
IBM Developer
SKILLS NETWORK
bhaskar das
Winning Space Race
with Data Science
Bhaskar Das
20 OCT 2021

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection Via API, SQL And Web Scrapping
  - Data Wrangling And Analysis
  - Interactive Maps With Folium
  - Predictive Analysis For Each Classification Model
- Summary of all results
  - Data Analysis Along With Interactive Visualization
  - Best Model For Predictive Analysis

Introduction

- Project background and context

  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 due to the fact that SpaceX can reuse the first stage. Therefore if we can predict the success ratio, This information can be used if any other company wants to bid for SpaceX rocket lunch.

- Problems you want to find answers
  - Factor for which rocket will land successfully?
  - The effect of each relationship of rocket variables on outcomes.
  - Co-relation that will aid for a successful landing.

Section 1

Methodology

Methodology

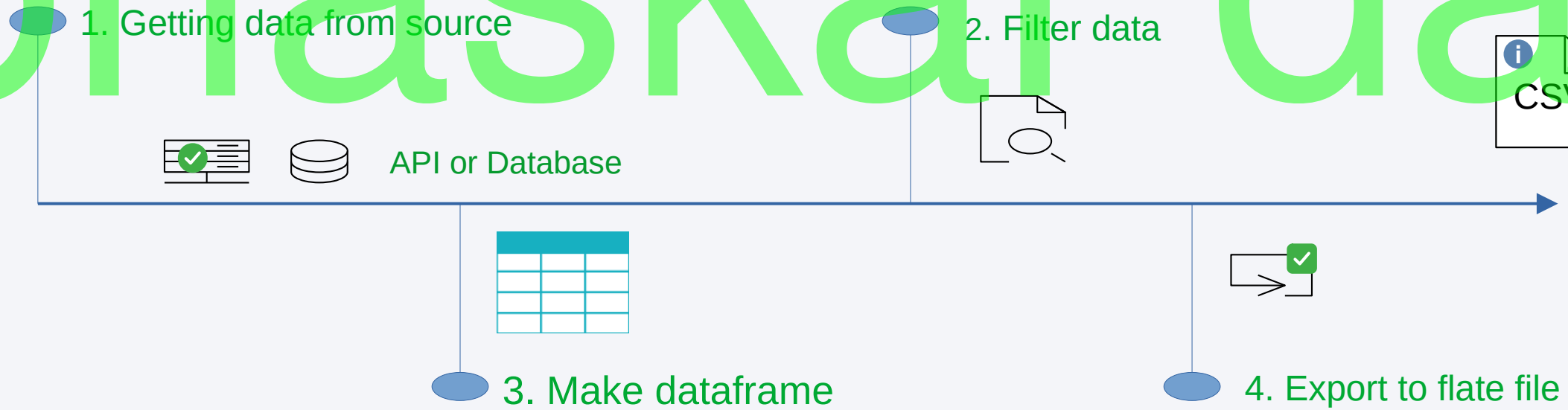
- Data collection methodology:
  - Via spaceX API
  - Web scrapping from wikipedia
- Perform data wrangling

  Transforming data for machine learning with one hot encoading and droping irrevalent columns.
- Perform Explanatory Data Analysis(EDA) using Visualization and SQL
  - Scatter and Bar graphs are used to show pattern between data.
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis for classification models

Data Collection

github link

Data Collection is the process of gathering and measuring information for our analysis, Which then enables us to answer relevant questions and evaluate answers.



Data Collection – 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.to_csv('csvs/dataset_part_1.csv', index=False)
```

Getting Response from API → Converting response to Json file → Cleaning Data with custom filters → Create Dataframe from dictionary → Filter data and export to flate file

```
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
0	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	False	False
1	2	2007-03-21	Falcon 1	NaN	LEO	Kwajalein Atoll	None None	1	False	False	False
2	4	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	False	False
3	5	2009-07-13	Falcon 1	200.0	LEO	Kwajalein Atoll	None None	1	False	False	False
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False

github link

Data Collection - Scraping

github link

- Getting response from HTML
- Creating BeautifulSoup Object
- Finding Tables
- Getting column names
- Dictionary creation and appending data to keys
- Converting dictionary to dataframe
- Export to CSV

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

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(data, 'html5lib')
```

```
html_tables=soup.find_all("table")
html_tables
```

```
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)
```

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']
```

	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\n	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\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success\n	F9 v1.0B0006.1	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success\n	F9 v1.0B0007.1	No attempt\n	1 March 2013	15:10

Data Wrangling: Basic Steps

Data wrangling is the process of cleaning and unifying messy data for clear evaluation and easy analysis. Here I mainly convert outcomes to 1 and 0 for successful and failed landing respectively.

Data Wrangling

1. Calculate the number of lunches each site

```
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
Name: Orbit, dtype: int64
```

2. Calculate the number of occurrence of each orbit

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

CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
```

3. 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
```

4. Create a landing outcome label

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

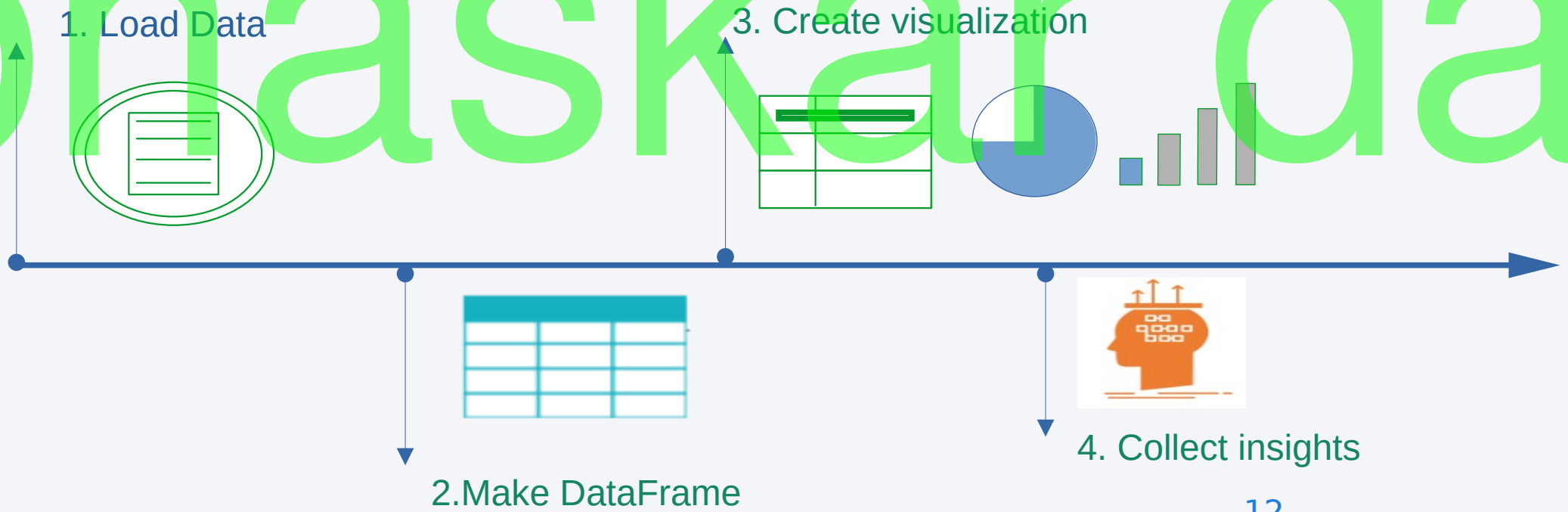
5. Export to CSV

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

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857	0
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857	0
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857	0
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632093	0
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857	0

EDA Basic Steps

Explanatory data analysis is an approach of analyzing datasets to summarize their main characteristic, using statistical graphics and other visualization tools.

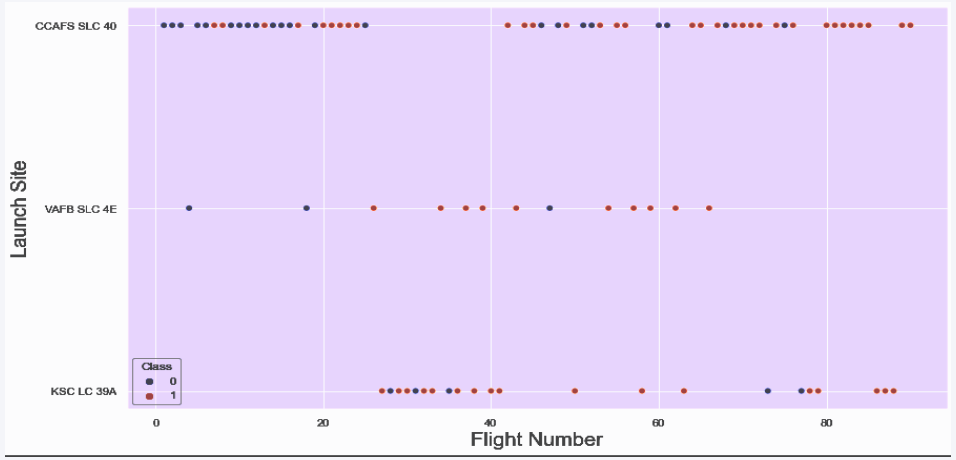


EDA with Data Visualization

Scatter Graph :

- Payload and Flight number.
- Flight number and Lunch site.
- Payload and Lunch site.
- Flight number and Orbit Type.
- Payload and Orbit Type.

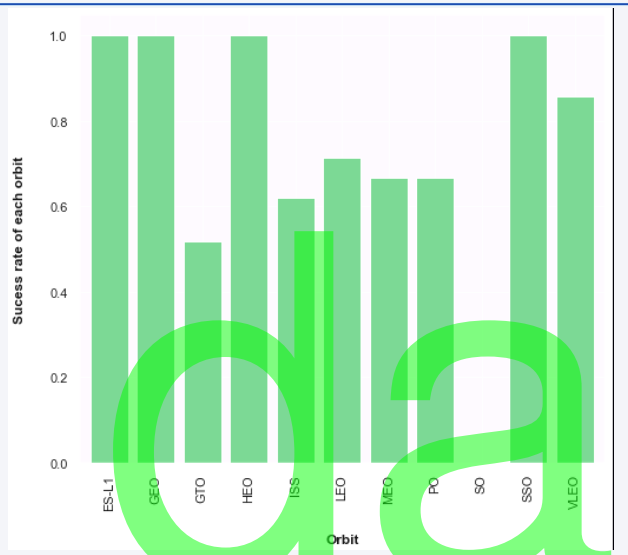
Scatter plot shows dependancy of attribute on each other,Once pattern is determined from graph it's easy to predict which factor will lead to maximum possibility of success.



Bar Graph :

- Succes vs Orbit Type

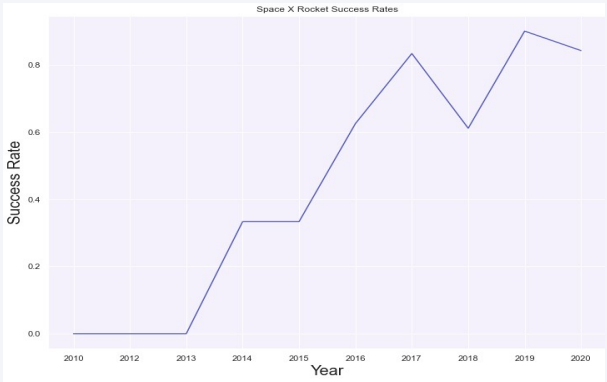
Bar graph are easiest to interpret a relation between attributes,With this bar graph we can conlude which orbit have higher chance of success.



Line Graph :

- Lunch Success Yearly Trend

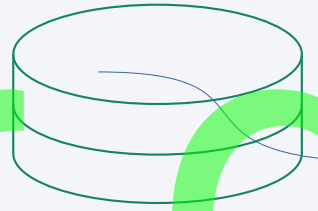
Line graph are useful in show trend clearly and aid for precise future prediction.



EDA with SQL

SQL is an indispensable tool for data scientiest as most of the real world data is stored in database. It's not only standard language for Realational database operation, but also an incredibly powerful tool for analysing data and drawing useful insight from it. Here we use IBM's Db2 for cloud, Which is 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>
```



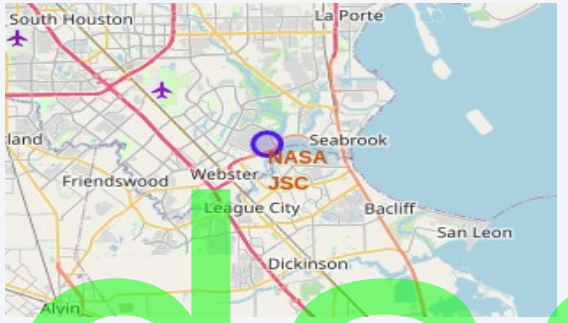
- IBM's Db2

I performed following quaries to gather information from the dataset

- Display the name of the unique lunch sites in the mission.
- Display 5 sites that begins with 'CCA'.
- Displaying the total payload mass carried by CRS.
- Display avarage payload carried by booster version F9V1.1.
- Listing the date where the succesful landing outcome was achived.
- Listing the names of the booster which was succesful in ground pad and have payload mass greater than 4000 but less than 6000.
- List the total number of successful and failure mission outcomes.
- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for the 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

Folium makes it easy to visualize data that's been manipulated in python on a interactive leaflet map.We use lantitude and longitude coordinates for each launch site and added Circle Marker around each launch Site. It is easy to visualize the number of success and failure for 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 Greean and Red marker on the map.



Map Object	Code	Result
Map marker	Folium.Marker( )	Map object to make a mark on map
Iron Marker	Folium.Icon( )	Create icon on a map
Circle Marker	Folium.Circle( )	Create a circle where marker is being placed
PolyLine	Folium.Poyline( )	Create a line between points
Marker Cluster Object	Marker.Cluster( )	Simplify a map with more marker of same cordinates
Ant Path	Folium.Plugins.AntPath( )	Craete a animated line between points

Build a Dashboard with Plotly Dash

Pie chart showing the total success of all sites oe by certain launch site.

- *Percentage of success in relation to launch site*

Scatter graph showing the correlationbetween payload and success for all sites or by certain launch site.

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

Map Object	Code	Result
Dash & it's components	Import dash From dash import html From dash import dcc	Plotyy stewards python is a data vizualisation library.
Pandas	Import pandas as pd	Fetching values from csv and creating a dataframe.
Plotly	Import plotly.express as px	Plot graph and interactive plotly library.
Dropdown	Dcc.dropdown( )	Create dropdown for launch site.
RangeSlider	Dcc.rangeslide( )	Create a rangeslider for payload mass range selection.
Pie chart and scatter chart	px.pie( ) px.scatter( )	Creating a pie chart and scatter chart For perchantage and correlation display.

Predictive Analysis (Classification)

Building Model

- Load our feature data into dataframe.
- Transform into Numpy arrays.
- Standardize and transform data.
- Split data into trainning and test data.
- Check how many test sample have been created.
- List down machine learnning algorithm to use.
- Set our parameter and algorithm to GridSeachCV.
- Fit our dataset to GridSeachCV object and train model.

```
Y = data['class'].to_numpy()
Transform = preprocessing.StanderdScaler()
X_train, X_test, y_train, y_test = train_test_split(
                    X,y,test_size=0.2,random_state=4)
y_test.shape
```

```
algorithms = {'KNN':knn_cv.best_score_,'Decision Tree':tree_cv.best_score_,'
Logistic Regression':l ogreg_cv. best_score_,'SVM':svm_cv.best_score_}
best_algorithm = max(algorithms, key= lambda x: algorithms[x])
print('The method which performs best is \"',best_algorithm,'\" with a
                    score of',algorithms[best_algorithm])
```

Finding Best Classification Model

- The model with best accurancy score wins the best performing model.

Best Model

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

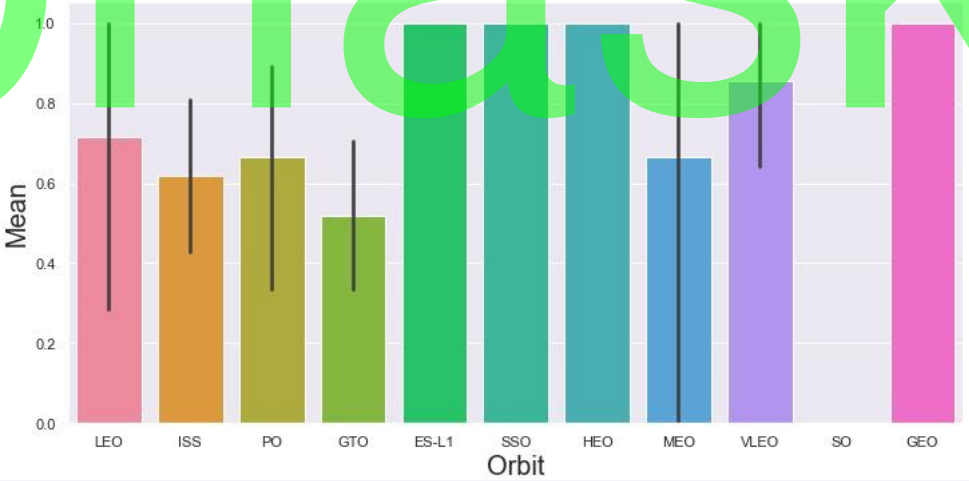
Evaluating Model

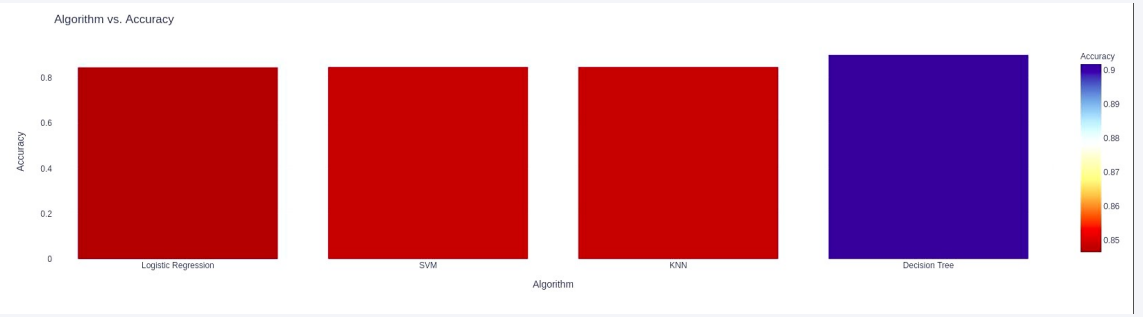
- Check for accurancy for each model
- Get best hyperparameter for each algorithm
- Plot confusion matrix

Results

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

	Algorithm	Accuracy
0	Logistic Regression	0.846429
1	SVM	0.848214
2	KNN	0.848214
3	Decision Tree	0.901786





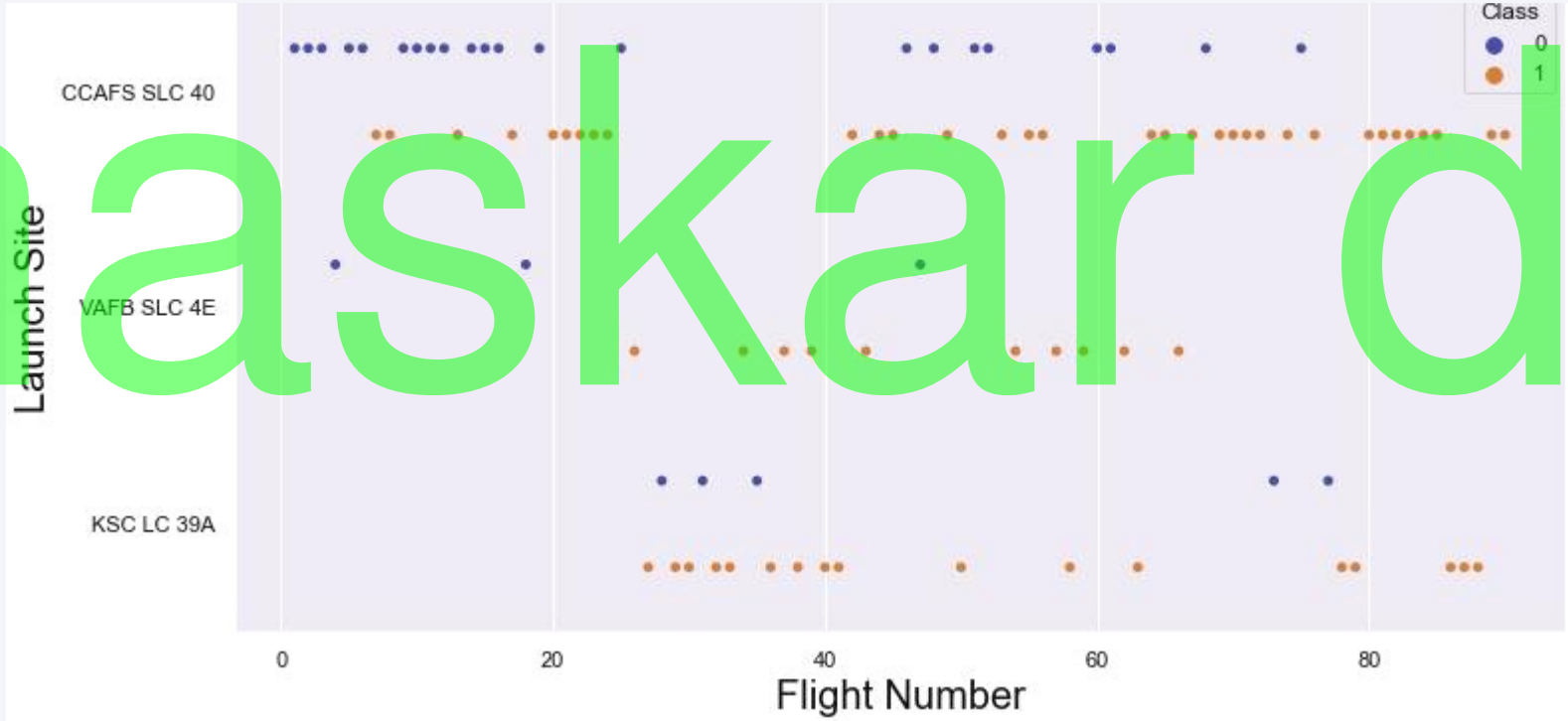
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

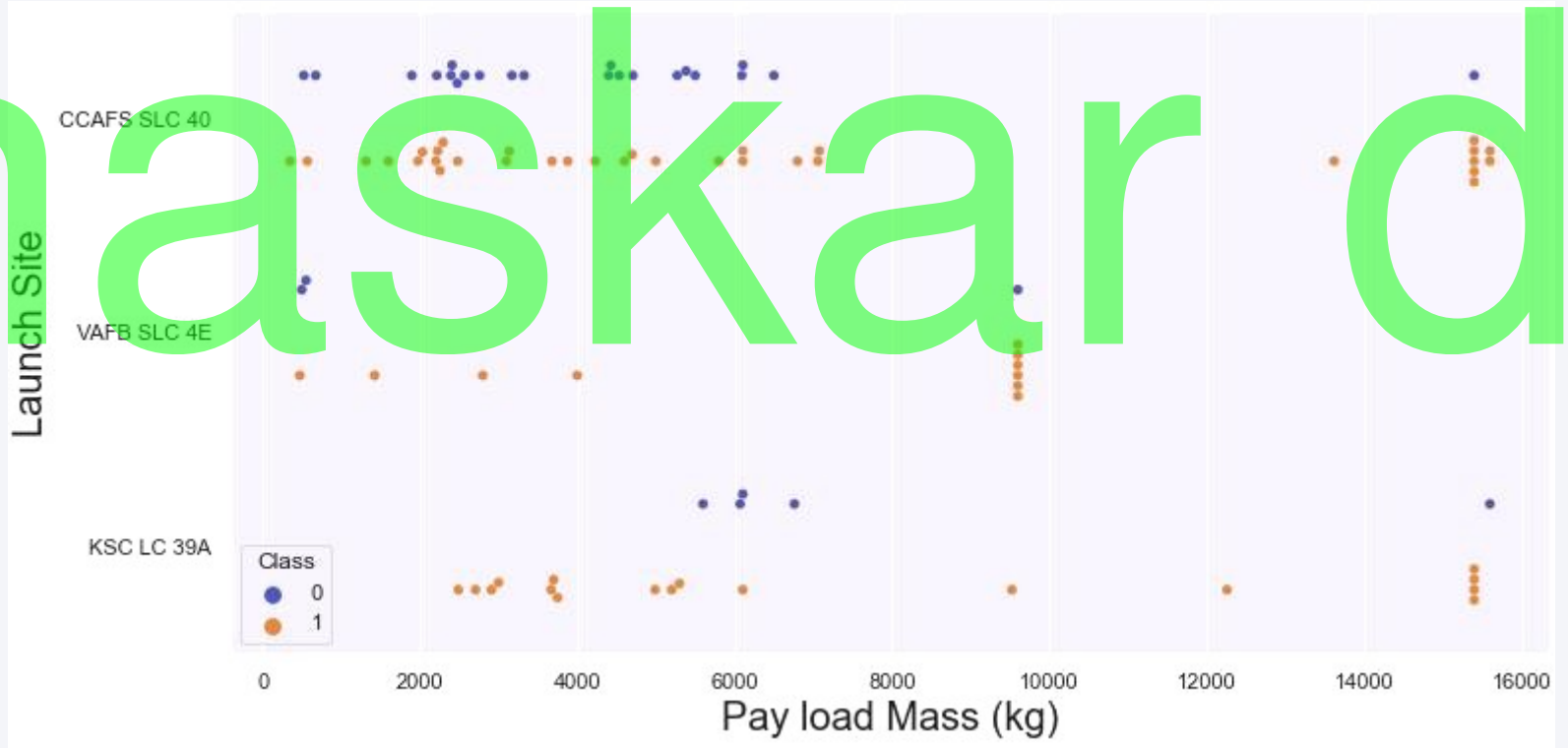
Github link

- With more Fllight Number(grater than 40) the success rate for rocket is Increasing.



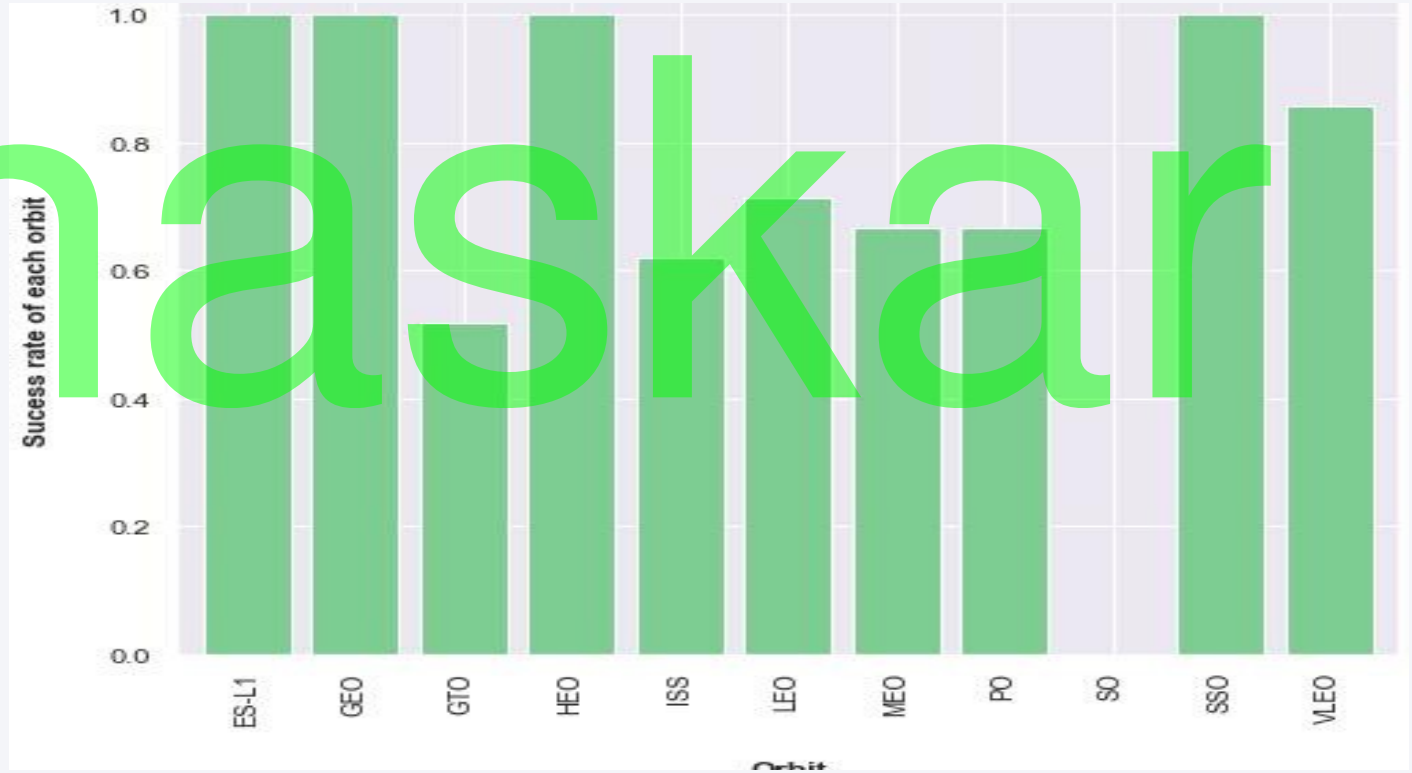
Payload vs. Launch Site

- The greater the payload mass (greater than 8000) higher the success rate for the Rocket. But theres no clear pattern to make a decision if the Launch Site is dependant on PayLoad Mass for a success launch.



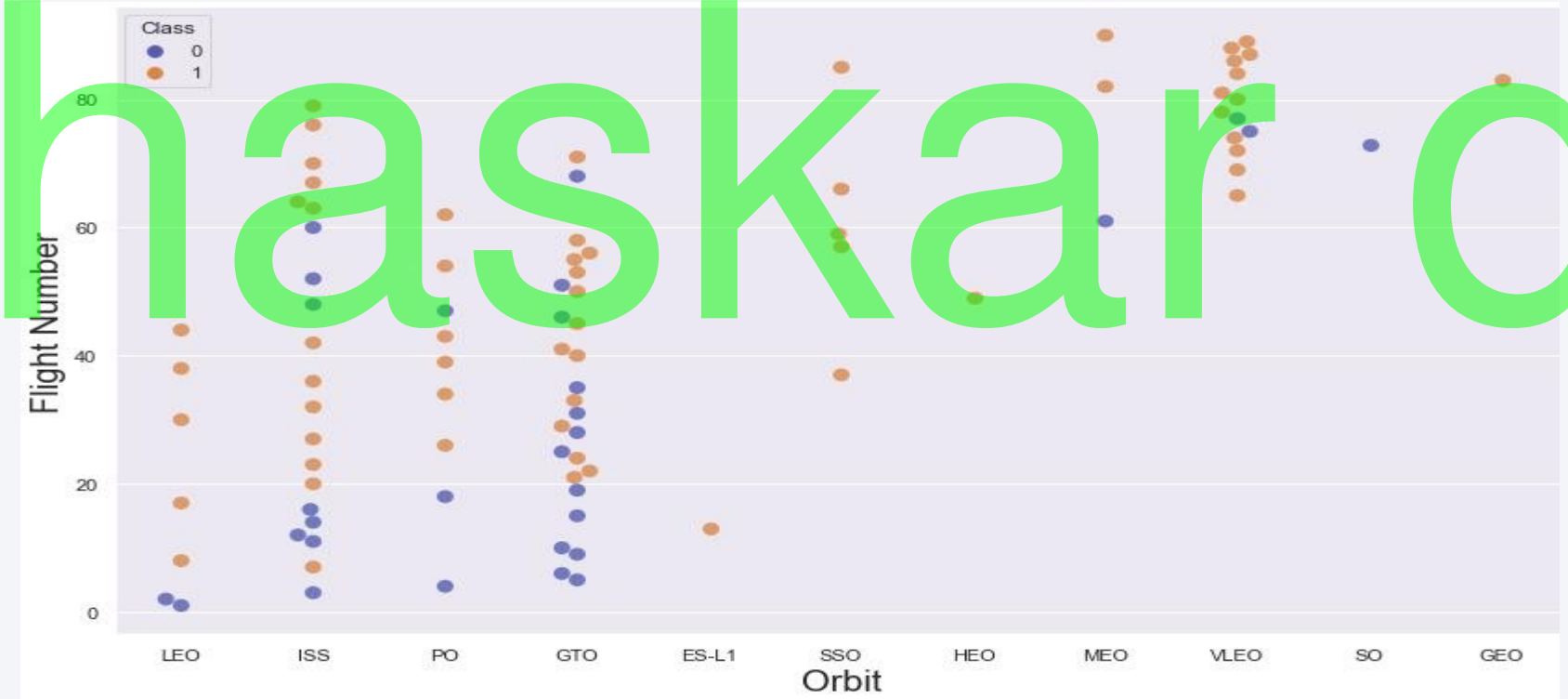
Success Rate vs. Orbit Type

- ES-L1, GEO, HEO, SSO has highest Sucess rates, SO has the lowest .



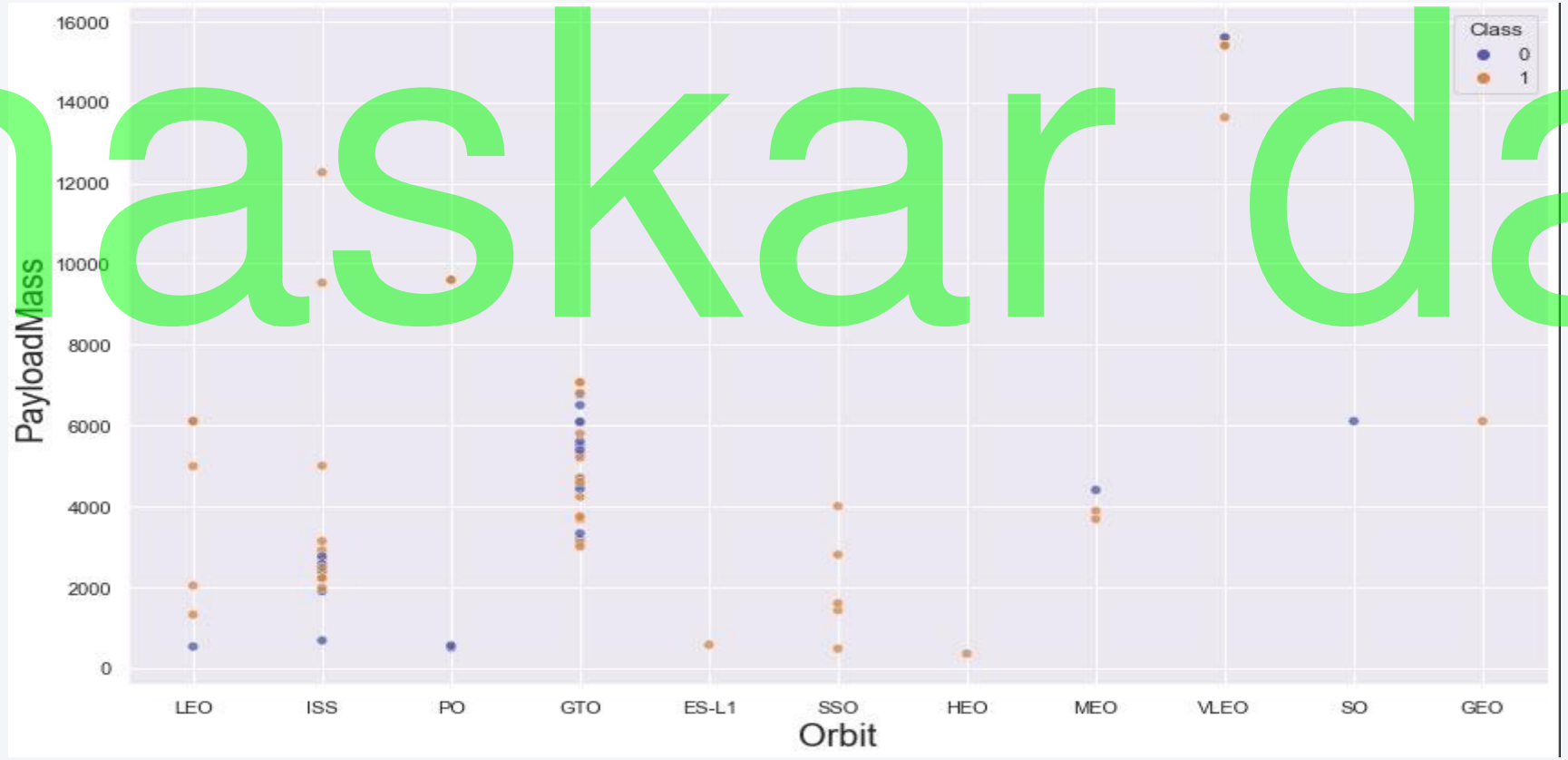
Flight Number vs. Orbit Type

- We can see for LEO orbit, success increase with the number of flight.
- There is no relationship between Flight number and GTO orbit.



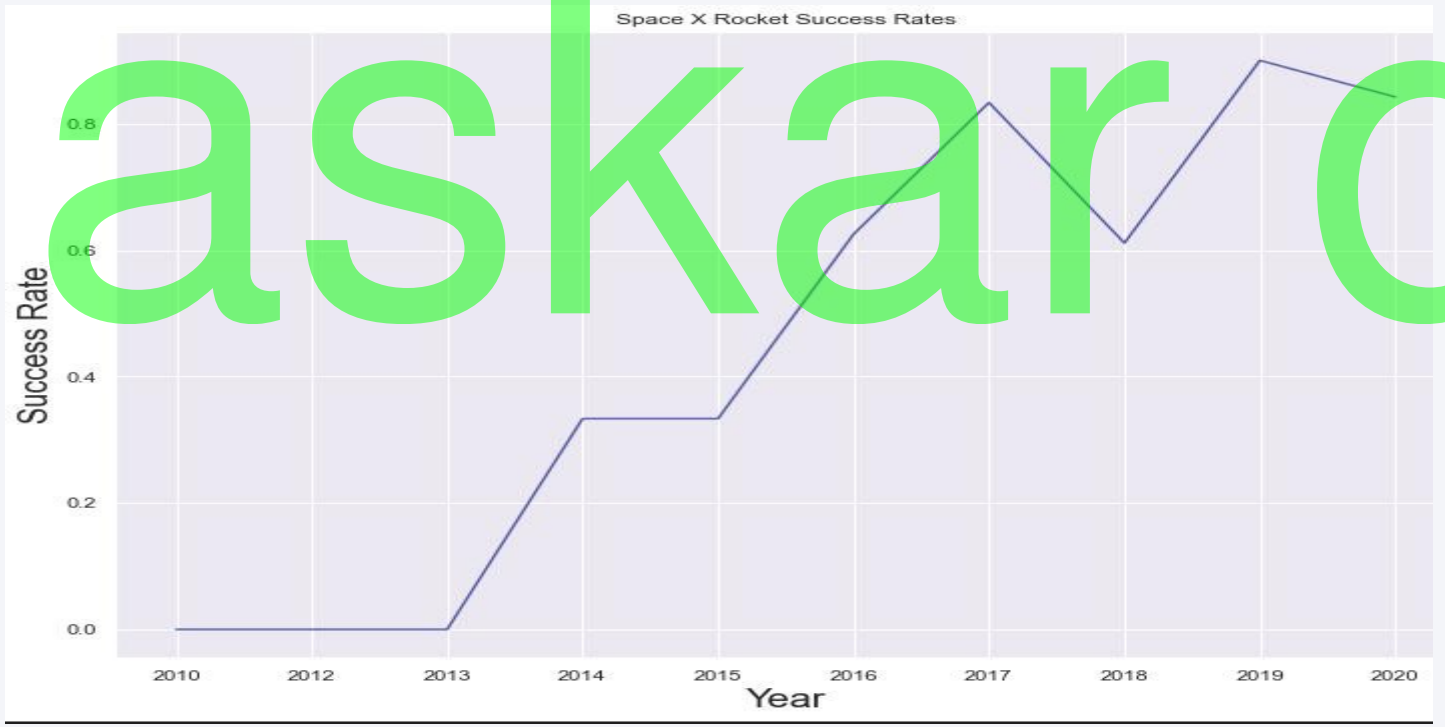
Payload vs. Orbit Type

- We can observe that payload have a negative influence on MEO,GTO & VLEO orbits.
- But it is positive on LEO, ISS orbit.



Launch Success Yearly Trend

- We can observe that success rate is inceasing since 2013 but there is slight dip in 2019.



All Launch Site Names

SQL github

SQL QUERY

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


Description

Using the word **DISTINCT** in the query we will pull unique values for **LAUNCH_SITE** column from table SPACEX.

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 record from SPACEX table and with **LIKE** keyword with wildcard – CCA**%**.The regular expression (%) suggest that LAUNCH_SITE 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 funtion SUM calculates the total number of calumn PAYLOAD_MASS_KG and WHERE clause filter the data to fetch Customer ny name NASA

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 \
WHERE BOOSTER_VERSION = 'F9 v1.1';
```


Description

Using the AVG funtion fetch the avarage in the column **PAYLOAD_MASS_KG_** the **WHERE** clause filter the dataset to only perform calculation on **Booster_version “F9 v1.1”**

First Successful Ground Landing Date

SQL QUERY

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


Description

Using the funtion **MIN** fetch minimum date in the column **DATE** and **WHERE** clause filter the data to only perform calculation on **Landing_Outcome** with values **“success(ground pad)”**

First Succesful 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

Here I am selecting only Booster_version, WHERE clause filter the dataset to **Landing_Outcome** = success(drone ship)
AND clause spacifies additional filter condition **PAYLOAD_MASS_KG_ > 4000** and **PAYLOAD_MASS_KG_<60000**

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

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;
```


Description

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

Case when **MISSION_OUTCOME LIKE** %success% then **1** else **0** end returns a boolean value which by adding we get the rquired result.

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 keyword **MAX f**etch out the maximum payload in the column **PAYLOAD_MASS_KG_** in a *sub query.*

WHERE clause filter out **BOOSTER_VERSION** which has **maximum payload** mass.

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 month(DATE) as Month, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEX WHERE year(DATE) = '2015' AND \
LANDING__OUTCOME = 'Failure (drone ship)';
```


Description

Here I listed the records which will display the month name , **Failure Landing_outcome** in drone ship, Booster_version , Launch_Site for the month of **2015**

Via year funtion extract the year and future where clause **Failure (drone ship)** fetches required values.

booster_version	launch_site
F9 v1.1 B1012	CCAFS LC-40
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 ;
```


Description

Selecting only LANDING_OUTCOME,
WHERE clause filter data with date between **‘2010-06-04’ and ‘2017-03-20’**.

Grouping by LANDING_OUTCOME and counting total by **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

Section 4

Launch Sites Proximities Analysis

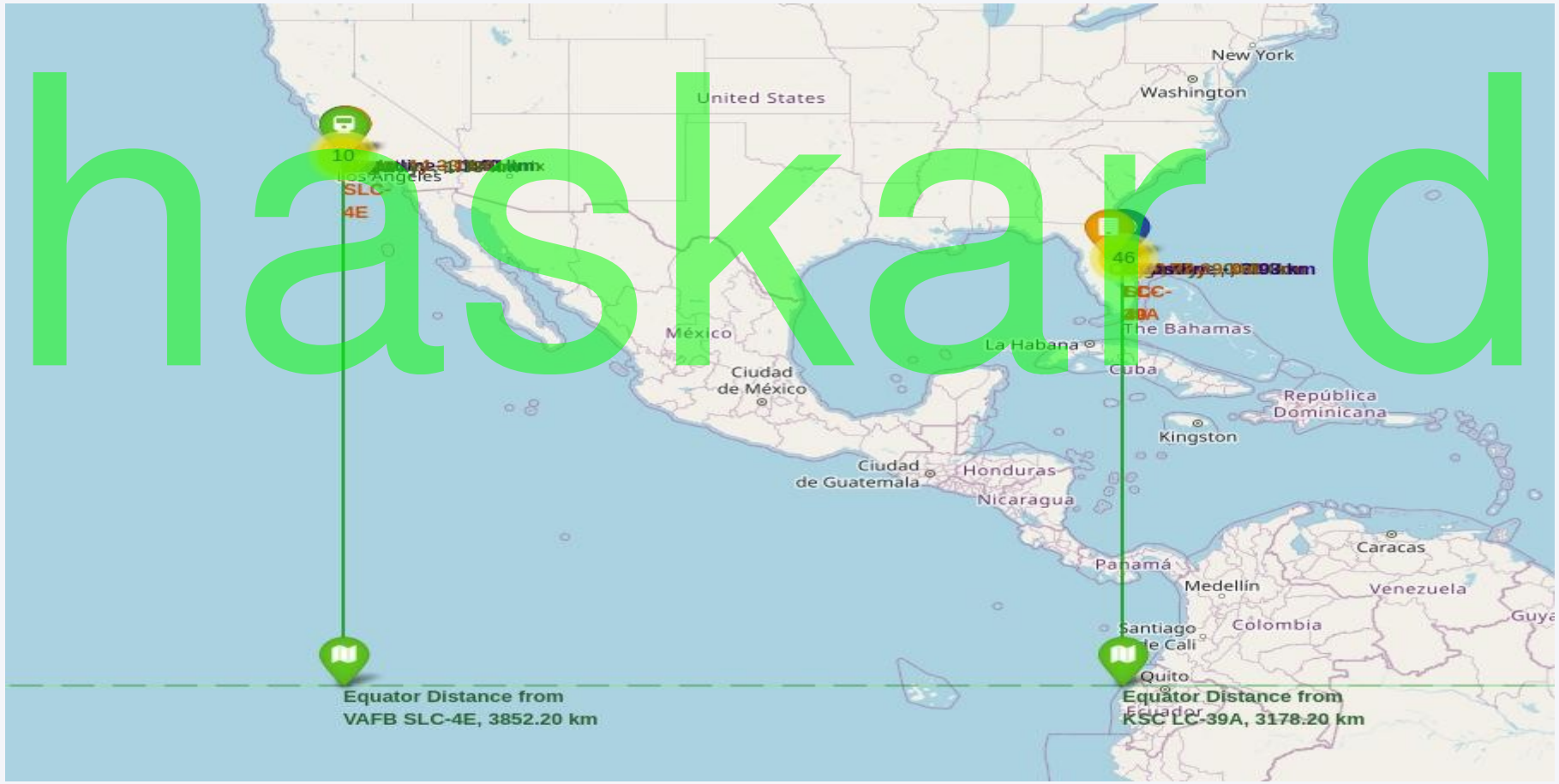
ALL Launch site on Folium map

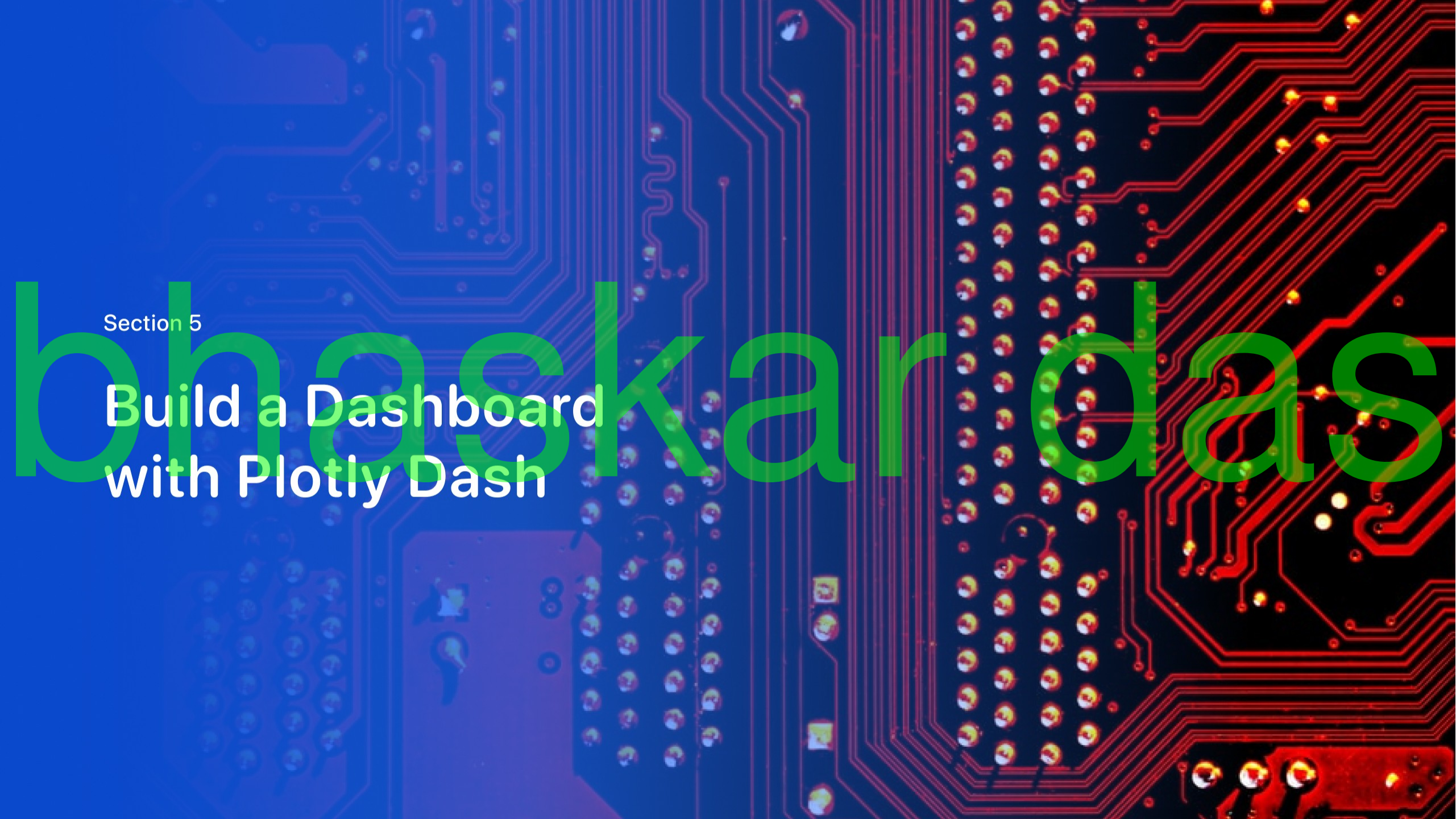
Folium github

We can see SpaceX Launch sites are situated over costel regions of USA.

Equator distance from Launch sites

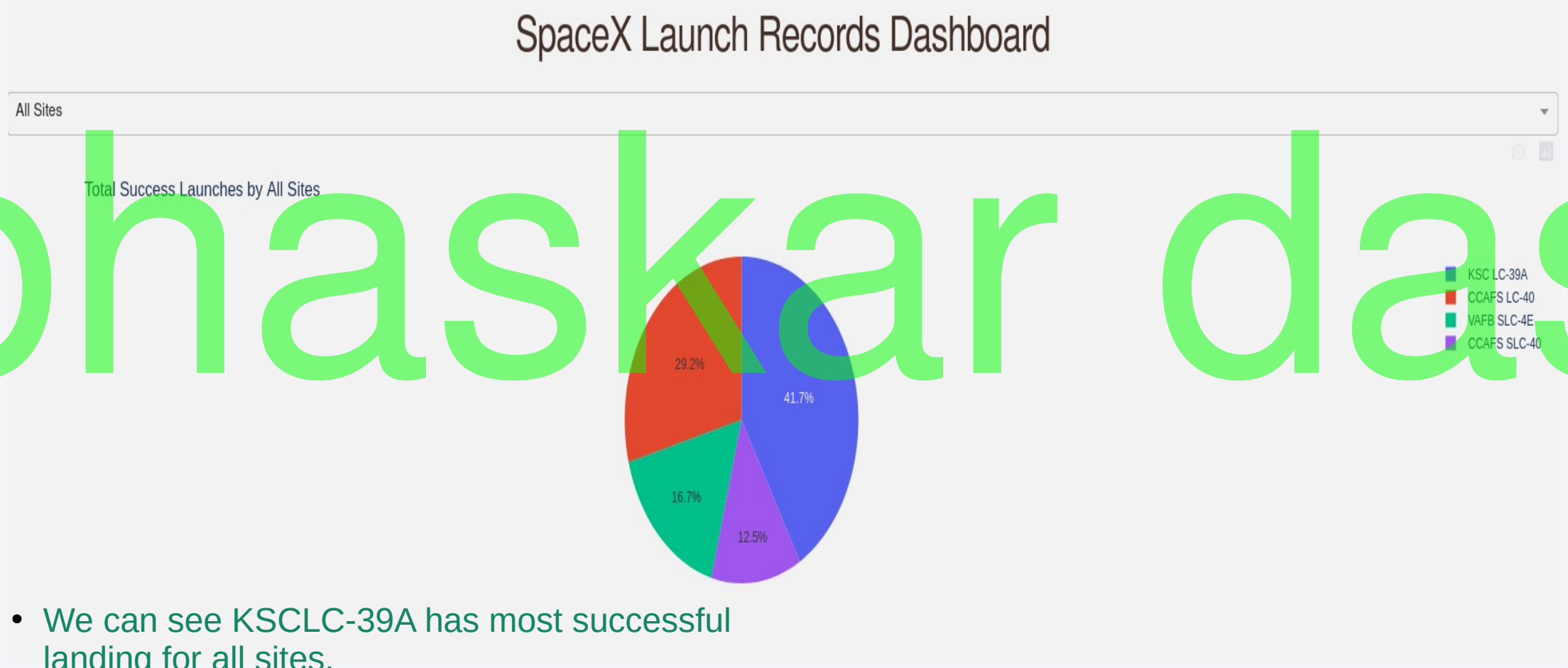
Equator distance from all sites are grater than 3000 KM for all sites.




Section 5
Build a Dashboard with Plotly Dash
bhaskar das

Launch success counts for All sites

Plotly code

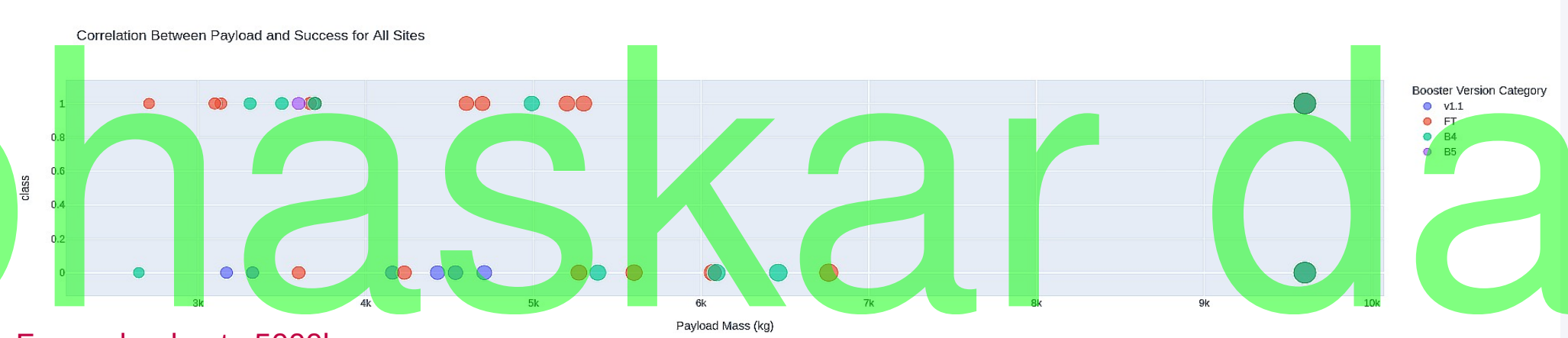


- We can see KSCLC-39A has most successful landing for all sites.

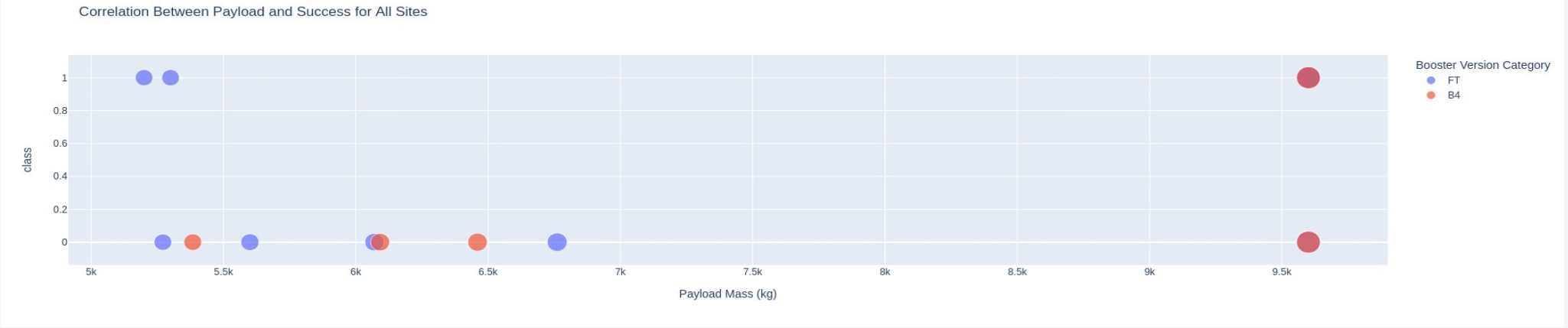
Payload VS Launch outcome plots

We can see success rate for small payload are higher as compared to higher payload

For payload upto 2500kg



For payload upto 5000kg



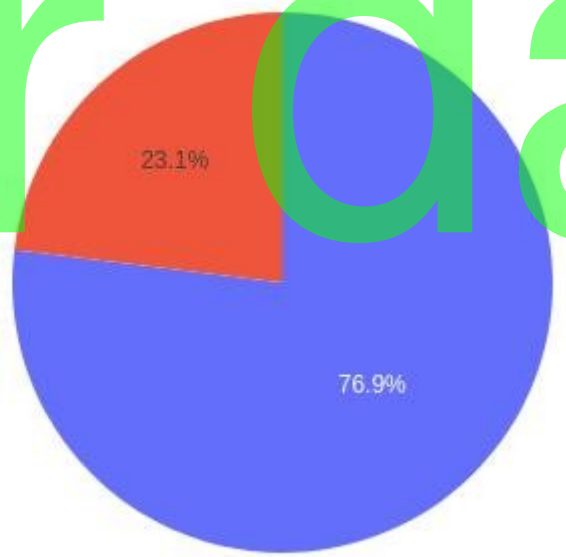
Highest successful launch site

KSC LC-39A achived a 76% success rate while getting a 23% failure rate , which is nominal as Compared to others.

Total Success Launches for SiteKSC LC-39A

After visual inspection using dashboard here are some insights :

- 2000 – 10000 kg payload has highest success rate for landing.
- 0 – 1000 kg payload has lowest landing success rate.
- FT F9 booster version have more landing success.



Section 6

Predictive Analysis (Classification)

Confusion matrix

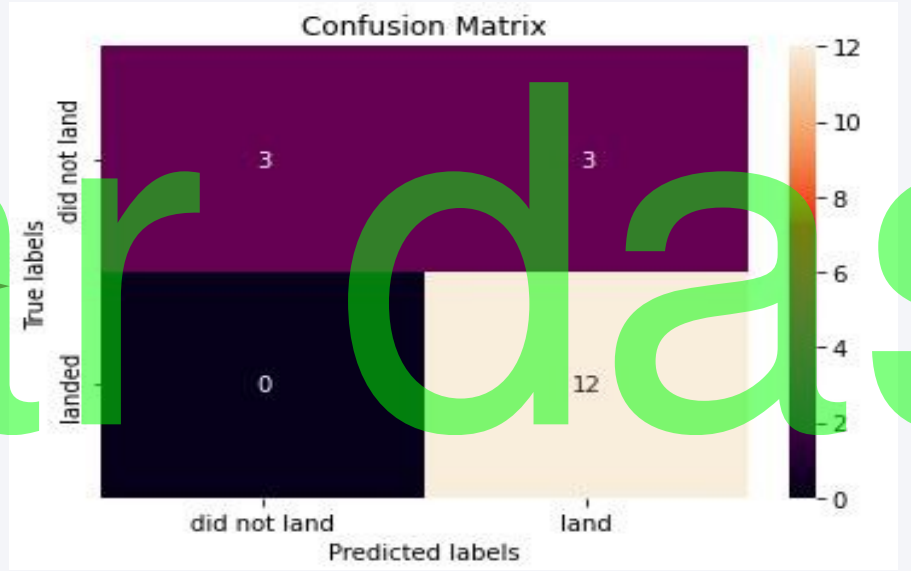
prediction github

Unfortunately all models have same confusion matrix, Here is the confision matrix for all algorithms.

Predicted value

Actual value

TN 3	FP 3	6	
FN 0	TP 12	12	
3	15	Total 18	



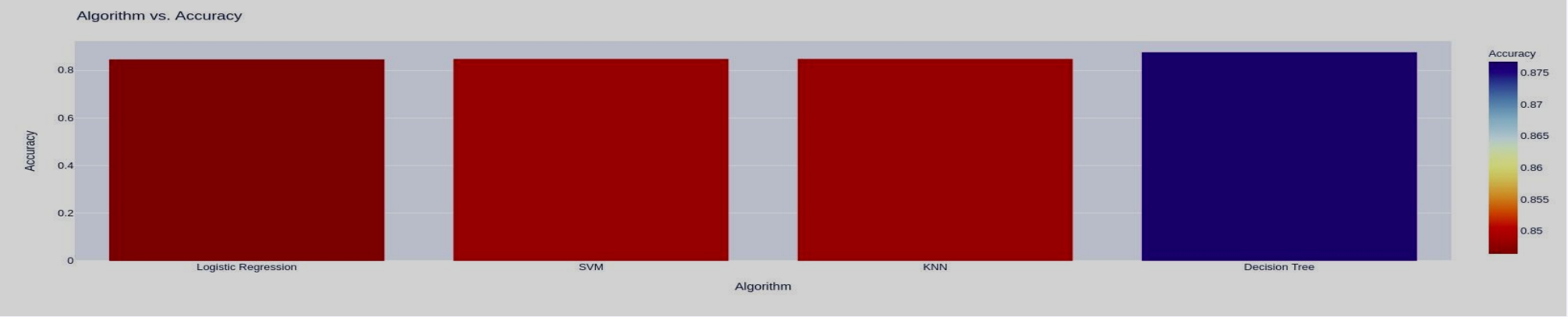
Accurancy (TP+TN)/Total = 0.833
True positive(TP/Actual Yes) = 1
True Negative(TN/Actual No) =0.5
Prevalence(Actual Yes/Total) = 0.6667

Misclassification rate(FP+FN)/Total = 0.1667
False Positive(FP/Actual No) = 0.5
Precision(TP/Predicted Yes)= 0.8

Classification Accuracy

As I can see accuracy for algorithms are close, but clearly winner is Decision tree with accuracy of 0.876786.Although I trained models with 83% accuracy rate.

Algorithm	Accurancy	Accuracy on test data	Tuned Hyperparameter
Logistic Regression	0.846429	0.833334	{‘c’:0.01,’penalty’:’l2’,’solver’:’lbfgs’
SVM	0.848214	0.833334	{‘c’:1.0,’gamma’:0.3123,’kernal’:’sigmoid’}
KNN	0.848214	0.833334	{‘algorithm’:’auto’,’n_neighbor’:10,’p’:1}
Decision Tree	0.876786	0.833334	{‘criterion’:’gini’,’max_depth’:10,’max_feature’:’sqrt’,’min_sample_leaf’:1,’min_samples_split’:2,’spliter’:’best’}



Conclusions

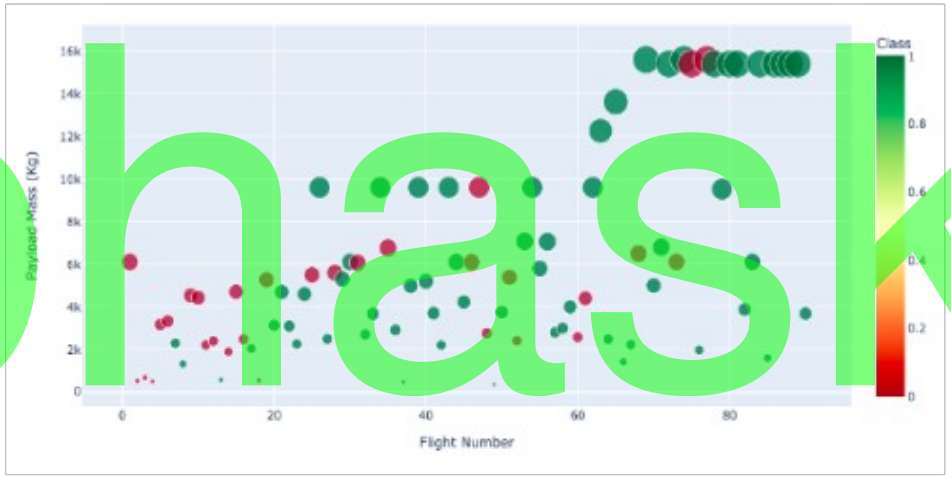
1. Orbit ES-L1,GEO,HEO,SSO has highest success rates.
2. Success rates for SPACEX launches has been increasing relatively with time and it lookes like soon it will reach to required target.
3. KSC LC-39A had the most successful launches but increasing payload mass seems to have negative impact on success.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

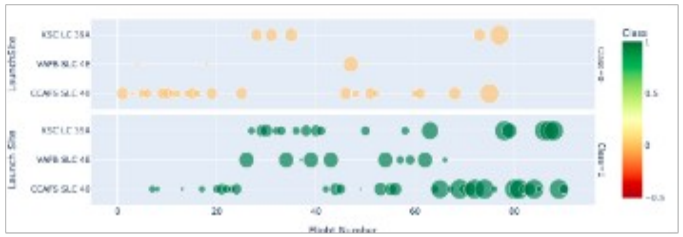
Appendix

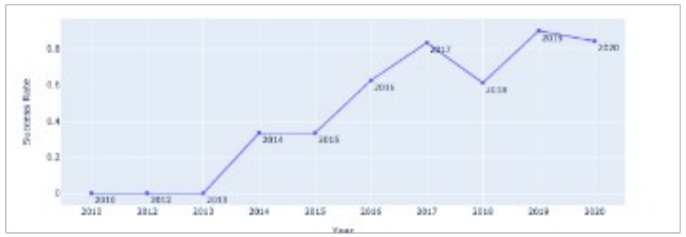
- Interactive Plotly
- Folium Measure control tool
- IBM Cognos visualisation
- Decision tree construction

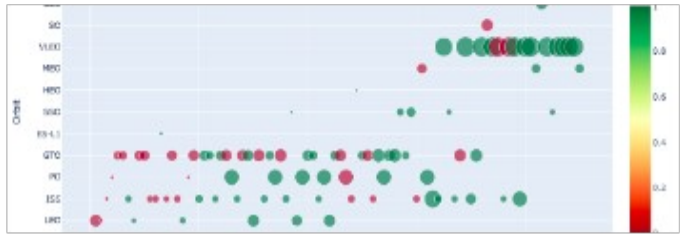
Interactive Plotly

Seaborn can also be used alternatively as they are more effective and customizable.



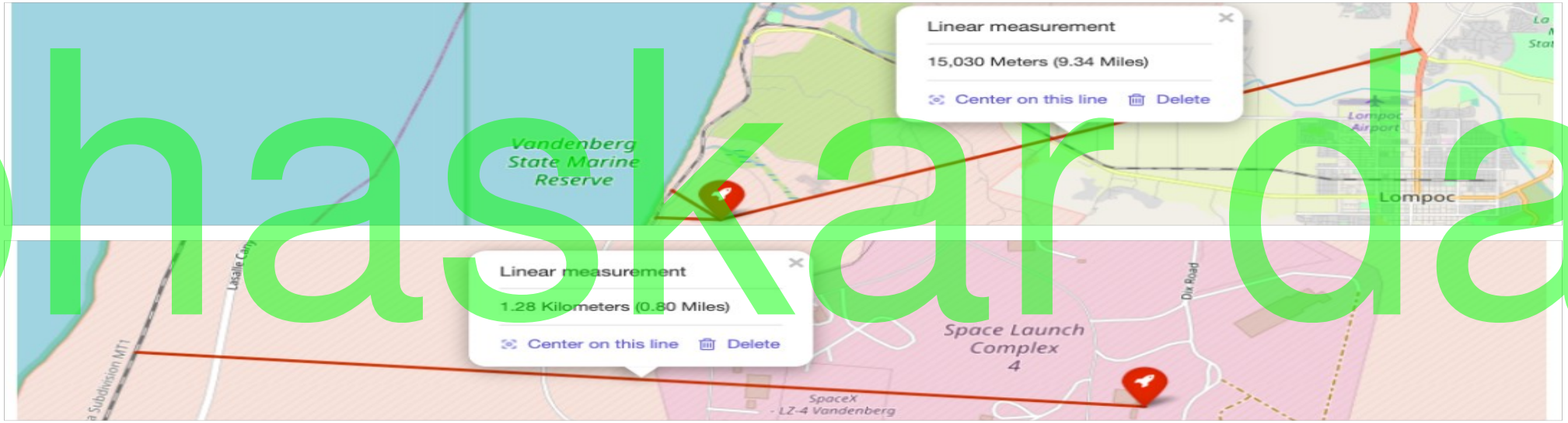


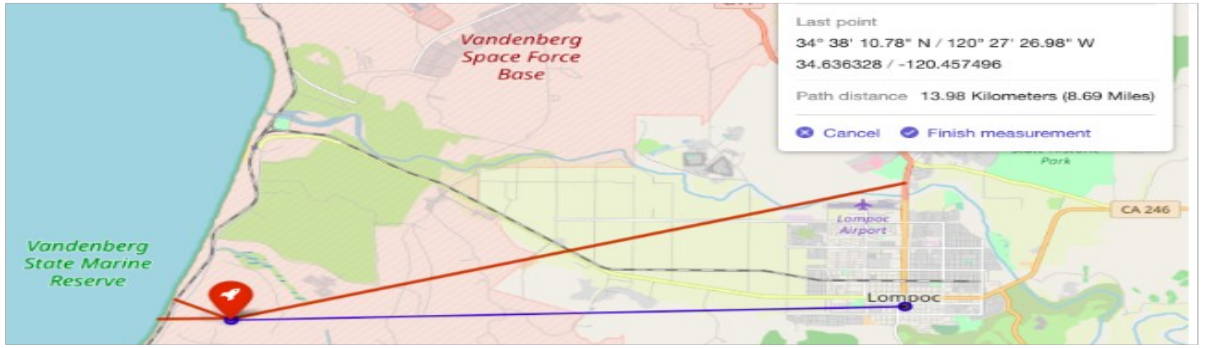


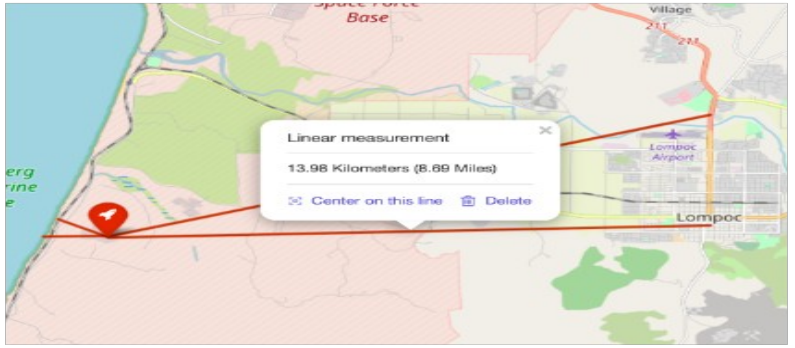


Folium Measure Control tool

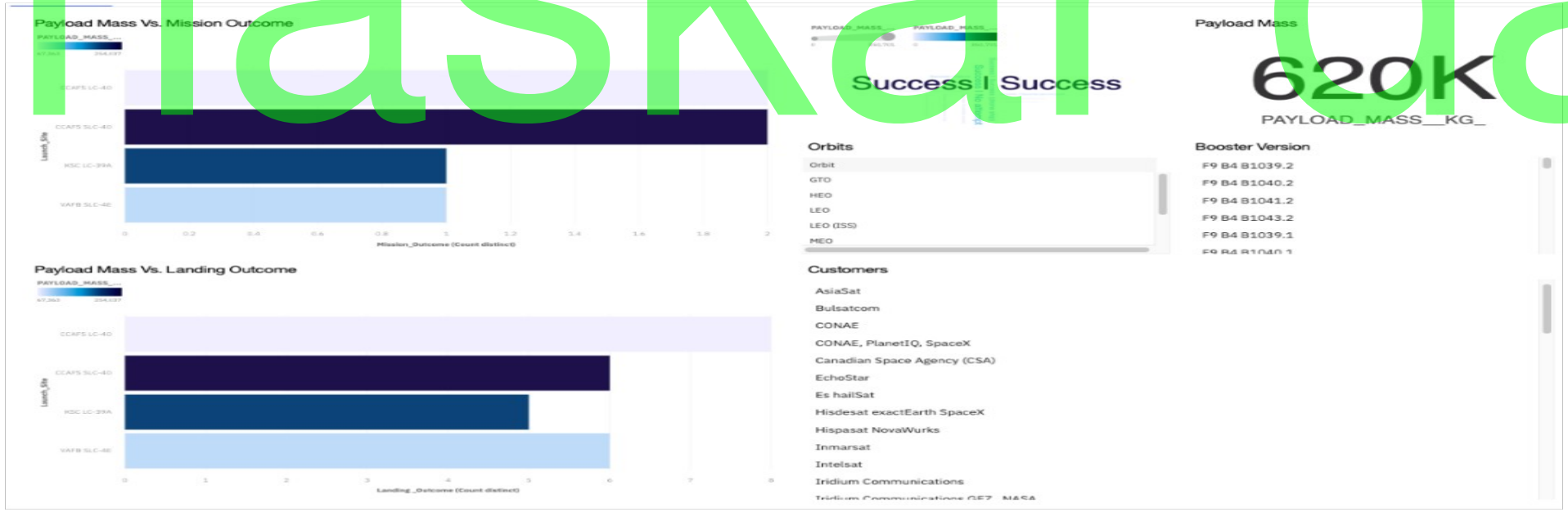
```
from folium.plugins import MeasureControl
Site.add.add_child(MeasureControl(primery_lenght_unit='kilometer',active_color='oo9ba'
site_map
```







IBM Cognos visualization tool

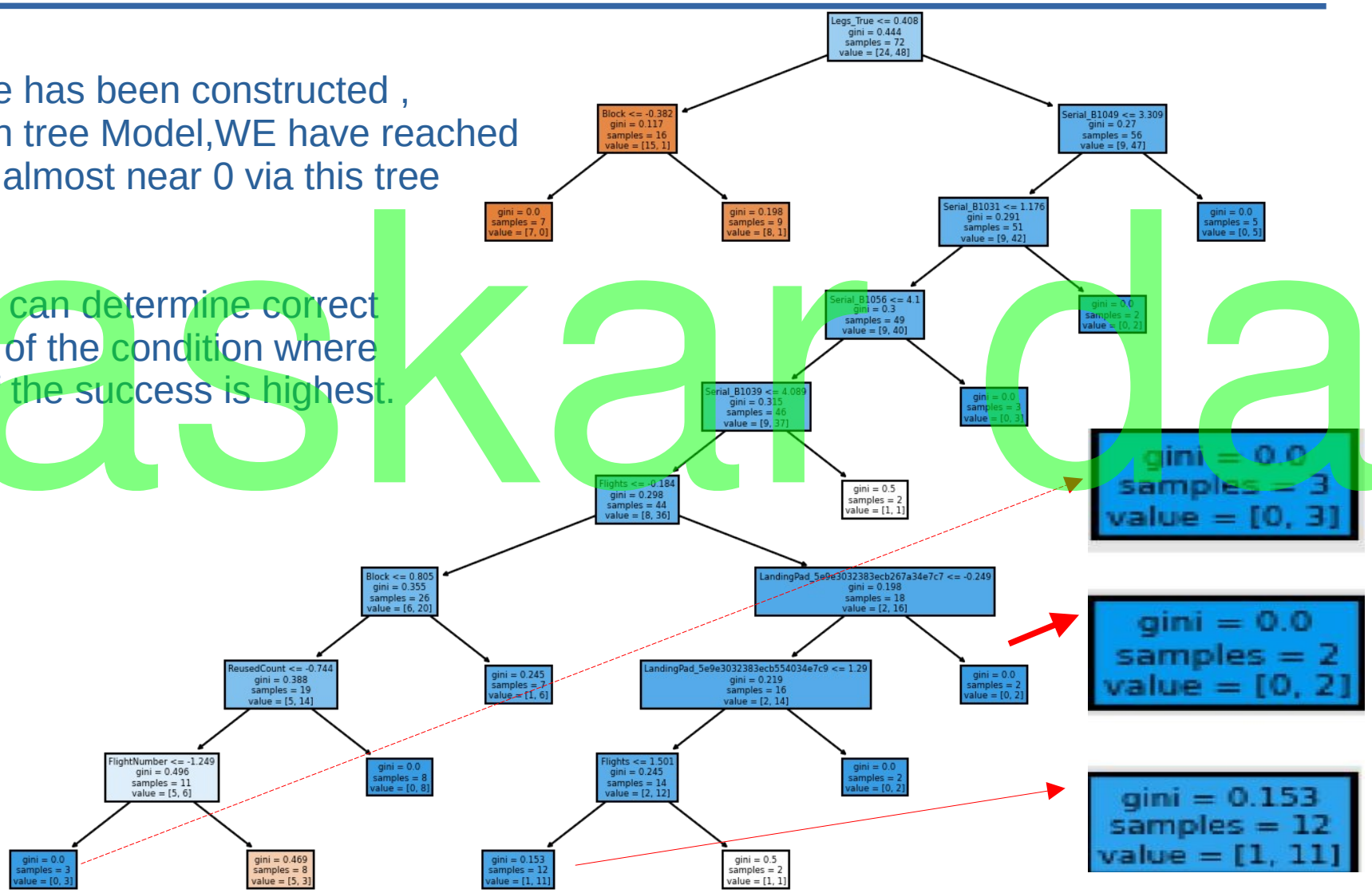


Decision Tree construction

Tree link

Decision Tree has been constructed ,
With Decision tree Model,WE have reached Gini impurity almost near 0 via this tree model.

From this we can determine correct Combination of the condition where Probability of the success is highest.




bhaskar das
Thank you!