IBM Developer SKILLS NETWORK

DATA SCIENCE CAPSTONE PROJECT

SALVI VATSA

10th MAY 2022

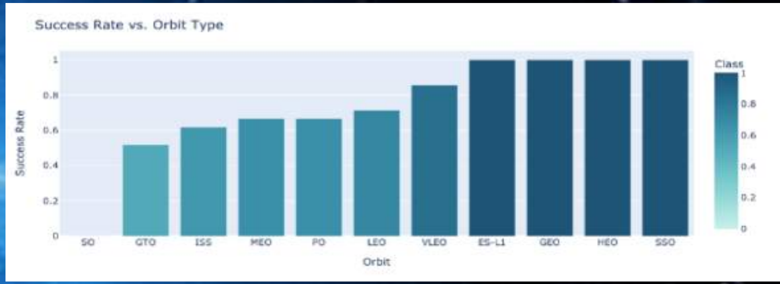
Outline

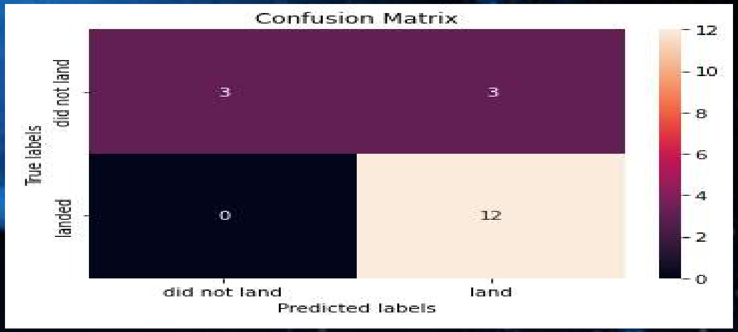
- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix

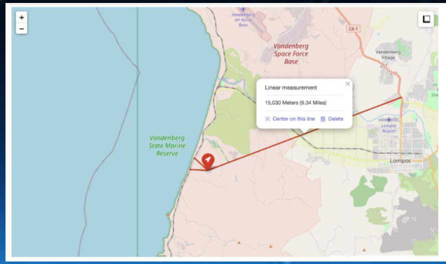
Executive Summary

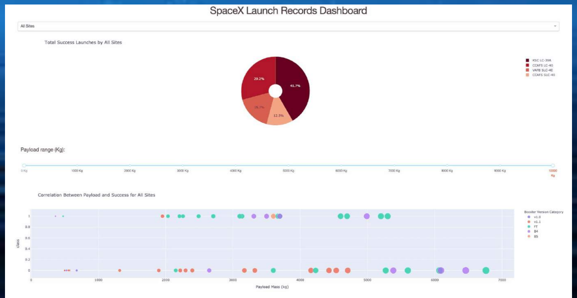
- **Summary of methodologies-**
  - Data Collection via API,SQL and Web scraping
  - Data wrangling and Analysis
  - Interactive Maps with Folium
  - Predictive Analysis for each classification model

- **Summary of all result-**
  - Data Analysis along with Interactive Visualization
  - Best model for Predictive Analysis









INTRODUCTION

- **Project background and context:**

Here we will predict if the Falcon 9 first stage will land successfully.

SpaceX advertises Falcon 9rocket 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 you 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 Scraping 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 evaluate classification models

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.

1. Getting Data from API or Web Page

2. Make dataframe from it

3.Filter dataframe as per requirement

4.Export to flat file

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

Getting response from API → Converting Response to a .json file → Apply custom functions to clean data → Assign list to dictionary then create dataframe → Filter dataframe and export to flat file

```
# Use json_normalize meethod to convert the json result into a dataframe
jlist = requests.get(static_json_url).json()
df2 = pd.json_normalize(jlist)
df2.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

Data Collection – 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

Dataframe to .CSV

	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

Data Wrangling-Meaning & Basic Steps

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

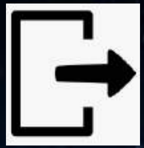
Here we mainly convert those outcomes into Training Labels with 1 means the booster successfully ;landed 0 means it was unsuccessful

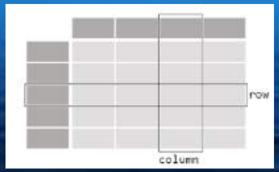
```
for outcome in df['Outcome']:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

1. Load Data

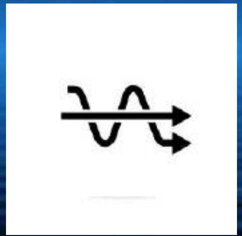
3. Cleaning Data

5. Export to flat file





2. Make dataframe from it



4. Simplifying it to boolean values

Data Wrangling

Calculate number of launches at each site

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

Calculate number and occurrence of each orbit

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

Calculate number and occurrence of mission outcome per orbit type

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

Create landing outcome label from Outcome column

```
landing_class =[]
for outcome in df['Outcome']:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
landing_class
```

Export dataset as .CSV

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

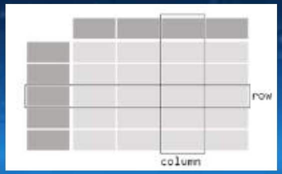
	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	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

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.

1. Load Data

2.Make dataframe from it



3.Create Visualizations

4. Collect Insights

EDA with Data Visualization

Scatter Graphs Drawn:

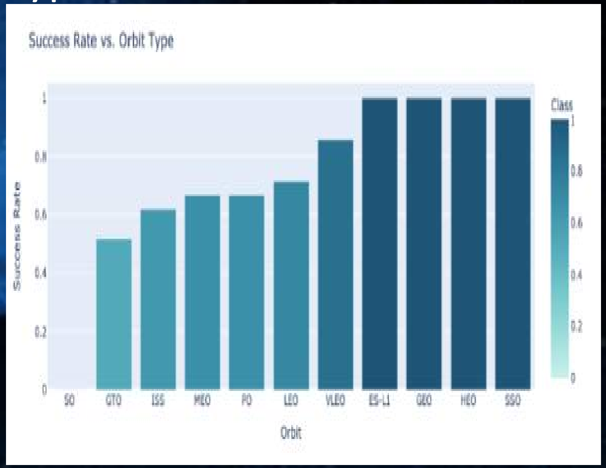
- Payload and Flight Number
- Flight Number and Launch Site
- Payload and Launch Site
- Flight Number and Orbit Type
- Payload and Orbit Type

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

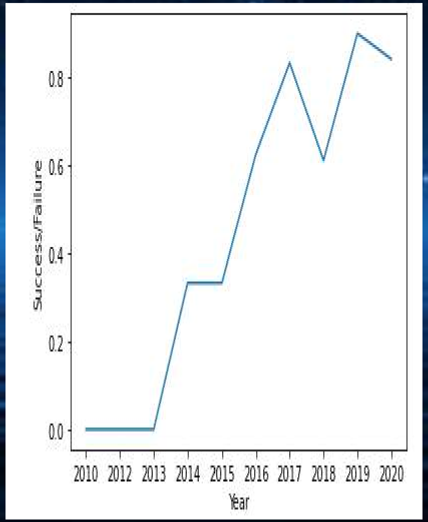
Bar Graphs are easiest to interpret a relationship between attributes. Via this bar graph we can easily determine which orbits have the highest probability of success



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.



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 manged SQL Database provided as a service

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 sited 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 F9v1.1
- 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.

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.

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
Dash and its components	Import dash Import dash_html_components as html Import dash_core_components as dcc From dash.dependencies import Input,Output	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 availables HTML tags as user-friendly Python classes
Pandas	Import pandas as pd	Fetching value from CSV and creating a dataframe
Plotly	Import plotly.express as px	Plot the graphs with interactive plotly library
Dropdown	dcc.Dropdown(	Create a dropdown for launch sites
Rangeslider	dcc.RangeSlider(	Create a rangeslider for Payload Mass range selection
Pie Chart	px.pie(	Creating a pie graph for success percentage display
Scatter Chart	px.scatter(	Creating the Scatter graph for correlation display

Predictive Analysis (Classification)

Building Model

- Load our feature engineered data into ataframe
- Tansform it into NumPy arrays
- Standardize and transform data
- Split data into training and testdata 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=la
mbda x:algorithms[x])
```

Finding Best performing classification model

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

```
Yhat=algorithm.predict(X_test)
Plot_confusion_matrix[Y_test,yhat]
```

Evaluating Model

- Check accuracy for each model
- Get best hyperparameters for each type
- Plot Confusion matrix

Best model

Results

Exploratory data analysis results

Interactive analytics demo in screenshots

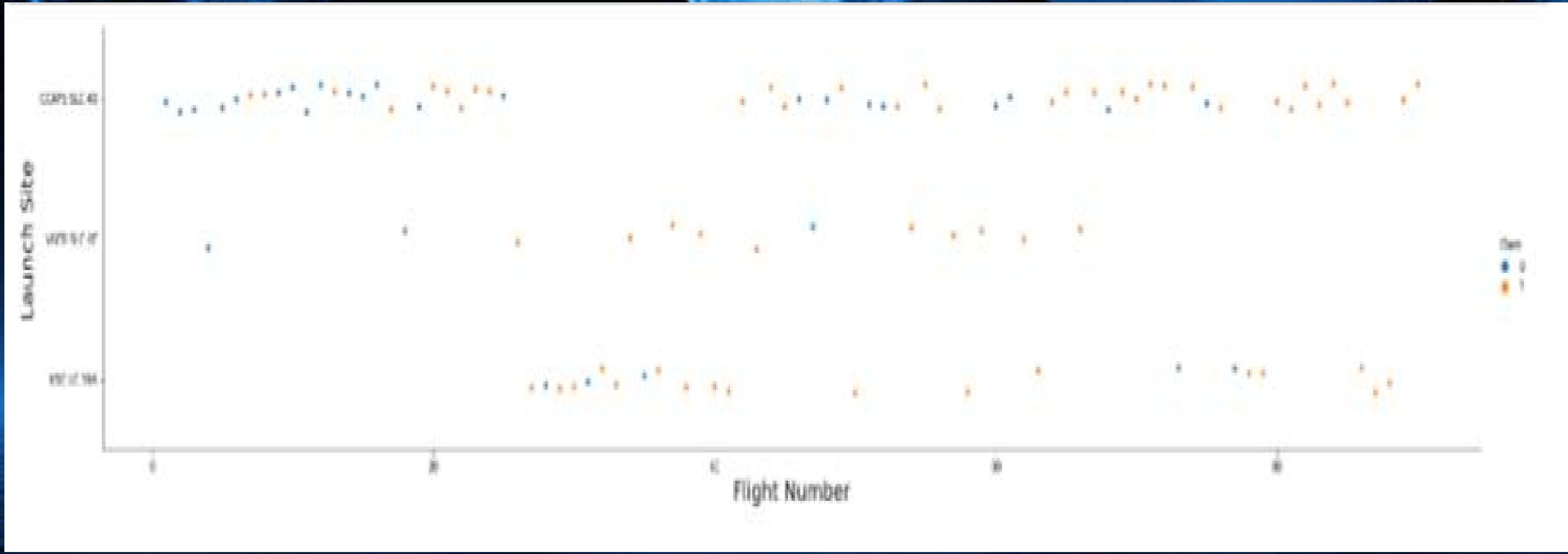
Predictive analysis results

EDA with Visualization

Flight Number vs. Launch Site

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

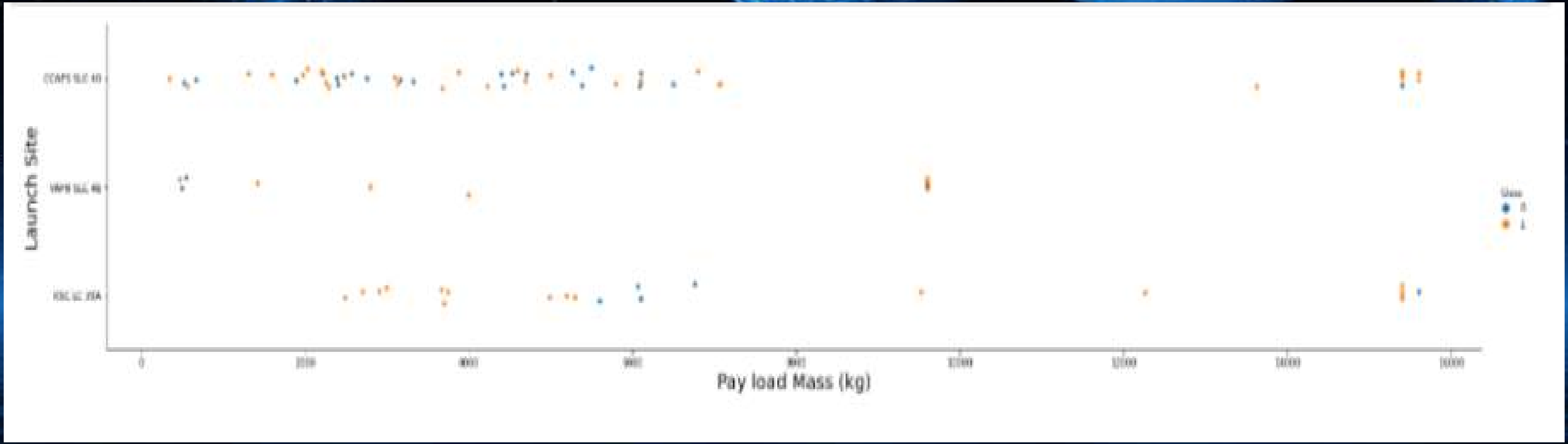
Flight Number vs. Launch Site



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.

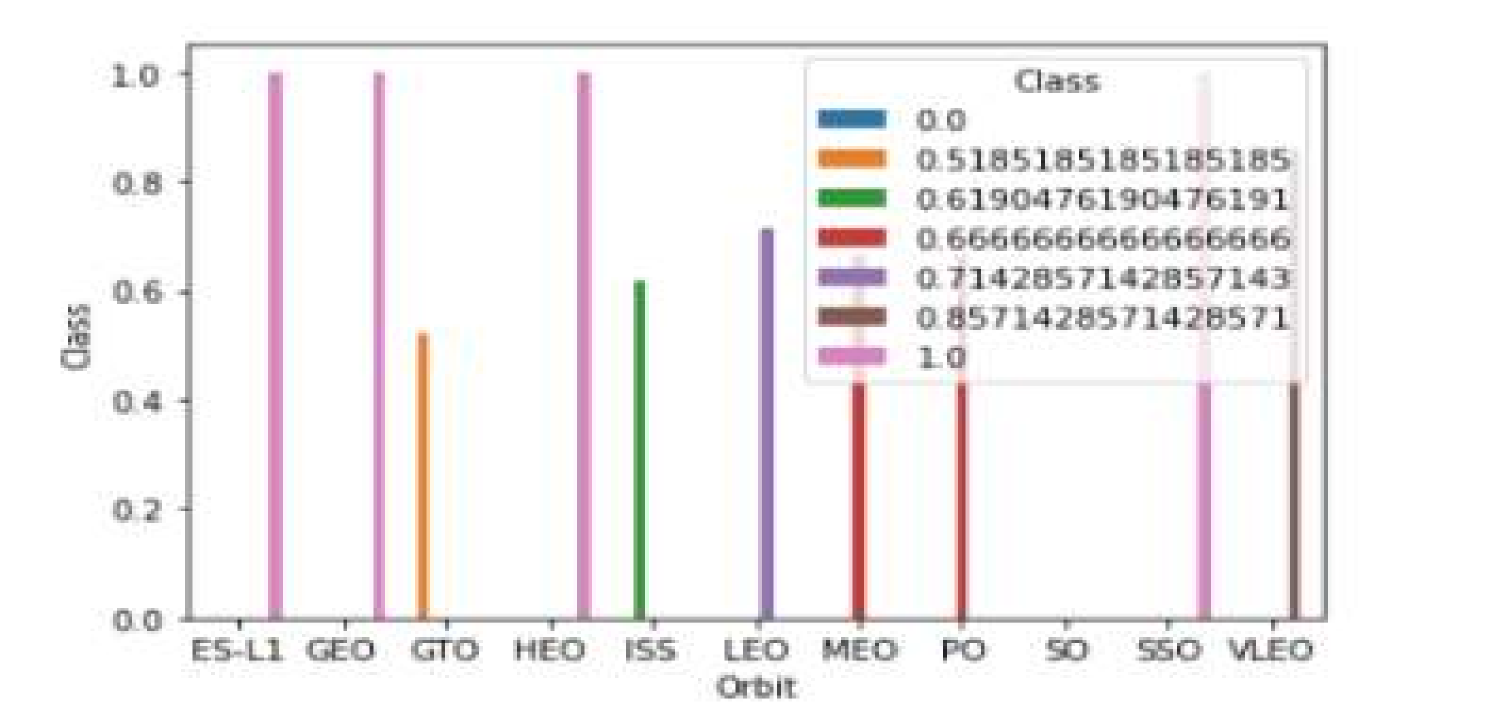
Payload Vs. Launch Site



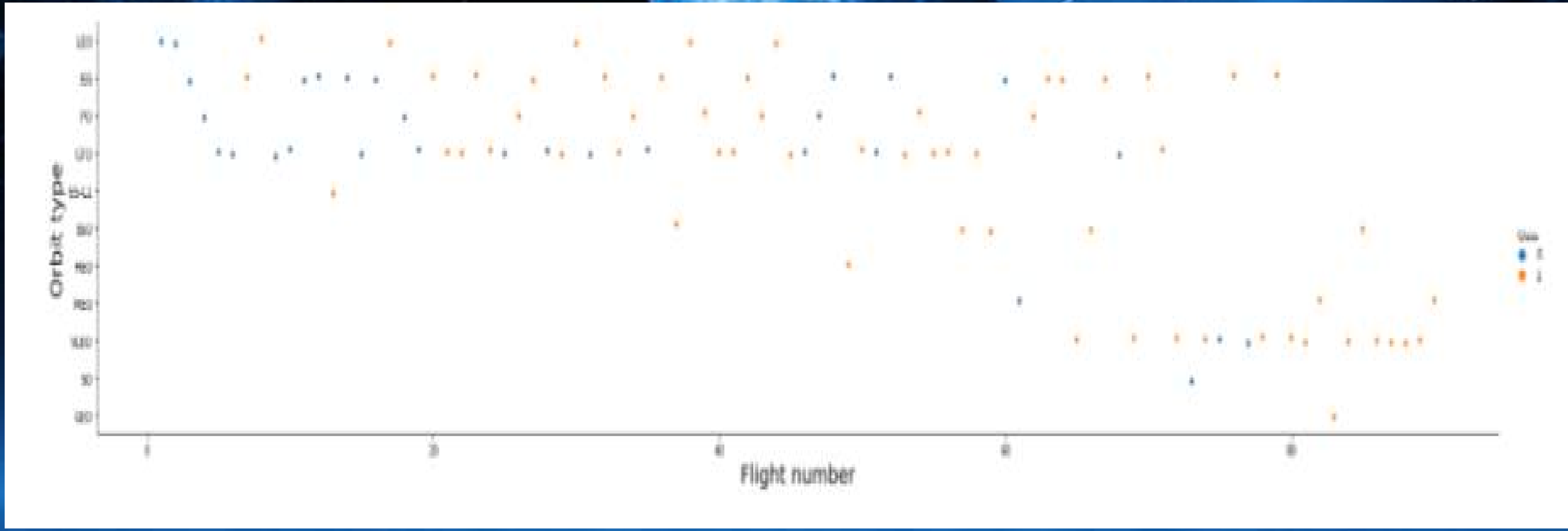
Success Rate vs. Orbit Type

- ES-L1,GEO,HEO,SSO has the highest success rates.

Success Rate Vs. Orbit Type



Flight Number vs. Orbit Type



EDA with SQL

All Launch Site Names

SQL Query

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

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

Description

Using the word DISTINCT in the query we pull unique values for Launch_Site column from table SPACEX

Launch Site Names Begin with 'CCA'

SQL Query

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

Description

Using keyword ‘LIMIT 5’ in the query 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 SPACEXTBL 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 SPACEXTBL \
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 Succesful Landing Outcome in Ground Pad" FROM SPACEXTBL \
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 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 SPACEXTBL 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

Total Number of Successful and Failure Mission Outcomes

SQL Query

```
%sql SELECT COUNT(MISSION_OUTCOME) AS "Successful Mission" FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE 'Success%';
```

Description

Selecting multiple count is a complex query. I have used Count clause on MISSION_OUTCOME named as “Successful Mission” from table spacex with condition LIKE keyword with wild card – ‘Success%’ .

Successful Mission
100

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 BOOSTER_VERSION, LAUNCH_SITE FROM SPACEX WHERE DATE LIKE '2015-%' AND \
LANDING__OUTCOME = 'Failure (drone ship)';
```

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

DESCRIPTION

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

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 ;
```

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

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

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

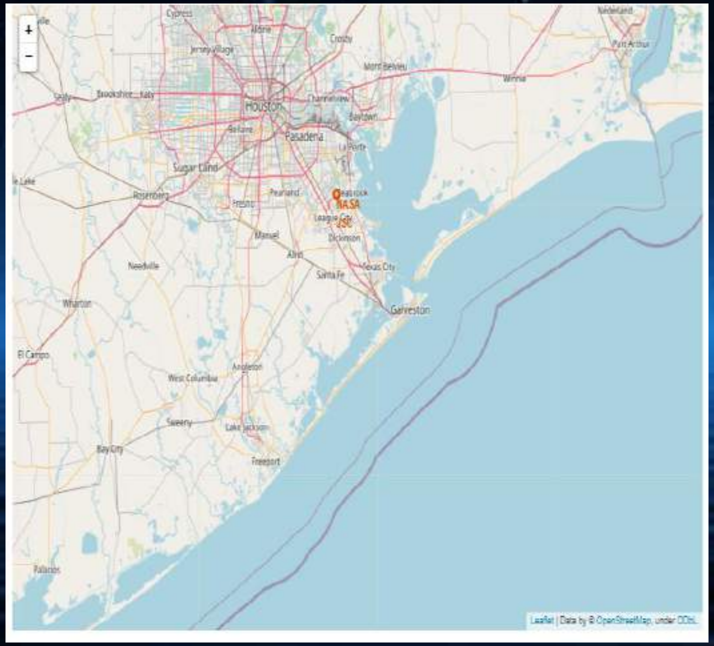
COUNT counts records in column LANDING_OUTCOME
WHERE filters data with '%Success%'
AND DATE>'2010-06-04'
AND DATE<'2017-03-20'

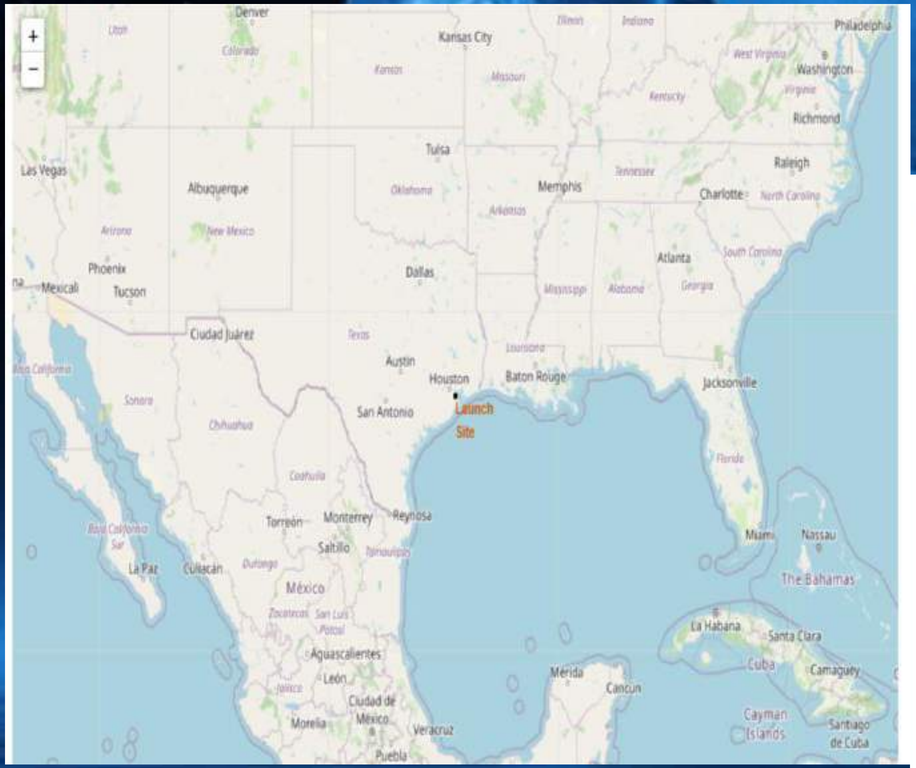
Rank success count between 2010-06-04 and 2017-03-20
8

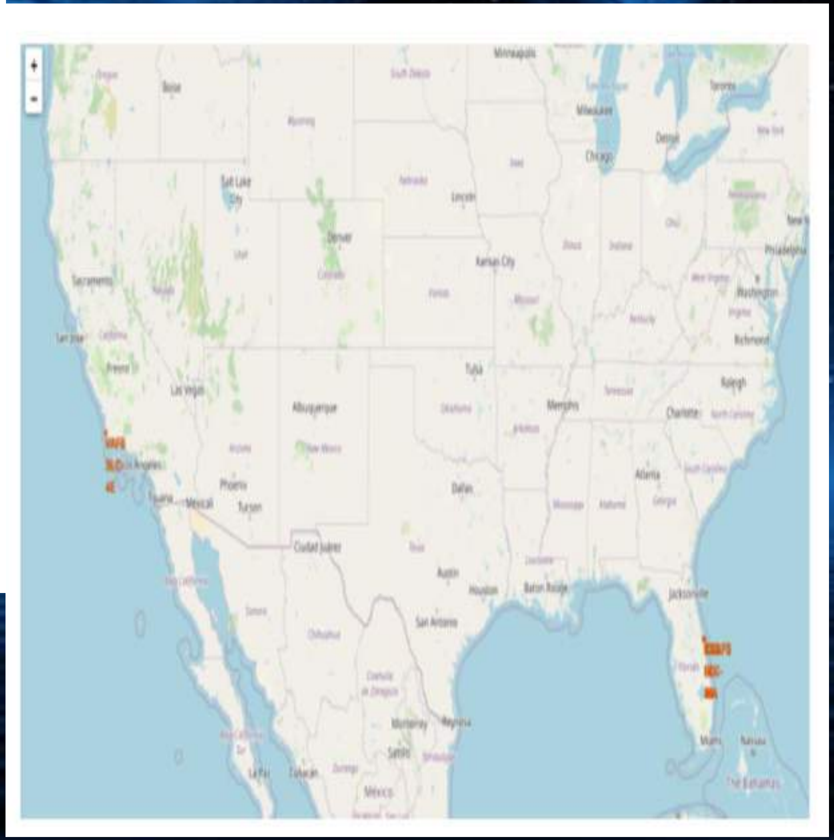
Interactive Map with Folium

All Launch Sites on Folium Map

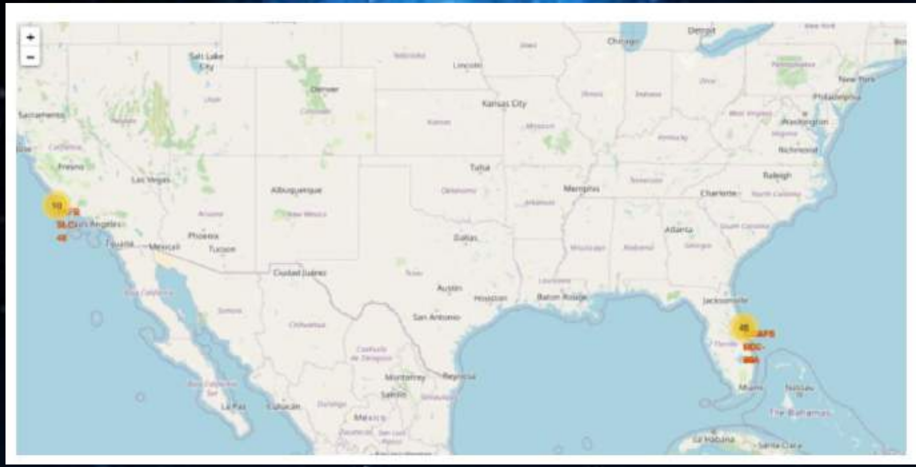
We can see that the SpaceX launch sites are near to the United States of America coasts i.e., Florida and California Regions.







COLOUR LABELED LAUNCH RECORDS



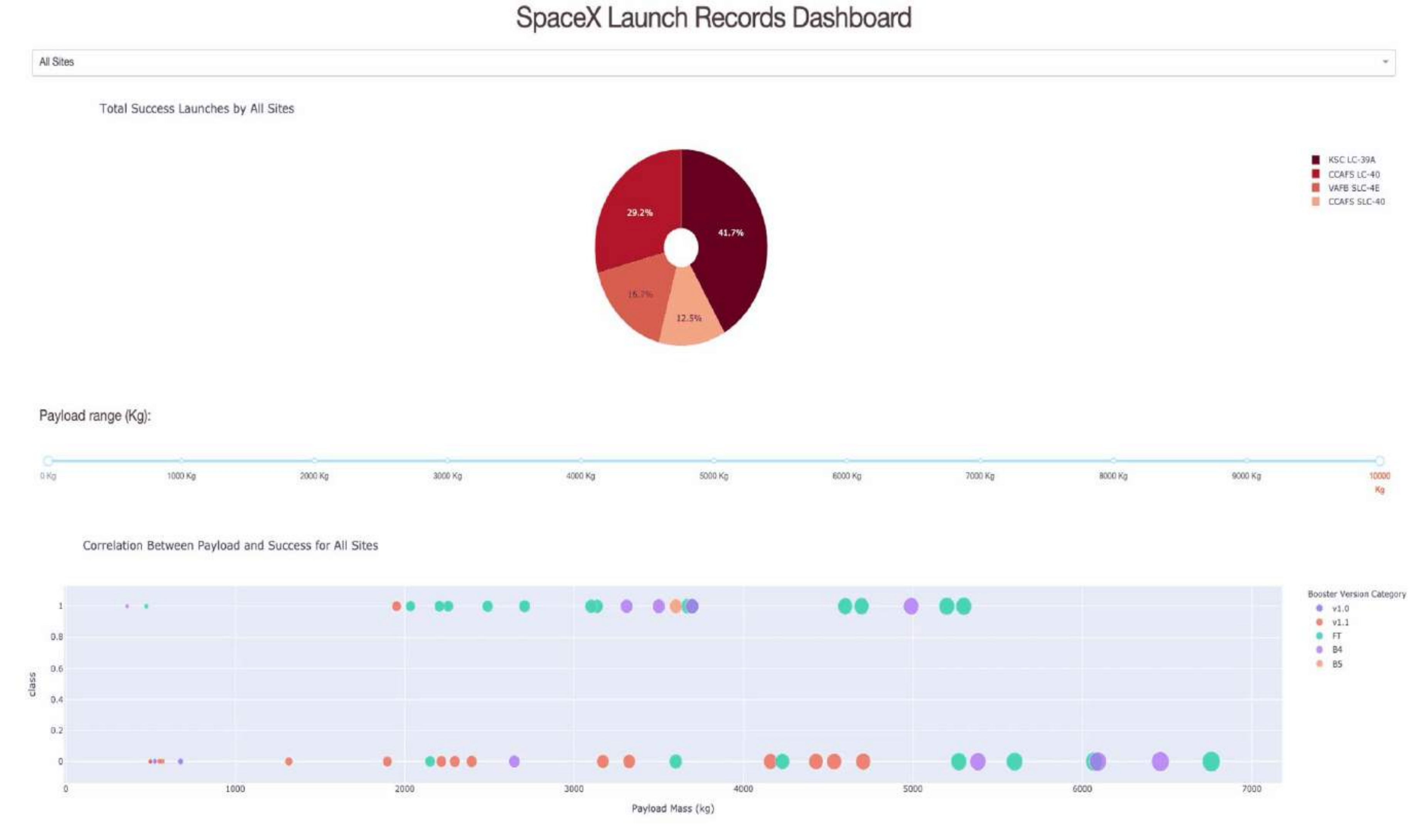
Green Marker shows successful launches and
Red Marker shows failures

Launch Site Distances from Equator & Railways

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

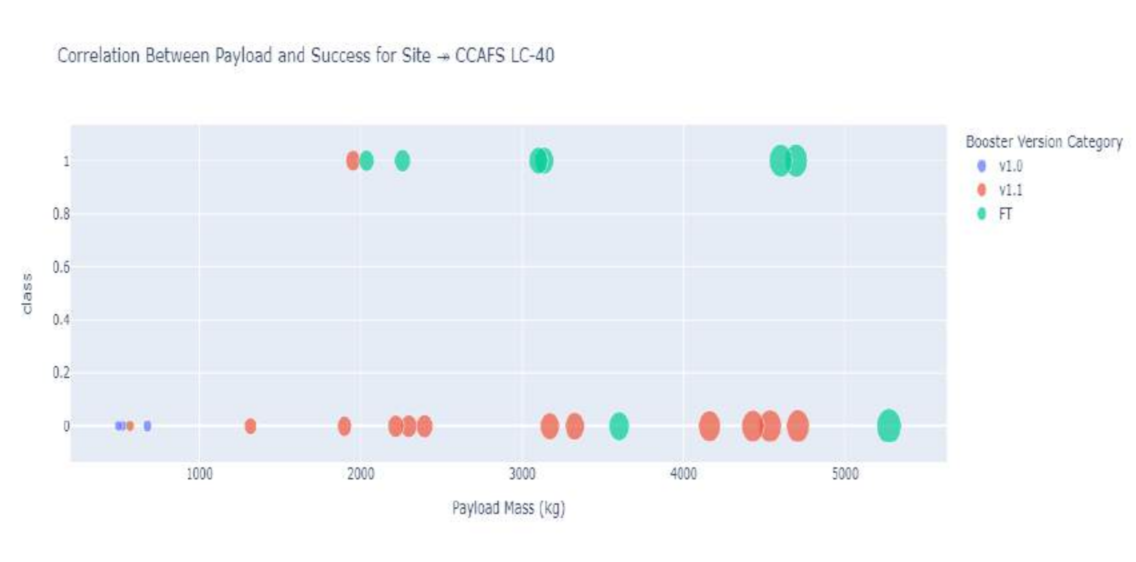
Build a Dashboard with Plotly Dash

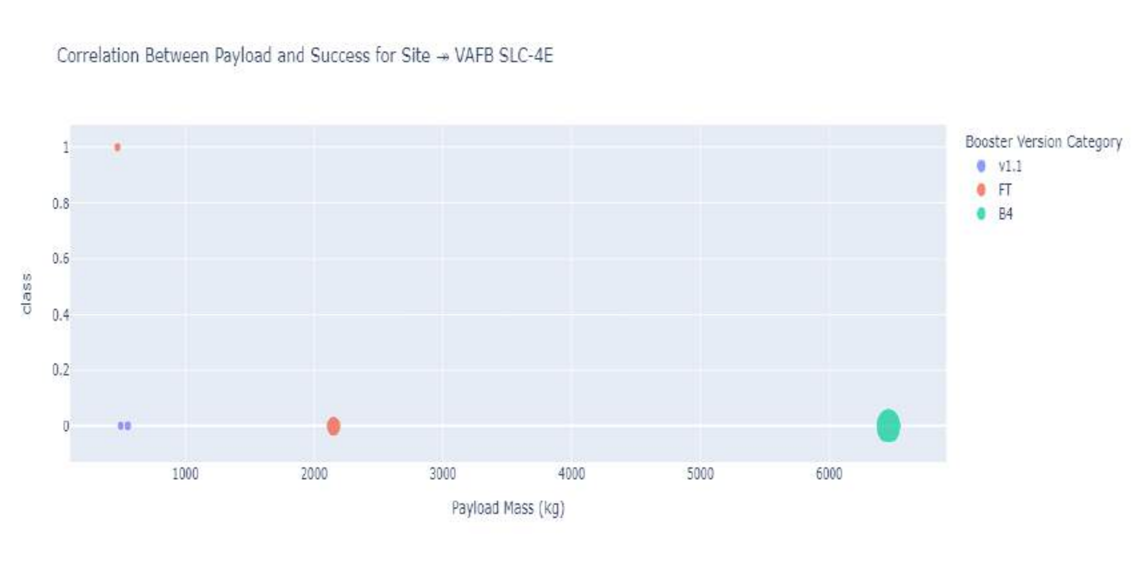
Launch Success Count for All Sites

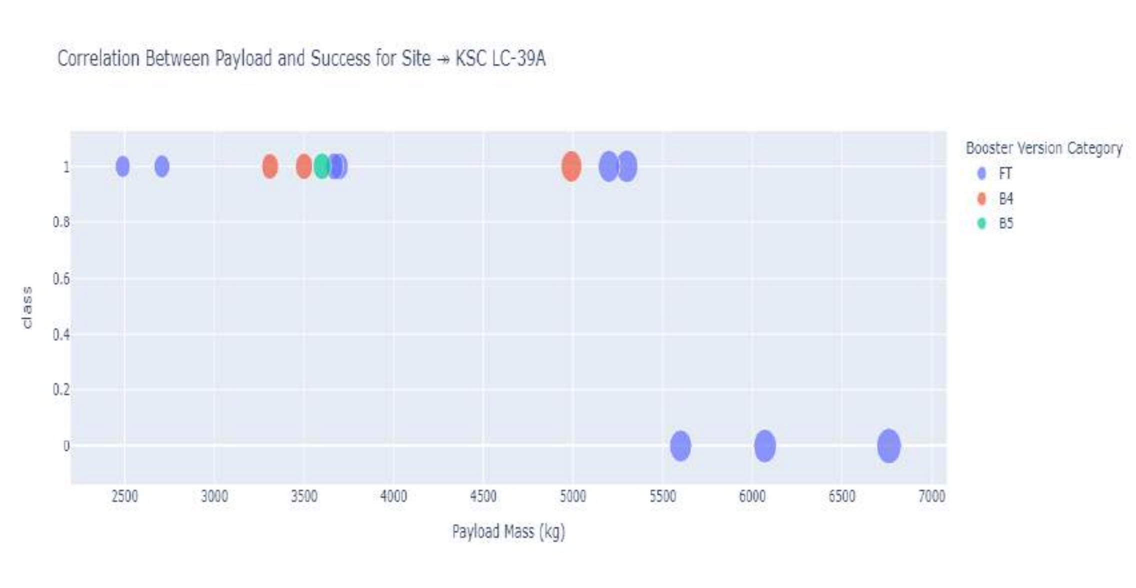


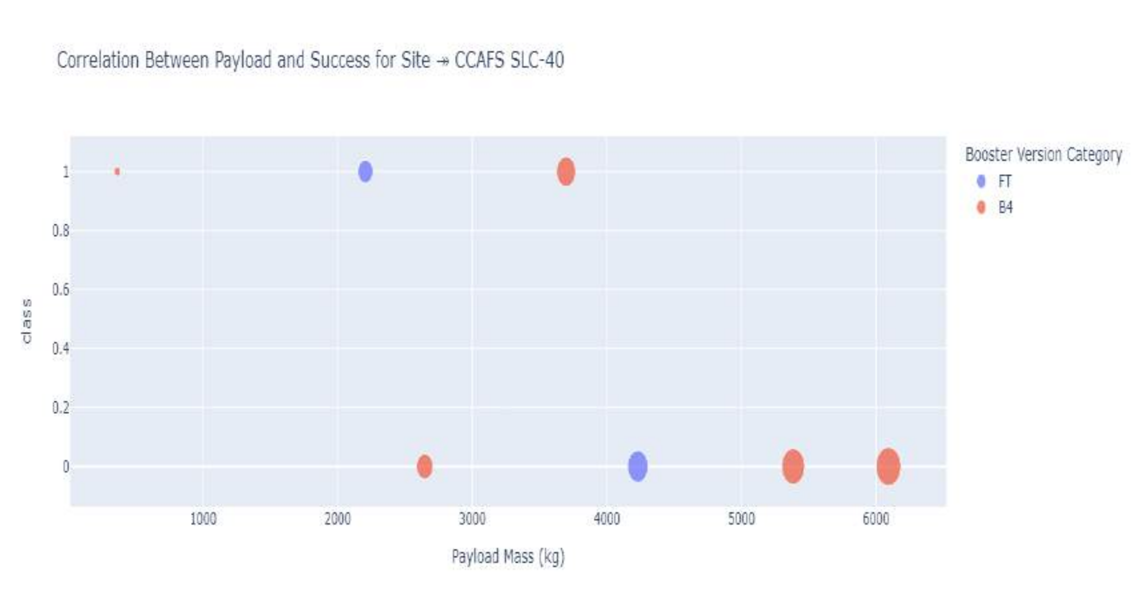
We can see that KSC LC-39A had the most successful launches from all the sites

Payload vs Launch Outcomes Scatter Plot for all sites

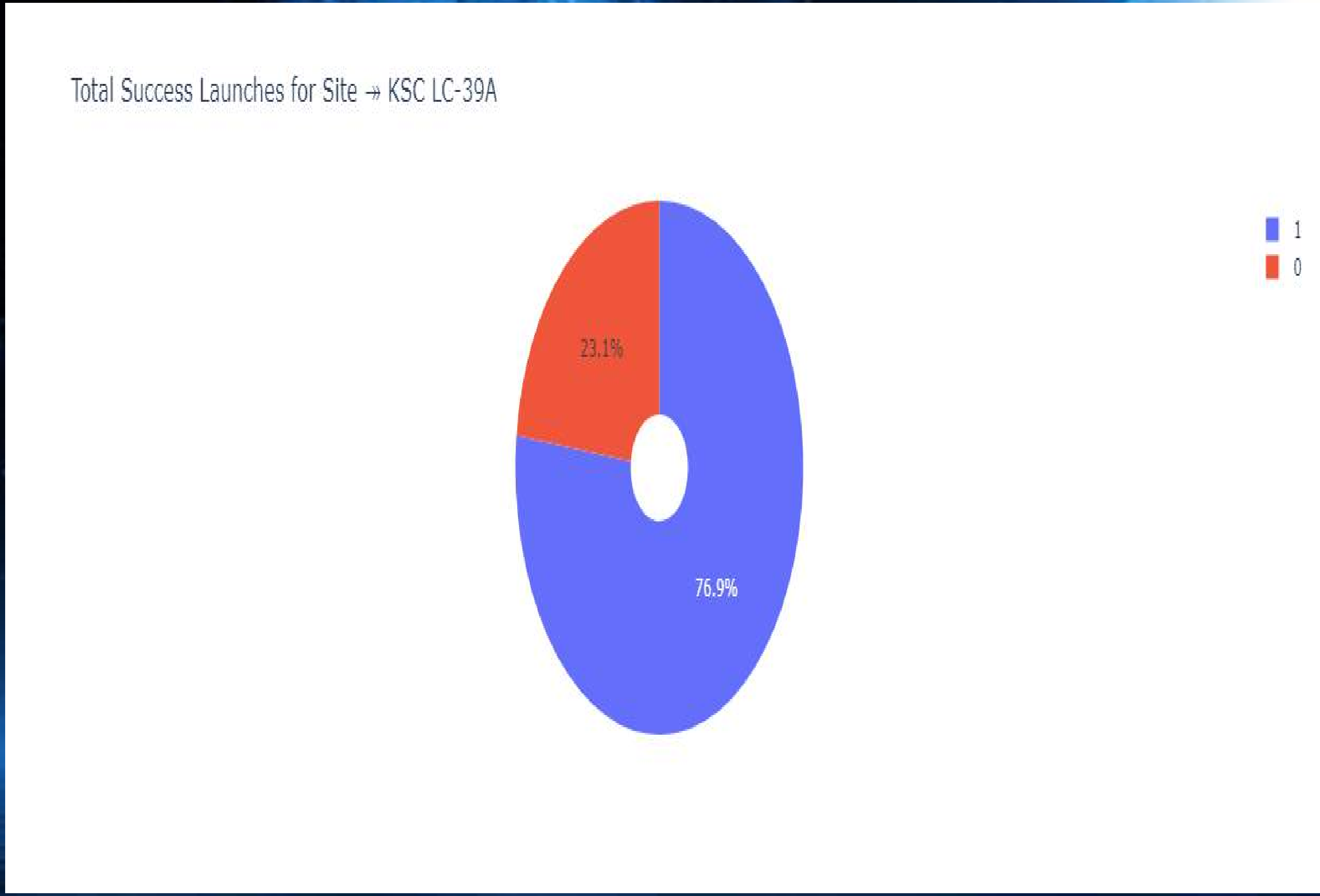








Launch Site with Highest Launch Success Ratio



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

After visual analysis using the dashboard,we are ablle 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

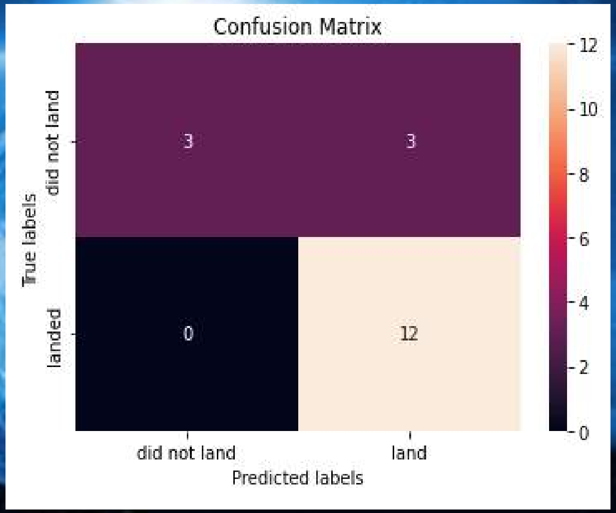
Predictive Analysis(Classification)

Confusion Matrix

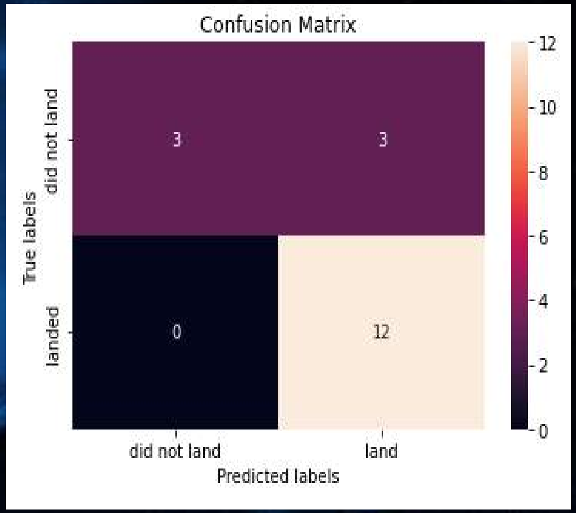
Out here for all models unfortunately, we have same confusion matrix.

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

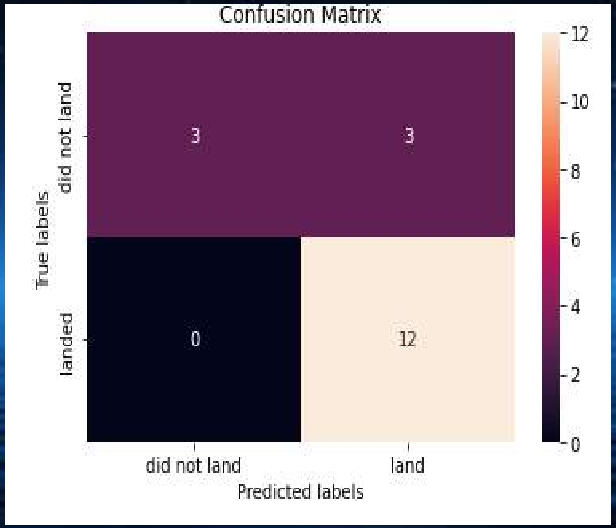
Logistic Regression



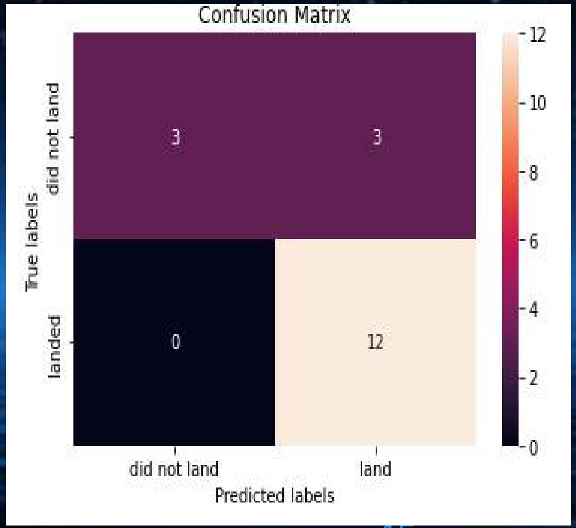
SVM



Decision Tree



KNN



Confusion Matrix

Accuracy: (TP+TN)/Total = (12+3)/18=0.83333

Misclassification Rate: (FP+FN)/Total=(3+0)/18=0.1667

True Positive Rate: TP/Actual yes=12/12=1

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

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

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

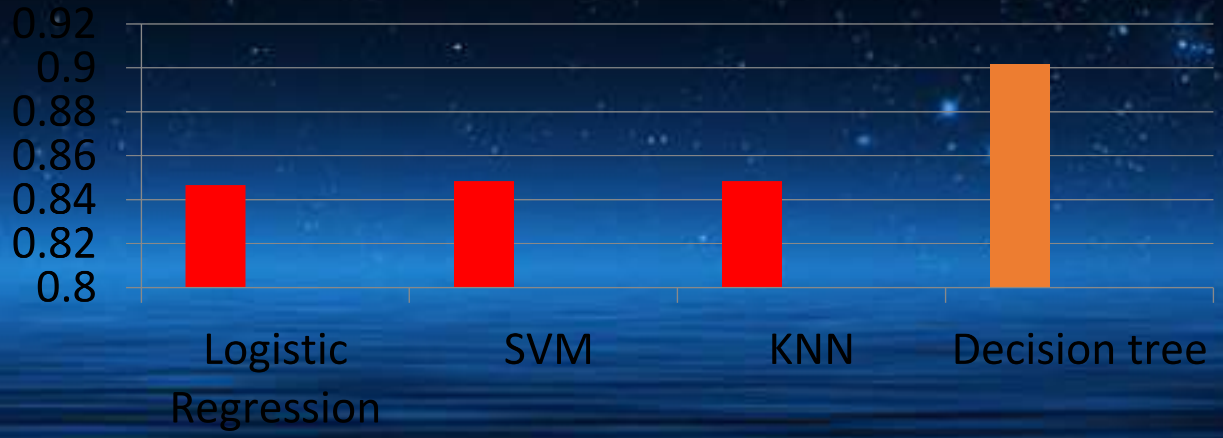
Prevalence: Actual yes/Total=12/18=0.6667

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
Logistic regression	0.846429	0.833334
SVM	0.848214	0.833334
KNN	0.848214	0.833334
Decision tree	0.901786	0.833334

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



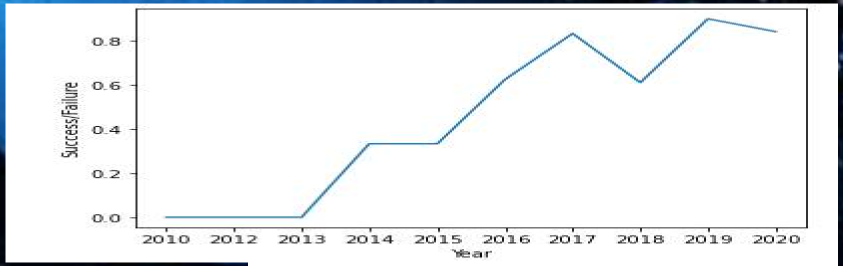
Conclusions

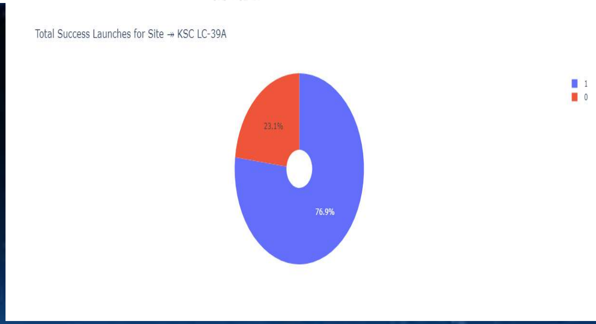
- ❑ Orbits ES-L1,GEO,HEO,SSO has the highest Success rates

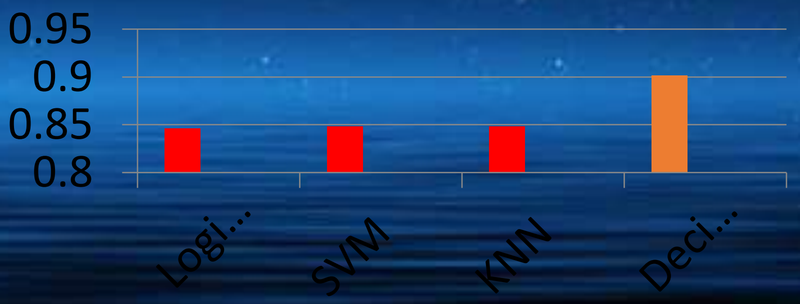
- ❑ Success rates for SpaceX launches has been increasing relatively with time and it looks like soon they will reach the required target

- ❑ KSC LC-39A had the most successful launches but increasing payload mass seems to have negative impact on success

- ❑ Decision Tree Classifier Algorithm is the best for Machine Learning Model for provided dataset





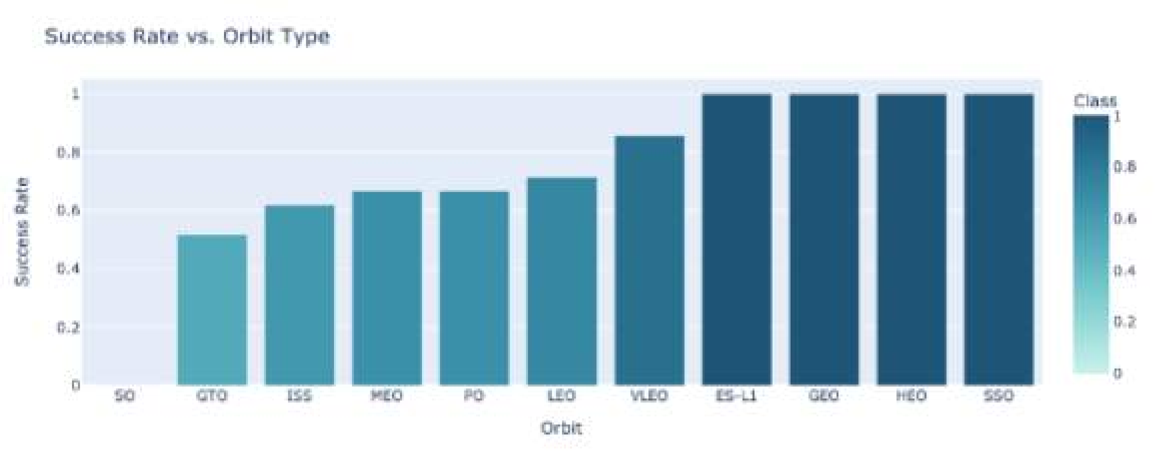


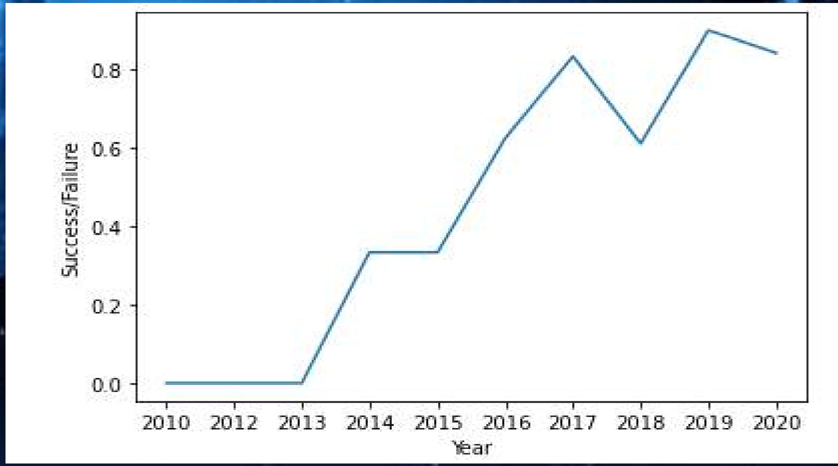
Appendix

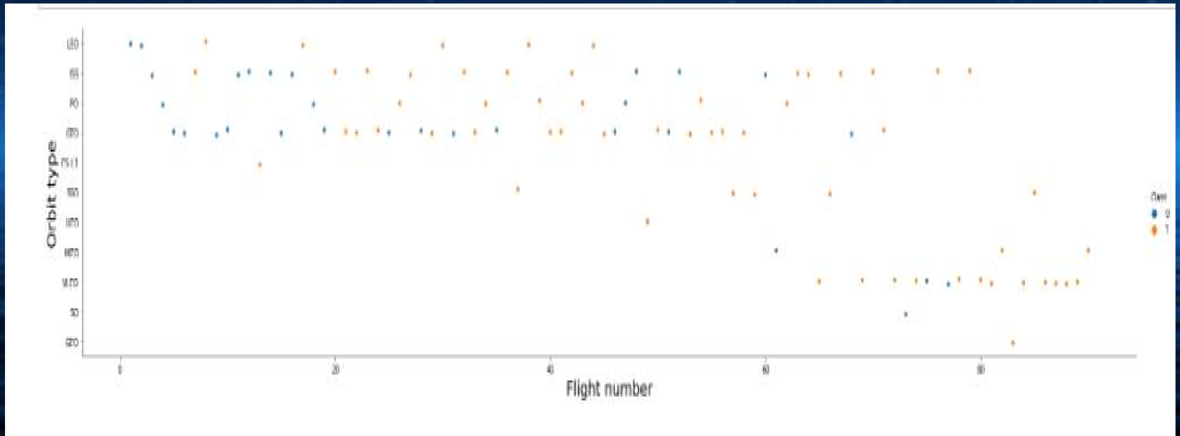
- Interactive Plotly
- “Python Anywhere” Live Site for plotly dashboard
- Folium MeasureControl Plugin Tool

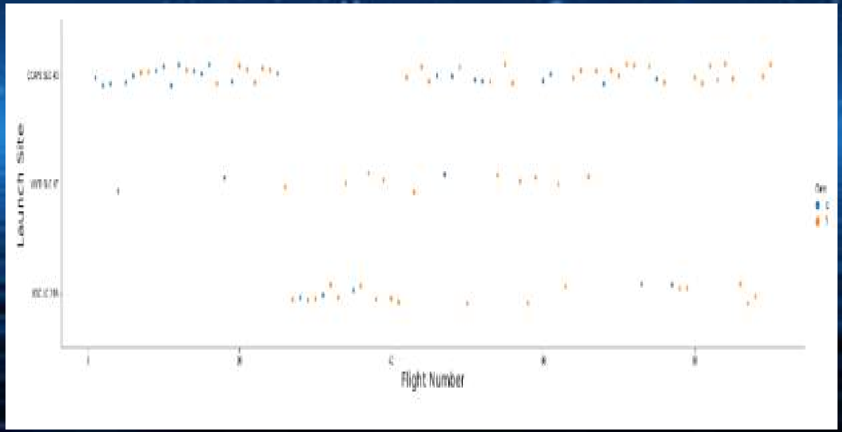
Interactive Plotly

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

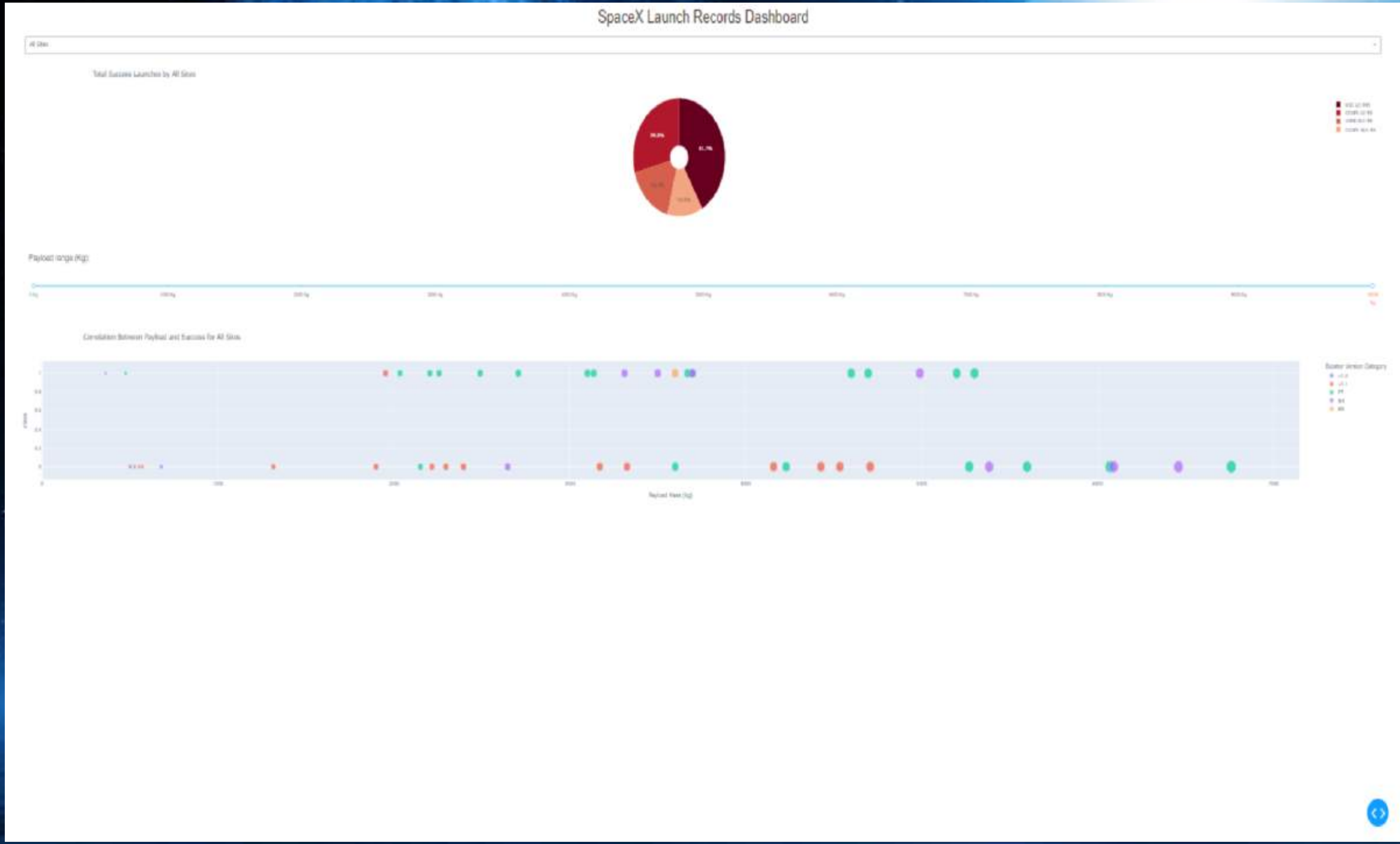








“Python Anywhere” Live Site for plotly dashboard



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

Folium MeasureControl Plugin Tool

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