IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Marcelino Tenorio Beato

26th October 2021

Outline

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

Executive Summary

Summary of methodologies

- Data Collection
- Data Wrangling
- EDA with data visualization
- EDA with SQL
- Building and interactive map with Folium
- Building Dashboard with Plotly Dash
- Predictive Analysis

Summary of all results

- Exploratory Data Analysis Results
- Interactive Analytics Demo with screenshots
- Predictive Analysis Results

Introduction

Project background and context

The commercial space age is here, companies are making space travel affordable for everyone. Virgin Galactic is providing suborbital spaceflights.

Rocket Lab is a small satellite provider. Blue Origin manufactures sub-orbital and orbital reusable rockets. Perhaps the most successful is SpaceX.

SpaceX's accomplishments include Sending spacecraft to the International Space Station. Starlink, a satellite internet constellation providing satellite Internet access.

Sending manned missions to Space. One reason SpaceX can do this is the rocket launches are relatively inexpensive. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upwards of 165 million dollars each, much of the savings is because

SpaceX can reuse the first stage.

Problems you want to find answers

- To determine if the first stage will land, we can determine the cos of a launch.
- To determine the price of each launch.
- To determine if SpaceX will reuse the first stage.
- Creating dashboards for your team

Section 1

Methodology

Methodology

Data collection methodology:

SpaceX Rest API

Web Scrapping

Perform data wrangling

-Transforming data for Machine Learning

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

Perform exploratory data analysis (EDA) using visualization and SQL

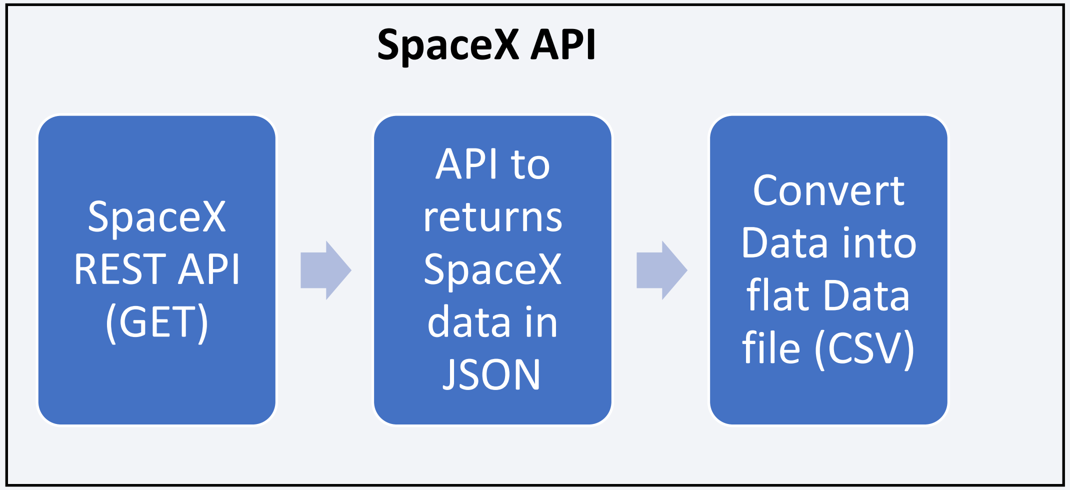
Perform interactive visual analytics using Folium and Plotly Dash

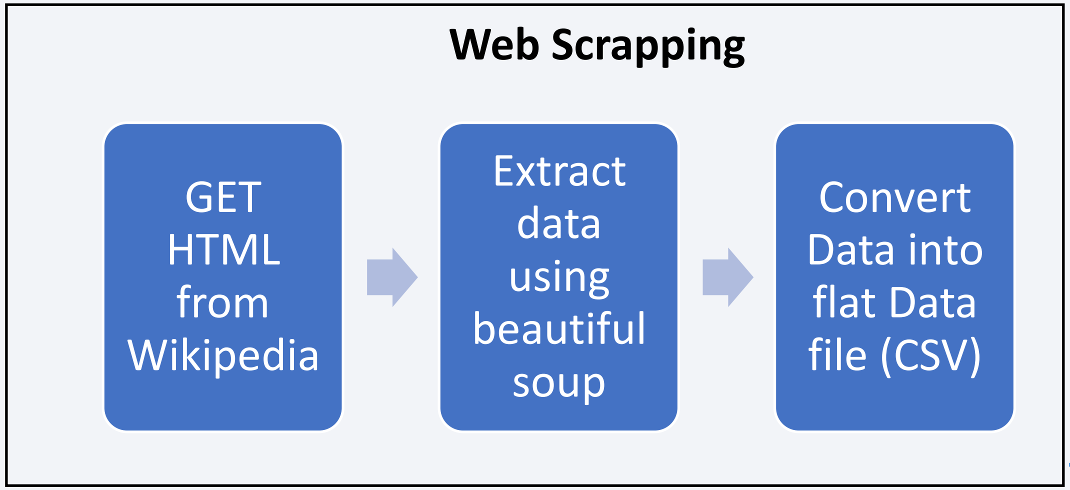
Perform predictive analysis using classification models

How to build, tune, evaluate classification models

Data Collection

- Data Collection Process
  - Gathering information about Space X.
  - Use SpaceX launch data that is gathered from the SpaceX REST API.
  - The API provides the data about launches, including information about the rocket used,payload delivered, launch specifications, landing specifications, and landing outcome.
  - The SpaceX REST API endpoints, or URL, starts with api.spacexdata.com/v4/.
  - Used Web scraping Wikipedia using BeautifulSoup for obtaining Falcon 9 Launch data.





Data Collection – SpaceX API

GitHub URL to Notebook

1. Getting Response from API
2. Converting Response to json file
3. Perform Data Cleansing
4. Assign the list to dictionary/data frame
5. Filter Data Frame and Export to CSV

1.
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url).json()
```

2.
```
response = requests.get(static_json_url).json()
data = pd.json_normalize(response)
```

3.
```
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
getBoosterVersion(data)
```

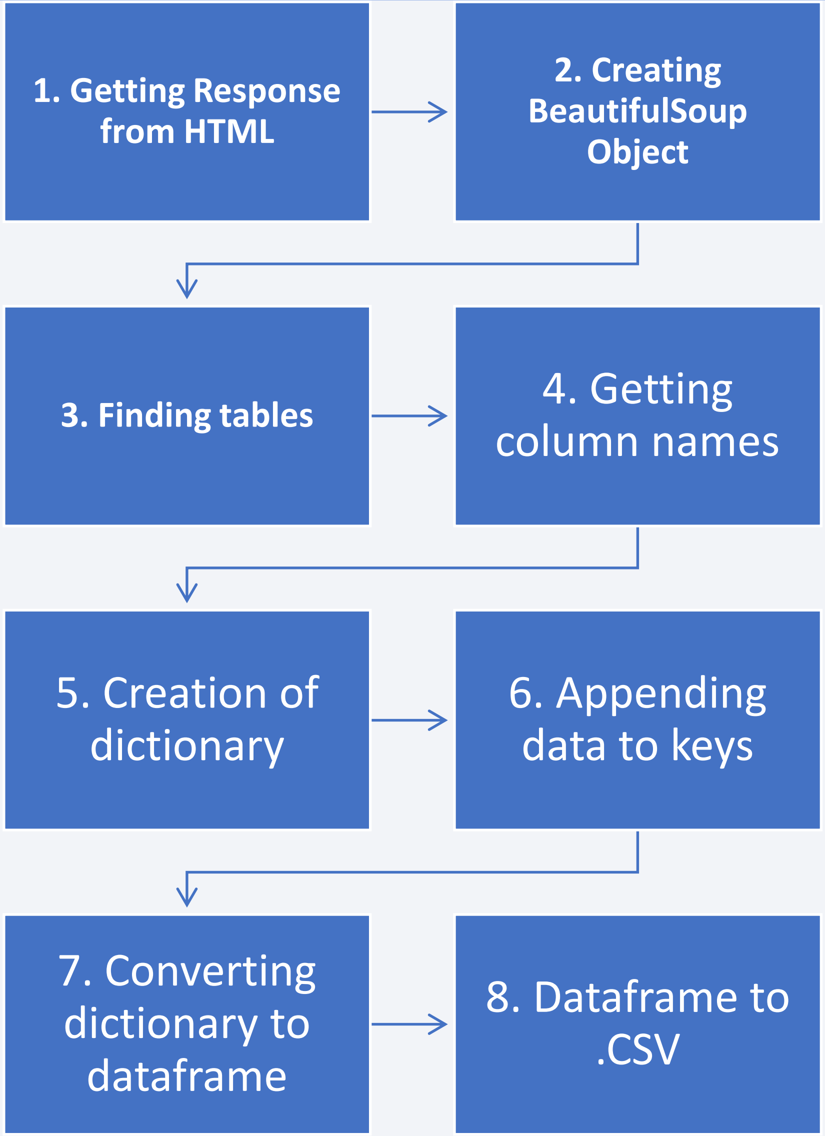
4.
```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

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

5.
```
data_falcon9 = df.loc[df['BoosterVersion']!="Falcon 1"]
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection - Scraping

GitHub URL to Notebook



1.
```
page = requests.get(static_url)
```

2.
```
soup = BeautifulSoup(page.text, 'html.parser')
```

3.
```
html_tables = soup.find_all('table')
```

4.
```
column_names = []
temp = soup.find_all('th')
for x in range(len(temp)):
    try:
     name = extract_column_from_header(temp[x])
     if (name is not None and len(name) > 0):
        column_names.append(name)
    except:
     pass
```

5.
```
launch_dict= dict.fromkeys(column_names)

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


launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

6.
```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value

            launch_dict["Flight No."].append(flight_number)

            datatimelist=date_time(row[0])
            # Date value

            date = datatimelist[0].strip(',')
            launch_dict["Date"].append(date)


            # Time value

            time = datatimelist[1]
            launch_dict["Time"].append(time)


            # Booster version

            bv=booster_version(row[1])
            if not(bv):
                bv=row[1].a.string
            launch_dict["Version Booster"].append(bv)


            # Launch Site

            launch_site = row[2].a.string
            launch_dict["Launch site"].append(launch_site)


            # Payload

            payload = row[3].a.string
            launch_dict["Payload"].append(payload)


            # Payload Mass
```

7.
```
df = pd.DataFrame.from_dict(launch_dict)
```

8.
```
df.to_csv('spacex_web_scraped.csv', index=False)
```

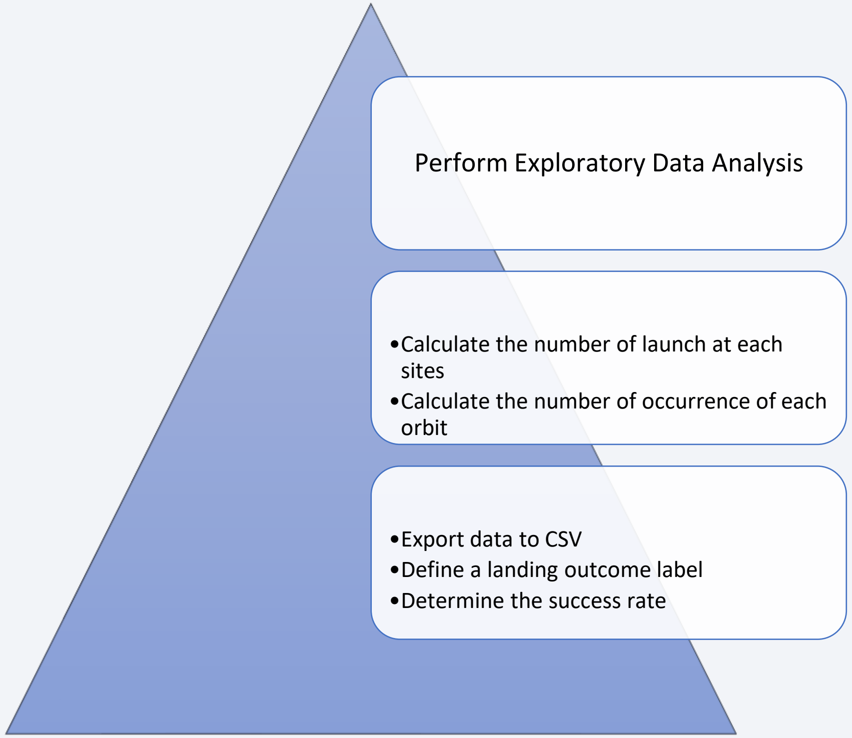
Data Wrangling

GitHub URL to Notebook

Definition

Data wrangling is the process of cleaning and unifying messy and complex data sets for easy access and analysis. This process typically includes manually converting and mapping data from one raw form into another format.
It helps data usability by transforming it to make it compatible with the end system as complex and intricate datasets can hinder data analysis and business processes. To make data usable for the end processes, data wrangling tools transform and organize data according to the target system's requirements.

Process



EDA with Data Visualization

GitHub URL to Notebook

Scatter Graphs:

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

Scatter plots' primary uses are to observe and show relationships between two numeric variables. The dots in a scatter plot not only report the values of individual data points, but also patterns when the data are taken as a whole. Scatter plots show how much one variable is affected by another. The relationship between two variables is called their correlation .Scatter plots usually used for a large body of data.

Bar Graph

- Mean VS Orbit

A Bar graphs are ideal for comparing two or more values, or values over time. Data is displayed either horizontally or vertically. Single bar graphs are used to convey discrete values of an item within a category. A bar diagram makes it easy to compare sets of data between different groups at a glance.

Line Graph

- Success Rate by Year

A line graph is usually used to show the change of information over a period of time. Line graphs are useful in that they show data variables and trends very clearly and can help to make predictions about the results of data not yet recorded.

EDA with SQL

GitHub URL to Notebook

Summary of SQL queries to gather, manipulate, process the required dataset.

- Displaying the names of the unique launch sites in the space mission
- Displaying 5 records where launch sites begin with the string 'KSC'
- Displaying the total payload mass carried by boosters launched by NASA (CRS)
- Displaying average payload mass carried by booster version F9 v1.1
- Listing the date where the successful landing outcome in drone ship was achieved.
- Listing the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000
- Listing the total number of successful and failure mission outcomes
- Listing the names of the booster_versions which have carried the maximum payload mass.
- Listing the records which will display the month names, successful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2015
- Ranking the count of successful landing outcomes between the date 2010-06-04 and 2017-03-20 in descending order.

Build an Interactive Map with Folium GitHub URL to Notebook

To visualize the Launch Data into an interactive map.

- ✓ Collect the Latitude and Longitude Coordinates at each launch site and added a Circle Marker around each launch site with a label of the name of the launch site.
- ✓ We assigned the dataframe launch_outcomes(failures, successes) to *classes 0 and 1* with Greenand Redmarkers on the map in a MarkerCluster()
- ✓ Using Haversine's formula to calculate the distance from the Launch Site to various landmarks to find various trends about what is around the Launch Site to measure patterns. Linesare drawn on the map to measure distance to landmarks.

Build a Dashboard with Plotly Dash

GitHub URL to Notebook

PythonAnywhere is first go-to place whenever or wherever is the team. Python Anywhere is to host the website live 24/7, it is helpful to the team specially to management to view the performance in real time.

The dashboard is built with Flask and Dash web framework.

Graphs

- **Pie Chart showing the total launches by a certain site/all sites**

  - display relative proportions of multiple classes of data.
  - size of the circle can be made proportional to the total quantity it represents.

- **Scatter Graph showing the relationship with Outcome and Payload Mass (Kg) for the different Booster Versions**

  -It shows the relationship between two variables.
  -It is the best method to show you a non-linear pattern.
  -The range of data flow, i.e. maximum and minimum value, can be determined.
  -Observation and reading are straightforward.

GitHub URL to Notebook

Predictive Analysis (Classification)

- **BUILDING MODEL**
- Load our dataset into NumPy and Pandas
- Transform Data
- Split our data into training and test data sets
- Check how many test samples we have
- Decide which type of machine learning algorithms we want to use
- Set our parameters and algorithms to GridSearchCV
- Fit our datasets into the GridSearchCVobjects and train our dataset.

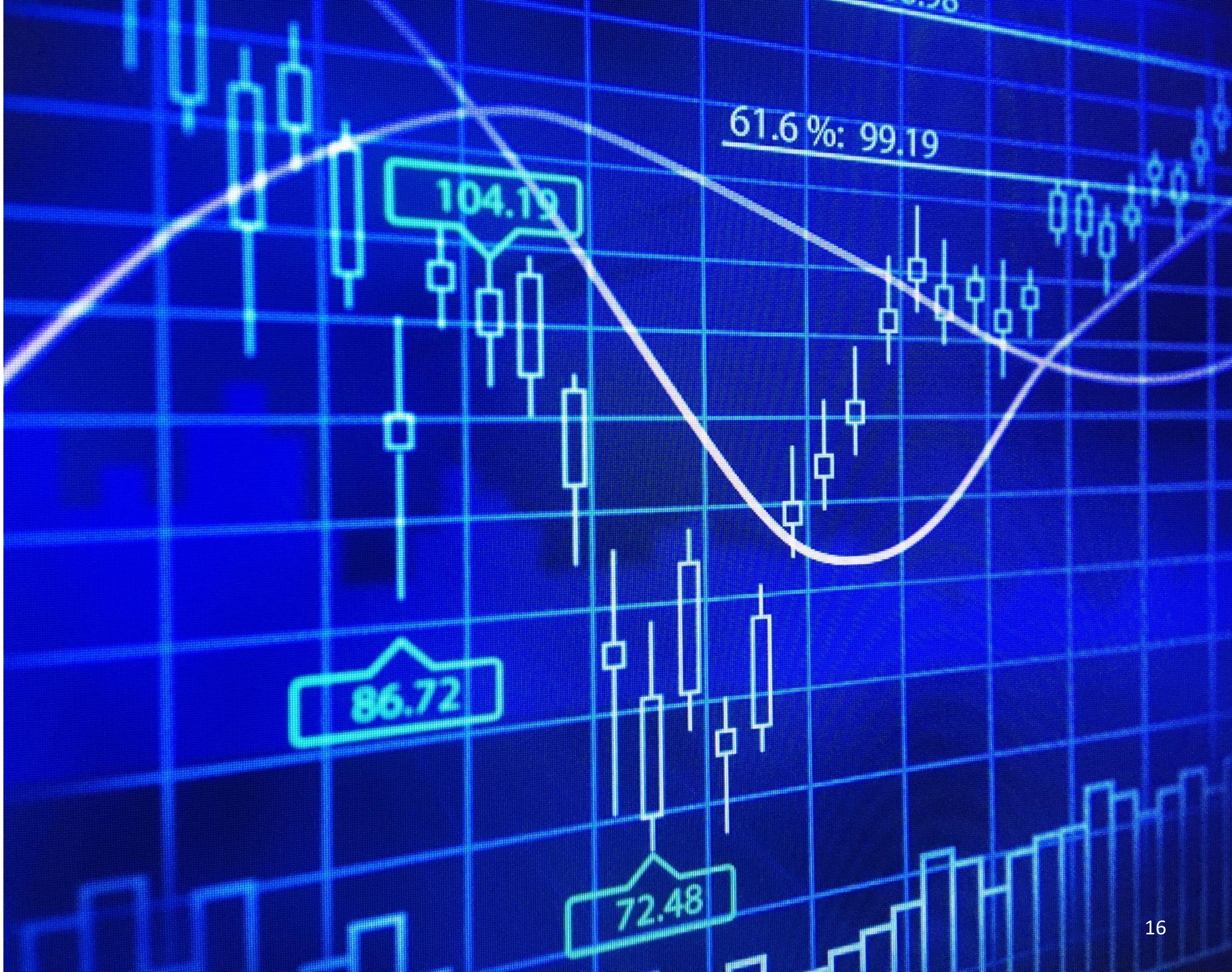
- **EVALUATING MODEL**
- Check accuracy for each model
- Get tuned hyperparameters for each type of algorithms
- Plot Confusion Matrix

- **IMPROVING MODEL**
- Feature Engineering
- Algorithm Tuning

- **FINDING THE BEST PERFORMING CLASSIFICATION MODEL**
- The model with the best accuracy score wins the best performing model
- In the notebook there is a dictionary of algorithms with scores at the bottom of the notebook.

Results

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

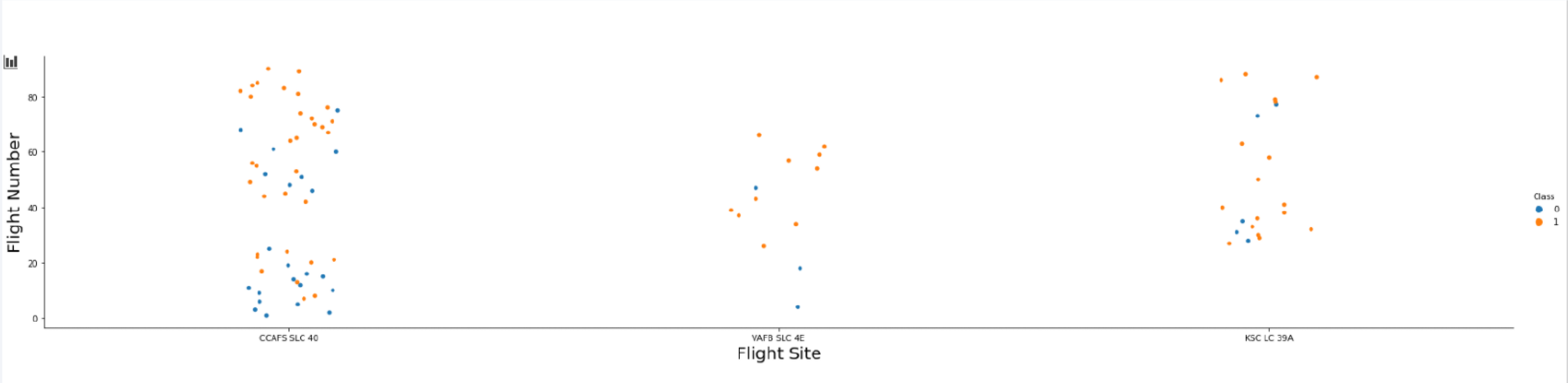


Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

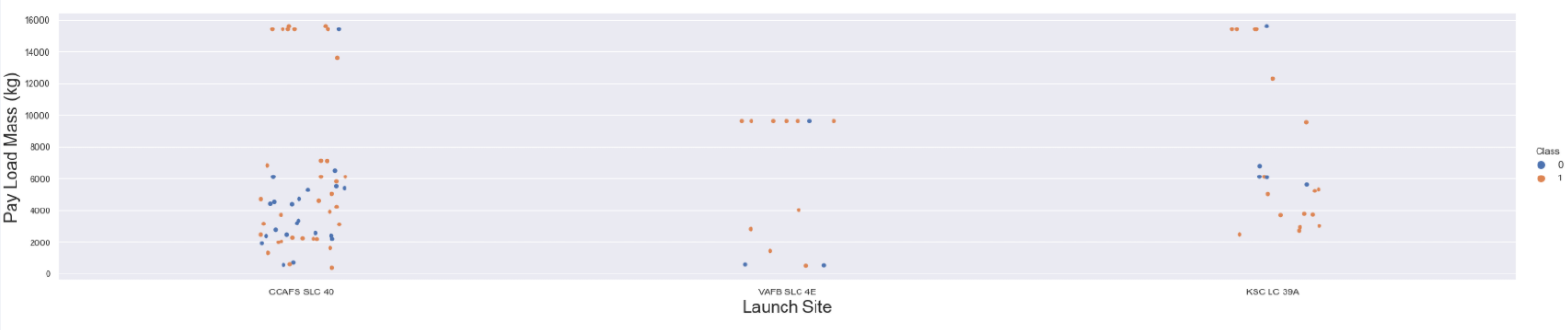
Flight Number vs. Launch Site



It shows in above graph, that the greater the success rate at a launch site based on number of flights.

Payload vs. Launch Site

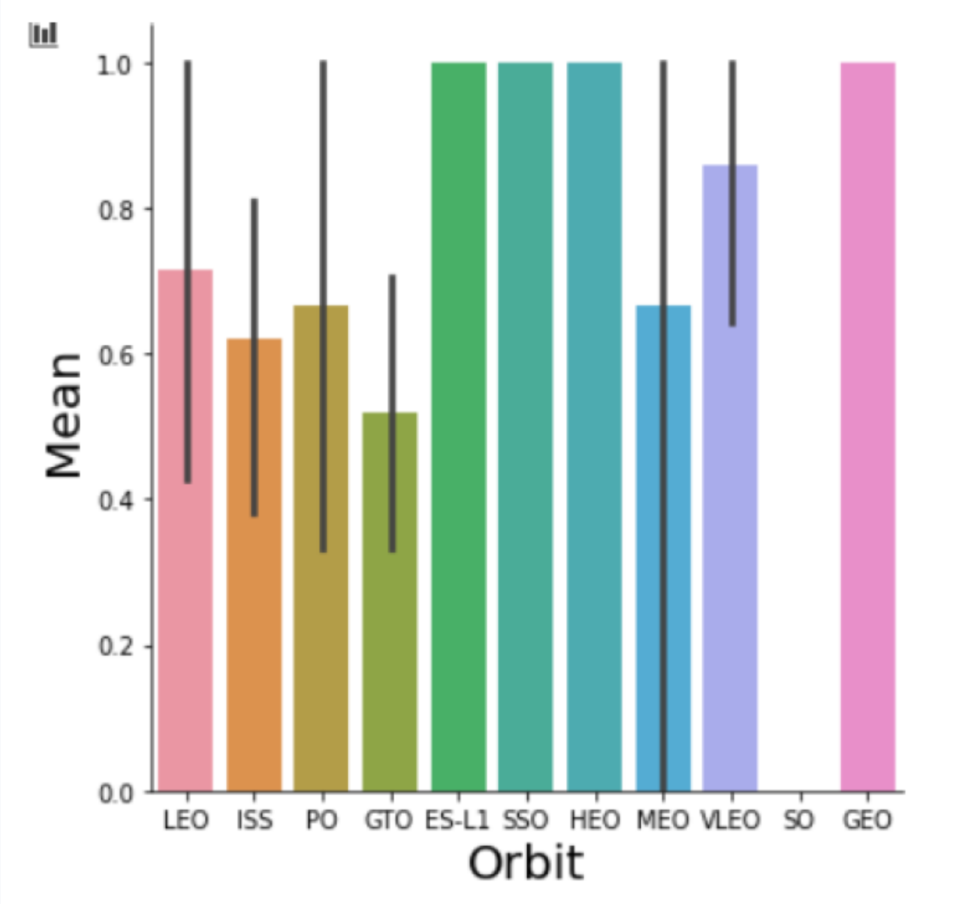
Payload vs. Launch Site



For Launch Site of CCAFS SLC 40, the greater the payload mass the higher success rate.

Success Rate vs. Orbit Type

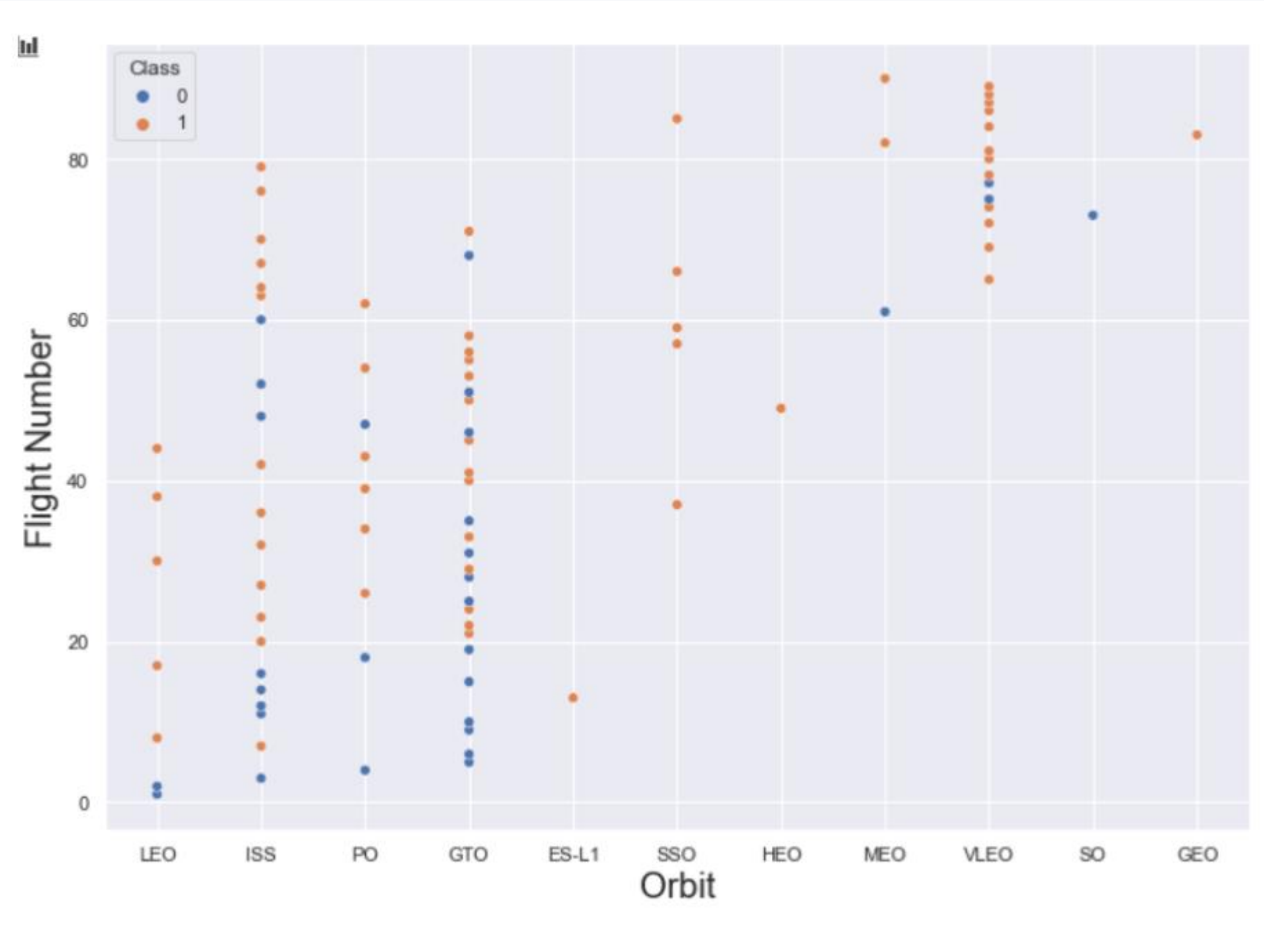
Success rate of each orbit type



The Orbit with highest success rate are GEO,HEO,SSO andES-L1.

Flight Number vs. Orbit Type

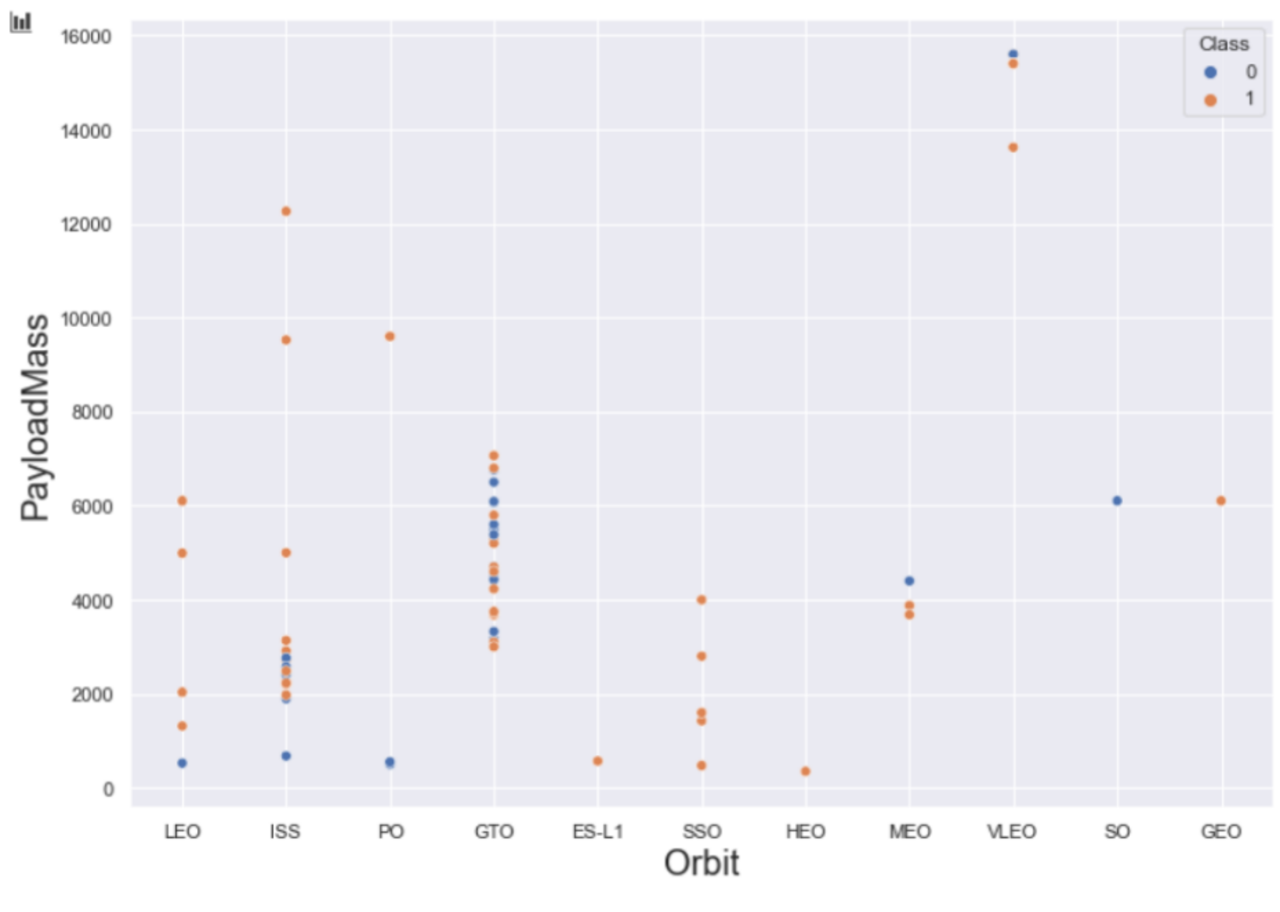
Flight number vs. Orbit type



The Success LEO orbit appears related to the number of flights and on the other hand, there is no relationship between flight number in GTO orbit.

Payload vs. Orbit Type

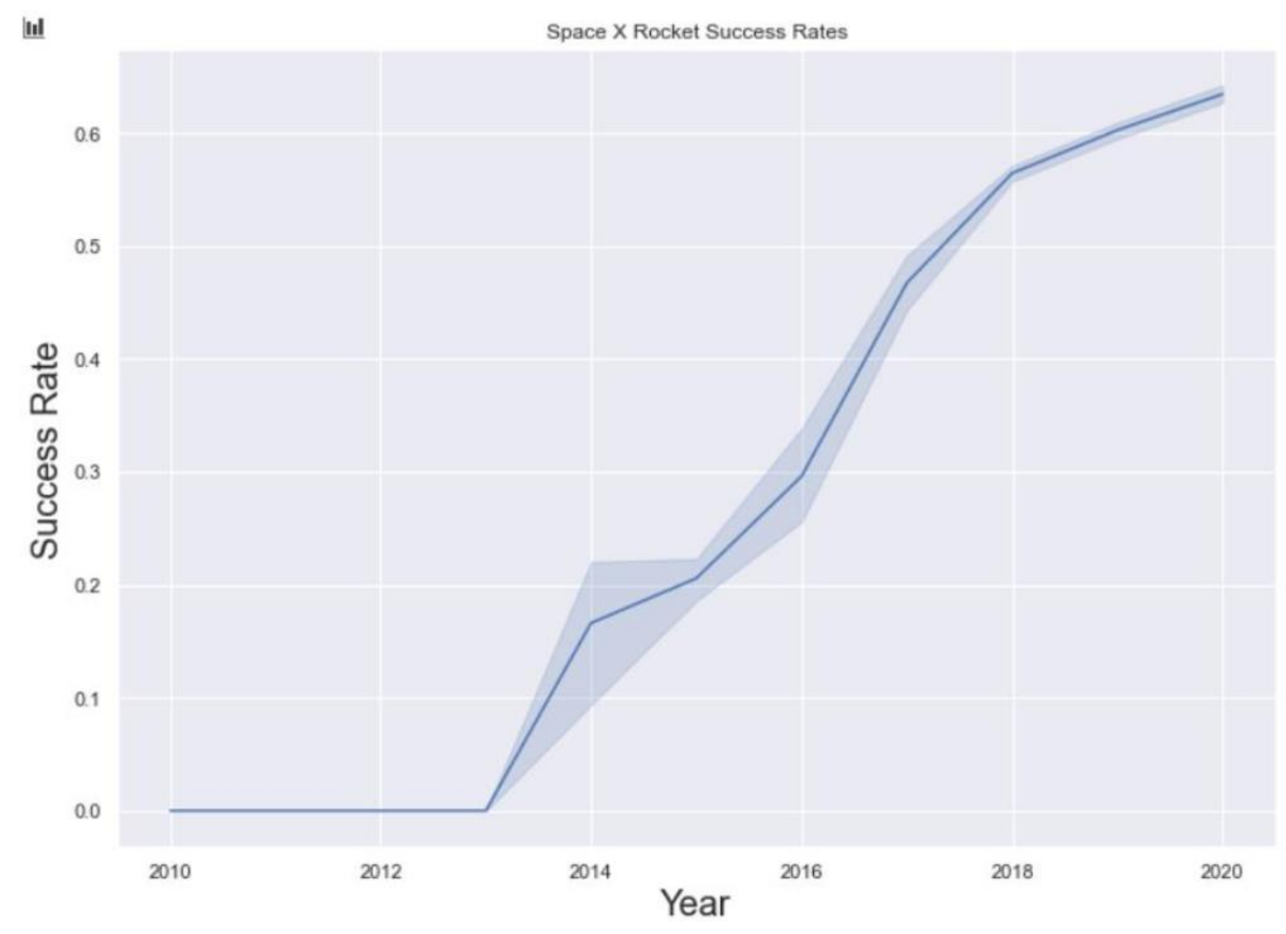
Payload vs. Orbit type



The heavy payloads have a negative impact on GTO orbits while positive impact in GTO and Polar LEO (ISS) orbits.

Launch Success Yearly Trend

Yearly Average Success Rate

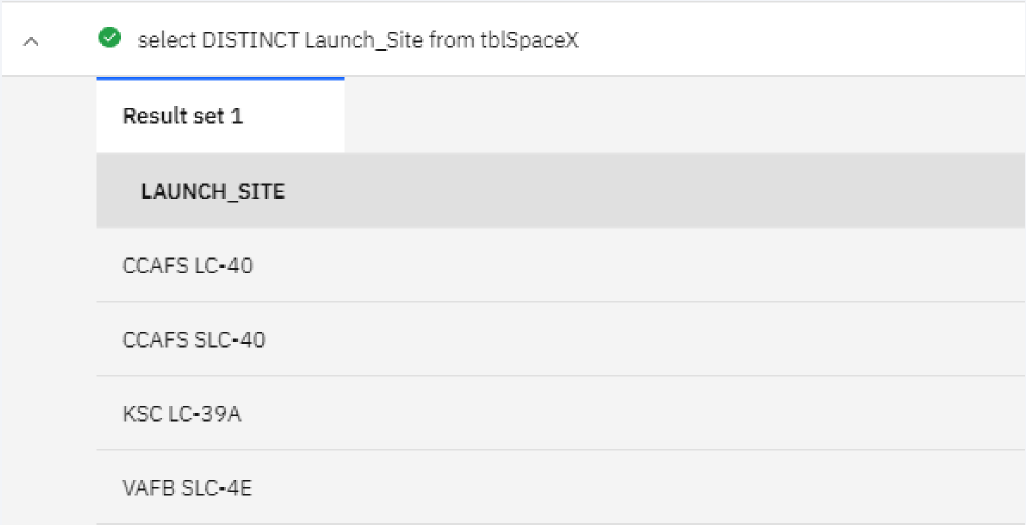


The success rate is continuously increasing from 2013 till 2020.

All Launch Site Names

SQL QUERY

select DISTINCT Launch_Site from tblSpaceX



SQL **DISTINCT** clause is used to remove the duplicates columns from the result set. Using the word ***DISTINCT*** in the query means that it will only show Unique values of ***Launch_Site*** column from ***tblSpaceX.***

Launch Site Names Begin with 'CCA'

<u>SQL QUERY</u>

Select * from tblSpaceX WHERE LAUNCH_SITE LIKE '%KSC%' LIMIT 5

The SQL **LIMIT** statement restricts how many rows a query returns. Using the word ***LIMIT 5*** in the query means that it will display 5 records from query results **.**

LIKE condition is a wild card. E.g. the Launch_site column is filtered with all values with words ***'KSC'***.

Select * from tblSpaceX WHERE LAUNCH_SITE LIKE '%KSC%' LIMIT 5 Run time: 0.005 s

Result set 1

DATE	TIME__UTC_	BOOSTER_VERSION	LAUNCH_SITE	PAYLOAD	PAYLOAD_M
2017-01-05	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300
2017-03-06	21:07:00	F9 FT B1035.1	KSC LC-39A	SpaceX CRS-11	2708
2017-05-07	23:38:00	F9 FT B1037	KSC LC-39A	Intelsat 35e	6761
2017-07-09	14:00:00	F9 B4 B1040.1	KSC LC-39A	Boeing X-37B OTV-5	4990
2017-11-10	22:53:00	F9 FT B1031.2	KSC LC-39A	SES-11 / EchoStar 105	5200

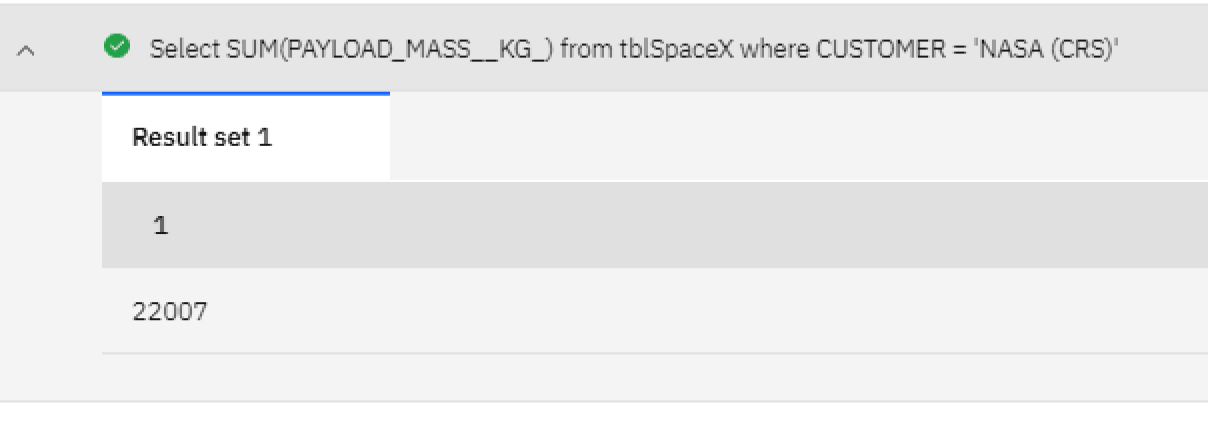
Total Payload Mass

SQL QUERY

Select SUM(PAYLOAD_MASS__KG_) from tblSpaceX where CUSTOMER = 'NASA (CRS)'

SQL **SUM** function is used to find out the sum of a field in various records. Using the function ***SUM*** the query will get the total of column ***PAYLOAD_MASS__KG_***

The SQL **WHERE** clause is used to specify a condition while fetching the data from a single table or by joining with multiple tables. If the given condition is satisfied, then only it returns a specific value from the table.
The ***WHERE*** clause filters the dataset on Customer column with ***NASA (CRS)*** values.

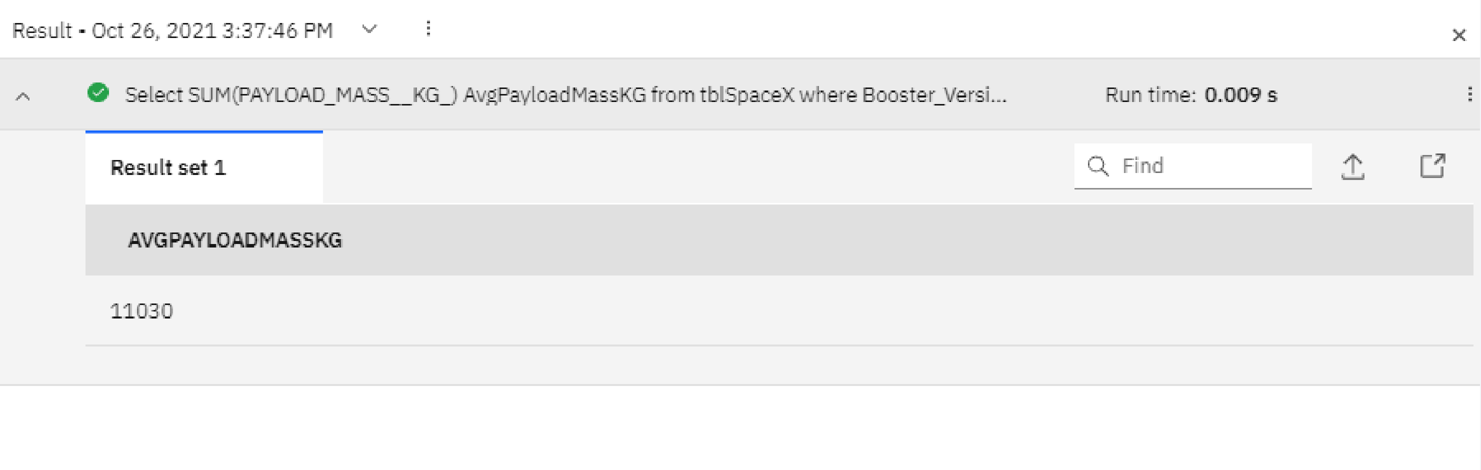


Average Payload Mass by F9 v1.1

SQL QUERY

Select SUM(PAYLOAD_MASS__KG_) AvgPayloadMassKG from tblSpaceX where Booster_Version like 'F9 v1.1'

The SQL Average function which is known as **AVG()** function in T-SQL. AVG() function is an aggregate function that calculates the average value of a numerical dataset that returns from the SELECT statement.
In the above qury, using the ***AVG** function* , it will get the average of the column ***PAYLOAD_MASS__KG_*** where Booster Version is F9 V.1.1

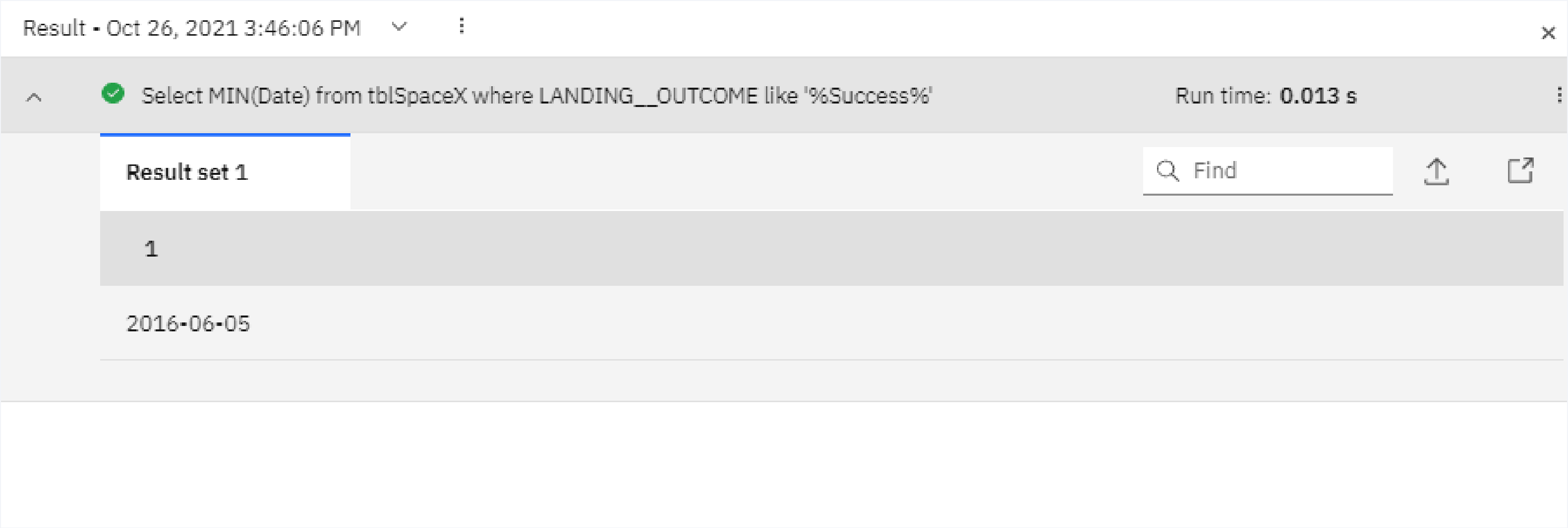


First Successful Ground Landing Date

SQL QUERY

Select MIN(Date) from tblSpaceX where LANDING__OUTCOME like '%Success%'

The **MIN()** function returns the smallest value of the selected column.
In above query, using the function ***MIN*** it get the minimum date in the column ***Date*** where Landing outcome has word "Success".



Successful Drone Ship Landing with Payload between 4000 and 6000

SQL QUERY

Select Booster_Version from tblSpaceX where LANDING__OUTCOME like '%Success%' AND (PAYLOAD_MASS__KG_>4000 AND PAYLOAD_MASS__KG_<6000)

The **AND** function returns TRUE if all its arguments evaluate to TRUE, and returns FALSE if one or more arguments evaluate to FALSE.

The **<** AND **>** is used for adding condition for a numeric values. It represent to Less than (<) and Greater than (>) the values.

Result - Oct 26, 2021 3:50:57 PM

Select Booster_Version from tblSpaceX where LANDING__OUTCOME like '%Success%' AND (... Run time: 0.006 s

Result set 1

BOOSTER_VERSION
F9 FT B1022
F9 FT B1032.1
F9 B4 B1040.1
F9 FT B1031.2
F9 B4 B1043.1

Result set is truncated, only the first 8 rows have been loaded. Select "View all loaded data" on the right top of the result to view all loaded rows. More

Total Number of Successful and Failure Mission Outcomes

SQL QUERY

```
Select Count(*) Success , (Select Count(*) Failure from tblSpaceX where
LANDING__OUTCOME like '%Failure%')
from tblSpaceX where LANDING__OUTCOME like '%Success%'
```

A **Subquery** or **Inner query** or a **Nested query** is a query within another SQL query and embedded within the WHERE clause. A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Result - Oct 26, 2021 4:01:10 PM

Select Count(*) Success , (Select Count(*) Failure from tblSpaceX where LANDING__OUTCO... Run time: 0.020 s

Result set 1

SUCCESS	FAILURE
27	5

Boosters Carried Maximum Payload

SQL QUERY

```
SELECT Booster_Version,MAX(PAYLOAD_MASS__KG_) MaxPayload
FROM tblSpaceX GROUP BY Booster_Version
ORDER BY MaxPayload DESC
```

The **GROUP BY** statement groups rows that have the same values into summary rows, like "find the maximum values of PAYLOAD_MASS__KG_ in each Booster_Version". The GROUP BY statement is often used with aggregate functions (COUNT() , MAX() , MIN() , SUM() , AVG()) to group the result-set by one or more columns.

Result • Oct 26, 2021 4:05:53 PM

SELECT Booster_Version,MAX(PAYLOAD_MASS__KG_) MaxPayload FROM tblSpaceX GROUP ... Run time: 0.005 s

Result set 1

Find

BOOSTER_VERSION	MAXPAYLOAD
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1049.5	15600
F9 B5 B1058.3	15600
F9 B5 B1060.2	15600

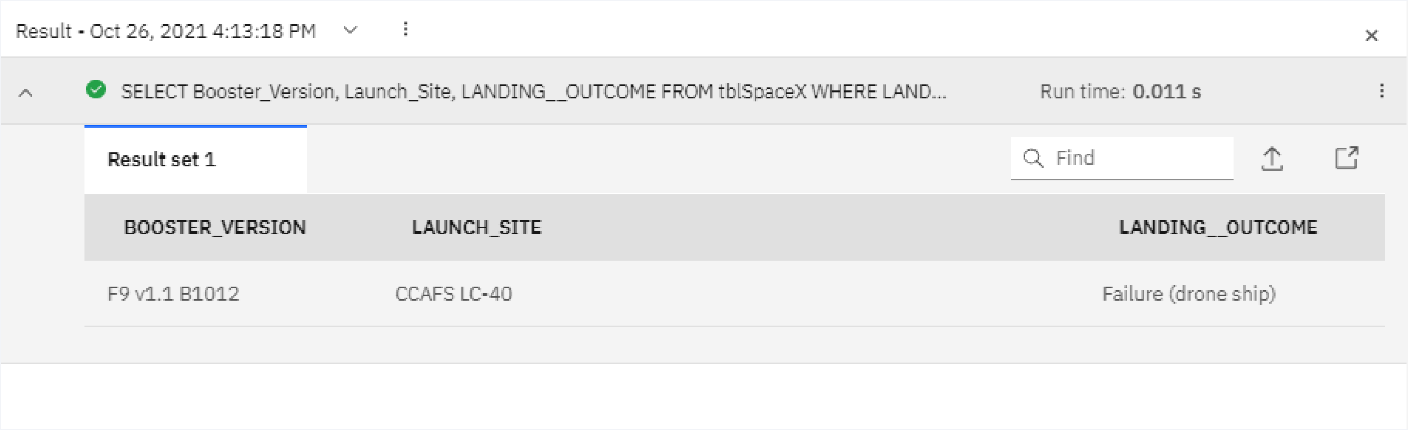
Result set is truncated, only the first 43 rows have been loaded. Select "View all loaded data" on the right top of the result to view all loaded rows. More

2015 Launch Records

SQL QUERY

```
SELECT Booster_Version, Launch_Site, LANDING__OUTCOME
FROM     tblSpaceX
WHERE  LANDING__OUTCOME like '%Failure%' AND YEAR(Date) = 201
```

The **YEAR()** function returns an integer value which represents the year of the specified date. The function accepts an argument which can be a literal date value.



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

SQL QUERY

```
Select Booster_Version, Launch_Site, Count(*) Success from
(select * from tblSpaceX where (Date >='06/04/2010' and Date <='03/20/2017') AND
LANDING__OUTCOME like '%Success%' )
group by  Booster_Version, Launch_Site
order by Success DESC
```

The below query show the highest ranking in Success Rate from 2010-06-04 and 2017-03-20.

To get the highest FAILURE, just change the condition of LANDING__OUTCOME to ‘%Failure%’

Result set 1

Find

BOOSTER_VERSION	LAUNCH_SITE	SUCCESS
F9 v1.1	CCAFS LC-40	3
F9 FT B1020	CCAFS LC-40	1
F9 FT B1021.1	CCAFS LC-40	1
F9 FT B1022	CCAFS LC-40	1
F9 FT B1032.1	KSC LC-39A	1

Result set is truncated, only the first 13 rows have been loaded. Select "View all loaded data" on the right top of the result to view all loaded rows.

More

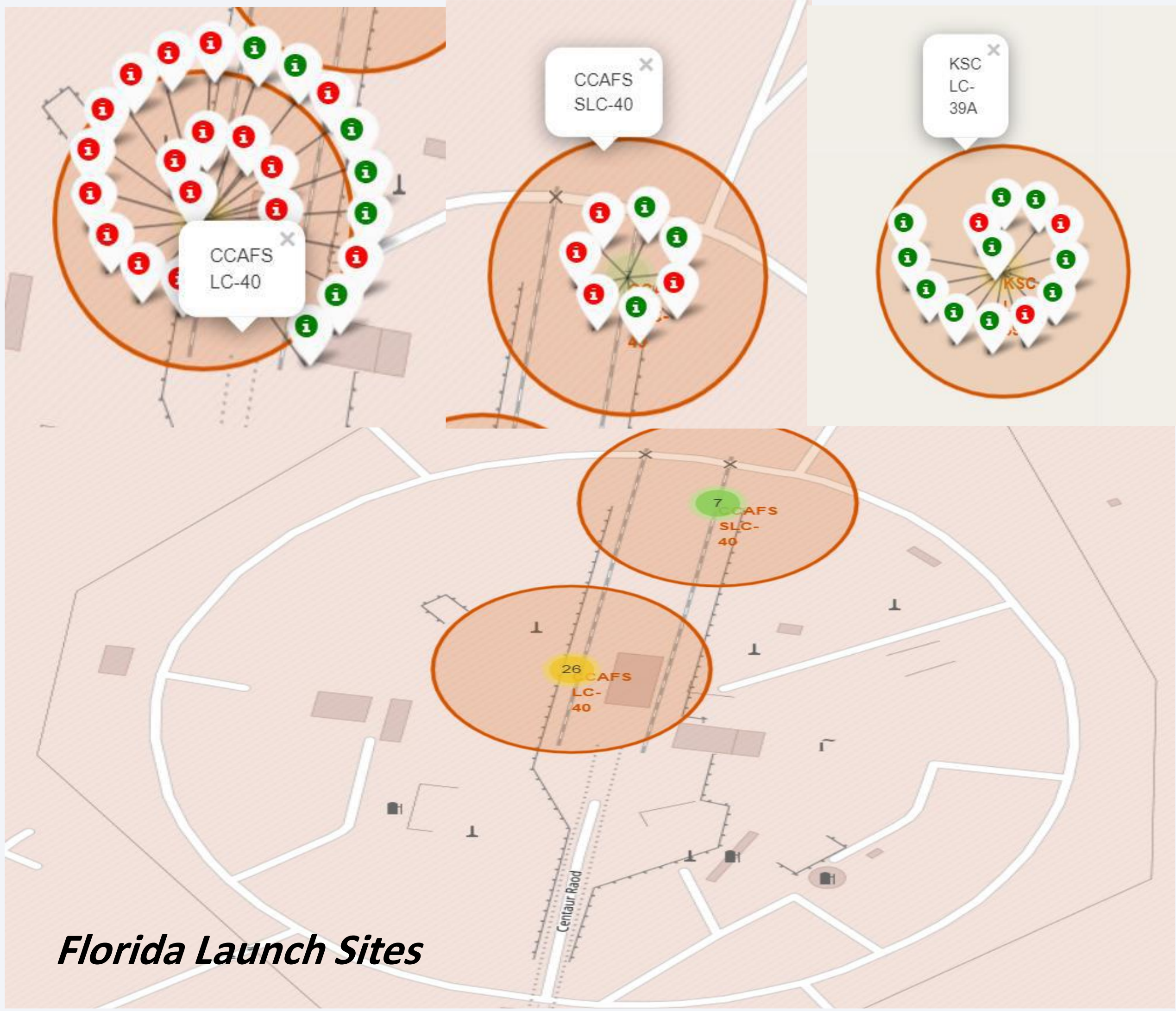
Section 4

Launch Sites Proximities Analysis

All Launch Sites in Global Map

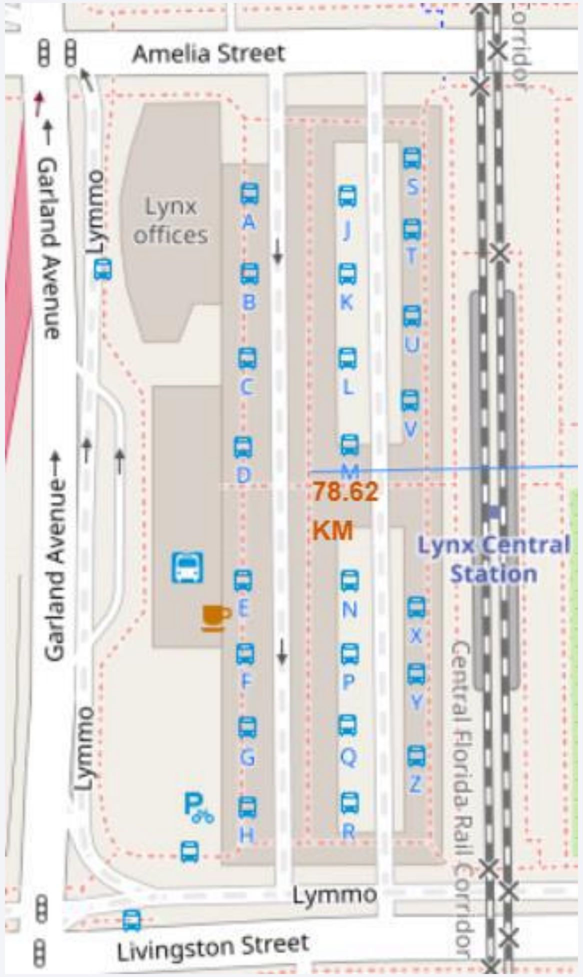
We can see that the SpaceX launch sites are in the United States of America coasts. Florida and California

Launch Sites Status



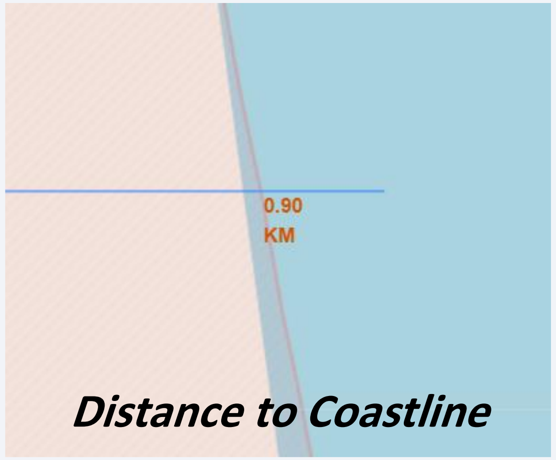
Green Marker shows successful Launches and
Red Marker shows Failures

Launch Sites Distance



Distance to Railway Station

Distance to closest Highway



Distance to Coastline

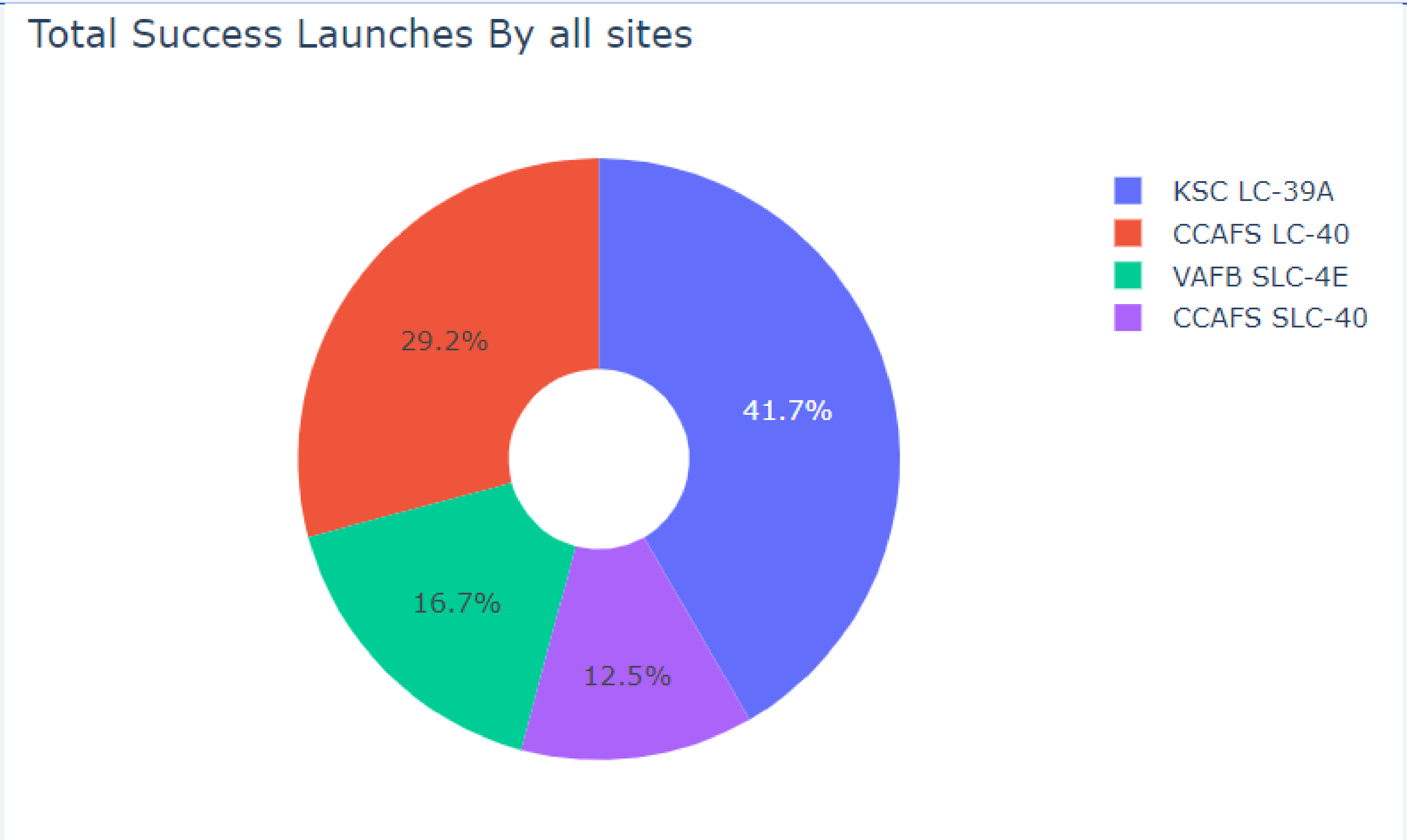
Distance to City

- ✓ Are launch sites in close proximity to railways? No
- ✓ Are launch sites in close proximity to highways? No
- ✓ Are launch sites in close proximity to coastline? Yes
- ✓ Do launch sites keep certain distance away from cities? Yes

Section 5

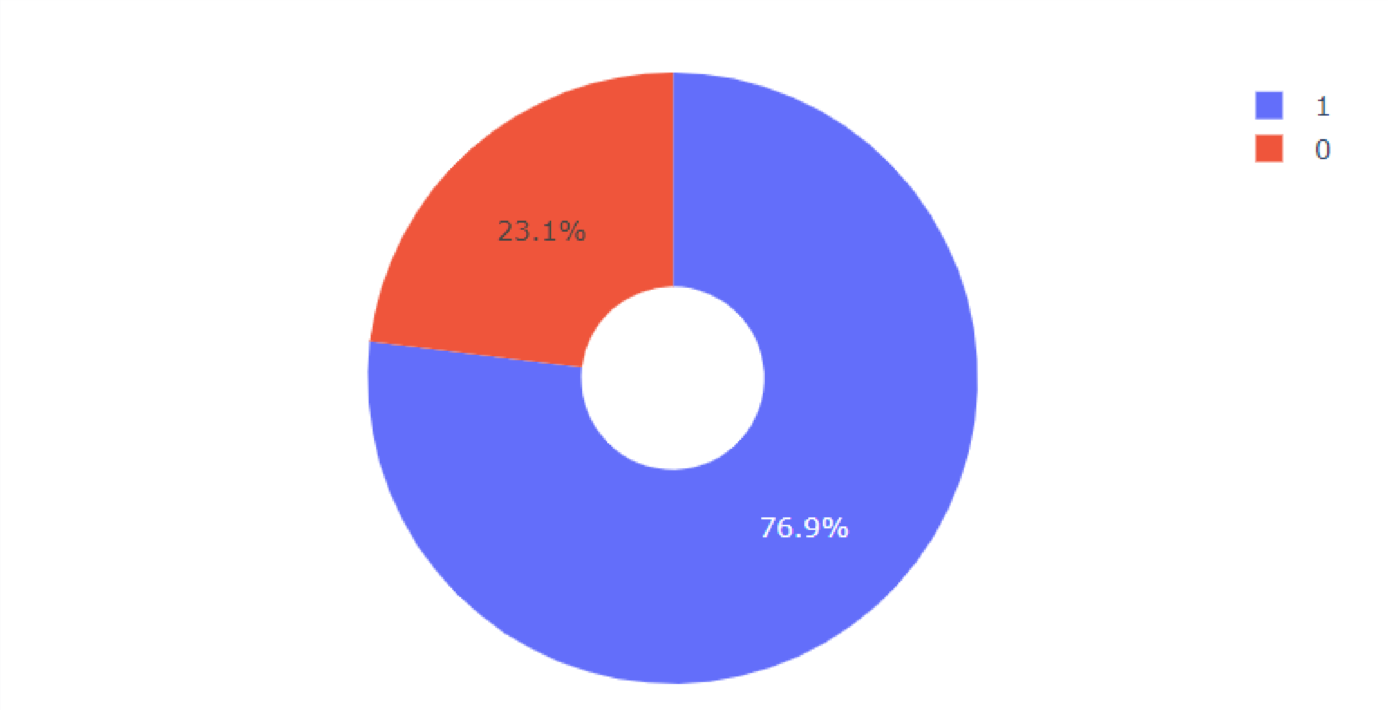
Build a Dashboard with Plotly Dash

Successful Launches By All Sites



The KSC LC-39A had the
most successful launches from all the sites.

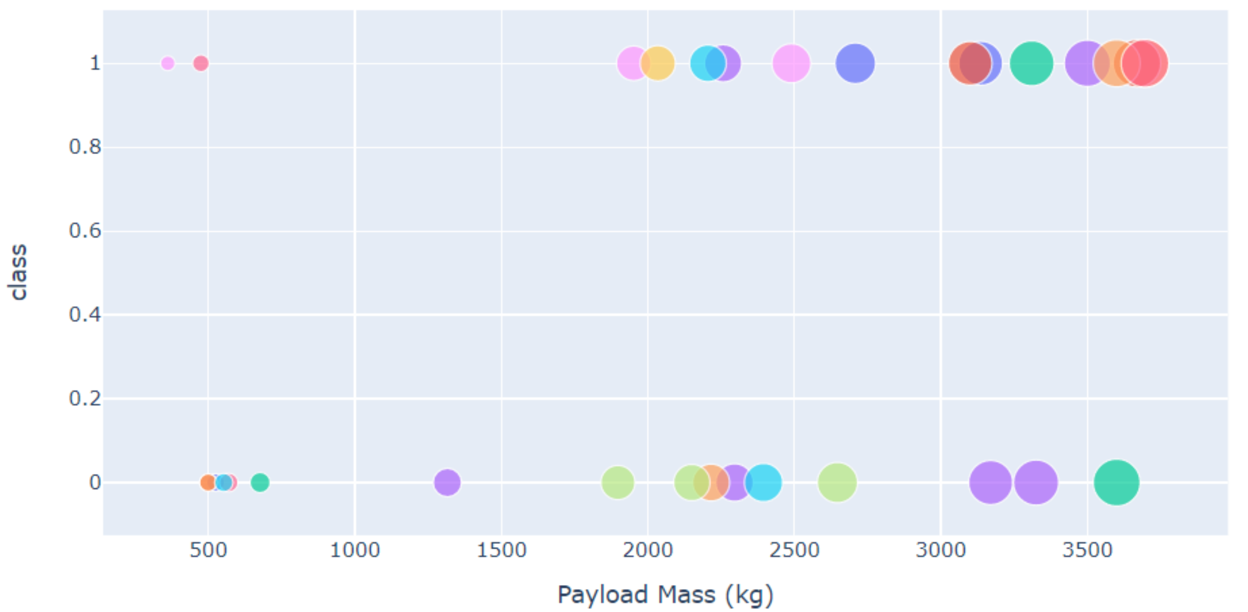
KSC LC-39A Success Ratio



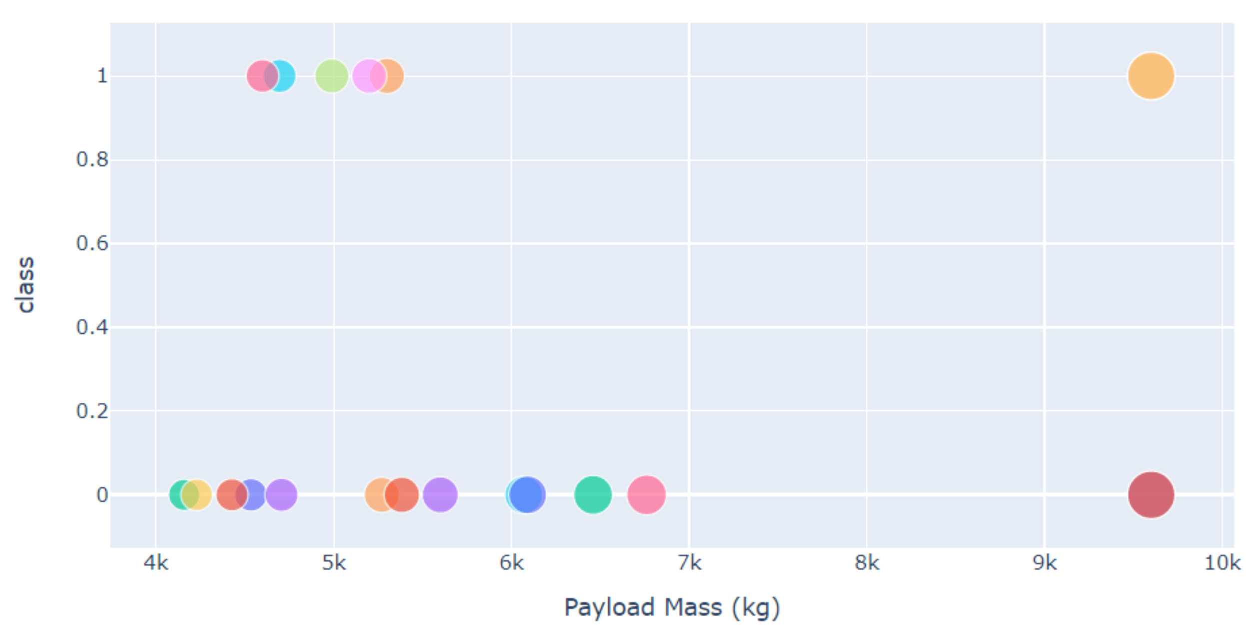
KSC LC-39A achieved a 76.9% success rate while 23.1% is a failure rate.

Payload vs. Launch Outcome in Different Payload

Low Weighted Payload 0kg –4000kg



Heavy Weighted Payload 4000kg –10000kg



The success rates for low weighted payloads is higher than the heavy weighted payloads.

Section 6

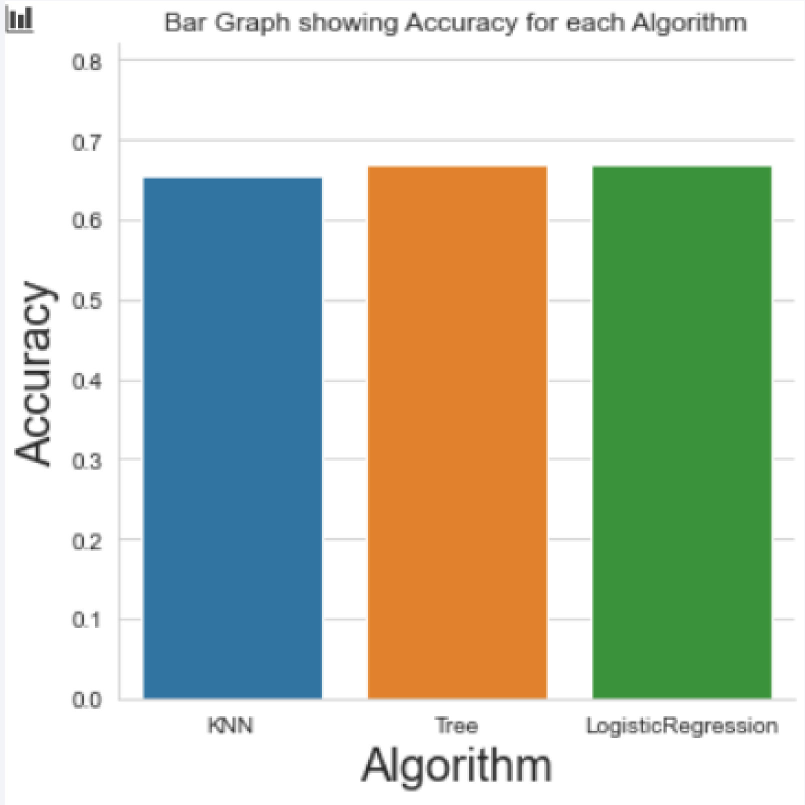
Predictive Analysis (Classification)

Classification Accuracy

The accuracy is extremely close, and Logistic regression and Tree tied as most accurate.

```
bestalgorithm = max(algorithms, key=algorithms.get)
```

	Accuracy	Algorithm
0	0.653571	KNN
1	0.667857	Tree
2	0.667857	LogisticRegression

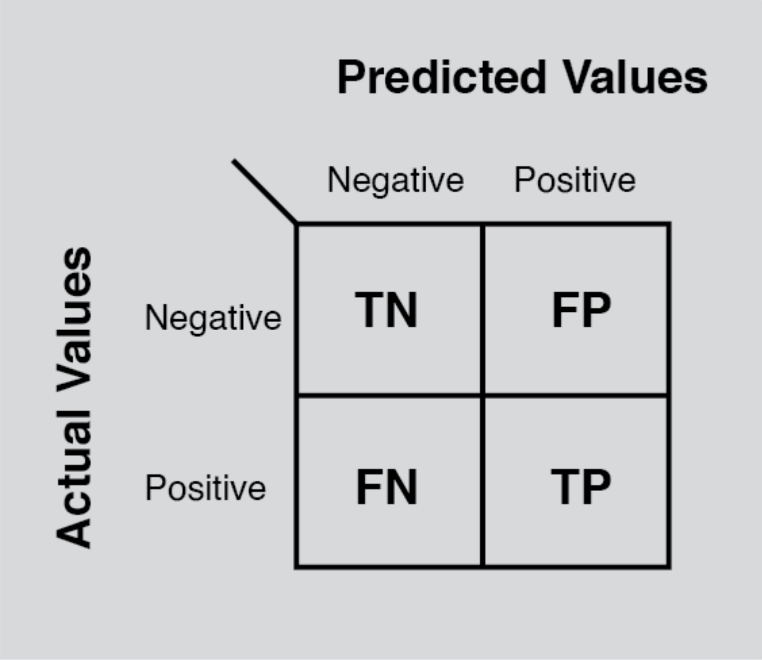


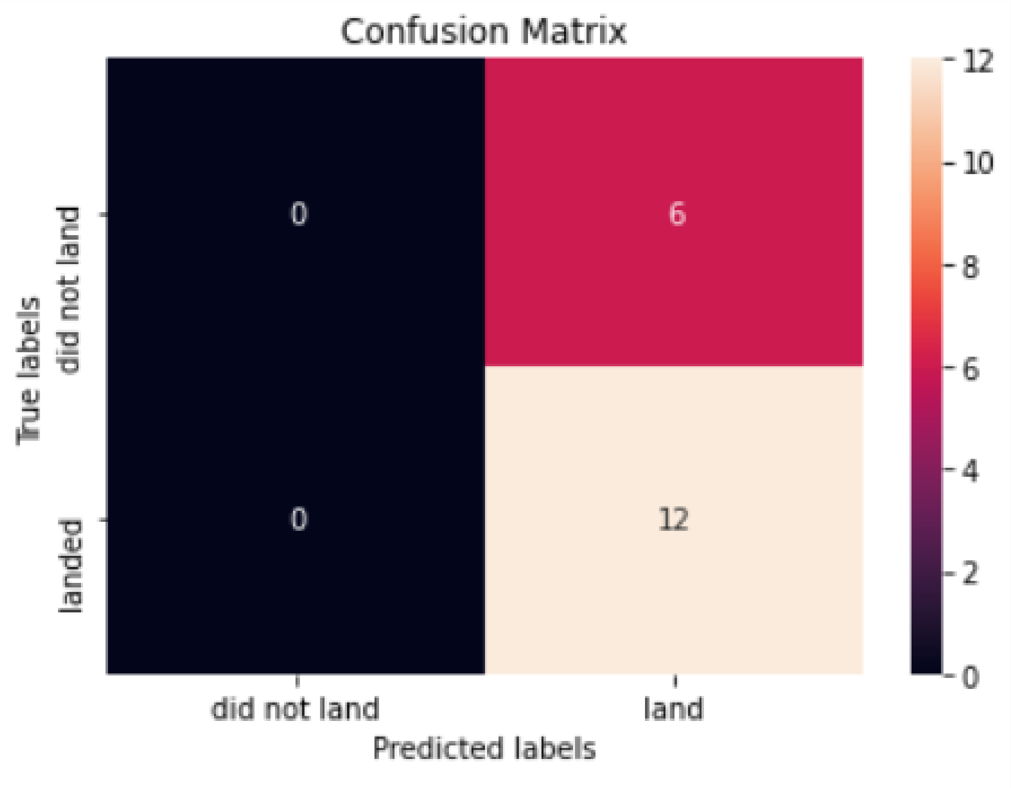

```
Best Algorithm is Tree with a score of 0.6678571428571429
Best Params is : {'criterion': 'gini', 'max_depth': 2, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'best'}
```

After selecting the best hyperparameters for the decision tree classifier using the validation data, we achieved 83.33% accuracy on the test data.

Confusion Matrix

The Tree can distinguish between the different classes and the major problem is false positives.





Conclusions

- The Tree Classifier Algorithm is the best for Machine Learning for this dataset
- Low weighted payloads perform better than the heavier payloads
- The success rates for SpaceX launches is directly proportional time in years they will eventually perfect the launches
- We can see that KSC LC-39A had the most successful launches from all the sites
- Orbit GEO,HEO,SSO,ES-L1 has the best Success Rate

Thank you!