracy

**The Accuracy Score and The Best Score for (Logistic Regression - SVM - Decision Tree - KNN) Classification Models**




The Decision Tree has the highest Classification Accuracy

- With Accuracy Score of : 94.4%
- With Best Score of : 88.9%

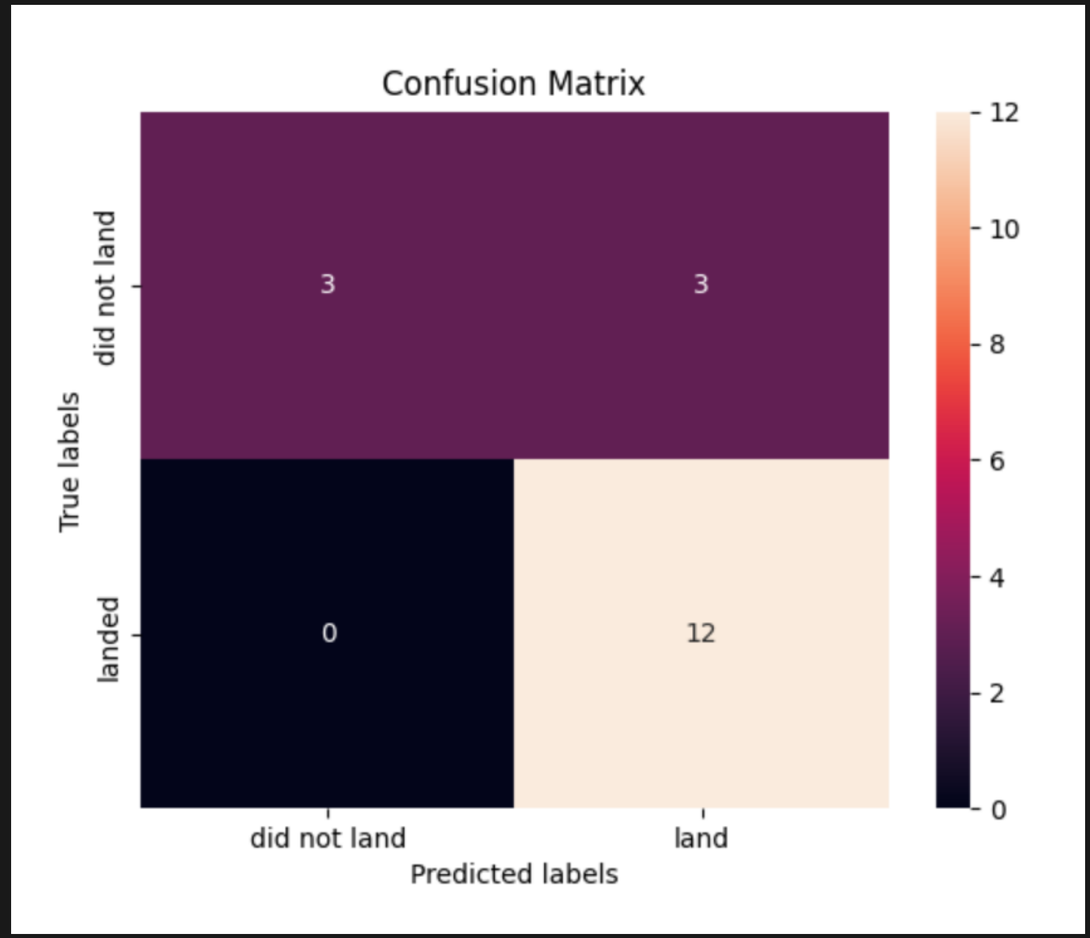
| | Models | Accuracy Score | Best Score |
|---|---|---|---|
| 0 | Logistic Regression | 0.833333 | 0.846429 |
| 1 | Support Vector Machine | 0.833333 | 0.848214 |
| 2 | Decision Tree | 0.944444 | 0.889286 |
| 3 | K Nearest Neighbours | 0.833333 | 0.848214 |

# Confusion Matrix

**The following figure shows the Confusion Matrix of Decision Tree Classification Model,  as indicated, only 3 classes were falsely marked**

Decision Tree Confusion Matrix

- True Positive : 3
- False Positive: 3
- False Negative : 0
- True Negative : 12



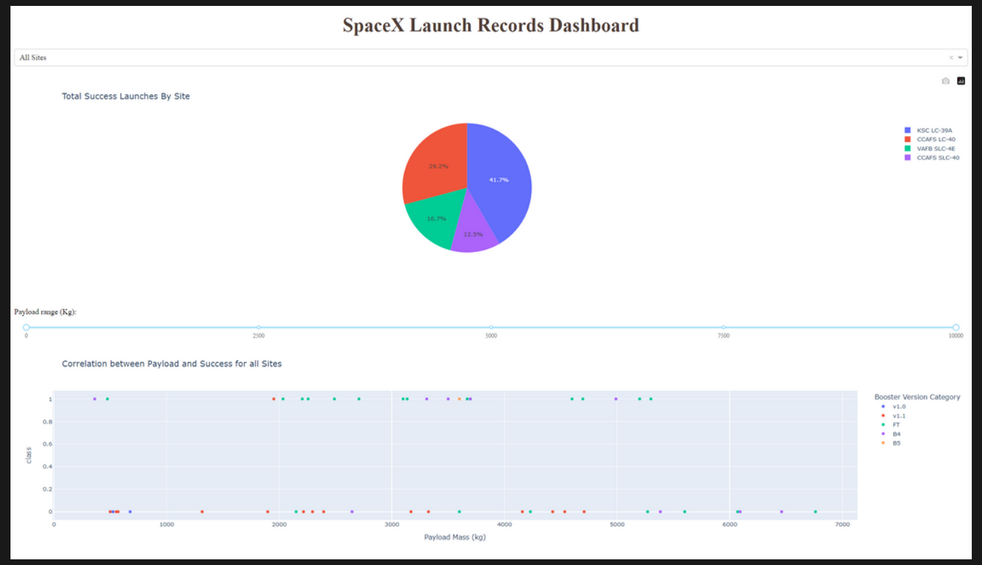
# Conclusions

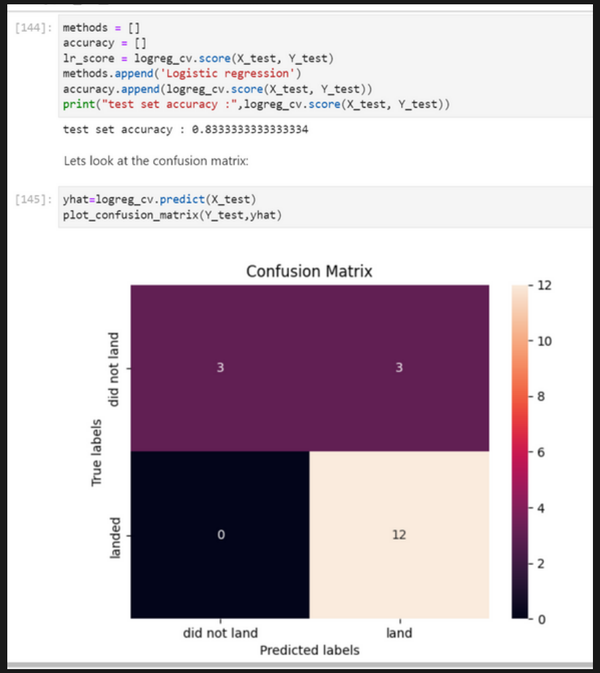
# Conclusion

- As the number of flights increases, the success rate increases, and latest flights are the most successful
- Orbital types SSO, HEO, GEO, and ES-L1 have the highest success rate (100%)
- The launch site is close to railways, highways, and coastline, but far from cities
- KSLC-39A has the highest number of launch successes and the highest success rate among all sites
- The launch success rate of low weighted payloads is higher than heavy weighted payloads
- In the predictive analysis, Decision Tree has the best accuracy score out of all classification models

# Appendix

🔗 GITHUB LINK



# Thank You!