IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Naga Sravanthi Pinnadhari
09-03-2023

Outline

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

Executive Summary

- **Summary of methodologies**
  - Data collection through API and web scraping
  - Data wrangling
  - Exploratory Data Analysis (EDA) with SQL and Data Visualization
  - Interactive Visual Analytics with Folium
  - Dashboard creation using Plotly
  - Predictive Analysis
- **Summary of all results**
  - Results from EDA and Visualization
  - Predictive Analysis results

Introduction

- **Project background and context**

In this project, we will predict if the Falcon 9 first stage will land successfully to determine the cost of the launch. Space X 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 because Space X can reuse the first stage. This information can be used if an alternate company wants to bid against space X for a rocket launch.

- **Problems you want to find answers**

  - Factors that determine the successful landing of the rocket
  - The impact of the relationship between the variables on the success or failure of the landing
  - The conditions which will allow SpaceX to achieve the best landing success results.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX API and through web scraping from Wikipedia
- Perform data wrangling
  - One-hot-encoding is used for categorical features
  - Creating a landing outcome label
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Build, tune, and evaluate classification models

Data Collection

SpaceX launch data was collected from SpaceX API using get request method

Data was then decoded using .json( ) and turned into a Pandas data frame using .json_normalize( ) and cleaned the data by checking for missing values and filling in the missing values using .mean( ) wherever necessary

Additional data was collected using Web scraping on Falcon 9 launch records from Wikipedia. These HTML tables were parsed using BeautifulSoup and converted it into a Pandas data frame

Data Collection – SpaceX API

1.Getting response from API

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

```
response = requests.get(spacex_url)
```

2. Convert response into JSON File

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

```
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}
```

3. Get the necessary details using the below functions and create a dictionary with data

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

4. Create a dataframe and filter it

```
data = pd.DataFrame({key:pd.Series(value) for key, value in launch_dict.items()})
```

```
data_falcon9 = data[data['BoosterVersion']!='Falcon 1']
```

Git hub API

Data Collection - Scraping

1. Get response from HTML

```
response = requests.get(static_url)
```

2. Create BeautifulSoup object

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

3.Find all tables

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

4. Get all the column names

```
for th in first_launch_table.find_all('th'):
    name = extract_column_from_header(th)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

5. Create Dictionary and convert it into a dataframe

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

```
# Let's initial the launch_dict with each va
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'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

Data Wrangling

In the data set, there are several different cases where the booster did not land successfully.

- True Ocean, True RTLS, and True ASDS – mission outcomes successful;
- False Ocean, False RTLS, False ASDS – mission outcomes unsuccessful.

Here, we need to transform string variables into categporical variables where is 1 for success and 0 for failure.

1. Calculate launch number for each site

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

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

2. Calculate the number and occurrence

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

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

3. Calculate the number and occurrence of mission outcome

```
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
```

4. Create a landing outcome label from Outcome column

```
landing_class = []
for key,value in df['Outcome'].items():
    if value in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
df['Class']=landing_class
df[['Class']].head(8)
```

5. Export it to a CSV

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

Github link

EDA with Data Visualization

We used three types of graphs: Scatter(to show correlation between variables) , Bar(to show relationship between numeric and categorical variables and line(to show the trend of the data variables) graphs

Scatter:

Flight Number vs Payload Mass

Flight Number vs Launch site

Payload vs Launch Site

Orbit vs Flight Number

Payload vs Orbit Type

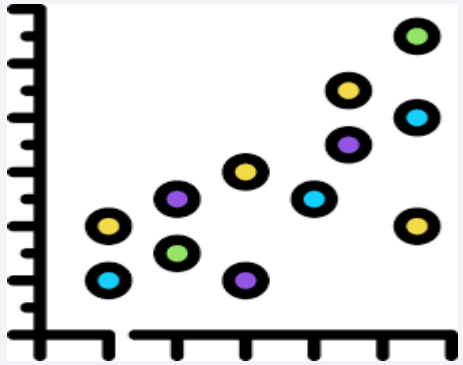
Orbit vs Payload mass

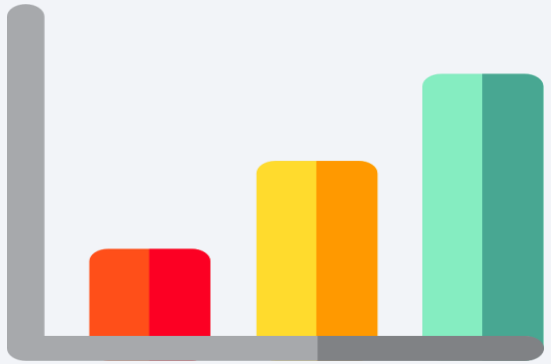
Bar Graph:

Success rate vs Orbit

Line Graph:

Success Rate vs Year





Github link

EDA with SQL

We performed certain SQL queries to gather insight from the data for EDA. Few of these queries are:

- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA’
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome in ground pad was acheived.
- List the total number of successful and failed mission outcomes

The link to this notebook is Github link SQL EDA

Build an Interactive Map with Folium

- We created a Folium Map object with NASA Johnson Space Center in Houston, Texas as the initial center location
- We used Circle and marker map objects to create a red circle with a label showing the location name
- The same has been done for all the launch sites to indicate their location and labels showing the location name
- We used Marker Cluster to display multiple points for the same coordinates by grouping the points in a cluster. This gives the total number of landings at a given location.
- Markers have been used to show successful and unsuccessful landings. Green indicates success and Red indicates failure.
- Markers are used again to show the distance from launch site location to the nearest railway, highway, coast, and nearest city and a line has been plotted between them
- These objects effectively provide information about launch sites, number of succeeded and failed landings, their vicinity to the nearby coast. Refer to the link: Git hub Folium

Build a Dashboard with Plotly Dash

- Dashboard with dropdown, range slider, pie chart and scatter plot with Plotly Dash
- The dropdown lets the user choose all launch sites or a specific launch site
- The range slider lets the user select payload mass in a fixed range.
- The corresponding pie-chart and scatter plot displays the information with respect to the launch site and the payload mass range selected. Thereby aids in a better understanding of the landings.

Refer to the link : Git hub Plotly Dash

Predictive Analysis (Classification)

Data Preparation:

- Load Dataset
- Normalize
- Split into train and test sets

Model preparation:

- GridSearchCV to tune hyperparameters
- Training models using GridSearch with training dataset

Model Evaluation:

- Get best hyperparameters for each type of model
- Compute accuracy for each model with test dataset
- Plot confusion matrix

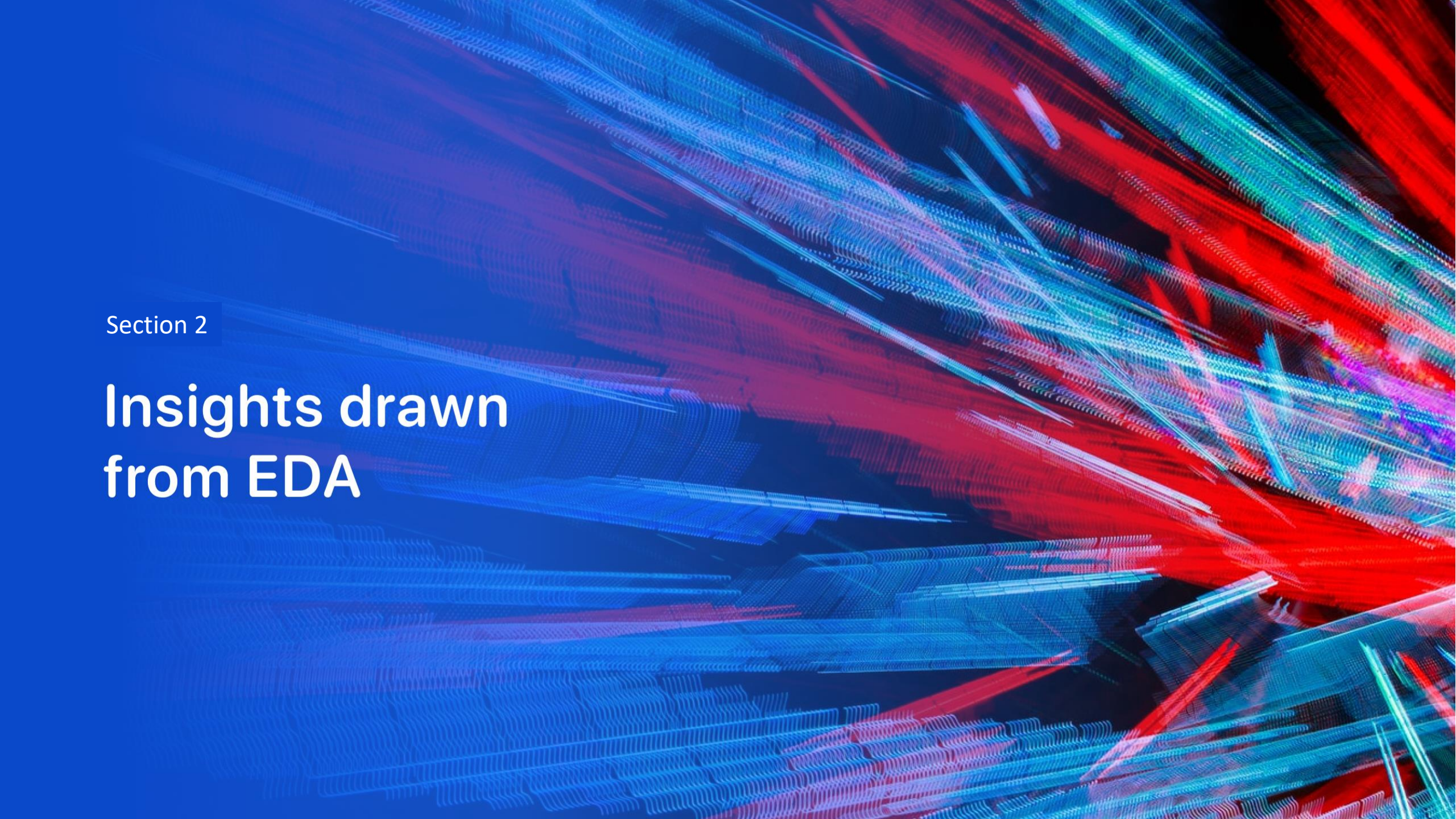
Model Comparison:

- Comparing models based on the accuracy metric

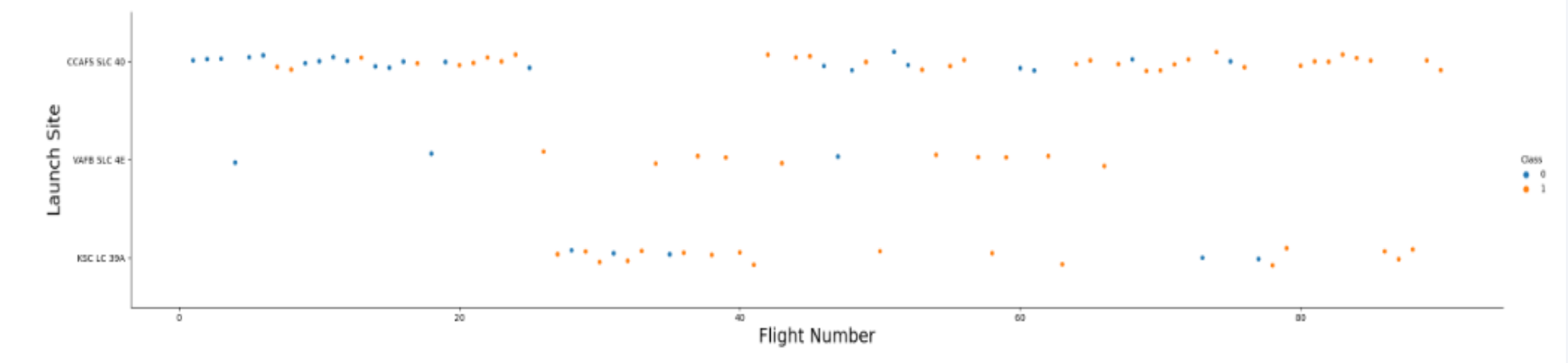
The best model was chosen based on the results : Refer Git hub Prediction Analysis

Results

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

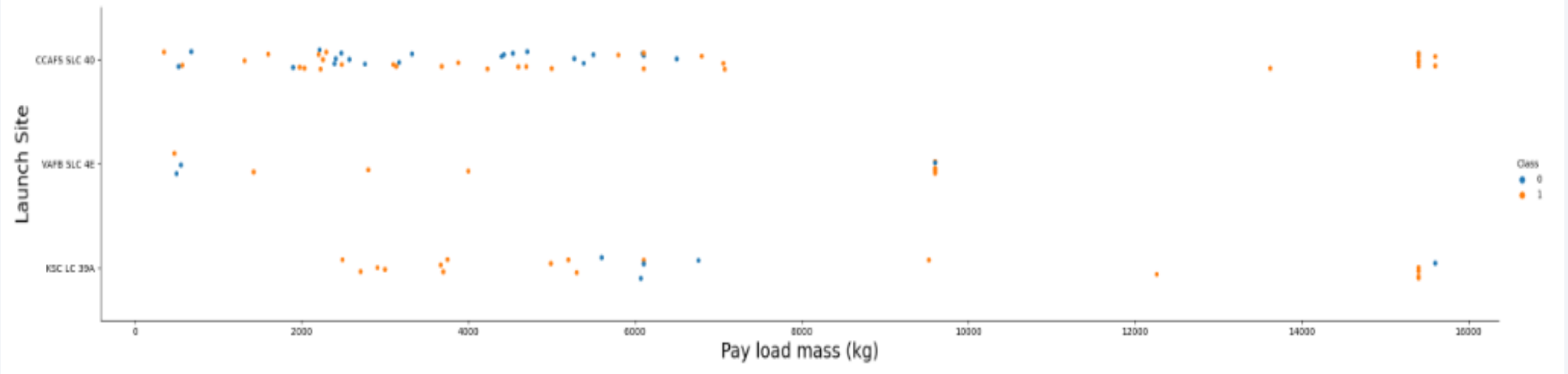

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



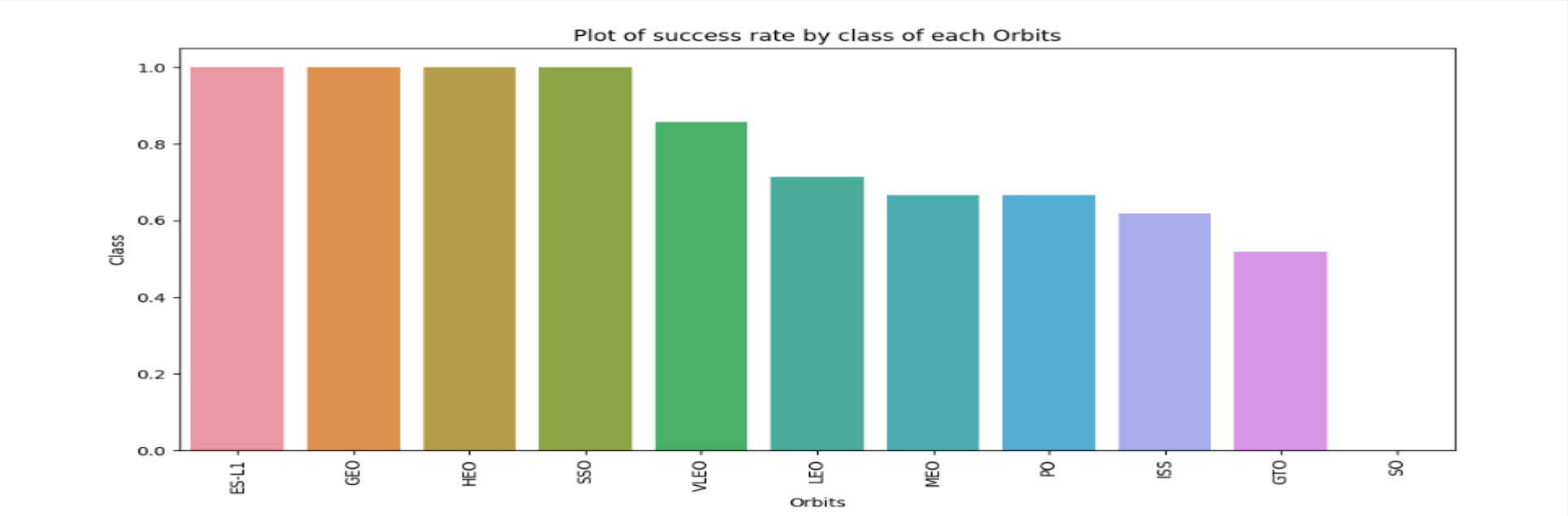
From the plot, we can infer that as the flight number at each site kept increasing, the success rate increased.

Payload vs. Launch Site



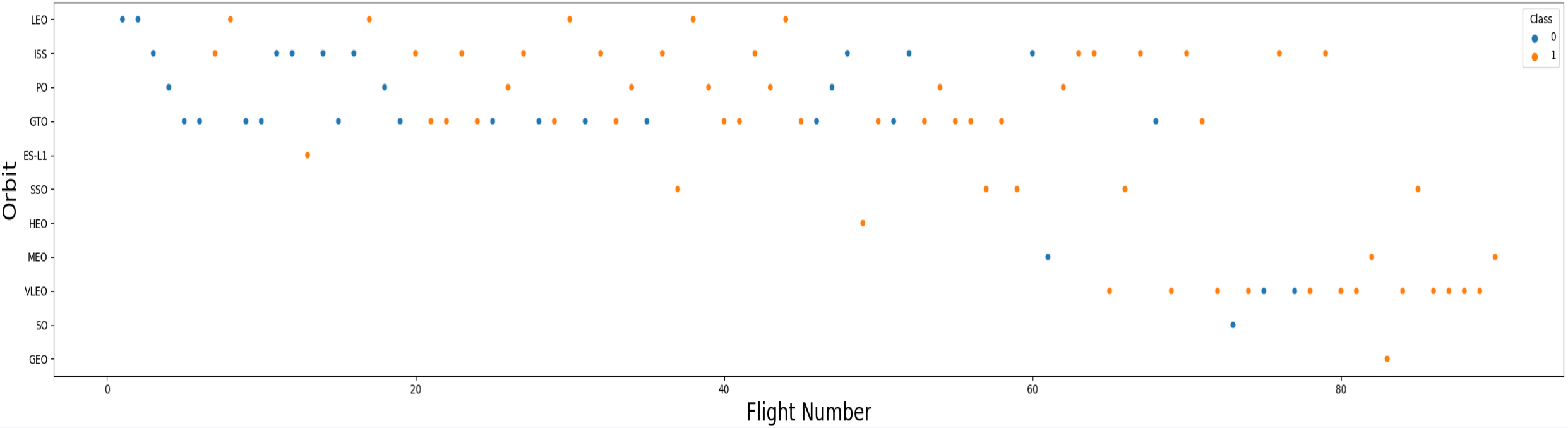
From the plot, we can see that the greater the payload mass, the higher the success rate of the rocket.

Success Rate vs. Orbit Type



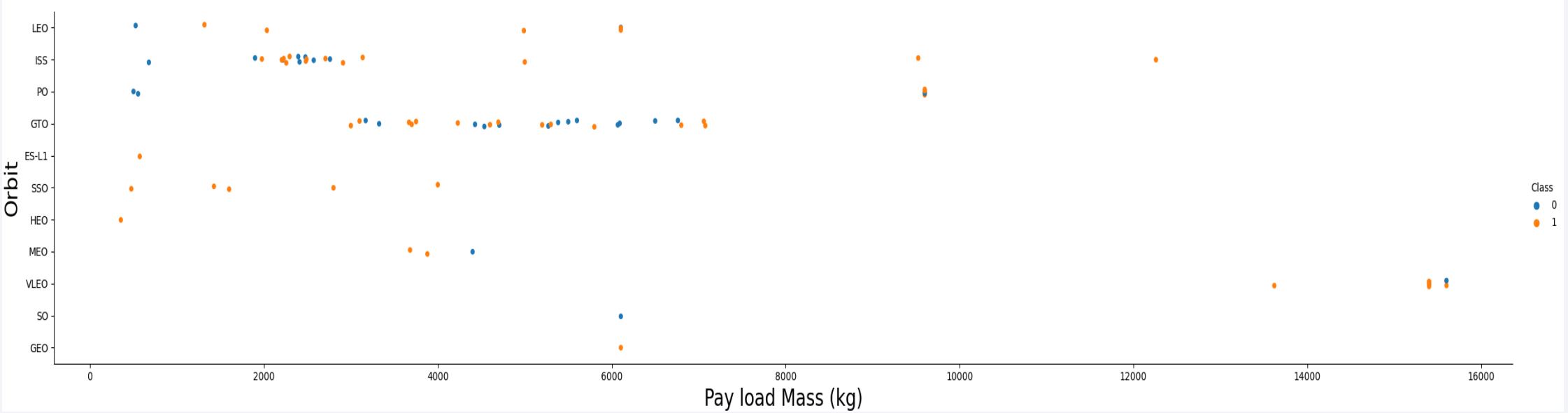
From the plot, we can see that ES-L1, GEO, HEO, SSO, and VLEO have the best success rate

Flight Number vs. Orbit Type



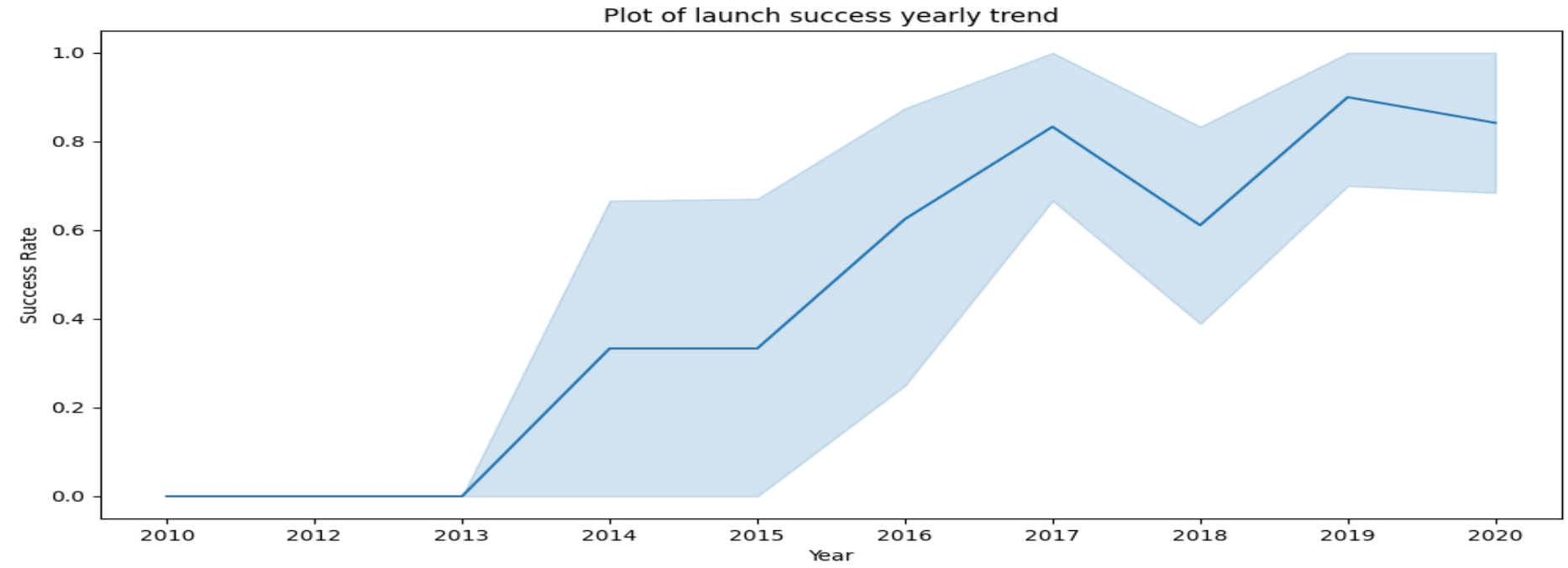
The success rate for LEO increased as the number of flights increased. Whereas for GTO, there is no relation between the number of flights and orbit. For SSO, VLEO the success rate is high as the number of flights increased.

Payload vs. Orbit Type



From the plot, we see that with heavy payloads, the landings are successful for PO, ISS, LEO

Launch Success Yearly Trend



From the plot, we see that the success rate increased from the year 2013 to 2020

All Launch Site Names

```
]: %sql SELECT DISTINCT "LAUNCH_SITE" FROM SPACEXTBL

 * sqlite:///my_data1.db
Done.
```

]:

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

The usage of DISTINCT in the query allows us to remove duplicate launch sites.

Launch Site Names Begin with 'CCA'

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

```
 * sqlite:///my_data1.db
Done.
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
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

Here, the usage of LIKE with wildcard character % enabled to get the results pertaining to CCAFS LC – 40 and the LIMIT enabled to limit the result set to 5 results.

Total Payload Mass

Display the total payload mass carried by boosters launched by NASA (CRS)

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

```
 * sqlite:///my_data1.db
Done.
```

SUM("PAYLOAD_MASS__KG_")
45596

Here we used SUM in the query to get the total payload mass

Average Payload Mass by F9 v1.1

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

```
 * sqlite:///my_data1.db
Done.
```

AVG("PAYLOAD_MASS__KG_")
2928.4

The usage of AVG in the query enabled us to get the average payload mass

First Successful Ground Landing Date

```
%sql SELECT MIN("DATE") FROM SPACEXTBL WHERE "Landing _Outcome" LIKE '%Success%'
```

```
 * sqlite:///my_data1.db
Done.
```

MIN("DATE")
01-05-2017

The usage of MIN in the query here enabled us to filter out the first ground landing date along with LIKE and wildcard character %SUCCESS%

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 SPACEXTBL WHERE "LANDING _OUTCOME" = 'Success (drone ship)' \
AND "PAYLOAD_MASS__KG_" > 4000 AND "PAYLOAD_MASS__KG_" < 6000;
```

```
 * sqlite:///my_data1.db
Done.
```

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

We used AND here to provide the range of payload mass to filter out the result set

Total Number of Successful and Failure Mission Outcomes

```
%sql SELECT (SELECT COUNT("MISSION_OUTCOME") FROM SPACEXTBL WHERE "MISSION_OUTCOME" LIKE '%Success%') AS SUCCESS, \
(SELECT COUNT("MISSION_OUTCOME") FROM SPACEXTBL WHERE "MISSION_OUTCOME" LIKE '%Failure%') AS FAILURE
```

```
 * sqlite:///my_data1.db
Done.
```

SUCCESS	FAILURE
100	1

We used subquery here. The first subquery SELECT gives us the Success count and the second one gives us the failure count

Boosters Carried Maximum Payload

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

```
 * sqlite:///my_data1.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 used subquery here with MAX to filter out only the maximum payload mass

2015 Launch Records

```
%sql SELECT substr("DATE", 4, 2) AS MONTH, "BOOSTER_VERSION", "LAUNCH_SITE" FROM SPACEXTBL\
WHERE "LANDING _OUTCOME" = 'Failure (drone ship)' and substr("DATE",7,4) = '2015'
```

```
 * sqlite:///my_data1.db
Done.
```

MONTH	Booster_Version	Launch_Site
01	F9 v1.1 B1012	CCAFS LC-40
04	F9 v1.1 B1015	CCAFS LC-40

Here we used substr function. It processes date in order to take month or year.

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

```
%sql SELECT "LANDING _OUTCOME", COUNT("LANDING _OUTCOME") FROM SPACEXTBL\
WHERE "DATE" >= '04-06-2010' and "DATE" <= '20-03-2017' and "LANDING _OUTCOME" LIKE '%Success%'\
GROUP BY "LANDING _OUTCOME" \
ORDER BY COUNT("LANDING _OUTCOME") DESC ;
```

```
 * sqlite:///my_data1.db
Done.
```

Landing _Outcome	COUNT("LANDING _OUTCOME")
Success	20
Success (drone ship)	8
Success (ground pad)	6

Here we used COUNT to get the total number and GROUP BY along with column name to specify the group and ORDER set to DESC to get the result set in descending order.

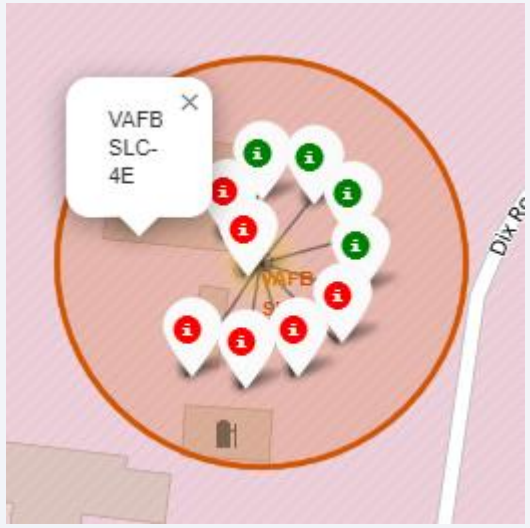
Section 3

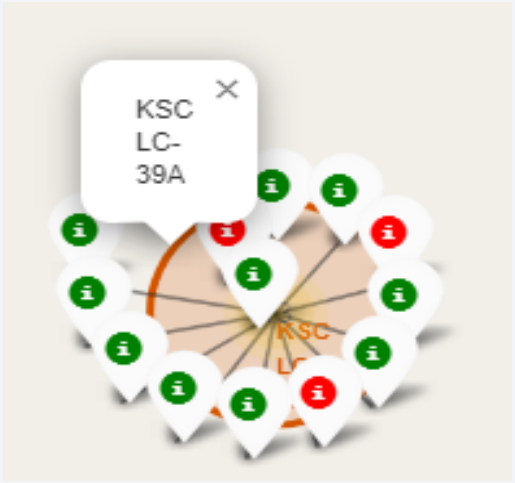
Launch Sites Proximities Analysis

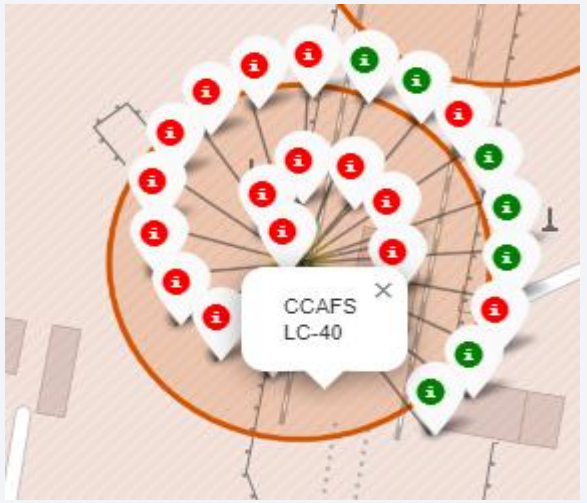
All Launch Sites

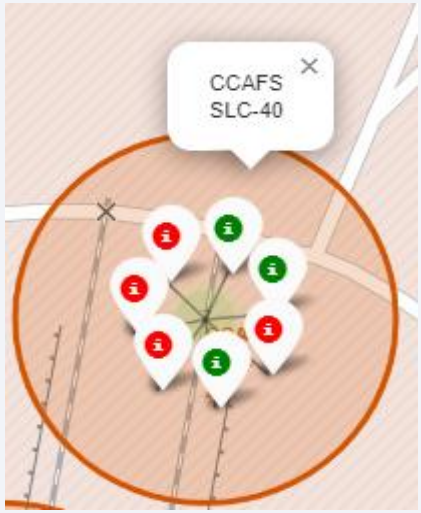
We see that the launch sites are in the US and in the vicinity of coast.

Colored Markers







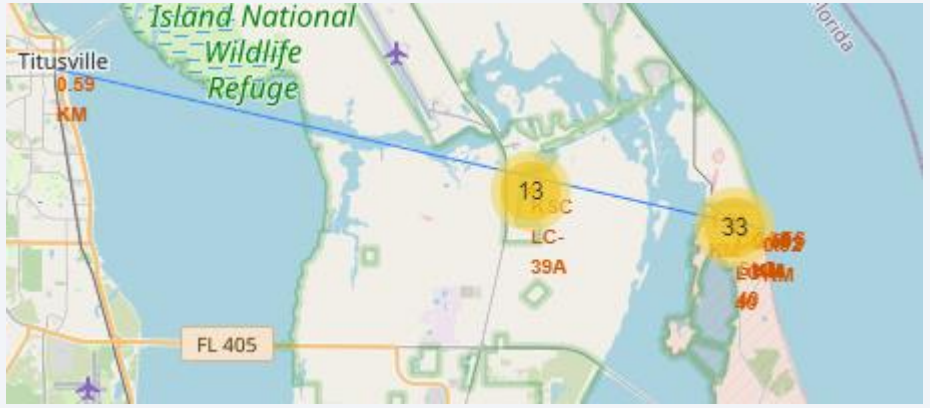


Here, the green marker shows the successful launches

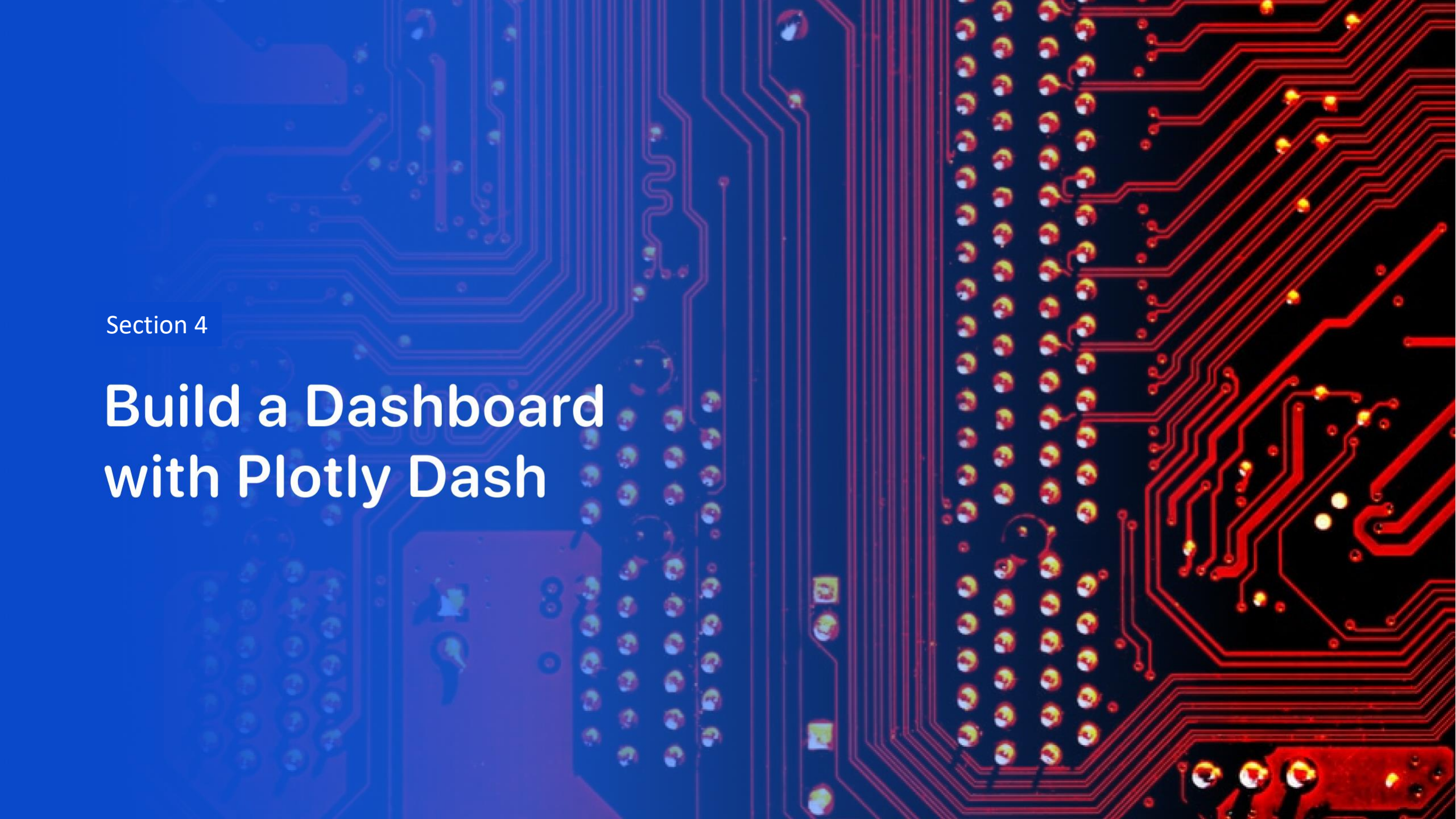
Whereas the red shows unsuccessful launches

We see that the site KSC LC -39A has higher success rate

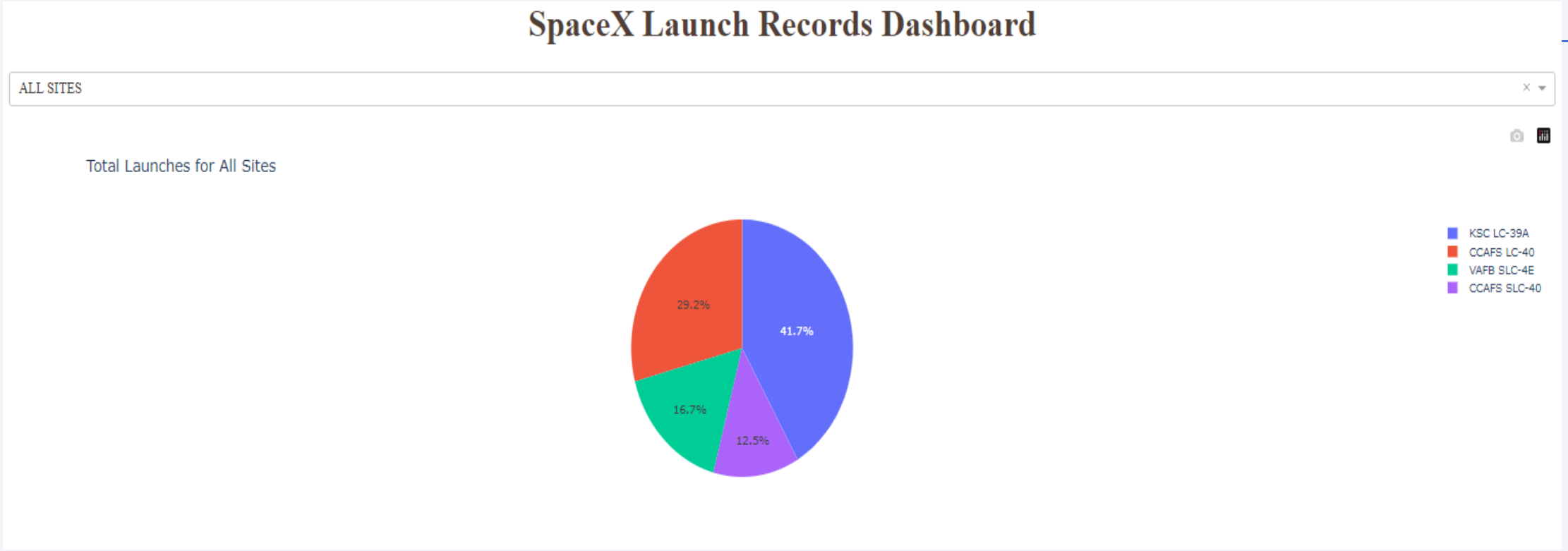
Distance of nearest Coast, Highway, Railway and City



From the first image we can see the line drawn to nearest coast, railway and highway from launch site and from the second figure we can see the line drawn from the launch site to the nearest city

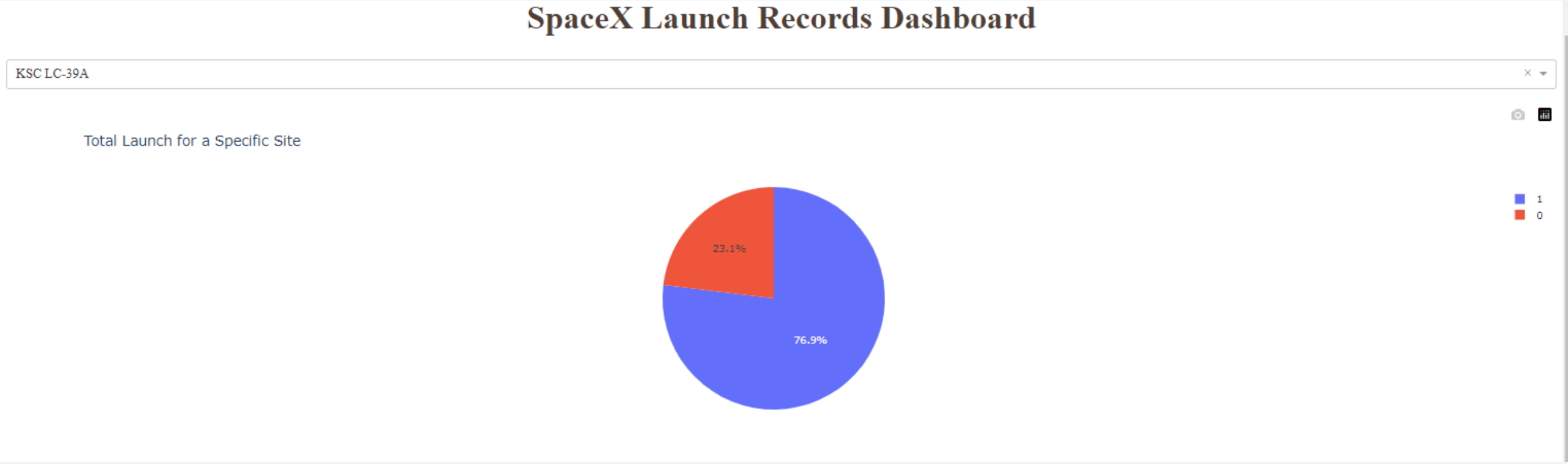

Section 4
Build a Dashboard with Plotly Dash

Dashboard - All Sites Pie chart



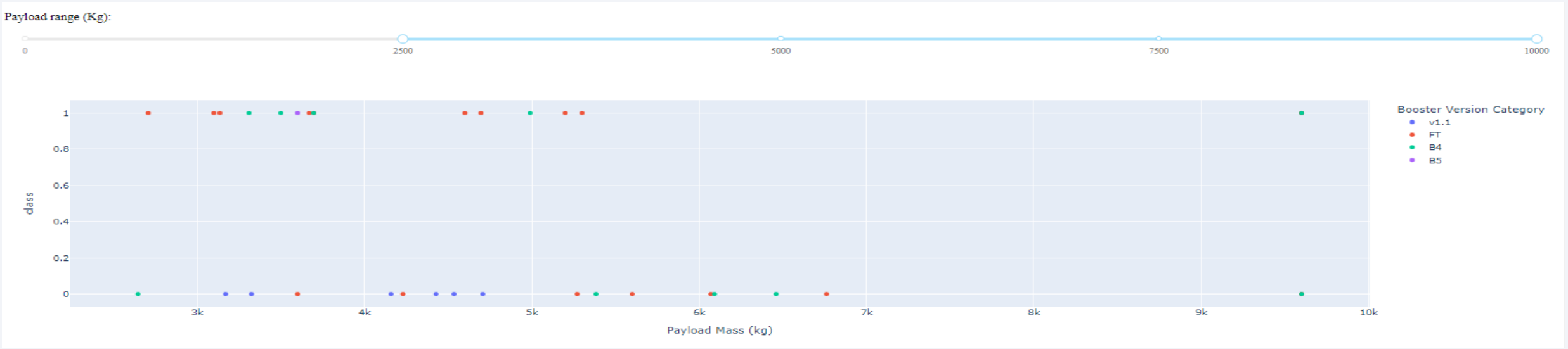
We see that the KSC LC – 39A has the highest success rate.

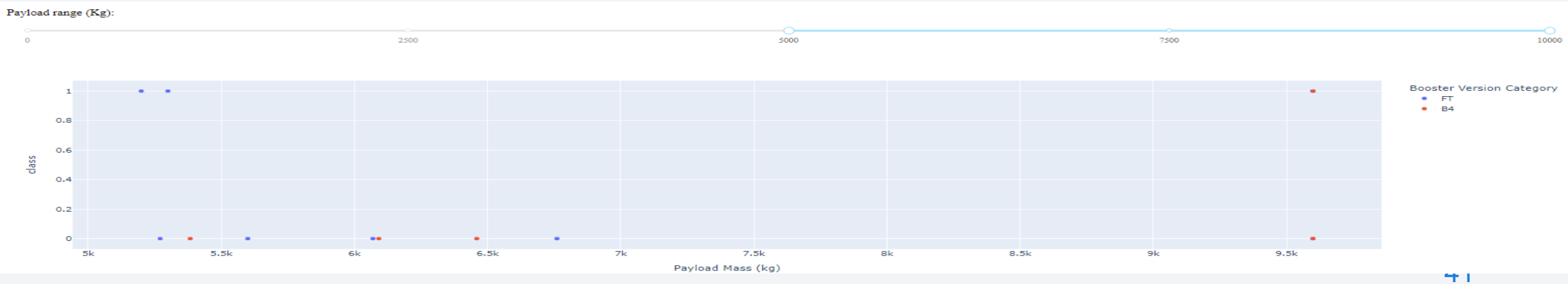
KSC LC – 39 A Launch outcome



The success rate is 76.9% whereas the failure rate is 23.1% for KSC LC – 39 A

Payload 2500 – 10000 vs Payload 5000 - 10000

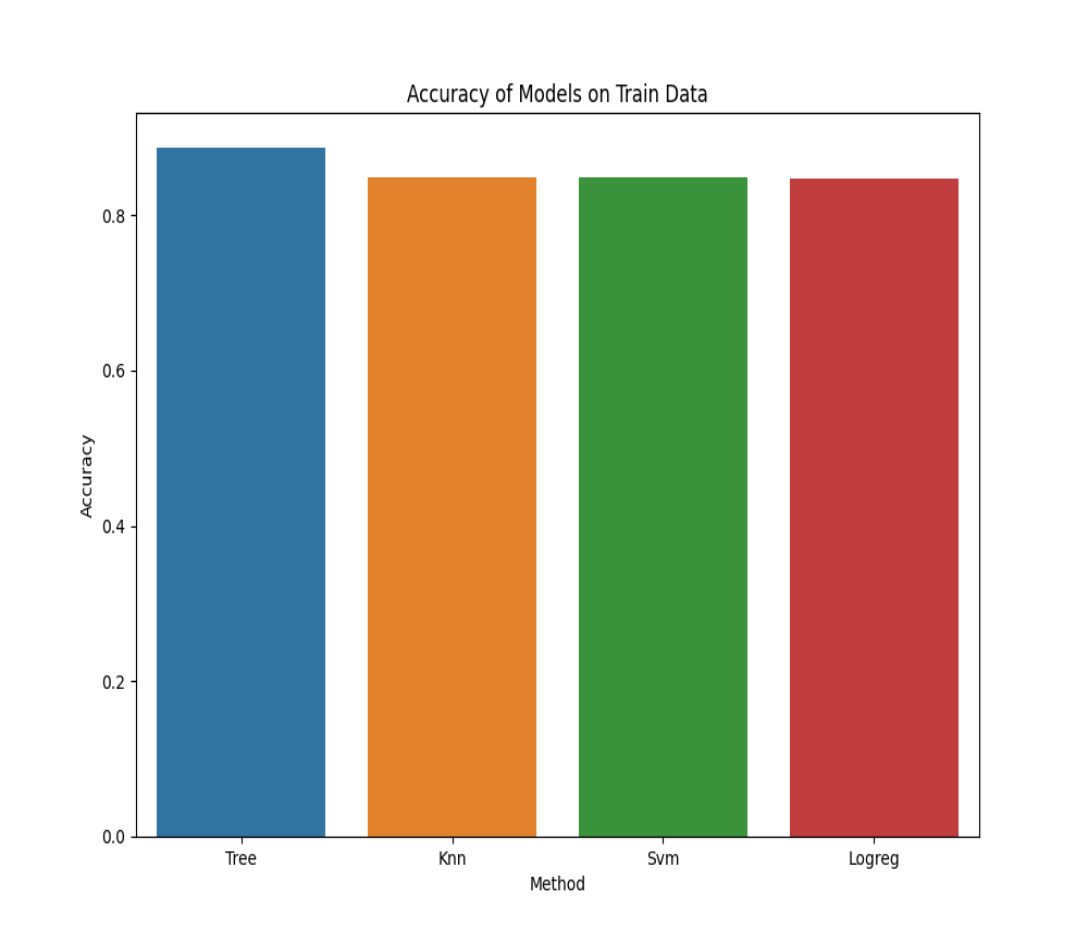


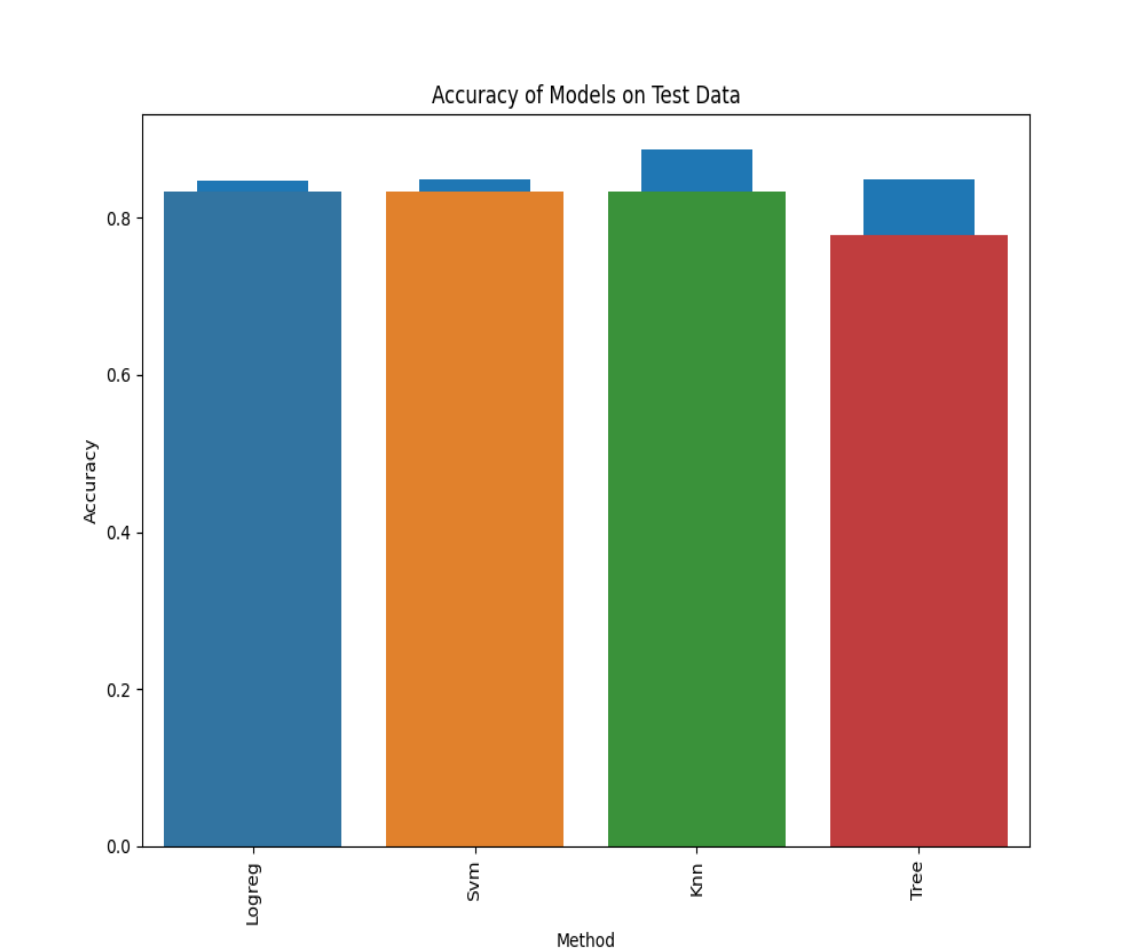


Section 5

Predictive Analysis (Classification)

Classification Accuracy





Classification Accuracy

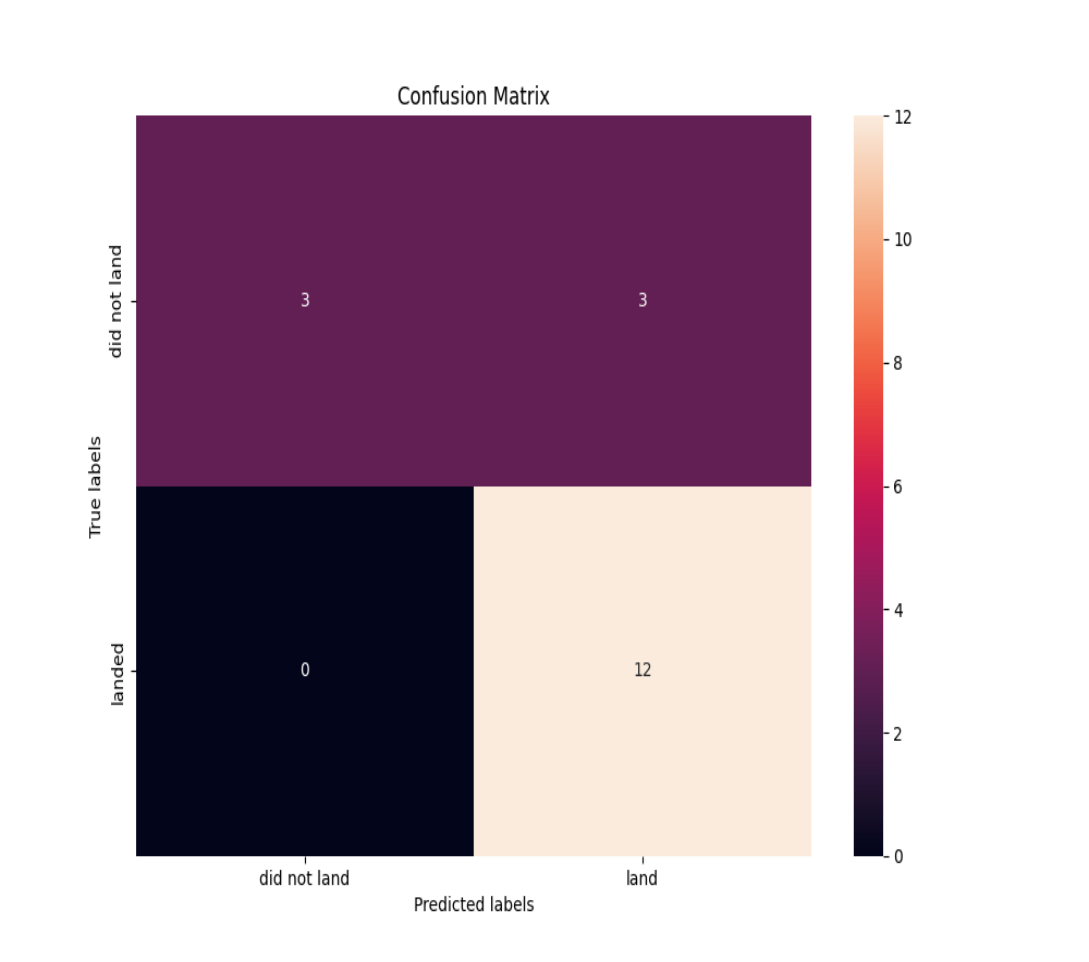
	Accuracy Train	Accuracy Test
Tree	0.887500	0.777778
Knn	0.848214	0.833333
Svm	0.848214	0.833333
Logreg	0.846429	0.833333

```
Best model is DecisionTree with a score of 0.8875
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'r
andom'}
```

From the graphs, we see that for the test set, all the models perform almost the same. From the Train set bar graph, we see that the Tree model performs better. So, from the results, we can select the Tree model as the appropriate model.

Confusion Matrix

- The confusion matrix for all the test models came out to be the same. The main issue with the confusion matrix is unsuccessful landing marked as successful by the classifier.



Conclusions

- The larger the number of flights from a launch site, the greater the success rate at the launch site.
- Success rates if launches constantly increased from the year 2013 till 2020
- Orbits GEO, HEO, SSO, VLEO, ES-L1 had the most success rate.
- The launch site KSC LC – 39A had the most successful launches
- Decision Tree Classifier is the best model for this analysis.

Thank you!