IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Sai Soum Mein
24 July 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
  - Building Dashboard with Plotly Dash
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis Result
  - Interactive Analytics in Screenshots
  - Predictive Analytics Result

Introduction

- Project background and context
  - SpaceX promotes Falcon 9 rocket launches on their website at a price of $62 million, whereas other providers charge over $165 million for each launch. The significant cost difference is primarily due to SpaceX's ability to reuse the first stage of their rockets. Consequently, if we can accurately predict whether the first stage will successfully land, we can determine the overall cost of a launch. This information is particularly valuable for alternate companies looking to compete with SpaceX in bidding for rocket launches. The objective of this project is to develop a machine learning pipeline that can effectively forecast the successful landing of the first stage.

- Questions/Problems
  - What factors influence if the rocket will launch successfully?
  - What conditions does SpaceX need to achieve to be able to get the best results?
  - The effect of relationship of each variables determining the success rate of successful landing.

Section 1

Methodology

Methodology

- Data collection methodology:
  - Data collection using SpaceX API
  - Data collection using web scraping from Wikipedia.
- Perform data wrangling
  - One-hot encoding was applied to categorical features
- Perform Exploratory Data Analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models

Data Collection

Data was collected from SPACEX API and Web Scraping used to collect the dataset

- Data Collection using REST API:
  - Initiating the process with a get request
  - Decoding the response content as JSON
  - Converting the JSON data into a pandas dataframe using json_normalize()
  - Cleaning the data and addressing missing values
- Data Collection using Wikipedia Web Scraping:
  - Utilizing BeautifulSoup to extract launch records as an HTML table
  - Parsing the table and converting it into a pandas dataframe
  - Preparing the data for further analysis

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 to JSON File

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

3. Transform Data

```
# Call getLaunchSite
getLaunchSite(data)

# Call getPayloadData
getPayloadData(data)

# Call getCoreData
getCoreData(data)
```

4. Create dictionary

```
#Global variables
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

5. Create dataframe

```
# Create a data from launch_dict
data = pd.DataFrame(launch_dict)
```

6. Filter dataframe

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = data[data['BoosterVersion']!='Falcon 1']
data_falcon9.head()
```

7. Deal with missing values

```
# Calculate the mean value of PayloadMass column
payload_mean = data['PayloadMass'].mean()

# Replace the np.nan values with its mean value
data['PayloadMass'] = data['PayloadMass'].replace(np.nan, payload_mean)

# Print the modified DataFrame
print(data)
```

8. Export to file

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

Link to Notebook

Data Collection - Web Scraping

1. Getting response from HTML

```
# use requests.get() method with the provided sta
# assign the response to a object
response = requests.get(static_url).text
```

2. Create BeautifulSoup Object

```
# Use BeautifulSoup() to create a BeautifulSoup ob
soup = BeautifulSoup(response, 'html.parser')
```

3. Find all tables

```
# Use the find_all function in the BeautifulS
# Assign the result to a list called `html_ta
html_tables = soup.find_all('table')
```

4. Get column names

```
column_names = []

# Apply find_all() function with `th` element on first
th_elements = first_launch_table.find_all('th')

# Iterate through each th element
for th_element in th_elements:
    # Apply the extract_column_from_header() function
    name = extract_column_from_header(th_element)

    # Append the non-empty column name to the list
    if name is not None and len(name) > 0:
        column_names.append(name)
```

5. Create dataframe

```
launch_dict= dict.fromkeys(column_names)

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

# Let's initial the launch_dict with each
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']=[]
```

6. Adding data to keys

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('ta
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as nu
        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')
```

7. Create dataframe

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

8. Export to file

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


Link to Notebook

Data Wrangling

In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident; for example,

- True Ocean, True RTLS, True ASDS means mission outcome was successful.
- False Ocean, False RTLS, False ASDS means mission outcome was unsuccessful(Fail).

Objective: Convert those outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful

1. Calculate the number of launches on each site

```
# Apply value_counts() on column
df.LaunchSite.value_counts()
```

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

2. Calculate number and occurrence of each orbit

```
# Apply value_counts on Orbit
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 occurence of mission outcome of the orbits

```
# Apply value_counts on Orbit
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
```

4. Create a landing outcome label from Outcome column

```
landing_class = []

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

5. Export to file

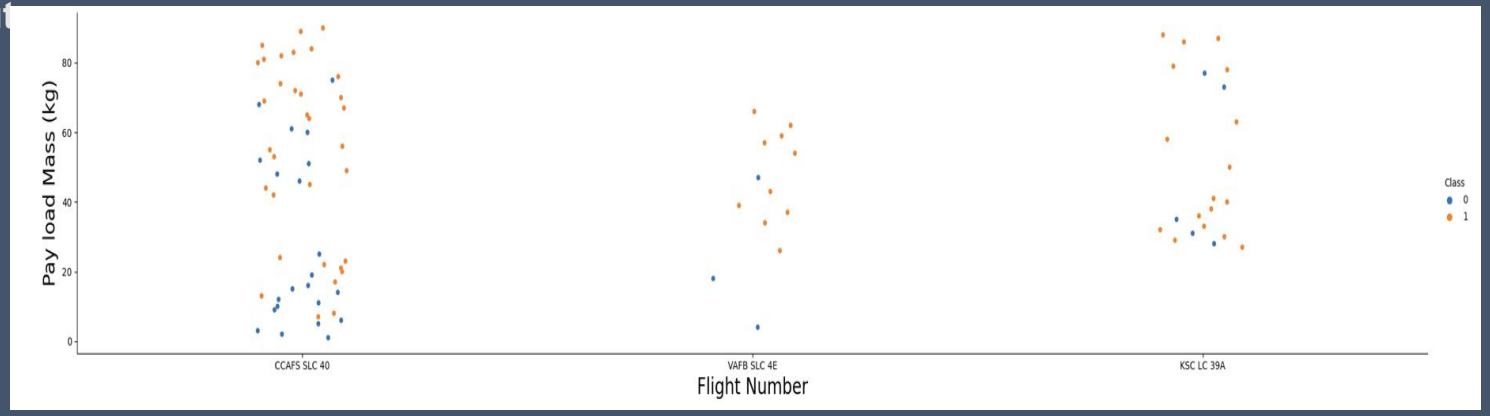
```
df.to_csv("dataset_part_2.csv", index=False)
```

Link to Notebook

EDA with Data Visualization

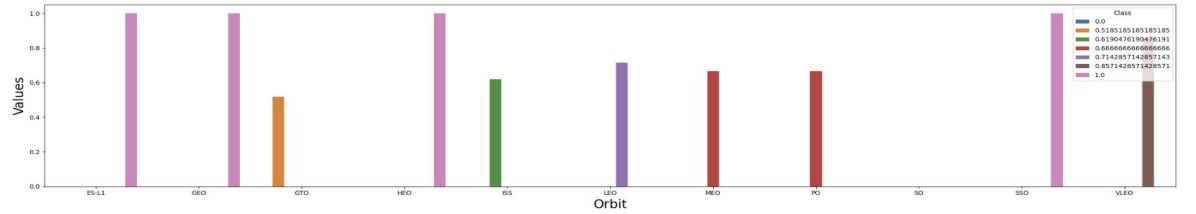
Visualized scatter plot for different attribut such as

- Flight Number and Payload Mass
- Flight Number and Launch Site
- Payload and Launch Site
- Orbit and Flight Number
- Flight Number and Orbit type
- Payload and Orbit type



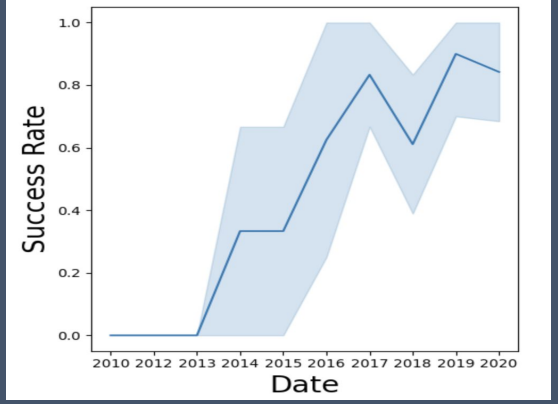
Visualized bar chart for success rate and orbit type

```
# HINT use groupby method on Orbit column and get the mean of Class column
orbit_success = df.groupby('Orbit').mean()
orbit_success.reset_index(inplace=True)
sns.barplot(x="Orbit",y="Class",data=orbit_success,hue='Class')
plt.xlabel('Orbit',fontsize=20)
plt.ylabel('Values',fontsize=20)
plt.show()
```



Visualized line chart for Success rate and year

```
# Plot a line chart with x axis to be the extracted year and y a
sns.lineplot(data=df, x="Date", y="Class")
plt.xlabel("Date",fontsize=20)
plt.ylabel("Success Rate",fontsize=20)
plt.show()
```



[Link to Notebook]

EDA with SQL

Data has been gathered by using SQL queries to get insight from the datasets.

- 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 succesful landing outcome in ground pad was acheived.
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
- 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.
- Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

Link to Notebook

Build an Interactive Map with Folium

Folium map is created where all the launch sites have been marked and added various map objects such as

- Markers, which indicate points like launch sites
- Circles, which indicate highlighted areas around specific coordinates, for example NASA Johnson Space Center
- Marker clusters, which indicate groups of events in each coordinate, for example launches in a launch site, and
- Lines, which indicate the distances between two coordinates.

This has allowed us to answer specific questions, such as whether the launch sites are located near railways, highways and coastlines.

Link to Notebook

Build a Dashboard with Plotly Dash

An interactive dashboard has been built with Plotly dash and the dashboard has dropdown, pie chart, range slider and scatter plot components.

- Dropdown allows users to choose the launch site (dash_core_components.Dropdown).
- Pie charts showing the total launches for the chosen launch site has been plotted using (plotly.express.pie).
- Range slider allows users to select pay load mass in a fixed range (dash_core_components.RangeSlider).
- Scatter graph has been plotted showing the relationship with Outcome and Payload Mass (Kg) for the different booster version using (plotly.express.scatter).

Link to Notebook

Predictive Analysis (Classification)

1. Data preparation
   - Load the dataframe
   - Create a NumPy array from the column Class in data
   - Standardize the data, then fitting and transform it
   - Split data into training and test sets using train_test_split function
2. Model preparation
   - Create a GridSearchCV object with cv = 10 to find the best parameters
   - Apply GridSearchCV on LogReg, SVM, Decision Tree, and KNN models
   - Set parameters for each algorithm to GridSearchCV and fit it to dataset
3. Model evaluation
   - Get best hyperparameters for each type of model
   - Compute accuracy for each model with test dataset using method.score()
   - Plot Confusion Matrix
4. Model comparison
   - Use Feature Engineering and Algorithm Tuning
   - The model with the best accuracy is chosen (See notebook for the results)

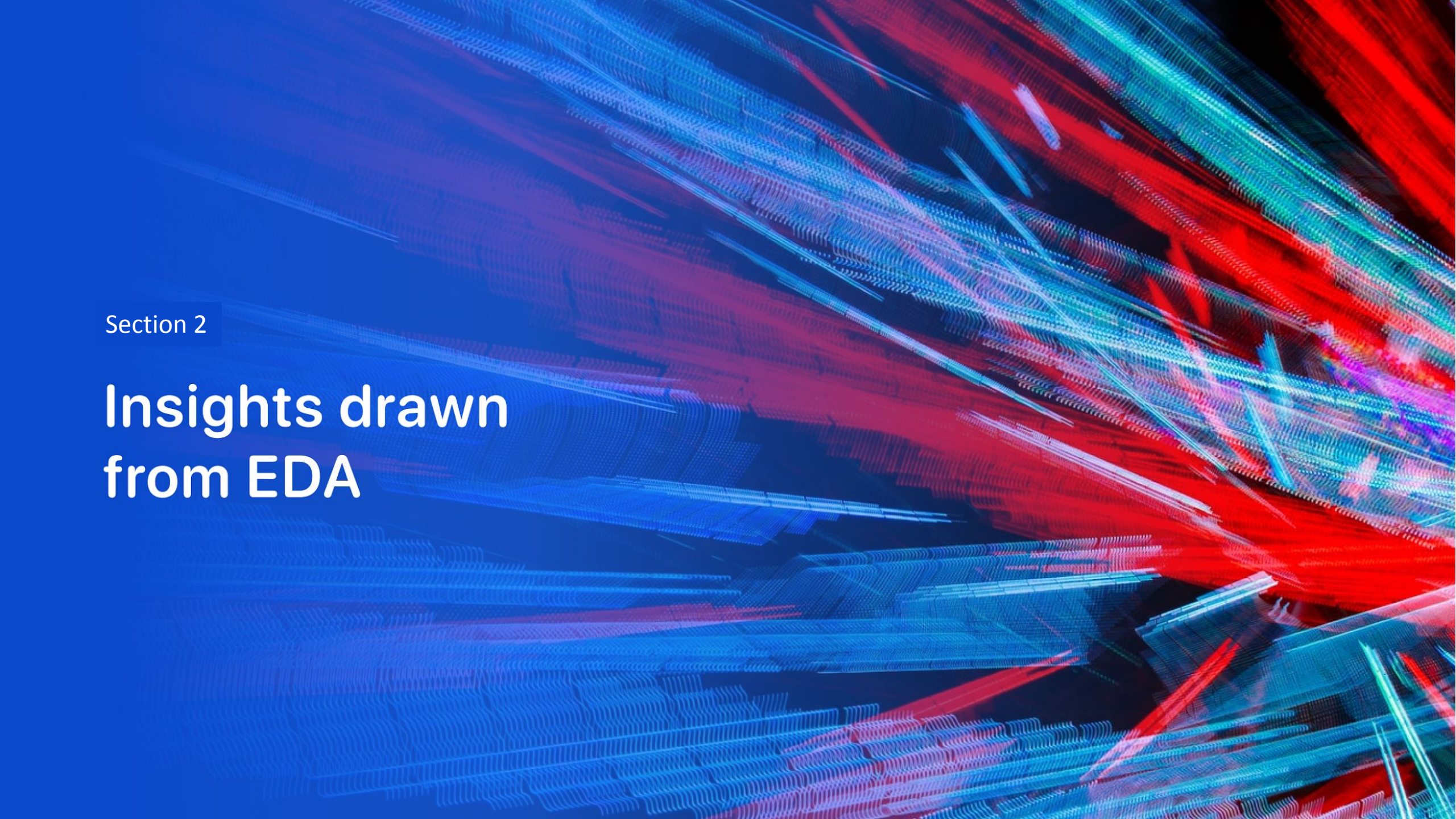
Link to Notebook

Results

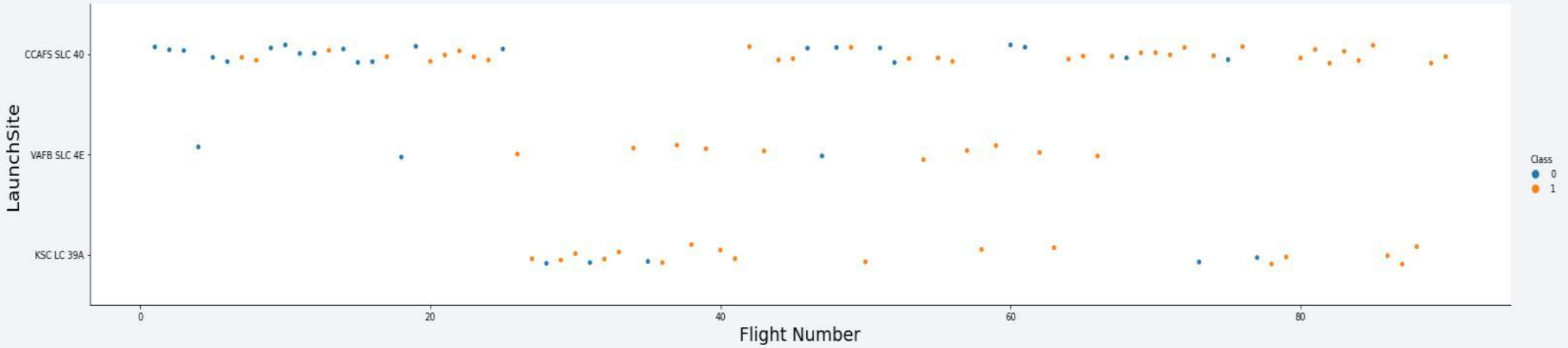
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

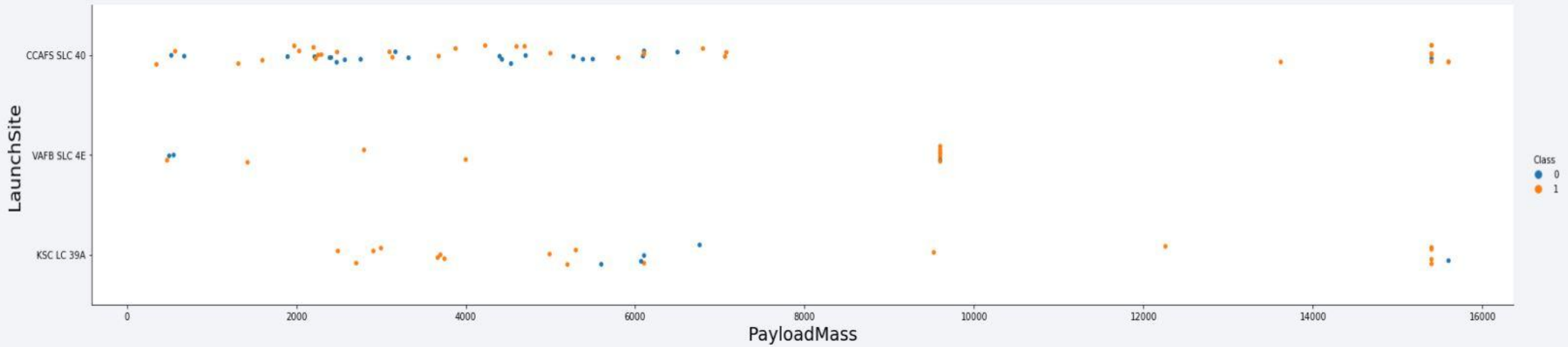


Flight Number vs. Launch Site



- Launches conducted in the past had a lower success rate, whereas those conducted recently had a higher success rate.
- Approximately half of the launches were carried out from the CCAFS SLC 40 launch site.
- VAFB SLC 4E and KSC LC 39A launch sites had a higher success rate.
- Hence, it can be concluded that the success rate of launches has improved with time.

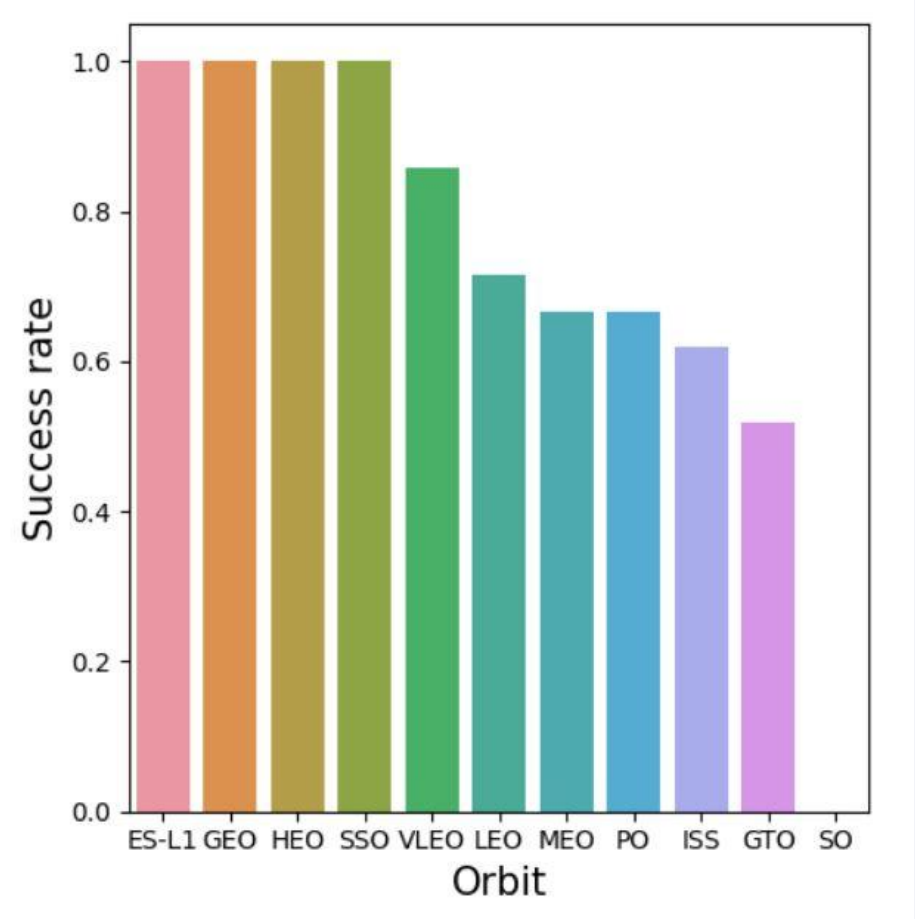
Payload vs. Launch Site



- There is a positive correlation between the payload mass (kg) and the success rate of launches.
- Payload mass greater than 7,000 kg has more successful outcome.
- KSC LC 39A launch site had a 100% success rate for launches with a payload mass less than 5,500 kg.
- VAFB SKC 4E launch site has not conducted any launch with a payload mass greater than approximately 10,000 kg.

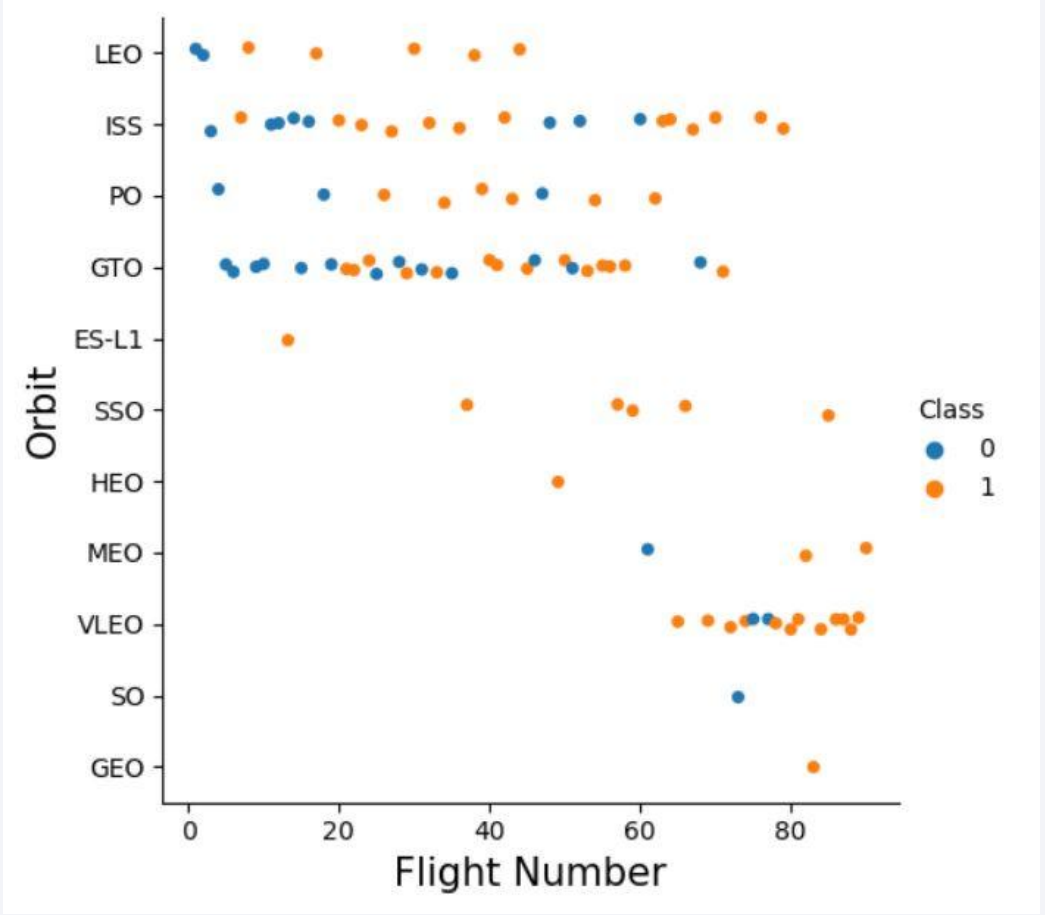
Success Rate vs. Orbit Type

- ES-L1, GEO, HEO, and SSO have 100% success rate.
- GTO, ISS, LEO, MEO, and PO had a success rate ranging from 50% to 80%.
- Orbit SO had a 0% success rate



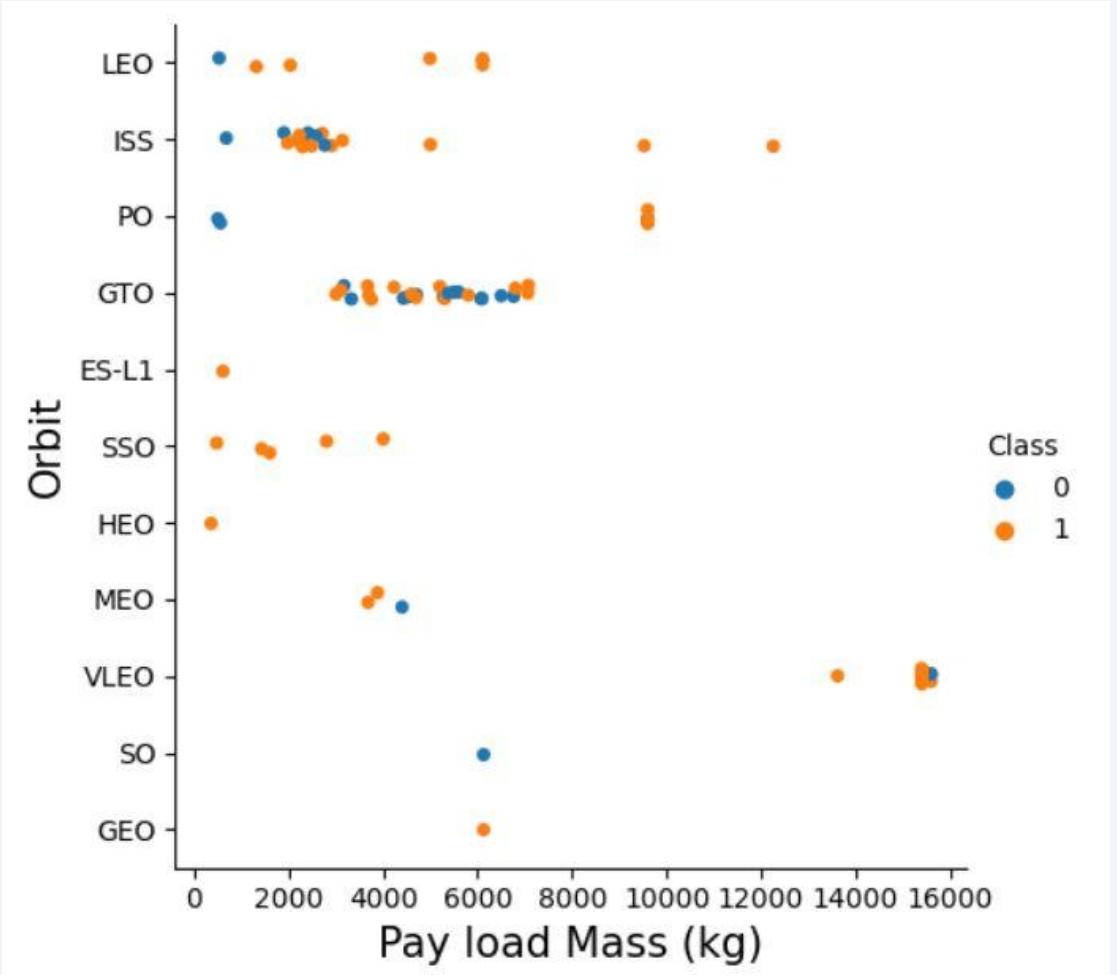
Flight Number vs. Orbit Type

- The success rate of launches generally increases with the number of flights for each orbit.
- This trend is particularly evident for the LEO orbit.



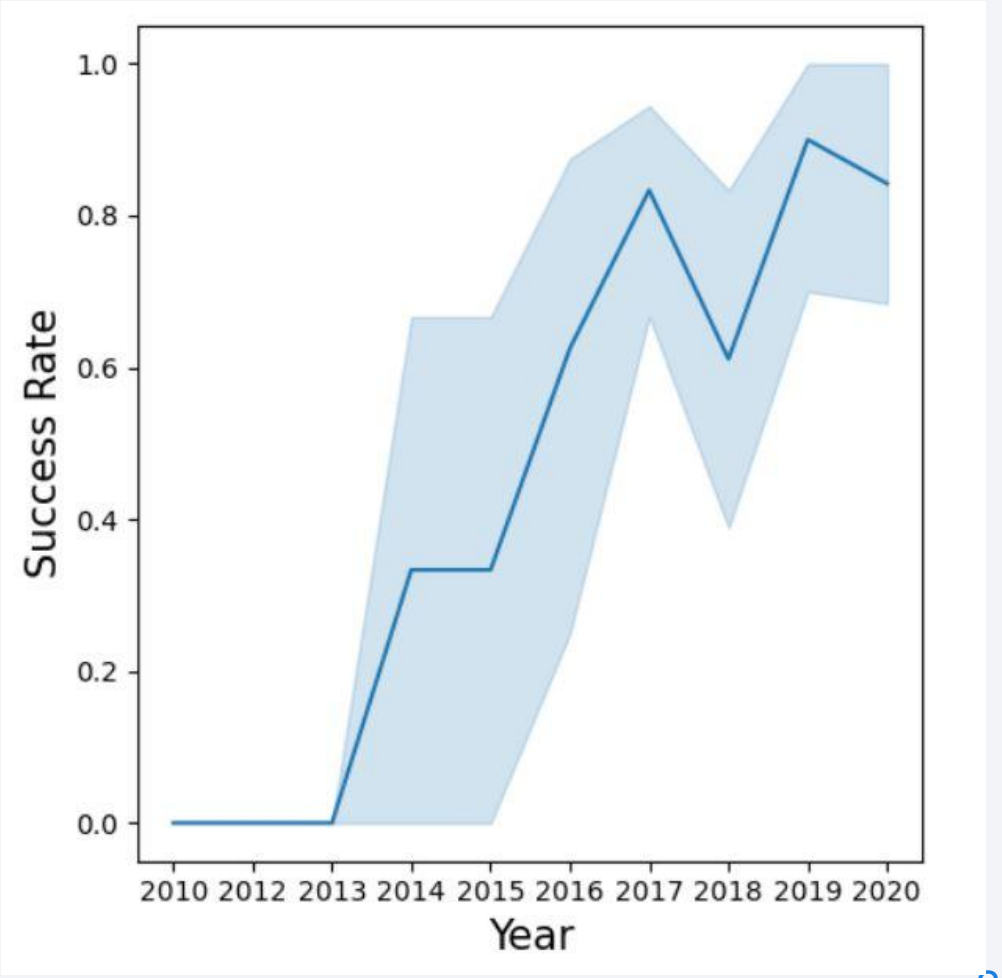
Payload vs. Orbit Type

- Heavy payloads are better with LEO, ISS and PO orbits
- The GTO orbit has mixed success with heavier payloads



Launch Success Yearly Trend

- The success rate improved from 2013-2017 and 2018-2019
- The success rate decreased from 2017-2018 and from 2019-2020
- Overall, the success rate has improved since 2013.



Launch Site Information

Launch Site Names

- Keyword DISTINCT is used to get unique launch site names from SPACEXTBL table.

```
%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

Launch Site Names Begin with 'CCA'

- Below query has been used to display 5 records where launch site 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_Outc
06/04/2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX	Success	Failure (paracl
12/08/2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0.0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (paracl
22/05/2012	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525.0	LEO (ISS)	NASA (COTS)	Success	No att
10/08/2012	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	NASA (CRS)	Success	No att
03/01/2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	NASA (CRS)	Success	No att

Total Payload Mass

- By using the SUM function, we can get total in the column PAYLOAD_MASS_KG_
- And WHERE clause filters the dataset to do the calculations only on customer NASA (CRS)

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

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

SUM(PAYLOAD_MASS__KG_)
45596.0

Average Payload Mass by Booster version F9 v1.1

- By using AVG function, we can get the average number of the column PAYLOAD_MASS_KG
- And WHERE clause filters the dataset to be BOSSTER_VERSION of 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

First Successful Ground Landing Date

- By using MIN function, we can get the minimum date in the column DATE
- And WHERE clause filters the LANDING_OUTCOME to be ‘Success (gound pad)’

```
%sql SELECT MIN(DATE) AS FIRST_SUCCESSFUL_LANDING FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (ground pad)';
```

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

FIRST_SUCCESSFUL_LANDING
01/08/2018

Successful Drone Ship Landing with Payload between 4000 and 6000

- Here, only BOOSTER_VERSION is selected from SPACEXTBL
- The WHERE clause filters LANDING_OUTCOME to be ‘Success (drone ship)’
- The AND clause indicates additional filter condition where PAYLOAD_MADD__KG_ is between 4000 AND 6000

```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL \
WHERE LANDING_OUTCOME = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000;
```

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

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

Total Number of Successful and Failure Mission Outcomes

- Here, MISSION_OUTCOME is selected and count every row as Total_number from SPACEXTBL group by MISSION_OUTCOME

```
%sql SELECT MISSION_OUTCOME, COUNT(*) AS Total_number \
FROM SPACEXTBL \
GROUP BY MISSION_OUTCOME;
```

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

Mission_Outcome	Total_number
None	898
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

- This query selects the "BOOSTER_VERSION" column from the table "SPACEXTBL" but only for the rows where the "PAYLOAD_MASS_KG" value matches the maximum payload mass value found in the subquery.
- In other words, it retrieves the booster version for the row(s) with the maximum payload mass.

```
%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

2015 Launch Records (Listing failed drone ship outcomes)

- SUBSTR(DATE, 4, 2) function extracts a substring from a “DATE” column starting at the fourth character and with a length of two characters and aliased as “MONTH” which represent the month portion of the date.
- The WHERE clause filters rows where LANDING_OUTCOME = ‘Failure(drone ship)’ and rows where the year portion of the date is ‘2015’.

```
%sql SELECT SUBSTR(DATE,4,2) AS MONTH, DATE,BOOSTER_VERSION, LAUNCH_SITE, Landing_Outcome \
FROM SPACEXTBL \
WHERE [Landing_Outcome] = 'Failure (drone ship)' and SUBSTR(Date,7,4)='2015';
```

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

MONTH	Date	Booster_Version	Launch_Site	Landing_Outcome
10	01/10/2015	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
04	14/04/2015	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

- Below query retrieves values of Landing_outcome column and performs a count of occurrences using count(*) aliased as count_outcomes.
- The WHERE clause filters the rows in the DATE column to be between 2010-06-04 and 2017-03-20 and GROUP BY [Landing_outcome] groups the results set by the “Landing_outcome” column.
- ORDER BY count_outcomes DESC sorts the result set in descending order based on the count of outcomes.

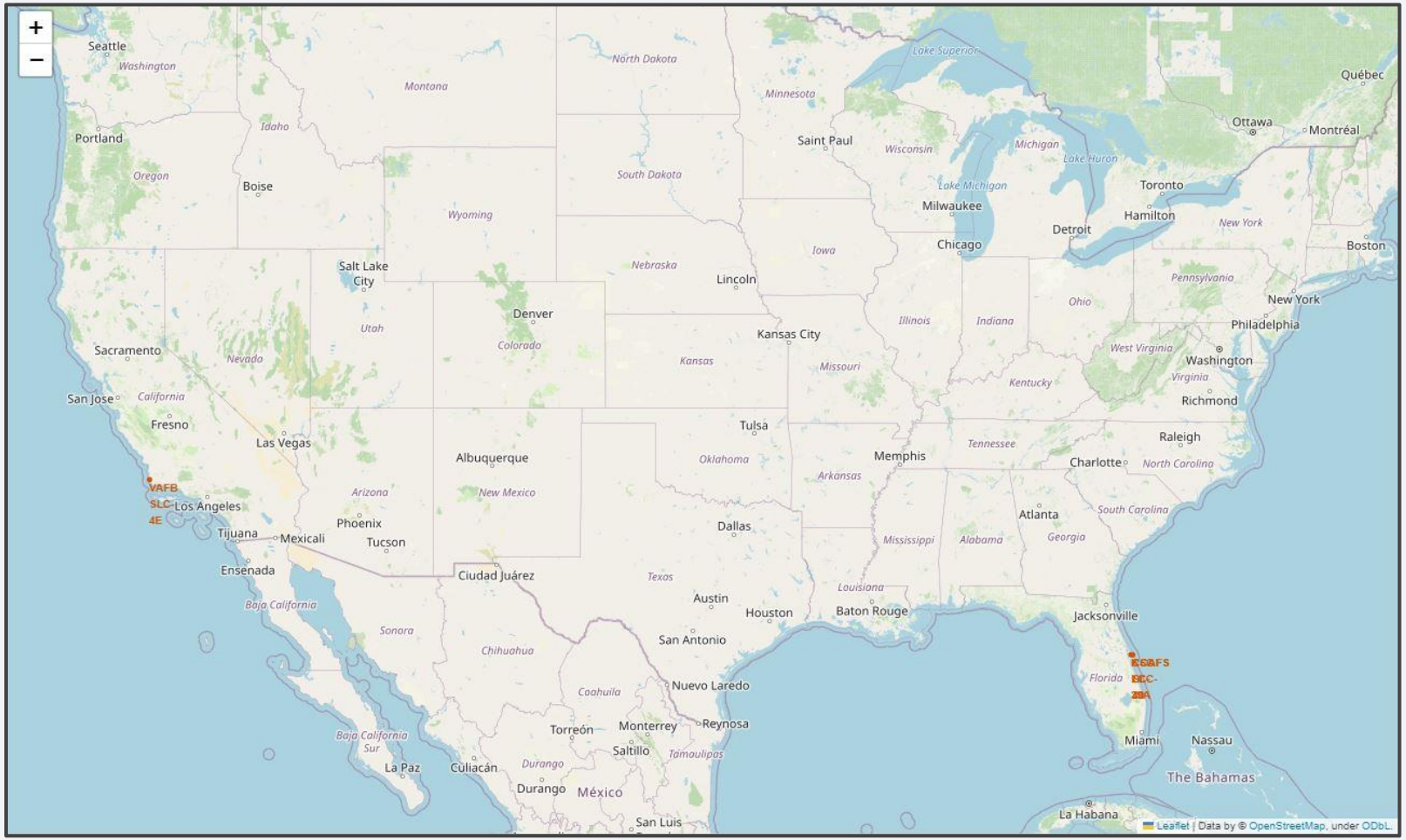
```
%sql SELECT [Landing_Outcome], count(*) as count_outcomes \
FROM SPACEXTBL \
WHERE DATE BETWEEN '04-06-2010' AND '20-03-2017' GROUP BY [Landing_Outcome] ORDER BY count_outcomes DESC;
```

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

Landing_Outcome	count_outcomes
Success	20
No attempt	10
Success (drone ship)	8
Success (ground pad)	7
Failure (drone ship)	3
Failure	3
Failure (parachute)	2
Controlled (ocean)	2
No attempt	1

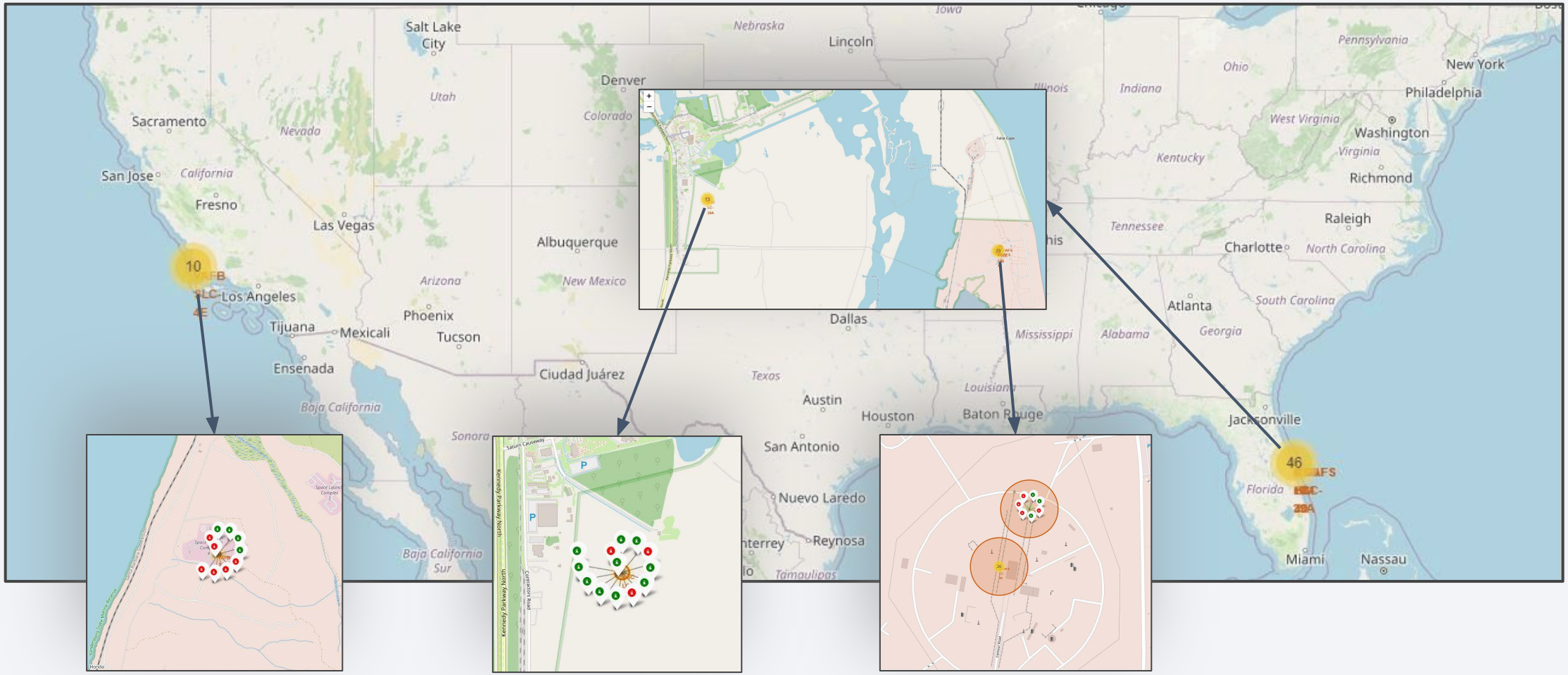

Section 3
Launch Sites
Proximities Analysis

All Launch Sites on Map



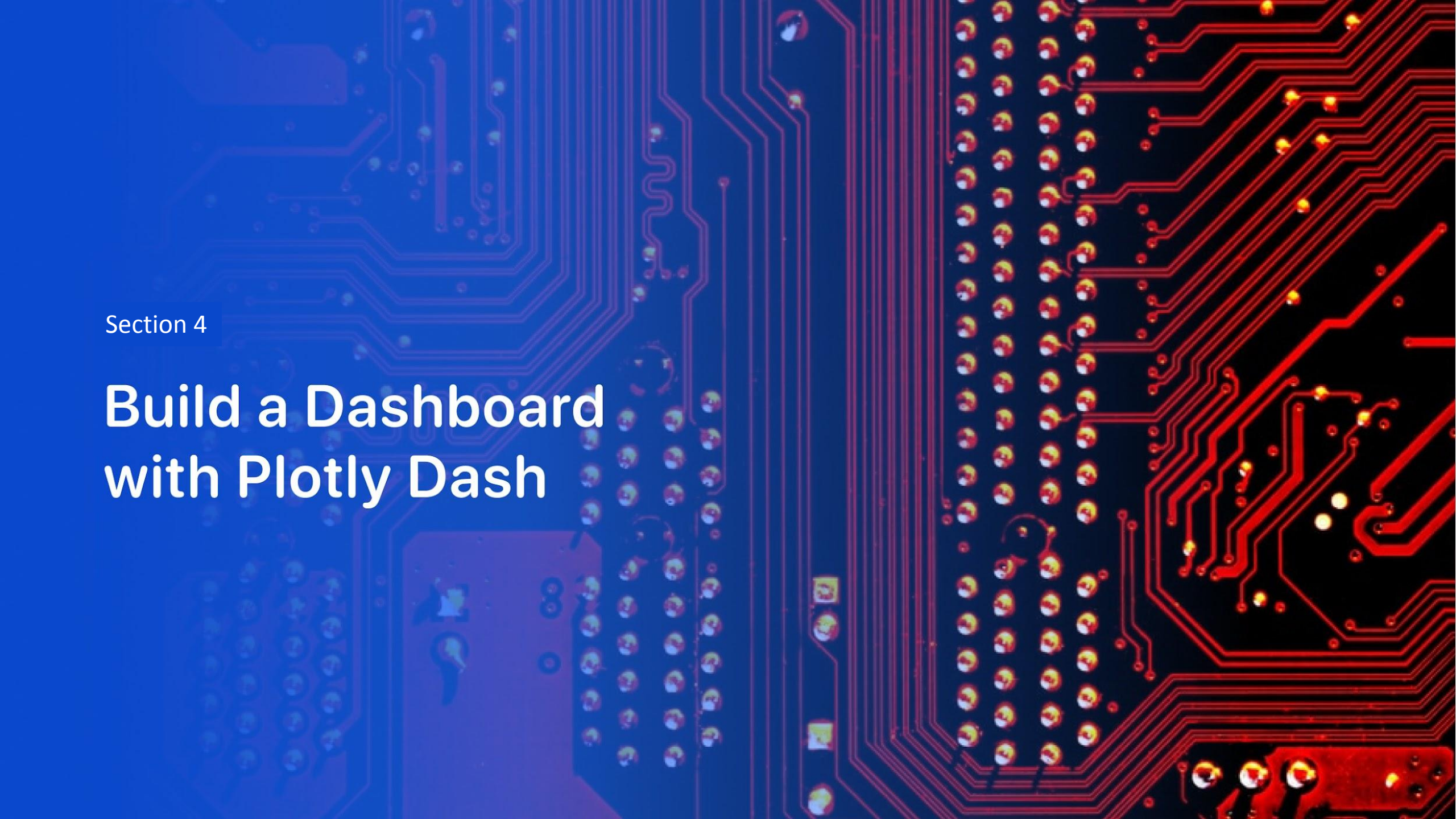
- SpaceX launch sites are in United States of America, near Florida and California coast.
- All launch sites are close to equator.

The Success/Failed Launches for Each Site on Map

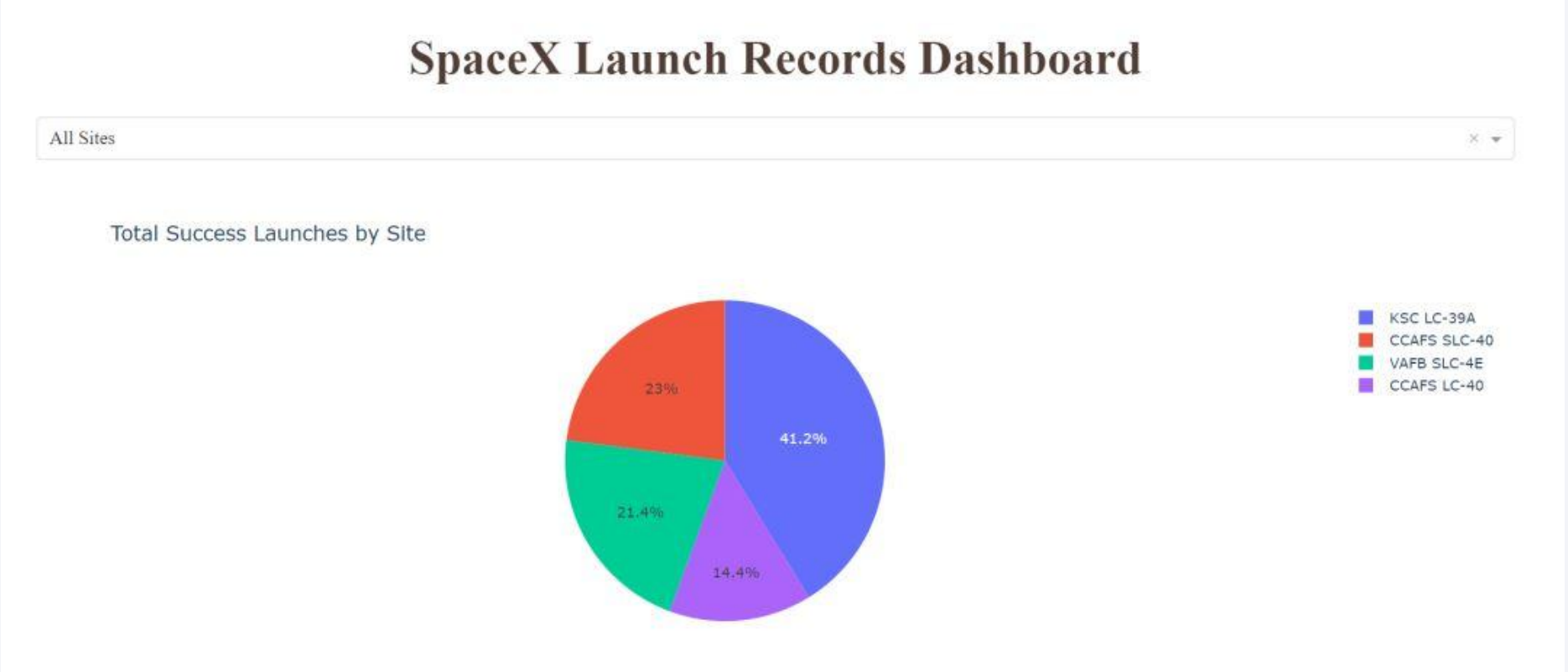


Distance to Different Areas

Launch sites are close to railways and highways which gives logistic access and relatively far from cities.


Section 4
Build a Dashboard with Plotly Dash

SpaceX Launch Records by Site



KSC LC-39A has the most successful launches amongst launch sites(41.2%).

Success/Fail Records for Site KSC LC-39A

KSC LC-39A has the highest success rate amongst launch sites (76.9%).

Payload Mass and Success Rate

Payloads between 2,000 kg and 5,000 kg have the highest success rate

- 1 indicating successful outcome and 0 indicating an unsuccessful outcome

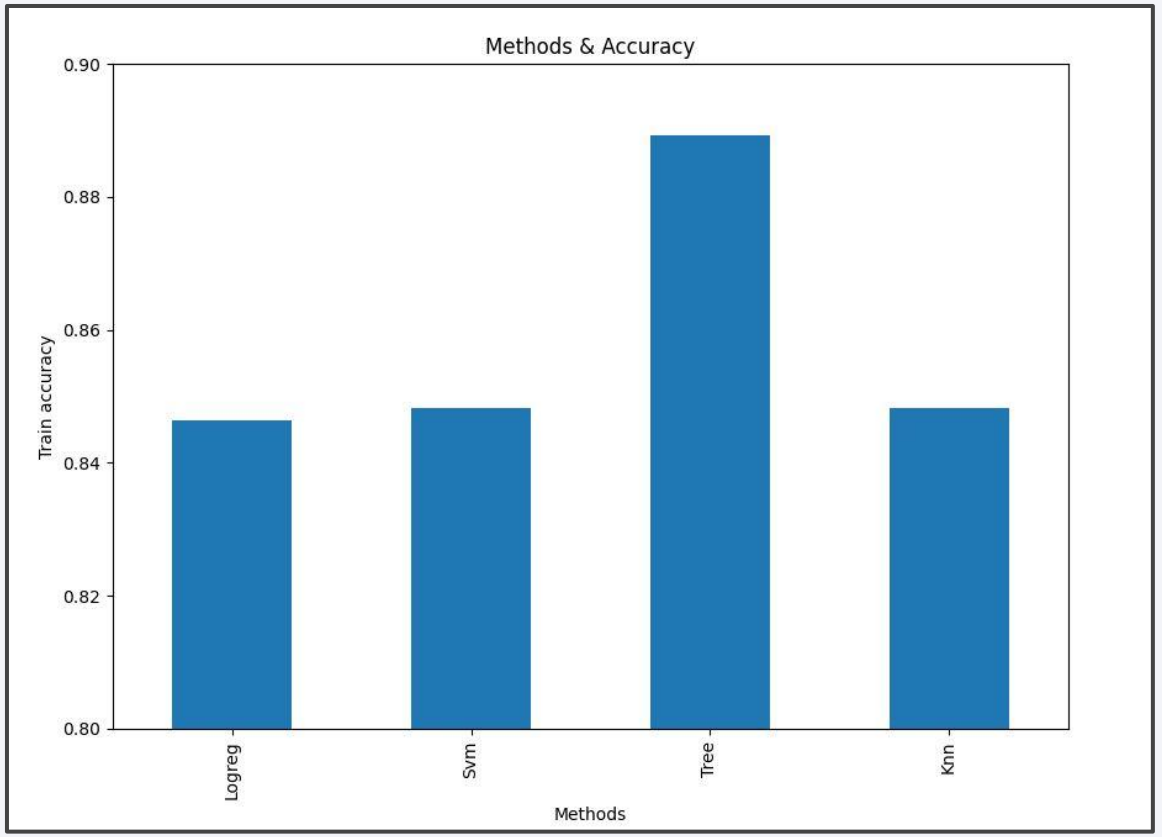
Section 5

Predictive Analysis (Classification)

Classification Accuracy

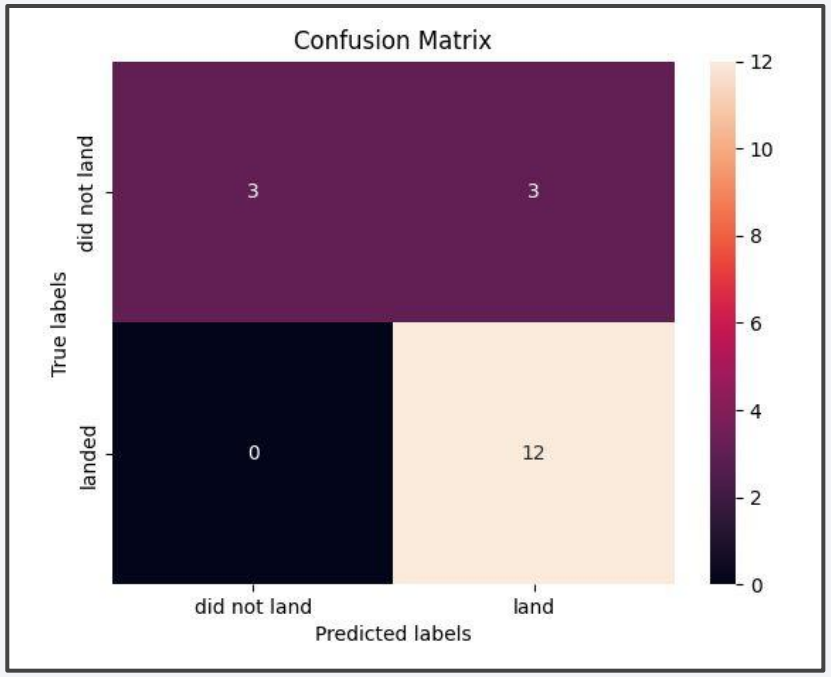
- All four classification models were tested and their accuracies are plotted here.
- Decision Tree Classifier has highest accuracy with more than 88%.

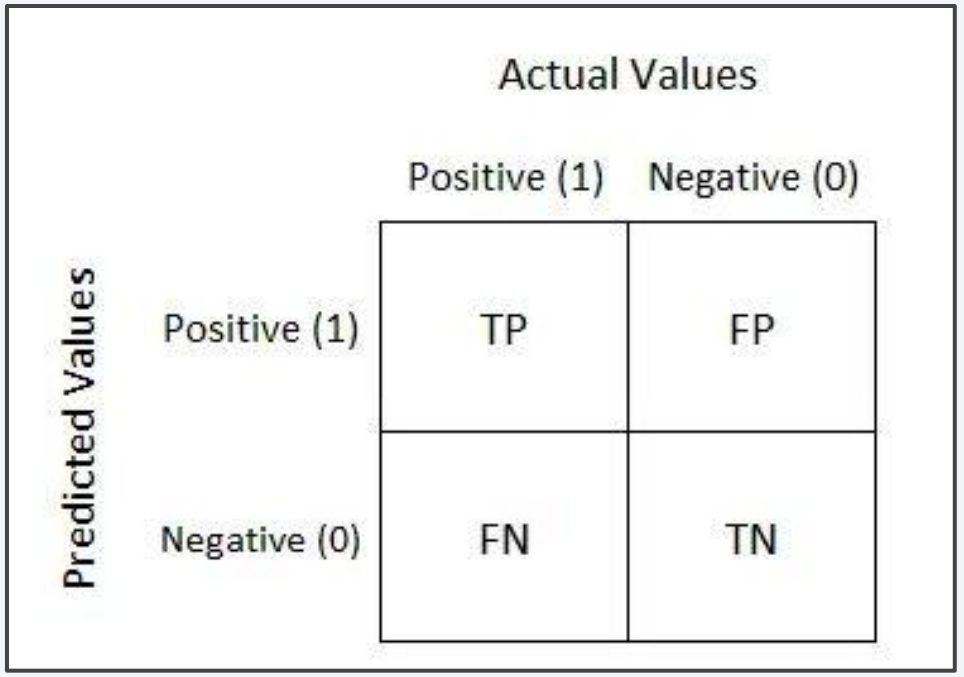
	Train_Accuracy	Test_Accuracy
Tree	0.889286	0.833333
Knn	0.848214	0.833333
Svm	0.848214	0.833333
Logreg	0.846429	0.833333



Confusion Matrix

- Confusion matrix summarizes the performance of a classification algorithm.
- By looking at the confusion matrix, the major problem found to be having false positive (Type 1 Error).
- Accuracy = (TP + TN) / (TP + TN + FP + FN) = .833 = 83.3%





Conclusions

Payload Mass

- There is a positive correlation between the payload mass (kg) and the success rate of launches.
- Payload mass greater than 7,000 kg has more successful outcome.

Launch Site

- KSC LC-39A (Launch site) has the highest successful launches.

Orbit

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

Location

- Equator: Most of the launch sites are near the equator for an additional natural boost due to the rotational speed of earth - which helps save the cost of putting in extra fuel and boosters
- Coast: All the launch sites are close to the coast (For safety measures) relatively far from cities.
- Logistic: In close proximity to railways and highways.

❖ The Tree Classifier Algorithm is the best machine learning method for this dataset.

Thank you!