

# **IBM Data Science Capstone Project: Space X**

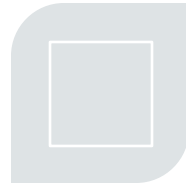
**Ademola  
Ayobami Tijani**

**16<sup>th</sup> November 2021**

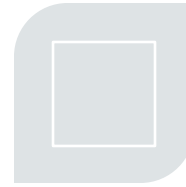




# Outline



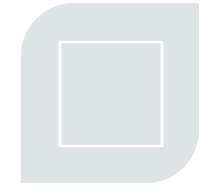
EXECUTIVE  
SUMMARY



INTRODUCTION



METHODOLOGY



RESULTS



CONCLUSION



APPENDIX

# Executive Summary

## Summary of methodologies –

Data collection via API, SQL and Web Scraping

Data wrangling and Analysis

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 –

Exploratory data analysis results

Interactive Visualizations in screenshots

Predictive Analysis results



# Introduction

---

## **Project background and context.**

The aim of this project is to predict if the falcon 9 first stage will land successfully. SpaceX advertises Falcon 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 the launch. This information can be used if an alternate company wants to bid against SpaceX.

---

## **Problems to be addressed**

- What influences if the rocket will land successfully?
- What is the effect of each relationship of rockets variables on outcomes?
- What are the Conditions which will aid SpaceX to achieve the best results?



Section 1

# Methodology

## Executive Summary

### Data collection methodology:

- SpaceX Rest API
- Web Scrapping from [wikipedia](#)

### Perform data wrangling (Transforming data for Machine Learning)

- One Hot Encoding data fields for Machine learning and dropping irrelevant columns

### Perform exploratory data analysis (EDA) using visualization and SQL

- Plotting : Scatter and bar graphs to show relationship between variables and to show patterns of data

### Perform interactive visual analytics:

- Using Folium and Plotly Dash visualizations

### Perform predictive analysis using classification models

- Build, tune, evaluate classification models

# Data Collection

---

- **How data sets were gathered.**
  - I worked with SpaceX launch data that is gathered from SpaceX REST API.
  - The API gives us data about launches, including information about the rocket used, payload delivered, launch specifications, landing specifications and landing outcome.
  - Our goal is to use this data to predict whether SpaceX will attempt to land a rocket or not.
  - The SpaceX REST API endpoints, or URL, start with `api.spacexdata.com/v4/`.
  - The second data source for obtaining Falcon 9 launch data is through web scraping Wikipedia using BeautifulSoup. (A popular Python library used for web scraping)

## SpaceX API.

Use SpaceX REST API => API returns SpaceX data in JSON => Normalize data into flat data file such as .csv

## Web Scrapping.

Get HTML Response from Wikipedia => Extract using BeautifulSoup library => Normalize into flat data such as .csv



# Data Collection – SpaceX API

---

---

Dats collection- SpaceX  
**REST API**

---

[Github url to notebook](#)



**1. Getting Response from API**



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



**3. Apply custom functions to clean data**



**4. Assign list to dictionary then dataframe**

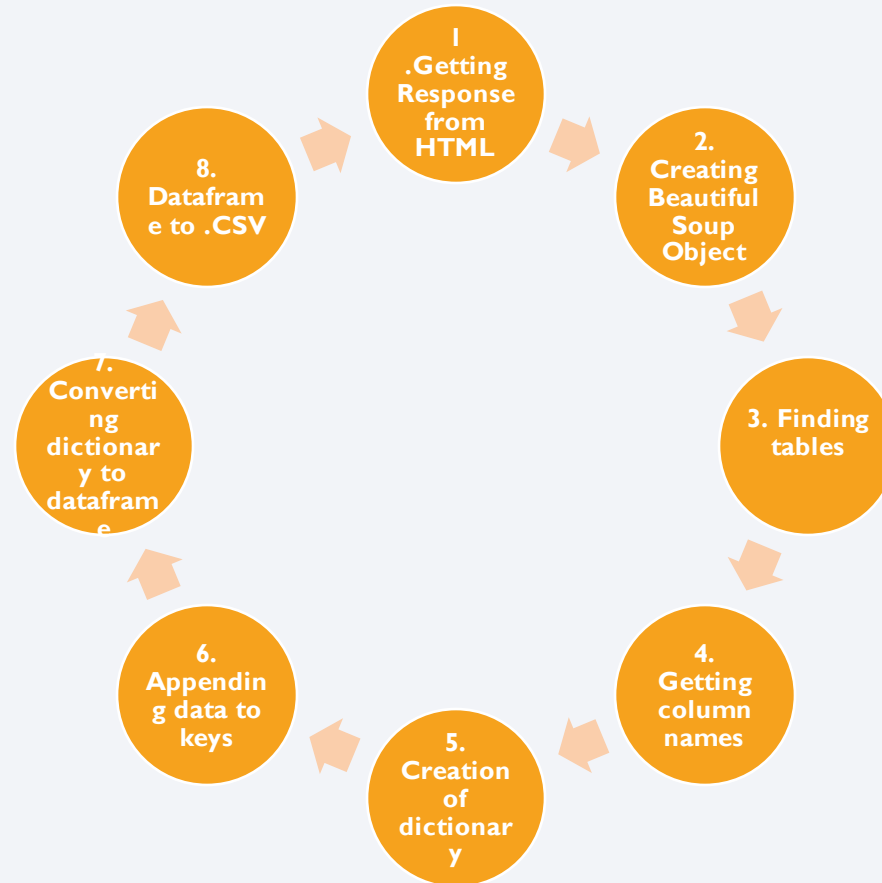


**5. Filter dataframe and export to flat file  
(.csv)**



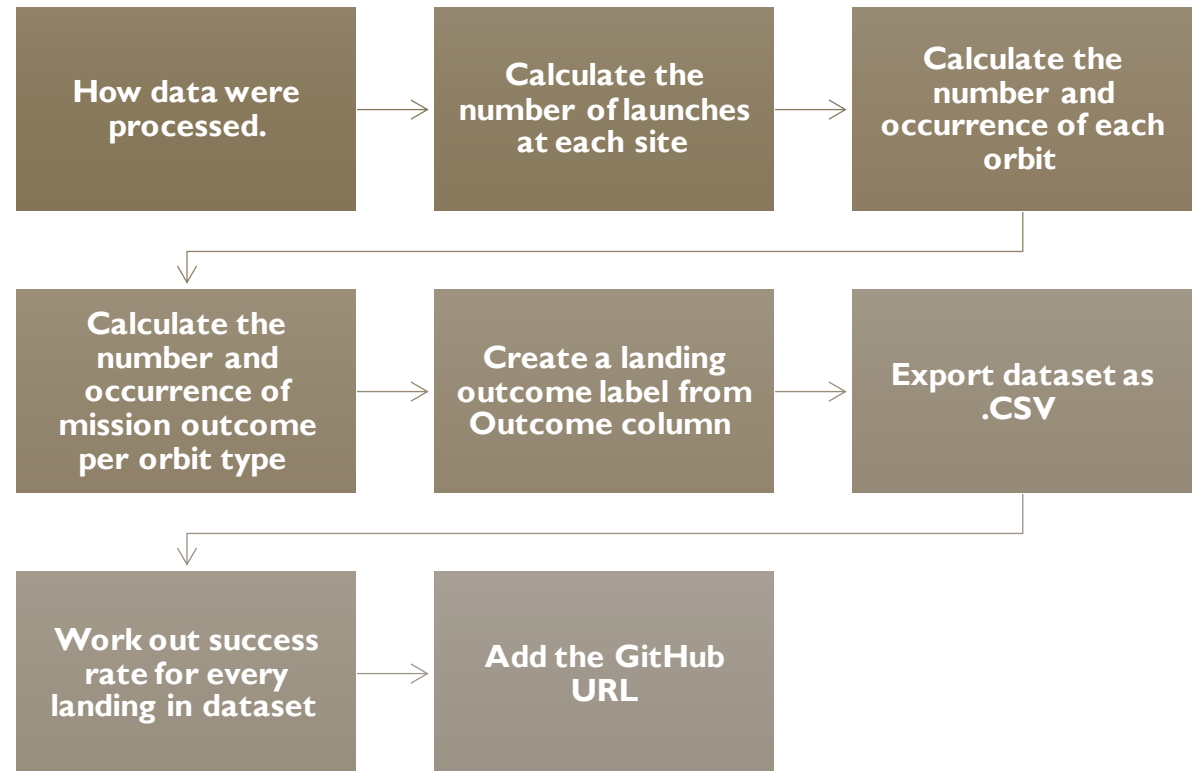
# Data Collection - Scraping

---



- [Github url to notebook](#)

# DATA WRANGLING



[Github url to notebook](#)

# EDA with Data Visualization

**1. Scatter Graphs being drawn:** Scatter plots show how much one variable is affected by another. The relationship between two variables is called their correlation. Scatter plots usually consist of a large body of data.

---

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

**2. Bar Graph being drawn:** A bar diagram makes it easy to compare sets of data between different groups at a glance. The graph represents categories on one axis and a discrete value in the other. The goal is to show the relationship between the two axes. Bar charts can also show big changes in data over time.

- Mean VS. Orbit

**3. Line Graph being drawn:** Line graphs are useful in that they show data variables and trends very clearly and can help to make predictions about the results of data not yet recorded

- Success Rate VS. Year

# EDA WITH VISUALIZATION

## • Summary of the SQL queries you performed

- **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 (CRS)**
- **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 the 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 between the date 2010-06-04 and 2017-03-20 in descending order.**

[GitHub URL to Notebook](#)

# Build an Interactive Map with Folium

- Folium makes it easy to visualize manipulated data in python on an interactive leaflet map. We use the latitude and longitude coordinates for each launch site and added a Circle Marker around each launch site with a label of the name of the launch site. Also, it is easy to visualize the number of success and failure for each launch site with green and Red markers on the map

Map Objects	Code	Result
Map Marker	<code>folium.Marker(</code>	Map object to make a mark to map
Icon Marker	<code>folium.icon(</code>	Create an icon on map
Circle Marker	<code>folium.Circle(</code>	Create a circle where marker is being placed
Polyline	<code>folium.Polyline(</code>	Create a line between points.
Marker Cluster Object	<code>MarkerCluster()</code>	This is a good way to simplify a map containing many markers having the same coordinate.
AntPath	<code>Folium.plugins.AntPath(</code>	Create an animated line between points



# Build a Dashboard with Plotly Dash

Pie chart shows the total success for all sites / by certain launch site

Scatter graph shows the correlation between payload and success for all sites

Map Objects	Code	Result
Dash and its components	Import dash, import dash_html_components as html	Plotly python's leading data viz and Ui libraries. With dash open source, Dash apps run on your local laptop or server. The dash core components library contain a set of higher-level components like slider, graph, dropdown and tables. Dash provides all html tags.
Pandas	Import pandas as pd	Fetching values from CSV and creating a dataframe
Plotly	Import plotly.express as px	Plot the graphs with interactive plotly library
Dropdown	dcc.Dropdown(	Create a dropdown for launch sites
Rangeslider	dcc.RangeSlider(	Create a rangeslider for payload mass range selection
Pie chart	Px.pie(	Creating the pie graph for success percentage display
Scatter chart	Px.scatter(	Creating the scatter graph for success correlation display

# Predictive Analysis (Classification)

- **Building Model:**
  - Load our dataset into Numpy and Pandas
  - Transform data
  - Split our data into training and test data sets
  - Check how many test samples we have
  - Decide which type of machine learning algorithms we want to use
  - Set our parameters and algorithms to GridSearchCv
  - Fit our datasets into the GridSearchCv objects and train our model
- **Evaluating Model**
  - Check accuracy for each model
  - Get tuned hyperparameters for each type of algorithms
  - Plot confusion Matrix
- **Improving Model**
  - Feature Engineering
  - Algorithm Tuning
- **Finding the best performing classification Model**
  - The model with the best accuracy score wins the best performing model

# Results

---

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



The background of the slide is an abstract composition. It features a solid blue area on the left side, which transitions into a complex pattern of diagonal streaks and lines in shades of blue, red, and cyan on the right. These streaks have a textured, almost woven appearance, suggesting a digital or data-driven theme. The overall effect is dynamic and modern.

Section 2

# Insights drawn from EDA



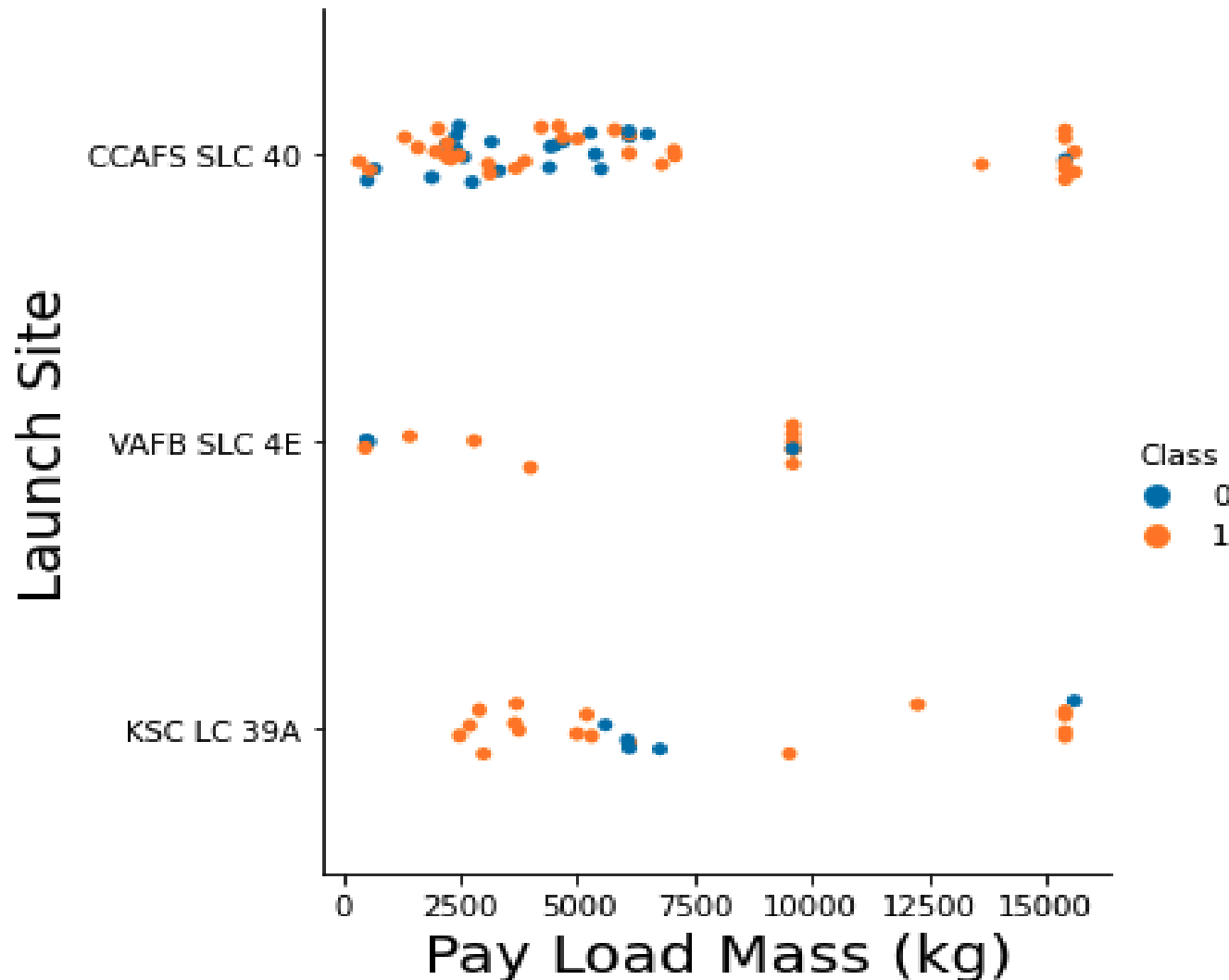
# Flight Number vs. Launch Site



The more flights at a launch site the greater the success rate at a launch site.



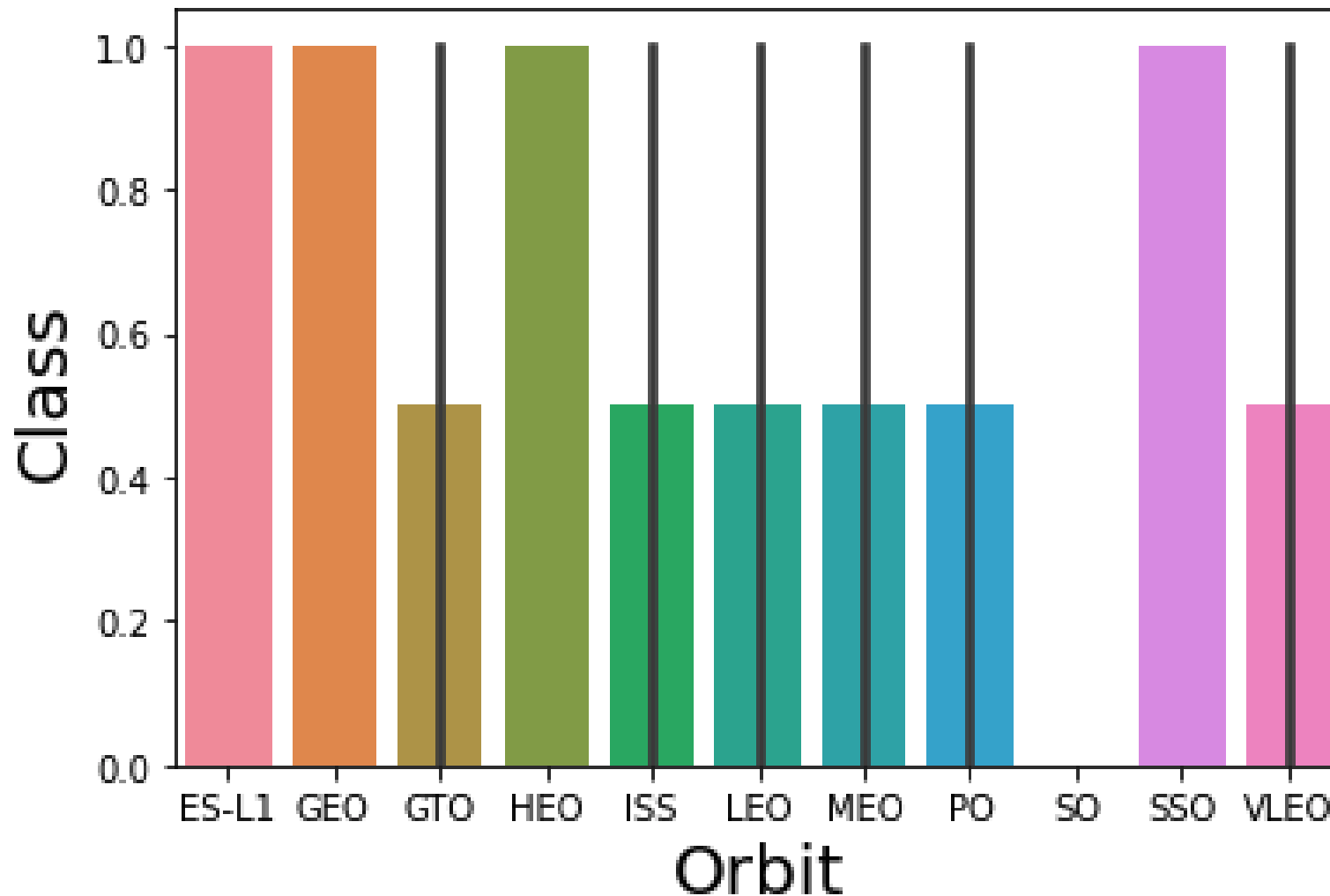
# Payload vs. Launch Site



- The more the payload mass for Launch Site CCAFS SLC 40 the higher the success rate for the Rocket. There is not quite a clear pattern to be found using this visualization to make a decision if the Launch Site is dependent on Pay Load Mass for a success launch.

[Github url to notebook](#)

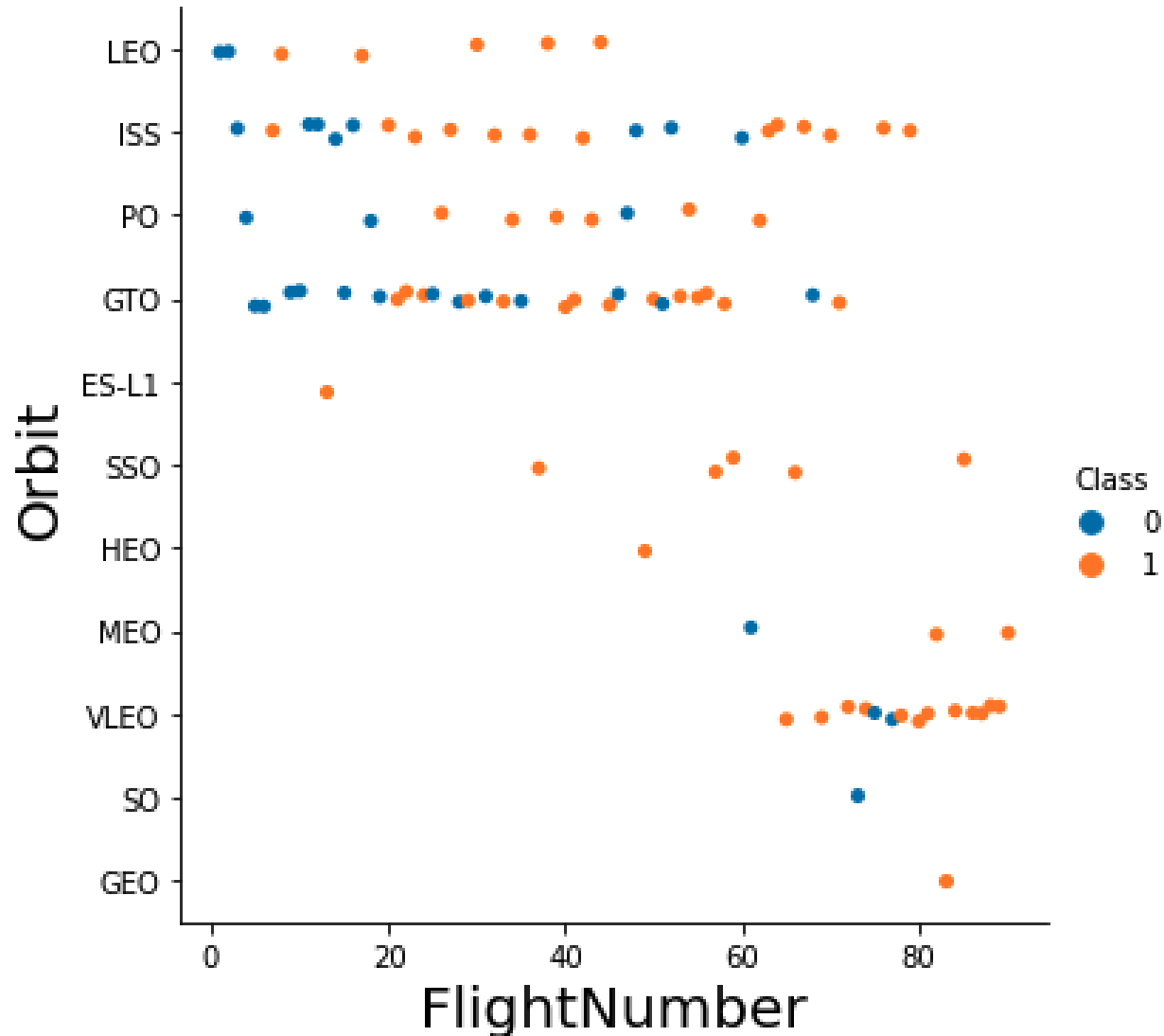
# Success Rate vs. Orbit Type



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

[Github url to notebook](#)

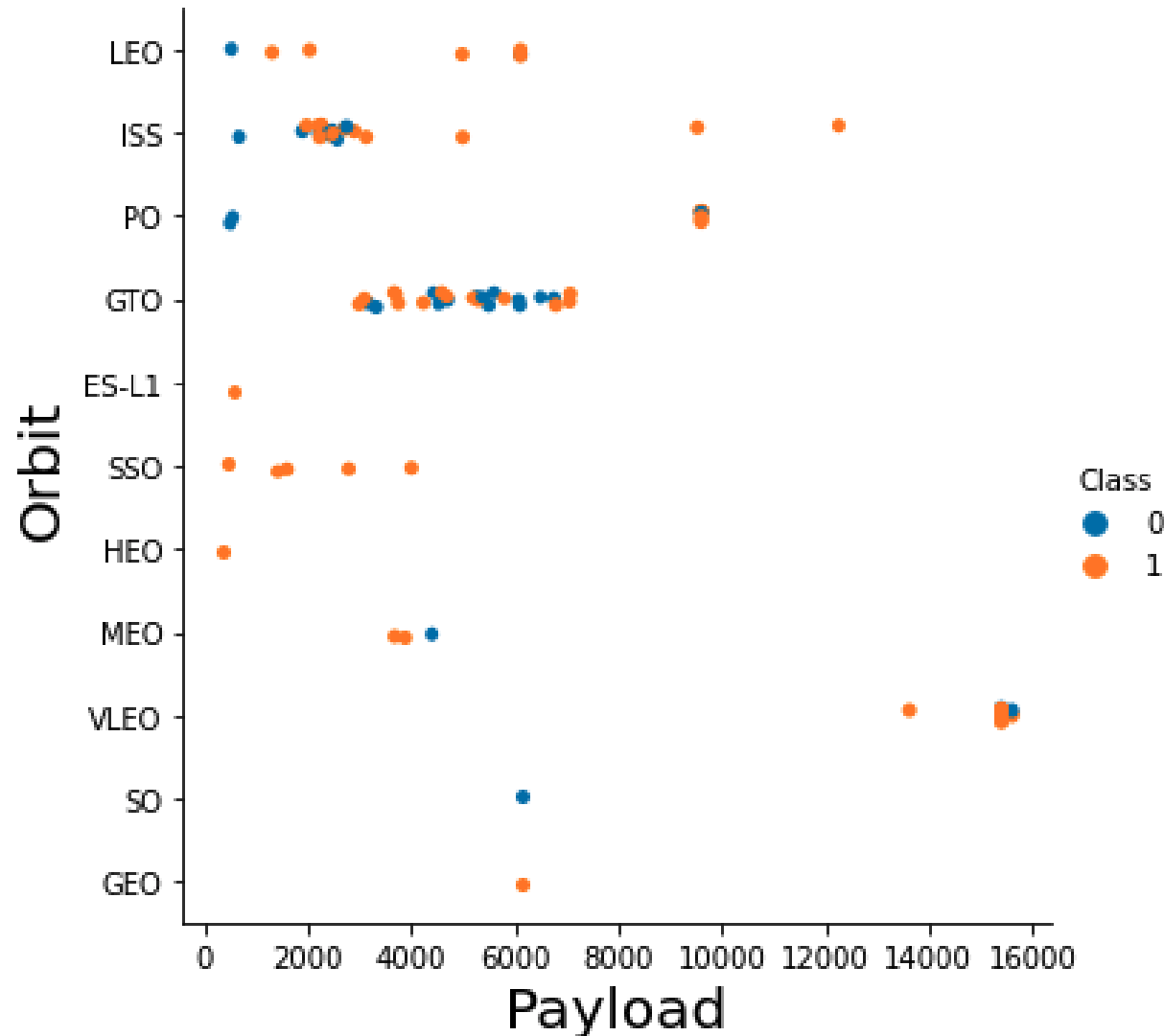
# Flight Number vs. Orbit Type



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.

[Github url to notebook](#)

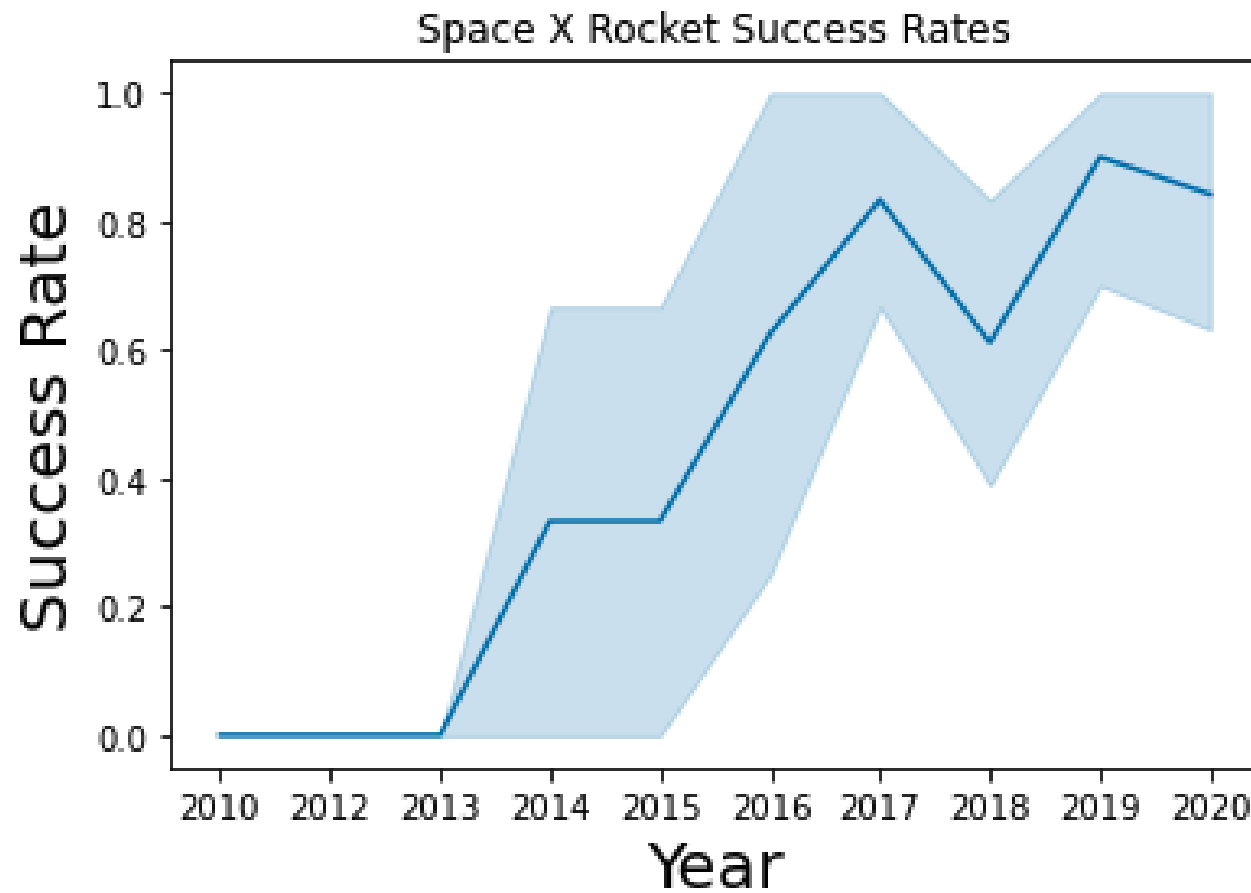
# Payload vs. Orbit Type



- We can observe that Heavy payloads have a negative influence on GTO orbits and positive on GTO and Polar LEO (ISS) orbits.

[Github url to notebook](#)

# Launch Success Yearly Trend



- As we can see that the success rate since 2013 kept increasing till 2020

[Github url to notebook](#)



# EDA WITH SQL

# All Launch Site Names

- Sql Query

```
%sql select distinct(LAUNCH_Site) from SPACEXTBL
```



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

**Description:** Using the word distinct in the query will pull the unique values for the launch Site column from the table SPACEXTBL

[Github url to notebook](#)

# Launch Site Names Begin with 'CCA'

- Sql Query

```
%sql select * from SPACEXTBL where LAUNCH_SITE like 'CCA%' limit 5
```

- Description:** Using keyword “Limit 5” in the query will fetch 5 records from table spacex, condition LIKE keyword with wild card “CCA%”. The percentage in the end suggest that the launch\_site name must start with CCA.

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS_KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

# Total Payload Mass

- SQL Query

```
%sql select sum(PAYLOAD_MASS_KG_) from SPACEXTBL where CUSTOMER = 'NASA (CRS)';|
```



**sum(PAYLOAD\_MASS\_KG\_)**

**45596**

**Description:** Using function SUM summate the total in the column PAYLOAD\_MASS\_KG and the WHERE clause filters the dataset to only perform calculations on customer NASA (CRS)

[Github url to notebook](#)

# Average Payload Mass by F9 v1.1

- Sql query

```
%sql select avg(PAYLOAD_MASS_KG_) from SPACEXTBL where BOOSTER_VERSION LIKE '%F9 v1.1';
```



**avg(PAYLOAD\_MASS\_KG\_)**

2928.4

## Description:

Using the function AVG works out the average in the column PAYLOAD\_MASS\_KG

The WHERE clause filters the dataset to only perform calculations on Booster\_version f9 v1.1

[Github url to notebook](#)



# First Successful Ground Landing Date

- **SQL Query:**

- `select MIN(Date) SLO from tblSpaceX where Landing_Outcome = "Success (drone ship)"`



Date which first Successful landing outcome in drone ship was acheived.	
0	06-05-2016

- **Description:** The function MIN works out the minimum date in the column date while the WHERE clause filters the dataset to only perform calculations on Landing\_outcome success

[Github url to notebook](#)

## Successful Drone Ship Landing with Payload between 4000 and 6000

- SQL Query:

- `select Booster_Version from tblSpaceX where Landing_Outcome = 'Success (ground pad)' AND Payload_MASS_KG_ > 4000 AND Payload_MASS_KG_ < 6000`



Date which first Successful landing outcome in drone ship was acheived.

0	F9 FT B1032.1
1	F9 B4 B1040.1
2	F9 B4 B1043.1

**Description:** Selecting only Booster\_Version. WHERE clause filters AND clause specifies additional filter.

[Github url to notebook](#)

# Total Number of Successful and Failure Mission Outcomes

---

- SQL Query:
- %sql select count(MISSION\_OUTCOME) from SPACEXTBL where MISSION\_OUTCOME = 'Success' or MISSION\_OUTCOME = 'Failure (in flight)'



```
count(MISSION_OUTCOME)
```

```
99
```

# Boosters Carried Maximum Payload

- Sql Query:

```
%sql select BOOSTER_VERSION from SPACEXTBL where PAYLOAD_MASS_KG_ = (select max(PAYLOAD_MASS_KG_) from SPACEXTBL)
```



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

**Description:** Using the function MAX works out the maximum payload in the column PAYLOAD\_MASS\_KG\_ in the sub query and WHERE clause filters Booster Version which had that maximum payload.

[Github url to notebook](#)

# 2015 Launch Records

---

- SQL Query
- %sql SELECT BOOSTER\_VERSION, LAUNCH\_SITE FROM SPACEXTBL WHERE DATE LIKE '2015-%' AND \
- LANDING\_\_OUTCOME = 'Failure (drone ship)';

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

- SQL query:

```
%sql select Count(LANDING__OUTCOME) AS "Rank success count between 2010-06-04 and 2017-03-20" from SPACEXTBL \
where LANDING__OUTCOME like 'Success%' and (DATE between '2010-06-04' and '2017-03-20') order by date desc
```



Rank success count between 2010-06-04 and 2017-03-20
8

## Description:

COUNT counts records in column LANDING\_\_OUTCOME. WHERE filters data with '%Success%'

[Github url to notebook](#)

Section 4

# Launch Sites Proximities Analysis





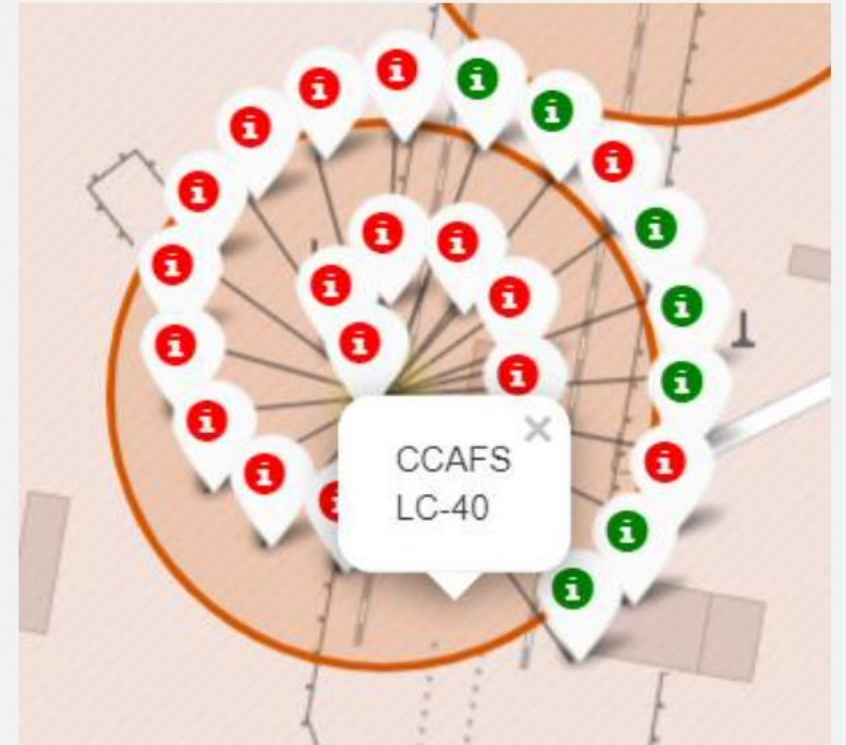
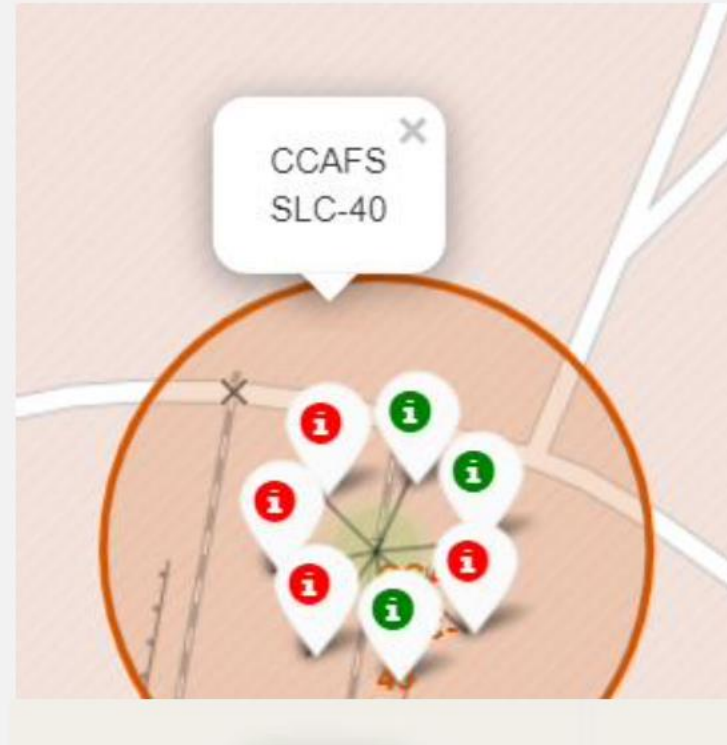
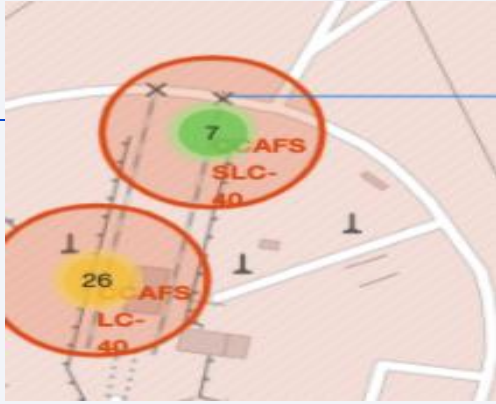
# All launch sites global map markers



According to the map:  
The SpaceX launch sites are in the united states of America coast.  
Florida and California

[Github url to notebook](#)

# Colour Labelled Markers

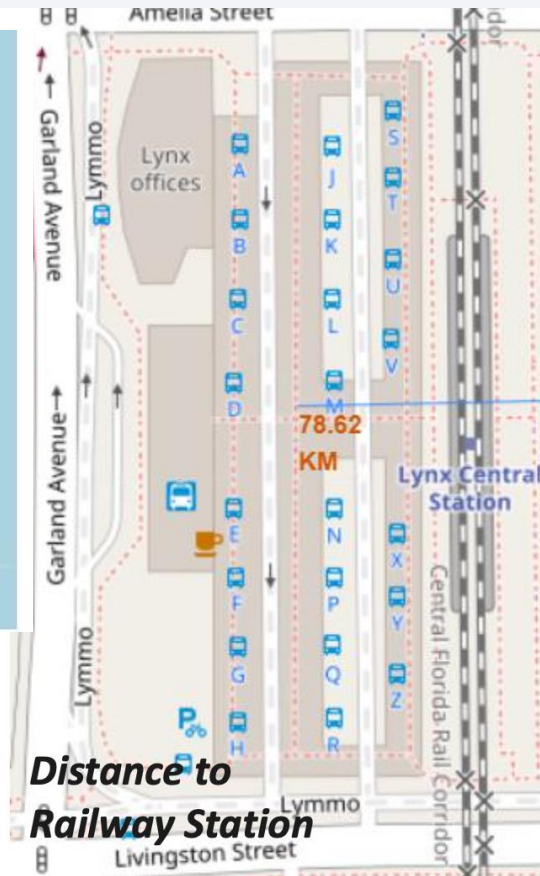
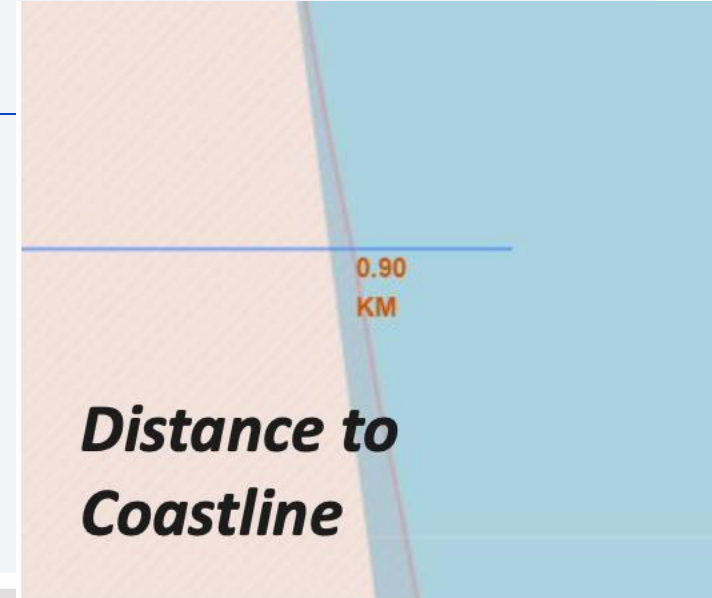
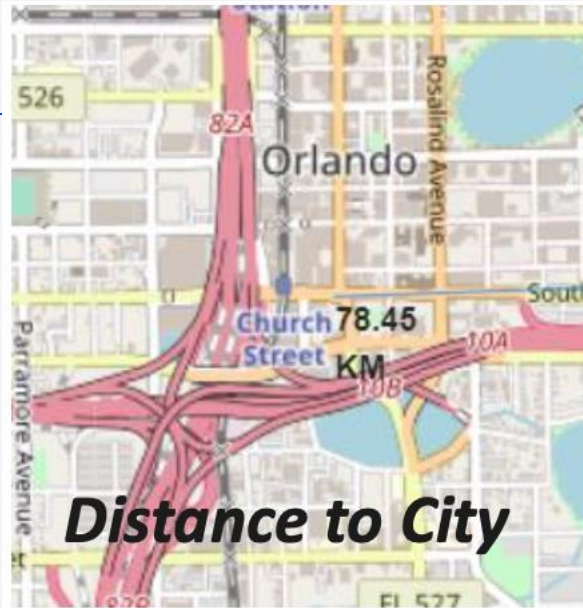


Green Marker shows successful launches and Red Marker shows Failures

[Github url to notebook](#)



# Working out launch sites distance to landmark to find trends with haversine formula using CCAFS-SLC-40 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



[Github url to notebook](#)



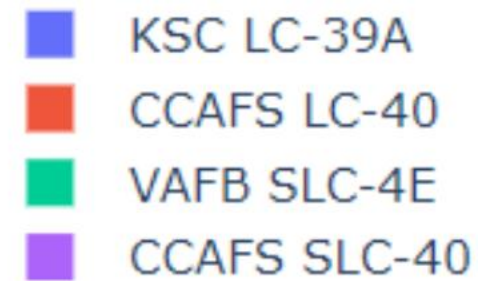
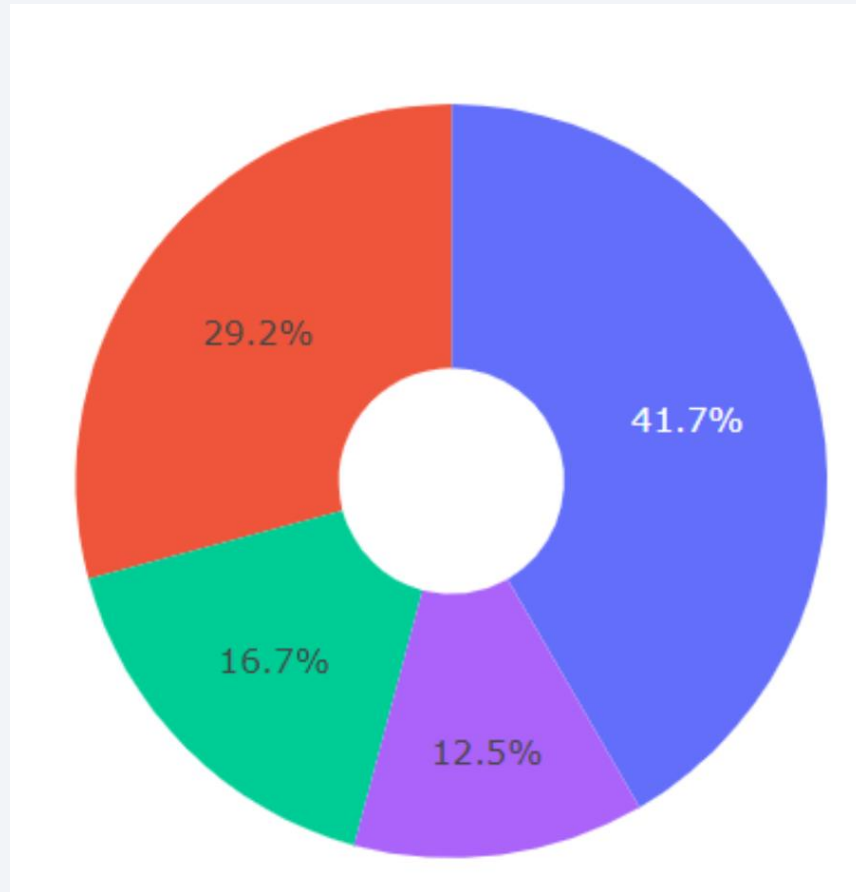


Section 5

# Build a Dashboard with Plotly Dash

# Total success launches by all sites

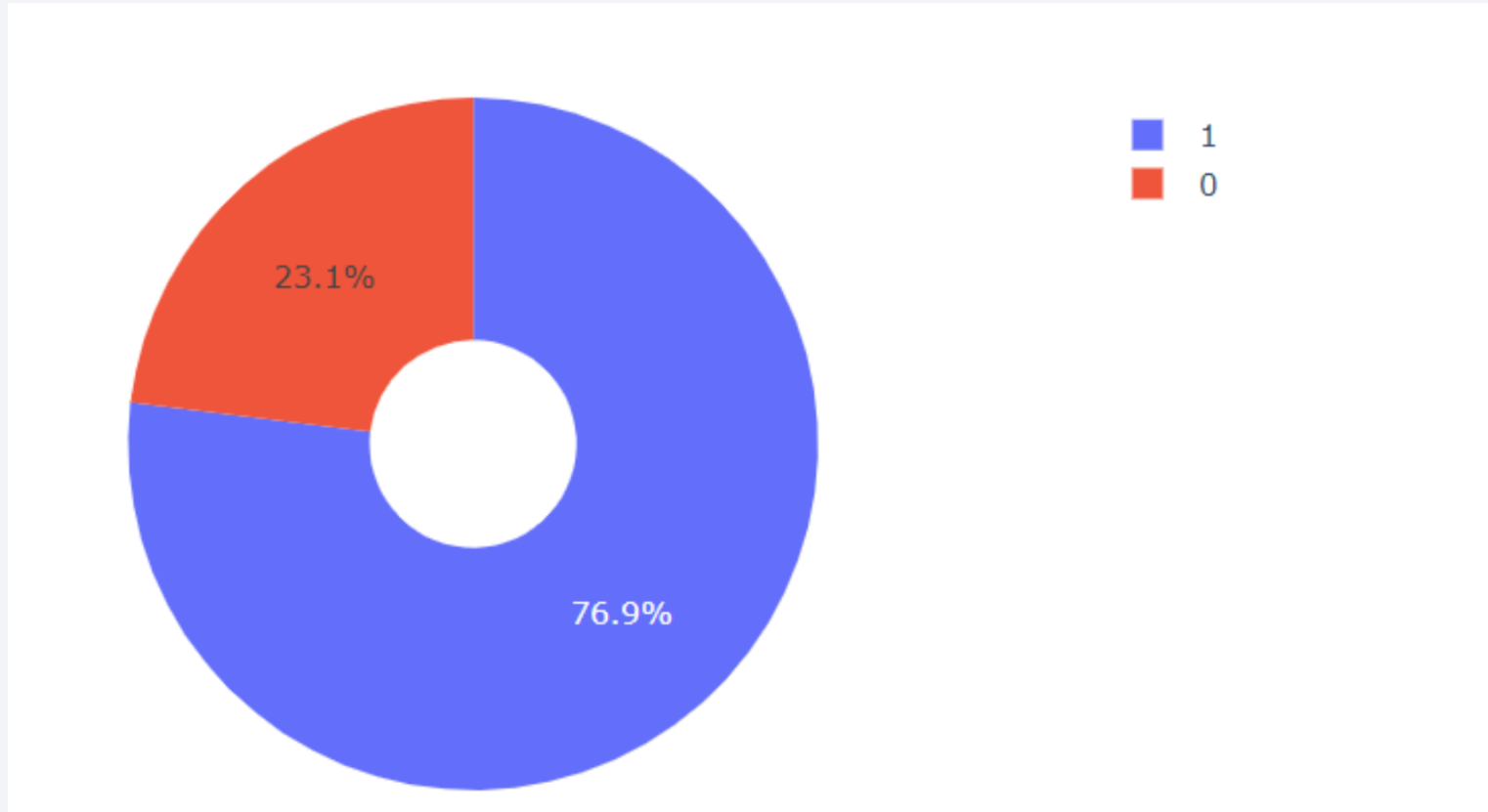
---



As we can see from the pie chart  
KSC LC-39A had the most successful launches from all the sites

## DASHBOARD – Pie chart for the launch site with highest launch success ratio

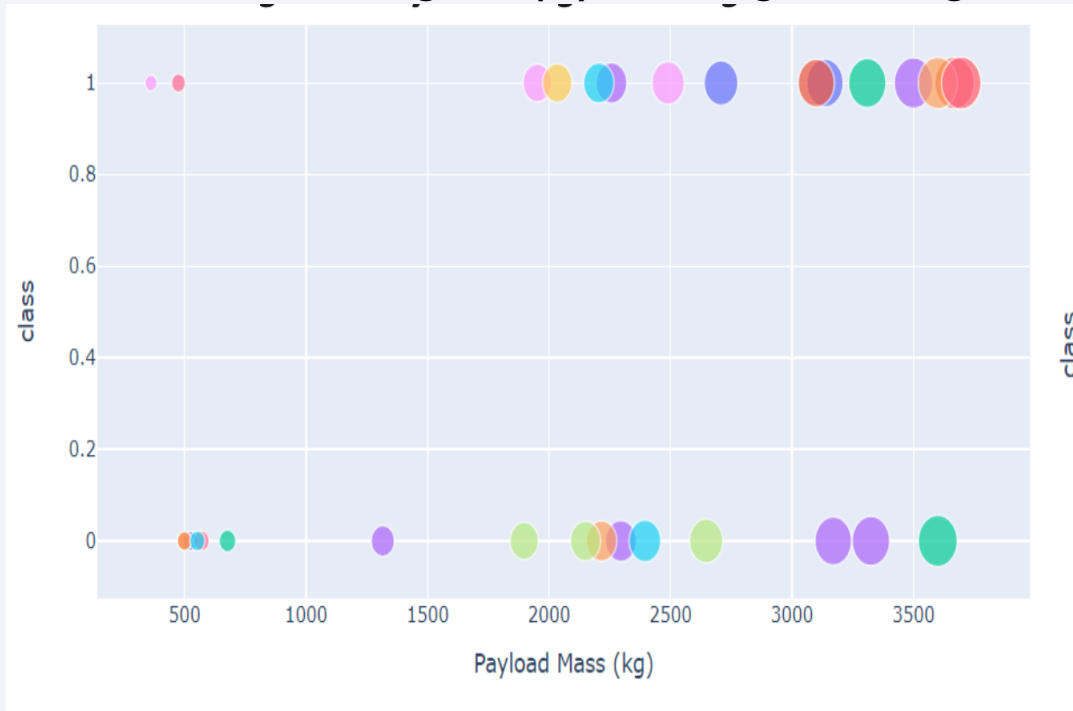
---



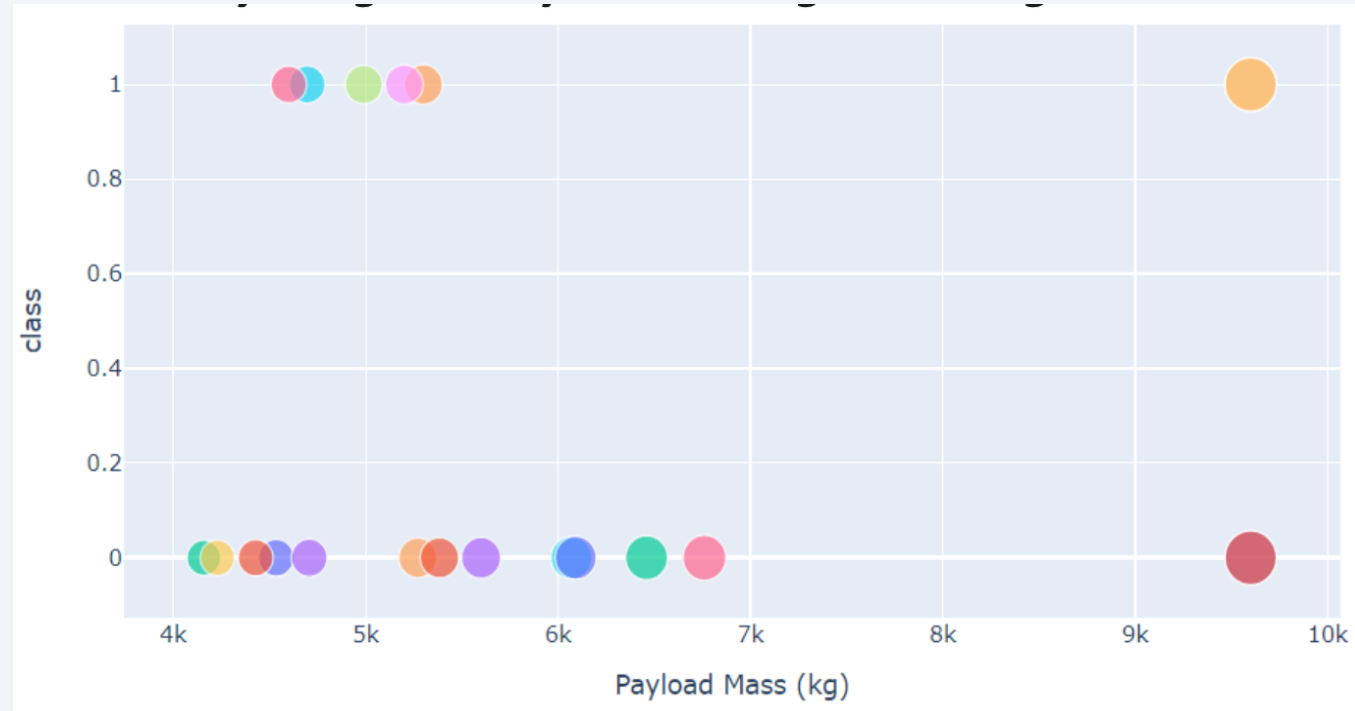
KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

# Dashboard- Payload vs launch outcome scatter plot for all sites, With different payload selected in the range slider

Low weighted payload 0kg – 4000kg



Heavy weighted payload 4000kg – 10000kg



As we can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 6

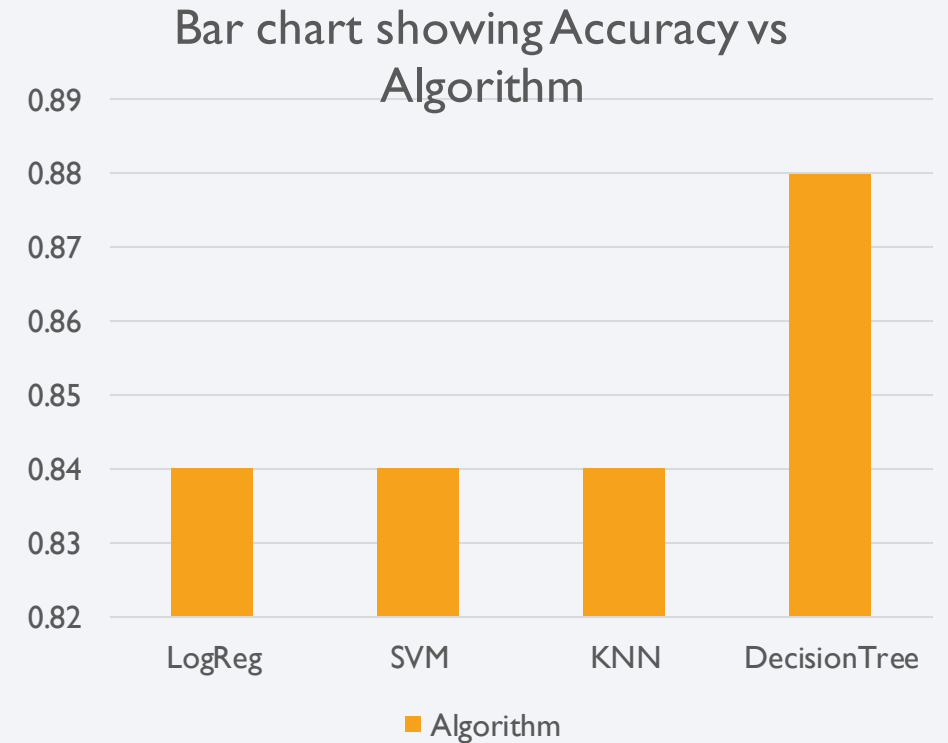
# Predictive Analysis (Classification)



# Classification Accuracy

The best perform accuracy is Decision tree with a score of 0.88. We trained four model and none of them had anything less than 0.83

Algorithm	Accuracy	Accuracy on test data	
Logistic Regression	0.8464	0.83334	
SVM	0.84821	0.83334	
KNN	0.8482	0.83334	
Decision Tree	0.8892	0.7222	



[Github url to notebook](#)

# Confusion Matrix

All models used have the same confusion matrix.

Accuracy:  $(TP+TN)/total = (12+3)/18=0.8333$

Misclassification Rate  $(FP+FN)/total=(3+0)/18=0.1667$

True Positive Rate:  $TP/Actual\ Yes = 12/12 = 1$

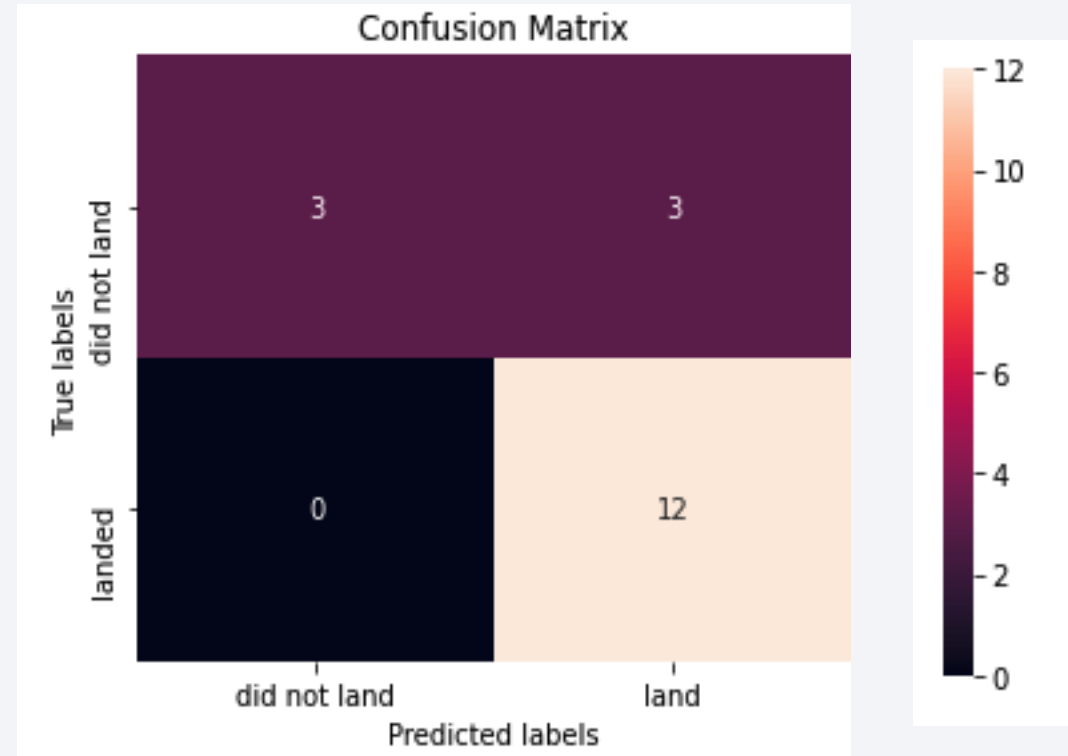
False Positive Rate:  $FP/Actual\ No=3/6=0.5$

True Negative Rate:  $TN/Actual\ No=3/6=0.5$

Precision:  $TP/Predicted\ Yes = 12/15=0.8$

Prevalence:  $Actual\ Yes/total=12/18 = 0.6667$

[Github url to notebook](#)




# CONCLUSIONS

Orbits ES-LI, GEO, HEO, SSO has highest success rates



The Success rates for SpaceX launches has been increasing relatively with time, they will eventually perfect the launches.



KSC LC\_39A had the most successful launches from all the sites



Low weighted payloads perform better than the heavier payloads



The tree classifier Algorithm is the best machine learning model for this dataset

# APPENDIX

Interactive plotly

Python Anywhere

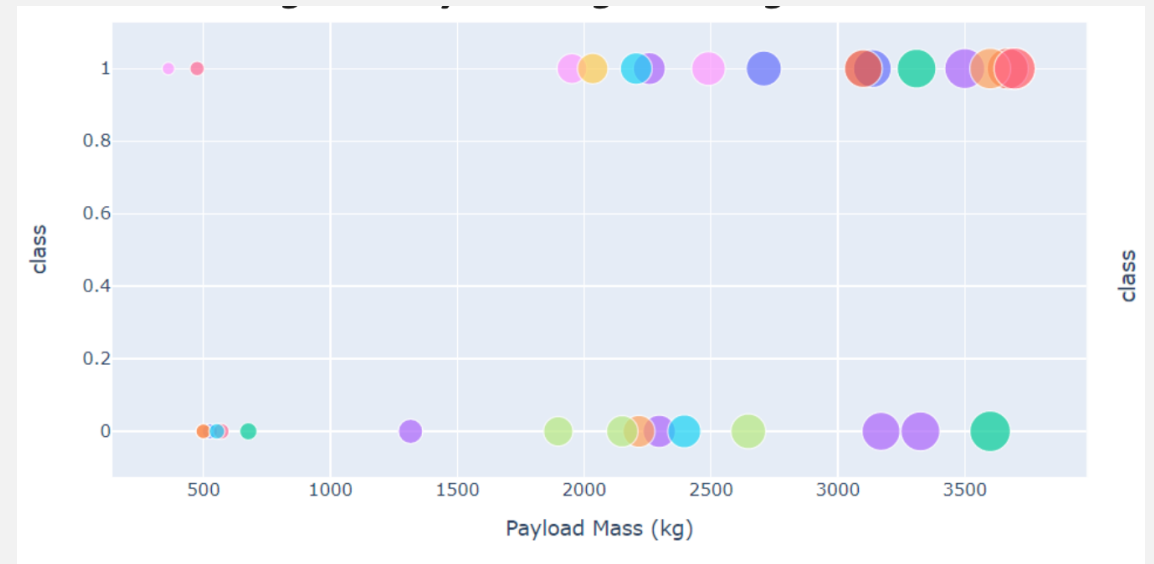
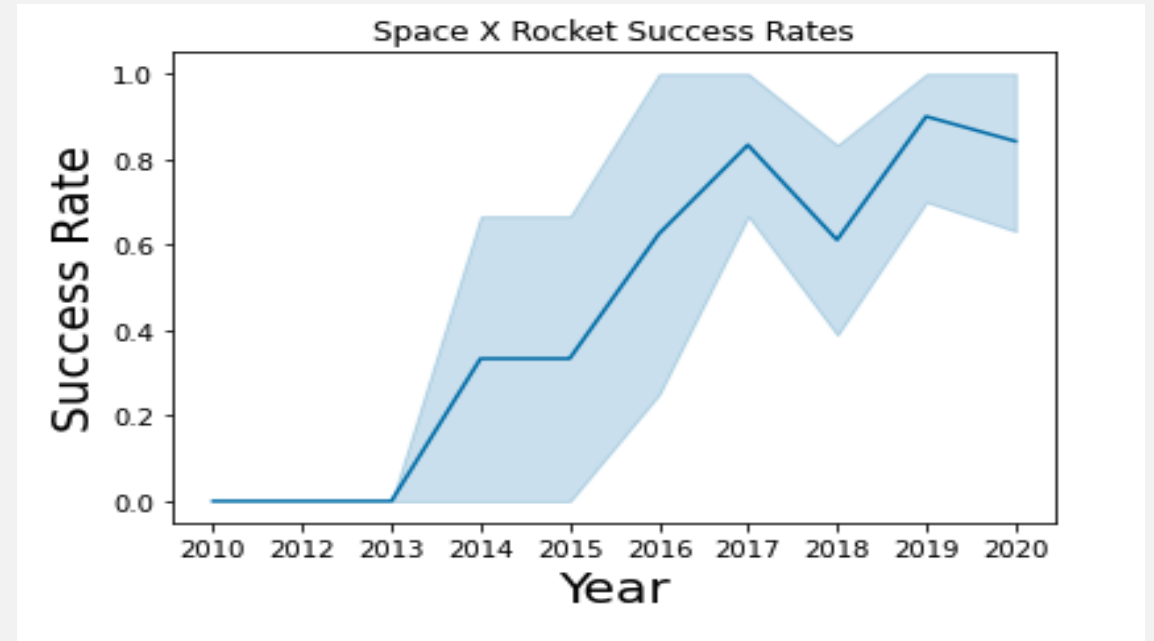
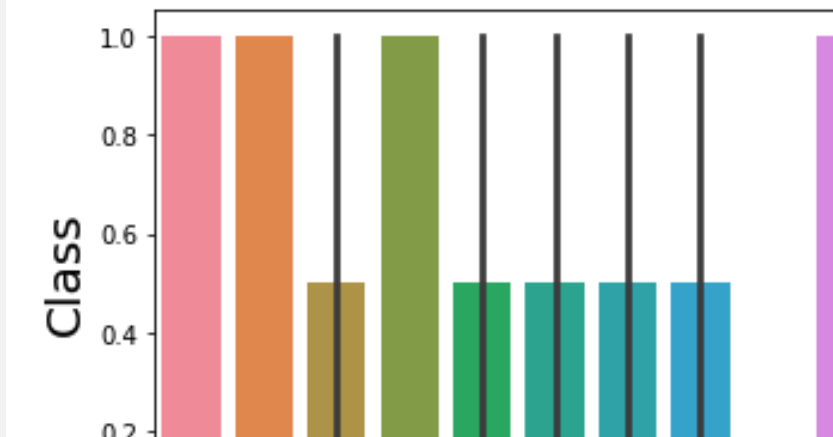
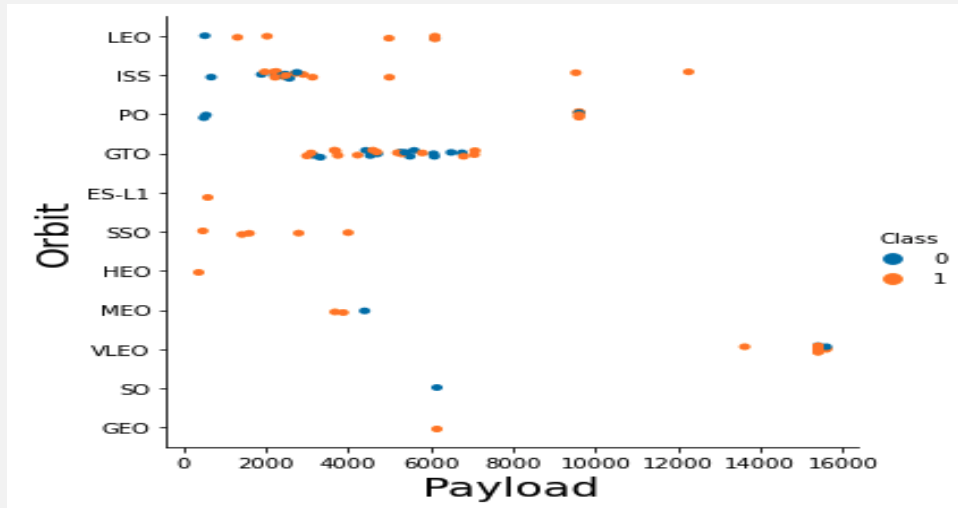
Folium MeasureControl Plugin Tool

Basic Decision Tree Construction

IBM cognos Visualization Tool

# Interactive Plotly

I used plotly because they are more interactive and easily customizable than seaborn

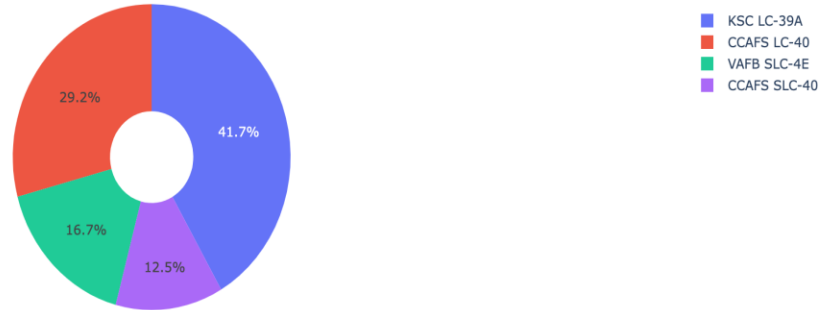


# SpaceX Launch Records Dashboard

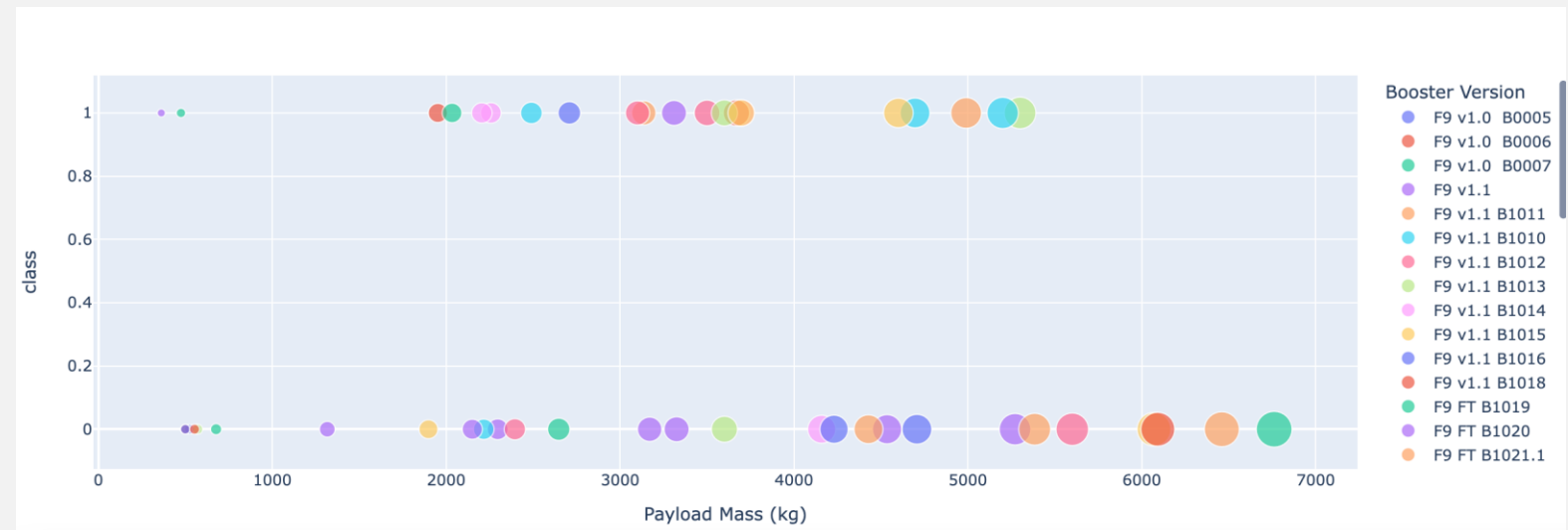
All Sites

×

Total Success Launches By all sites



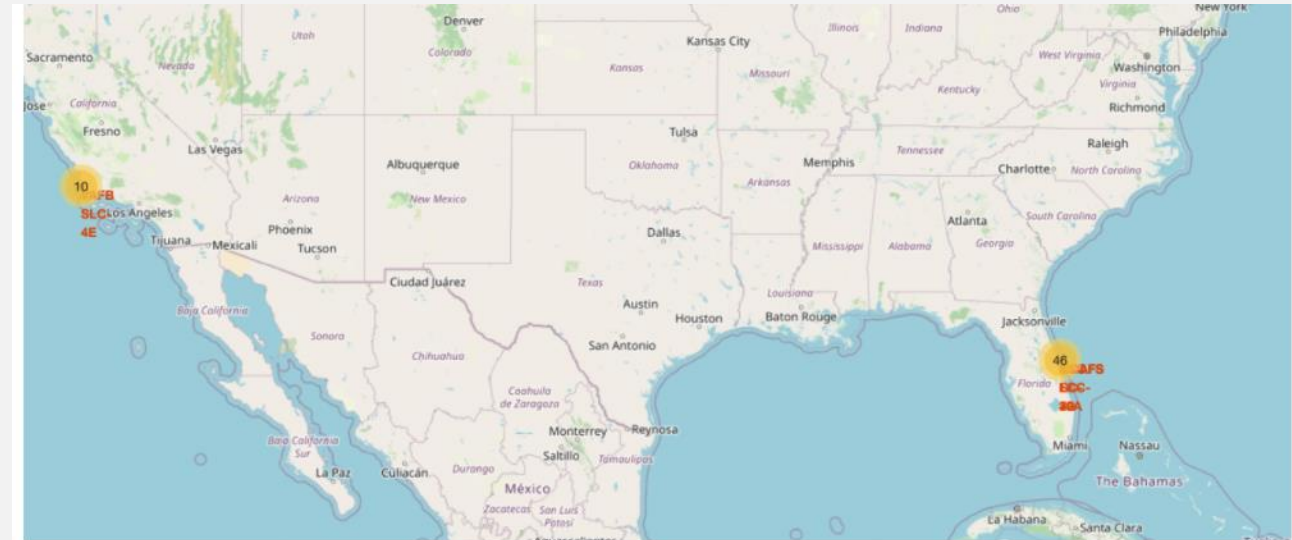
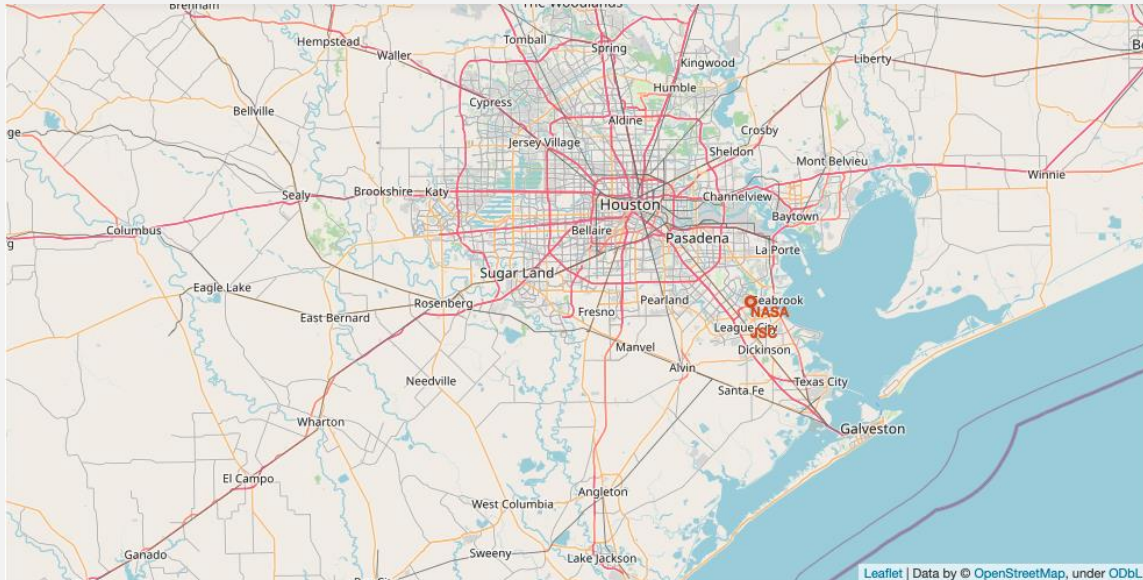
**Python Anywhere ..This enables a live site for Plotly Dashboard**



# Folium MeasureControl Plugin Tool

I used Folium custom to create custom layer to understand the location of the launch site

```
lines=folium.PolyLine(locations=coordinates, weight=1)
site_map.add_child(lines)
distance = calculate_distance(coordinates[0][0], coordinates[0][1], coordinates[1][0], coordinates[1][1])
distance_circle = folium.Marker(
    [28.56342, -80.56794],
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % "{:10.2f} KM".format(distance),
    )
)
site_map.add_child(distance_circle)
site_map
```





Thank you!

