# Winning Space Race With Data Science: IBM Data Science Capstone Project

Tanya Rawat

23.09.2021

# Outline

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

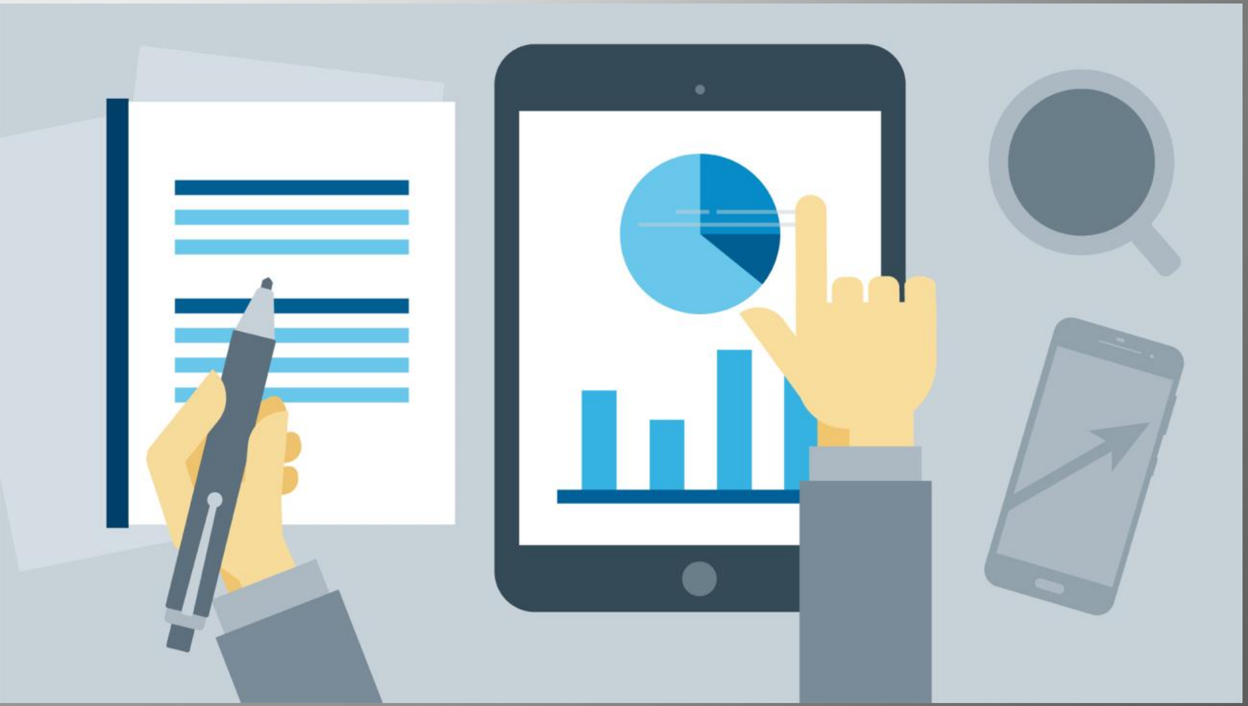
# Executive Summary

**Summary of methodologies**

- Data collection
- Data wrangling
- EDA with data visualization
- EDA with SQL
- Interactive Visual Analytics with Folium
- Interactive Dashboard with Plotly Dash
- Predictive Analysis

**Summary of all results**

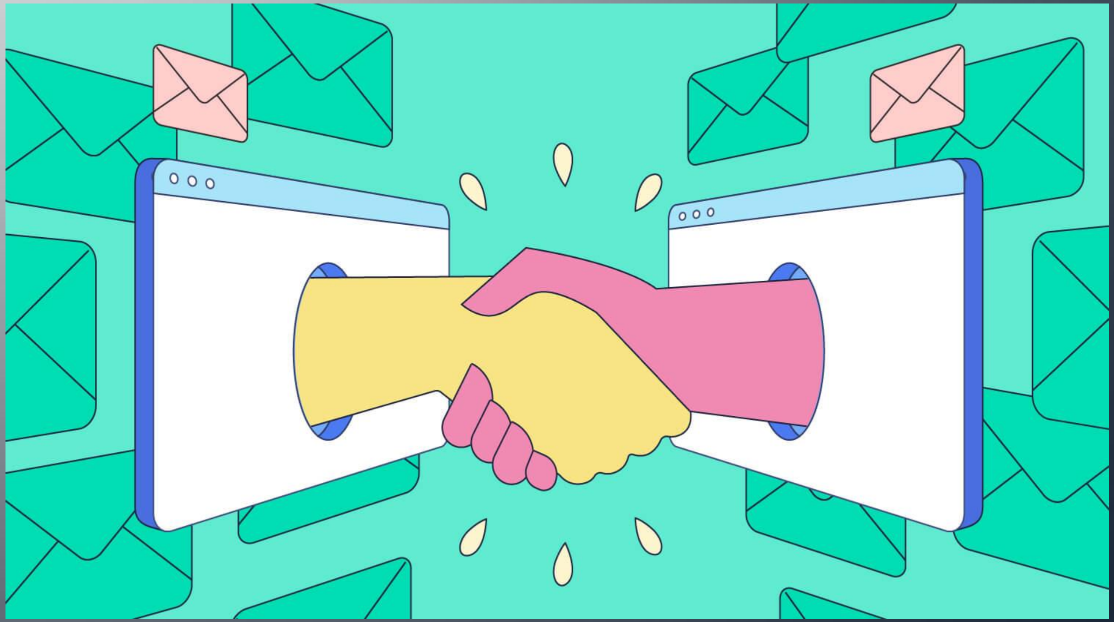
- Findings from Exploratory data analysis
- Interactive analytical demonstrations with screenshots



# Introduction

- **SpaceX** advertises Falcon9 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. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch

- The **QUESTION** is
  - Whether the Falcon9 first stage will land successfully
  - What will influence its landing
  - How will different components of rocket influence the outcome of successful landing
  - What are the conditions SpaceX must attain in order to achieve the successful landing rate in the best possible way



# Methodology

- **Data collection methodology**
  - Data was collected using SpaceX REST API
  - Web Scraping using Wikipedia pages
- **Perform data wrangling**
  - Using One Hot Encoding and Dropping irrelevant columns
- **Perform exploratory data analysis (EDA) using visualization and SQL**
  - Plotting  Scatter and ar plots between different variable to show relationship/patterns of data.
- **Perform interactive visual analytics using Folium and Plotly Dash**
- **Perform predictive analysis using classification models**
  - To build, tune and evaluate classification models

# Methodology

IBM

## DATA COLLECTION

1. Request to the SpaceX REST API
2. API returns data in json file
3. Clean the requested data

## DATA WRANGLING

1. Extract a Falcon 9 launch records HTML table from Wikipedia
2. Web scrap using BeautifulSoup
3. Parse the table and convert it into a Pandas data frame

# Data collection : SpaceX REST API

GitHub Link

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

**1.Requesting rocket launch data from SpaceX API**

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

**2.Getting Response from API**

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_
data =pd.json_normalize(response.json())
```

**3.Converting Response to a .json file**

```
getBoosterVersion(data)
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

**4. Apply previously made functions to clean data**

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

**5.Construct dataset using the data obtained. Then combine the columns into a dictionary.**

```
booleans = LaunchData['BoosterVersion']!='Falcon 1'
data_falcon9 = LaunchData[booleans]
```

**6.Finally remove the Falcon 1 launches keeping only the Falcon 9 launches**

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

**7. Dataframe to csv file**

# Data Collection : Web Scraping

[GitHub Link]

```
import requests
r=requests.get(static_url).text
```

1. Use requests.get() method with the provided static_url

```
soup = BeautifulSoup(r,'html5lib')
```

2. Create a BeautifulSoup object from HTML response

```
html_tables = soup.find_all('tr')
```

```
column_names = []

for row in first_launch_table.find_all('th'):
    name = extract_column_from_header(row)
    if (name != None and len(name) > 0):
        column_names.append(name)

column_names
```

3. Collect all relevant column names from the HTML table header

```
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']=[]
```

4. Create a data frame by parsing the launch HTML tables

```
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"):
```

5. Add launch records extracted from table rows to launch_dict

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

6. Converting dictionary to dataframe

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

7. Converting dictionary to dataframe

# Data Wrangling

GitHub Link

In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident; for example, 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 on a drone ship False ASDS means the mission outcome was unsuccessfully landed on a drone ship. We mainly convert those outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful.

**Calculate the number of launches on each site**

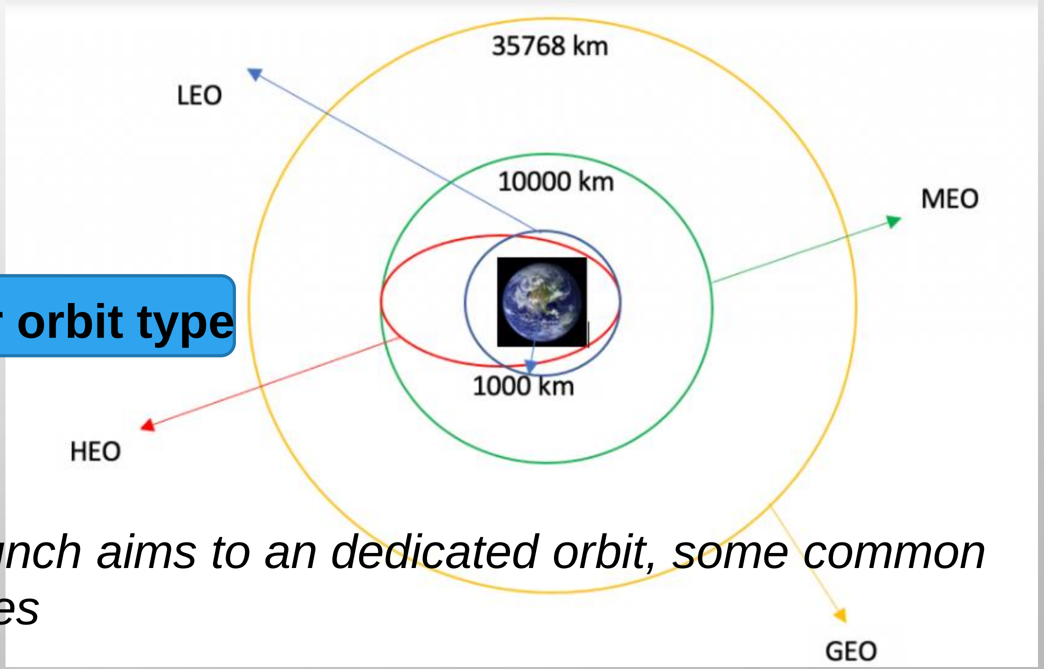
**Calculate the number and occurrence of each orbit**

**Calculate the number and occurrence of mission outcome per orbit type**

**Create a landing outcome label from Outcome column**

**Determining the success rate**

**Converting dictionary to dataframe**



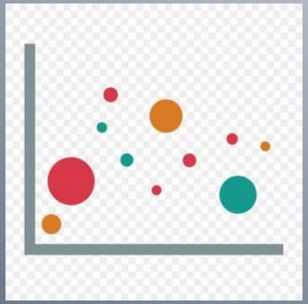

*Each launch aims to an dedicated orbit, some common orbit types*

# EDA with Data Visualization

[GitHub Link]

## Scatter Graphs

- Flight Number VS. Payload Mass
- Flight Number VS. Launch Site
- Payload VS. Launch Site
- Orbit Type VS. Flight Number
- Payload VS. Orbit Type
- Orbit Type VS. Payload Mass

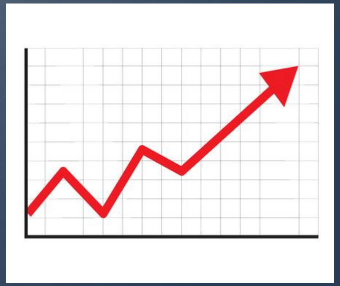


## Bar Graph

- Mean VS. Orbit

## Line Graph

- Average Success Rate VS. Year



# EDA with SQL

GitHub Link

- 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. USE A SUBQUERY
- LIST THE FAILED LANDING_OUTCOMES IN DRONE SHIP, THEIR BOOSTER VERSIONS, AND LAUNCH SITE NAMES FOR IN YEAR 2015
- 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

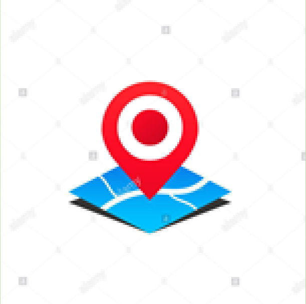
# Build an Interactive Map with Folium

[GitHub Link]

## Mark all launch sites on a map

**circle** as folium.Circle object was used to add a highlighted circle area with text label on all Launch Sites. **dist_marker** as folium.Marker object was added to show the distance.

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



We assigned the dataframe **launch_outcomes(failures, successes) to classes 0** and **1 with Green and Red markers** on the map in a MarkerCluster()

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

Using **Haversine's formula**, that is, the shortest distance over the earth's surface we calculated the distance from the Launch Site to various landmarks to find various trends about what is around the Launch Site to measure patterns. Lines are drawn on the map to measure distance to landmarks

[GitHub Link to Dashboard Code]

# Build a Dashboard with Plotly Dash

**THE DASHBOARD IS BUILT WITH FLASK AND DASH WEB FRAMEWORK.**

- **FIRST WE ADD A DROPDOWN LIST TO ENABLE LAUNCH SITE SELECTION.**
- **THEN WE ADD A PIE CHART TO VISUALIZE THE TOTAL SUCCESSFUL LAUNCHES COUNT FOR ALL SITES.**
- **ADDED A SLIDER TO SELECT PAYLOAD RANGE.**
- **LASTLY, ADDED A SCATTER CHART TO SHOW THE CORRELATION BETWEEN PAYLOAD MASS(IN KGS) AND LAUNCH SUCCESS.**

Flask

# Predictive Analysis

GitHub Link

## Building a Model

- First we created a NumPy array from the column Class and assigning it to variable Y.
- Standardize the data in X
- Split the data X and Y into training and testing data using train_test_split.

## Evaluation

- Now we found the best parameters for different models like Logistic Regression, Support Vector Machine(SVM), Decision Tree and K-Nearest Neighbors.
- Then we will find the accuracy on test data for each models.
- We will also plot Confusion Matrix for each models.

## Classification Model

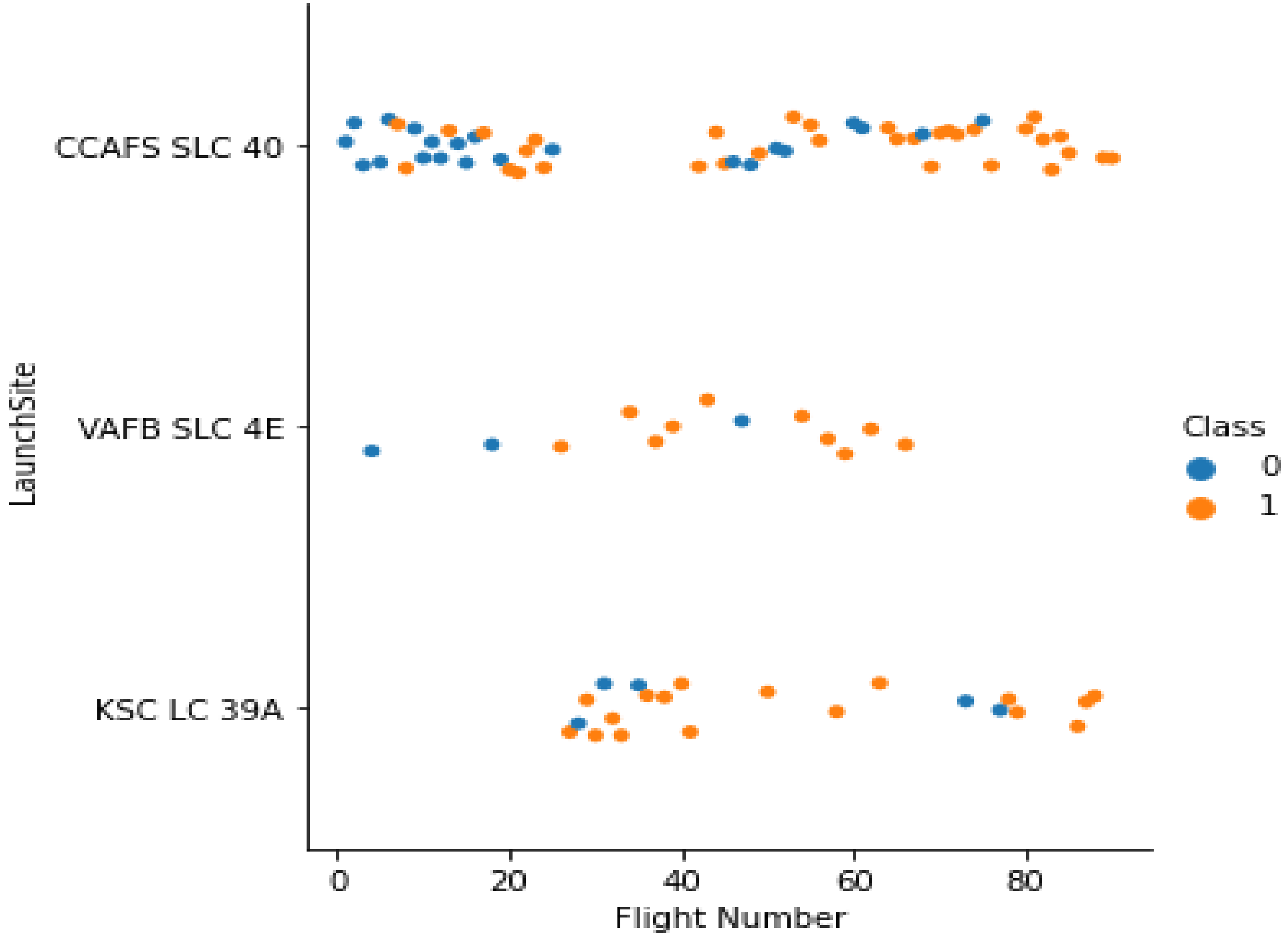
- Model with best accuracy will be chosen as the best performing classification model

# Results

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


# EDA with VISUALIZATION

# Flight Number vs. Launch Sites

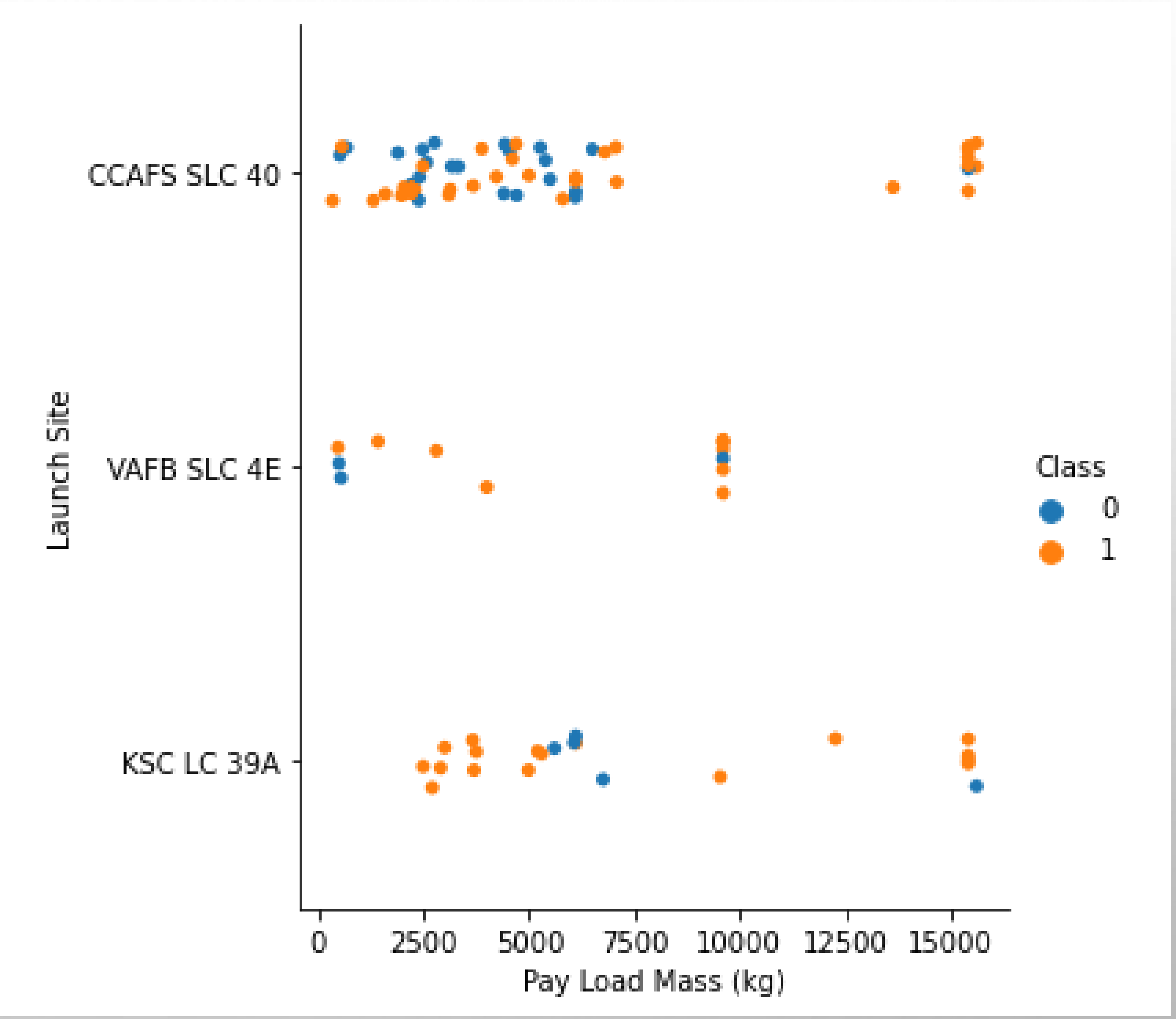



The larger the number of launch attempts, the greater the success rate

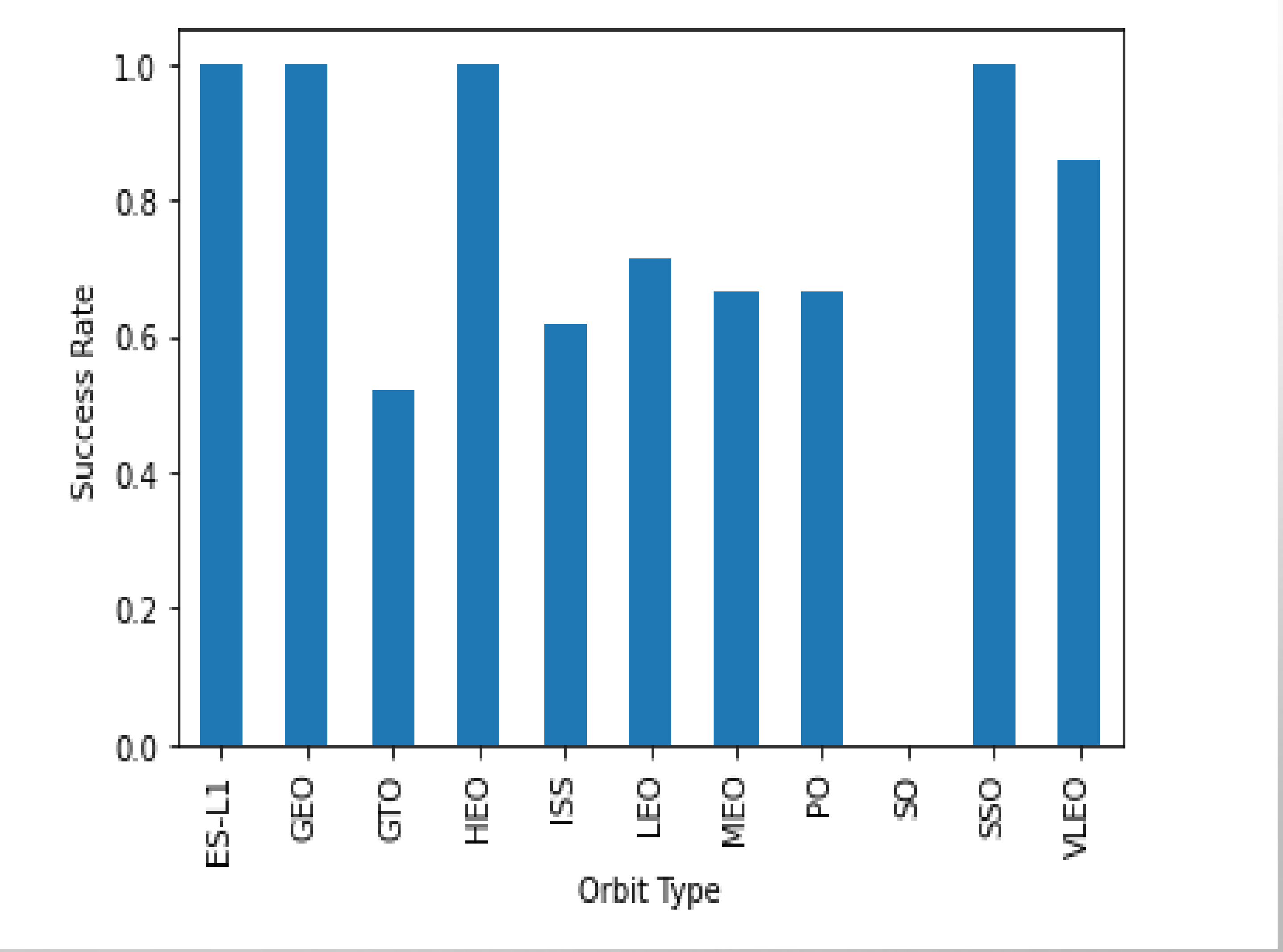
# Payload Mass vs. Launch Site

The success rate for CCAFS SLC 40 increase when payload mass is more than 10000kg

The success rate for VAFB SLC 4E and KSC LC 39A is not clear with respect to payload mass.

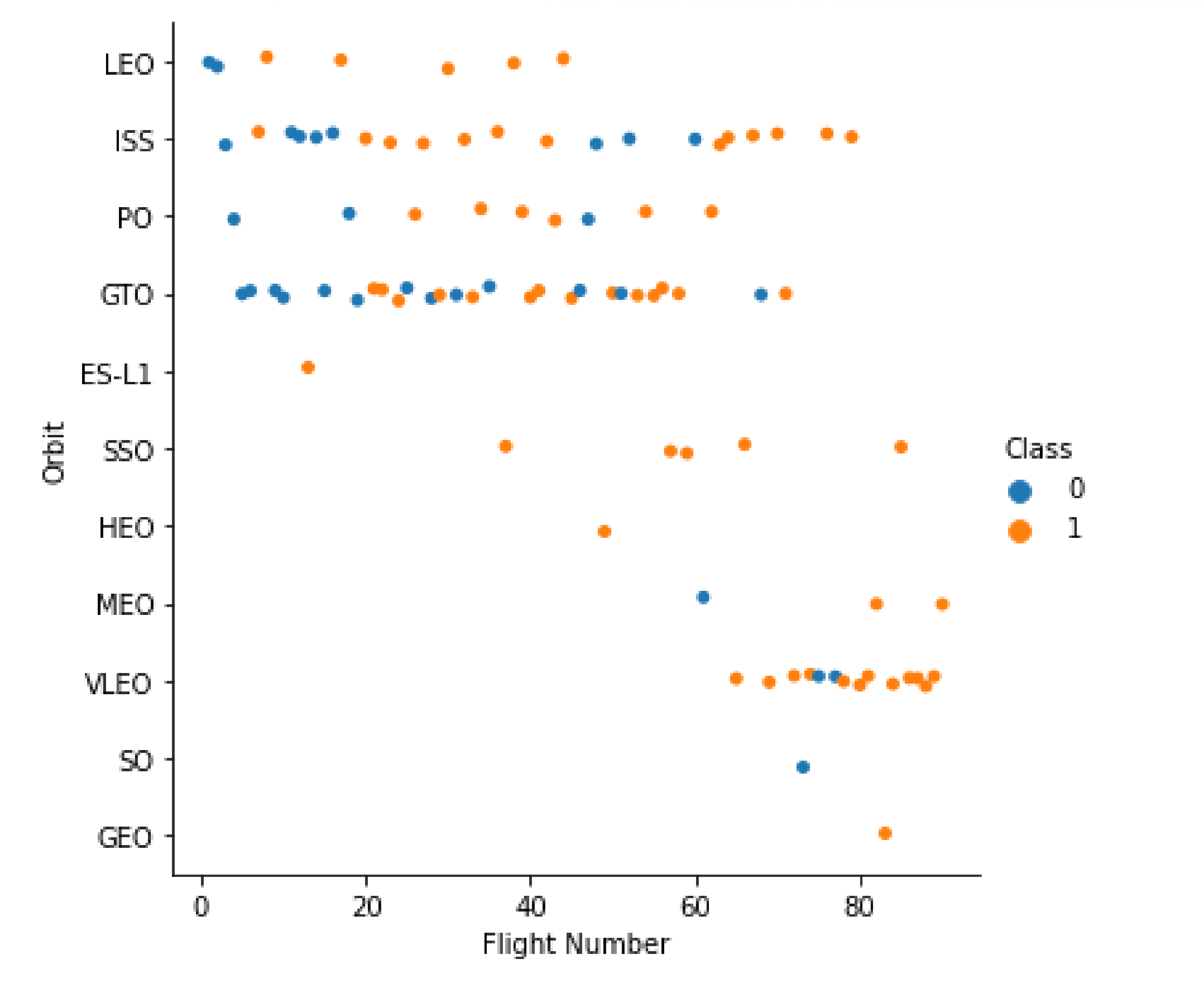


# Success Rate vs. Orbit Type



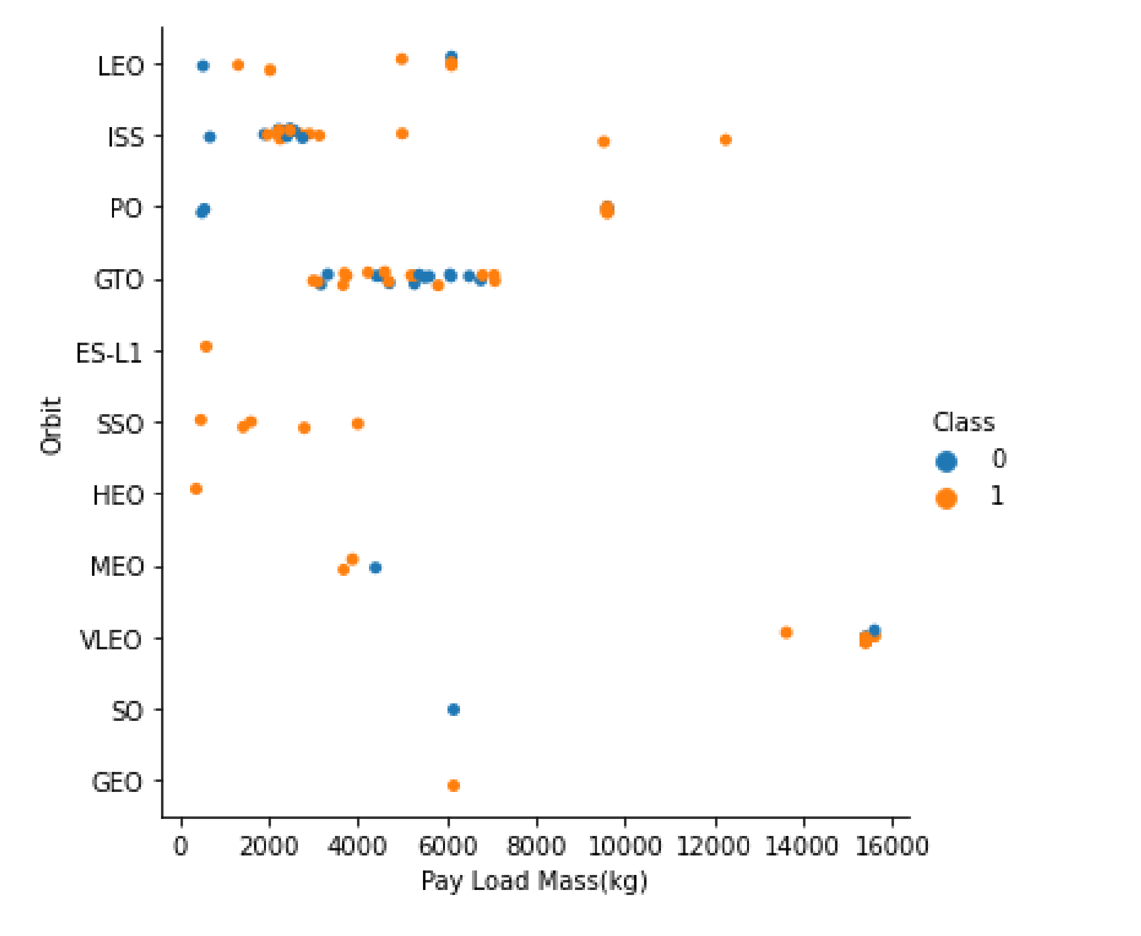

Orbit Type ES-L1,GEO,HEO,SSO has the best Success Rate

# Flight Number vs. Orbit Type



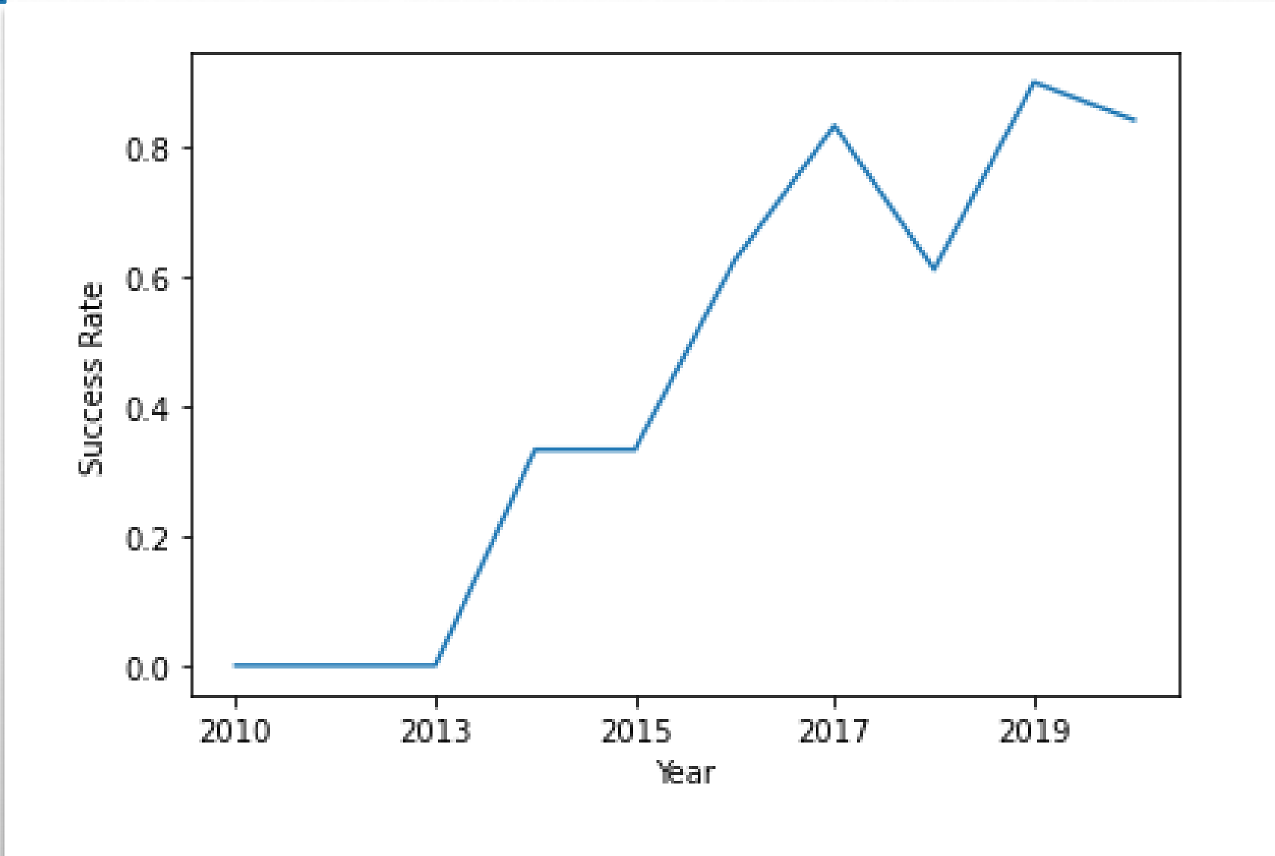

In the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

# Payload vs. Orbit Type

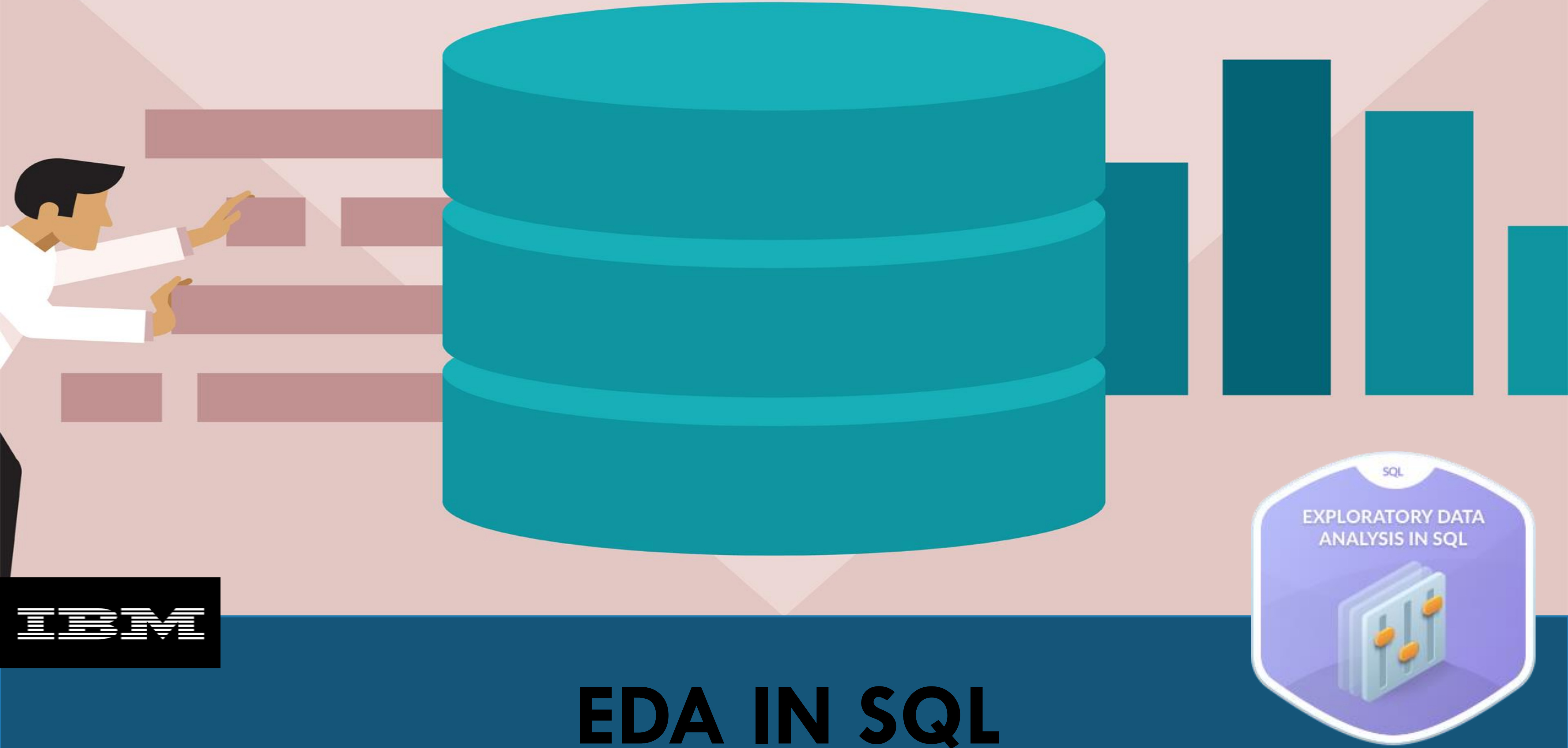



Heavy payloads have a negative influence on GTO orbits and positive on GTO and Polar LEO (ISS) orbits.

# Launch Success Yearly Trend




The success rate since 2013 kept increasing till 2020




# EDA IN SQL

# All Launch Site Names

## SQL Query

```
%%sql
SELECT DISTINCT launch_site from SpaceXtbl
```

## Result

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

## Explanation

Here DISTINCT is used to show all the UNIQUE launch sites in SpaceXtbl

# Launch Site Names Begin with 'CCA'

## SQL Query

```
%%sql
SELECT * from SpaceXtbl
where launch_site like 'CCA%' limit 5
```

## Explanation

Here we have used LIMIT 5 as question suggest that only TOP 5 launch sites should be displayed. Also we have used launch_site LIKE ‘CCA%’ so that only launch sites starting from CCA are displayed.

## Result

| 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 | 07:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525 | LEO (ISS) | NASA (COTS) | Success | No attempt |
| 2012-10-08 | 00:35:00 | 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 Query**

```
%%sql
SELECT SUM(PAYLOAD_MASS__KG_) from SpaceXtbl
where Customer = 'NASA (CRS)'
```

**Result**

| 1 |
| --- |
| 45596 |

## Explanation

Here we use SUM with Payload_mass__kg_ which will provide us with Total Payload Mass. Also WHERE clause is used so that only ‘NASA (CRS)’ Customer is accounted for.

# Average Payload Mass by F9 v1.1

## SQL Query

```
%%sql
SELECT AVG(PAYLOAD_MASS__KG_) from SpaceXtbl
where Booster_version like 'F9 v1.1%'
```

## Result

| 1 |
|---|
| 2534 |

## Explanation

Here we use AVG with Payload_mass__kg_ which will provide us with Average Payload Mass. Also WHERE clause is used so that only ‘F9 v1.1%’ Booster_version is accounted for.

# First Successful Ground Landing Date

## SQL Query

```
%%sql
SELECT min(Date) from SpaceXtbl
where Landing__outcome like 'Success (ground pad)'
```

## Result

| 1 |
|---|
| 2015-12-22 |

## Explanation

Here we use MIN with Date which will provide us with minimum date in the column. WHERE clause is used so that only 'Success (ground pad)' in Landing__outcome column is considered.

# Successful Drone Ship Landing with Payload between 4000 and 6000

## SQL Query

```
%%sql
SELECT BOOSTER_VERSION from SpaceXtbl
where Landing__outcome = 'Success (drone ship)' and
payload_mass__kg_ BETWEEN 4000 AND 6000
```

## Result

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

## Explanation

Here we use WHERE clause so that only Successful Drone Ship landing is considered form landing__outcome column.
‘BETWEEN’ and ‘AND’ clause is used to filter conditions
payload_mass_kg_ > 4000 AND payload_mass_kg_ < 6000

# Total Number of Successful and Failure Mission Outcomes

## SQL Query

```
%%sql
SELECT count(CASE when Mission_outcome like 'Success%' THEN 1 END ) as Success,
count(CASE when Mission_outcome like 'Failure%' THEN 1 END) as Failure from SpaceXtbl
```

## Result

| success | failure |
|---|---|
| 100 | 1 |

## Explanation

Here  we use 'COUNT' and 'CASE' to first segregate Success and Failure and then count them.

# Boosters Carried Maximum Payload

## SQL Query

```
%%sql
SELECT Booster_version from SpaceXtbl
where payload_mass__kg_ = (select max(payload_mass__kg_) from Spacextbl)
```

### Explanation

Here  we use sub-query and MAX to select all the Booster_version with maximum Payload Mass

## Result

| 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 |

# 2015 Launch Records

## SQL Query

```
%%sql
SELECT BOOSTER_VERSION, launch_Site from SpaceXtbl
where landing__outcome = 'Failure (drone ship)' AND year(Date) = 2015
```

## Result

| booster_version | launch_site |
|---|---|
| F9 v1.1 B1012 | CCAFS LC-40 |
| F9 v1.1 B1015 | CCAFS LC-40 |

## Explanation

Here  we use YEAR in WHERE clause to select only those Failure drone ship landing occurred in year ‘2015’

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

## SQL Query

```
%%sql
Select distinct landing__outcome, count(landing__outcome) from SPACEXTBL
where (DATE  between '2010-06-04' and '2017-03-20')
group by landing__outcome
order by count(landing__outcome) desc
```

## Result

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

## Explanation

DISTINCT and COUNT is used so that only UNIQUE landing outcomes are considered

DATE in used in WHERE clause to filter only those landing between  dates 2010-06-04 and 2017-03-20

GROUP BY, ORDER BY and DESC is used to display all landing according to their total counts.

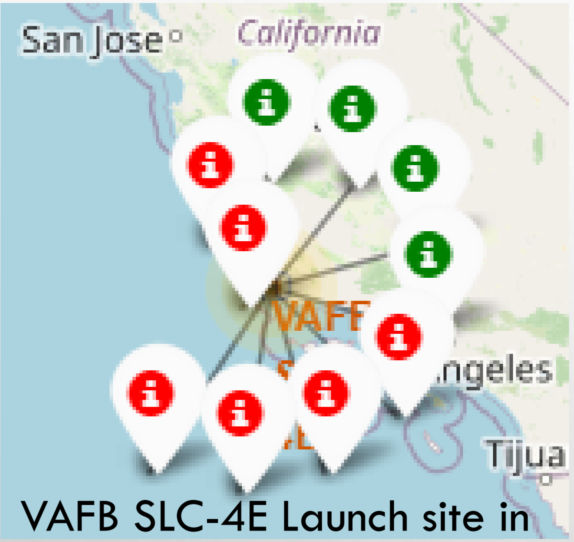
# Interactive maps

## with Folium

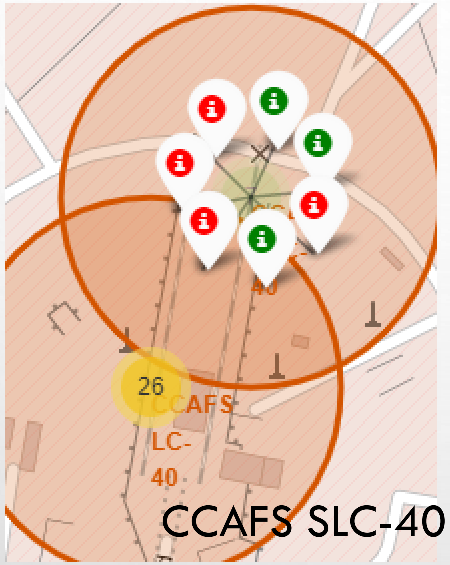
# SpaceX Launch Sites


It is visible that the SpaceX launch sites are in the coastal regions of United States

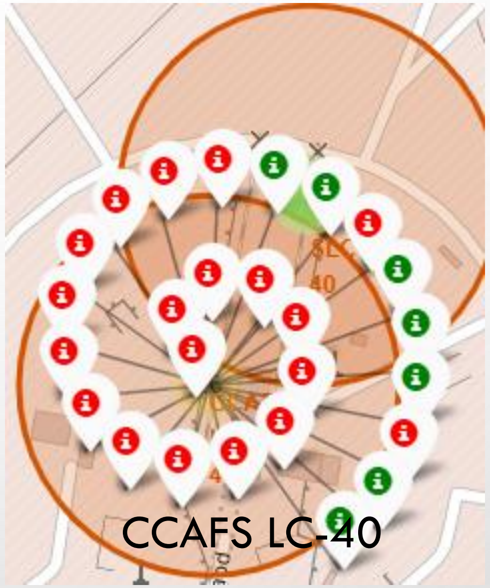
# Successful AND Failed launches



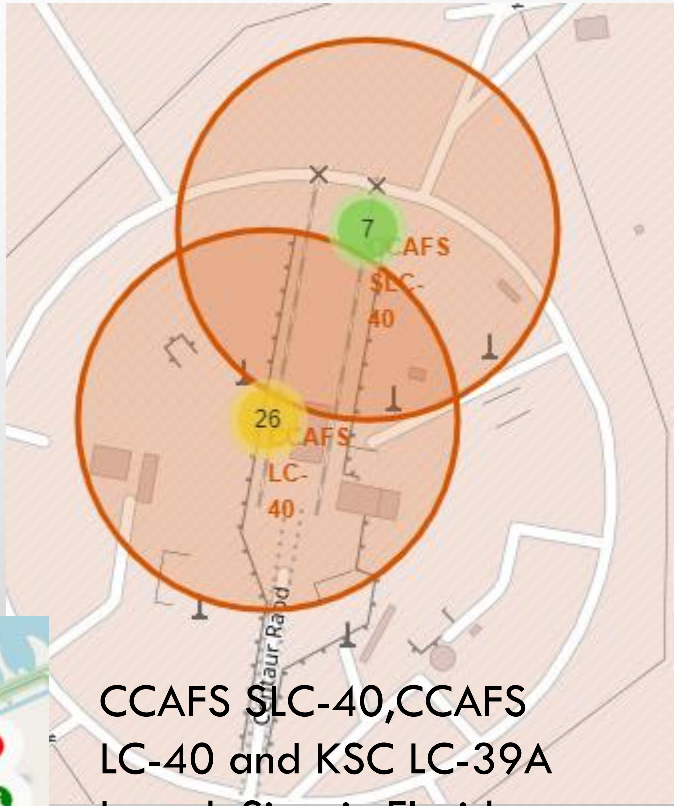

VAFB SLC-4E Launch site in California




CCAFS SLC-40



CCAFS LC-40



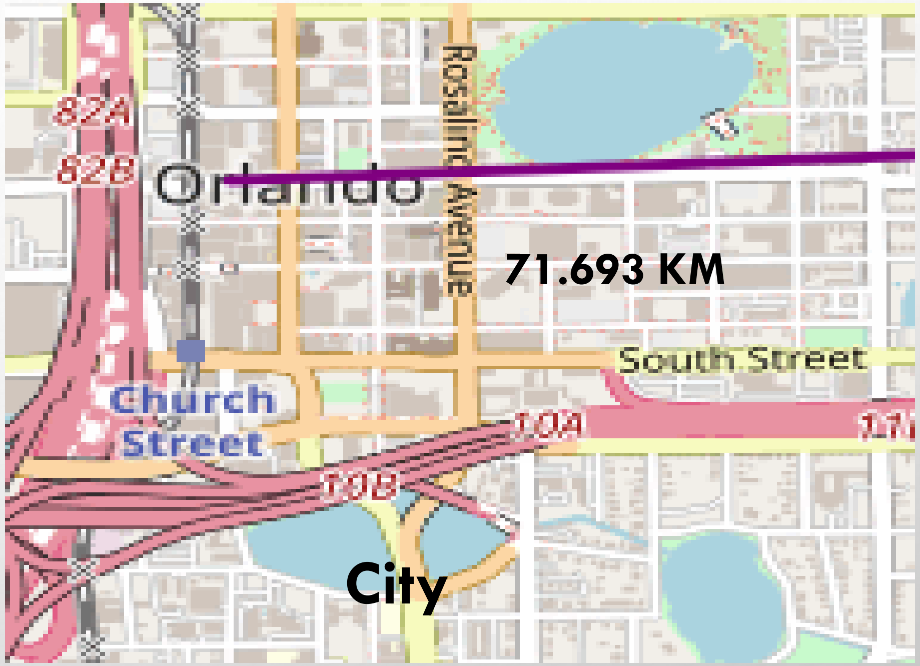

CCAFS SLC-40,CCAFS LC-40 and KSC LC-39A Launch Sites in Florida


KSC LC-39A

Green Marker represents Successful Launches and Red Marker represents Failed Launches

# Distance from Launch Sites to landmarks using KSC LC-39A as a reference



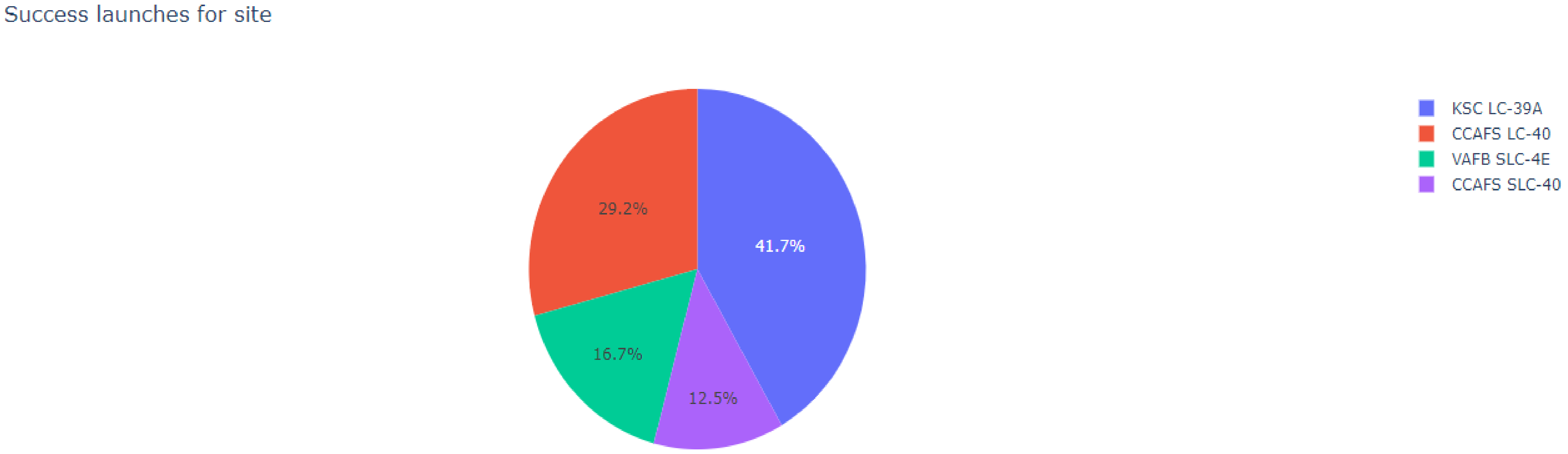




- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

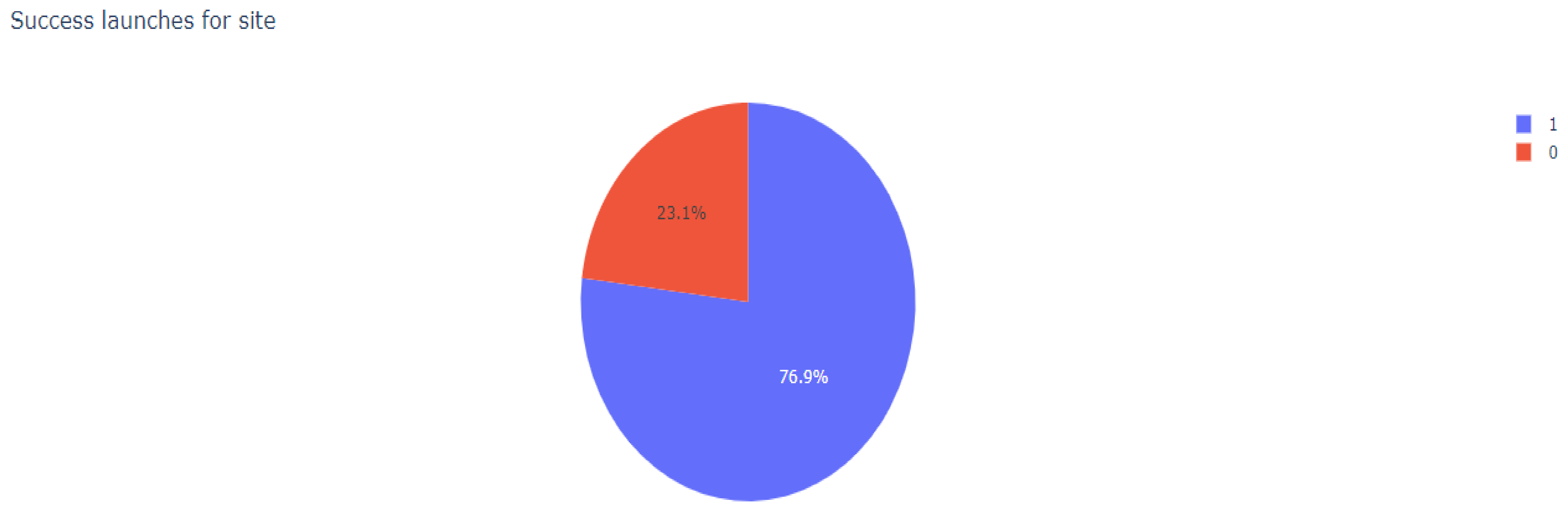

DASH
85%
50%
30%

# Pie Chart : Success Launches For Sites

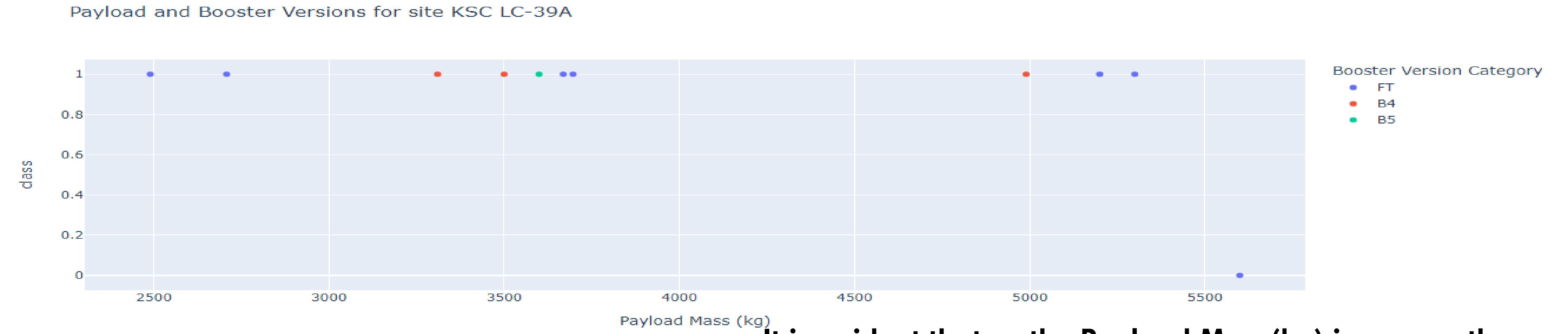



It is quite visible that KSC LC-39A has had the most successful launches from all the sites followed by CCAFS LC-40

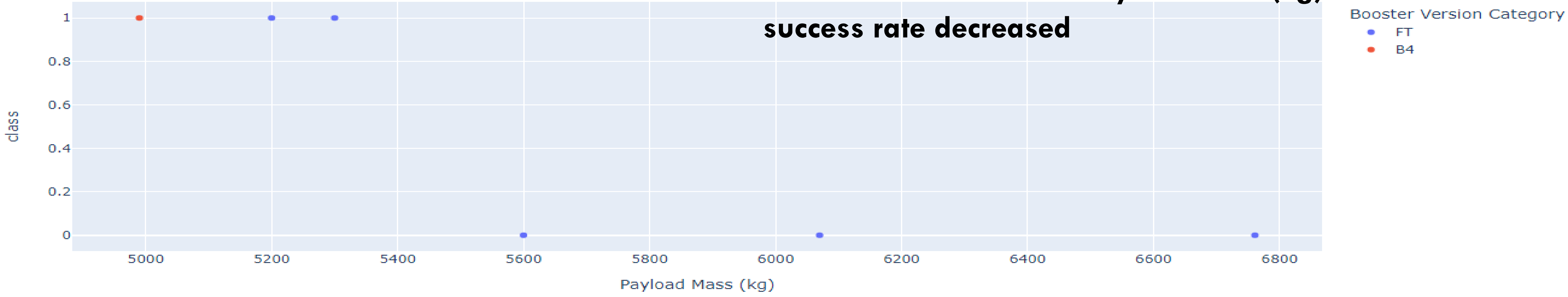
# Pie Chart : Launch Site with highest launch success ratio




KSC LC-39A has a success rate of 76.9% with failure rate of 23.1%



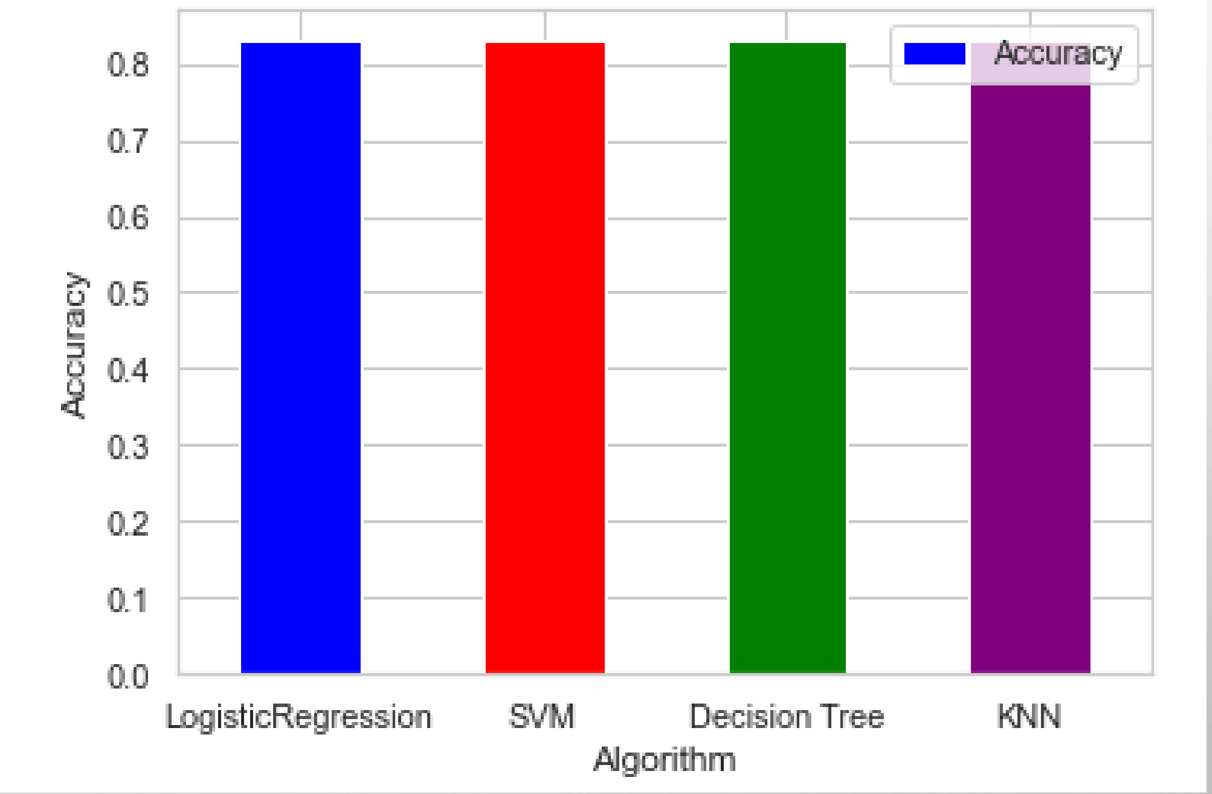

It is evident that as the Payload Mass(kg) increases the success rate decreased



# Predictive Analysis (Classification)

# Classification Accuracy(using Testing data)

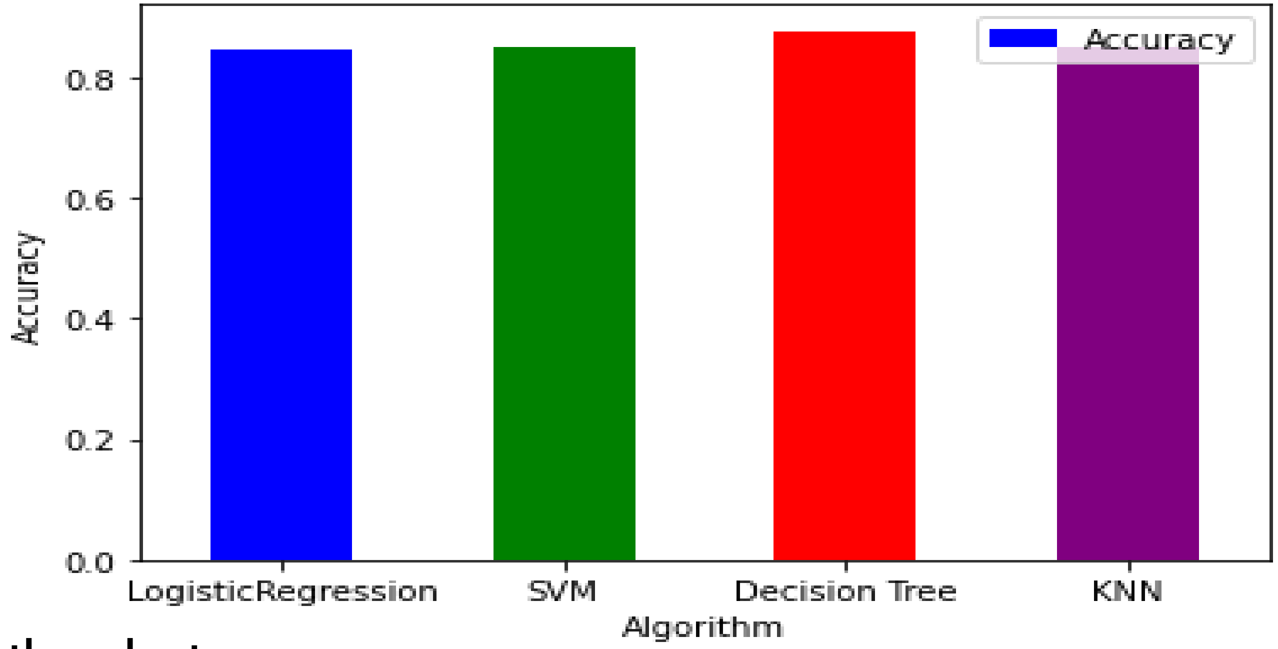
| | Algorithm | Accuracy |
|---|---|---|
| 0 | LogisticRegression | 0.83 |
| 1 | SVM | 0.83 |
| 2 | Decision Tree | 0.83 |
| 3 | KNN | 0.83 |




**As it is clearly visible that the accuracy when testing data is used, is same for ever model.**

# Classification Accuracy(using Training data)

| | Algorithm | Accuracy |
|---|---|---|
| 0 | LogisticRegression | 0.846429 |
| 1 | SVM | 0.848214 |
| 2 | Decision Tree | 0.876786 |
| 3 | KNN | 0.848214 |

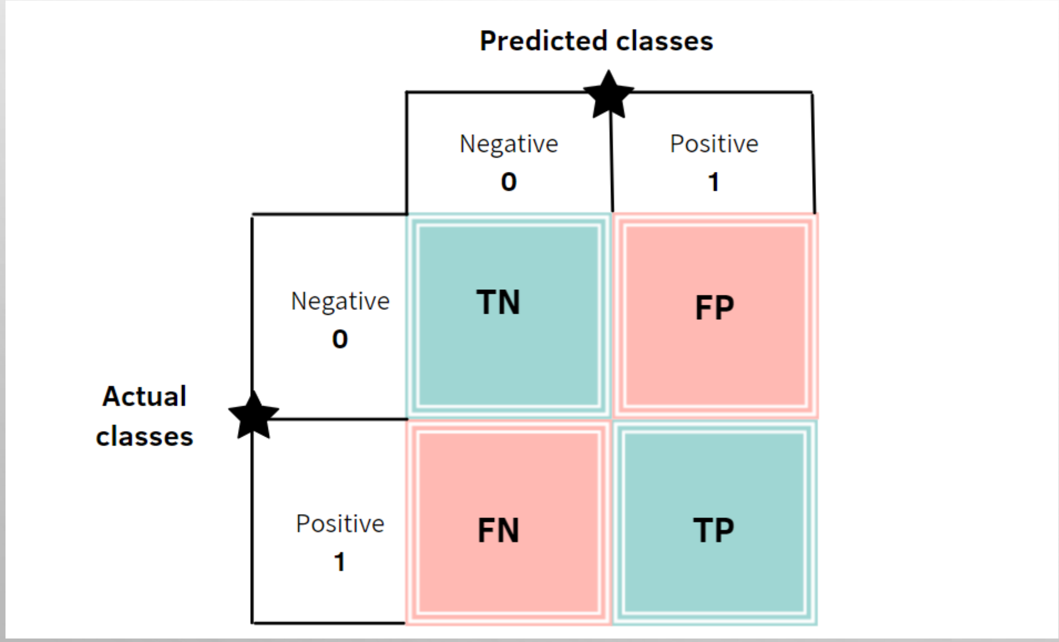


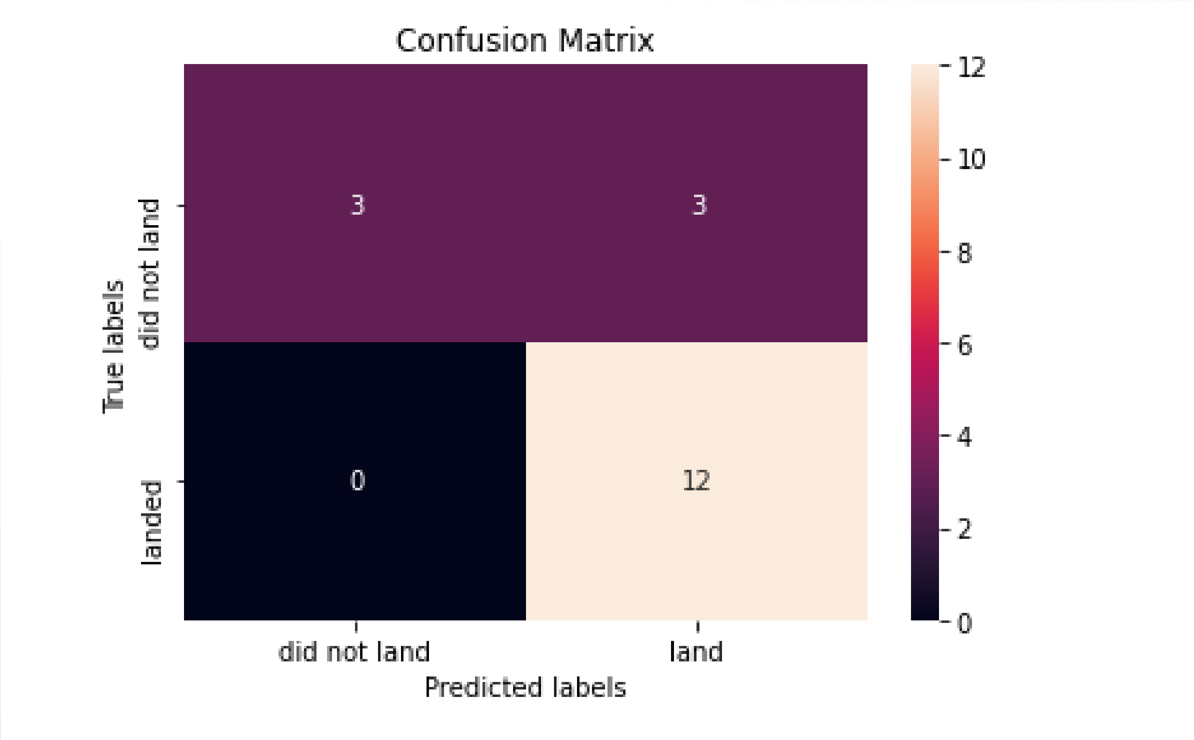

Accuracies are extremely close to each other but **Decision Tree** has the highest one

```
Best Algorithm is Tree with a score of 0.8767857142857143
Best Params is : {'criterion': 'gini', 'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5,
'splitter': 'random'}
```

# Confusion Matrix

All the classification has same confusion matrix. All the classification models has same problem i.e. **false positives.**



# Conclusions

- Success rate of launches increases with the number of launches that have taken place over the years.
- Orbit Type ES-L1,GEO,HEO,SSO has the best Success Rate.
- KSC LC-39A had the most successful launches from all the sites.
- Lighter payloads has higher success rate than the heavier payloads.
- All the models have same accuracy w.r.t. testing dataset
- The Decision Tree Algorithm is the best for Machine Learning w.r.t. training dataset.

# Appendix

## *Haversine formula*

This uses the '**haversine**' formula to calculate the great-circle distance between two points – that is, the shortest distance over the earth's surface – giving an 'as-the-crow-flies' distance between the points

## *Formula*

$$a = \sin^2(\Delta\varphi/2) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2(\Delta\lambda/2)$$

$$c = 2 \cdot \operatorname{atan2}(\sqrt{a}, \sqrt{(1-a)})$$

$$d = R \cdot c$$

*$\phi$ is latitude, $\lambda$ is longitude, R is earth's radius (mean radius = 6,371km);*
*note that angles need to be in radians to pass to trig functions*

**Code used:**

```python
from math import sin, cos, sqrt, atan2, radians

def calculate_distance(lat1, lon1, lat2, lon2):
    # approximate radius of earth in km
    R = 6373.0

    lat1 = radians(lat1)
    lon1 = radians(lon1)
    lat2 = radians(lat2)
    lon2 = radians(lon2)

    dlon = lon2 - lon1
    dlat = lat2 - lat1

    a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
    c = 2 * atan2(sqrt(a), sqrt(1 - a))

    distance = R * c
    return distance
```

# THANK YOU