Winning Space Race with Data Science

Vijay Yadav
17-Feb-2023

Outline

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

Executive Summary

- Summary of methodologies
    - Data Collection through API
    - Data Collection with Web Scraping
    - Data Wrangling - Exploratory Data Analysis with SQL
    - Exploratory Data Analysis with Data Visualization
    - Interactive Visual Analytics with Folium
    - Machine Learning Prediction

- Summary of all results
    - Exploratory Data Analysis result
    - Interactive analytics in screenshots
    - Predictive Analytics result from Machine Learning Lab

Introduction

- Project background and context

  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. We have to predict the Falcon 9 first stage will land successfully or not.

- Problems you want to find answers

  - What influences if the rocket will land successfully?
  - The effect each relationship with certain rocket variables will impact in determining the success rate of a successful landing.
  - What conditions does SpaceX have to achieve to get the best results and ensure the best rocket success landing rate.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX Rest API and web scrapping from Wikipedia.
- Perform data wrangling
  - One Hot Encoding data field for Machine Learning and dropping non-necessary columns
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Use Scatter Graphs, Bar graphs to show relationship between data.
- 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 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. As mentioned, the dataset was collected by REST API and Web Scrapping from Wikipedia.

I followed following steps:

1. Getting data from API or Web Site
2. Make dataframe from it
3. Filter dataframe as per requirement
4. Export filtered dataframe to flate file (.csv)

Data Collection – SpaceX API

Getting Response from API

```
spacex_url="https://api.spacexdata.com/v4/launches/past"

response = requests.get(spacex_url)
```

Converting Response to a .json file

```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

Data Cleaning and filling missing values

```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that have multiple payloads in a si
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the feature.
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

Github Link

Data Collection - Scraping

Request the Falcon9 Launch Wiki page from its URL

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
# use requests.get() method with the provided static_url
# assign the response to a object
data = requests.get(static_url).text
```

Create a BeautifulSoup object from HTML response

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

Extract all column/variable names from the HTML table header

```
first_launch_table = html_tables[2]
print(first_launch_table)
```

```
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
```

[Github Link]

Data Wrangling

Calculate the number of launches on each site

```
df.LaunchSite.value_counts()
```

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

Calculate the number and occurrence of each orbit

```
df.Orbit.value_counts()
```

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

Calculate the number and occurrence of mission outcome per orbit type

```
landing_outcomes = df.Outcome.value_counts()
landing_outcomes
```

```
True ASDS        41
None None        19
True RTLS        14
False ASDS        6
True Ocean        5
False Ocean       2
None ASDS         2
False RTLS        1
Name: Outcome, dtype: int64
```

[Github Link]

Create a landing outcome label from Outcome column

```
df['Class'] = df['Outcome'].apply(lambda x: 'value if condition is met' if x == 0 else 'value if condition is not met')
```

Export dataset as .csv

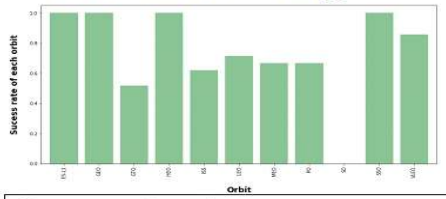
```
df = df.to_csv(r'C:\Vijay\ZTEST\DataScience\10.Applied Data Science Capstone\dataset_part_2.csv')
```

EDA with Data Visualization

Bar Graph:

Bar chart makes it easy to compare data between different groups at glance.

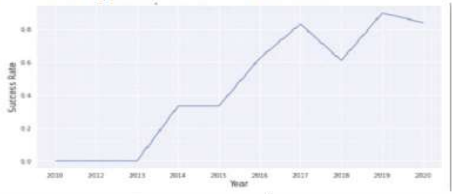
- **Success rate Vs. Orbit Type**



Line Graph:

Line graph is the easiest way to show trends and it help to make predictions about the results of data not yet recorded.

- **Yearly success rate trend**



Scatter Graphs:

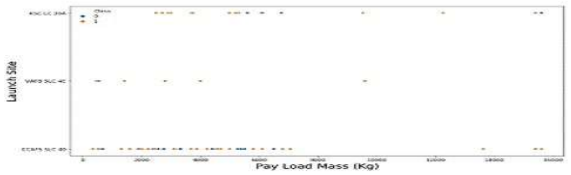
Scatter plots show attributes dependency on each other. With the help of scatter graphs it very each to predict which factor will lead to maximum probability of success in both outcome and landing.

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

[Github Link]

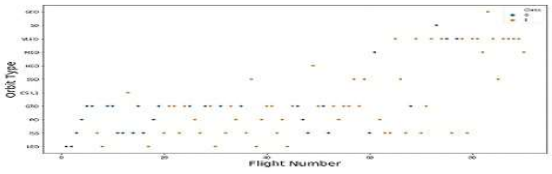
TASK 2: Visualize the relationship between Payload and Launch Site

```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis
plt.figure(figsize=(14,8))
sns.scatterplot(x="PayloadMass", y="LaunchSite", hue="Class", data = df)
plt.xlabel("Pay Load Mass (Kg)",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



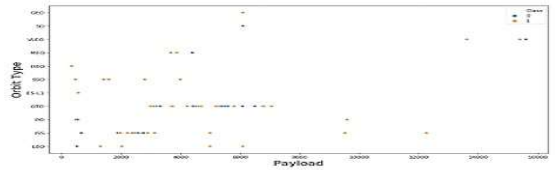
TASK 4: Visualize the relationship between FlightNumber and Orbit type

```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the
plt.figure(figsize=(14,8))
sns.scatterplot(x="FlightNumber", y="Orbit", hue="Class", data = df)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Orbit Type",fontsize=20)
plt.show()
```



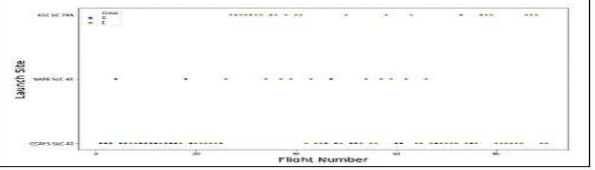
TASK 5: Visualize the relationship between Payload and Orbit type

```
# Similarly, we can plot the Payload vs. Orbit scatter point charts to rev
# Plot a scatter point chart with x axis to be Payload and y axis to be th
plt.figure(figsize=(14,8))
sns.scatterplot(x="PayloadMass", y="Orbit", hue="Class", data = df)
plt.xlabel("Payload",fontsize=20)
plt.ylabel("Orbit Type",fontsize=20)
plt.show()
```



TASK 1: Visualize the relationship between Flight Number and Launch Site

```
plt.figure(figsize=(14,8))
sns.scatterplot(x="FlightNumber", y="LaunchSite", hue="Class", data = df)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



EDA with SQL

Here I used **sqlite** as Database

```
!pip install sqlalchemy==1.3.9
import csv, sqlite3
import pandas as pd

%load_ext sql
con = sqlite3.connect("my_data2.db")
cur = con.cursor()

%sql sqlite:///my_data2.db
df = pd.read_csv(r"https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/labs/module_2/da
df.to_sql("SPACEXTBL", con)
```

I gathered information of points using SQL queries:

- 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 2017
- Ranking the count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order

Guthub Link

Build an Interactive Map with Folium

To visualize the Launch Data into an interactive map. We took 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 **Green** and **Red** markers on the map in a **MarkerCluster()**

Using Haversine's formula we calculated the distance from the Launch Site to various landmarks to find various trends about what is around the Launch Site to measure patterns.

Lines are drawn on the map to measure distance to landmarks

Example of some trends in which the Launch Site is situated in.

- Are launch sites in close proximity to railways?
- Are launch sites in close proximity to highways?
- Are launch sites in close proximity to coastline?
- Do launch sites keep certain distance away from cities?

Github Link

Build a Dashboard with Plotly Dash

The dashboard is built with Flask and Dash web framework.

- Pie Chart showing the total success for all launches from a selected sites or all sites (% of success in relation to launch 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.

Used Python Anywhere to host the website.

Guithub Link to Source Code

Predictive Analysis (Classification)

FINDING THE BEST PERFORMING CLASSIFICATION MODEL

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

IMPROVING MODEL

- Feature Engineering
- Algorithm Tuning

EVALUATING MODEL

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

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

BUILDING MODEL

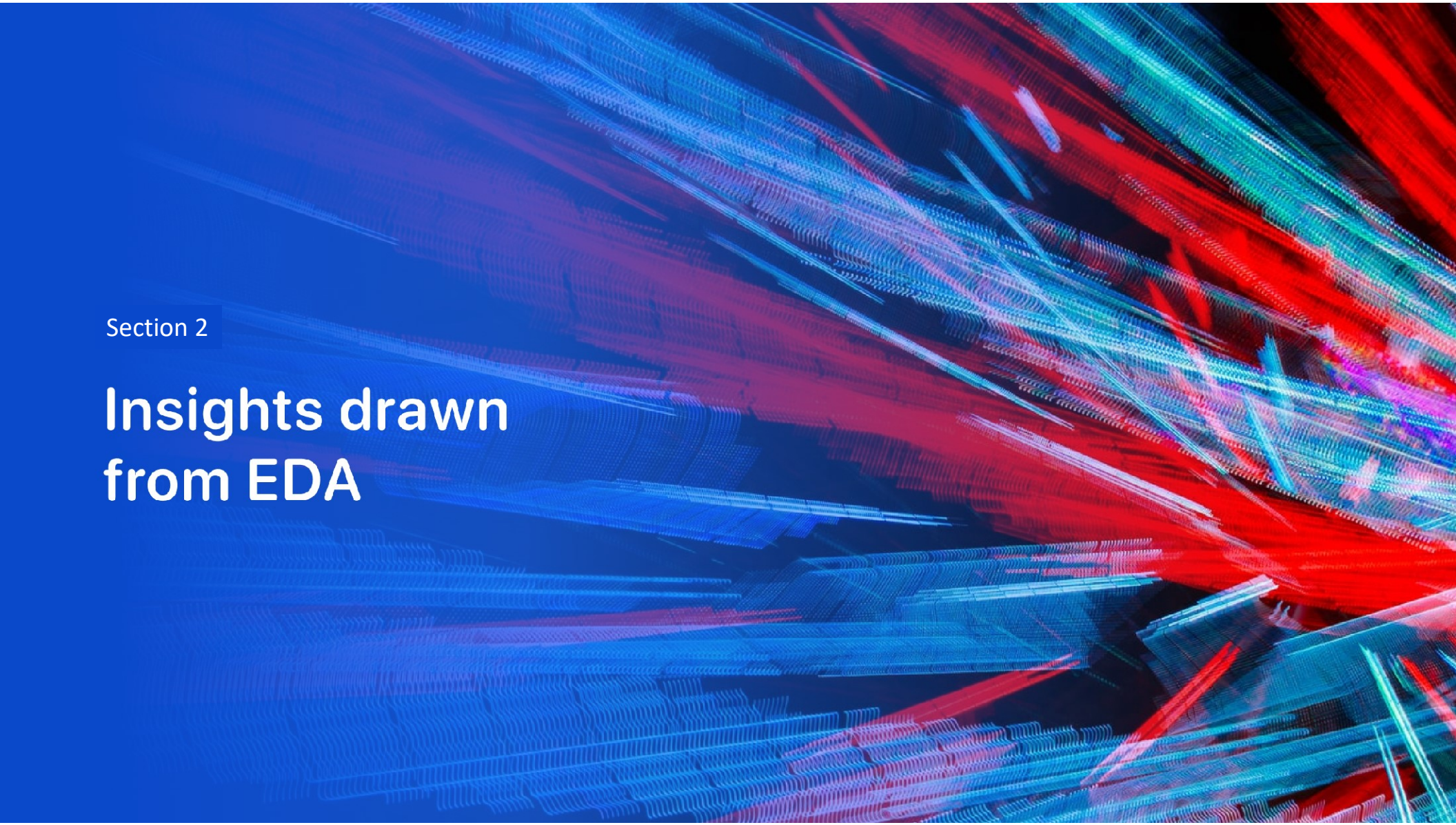
- Load our dataset into NumPy and Pandas
- Transform Data
- Split our data into training and test data sets
- Check how many test samples we have
- Decide which type of machine learning algorithms we want to use
- Set our parameters and algorithms to GridSearchCV
- Fit our datasets into the GridSearchCV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

[Github Link]

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

Results

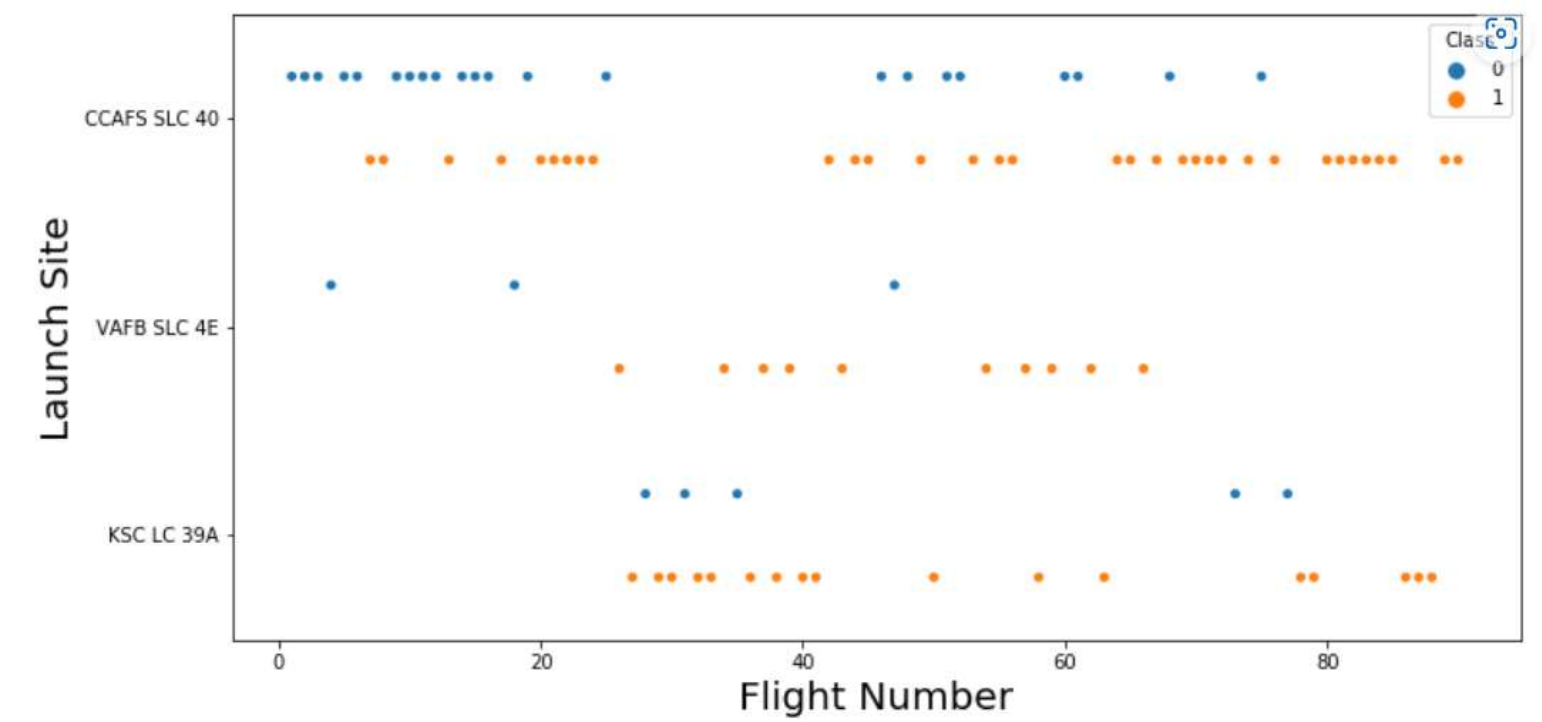
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results



Section 2

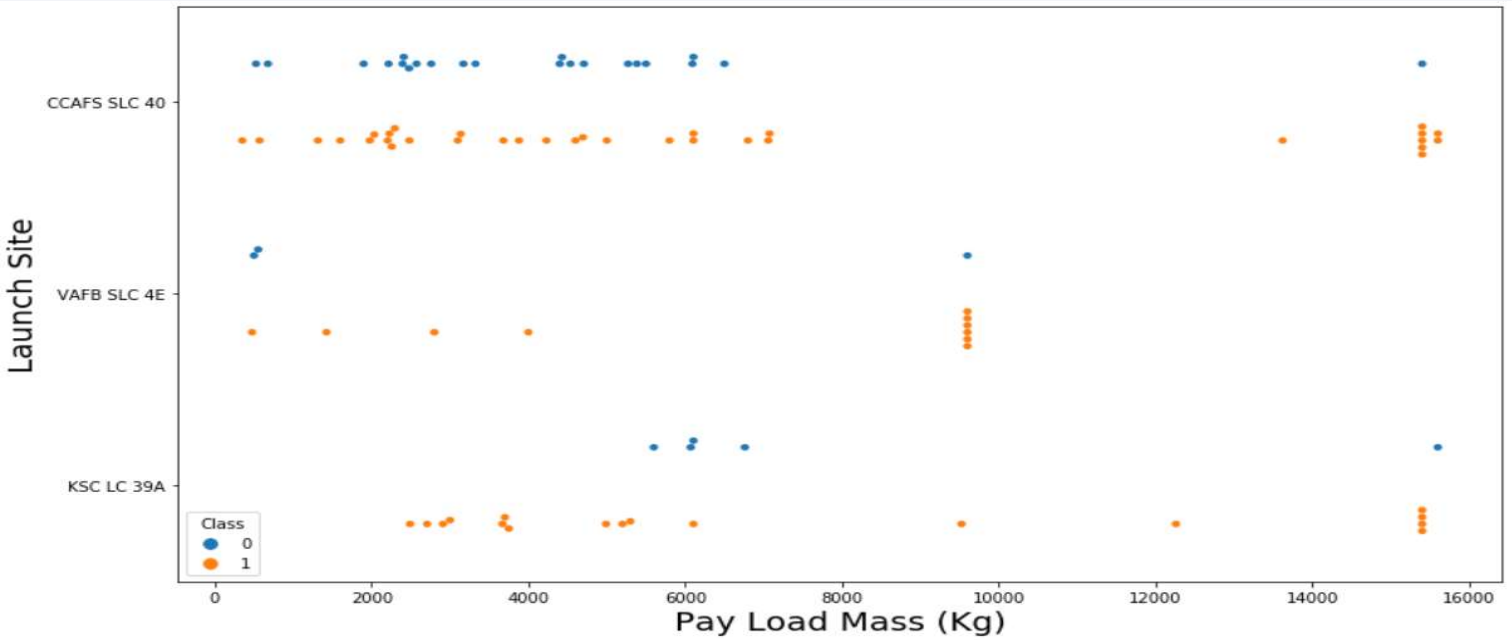
Insights drawn from EDA

Flight Number vs. Launch Site



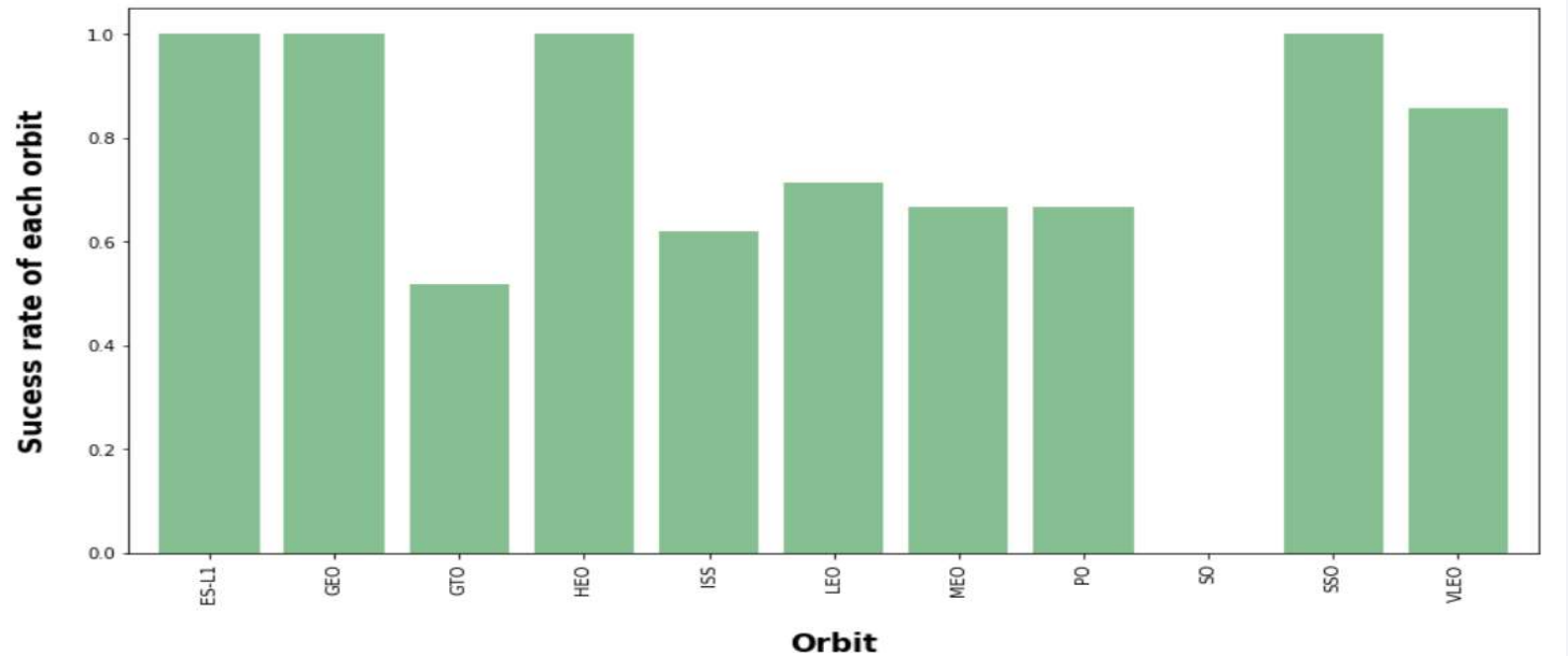
The large amount of flights at a launch site will increase the success rate at a launch site

Payload vs. Launch Site



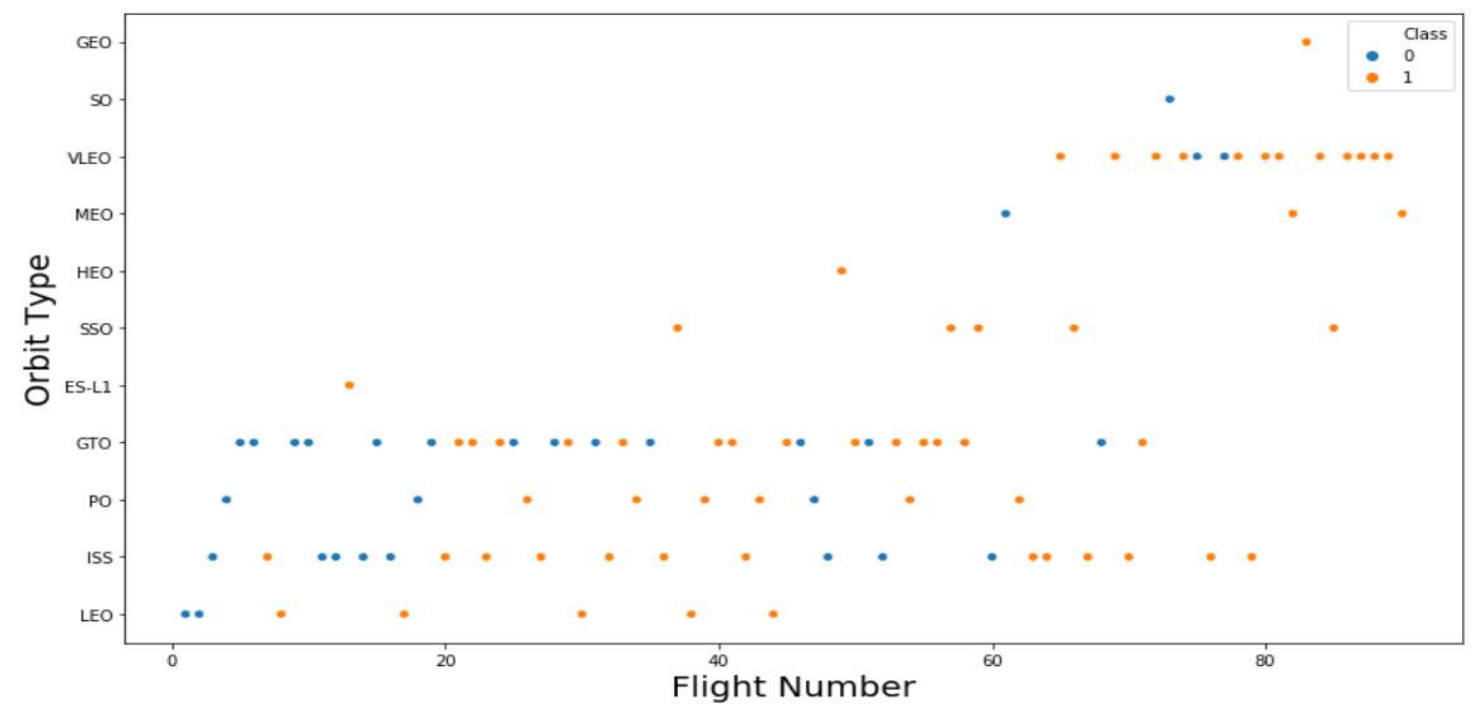
Grater the payload mass (>7000 kg), increase the success rate. But there is no clear pattern to say the launch site is dependent to the payload mass for the success rate

Success Rate vs. Orbit Type



Orbit ES-L1, GEO, HEO and SSO has the highest success rate

Flight Number vs. Orbit Type



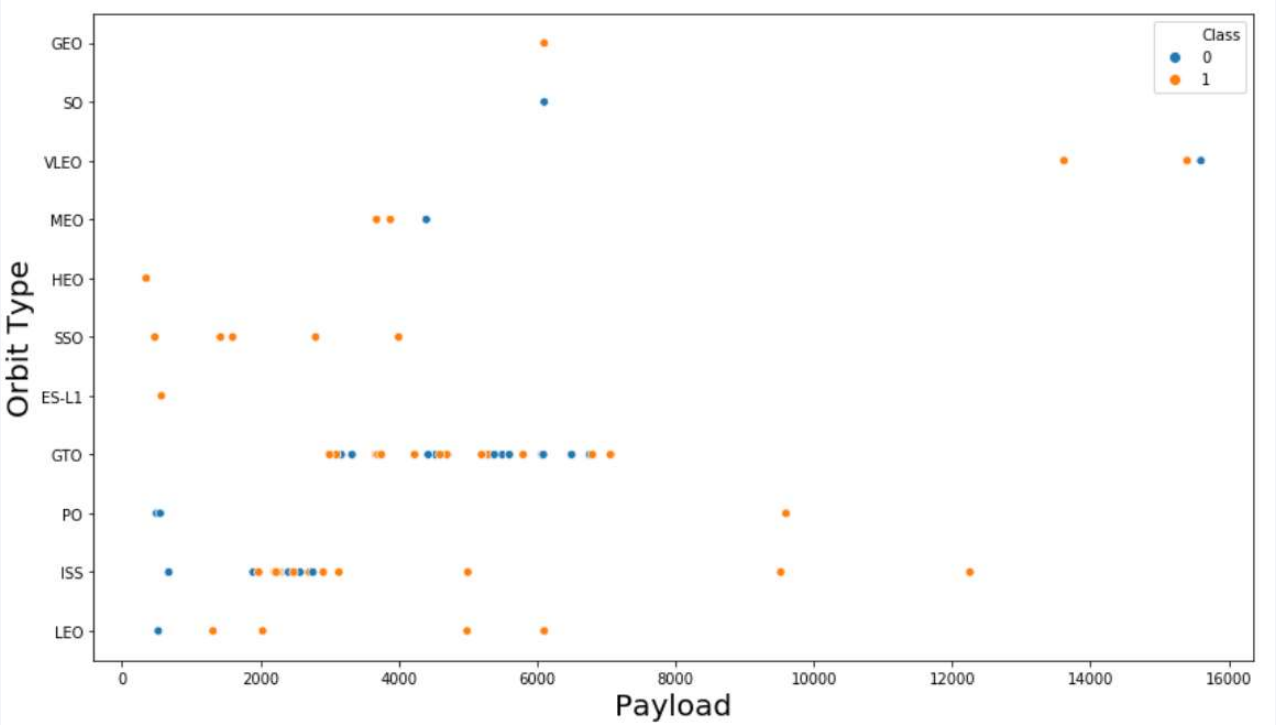
This plot shows generally, the larger the flight number on each orbits is greater the success rate (especially LEO orbit) except for GTO orbit which depicts no relationship between both attributes.

Payload vs. Orbit Type

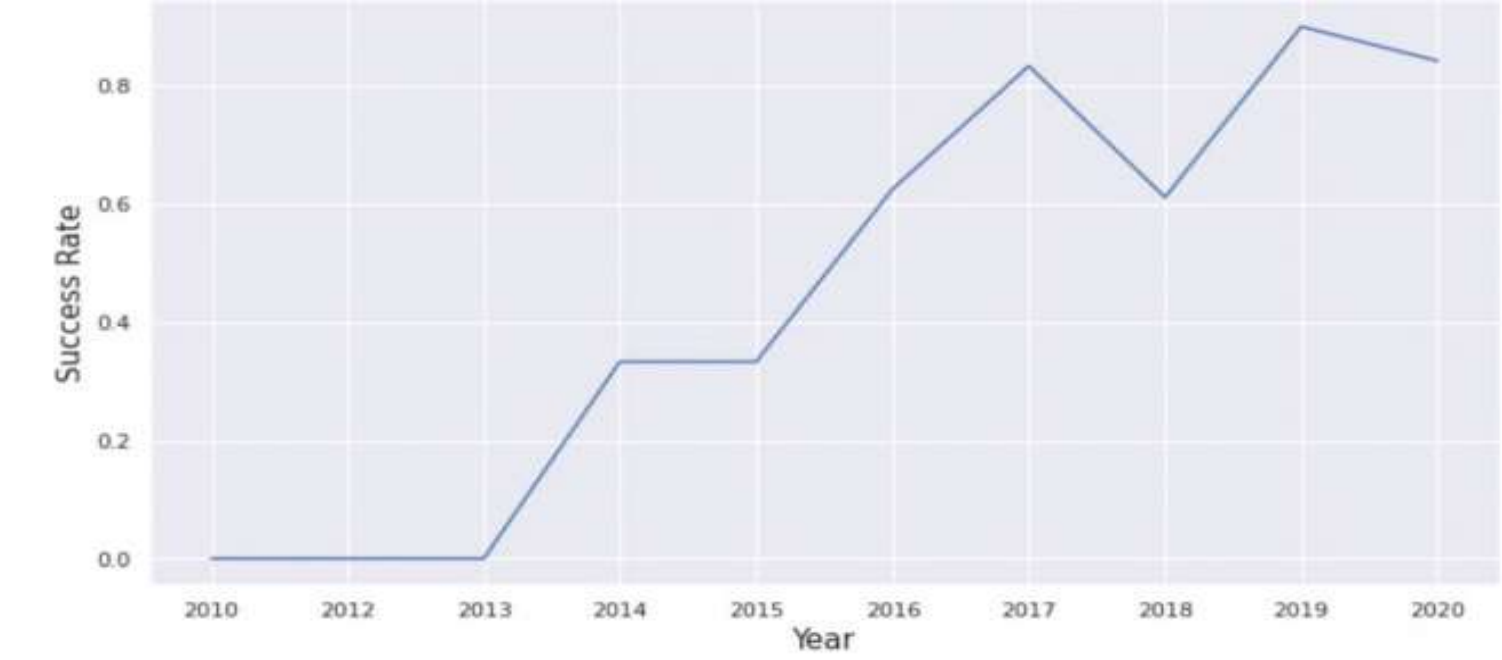
Heavier payload has positive impact on LEO, ISS and PO orbit. However, it has negative impact on MEO and VLEO orbit.

GTO orbit seem to depict no relation between the attributes.

Meanwhile, again, SO, GEO and HEO orbit need more dataset to see any pattern or trend



Launch Success Yearly Trend



This is clearly show the increasing trend from the year 2013 to until 2020.

All Launch Site Names

```
%sql SELECT Distinct LAUNCH_SITE FROM SPACEXTBL
```

```
 * sqlite:///my_data2.db
Done.
```

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

Using the word **DISTINCT** in the query means that it will only show Unique values in the Launch_Site column

Launch Site Names Begin with 'CCA'

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

```
 * sqlite:///my_data2.db
Done.
```

index	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
0	04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	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)
2	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
3	08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Here I used **LIMIT** the query above to display 5 records where launch sites begin with “CCA”.

Total Payload Mass

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE CUSTOMER='NASA (CRS)'
```

```
 * sqlite:///my_data2.db
Done.
```

SUM(PAYLOAD_MASS__KG_)
45596

Using **SUM** function to calculate total payload mass and **where** clause to filter the data by customer name ‘NASA (CRS)’

Average Payload Mass by F9 v1.1

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE BOOSTER_VERSION='F9 v1.1'
```

```
 * sqlite:///my_data2.db
Done.
```

AVG(PAYLOAD_MASS__KG_)
2928.4

Use **AVG** function to calculate the average payload mass carried by booster version F9 v1.1

First Successful Ground Landing Date

```
%sql SELECT min(DATE) FROM SPACEXTBL WHERE Landing_Outcome='Success (ground pad)'
```

```
 * sqlite:///my_data2.db
Done.
```

min(DATE)
01-05-2017

Use **MIN** function with date column to find out the first successful date.

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ between 4000 and 6000 AND Landing_Outcome='Success (drone ship
```

```
 * sqlite:///my_data2.db
Done.
```

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

Here i used the **WHERE** clause to filter for boosters which have successfully landed on drone ship and applied the **BETWEEN – AND** -- condition to determine successful landing with payload mass between 4000 and 6000

Total Number of Successful and Failure Mission Outcomes

```
%sql SELECT COUNT(*) FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE '%Success%' OR MISSION_OUTCOME LIKE '%Failure%'
```

```
 * sqlite:///my_data2.db
Done.
```

COUNT(*)
101

Use **wildcard like '%'** to filter for WHERE Mission_Outcome was a success or a failure.

Boosters Carried Maximum Payload

```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
```

```
 * sqlite:///my_data2.db
Done.
```

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

We determined the booster that have carried the maximum payload using a **subquery** in the WHERE clause and the **MAX()** function.

2015 Launch Records

```
%sql SELECT LANDING_OUTCOME AS LANDING_OUTCOME, BOOSTER_VERSION AS BOOSTER_VERSION, \
LAUNCH_SITE AS LAUNCH_SITE FROM SPACEXTBL \
WHERE LANDING_OUTCOME = 'Failure (drone ship)' AND DATE LIKE '%2015%'
```

```
 * sqlite:///my_data2.db
Done.
```

LANDING_OUTCOME	BOOSTER_VERSION	LAUNCH_SITE
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

We used a combinations of the **WHERE** clause, **LIKE, AND** conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

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

```
%sql SELECT LANDING_OUTCOME, COUNT(LANDING_OUTCOME) as COUNT FROM SPACEXTBL \
     where date BETWEEN '04-06-2010' and '20-03-2017' AND LANDING_OUTCOME IN ('Failure (drone ship)', 'Success (ground pad)') \
     group by LANDING_OUTCOME \
     ORDER BY COUNT(LANDING_OUTCOME) DESC
```

```
 * sqlite:///my_data2.db
Done.
```

Landing_Outcome	COUNT
Success (ground pad)	6
Failure (drone ship)	4

```
%sql SELECT LANDING_OUTCOME, COUNT(LANDING_OUTCOME) as COUNT FROM SPACEXTBL \
     where date BETWEEN '04-06-2010' and '20-03-2017' \
     group by LANDING_OUTCOME \
     ORDER BY COUNT(LANDING_OUTCOME) DESC
```

```
 * sqlite:///my_data2.db
Done.
```

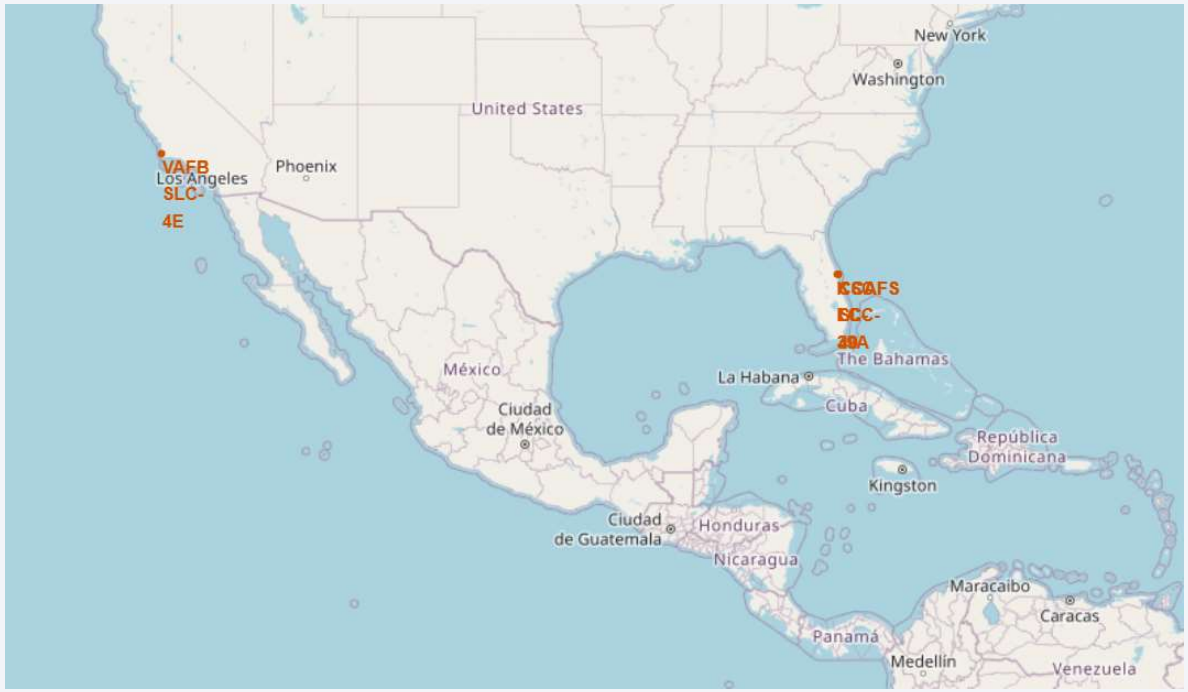
Landing_Outcome	COUNT
Success	20
No attempt	10
Success (drone ship)	8
Success (ground pad)	6
Failure (drone ship)	4
Controlled (ocean)	3
Failure	3
Failure (parachute)	2
No attempt	1

We applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome in descending order

Section 3

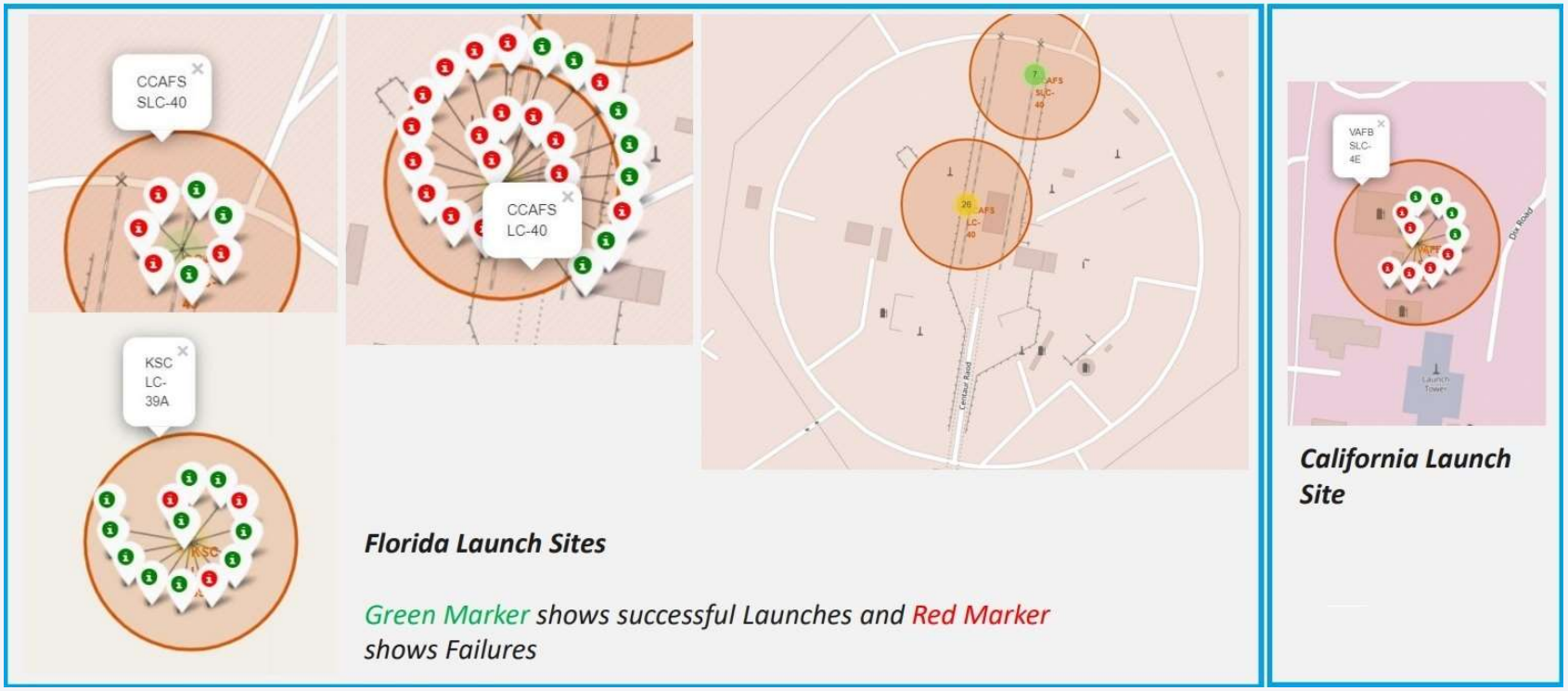
Launch Sites Proximities Analysis

Location of all the Launch Sites

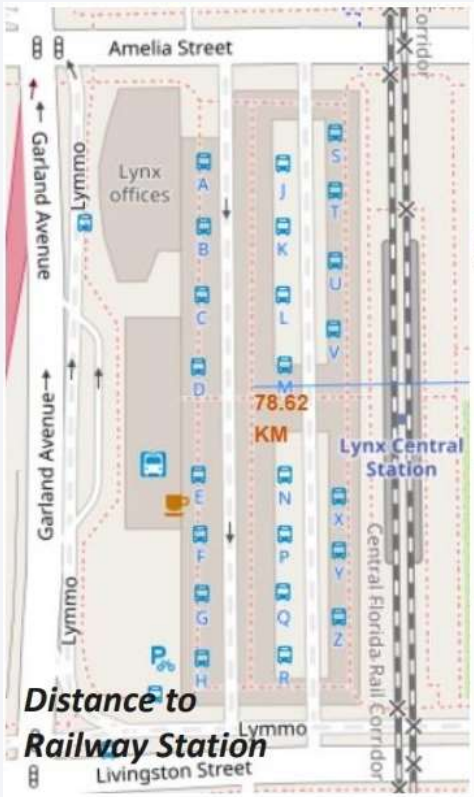


All SpaceX launch sites are in the United States of America coasts. In Florida and California

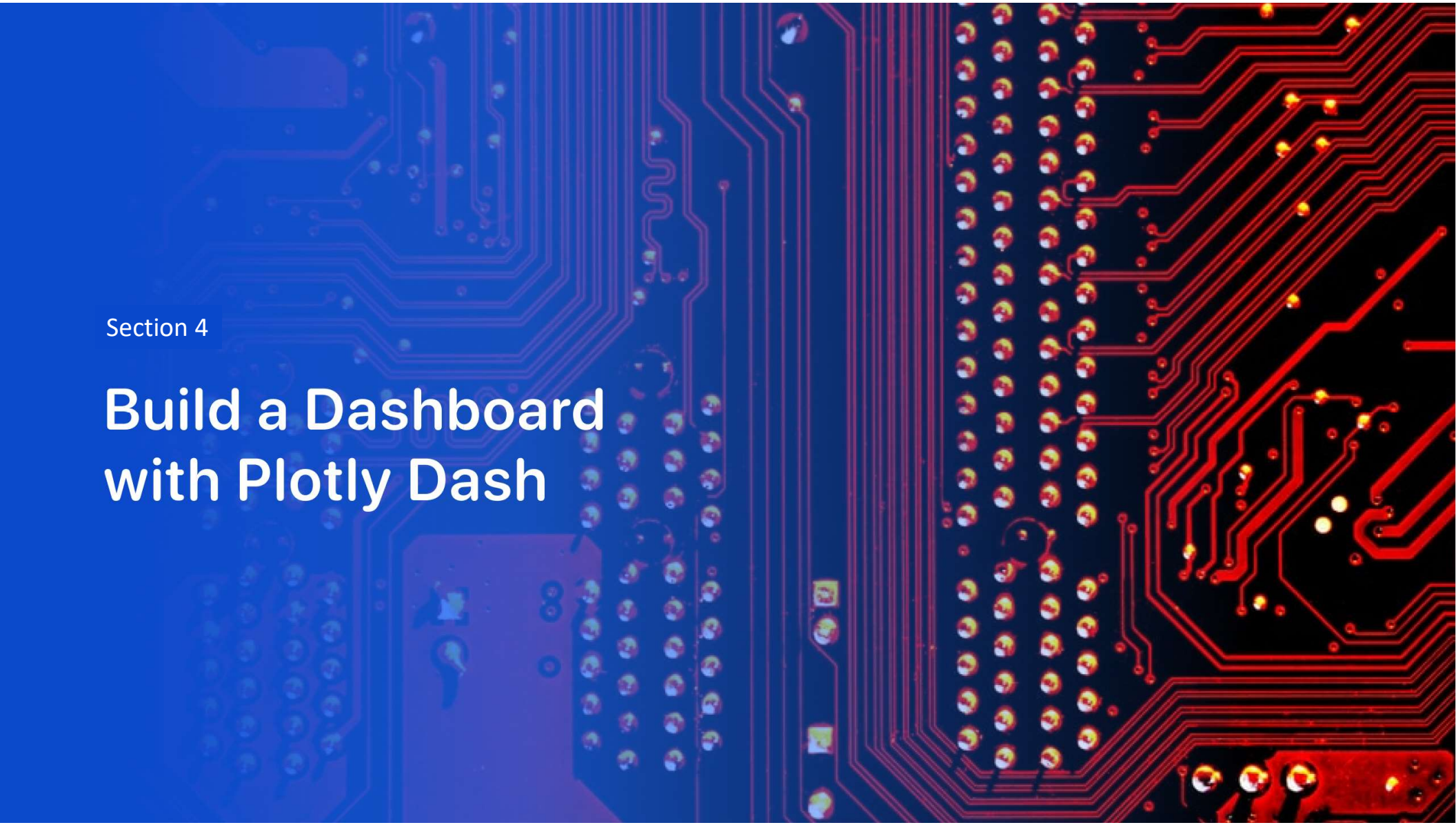
Markers showing Launch Sites with color labels



Launch Site distance to landmarks



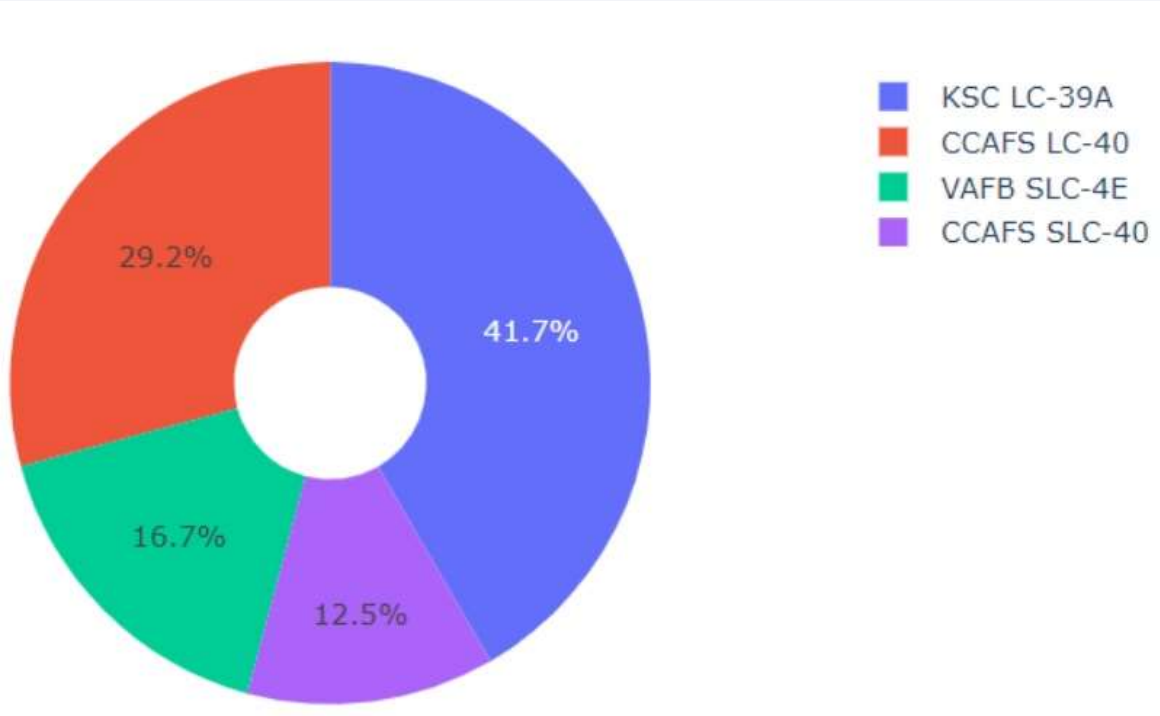
- Are all launch sites in proximately to Equator line?
  No (distance between 3000 t0 4000 km)
- Are launch sites in close proximately to railways?
  Yes (distance 0.5 to 2 km)
- Are launch sites in close proximately to highway ?
  No (distance 5 to 15 km)
- Are launch sites in close proximately to coastline?
  Yes (distance 0.5 to 5 km)
- Do launch sites keep distance away from cities?
  Yes (distance 15 to 80 km)



Section 4

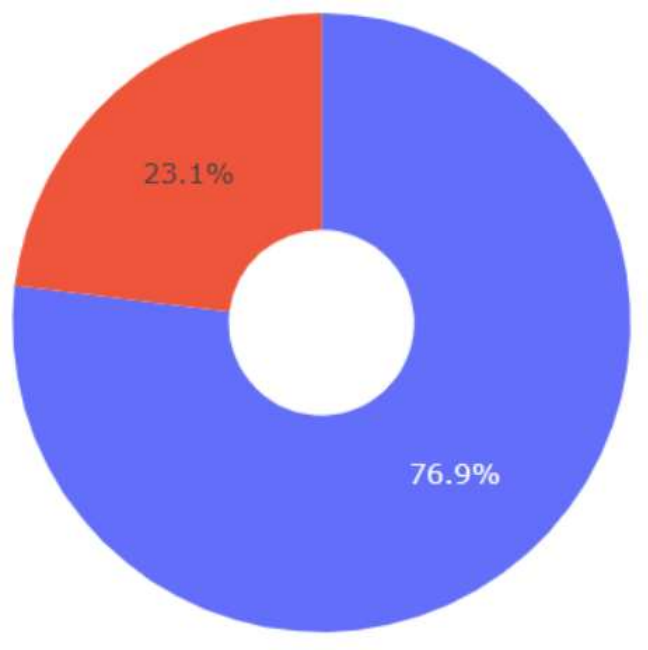
Build a Dashboard with Plotly Dash

Dashboard : Success percentage by all sites



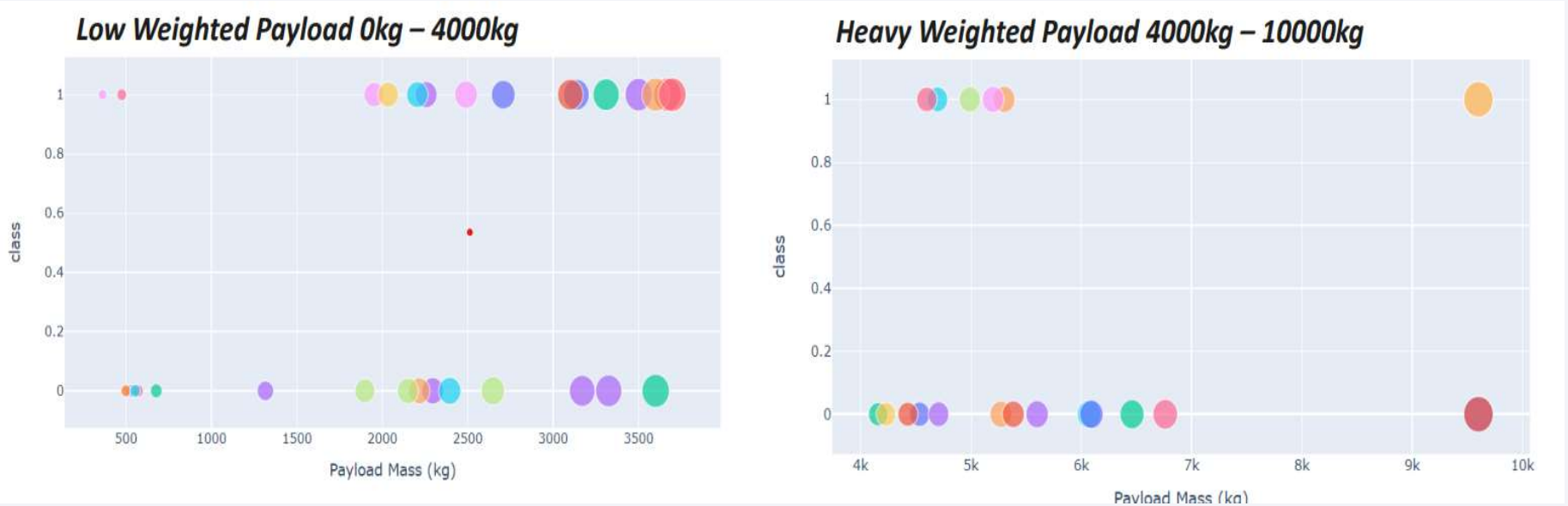
It shows KSC LC – 39A had highest % of successful launches

Dashboard : Launch success ratio of site KSC LC – 39A



KSC LC – 39A achieved a 76.9 % success rate.

Dashboard : Payload Vs. Launch Outcome Scatter plot for all sites



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

Section 5

Predictive Analysis (Classification)

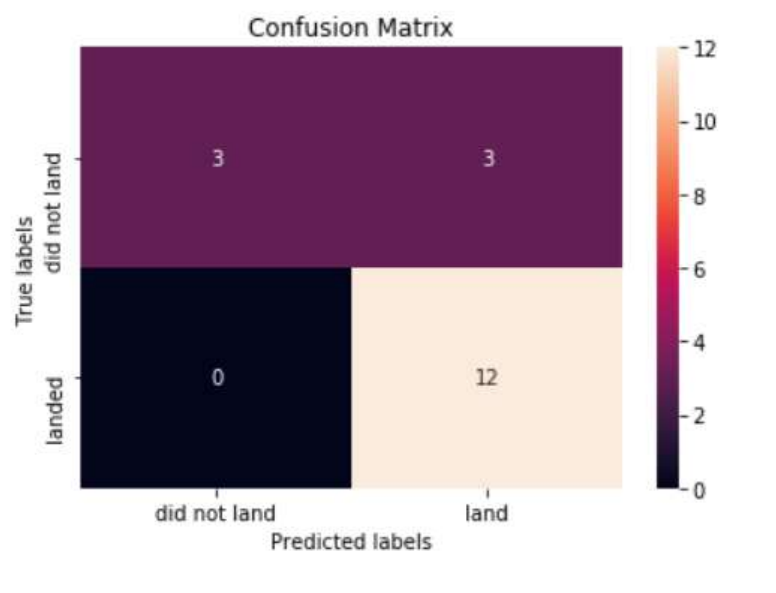
Classification Accuracy

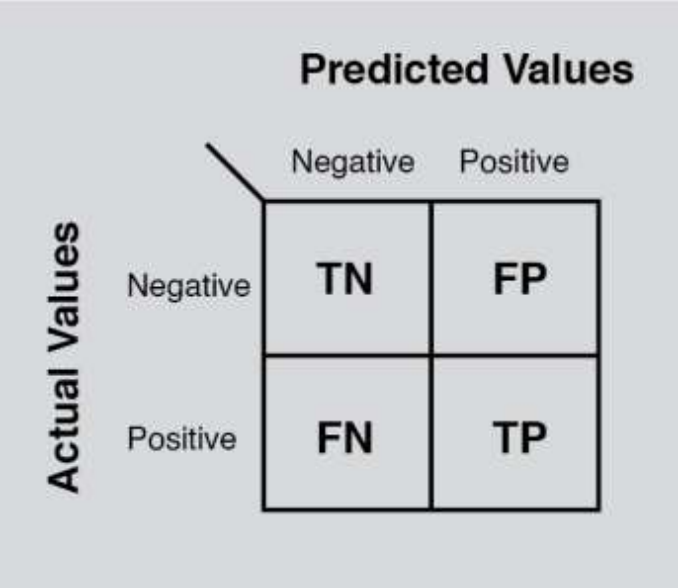
```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
```

```
Best Algorithm is Tree with a score of 0.9017857142857142
Best Params is : {'criterion': 'entropy', 'max_depth': 10, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_sampl
es_split': 10, 'splitter': 'random'}
```

As we can see with highest classification accuracy, Decision Tree Algorithm is the best algorithm.

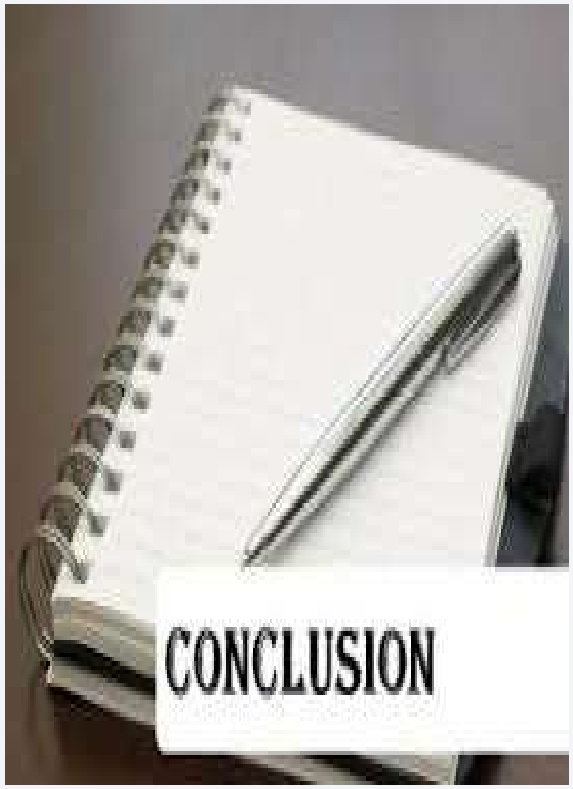
Confusion Matrix





As we see in the confusion matrix, the Tree can distinguish between the different classes. We see that the major problem is **false positives**

Conclusions



- Orbit ES-L1, GEO, HEO, and SSO has highest success rates.
- Low weighted payloads perform better than the heavier payloads.
- KSC LC-39A had the most successful launches from all the sites.
- But increasing payload mass seems to have negative impact on its success.
- The success rates for SpaceX launches has been increasing relatively with time.
- The Decision Tree Classifier Algorithm is the best for Machine Learning for
- given dataset.

Thank you!