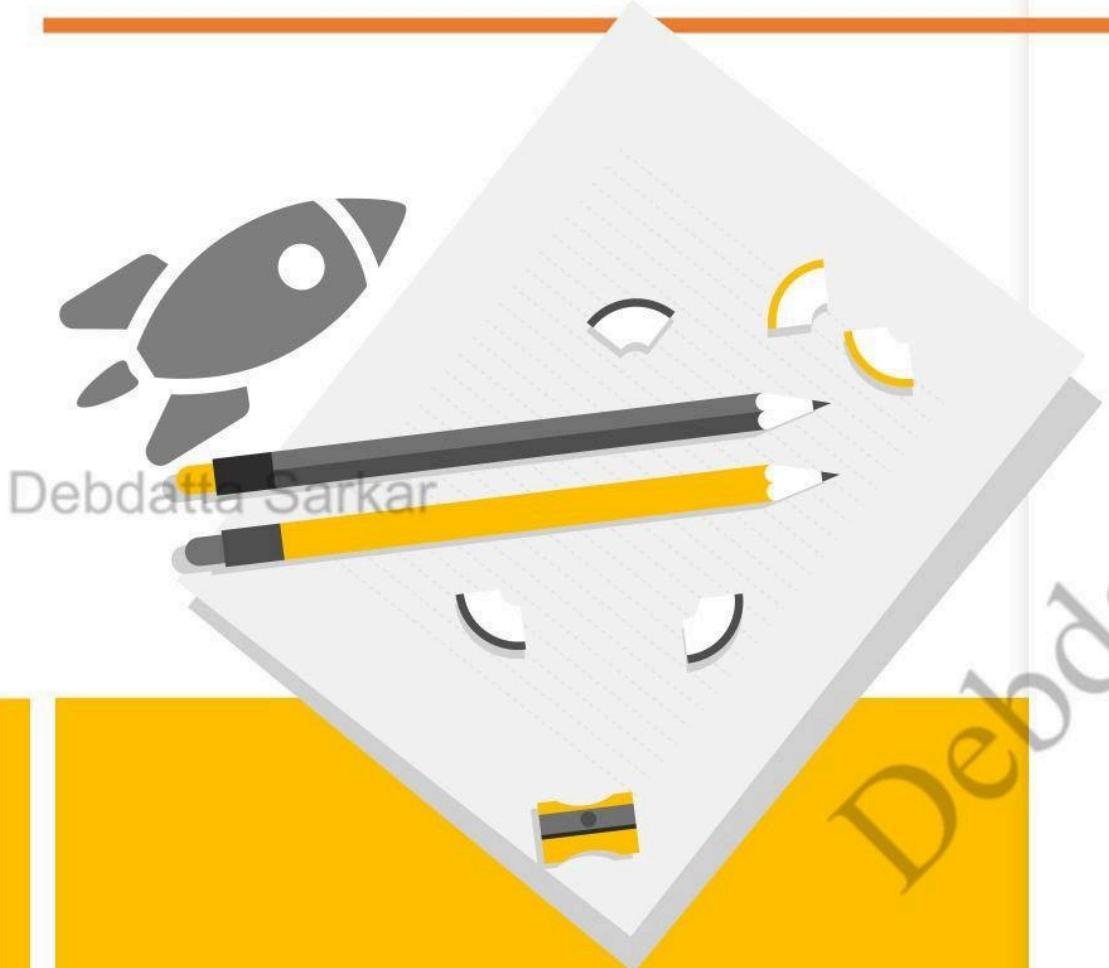


# IBM Data Science Capstone Project

Kota Varun Chand  
19<sup>th</sup> March 2024

# Outline

---



Executive Summary

Introduction

Methodology

Results

Conclusion

Appendix

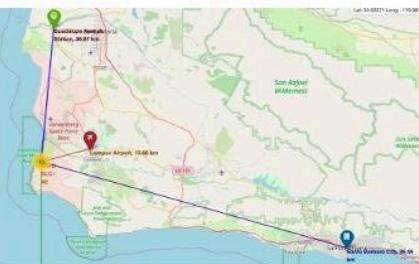
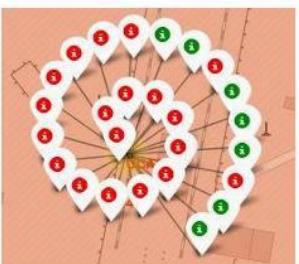
# Executive Summary

## ■ Summary of methodologies -

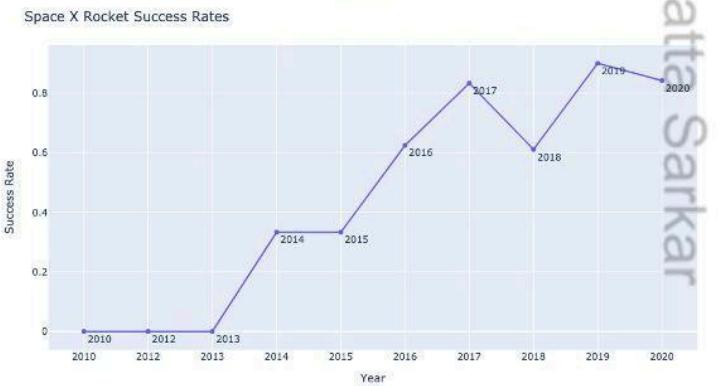
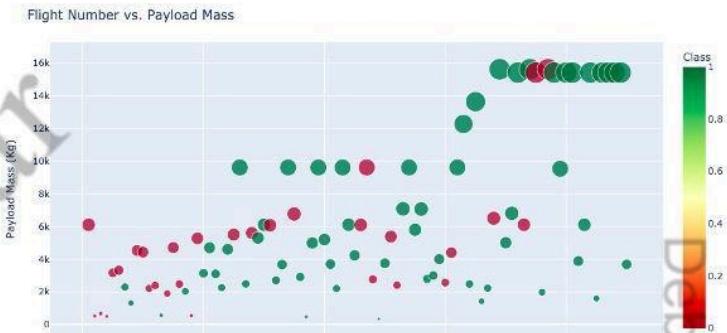
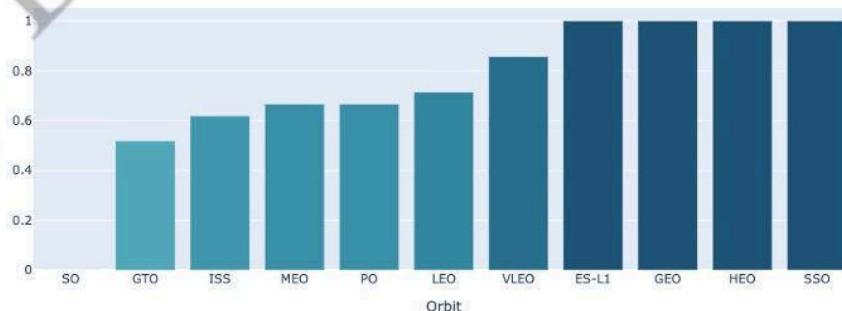
- Data Collection via API, SQL and Web Scraping
- Data Wrangling and Analysis
- Interactive Maps with Folium
- Predictive Analysis for each classification model

## ■ Summary of all results -

- Data Analysis along with Interactive Visualizations
- Best model for Predictive Analysis



Success Rate



# Introduction

---



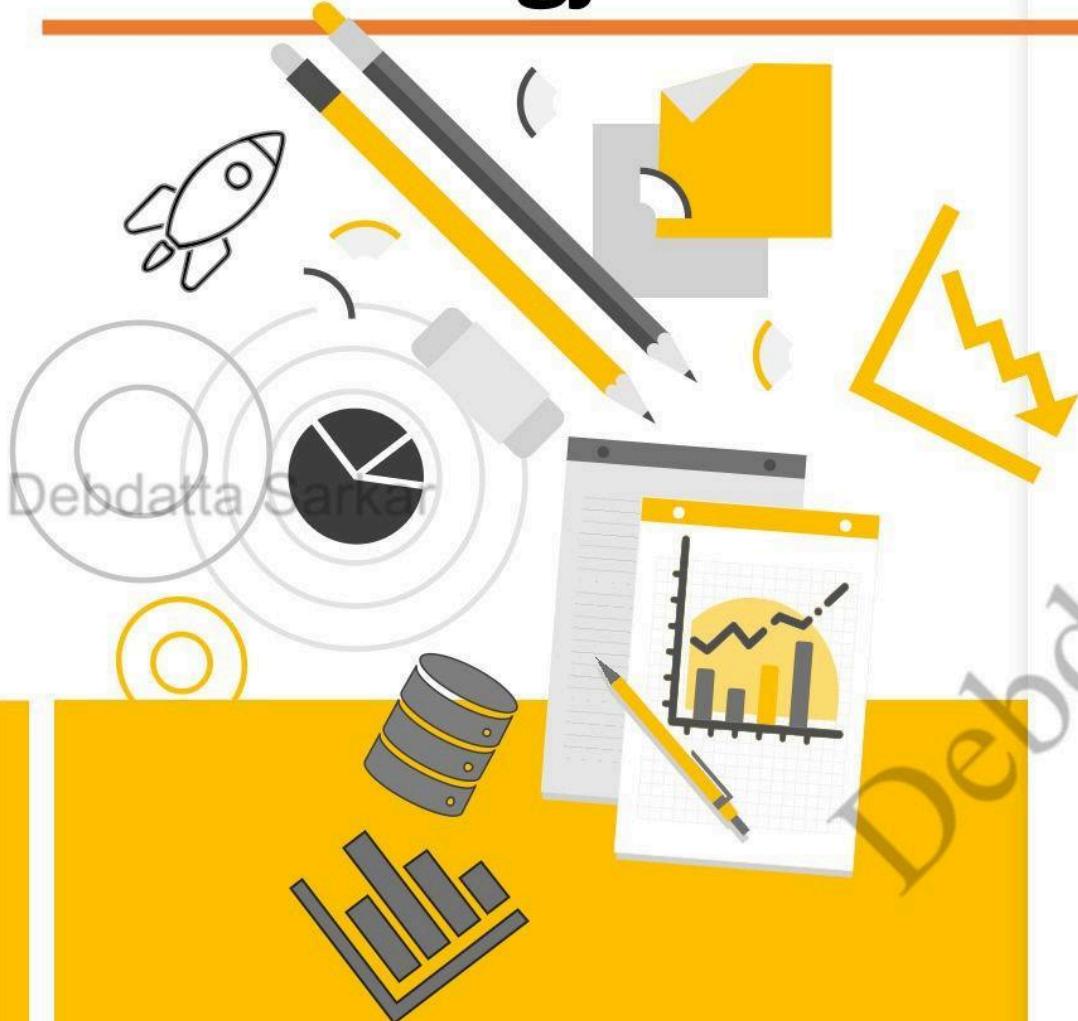
## ▪ Project background and context:

Here we will predict if the Falcon 9 first stage will land successfully. SpaceX advertises Falcon 9 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 successfully. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

## ▪ Problems we want to find answers:

- With what factors, the rocket will land successfully?
- The effect of each relationship of rocket variables on outcome.
- Conditions which will aid SpaceX have to achieve the best results.

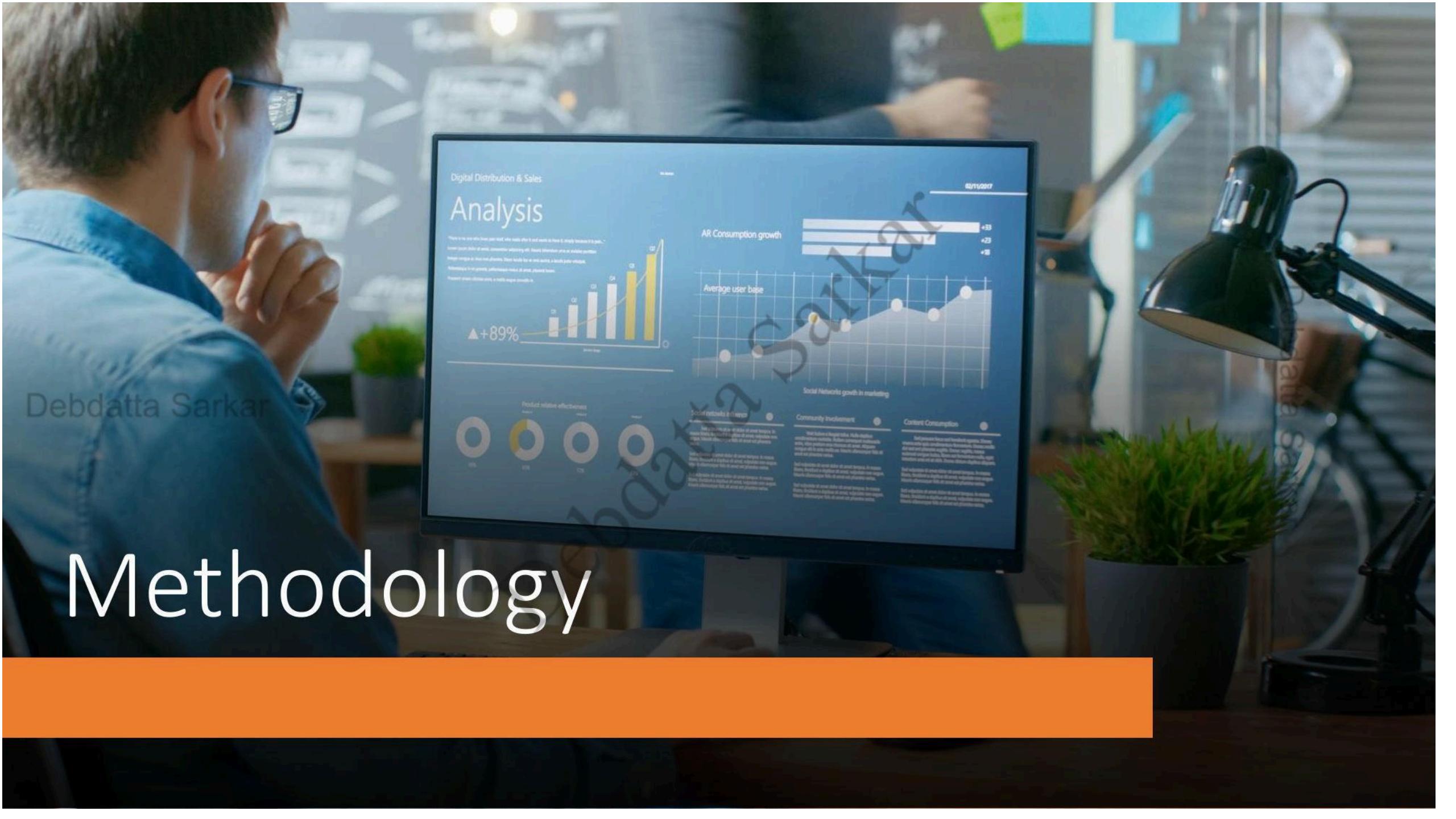
# Methodology



- **Data collection methodology:**
  - Via SpaceX Rest API
  - Web Scrapping from [Wikipedia](#)
- **Perform data wrangling:**
  - One hot encoding data fields for machine learning and dropping irrelevant columns (Transforming data for Machine Learning)
- **Perform exploratory data analysis (EDA) using visualization and SQL:**
  - Scatter and bar graphs to show patterns between data
- **Perform interactive visual analytics:**
  - Using Folium and Plotly Dash Visualizations
- **Perform predictive analysis using classification models:**
  - Build and evaluate classification models

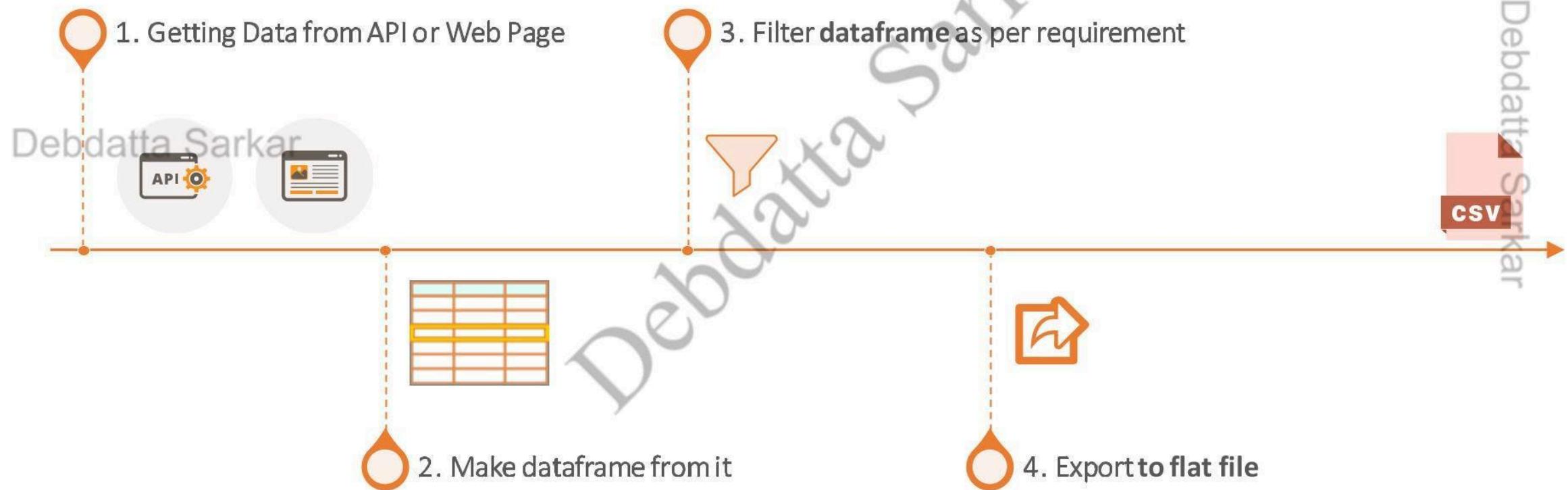
Debdatta Sarkar

# Methodology



# Data Collection - Meaning & Basic Steps

Data collection is the process of gathering and measuring information on targeted variables in an established system, which then enables one to answer relevant questions and evaluate outcomes.



# Data Collection - Via SpaceX API

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url)
```

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

```
data_falcon9.drop(data_falcon9[data_falcon9['BoosterVersion']!='Falcon 9'].index, inplace = True)
data_falcon9.loc[:, 'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```



```
jlist = requests.get(static_json_url).json()
df = pd.json_normalize(jlist)
df.head()
```

```
launch_dict = {'FlightNumber': list(data['flight_number']),
               'Date': list(data['date']),
               'BoosterVersion': BoosterVersion,
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
4	1	2010-06-04	Falcon 9	6123.547647	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003
5	2	2012-05-22	Falcon 9	525.000000	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0005
6	3	2013-03-01	Falcon 9	677.000000	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0007
7	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None	1.0	0	B1003
8	5	2013-12-03	Falcon 9	3170.000000	GTO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B1004

[GitHub URL](#)

# Data Collection - Via Web Scraping

Getting Response from HTML

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
data = requests.get(static_url).text
```

Creating BeautifulSoup Object

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

Finding tables

```
html_tables=soup.find_all("table")
first_launch_table = html_tables[2]
```

Getting column names

```
ths = first_launch_table.find_all('th')
for th in ths:
    name = extract_column_from_header(th)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

Creation of dictionary and appending data to keys

```
launch_dict= dict.fromkeys(column_names)
```

Converting dictionary to dataframe

[GitHub URL](#)

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	F9 v1.0B0003.1	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	NASA	Success	F9 v1.0B0005.1	No attempt	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success	F9 v1.0B0006.1	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success	F9 v1.0B0007.1	No attempt	1 March 2013	15:10

Dataframe to .CSV

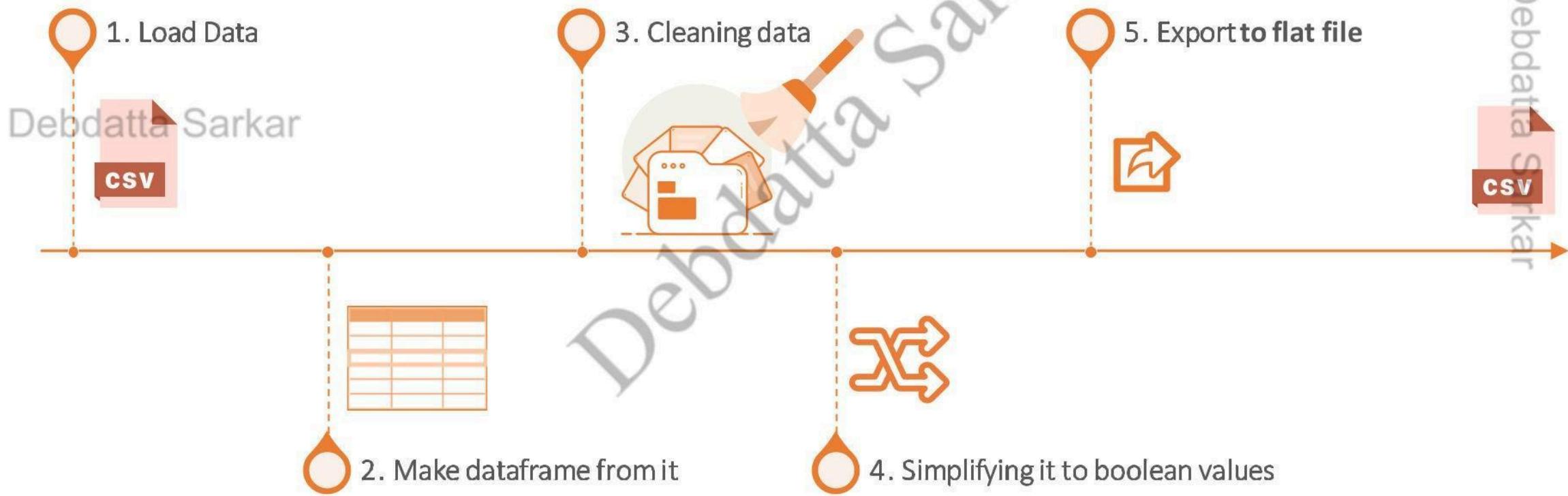
Debdatta Sarkar

# Data Wrangling - Meaning & Basic Steps

Data wrangling is the process of cleaning and unifying messy and complex data sets for easy access and analysis.

Here we mainly convert those outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful.

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



# Data Wrangling

Calculate number of launches at each site

```
df['LaunchSite'].value_counts()  
CCAFS SLC 40    55  
KSC LC 39A      22  
VAFB SLC 4E     13
```

```
df['Class'] = df['Outcome'].apply(lambda landing_class: 0 if landing_class in bad_outcomes else 1)  
df[['Class']].head(8)
```

Calculate number and occurrence of each orbit

```
df['Orbit'].value_counts()
```

```
GTO      27  
ISS      21  
VLEO     14  
PO       9  
LEO      7  
SSO      5  
MEO      3  
ES-L1    1  
HEO      1  
SO       1  
GEO      1
```

Calculate number and occurrence of mission outcome per orbit type

```
landing_outcomes = df['Outcome'].value_counts()  
landing_outcomes
```

```
True ASDS      41  
None None      19  
True RTLS      14  
False ASDS     6  
True Ocean     5  
None ASDS      2  
False Ocean    2  
False RTLS      1
```

Create landing outcome label from Outcome column

Export dataset as .CSV

```
df.to_csv("csvs/dataset_part_2.csv", index=False)
```

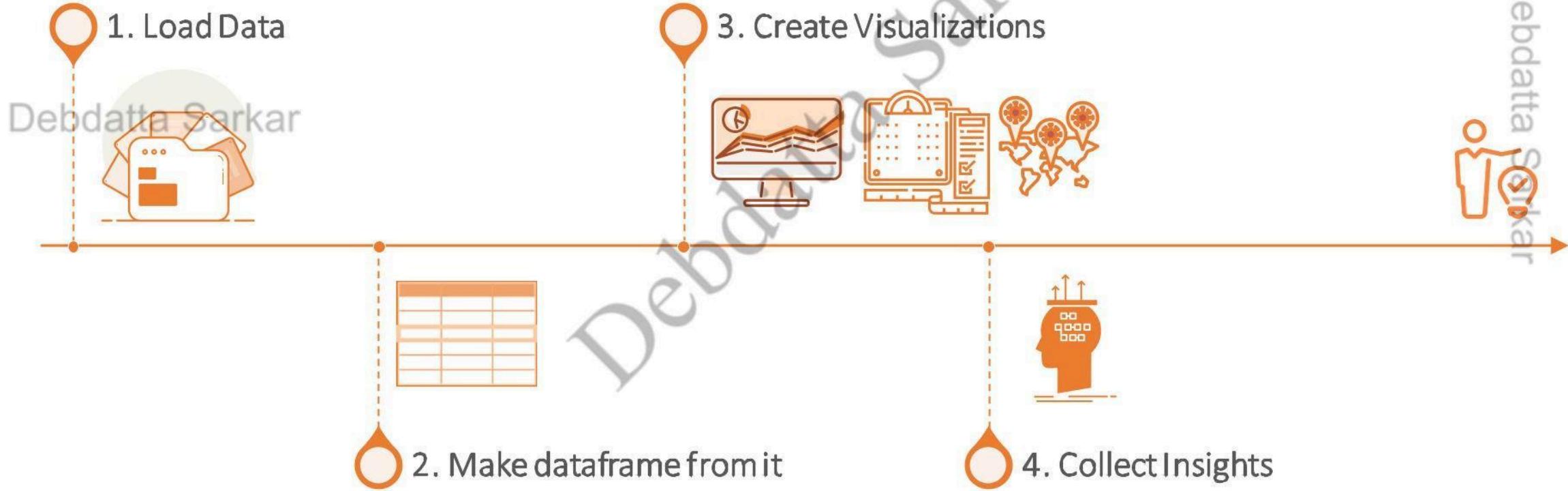
[GitHub URL](#)

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004

Debdatta Sarkar

# EDA - Meaning & Basic Steps

Exploratory data analysis is an approach of analyzing data sets to summarize their main characteristics, using statistical graphics and other data visualization methods.

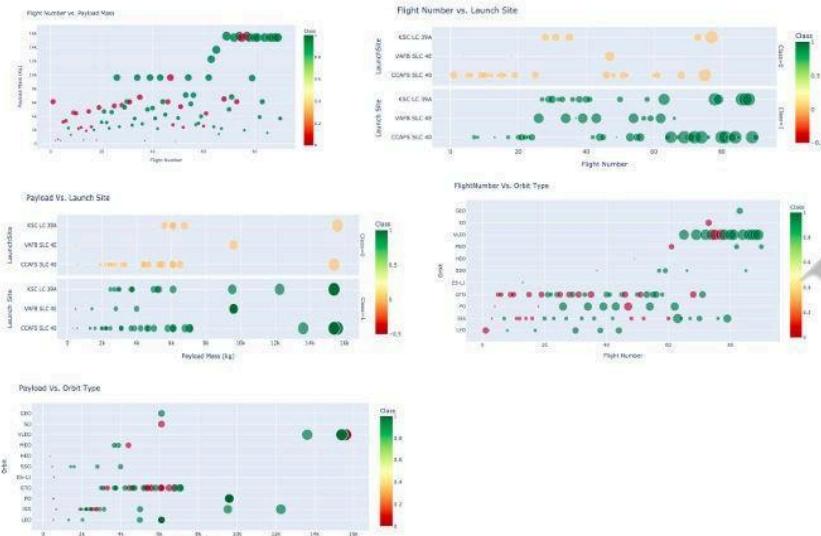


# EDA with Data Visualization

## Scatter Graphs Drawn:

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

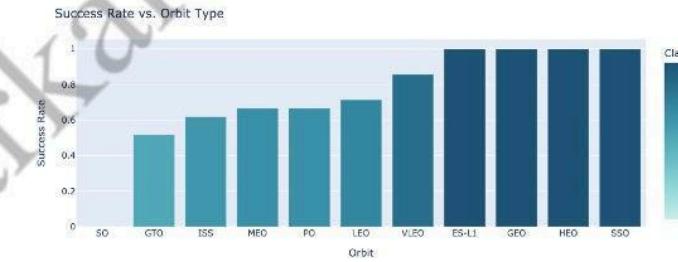
Scatter plots show dependency of attributes on each other. Once a pattern is determined from the graphs it's very easy to predict which factors will lead to maximum probability of success in both outcome and landing.



## Bar Graph Drawn:

### Success Rate VS. Orbit Type

Bar graphs are easiest to interpret a relationship between attributes. Via this bar graph we can easily determine which orbits have the highest probability of success.

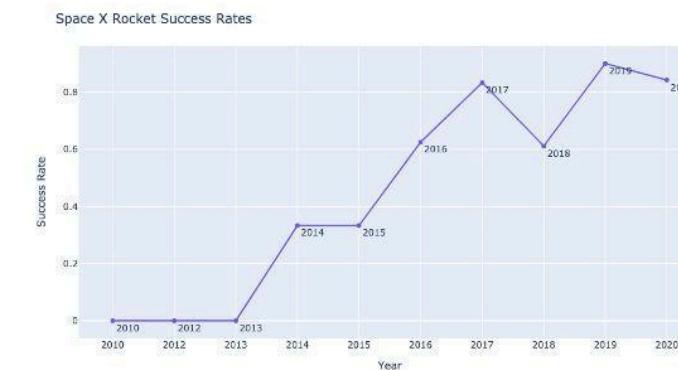


Debdatta Sarkar

## Line Graph Drawn:

### Launch Success Yearly Trend

Line graphs are useful in that they show trends clearly and can aid in predictions for the future.



[Seaborn Graphs GitHub URL](#)

[Plotly Graphs GitHub URL](#)

# EDA with SQL

SQL is an indispensable tool for Data Scientists and analysts as most of the real-world data is stored in databases. It's not only the standard language for Relational Database operations, but also an incredibly powerful tool for analyzing data and drawing useful insights from it. Here we use IBM's Db2 for Cloud, which is a fully managed SQL Database provided as a service.

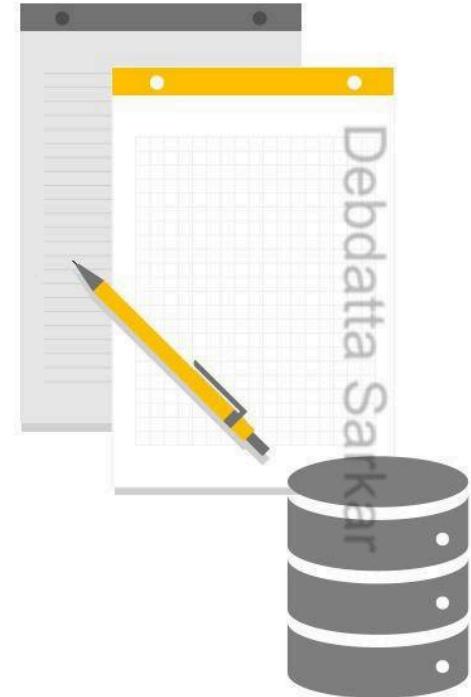
```
!pip install sqlalchemy==1.3.9
!pip install ibm_db_sa
!pip install ipython-sql
%load_ext sql

%sql ibm_db_sa://my-username:my-password@my-hostname:my-port/my-db-name
%sql <your query>
```

Debdatta Sarkar

We performed SQL queries to gather information from given dataset :

- Displaying the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- 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 failed landing\_outcomes in drone ship, their booster versions, and launch site names for the year 2015
- Ranking 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



[GitHub URL](#)

# Build an Interactive Map with Folium

Folium makes it easy to visualize data that's been manipulated 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. It is also 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	folium.Marker(	Map object to make a mark on map.
Icon Marker	folium.Icon(	Create an icon on map.
Circle Marker	folium.Circle(	Create a circle where Marker is being placed.
PolyLine	folium.PolyLine(	Create a line between points.
Marker Cluster Object	MarkerCluster()	This is a good way to simplify a map containing many markers having the same coordinate.
AntPath	folium.plugins.AntPath(	Create an animated line between points.

[Original Project - GitHub URL](#)

[Clean Distance Markers GitHub URL](#)



# Build a Dashboard with Plotly Dash

Pie Chart showing the total success for all sites or by certain launch site

- Percentage of success in relation to launch site

Scatter Graph showing the correlation between Payload and Success for all sites or by certain launch site

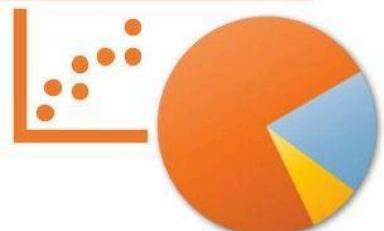
- It shows the relationship between Success rate and Booster Version Category.

Map Objects	Code	Result
Debdatta Sarkar	Dash and its components <pre>import dash import dash_html_components as html import dash_core_components as dcc from dash.dependencies import Input, Output</pre>	Plotly stewards Python's leading data viz and UI libraries. With Dash Open Source, Dash apps run on your local laptop or server. The Dash Core Component library contains a set of higher-level components like sliders, graphs, dropdowns, tables, and more. Dash provides all of the available HTML tags as user-friendly Python classes.
Pandas	<pre>import pandas as pd</pre>	Fetching values from CSV and creating a dataframe
Plotly	<pre>import plotly.express as px</pre>	Plot the graphs with interactive plotly library
Dropdown	<pre>dcc Dropdown(</pre>	Create a dropdown for launch sites
Rangeslider	<pre>dcc RangeSlider(</pre>	Create a rangeslider for Payload Mass range selection
Pie Chart	<pre>px pie(</pre>	Creating the Pie graph for Success percentage display
Scatter Chart	<pre>px scatter(</pre>	Creating the Scatter graph for correlation display

[GitHub Code URL](#)

Used "Python Anywhere" to host a live website. The live site dashboard is built with Flask and Dash.

[Live Site URL](#)



# Predictive Analysis (Classification)

## Building Model

- Load our feature engineered data into dataframe
- Transform it into NumPy arrays
- Standardize and transform data
- Split data into training and test data sets
- Check how many test samples has been created
- List down 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

```
y = data['Class'].to_numpy()
transform = preprocessing.StandardScaler()
X = transform.fit(X).transform(X)
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.2, random_state=2)
Y_test.shape
```

```
algorithms = {'KNN':knn_cv.best_score_',
              'Decision Tree':tree_cv.best_score_',
              'Logistic Regression':logreg_cv.best_score_}
best_algorithm = max(algorithms, key=lambda x: algorithms[x])
```

## Finding Best Performing Classification Model

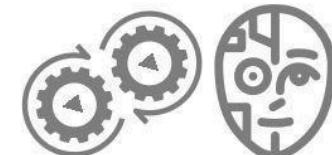
- The model with best accuracy score wins the best performing model

Debdatta Sarkar

## Evaluating Model

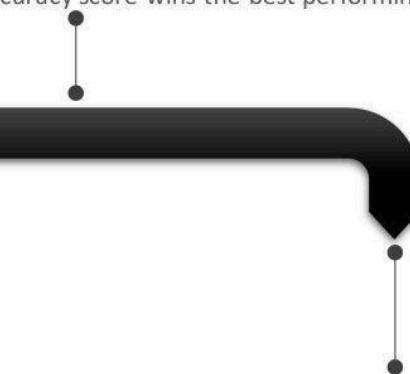
```
yhat=algorithm.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

- Check accuracy for each model
- Get best hyperparameters for each type of algorithms
- Plot Confusion Matrix



Best Model

[GitHub URL](#)



Debdatta Sarkar

# Results



Debdatta Sarkar

Exploratory data analysis results

Interactive analytics demo in screenshots

Predictive analysis results

Debdatta Sarkar

A woman with short dark hair and blue eyes is looking directly at the camera with a slight smile. She is wearing a white collared shirt. In front of her is a tablet device displaying various data visualizations, including bar charts and pie charts. The background is a blurred office environment.

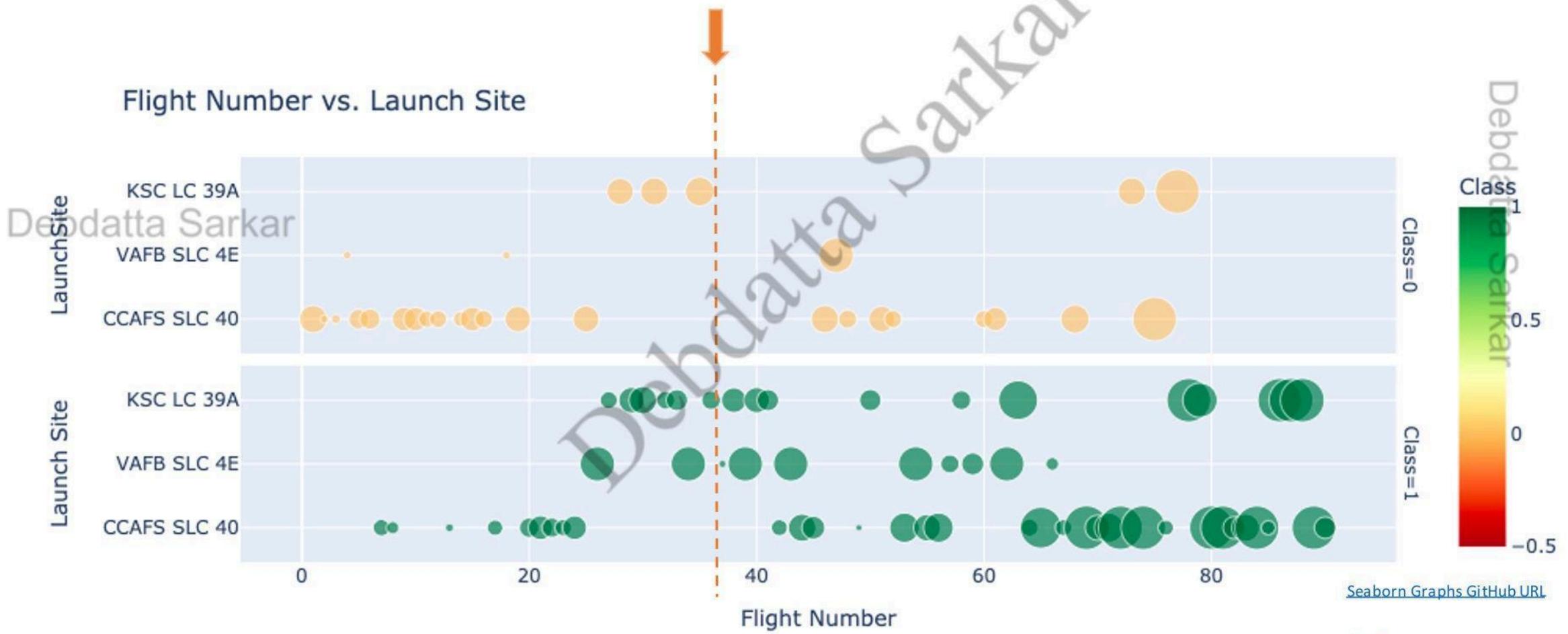
# EDA with Visualization

Debdatta Sarkar

Debdatta Sarkar

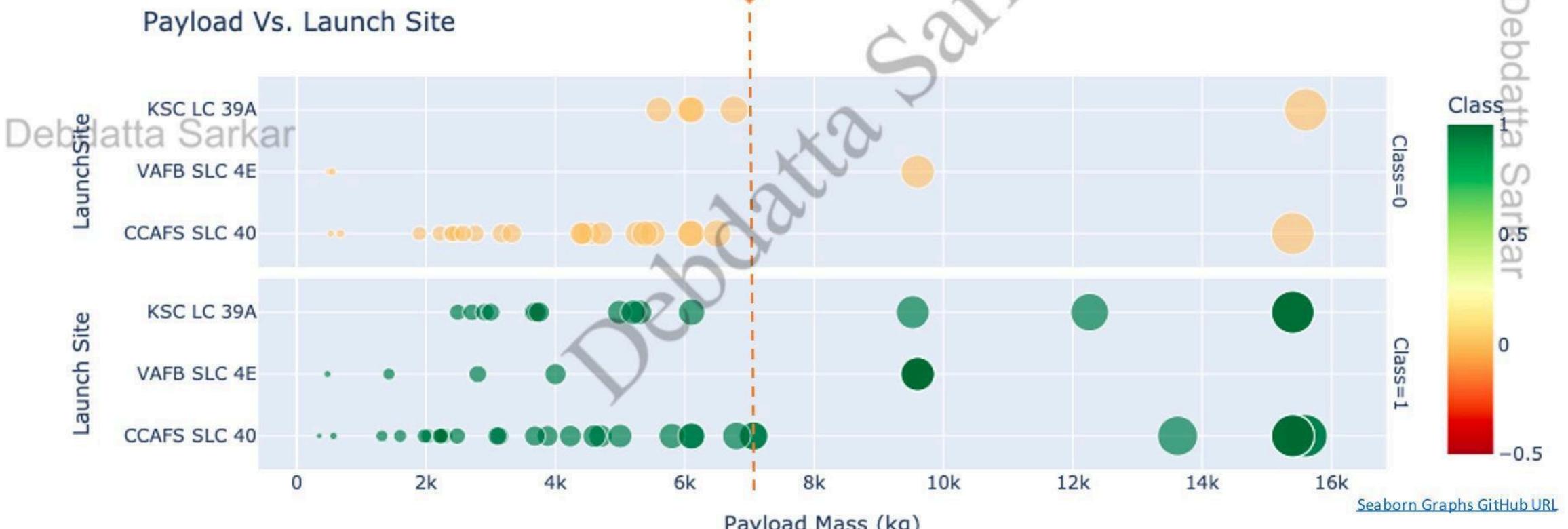
# Flight Number vs. Launch Site

- With higher flight numbers (greater than 30) the success rate for the Rocket is increasing.



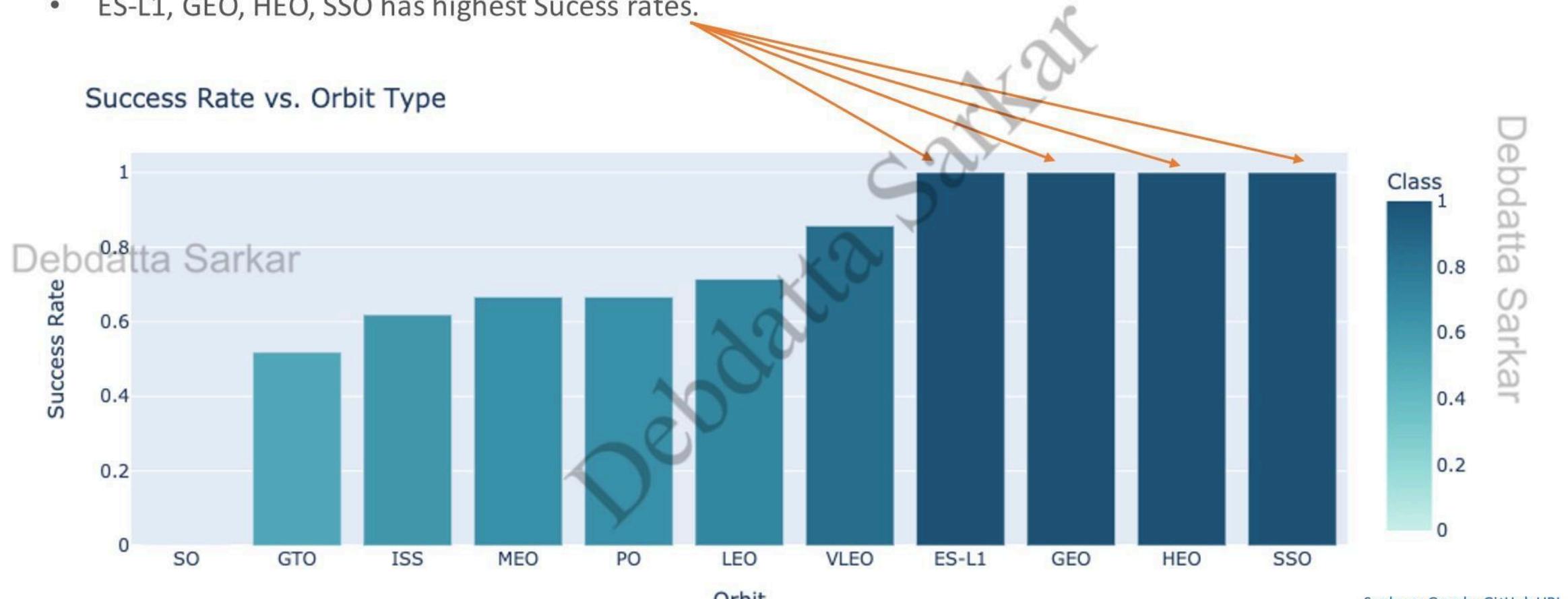
# Payload vs. Launch Site

- The greater the payload mass (greater than 7000 Kg) higher the success rate for the Rocket. But there's no clear pattern to take a decision, if the launch site is dependent on Pay Load Mass for a success launch.



# Success Rate vs. Orbit Type

- ES-L1, GEO, HEO, SSO has highest Success rates.

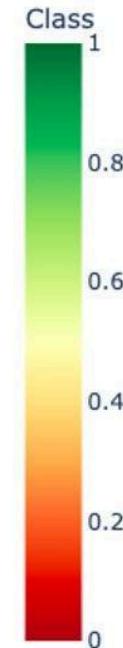
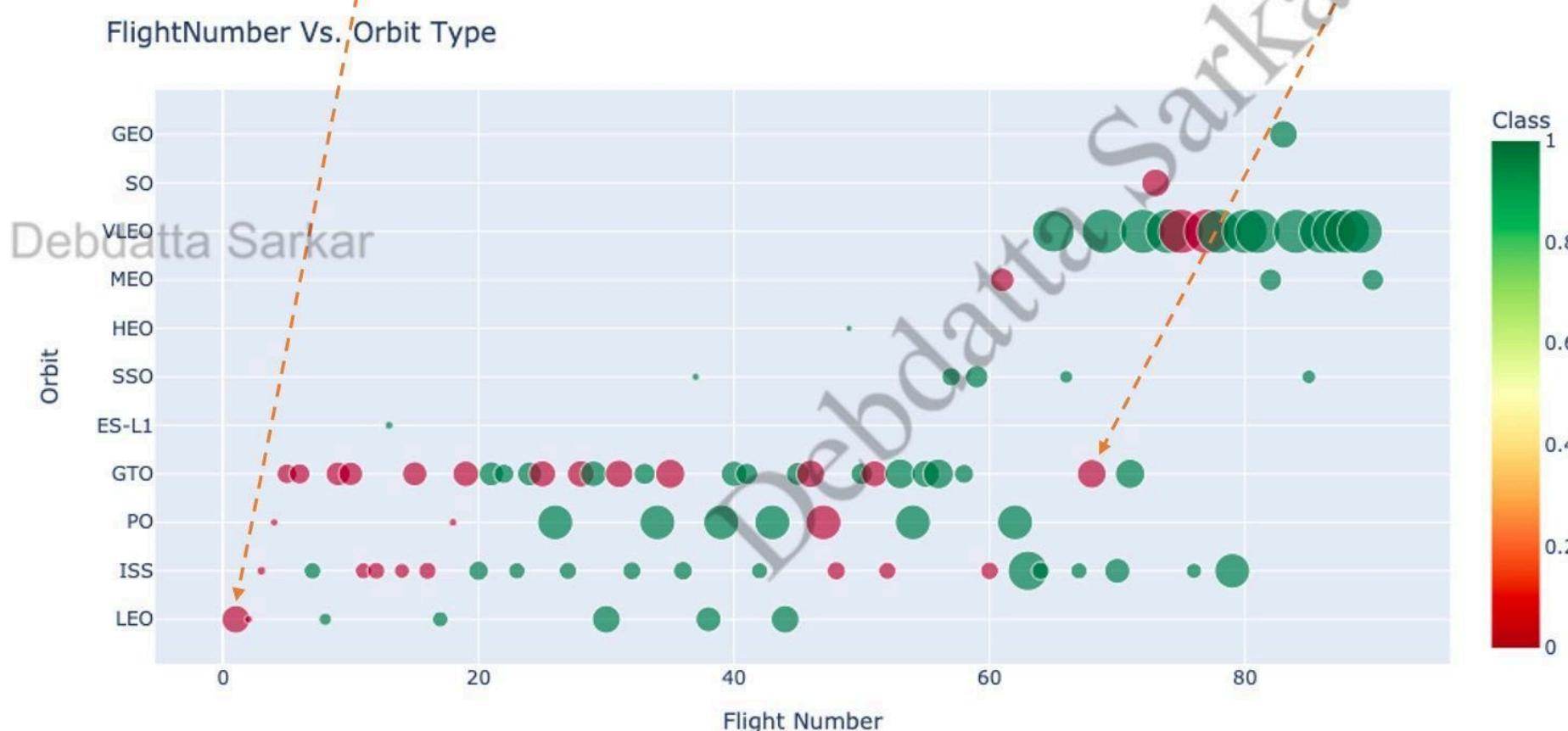


[Seaborn Graphs GitHub URL](#)

[Plotly Graphs GitHub URL](#)

# Flight Number vs. Orbit Type

- We see that for **LEO** orbit the success increases with the number of flights
- On the other hand, there seems to be no relationship between flight number and the **GTO** orbit.



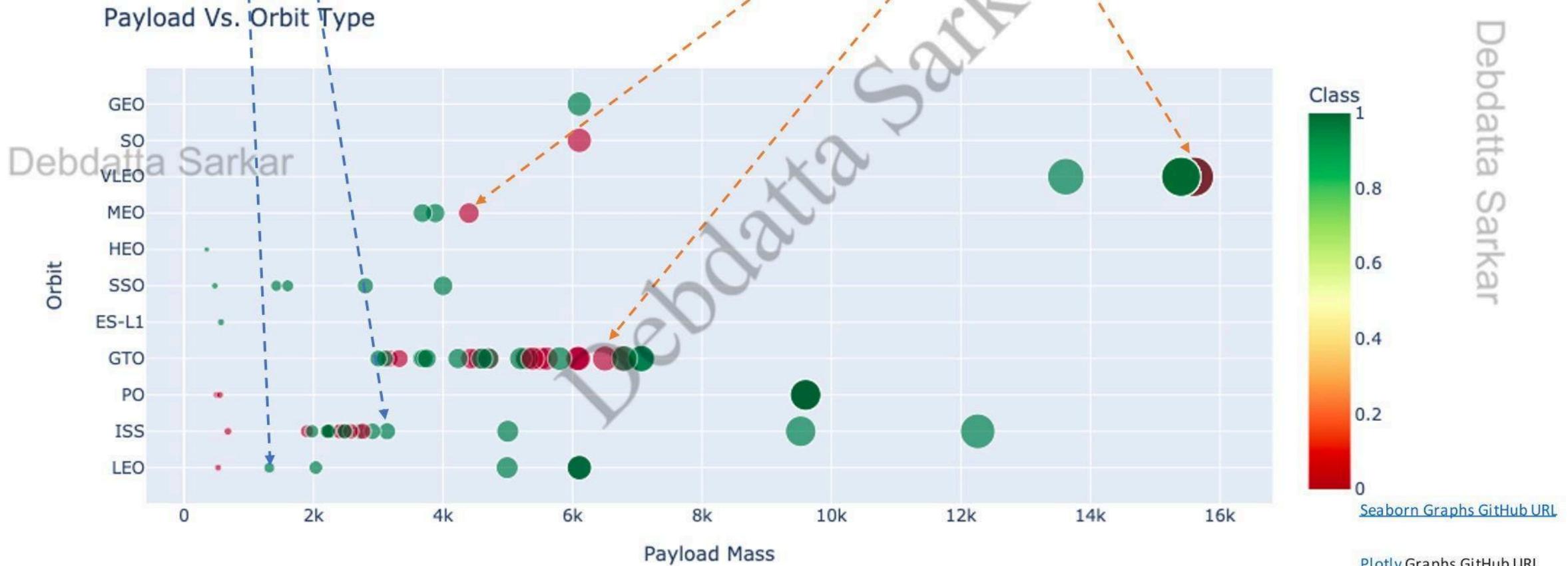
Debdatta Sarkar  
Debdatta Sarkar  
Debdatta Sarkar

[Seaborn Graphs GitHub URL](#)

[Plotly Graphs GitHub URL](#)

# Payload vs. Orbit Type

- We observe that heavy payloads have a negative influence on MEO, GTO, VLEO orbits
- Positive on LEO, ISS orbits



# Launch Success Yearly Trend

We can observe that the success rate since 2013 kept increasing relatively though there is slight dip after 2019.

Space X Rocket Success Rates



[Seaborn Graphs GitHub URL](#)

[Plotly Graphs GitHub URL](#)

# EDA with SQL



# All Launch Site Names

---

## SQL Query

```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEX;
```

## Description

Using the word DISTINCT in the query we pull unique values for Launch\_Site column from table SPACEX.

### Launch\_Sites

CCAFS LC-40

CCAFS SLC-40

KSC LC-39A

VAFB SLC-4E

# Launch Site Names begin with 'CCA'

## SQL Query

```
%sql SELECT * FROM SPACEX WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

## Description

Using keyword 'LIMIT 5' in the query we fetch 5 records from table spacex and with condition LIKE keyword with wild card - 'CCA%' . The percentage in the end suggests 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
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_) AS "Total Payload Mass by NASA (CRS)" FROM SPACEX WHERE CUSTOMER = 'NASA (CRS)';
```

## Description

Using the function SUM calculates the total in the column PAYLOAD\_MASS\_KG\_ and WHERE clause filters the data to fetch Customer's by name "NASA (CRS)".

**Total Payload Mass by NASA (CRS)**

---

45596

# Average Payload Mass by F9 v1.1

## SQL Query

```
%sql SELECT AVG(PAYLOAD_MASS_KG_) AS "Average Payload Mass by Booster Version F9 v1.1" FROM SPACEX \n WHERE BOOSTER_VERSION = 'F9 v1.1';
```

## 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".

Average Payload Mass by Booster Version F9 v1.1

2928

# First Successful Ground Landing Date

## SQL Query

```
%sql SELECT MIN(DATE) AS "First Successful Landing Outcome in Ground Pad" FROM SPACEX \
WHERE LANDING_OUTCOME = 'Success (ground pad)' ;
```

## Description

Using the function MIN works out the minimum date in the column Date and WHERE clause filters the data to only perform calculations on Landing\_Outcome with values "Success (ground pad)".

**First Successful Landing Outcome in Ground Pad**

2015-12-22

# Successful Drone Ship Landing with Payload between 4000 and 6000

## SQL Query

```
%sql SELECT BOOSTER_VERSION FROM SPACEX WHERE LANDING_OUTCOME = 'Success (drone ship)' \
AND PAYLOAD_MASS_KG_ > 4000 AND PAYLOAD_MASS_KG_ < 6000;
```

## Description

Selecting only Booster\_Version,  
WHERE clause filters the dataset to Landing\_Outcome= Success (drone ship)

AND clause specifies additional filter conditions  
Payload\_MASS\_KG\_ > 4000 AND Payload\_MASS\_KG\_ < 6000

**booster\_version**

F9 FT B1022

F9 FT B1026

F9 FT B1021.2

F9 FT B1031.2

Debdatta Sarkar

# Total number of Successful and Failure Mission Outcomes

---

## SQL Query

```
%sql SELECT sum(case when MISSION_OUTCOME LIKE '%Success%' then 1 else 0 end) AS "Successful Mission",  
       sum(case when MISSION_OUTCOME LIKE '%Failure%' then 1 else 0 end) AS "Failure Mission" \  
FROM SPACEX;
```

Debdatta Sarkar

## Description

Selecting multiple count is a complex query. I have used case clause within sub query for getting both success and failure counts in same query.

Case when MISSION\_OUTCOME LIKE '%Success%' then 1 else 0 end" returns a Boolean value which we sum to get the result needed.

Debdatta Sarkar

Successful Mission	Failure Mission
100	1

# Boosters carried Maximum Payload

## SQL Query

```
%sql SELECT DISTINCT BOOSTER_VERSION AS "Booster Versions which carried the Maximum Payload Mass" FROM SPACEX \
WHERE PAYLOAD_MASS_KG_ =(SELECT MAX(PAYLOAD_MASS_KG_) FROM SPACEX);
```

## Description

Using the function MAX works out the maximum payload in the column PAYLOAD\_MASS\_KG\_ in sub query.

WHERE clause filters Booster Version which had that maximum payload.

Booster Versions which carried the Maximum Payload Mass
F9 B5 B1048.4
F9 B5 B1048.5
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1049.7
F9 B5 B1051.3
F9 B5 B1051.4
F9 B5 B1051.6
F9 B5 B1056.4
F9 B5 B1058.3
F9 B5 B1060.2
F9 B5 B1060.3

# 2015 Launch Records

## SQL Query

```
%sql SELECT {fn MONTHNAME(DATE)} as "Month", BOOSTER_VERSION, LAUNCH_SITE FROM SPACEX WHERE year(DATE) = '2015' AND \
LANDING_OUTCOME = 'Failure (drone ship)';
```

## Description

Debdatta Sarkar

We need to 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.

Via year function we extract the year and future where clause 'Failure (drone ship)' fetches our required values.

Also, am using {fn MONTHNAME(DATE)} to get the Month name.

Month	booster_version	launch_site
January	F9 v1.1 B1012	CCAFS LC-40
April	F9 v1.1 B1015	CCAFS LC-40

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

## SQL Query

```
%sql SELECT LANDING_OUTCOME as "Landing Outcome", COUNT(LANDING_OUTCOME) AS "Total Count" FROM SPACEX \
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
GROUP BY LANDING_OUTCOME \
ORDER BY COUNT(LANDING_OUTCOME) DESC ;
```

Debdatta Sarkar

## Description

Selecting only LANDING\_OUTCOME,  
WHERE clause filters the data with DATE BETWEEN '2010-06-04' AND '2017-03-20'

Grouping by LANDING\_OUTCOME  
Order by COUNT(LANDING\_OUTCOME) in Descending Order.

Landing Outcome	Total Count
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

# Rank Success Count between 2010-06-04 and 2017-03-20

---

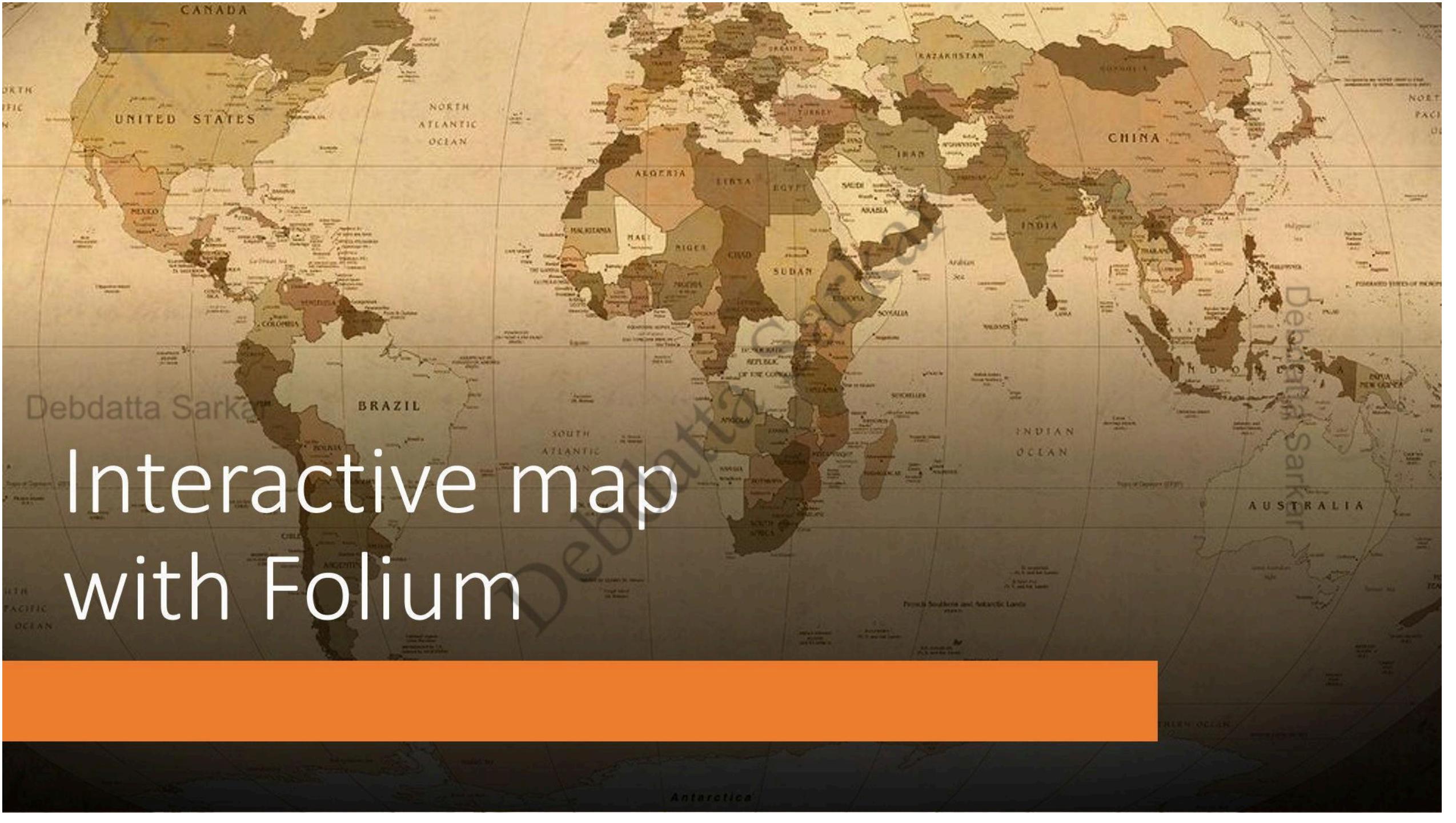
## SQL Query

```
%sql SELECT COUNT(LANDING_OUTCOME) AS "Rank success count between 2010-06-04 and 2017-03-20" FROM SPACEX^  
WHERE LANDING_OUTCOME LIKE '%Success%' AND DATE > '2010-06-04' AND DATE < '2017-03-20' ;
```

## Description

COUNT counts records in column LANDING\_OUTCOME  
WHERE filters data with '%Success%'  
AND DATE > '2010-06-04'  
AND DATE < '2017-03-20'

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



# Interactive map with Folium

# All Launch Sites on Folium Map

We can see that the SpaceX launch sites are near to the United States of America coasts i.e., Florida and California Regions.

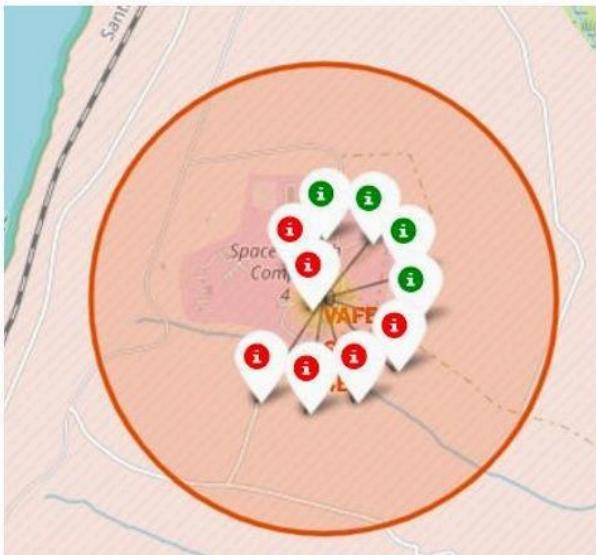


# Color Labeled Launch Records

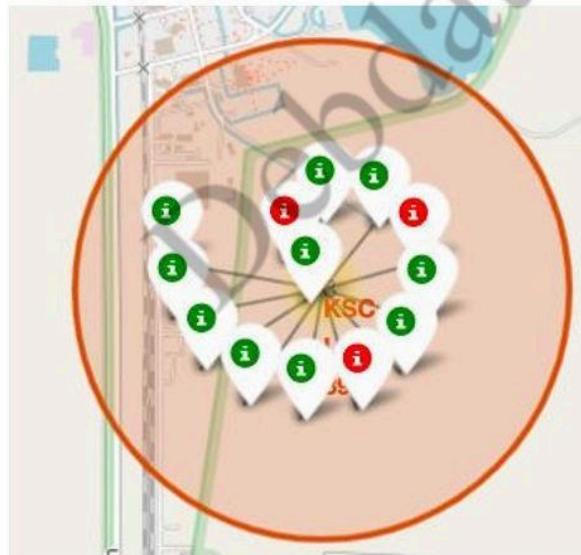


Debdatta Sarkar

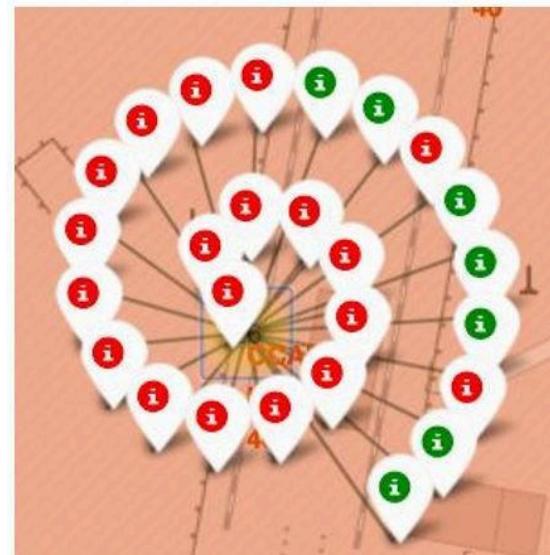
VAFB SLC-4E



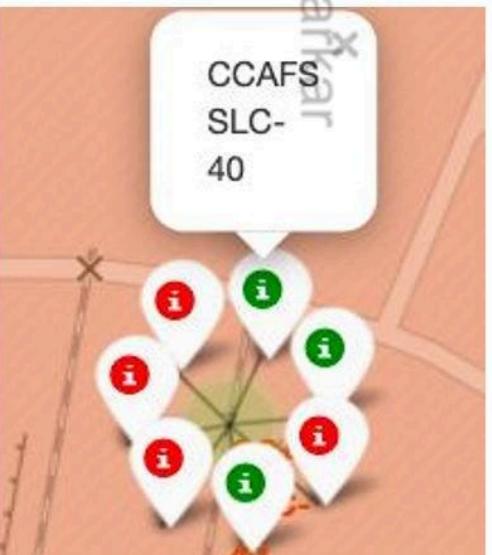
KSC LC-39A



CCAFS LC-40

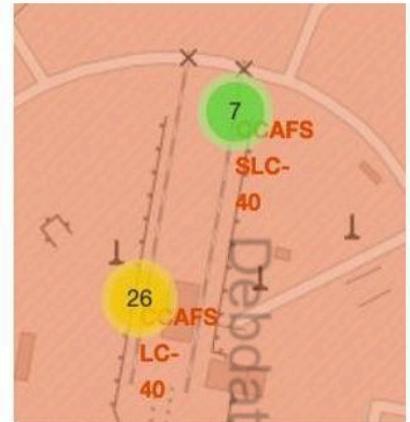


CCAFS SLC-40



Green Marker shows successful launches and Red Marker shows failures.

From these screenshots its easily understandable that KSC LC-39A has the maximum probability of success.

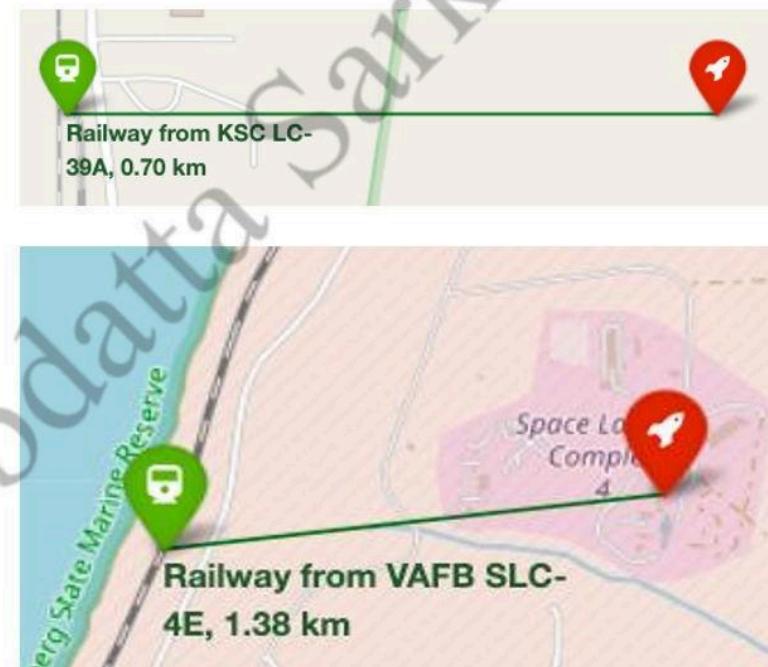


# Launch Site Distances from Equator & Railways

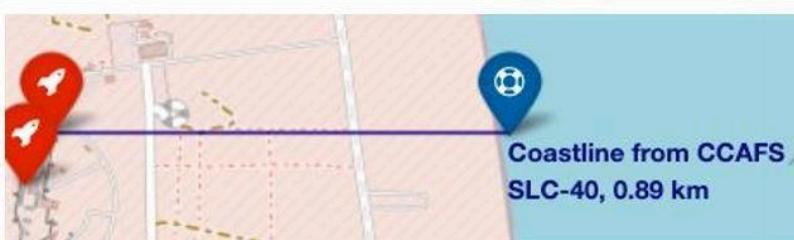
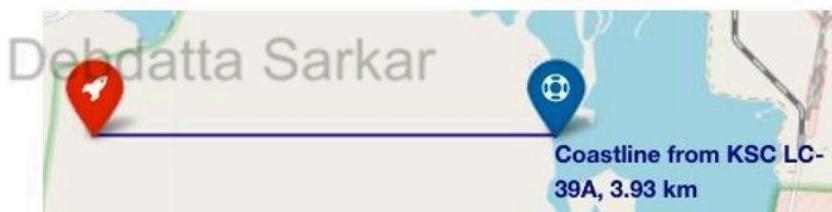
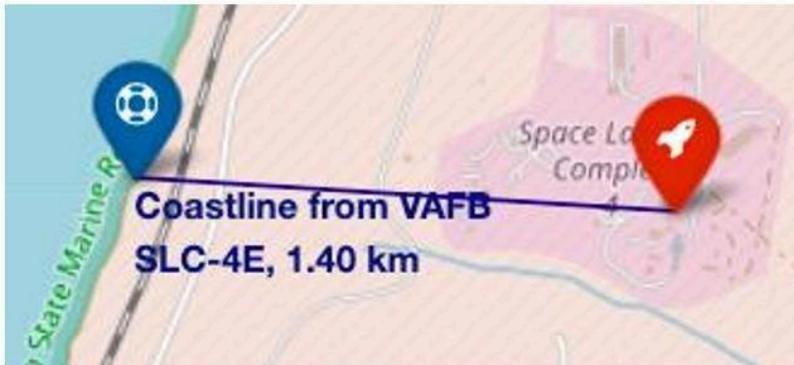
Distance from Equator is greater than 3000 Km for all sites.



Distance for all launch sites from railway tracks are greater than .7 Km for all sites. So, launch sites are not so far away from railway tracks.

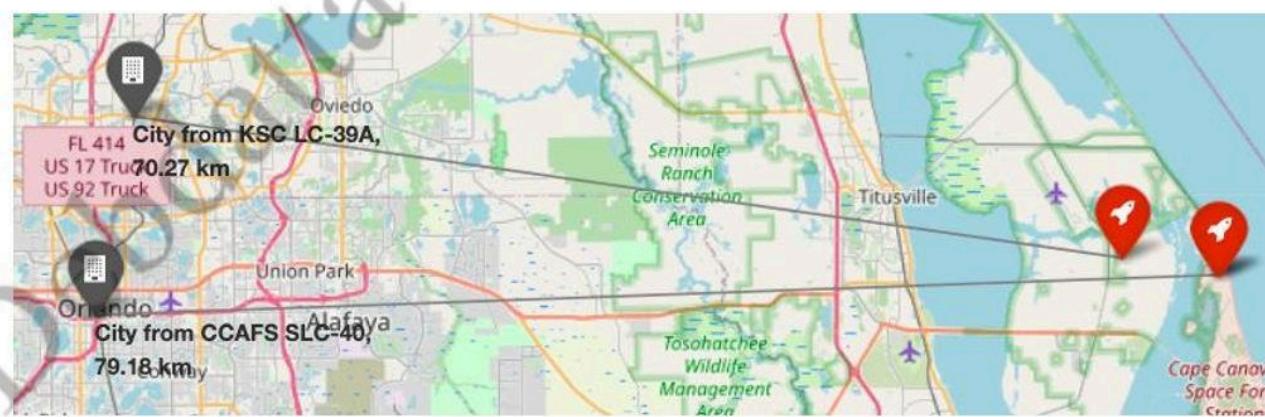


# Launch Site Distances from Coastlines & Cities



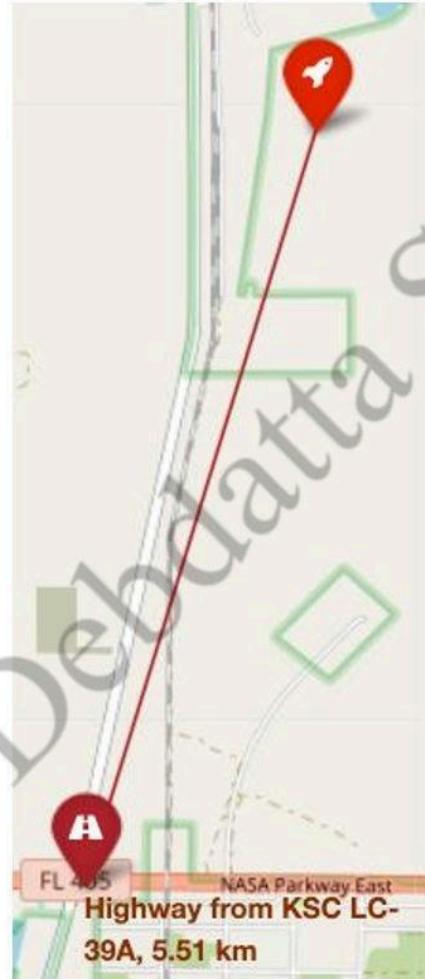
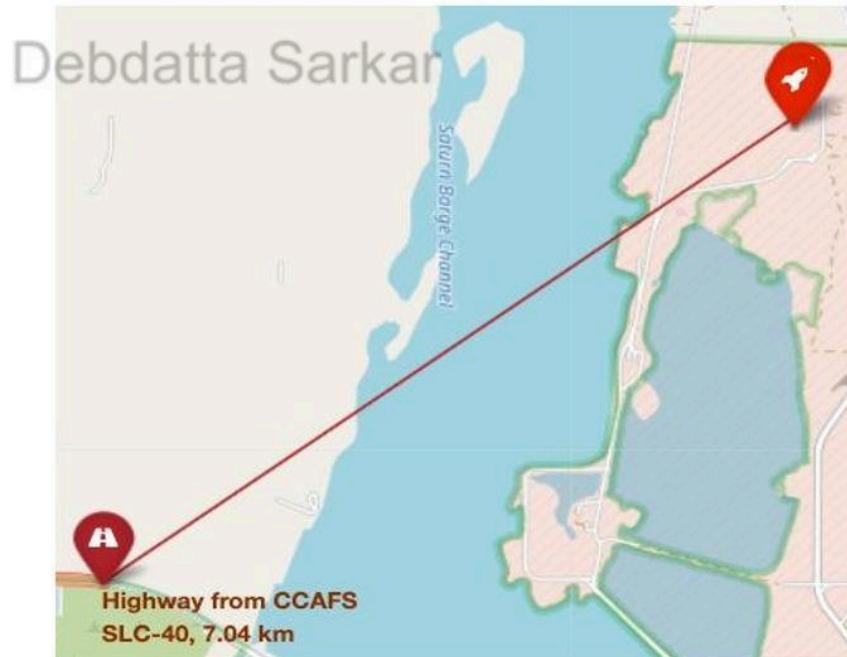
Distance for all launch sites from coastline is less than 4 Km.

Distance for all launch sites from cities is greater than 14 Km for all sites. So, launch sites are far away from cities.



Debdatta Sarkar

# Launch Site Distances from Highways

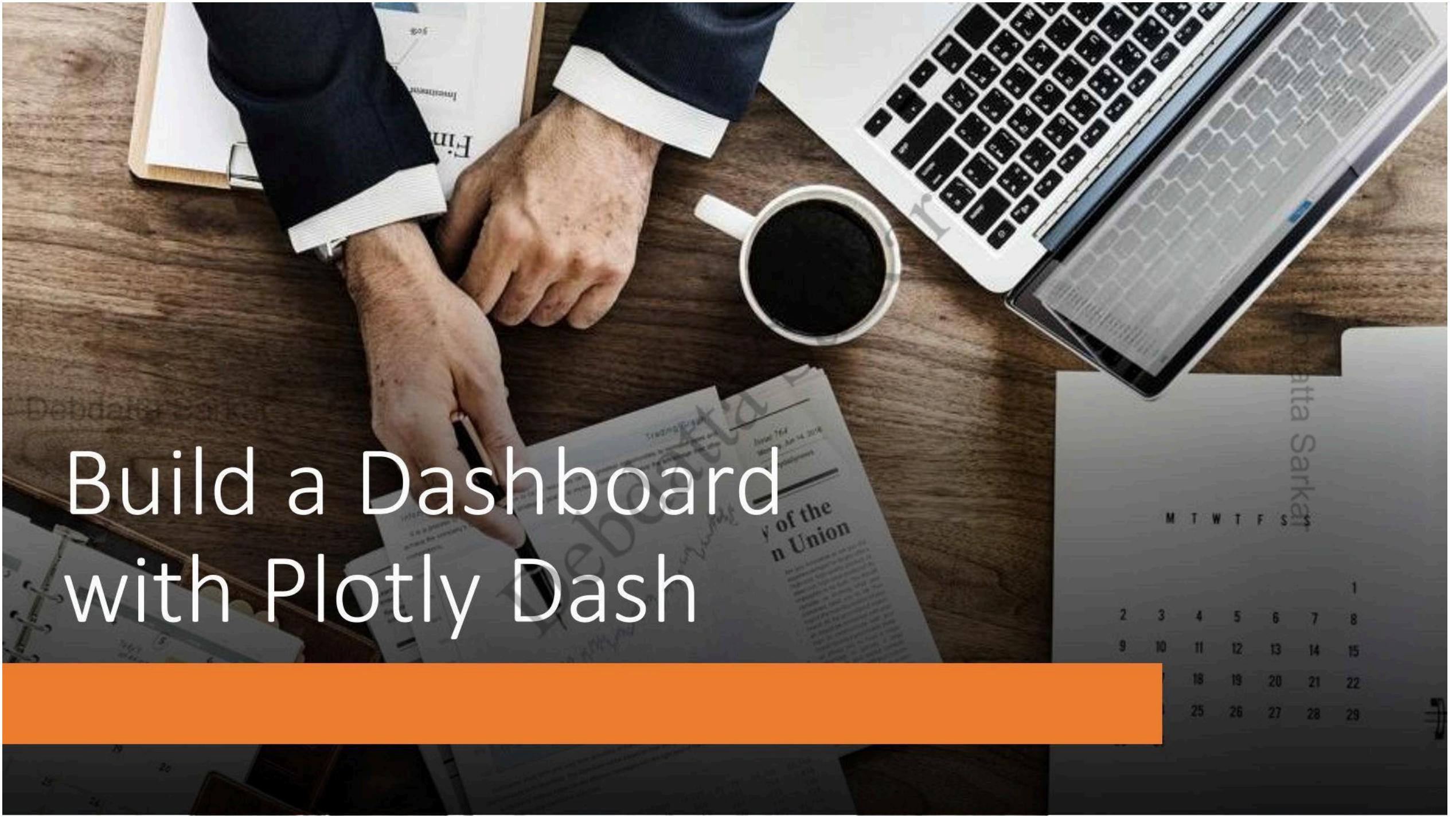


Distance for all launch sites from highways is greater than 5 Km for all sites. So, launch sites are relatively far away from highways.

## Conclusion:

- Are all launch sites in proximity to the Equator line?  
No ( $4000 \text{ Km} > \text{distance} > 3000 \text{ Km}$ )
- Are launch sites in close proximity to railways?  
Yes ( $2 \text{ Km} > \text{distance} > .5 \text{ Km}$ )
- Are launch sites in close proximity to highways?  
No ( $15 \text{ Km} > \text{distance} > 5 \text{ Km}$ )
- Are launch sites in close proximity to coastline?  
Yes ( $5 \text{ Km} > \text{distance} > .5 \text{ Km}$ )
- Do launch sites keep certain distance away from cities?  
Yes ( $15 \text{ Km} > \text{distance} > 80 \text{ Km}$ )

# Build a Dashboard with Plotly Dash

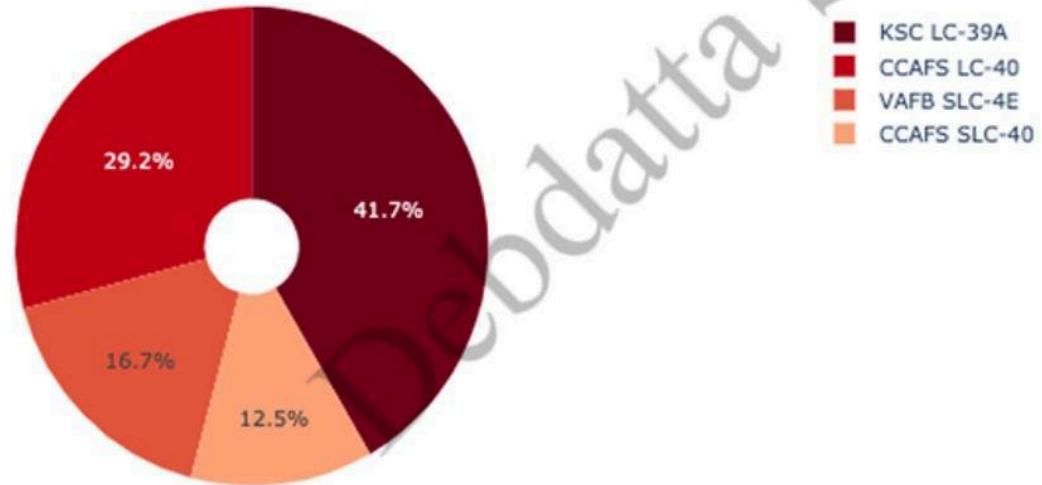


# Launch Success Count for All Sites

## SpaceX Launch Records Dashboard

All Sites

Total Success Launches by All Sites

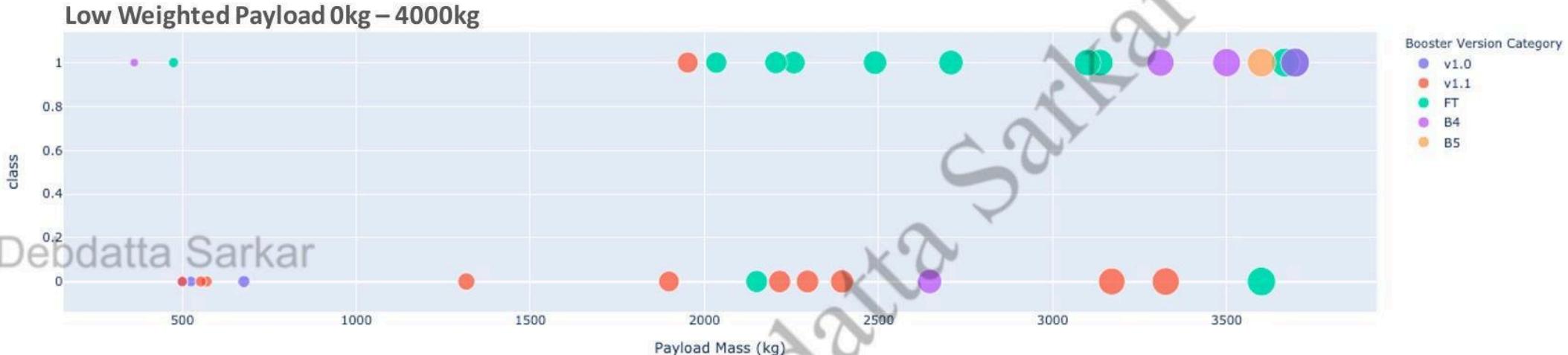


We can see that KSC LC-39A had the most successful launches from all the sites.

Debdatta Sarkar

# Payload vs. Launch Outcome Scatter Plot for All Sites

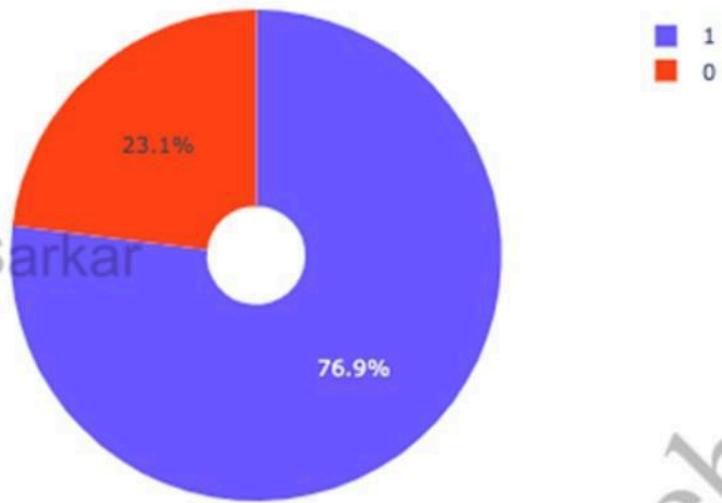
We can see the success rates for low weighted payloads is higher than the heavy weighted payloads



[Live Site URL](#) [Code](#)

# Launch Site with Highest Launch Success Ratio

Total Success Launches for Site → KSC LC-39A



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

After visual analysis using the dashboard, we are able to obtain some insights to answer these questions:

- Which site has the highest launch success rate?  
**KSC LC – 39A**
- Which payload range(s) has the highest launch success rate?  
**2000 Kg – 10000 Kg**
- Which payload range(s) has the lowest launch success rate?  
**0 Kg – 1000 Kg**
- Which F9 Booster version (v1.0, v1.1, FT, B4, B5, etc.) has the highest launch success rate?  
**FT**

Debdatta Sarkar



# Predictive analysis (Classification)

Debdatta Sarkar

# Confusion Matrix

Out here for all models unfortunately, we have same confusion matrix.

		Predicted Values		Total Cases = 18
		Predicted No	Predicted Yes	
Actual No	True Negative TN = 3	False Positive FP = 3	6	
	False Negative FN = 0	True Positive TP = 12	12	
	3	15		

**Accuracy:**  $(TP+TN)/\text{Total} = (12+3)/18 = 0.83333$

**Misclassification Rate:**  $(FP+FN)/\text{Total} = (3+0)/18 = 0.1667$

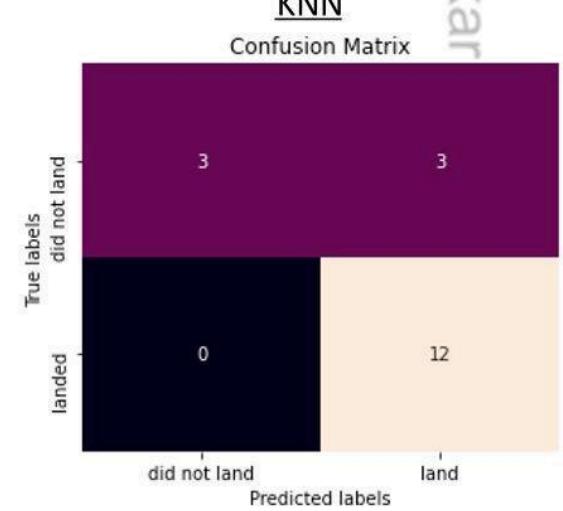
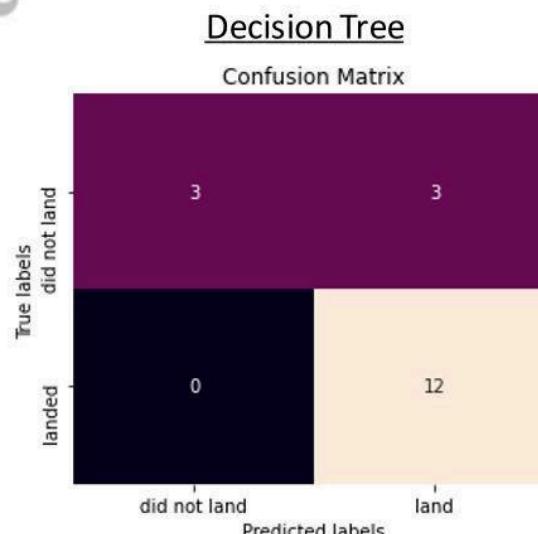
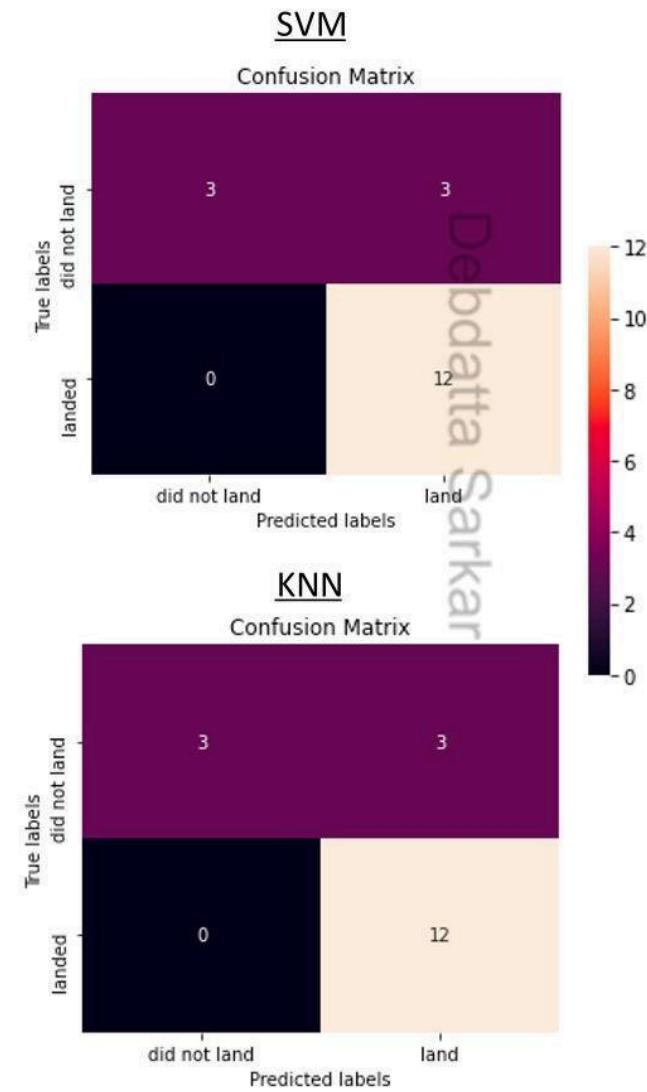
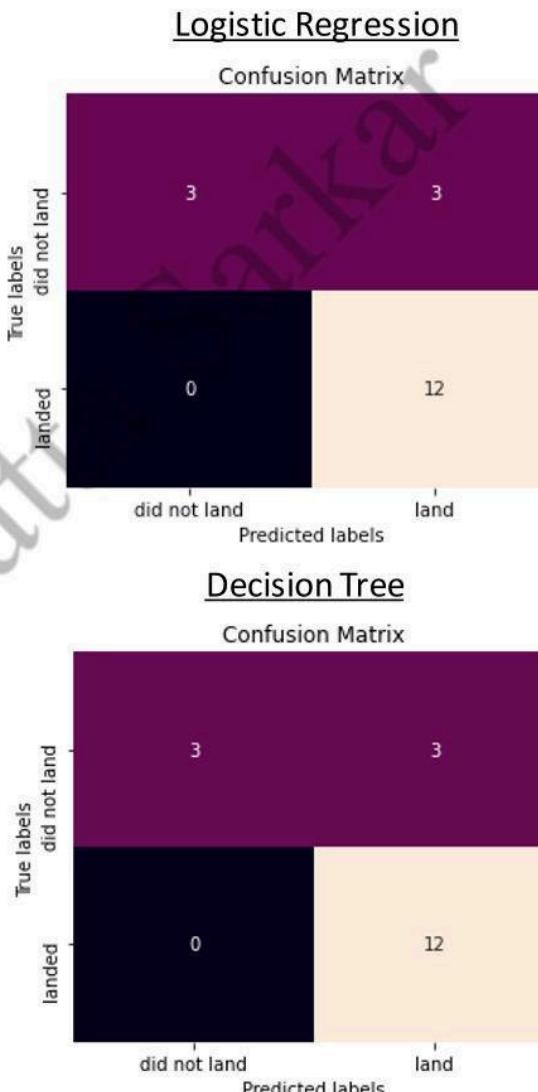
**True Positive Rate:**  $TP/\text{Actual Yes} = 12/12 = 1$

**False Positive Rate:**  $FP/\text{Actual No} = 3/6 = 0.5$

**True Negative Rate:**  $TN/\text{Actual No} = 3/6 = 0.5$

**Precision:**  $TP/\text{Predicted Yes} = 12/15 = 0.8$

**Prevalence:**  $\text{Actual yes}/\text{Total} = 12/18 = 0.6667$

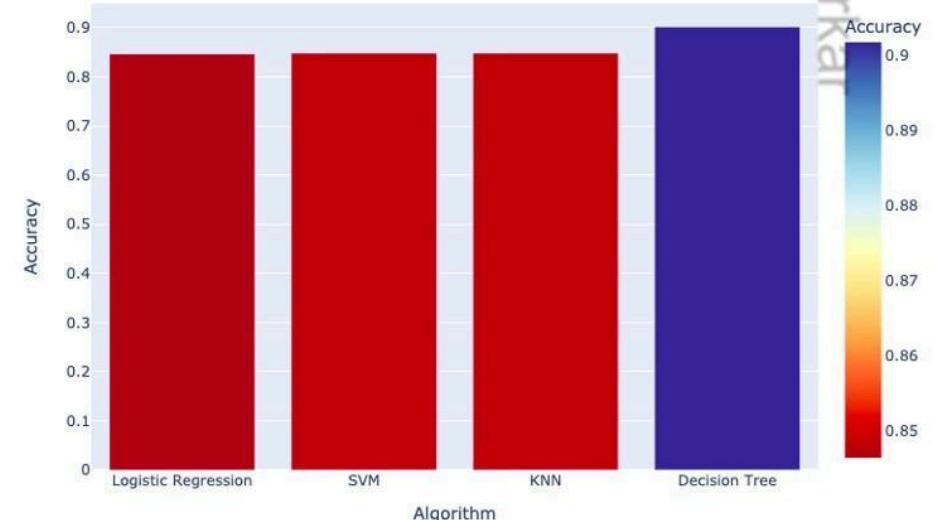


# Classification Accuracy

As you can see our accuracy is extremely close, but we do have a clear winner which performs best - "**Decision Tree**" with a score of 0.90178.

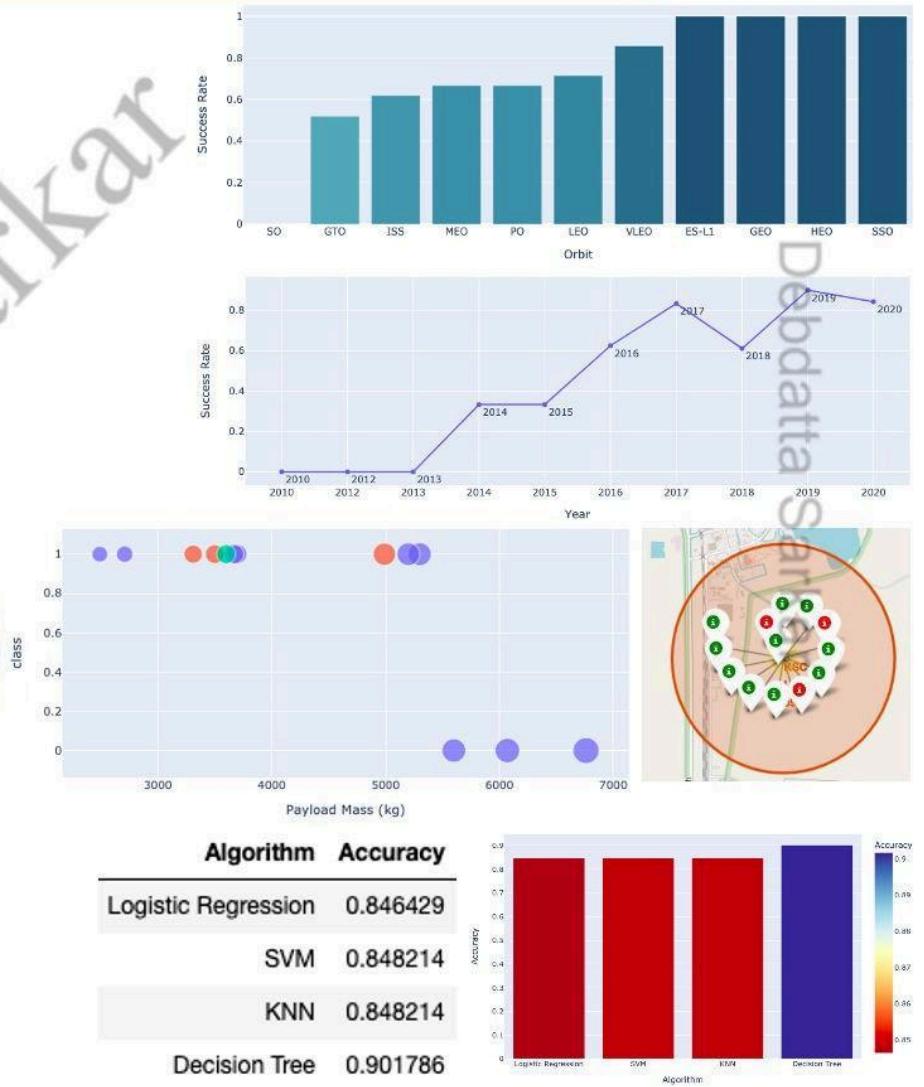
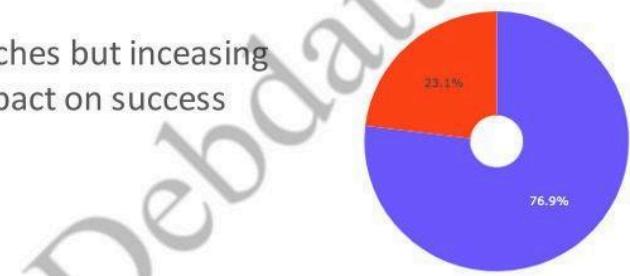
Algorithm	Accuracy	Accuracy on Test Data	Tuned Hyperparameters
Logistic Regression	0.846429	0.833334	{'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
SVM	0.848214	0.833334	{'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
KNN	0.848214	0.833334	{'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
Decision Tree	0.901786	0.833334	{'criterion': 'gini', 'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'best'}

We trained four different models which each had an 83% accuracy rate.



# Conclusion

- 1 Orbit ES-L1, GEO, HEO, SSO has highest Success rates
- 2 Success rates for SpaceX launches has been increasing relatively with time and it looks like soon they will reach the required target
- 3 KSC LC-39A had the most successful launches but increasing payload mass seems to have negative impact on success
- 4 Decision Tree Classifier Algorithm is the best for Machine Learning Model for provided dataset



# Appendix



Debdatta Sarkar

Interactive Plotly

Folium MeasureControl Plugin Tool

Folium Custom Title Layers with Labels

IBM Cognos Visualization Tool

Basic Decision Tree Construction

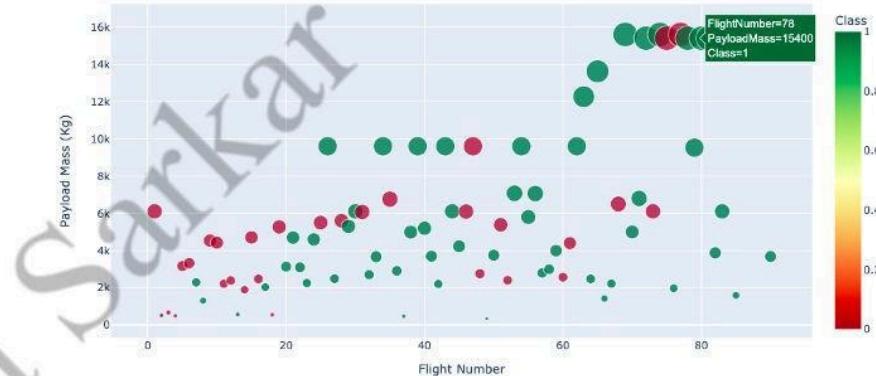
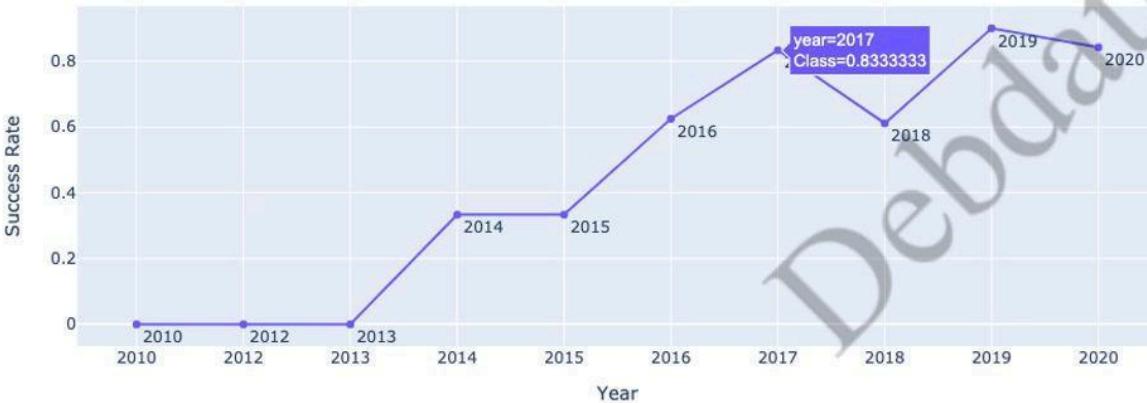
Debdatta Sarkar

# Interactive Plotly

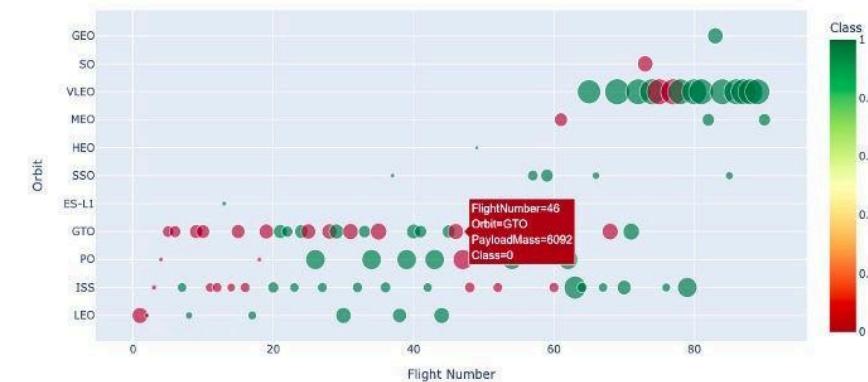
Used plotly instead of seaborn. They are more interactive and easily customizable as well.



Debdatta Sarkar

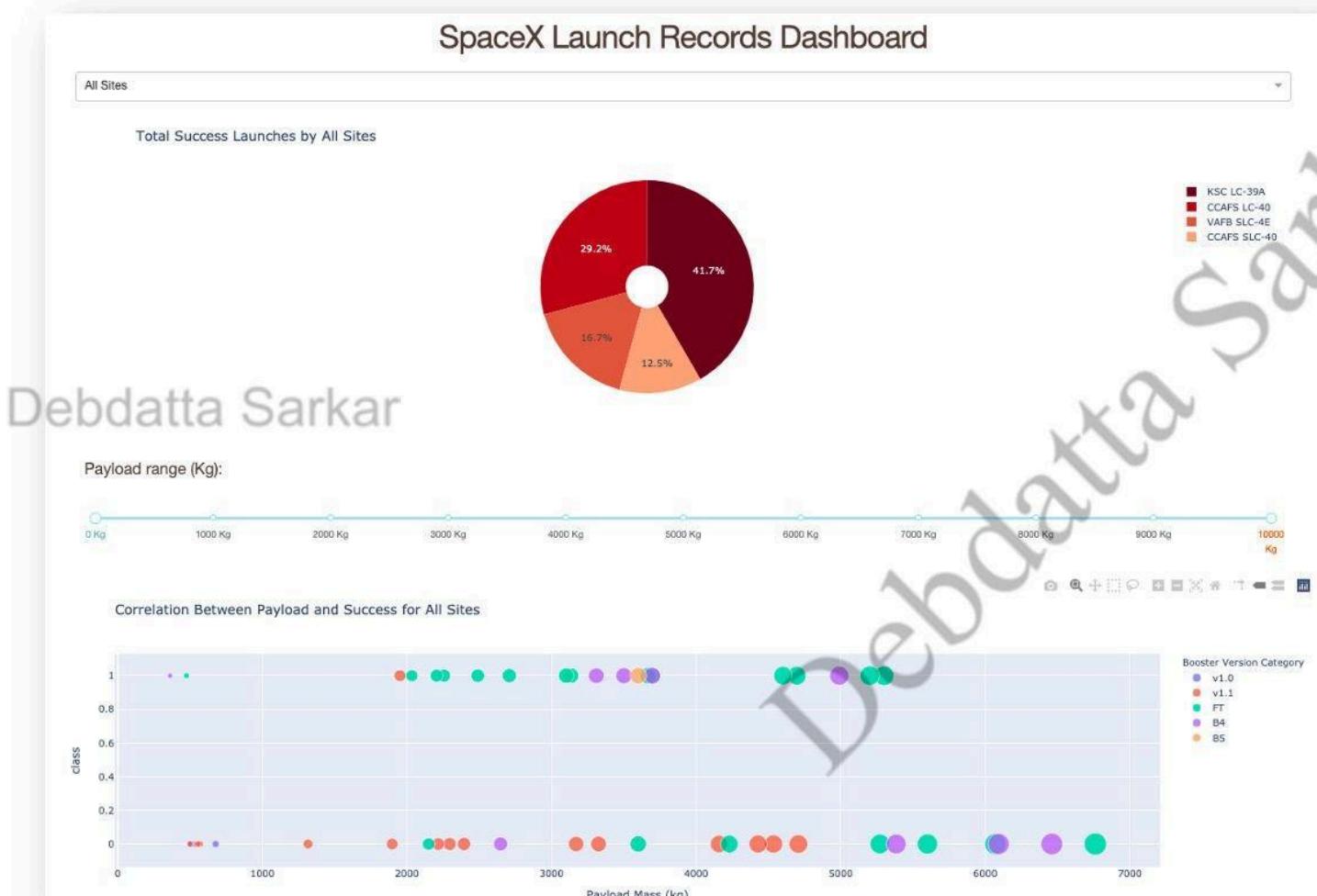


Debdatta Sarkar



Code

# "Python Anywhere" Live Site for Plotly Dashboard



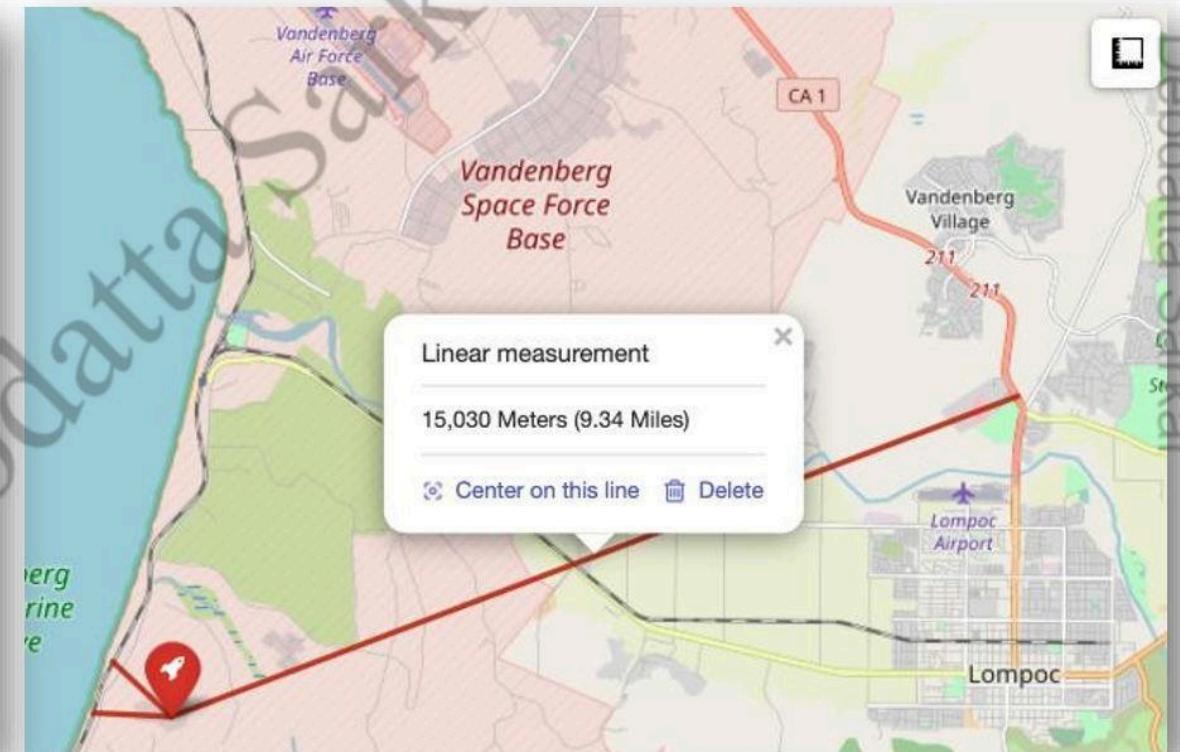
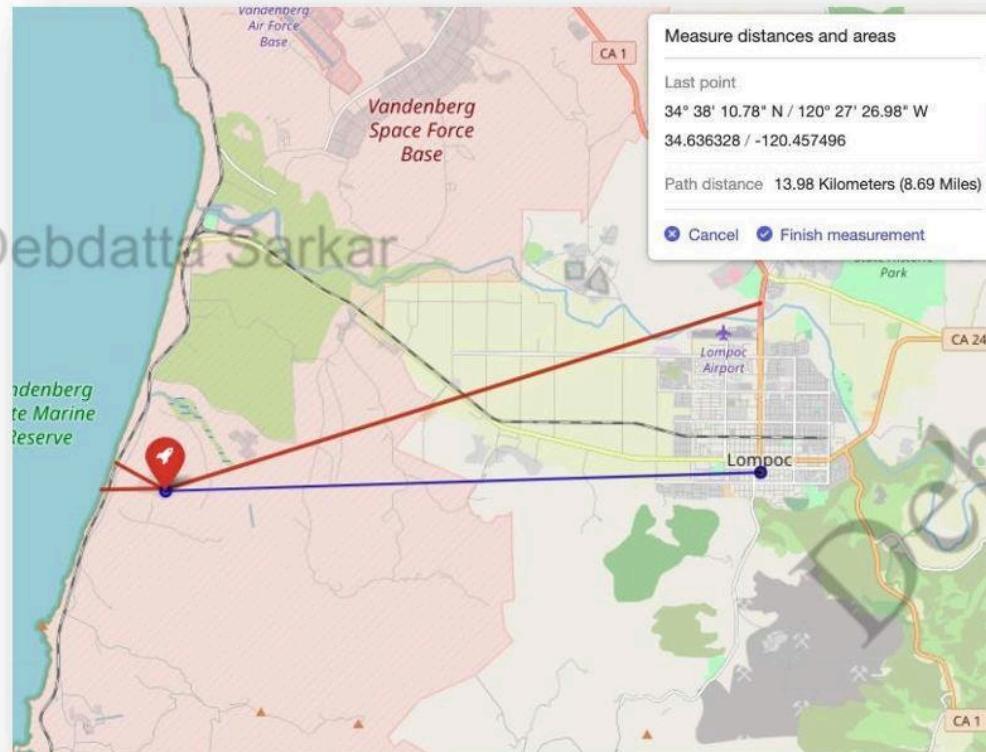
Used "Python Anywhere" to host a live website. The live site dashboard is built with Flask and Dash.

[Live Site](#) [Code](#)

# Folium MeasureControl Plugin Tool

With Measure Control Plugin Tool, we don't need to write manual distance calculation code and it's very easy to use.

```
from folium.plugins import MeasureControl
site_map.add_child(MeasureControl(primary_length_unit='kilometers',active_color='#0900ba', completed_color='#ba2f00'))
site_map
```

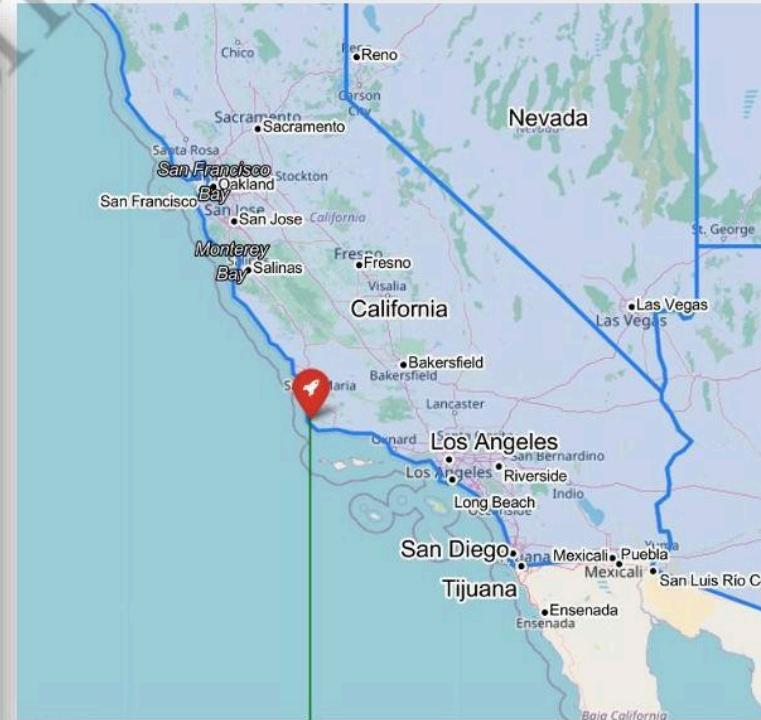


[Code](#)

# Folium Custom Title Layers with Labels

Created Custom Title Layer to understand the locations of launch site in a better way.

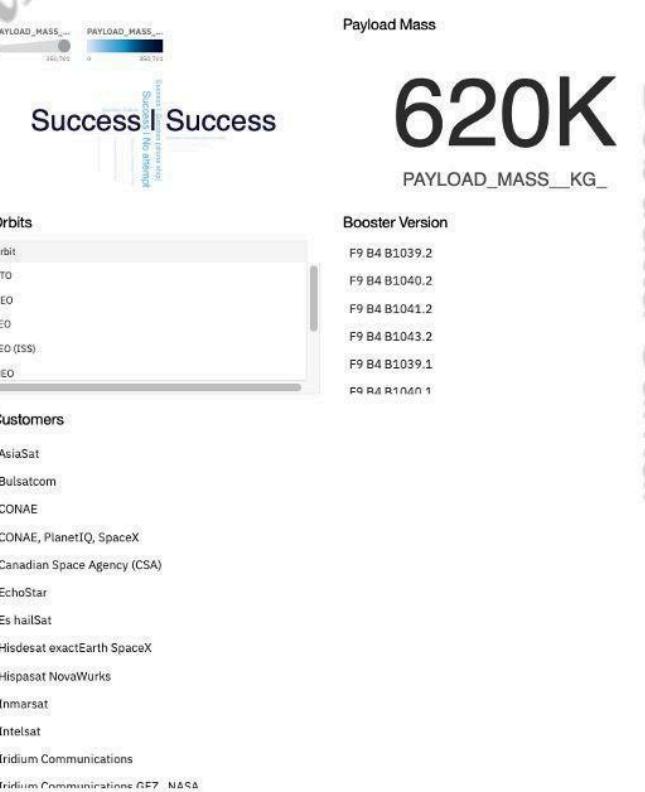
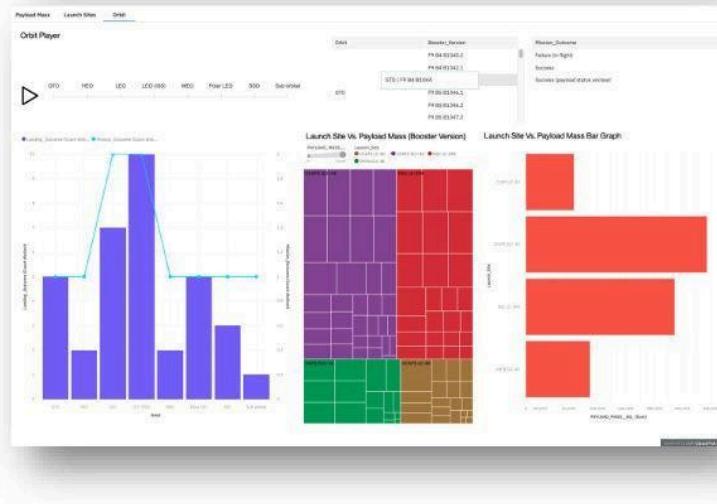
```
folium.GeoJson(geo_json_data).add_to(site_map)
folium.map.CustomPane("labels").add_to(site_map)
folium.TileLayer("stamentonerlabels", pane="labels").add_to(site_map)
site_map
```



[Code](#)

# IBM Cognos Visualization Tool

IBM Cognos Analytics provides analytic insights that help you to detect and validate important relationships and meaningful differences based on the data that is presented by the visualization.

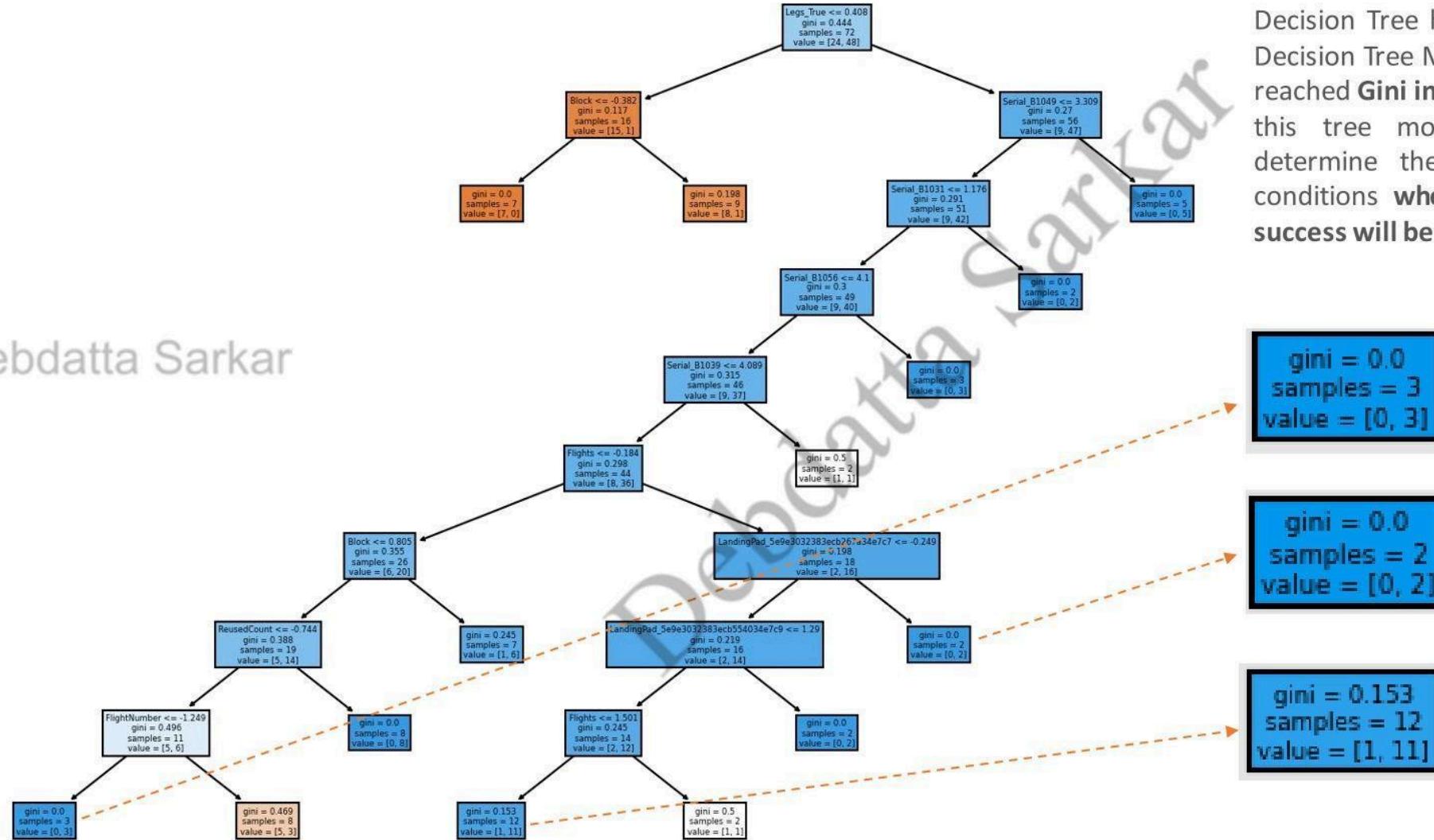


[Dashboard](#)

Powered by IBM Cloud Pak for Data

# Basic Decision Tree Construction

Debdatta Sarkar



Decision Tree has been constructed, with Decision Tree Model. We see that we have reached **Gini impurity almost near to 0** via this tree model. From this we can determine the correct combination of conditions **where the probability of the success will be highest.**

Code

Thank You!

Debdatta Sarkar

Debdatta Sarkar

Debdatta Sarkar