SPACEX
Capstone Project Report

Manjunath S Karamudi

<Date>

Manjusk017@gmail.com

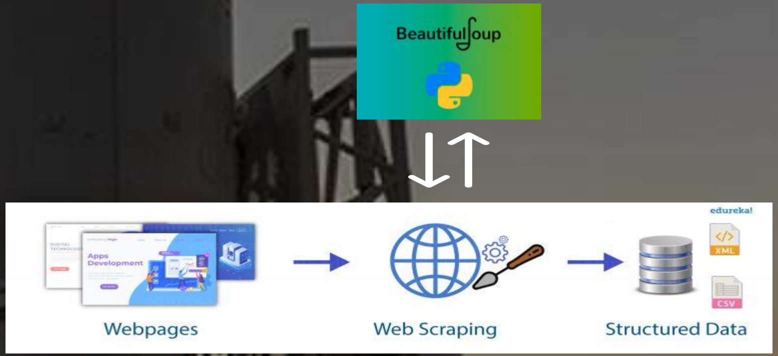
www.github.com/sk1793

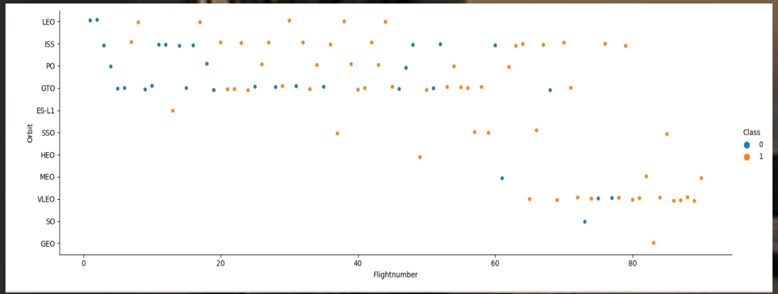
Outline

Executive Summary

Summary of methodologies

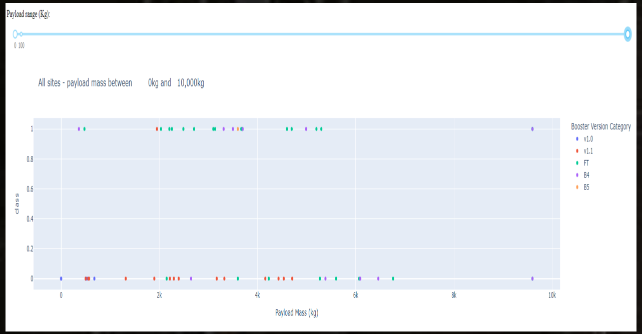
- Data Collection via API,SQL & Web Scrapping
- Data Wrangling & Analysis
- Interactive Maps(using Folium Library)
- Predictive Analysis for various Classfifcation Models (namely logistic Regression,Support Vector Machine(SVM),K-Nearest Neighbours(KNN),Random Forest)

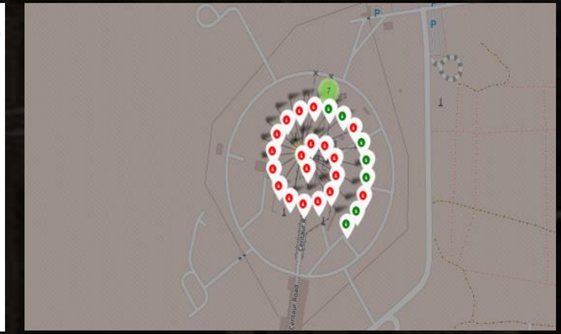


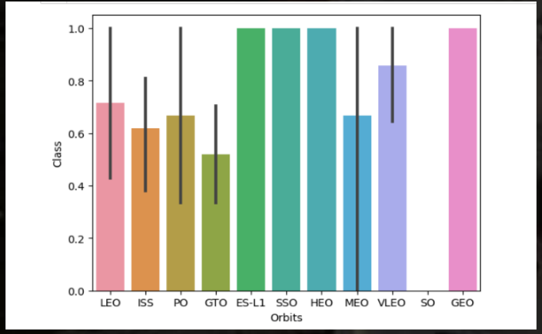


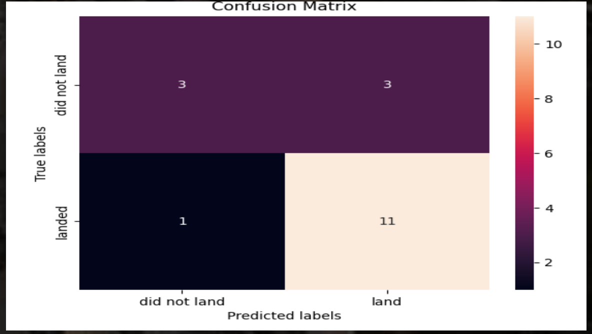
Summary of all results

- Data Analysis along with Interactive Visualizations
- Best models 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 because SpaceX can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

Problems you want to find answers

- Predicting if the first stage of the SpaceX Falcon 9 rocket will land successfully.
- Working out with various Conditions and finding their Relation with the Outcome.
- Finally We will find the how SPACEX can get Better Results in future.

Section - 1

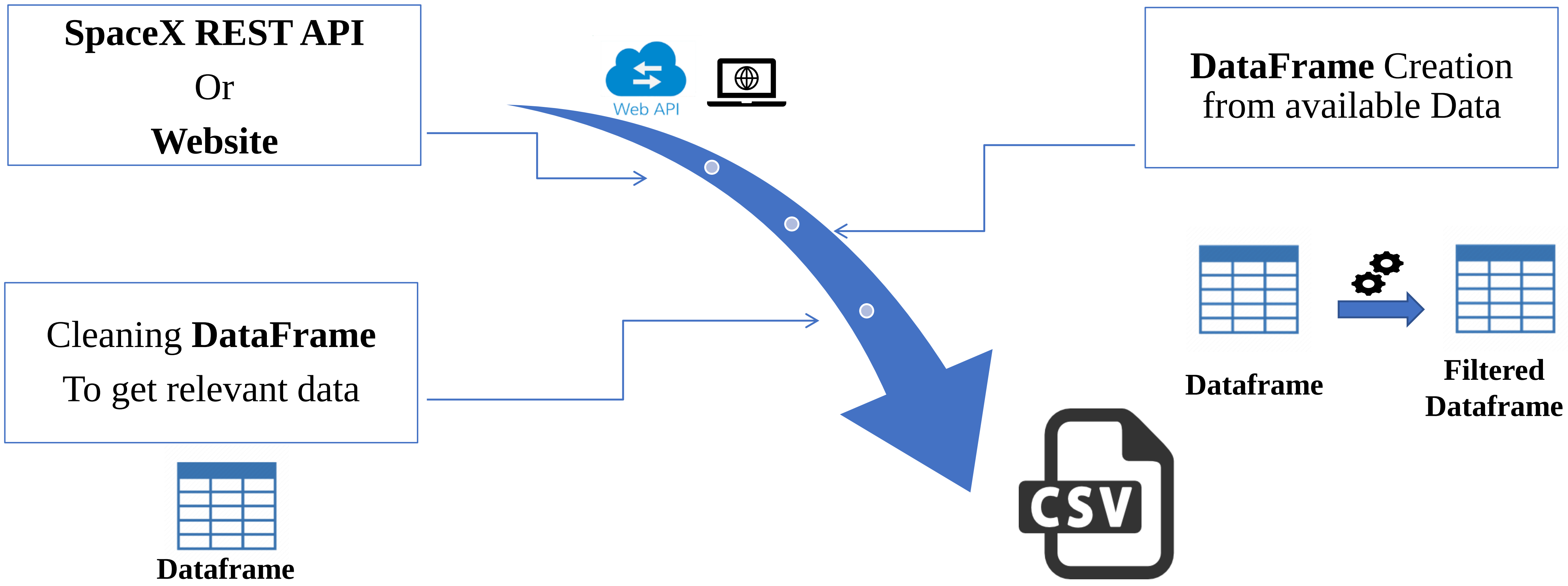
METHODOLOGY

Methodology

Executive Summary

- Data collection methodology:
  - Using SPACEX REST API.
  - Using BeautifulSoup for Web Scrapping Falcon 9 Launch data.
- Perform data wrangling
  - We Clean & Cleanse Data to Only get Relevent records for our Project.
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Using Barplots & Scatterplots for Better Understanding b/w Data.
- Perform interactive visual analytics using Folium and Plotly Dash
  - Using Maps(Folium) & Plotly Dash App for Interactice data Visualization.
- Perform predictive analysis using classification models
  - Initialize & Find Accuracy of Each Classification Models.

Data Collection



Data collection is the procedure of collecting, measuring and analyzing accurate insights for research using standard validated techniques. Here,

- We collect The Data Using **SpaceX Rest API /Web scrapping** Website.
- Then we Make a **DataFrame** from collected Data & Filter the data for our Research Accordingly.
- We save the Filtered data into a **CSV**(Comma Seperated Values) File For Further Process.

Data Collection – SpaceX API

```
In [7]: spacex_url="https://api.spacexdata.com/v4/launches/past"
In [8]: response = requests.get(spacex_url)
```

```
# Call getBoosterVersion
getBoosterVersion(data)
```

```
# Call getLaunchSite
getLaunchSite(data)
```

```
getPayloadData(data)
```

```
getCoreData(data)
```

```
#remove BoosterVersion version other than falcon 9
df.drop(df.index[(df['BoosterVersion']=="Falcon 1")],axis=0,inplace=True)
```

```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

```
mean_pm=data_falcon9['PayloadMass'].mean()
mean_pm
```

```
6123.547647058824
```

```
data_falcon9['PayloadMass'].replace(np.nan,mean_pm)
```

```
data_falcon9.to_csv('falcon9_df.csv',index=False)
```

Response From **API**

Store ResponseTo **.Json File**

Create **Functions** to Filter Data

Store Values to Dictionary & create a **Dataframe**

Filter DataFrame & Store to a **CSV File**

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

```
data=pd.json_normalize(response.json())
```

```
data.head()
```

	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

Github Link

Data Collection - Scraping

Get Response from **Web Site/URL**

Create **BeautifulSoup** Object

Filter to Find **Tables** from Site

Fetching **Column** Names

Create a **Dictionary** and append Values to its **Keys**

Dictionary to **DataFrame**

Dataframe to **CSV** File

```
#Web Scrapping the Wiki Page
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
```

```
soup = BeautifulSoup(data,'html.parser' )
html_tables=soup.find_all("table")
first_launch_table = html_tables[2]
#first_launch_table
```

```
ths = first_launch_table.find_all('th')
column_names=[]
for th in ths:
    name = th.text
    name=name.replace('\n','')
    if name is not None and len(name)>1:
        column_names.append(name)
column_names
```

```
['Flight No.',
 'Date andtime (UTC)',
 'Version,Booster [b]',
 'Launch site',
 'Payload[c]',
 'Payload mass',
 'Orbit',
 'Customer',
 'Launchoutcome',
 'Boosterlanding']
```

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

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

```
df.describe()
```

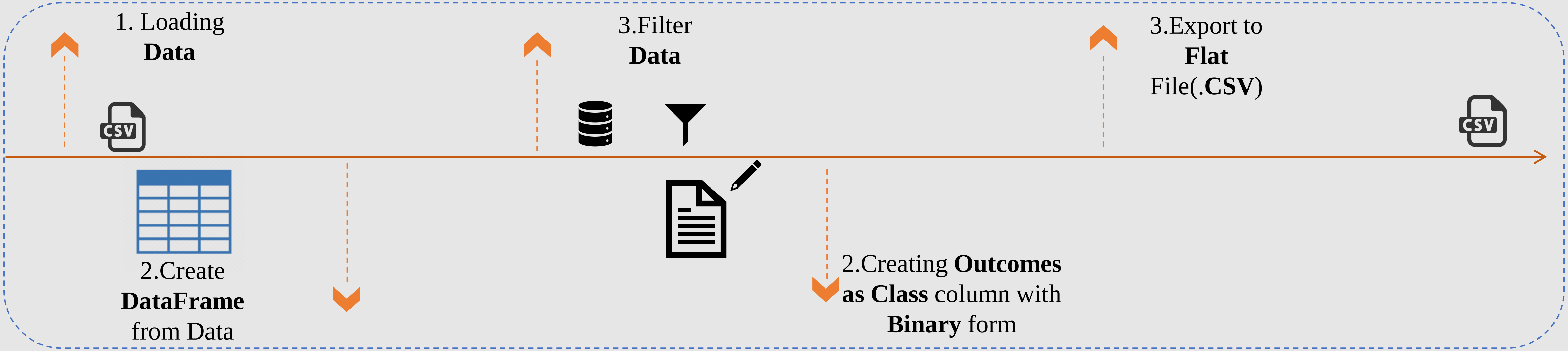
```
df.head()
```

	index	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
0	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	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	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	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	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

[Github Link]

Data Wrangling

Data Wrangling is the process of gathering, collecting, and transforming Raw data into another format for better understanding, decision-making, accessing, and analysis in less time.



1.Calculate no. of Launches at each site

```
# Apply value_counts() on column LaunchSite
df.value_counts(df['LaunchSite'])
```

```
LaunchSite
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
dtype: int64
```

2.Calculate no. & occurances of each orbit

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

```
Orbit
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
SO        1
HEO       1
GEO       1
ES-L1     1
dtype: int64
```

3.Calculate different types of Outcome

```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

```
0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 None ASDS
6 False Ocean
7 False RTLS
```

4. Separate Bad Outcomes From Outcomes

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

```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

5.Code

```
class_values=[1]*len(df)
c=0
for x in df['Outcome']:
    if x in bad_outcomes:
        class_values[c]=0
        c+=1
```

6.Result

	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
6	7	2014-04-18	Falcon 9	2296.000000	ISS	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1006	-80.577366	28.561857	1
7	8	2014-07-14	Falcon 9	1316.000000	LEO	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1007	-80.577366	28.561857	1

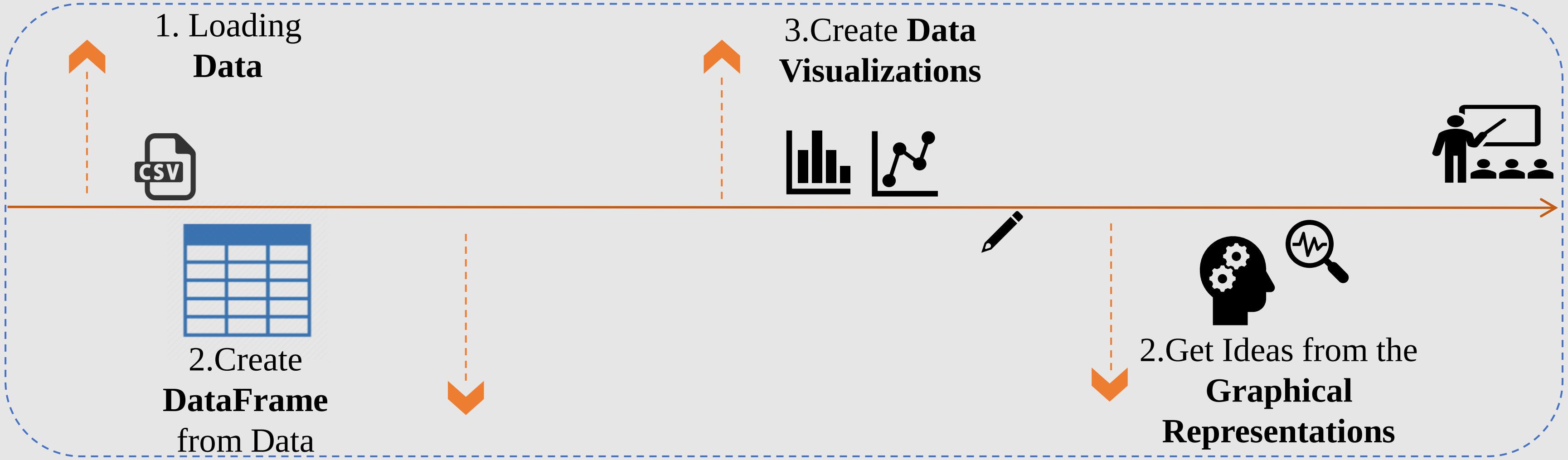
7.Export to .CSV

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

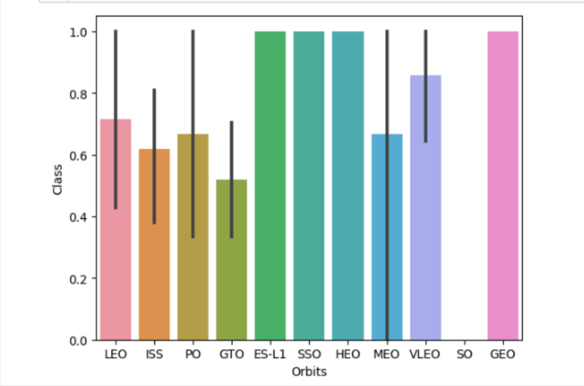
[Github Link]

EDA with Data Visualization

Exploratory data analysis (EDA) involves **using graphics and visualizations to explore and analyze a data set**. The goal is to explore, investigate and learn, as opposed to confirming statistical hypotheses.

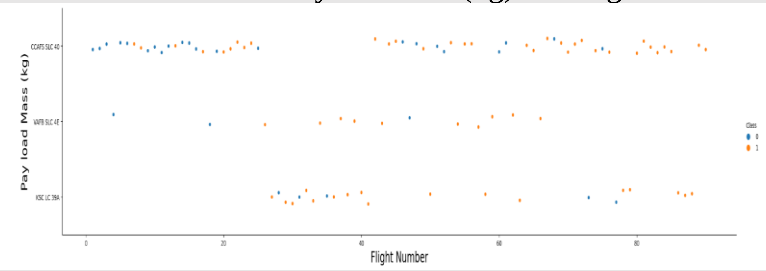


Bar Plots

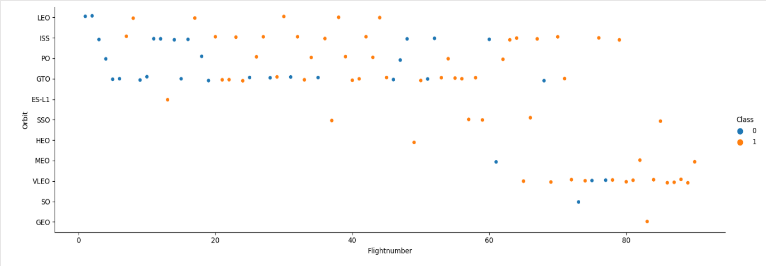


Scatter Plots

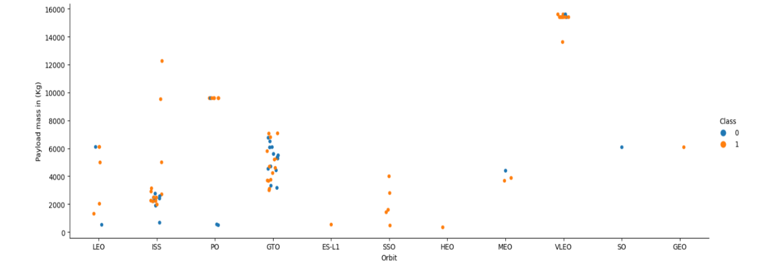
Payload Mass(kg) VS Flight No.



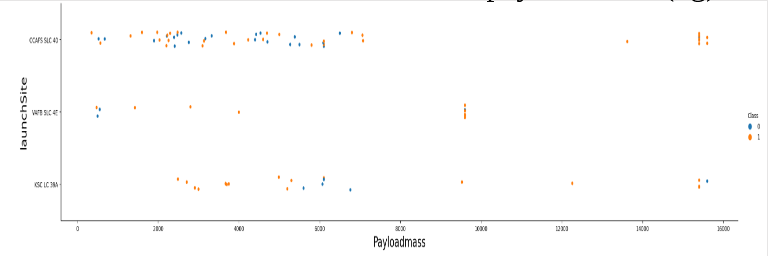
Orbit VS Flight No.



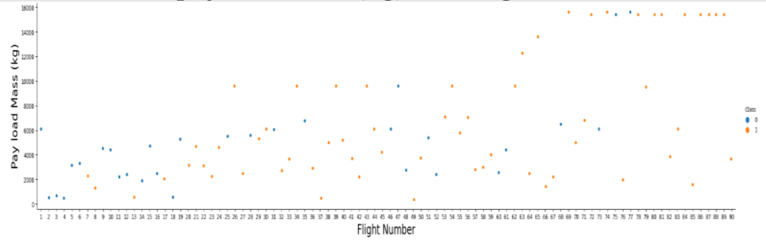
launchSIte VS payload Mass(kg)



launchSIte VS payload Mass(kg)

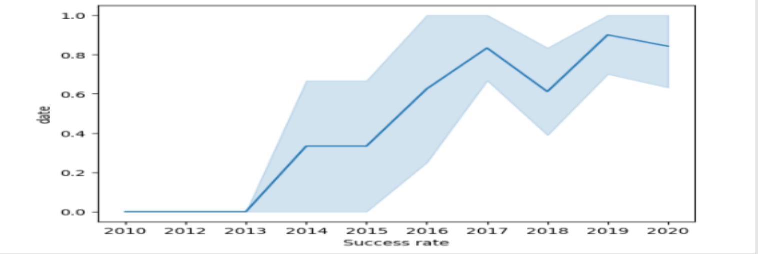


payload Mass(kg) VS Flight No.



Line Plots

launchSIte VS payload Mass(kg)



Github Link

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.

Following Are the Information We grabbed from dataset using SQL -

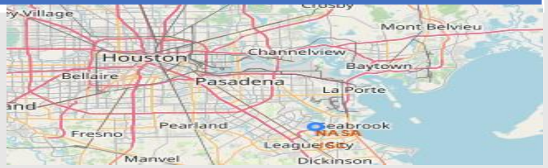
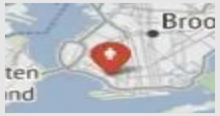
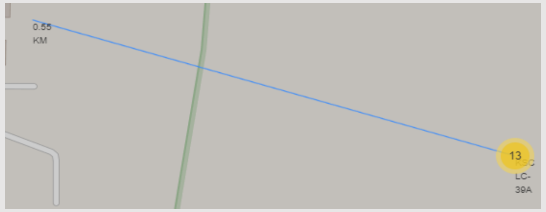
1. Getting Unique names of **LaunchSites** from SPACEX Launch Data.
2. Display 5 records where launch sites begin with the string **'CCA'**.
3. Getting the total payload mass carried by boosters launched by **NASA(CRS)**.
4. Getting average payload mass carried by booster version **F9v1.1**.
5. Listing the date where the **successful** landing outcome in drone ship was achieved.
6. Listing the names of the boosters which have success in ground pad and have payload mass **greater than 4000 but less than 6000**.
7. Listing the total **number of successful and failure** mission outcomes.
8. Listing the names of the **booster_versions** which have carried the maximum payload mass.
9. Listing the failed **landing_outcomes** in drone ship, their **booster versions**, and **launch site** names for the year 2015.
10. 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 Link

Build an Interactive Map with Folium

Github Link

Folium is a powerful Python library that makes it easy to visualise geospatial data.With it ,it’s 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.

Title	function	Syntax	Picture
Map Marker	Map object to make a mark on map.	folium.Marker()	
Icon Marker	Create an icon on map.	folium. Icon()	
Circle Marker	Create a circle where Marker is being placed	folium.Circle()	
Poly Line	Create a line between points	folium.PolyLine()	
Marker Cluster Object	This is a good way to simplify a map containing many markers having the same coordinate.	MarkerCluster()	

Build a Dashboard with Plotly Dash

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

- Here we used to get the **Success rate** based on launch site in **Percentage**(%).

Scatter Plot showing the correlation between **Payload** and **Success rate** for all sites or by certain **launch site**.

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

Item	Syntax	Usage
1. Pandas	Import pandas	Fetching Values from CSV file & Create **DataFrame**
2.Dash and its components	Import dash import dash_html_components as html Import dash_core_components as dcc from dash.dependencies import Input,Output	With **Dash** Open Source, Dash apps run on your System
3.Plotly	Import plotly.express as pe	Plot the graphs with interactive **plotly** library
4.RangeSlider	dcc.RangeSlider()	Create a **RangeSlider** for **Payload Mass** range selection
4.PieChart	pe.pie()	Create a **Pie Chart** for Successfull Launches by **LaunchSite**
5.Dropdown	dcc.Dropdown()	Create a **Dropdown** to select **Launch Site**
6.ScatterPlot	pe.scatter()	Create a **ScatterPlot** for Succesull Launches by **LaunchSite**

Github Link

Predictive Analysis (Classification)

Steps:

- Load Data & Create a **DataFrame**(DF).
- Importing Libraries & Converting DF into **Numpy** Arrays.
- **Standardize** & **Transform** the Data.
- Split Data into **Test**, **Train** sets & find number of test Samples Created.
- Initialize **ML models(**LogReg,SVM, DecisionTreeClassifier
- ,KNN**)** we want to use.
- Initialize & Fit earlier Models to **GridSearchCV** and Train.

```
Y=np.array(data['Class'].to_numpy())

transform = preprocessing.StandardScaler()

X=transform.fit(X).transform(X.astype(float))

xtrain,xtest,ytrain,ytest=train_test_split(X,Y
,test_size=0.2,random_state=2)

ytest.shape
```

Finding Best Performing Classification Model

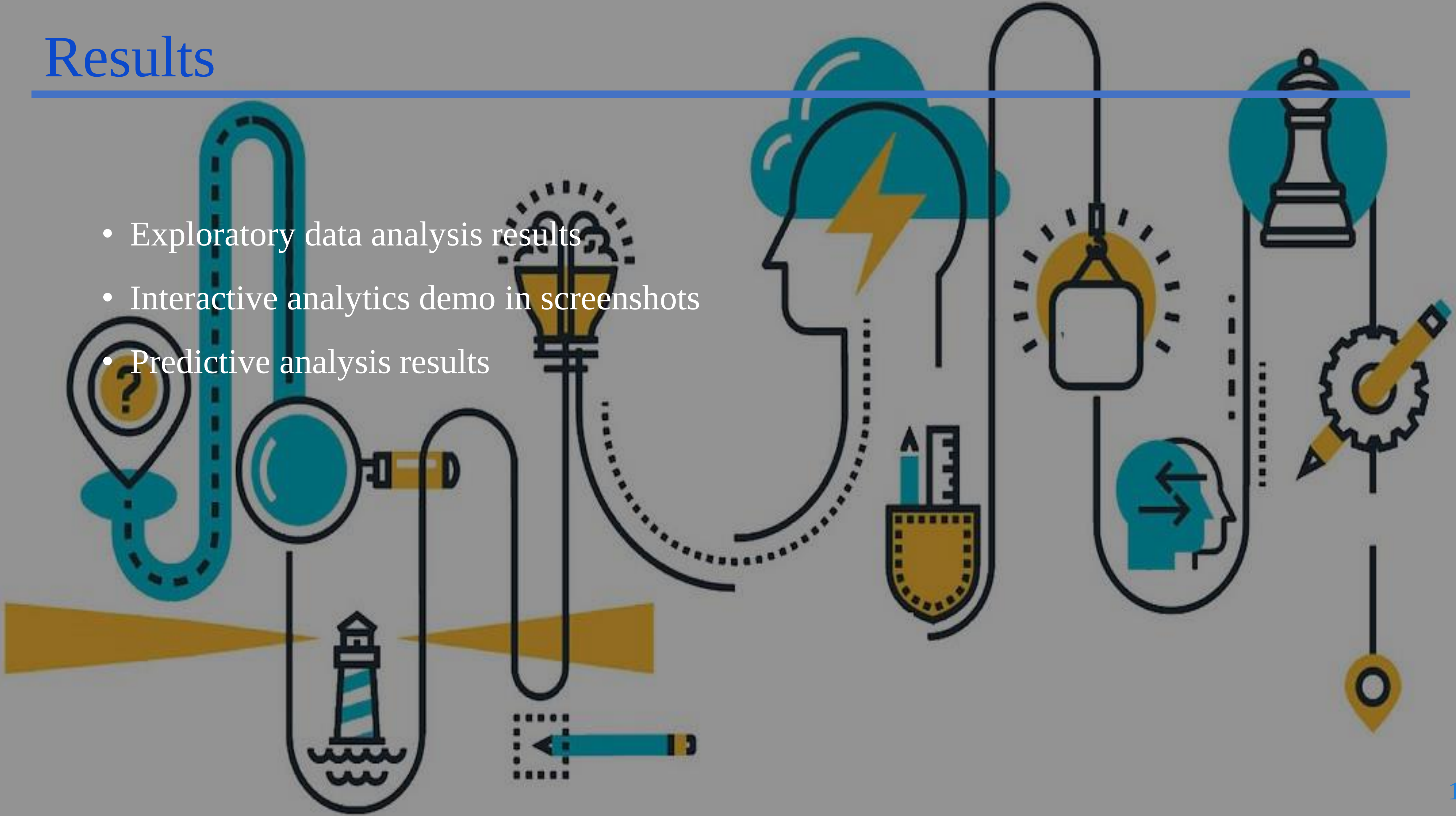
```
yhat=algorithm.predict(X_test)
plot_confusion_matrix(Y_test, yhat)
```

Evaluating Each Model

- Check Accuracy for each model.
- Get best hyperparameters for each type of algorithms.
- Form Confusion Matrix.

Best Peforming Model

Results



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

Section - 2

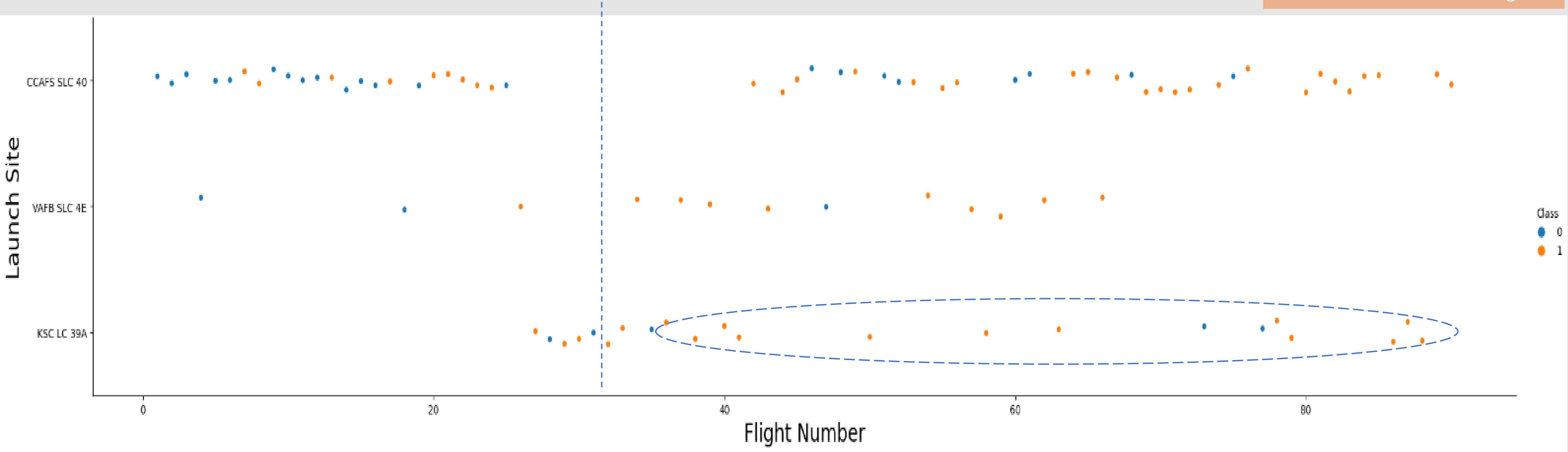
EDA Review with Visualization

Flight Number vs. Launch Site

Point After Which Success Rate has increased

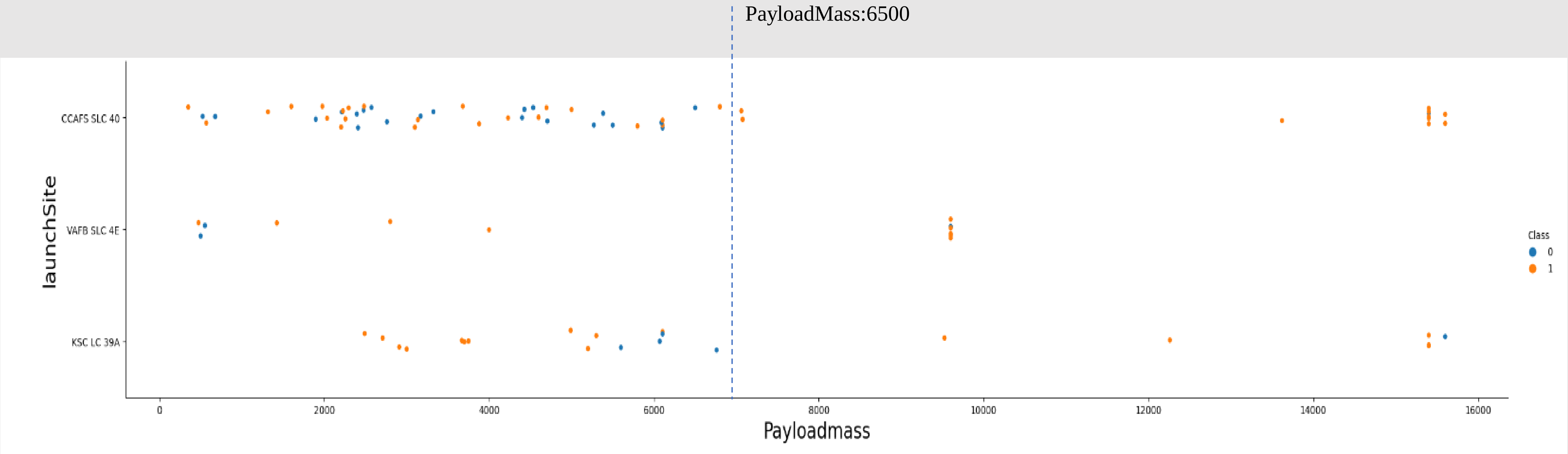
Class 0:Failed Landings

Class 1:Successful Landings



- It is Noticed that from **Flight number 35** the Success rate increased. And Among LaunchSites **KSC LC 39A** has Better Success Rate.

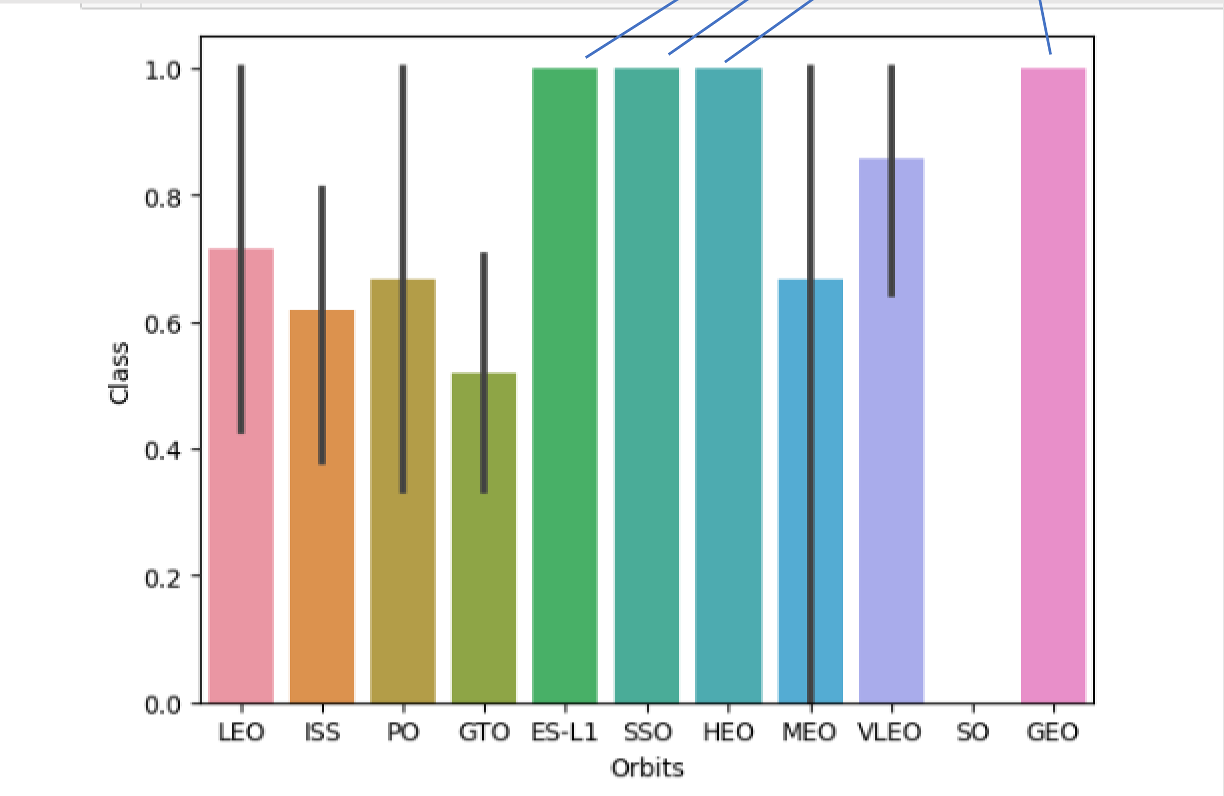
Payload vs. Launch Site



- It is Noticed that from **PayloadMass** Greater Than **6500** The Success Rate has increased.Although we don't have appropriate resources to say that payload Mas Correlation is enough to estimate Success rate.

Success Rate vs. Orbit Type

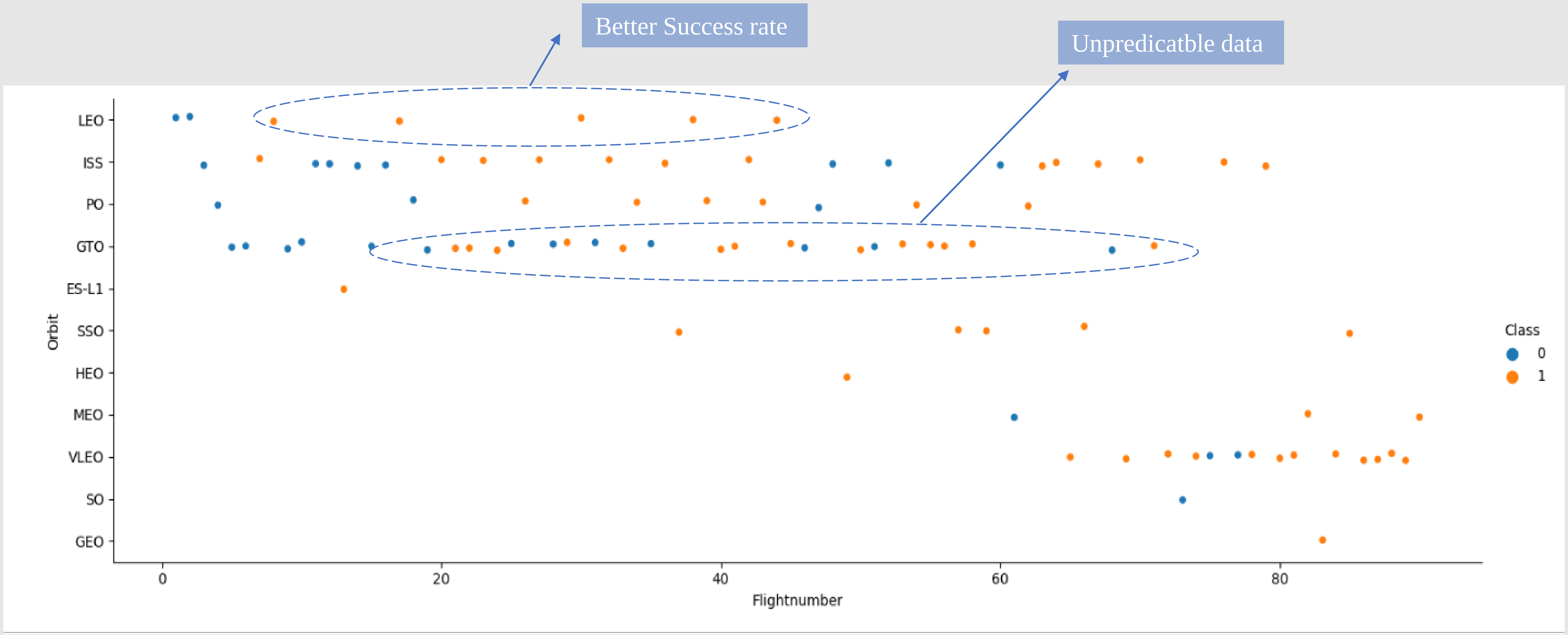
Successfully Landed



- Geostationary orbit (**GEO**)
- Low Earth orbit (**LEO**)
- Medium Earth orbit (**MEO**)
- Sun-synchronous orbit (**SSO**)
- Geostationary transfer orbit (**GTO**)
- Very Low Earth Orbits(**VLEO**)
- International Space Station(**ISS**)
- Highly Eliptical Orbit(**HEO**)
- Solar Orbit(**SO**)
- Polar Orbit(**PO**)
- Earth-Sun Lagrangian Points(**ES-L1**)

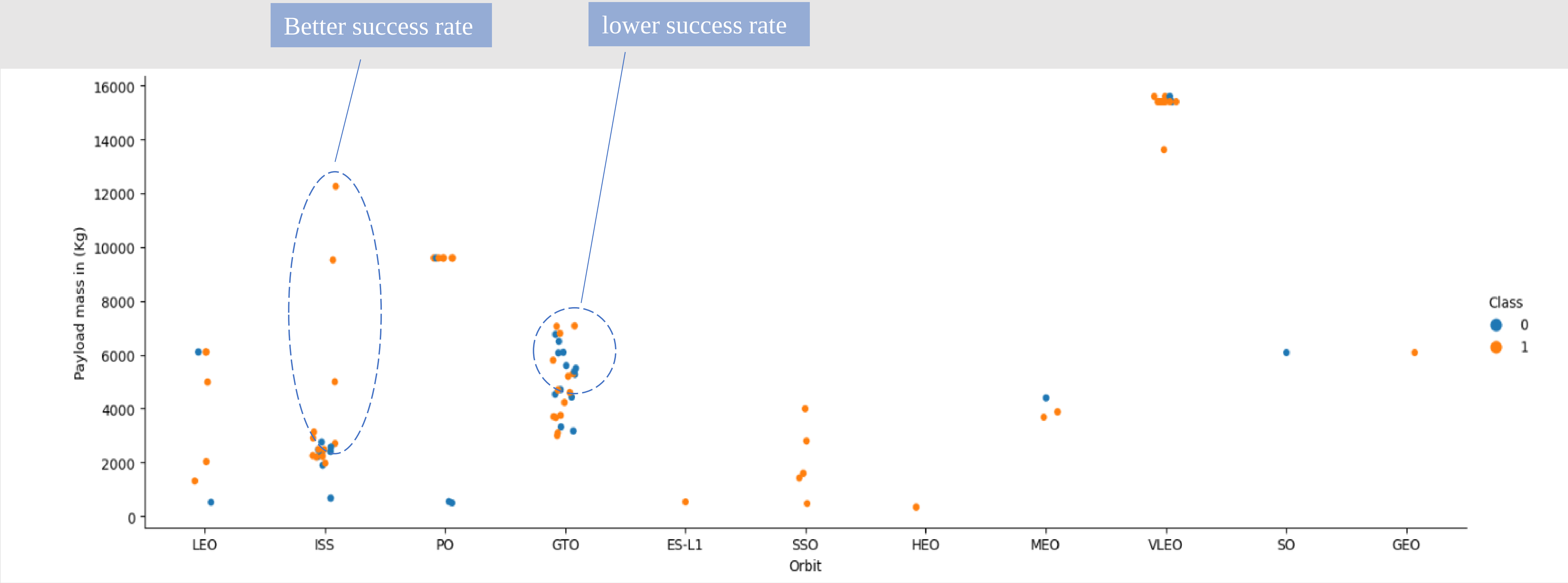
- It is Noticed that Orbits **ES-L1,SSO,HEO,GEO** have better Success rate among other Oribts.

Flight Number vs. Orbit Type



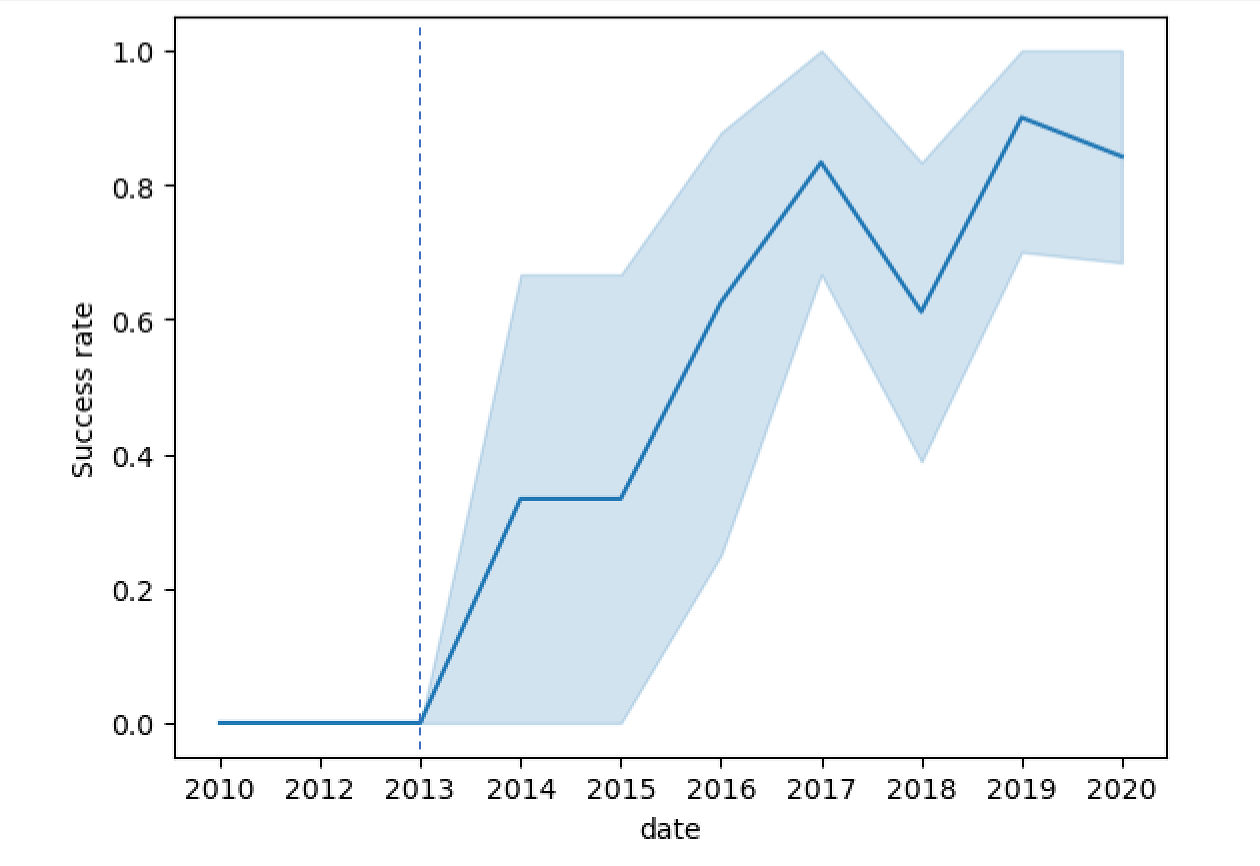
- It is Noticed that **LEO** Orbit has better Success rate with increase in **Flight Number**, rest of them can’t be concluded because of not enough data.
- Meanwhile Orbit **GTO** has not much Effect Towards **Flight Number**.

Payload vs. Orbit Type



- It is Noticed that Orbit **ISS** has **Positive** Result in terms of Success rate with increase in **Payload Mass**.
- Meanwhile Orbit **GTO** has **negative** Result with increase in **Payload Mass** with lower Success rate.

Launch Success Yearly Trend



- It is Noticed that from 2013 there's a higher Success rate.
- And can be predicted with rising tech. in days to come there would be more success achievable.

Section – 2.1

EDA Review with SQL

All Launch Site Names

```
#Display the names of the unique launch sites in the space mission
%sql select distinct Launch_Site from spacex
```

```
 * sqlite:///FinalDB.db
Done.
```

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

Query

Result

Interpretation:

- Here we get **LaunchSites Name** excluding Duplicates by using Keyword ‘**Distinct**’ in SQL Statement.

Launch Site Names Begin with 'CCA'

```
#Display 5 records where launch sites begin with the string 'CCA'
%sql select * from spacex \
where Launch_Site like 'CCA%' limit 10
```

```
 * sqlite:///FinalDB.db
Done.
```

Query

Result

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Result
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
03-12-2013	22:41:00	F9 v1.1	CCAFS LC-40	SES-8	3170	GTO	SES	Success	No attempt
06-01-2014	22:06:00	F9 v1.1	CCAFS LC-40	Thaicom 6	3325	GTO	Thaicom	Success	No attempt
18-04-2014	19:25:00	F9 v1.1	CCAFS LC-40	SpaceX CRS-3	2296	LEO (ISS)	NASA (CRS)	Success	Controlled (ocean)
14-07-2014	15:15:00	F9 v1.1	CCAFS LC-40	OG2 Mission 1 6 Orbcomm-OG2 satellites	1316	LEO	Orbcomm	Success	Controlled (ocean)
05-08-2014	08:00:00	F9 v1.1	CCAFS LC-40	AsiaSat 8	4535	GTO	AsiaSat	Success	No attempt

Interpretation:

- Here we get all Details from Spacex Table using **Asterisk**(*) where **LaunchSite** starts with keyword ‘**CCA**’ using keyword **like**.

Total Payload Mass

```
#Display the total payload mass carried by boosters launched by NASA (CRS)
%sql select sum(PAYLOAD_MASS__KG_) from spacex where Customer='NASA (CRS)'
```

Query

```
 * sqlite:///FinalDB.db
Done.
```

sum(PAYLOAD_MASS__KG_)
45596

Result

Interpretation:

- Here we find **Total PayloadMass** (in Kg's) of the Customer who's name is '**NASA (CRS)**'

Average Payload Mass by F9 v1.1

```
#Display average payload mass carried by booster version F9 v1.1
%sql select avg(PAYLOAD_MASS__KG_) from spacex where Booster_Version='F9 v1.1'
```

Query

```
 * sqlite:///FinalDB.db
Done.
```

avg(PAYLOAD_MASS__KG_)
2928.4

Result

Interpretation:

- Here we find **Average** of **PayloadMass** (in Kg’s) where Booster Version is ‘**F9 v1.1**’.

First Successful Ground Landing Date

```
#List the date when the first successful landing outcome in ground pad was acheived
%sql select date  from spacex where Landing_Result='Success (ground pad)' limit 1
```

```
 * sqlite:///FinalDB.db
Done.
```

Date
22-12-2015

Query

Result

Interpretation:

- Here we fetch date when the first Successful Landing Ouctome in ground pad was Achieved using keyword ‘**Success (ground pad)**’ and limiting to one value with keyword **limit** by 1 since it is in **assending order** by **default** we’ll get first Launch date .

Successful Drone Ship Landing with Payload between 4000 and 6000

```
#List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
%sql select Booster_Version from spacex where Landing_Result='Success (drone ship)'\
and PAYLOAD_MASS__KG_ between 4000 and 6000
```

Query

```
 * sqlite:///FinalDB.db
Done.
```

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

Result

Interpretation:

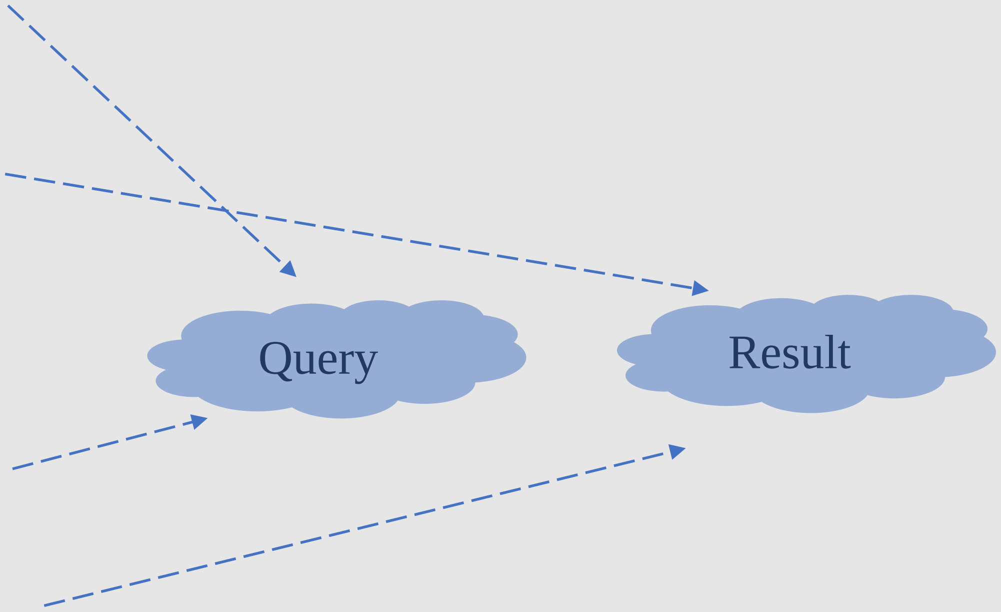
- Here we fetch **Booster Versions** which have Success in drone ship using ‘**Success (drone ship)**’, and have **PayloadMass** in Between **4000-6000** Kg’s with Keyword **between** .

Total Number of Successful and Failure Mission Outcomes

```
#List the total number of successful and failure mission outcomes
%sql select count(*),Mission_Outcome as Sucessfull_missions\
from spacex where Mission_Outcome like '%Success%'
```

```
 * sqlite:///FinalDB.db
Done.
```

count(*)	Sucessfull_missions
100	Success




```
%sql select count(*),Mission_Outcome as failed_missions\
from spacex where Mission_Outcome not like '%Success%'
```

```
 * sqlite:///FinalDB.db
Done.
```

count(*)	failed_missions
1	Failure (in flight)

Interpretation:

- Here we first find number of **Successful mission** outcomes using Keyword ‘**Count**’ and ‘**%Success%**’ using keyword **like**.
- Here we also find number of **Failed mission** outcomes like above but with **not** keyword

Boosters Carried Maximum Payload

```
#List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
%sql select Booster_Version from spacex\
where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from spacex)
```

Query

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

Result

Interpretation:

- Here we get names of **Booster Versions** which have carried the maximum **PayloadMass**(in Kg’s) , with using nested query Statements in between ‘(‘ ‘)’ **parentheses** and maximum Value with Keyword ‘**max**’.

2015 Launch Records

```
#List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
%sql select Booster_Version,Launch_Site from spacex \
where Landing_Result like '%drone ship%' and date like '%2015%'
```

Query

```
 * sqlite:///FinalDB.db
Done.
```

Booster_Version	Launch_Site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40
F9 v1.1 B1018	CCAFS LC-40

Result

Interpretation:

- Here we get **Booster Versions , LaunchSite** names, for **failed landing Outcomes** in **2015** using keyword **like** with '**%drone ship%**' and '**%2015%**' .

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

```
#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
%sql select count(Landing_Result),Landing_Result \
from spacex  where date between '04-06-2010' and '20-03-2017'\
group by Landing_Result \
order by count(Landing_Result) desc
```

count(Landing_Result)	Landing_Result
20	Success
11	No attempt
8	Success (drone ship)
6	Success (ground pad)
4	Failure (drone ship)
3	Failure
3	Controlled (ocean)
2	Failure (parachute)

Query

Result

Interpretation:

- Here we get count of **landing outcomes** in between date **2010-06-04** and **2017-03-20** in **descending** order with keyword **count** for landing_Result Column, '**between**' for dates and then keyword '**group**' result by Landing_result column and using keyword '**Desc**' for result to displayed in **descending** order of Landing_result.

Section – 3

CREATE MAPS USING DATA WITH FOLIUM

All Launch Sites on Folium Map

We can see the **SPACEX** Locations in **USA** on World Map here.

- Can see the marker with **blue** dot on the Fig.1.
- Can see the Location surrounded by **Black Circle** slightly transparent around the **SPACEX** Location on Fig.2.

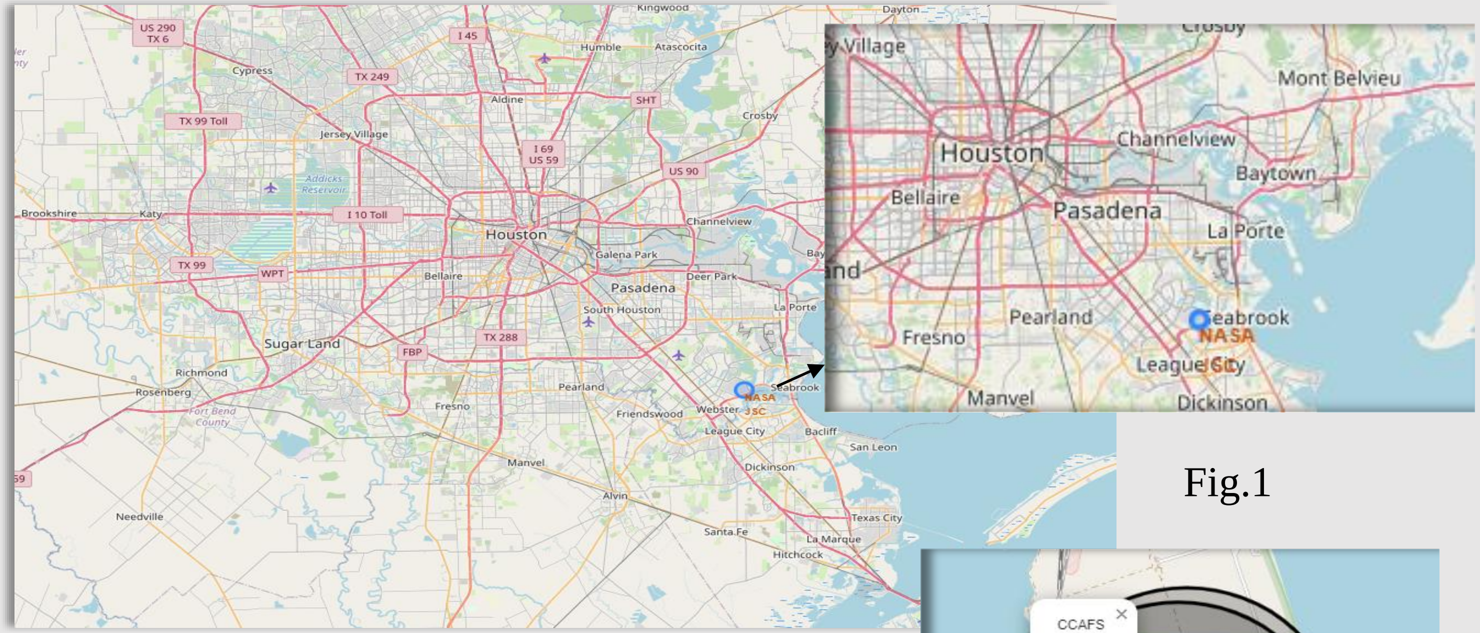


Fig.1

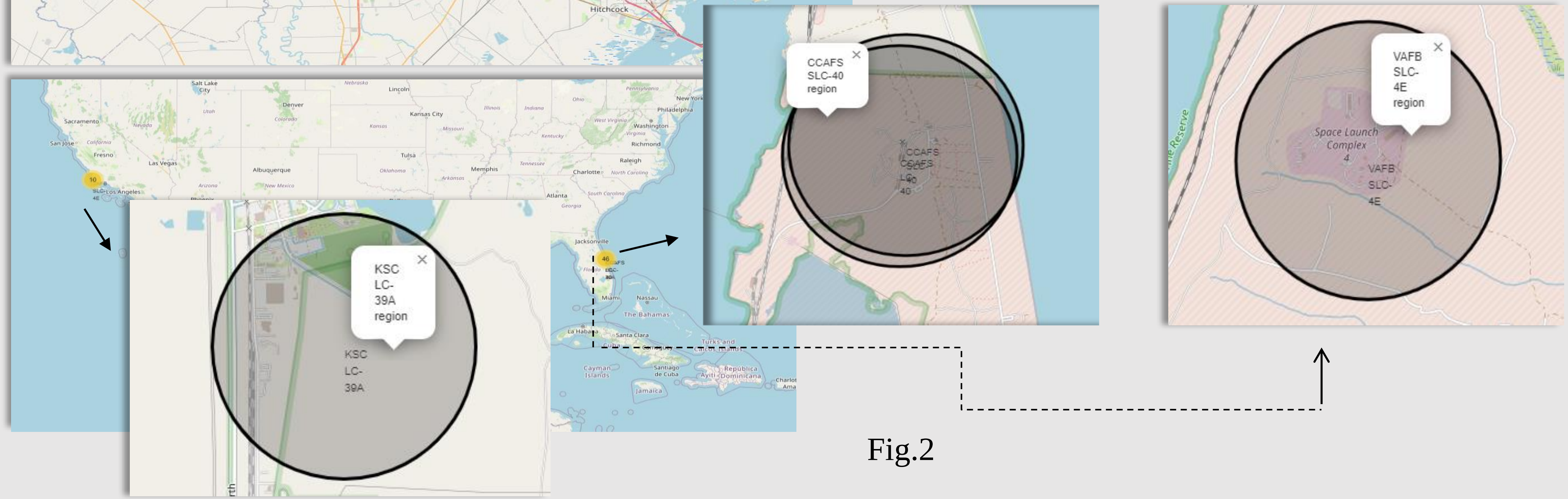
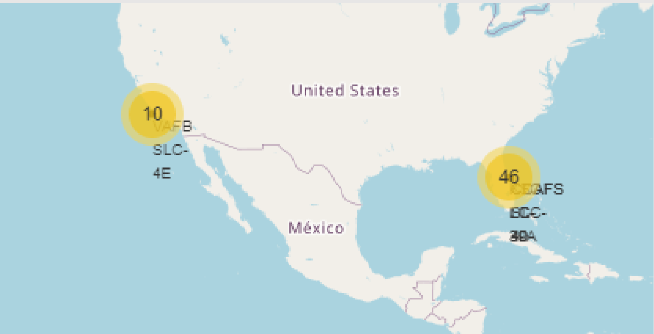


Fig.2

COLOR LABELED LAUNCH RECORDS



The Yellow **Circle Markers** Denotes the **SPACEX** Launch Locations and **Numbers** on them the number of Launches.

Denotes **Failed** SPACEX Launches.

Denotes **Successfull** SPACEX Launches.

From following Figures below **Fig 1,2,3,4** we can conclude that **Fig.3** which is **KSC LC-39A** has **maximum** Success Launch Rate and **Fig.1** which is **CCAFS SLC-40** has **minimum** Success Launch Rate respectively.

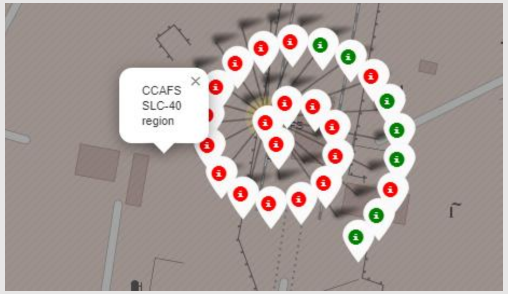


Fig.1

Fig.2

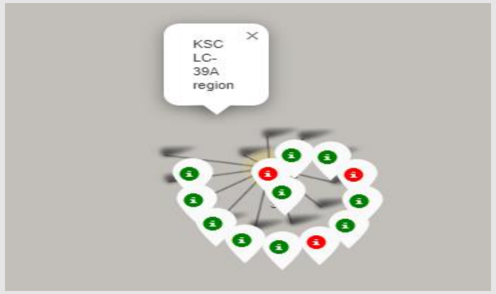


Fig.3

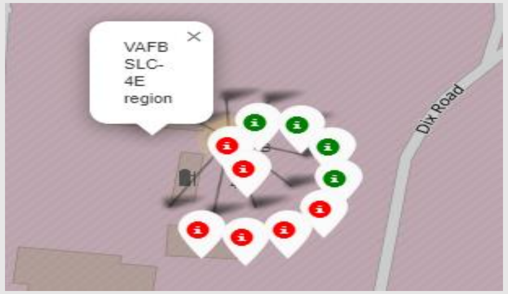


Fig.4

LAUNCH SITE DISTANCES FROM RAILWAYS

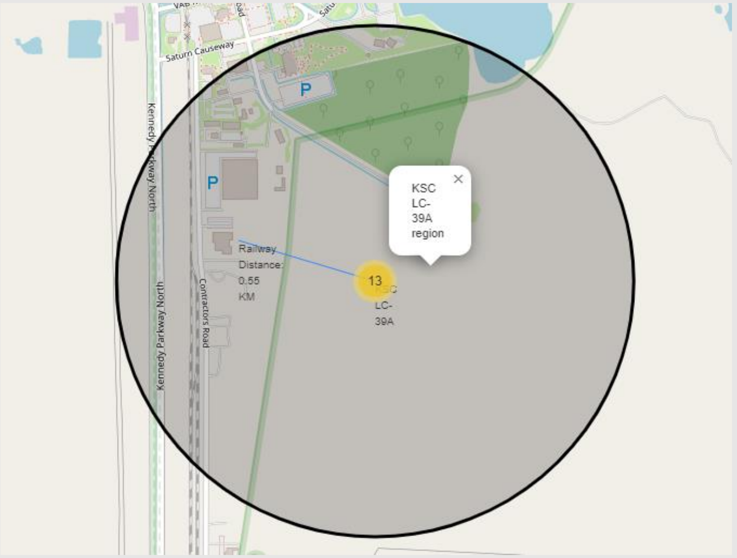


Fig.1

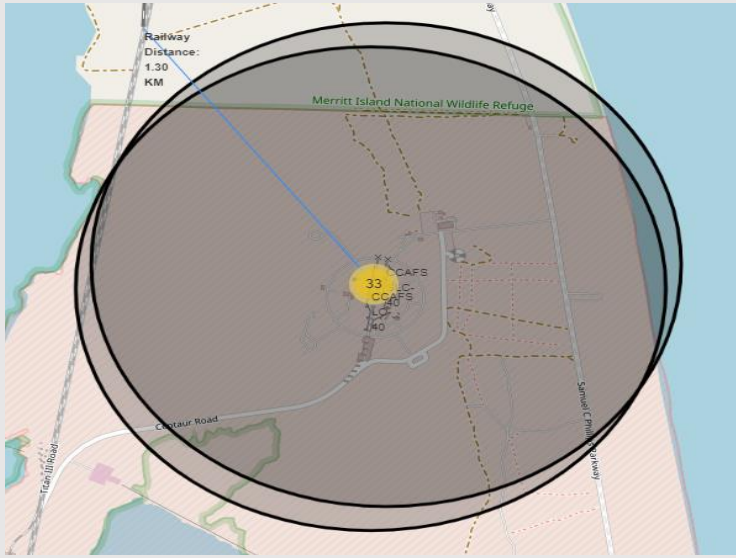


Fig.2

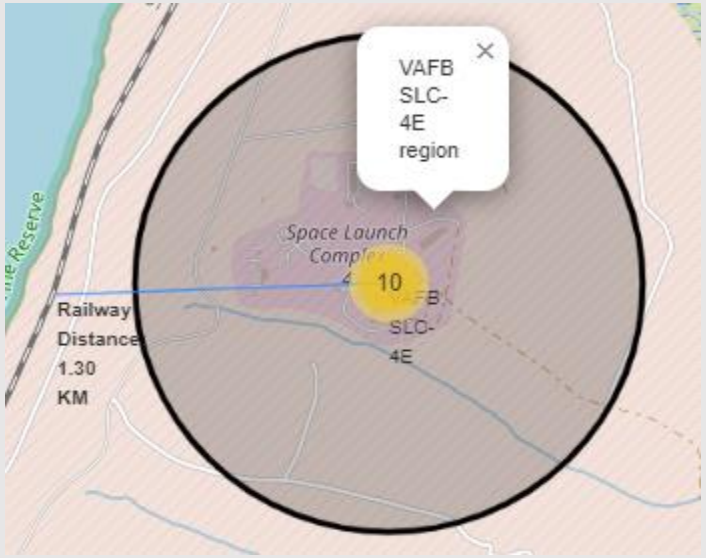


Fig.3

- As we can see all the distances of Launch Locations from **Railway Track** are greater than **0.5** kms.
- Launch site **KSC LC 39A** has least distance from **Railway Track** as seen in **Fig.1** i.e 0.55 kms.

LAUNCH SITE DISTANCES FROM CITY

Fig.1

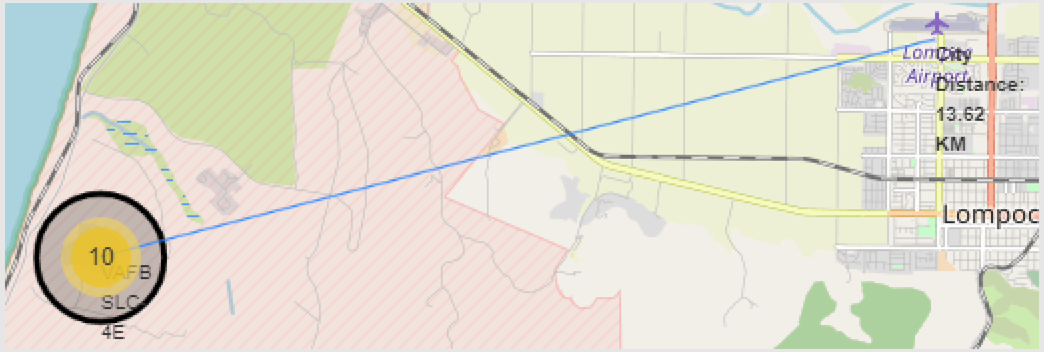


Fig.2

- As we can see all the distances of Launch Locations from **City** are greater than 12 kms.
- Launch site **VAFB SL 4E** has the least distance from **City** i.e 13.62 kms as seen in in **Fig.2.**

LAUNCH SITE DISTANCES FROM COASTLINE

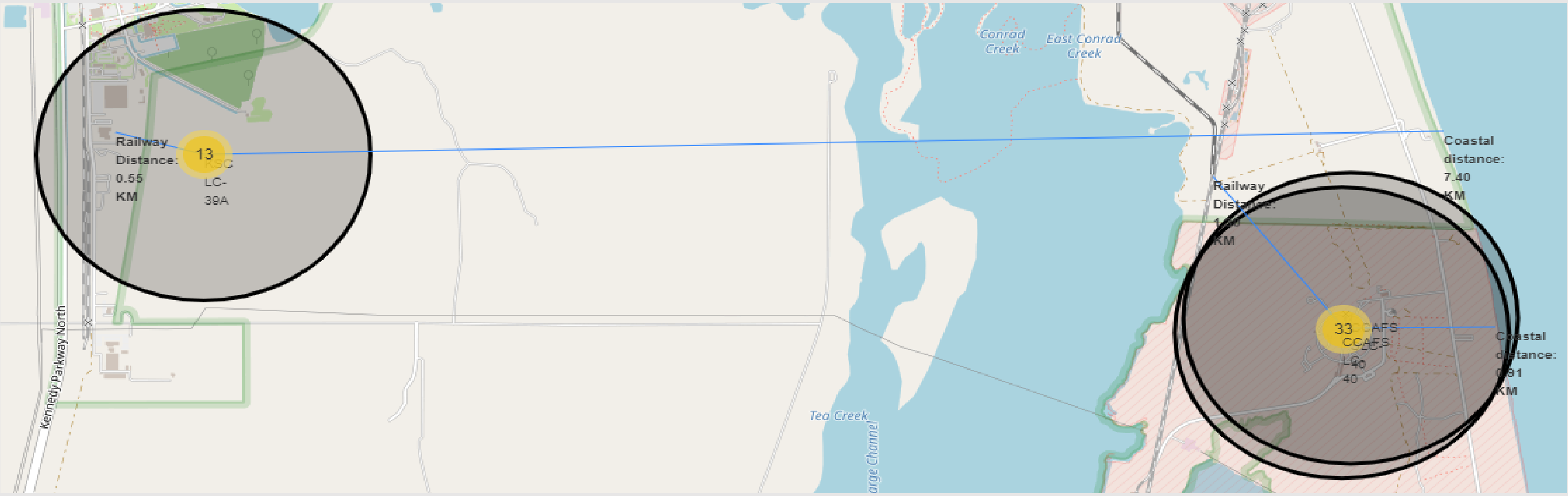


Fig.1

Fig.2

- As we can see all the distances of Launch Locations from **Coastal region** are greater than 0.8 kms.
- Launch site **CCAFS SLC/LC 40** has the least distance from **Coastal region** i.e 0.91 kms as seen in in **Fig.1.**

LAUNCH SITE DISTANCES FROM HIGHWAY

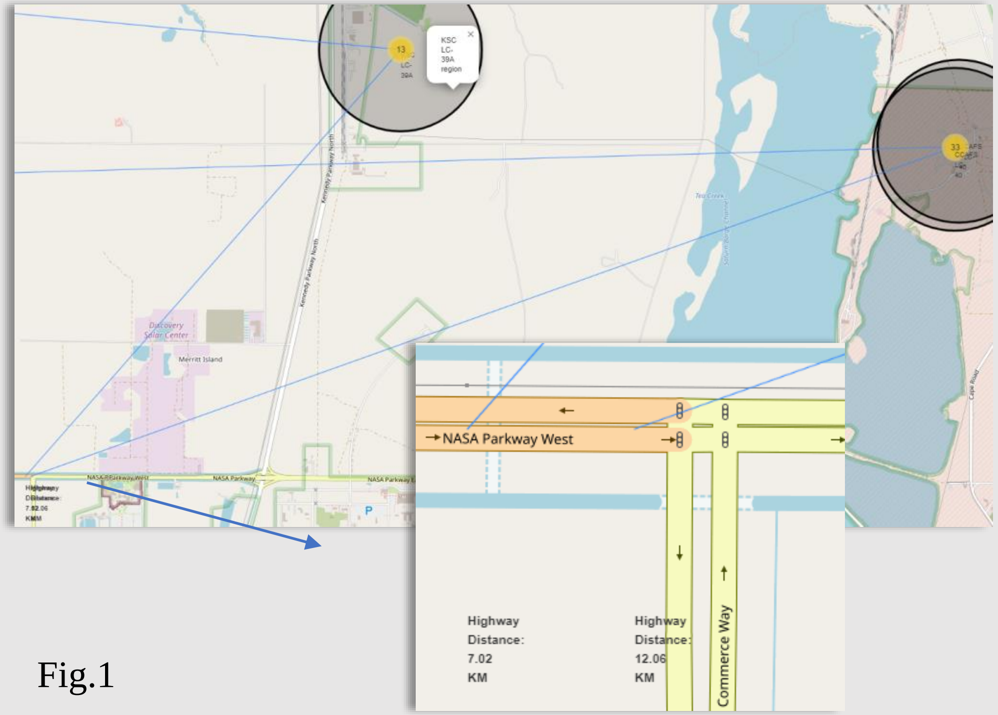


Fig.1

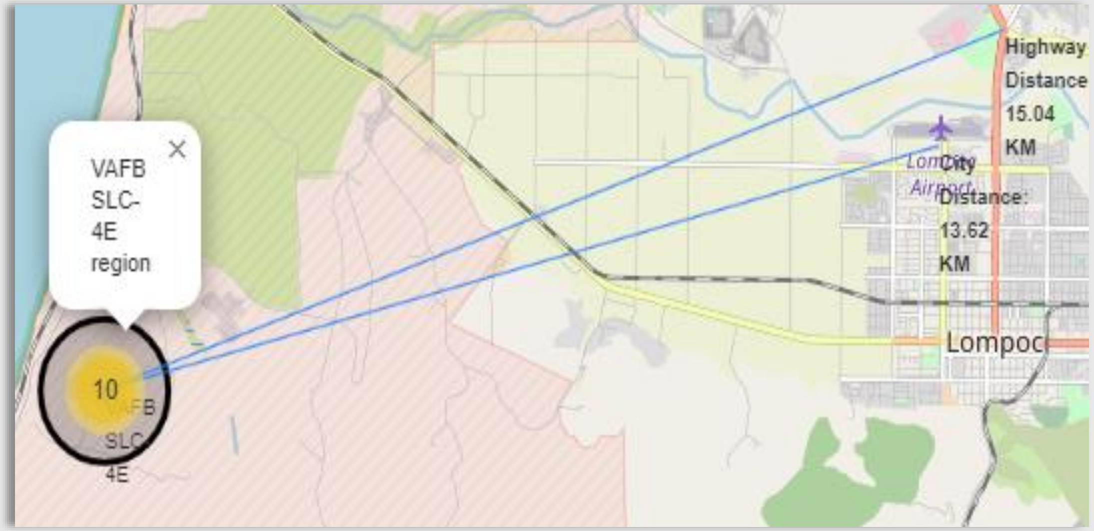


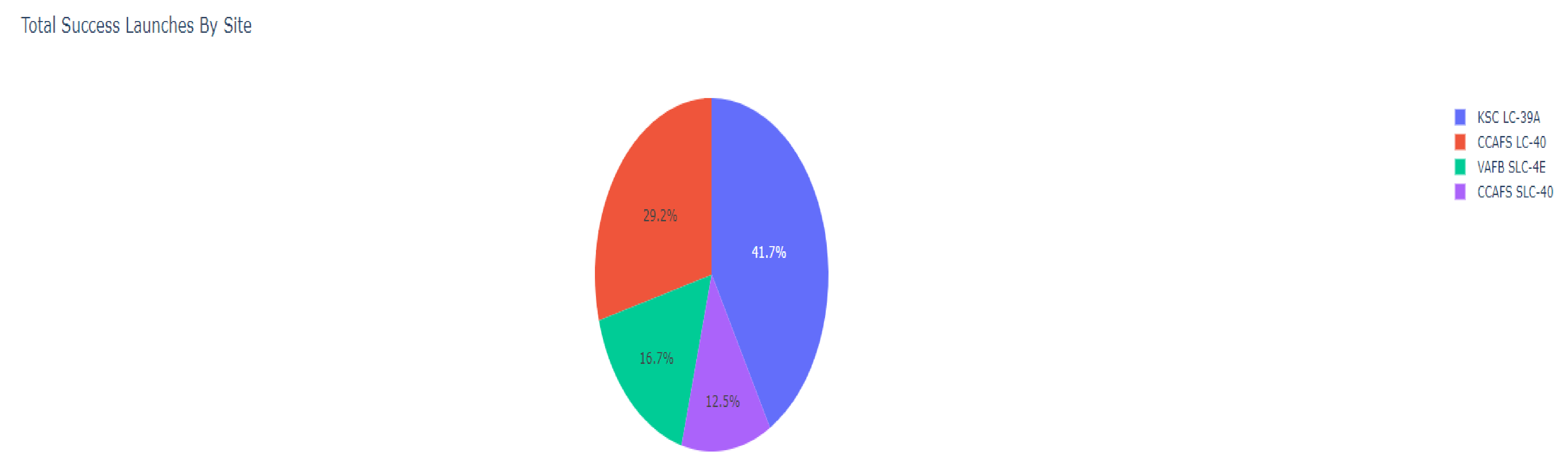
Fig.2

- As we can see all the distances of Launch Locations from **Highway** are greater than 12 kms.
- Launch site **KSC LC 39A** has the least distance from **Highway** i.e 7.62km as seen in in **Fig.1.**

Section – 4

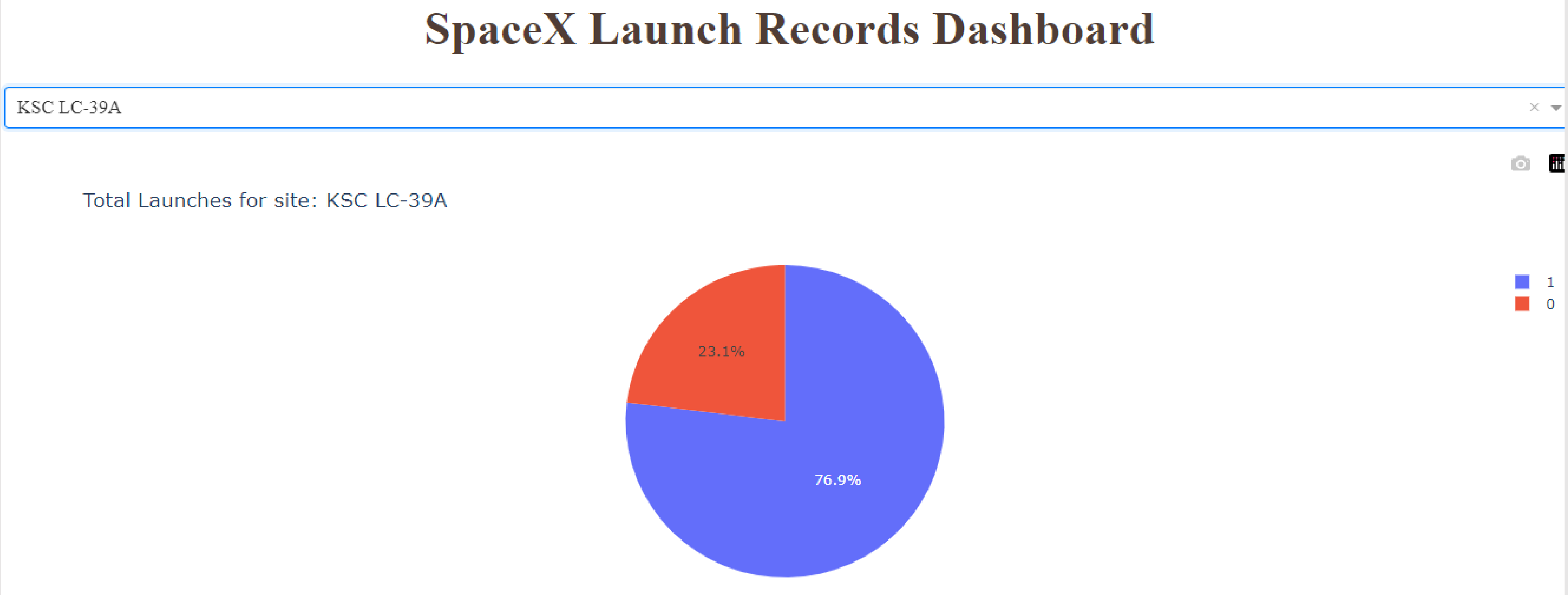
DASHBOARD BY PLOTLY DASH

LAUNCH SUCCESS MEASURE FOR ALL SITES



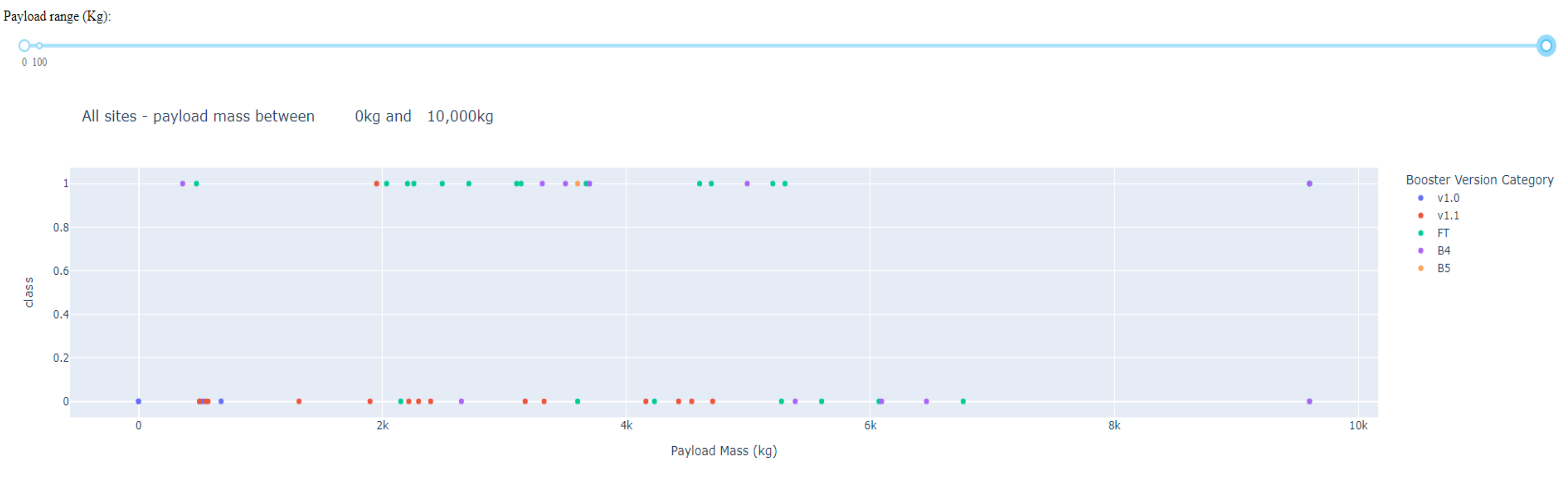
- It is absorbed that **KSC LC-39A** had the most successful launches from all the sites from above referenced figure.

LAUNCH SITE WITH HIGHEST SUCCESS RATIO



- It is absorbed that **KSC LC-39A** has achieved a **76.9%** success rate.
- Between payload Mass **2000-10000 kg** highest launch success rate.
- Between payload Mass **0-1000 kg** has the lowest launch success rate.
- F9 Booster version **FT** has the highest launch success rate(among **FT,B4,B5,v1.0.v1.1**).

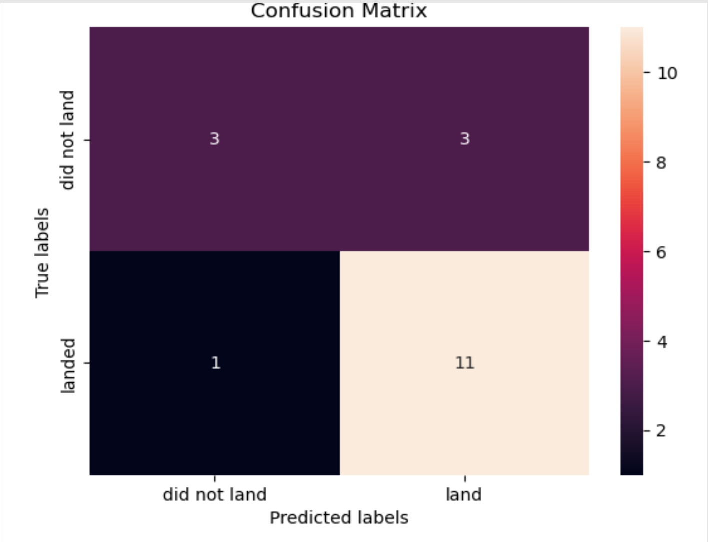
PAYLOAD VS. LAUNCH OUTCOME SCATTER PLOT FOR ALL SITES



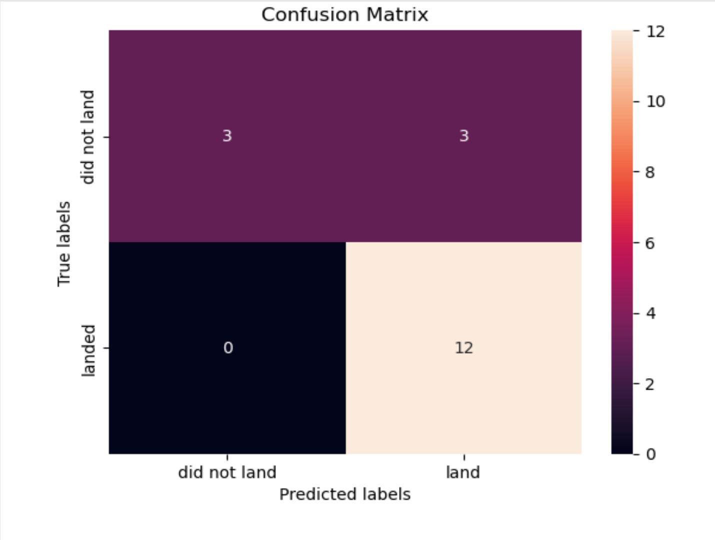
- It is absorbed that success rates for **low weighted** payloads is **higher** than the **heavy weighted** payloads from the above referenced figure.

Section – 5

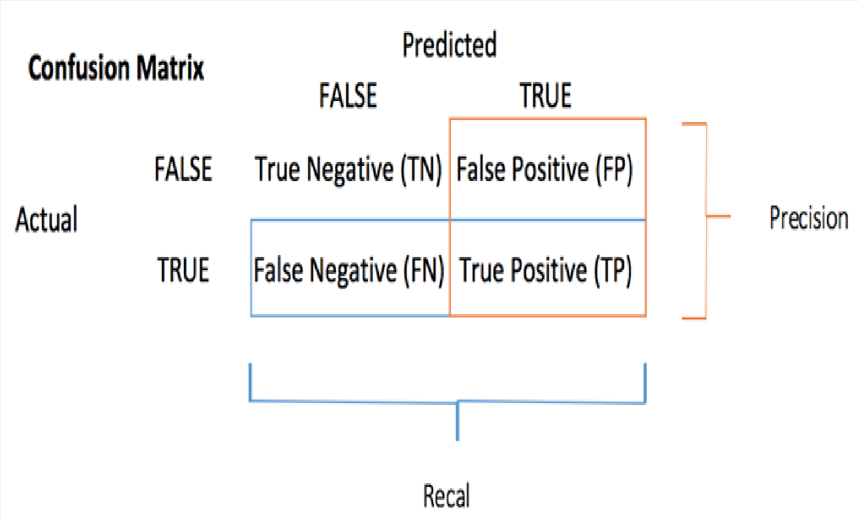
CONFUSION MATRIX



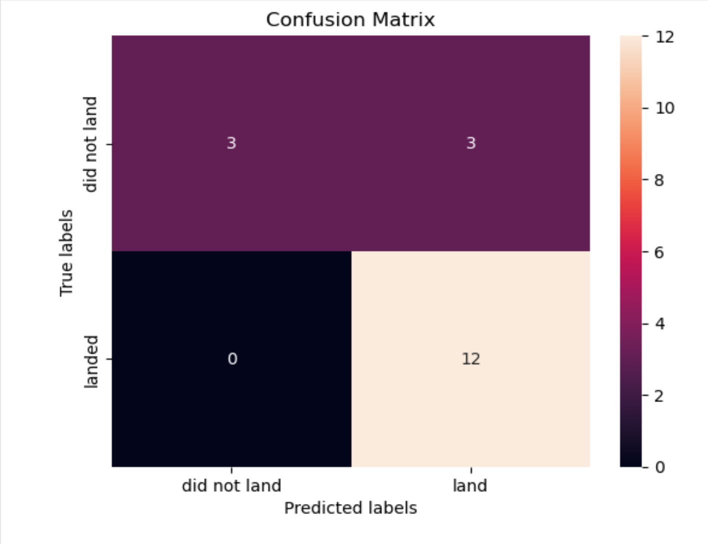
Decision Tree Classifier



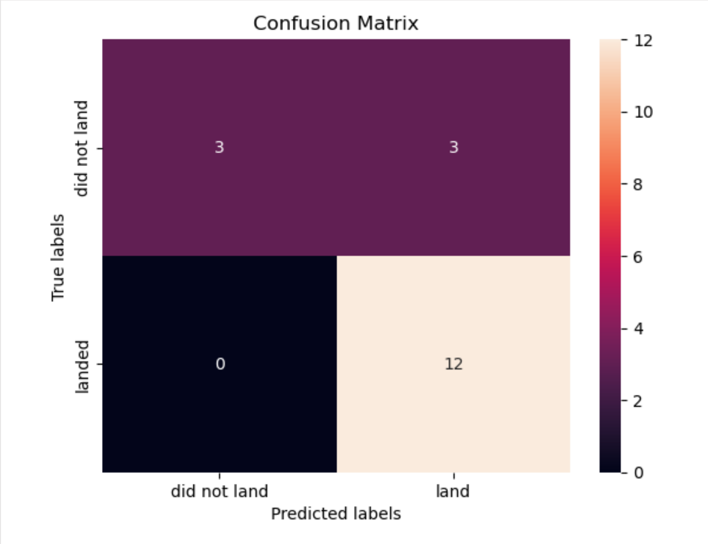
KNN



We can notice for all the models matrix being Same.



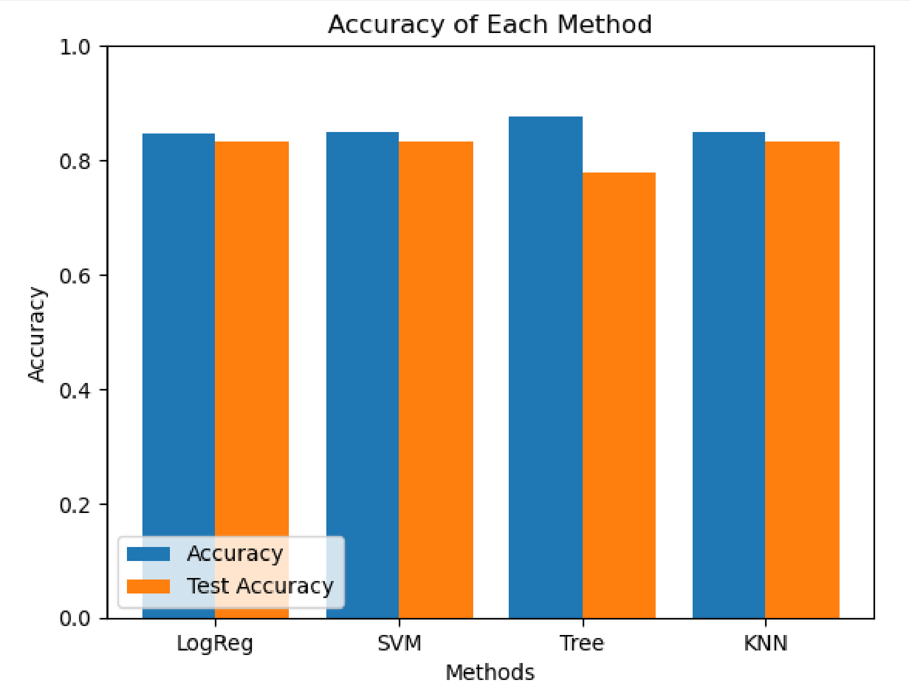
Logistic Regression



Support Vector Machine

- **Accuracy**: (TP+TN)/Total = (12+3)/18
- =0.83333
- **Misclassification Rate**: (FP+£N)/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

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. AN**

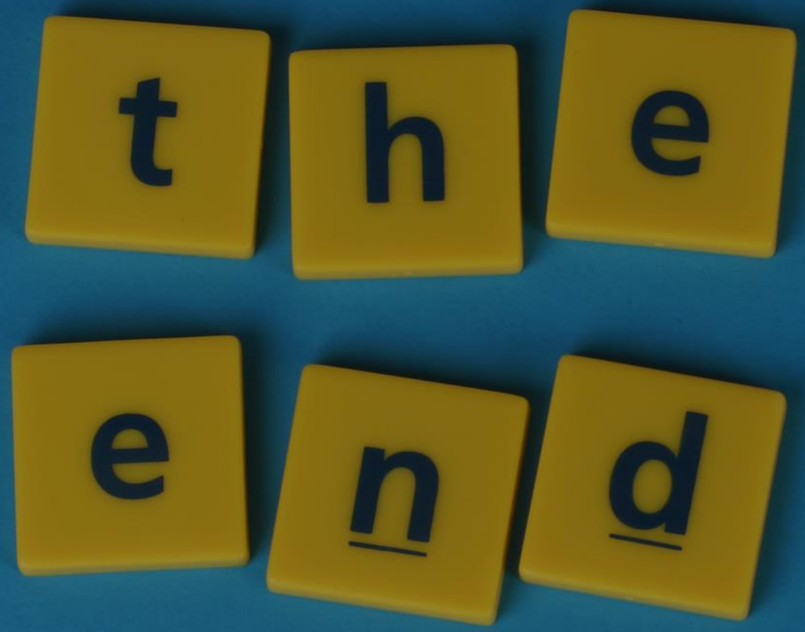
ML Model	Accuracy	Accuracy(Test Data)	Hyperparameters
Logistic Regression	0.846429	0.83334	{'C':0.01, 'penalty': "12", "solver: 'Ibfgs'}
SVM	0.848214	0.83334	{'C':1.0, 'gamma': 0,03162277660168379, "kernel': sigmoid}
KNN	0.848214	0.83334	{'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
Decision Tree	0.901786	0.83334	{'criterion':'gini', 'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf':1,'min, _split':2, 'splitter': best'}

Conclusions

- Orbits **E-L1, GEO, HEO, SSO** has highest Sucess rates.
- **KSC LC-39A** had greater **Succesfull Launches rate**
- **Success rates** for SpaceX launches has been increasing **relatively** with **time** and it looks like soon they will reach the **required target.**
- **Decision Tree Classifier** Algorithm is the best Suited ML **Model** for the dataset used.

Appendix

- **Plotly** for Interactive Graphics like plots, scatter plots, area charts, bar charts etc.
- **BeautifulSoup4** python library for Web Scrapping while Data Collection Process.
- **Pandas** for Data related operations including Data analysis.
- **Numpy** for Mathematical operations.
- **Sqlite3** for data quering using dataframe.
- **Folium** for Interactive Map Interfaces fro locating and measuring distances.
- **Plotly** to create Dashboard App for data Visualization.
- **Jupyter** notebook by Anaconda Navigator for almost all operations.
- **Visual studio code** to create run Dash App.



Have a nice day
:)