# Winning Space Race with Data Science

LAYEEQ AHMED
26-12-2021

# Outline

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

# Executive Summary

- Summary of methodologies
  - Data Collection with WebScraping
  - Data Wrangling
  - Exploratory Data Analysis
  - Interactive Visual Analytics
  - Predictive Analysis
- Summary of all results
  - Findings
  - Best Method based on Predictive Analysis

# Introduction

- Project background:

  The emphasis of this work is to predict the success of Falcon 9. The reusability of the SPACEX first stages enables lower rocket launch costs in comparison to other providers. Predicting the success of the Falcon 9 launch will attract the customers to use SPACEX Falcon 9 launches with 62% savings in the launch costs.

- Anticipated Answers:
  - Factors affecting the Success or Failure of the Launch.
  - Relationship between the various attributes of the SPACEX launches.
  - Predicting the best factors for the Success of the Launch.

Section 1

# Methodology

# Methodology

Executive Summary

- Data collection methodology:
  - Data for this work was collected using SPACEX API and Webscraping Wikipedia.
- Performing data wrangling
  - Dropping unnecessary/missing data and One-hot coding to facilitate predictive analysis.
- Performing exploratory data analysis (EDA) using visualization and SQL
  - EDA with SQL and Scatter, Bar Charts
- Performing interactive visual analytics using Folium and Plotly Dash
- Performing predictive analysis using classification models.

# Data Collection

- Describe how data sets were collected.

  Step 1 : Data Collection using SPACEX API and Wikipedia

  Step 2 : Converting the response to .json using json_normalize / BeautifulSoup

  Step 3 : Converting data into dataframe using defined functions

  Step 4 : Filtering the data for Falcon 9 and replacing missing data using mean column function.

# Data Collection – SpaceX API

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

```
data1 = resp.json()
data = pd.json_normalize(data1)
```

```
# Call getLaunchSite
getLaunchSite(data)
```

```
# Call getPayloadData
getPayloadData(data)
```

```
# Call getCoreData
getCoreData(data)
```

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

```
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan,data_falcon9['PayloadMass'].mean())
```

Data Collection using SPACEX API and Wikipedia

Converting the response to .json using json_normalize

Converting data into dataframe using defined functions

Filtering the data for Falcon 9 and replacing missing data using mean column function.



NB_Link

# Data Collection - Scraping

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

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

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

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

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

Data Collection on Falcon Launches Wikipedia

↓

Creating a BeautifulSoup Object

↓

Finding Tables and Extracting Column names

↓

Converting into DataFrame



NB_Link

# Data Wrangling

Data wrangling is the process of cleaning, structuring and enriching raw data into a desired format for better decision making in less time.

Loading Data

```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv")
df.head(10)
```

Converting into DataFrame

Cleaning the Dataframe

Simplifying the DataFrame

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

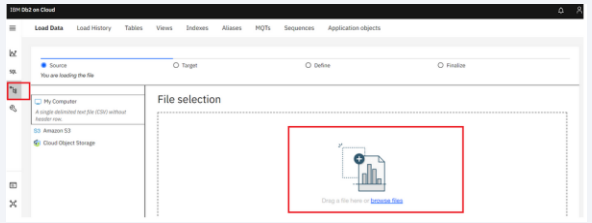
Normalising the Data



NB_Link

# EDA with SQL

EDA performed for this work with SQL:




- Loading the Data into the DataBase
- Connecting to Database


- Performing queries to study the database

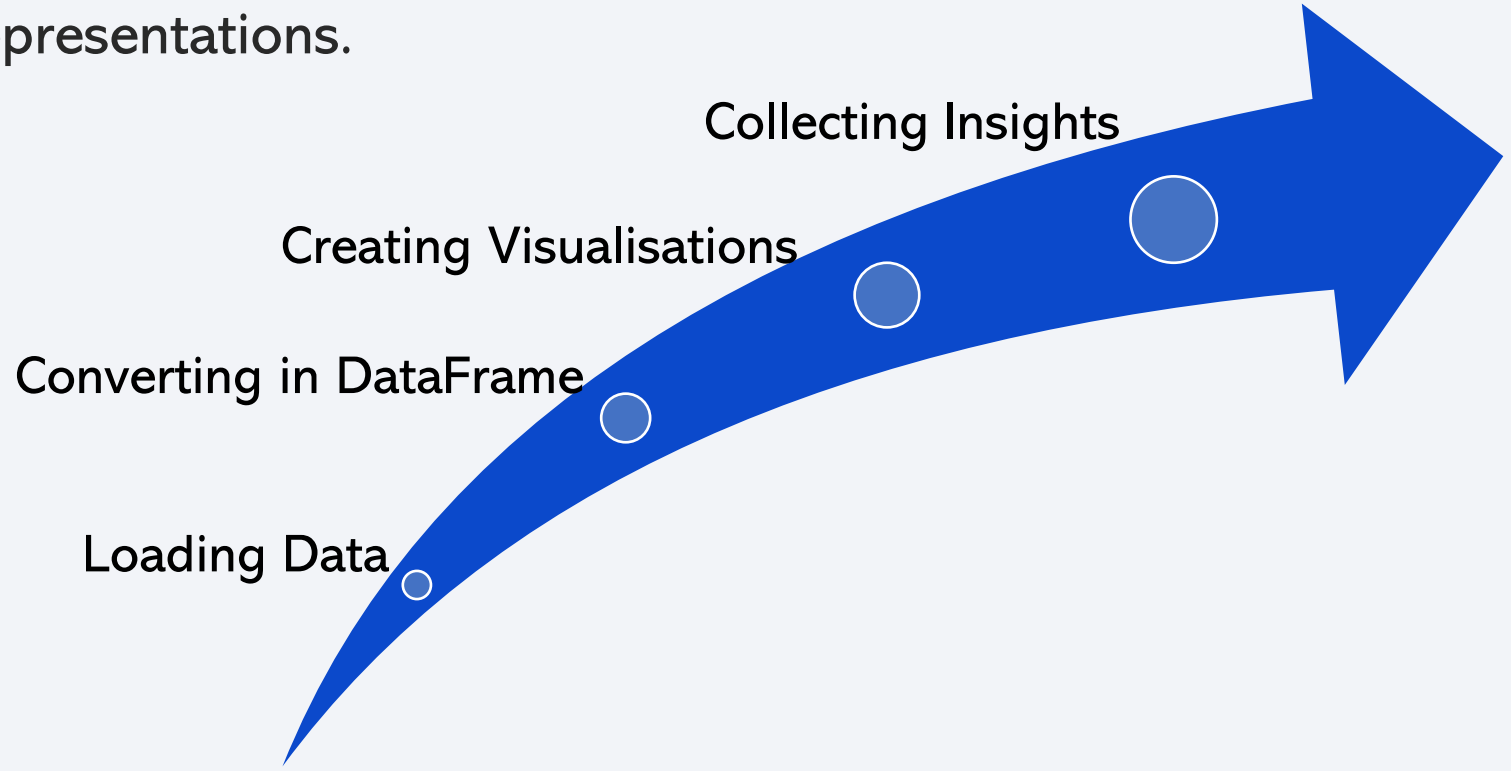
•Displaying the names of the unique launch sites in the space mission
•Displaying 5 records where launch sites begin with the string 'KSC'
•Displaying the total payload mass carried by boosters launched by NASA (CRS)
•Displaying average payload mass carried by booster version F9 v1.1
•Listing the date 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_outcomesin ground pad ,booster versions, launch_sitefor the months in year 2017
•Ranking the count of successful landing_outcomesbetween the date 2010-06-04 and 2017-03-20 in descending order.



NB_Link

# EDA with Data Visualization

- EDA stands for Exploratory Data Analysis, Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.



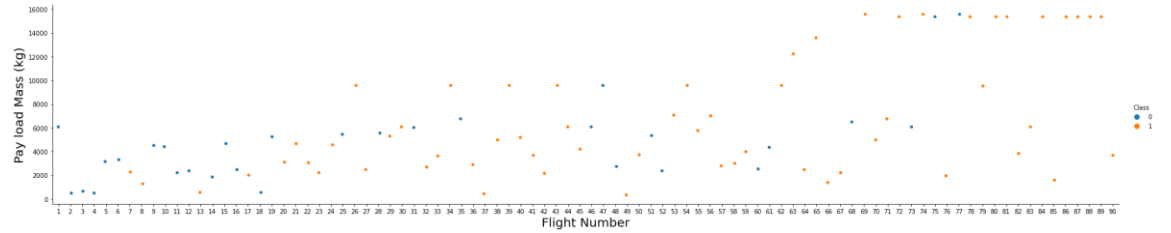


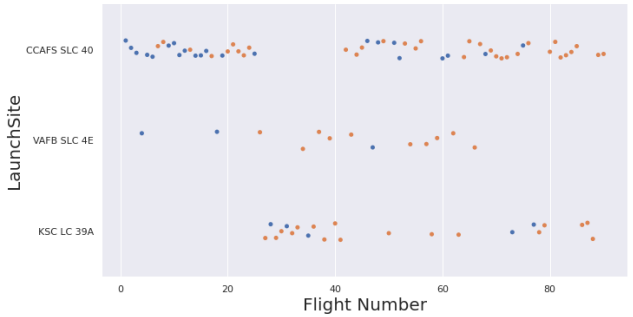

NB Link

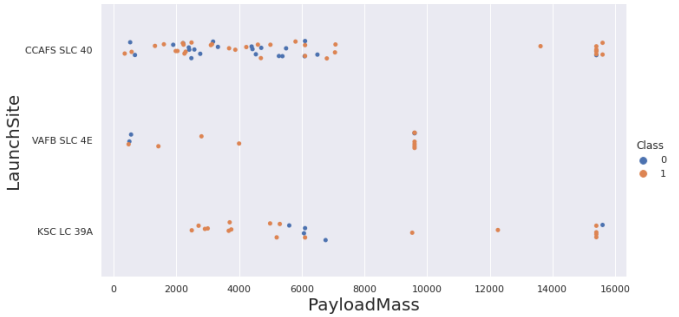
# Data Visualisation

## Data Visualisation:

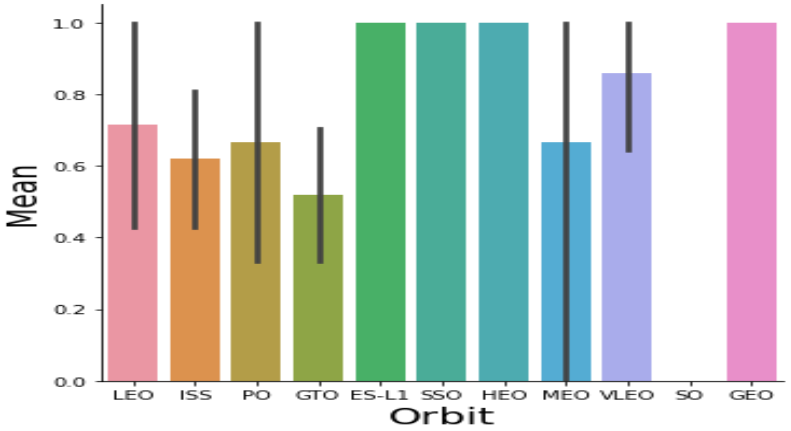
- Scatter Point Graphs
  - FlightNumber vs. PayloadMass
  - FlightNumber vs LaunchSite
  - Payload vs Launch Site
  - Flight Number vs Orbit
  - Payload vs Orbit



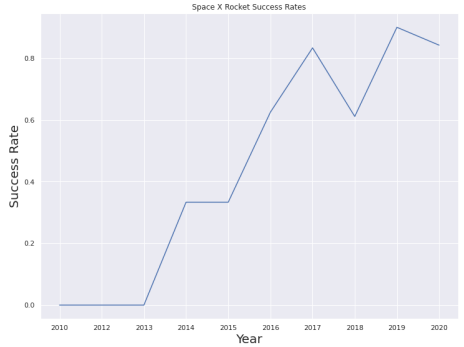




- Bar Graph
  - Orbit vs Mean Class




- Line Graph
  - Year vs Success Rate



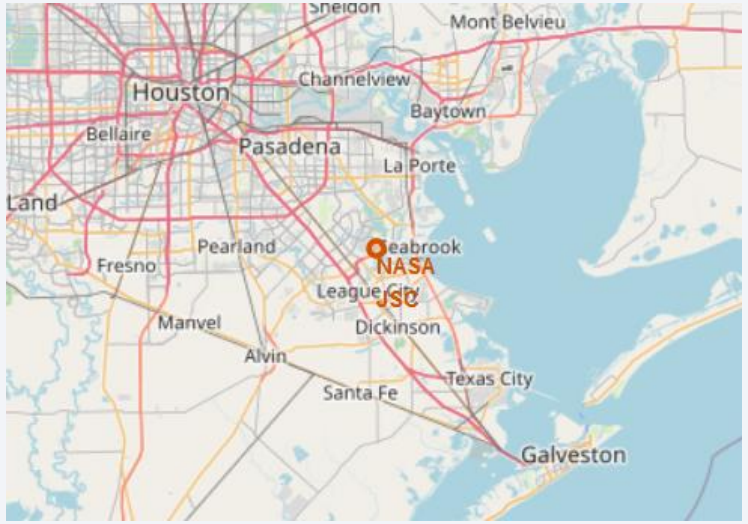


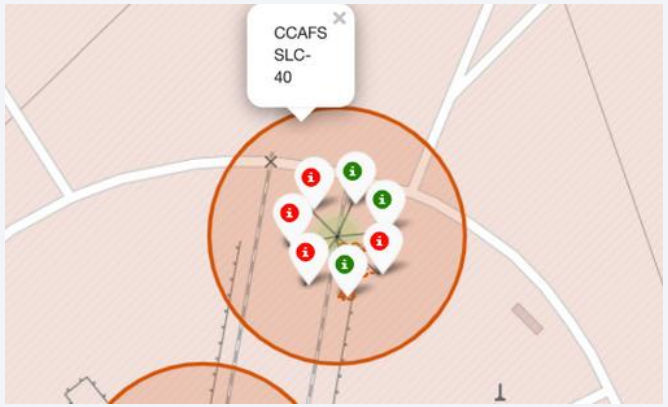

NB Link

# Build an Interactive Map with Folium

- Visualization of Launch Data into the Interactive Map with Folium
  - Latitude and Longitude of Launche Sites
  - Circle Marker around the Launch Sites.
  - Launch Outcome Classes Success and Failure with Green & Red Markers

- Map Objects used:
  - Map Marker
  - Icon Marker
  - Circle Marker
  - PolyLine
  - Marker Cluster object
  - Antpath

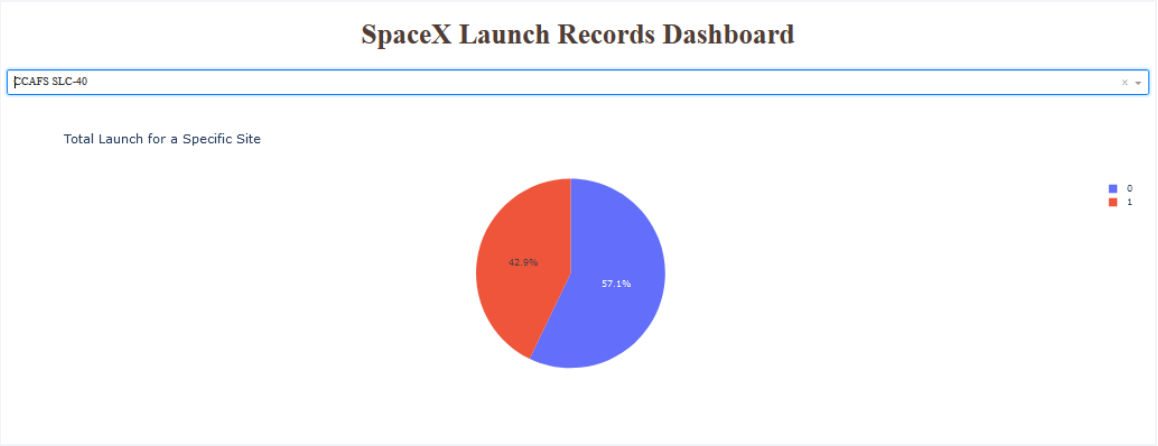



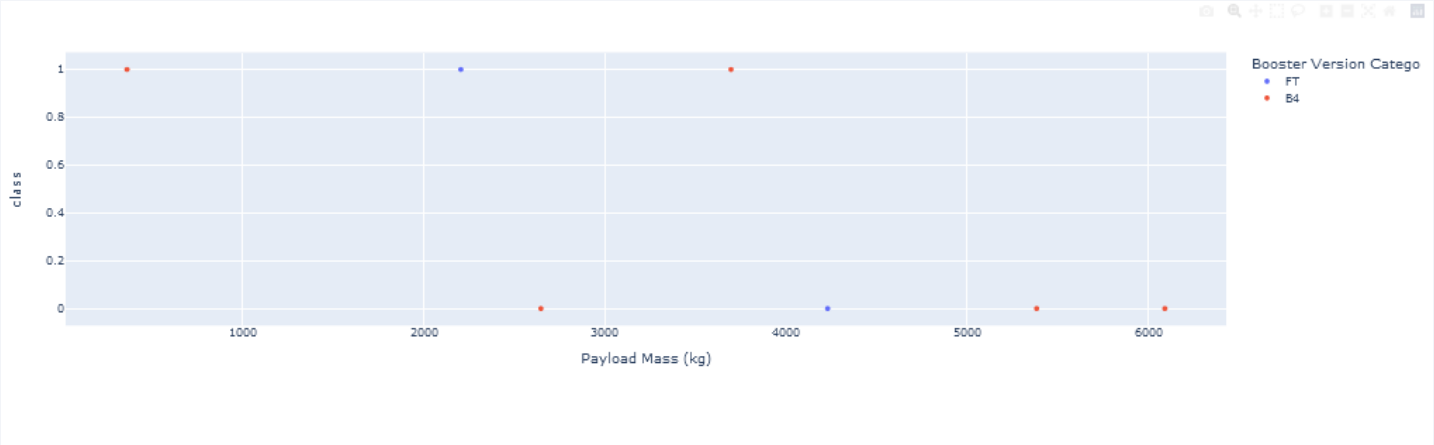



NB_Link

# Build a Dashboard with Plotly Dash

- Pie Chart
  - This chart shows the success at each launch site.
- Scatter Charts
  - The charts shows the Payload vs Success for different booster versions




- Map Objects used:
  - Dash and Dash Components
  - Pandas
  - Plotly
  - Dropdown
  - Rangeslider
  - Pie Chart
  - Scatter Chart



NB_Link

# Predictive Analysis (Classification)

Building Model

- Loading the Data
- Transforming the Data
- Spliting Data for Test & Train
- Creating GridSerachCV object
- Fitting through Machine Learning Algorithms

Evaluating Model

- Estimating Accuracy of the Model
- Identifying Best Hyperparameters
- Visualization through Confusion Matrix

Identifying Best Model

- Comparison of Accuracy & Hyperparameters of different Classfication Method

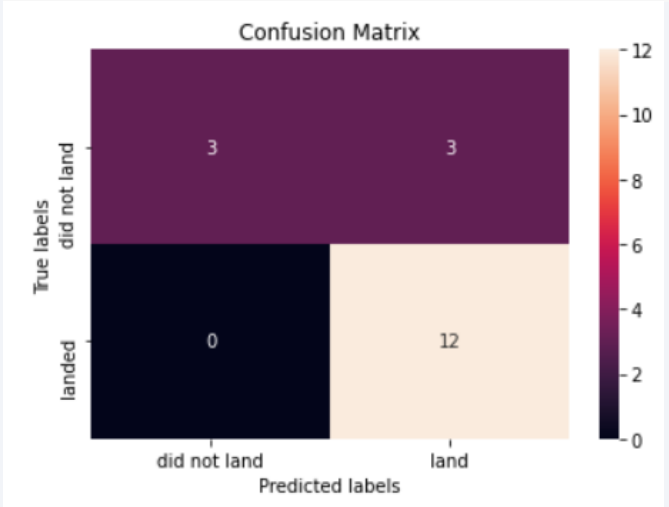
```
X = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_3.csv')
```

```
X = transform.fit(X).transform(X)
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.2, random_state=2)
```

```
gs_cv = GridSearchCV(lr, parameters, scoring='accuracy', cv=10)
logreg_cv = gs_cv.fit(X_train, Y_train)
```

```
print("TUNED HYPERPARAMETERS: ",svm_cv.best_params_)
print("ACCURACY :",svm_cv.best_score_)
```





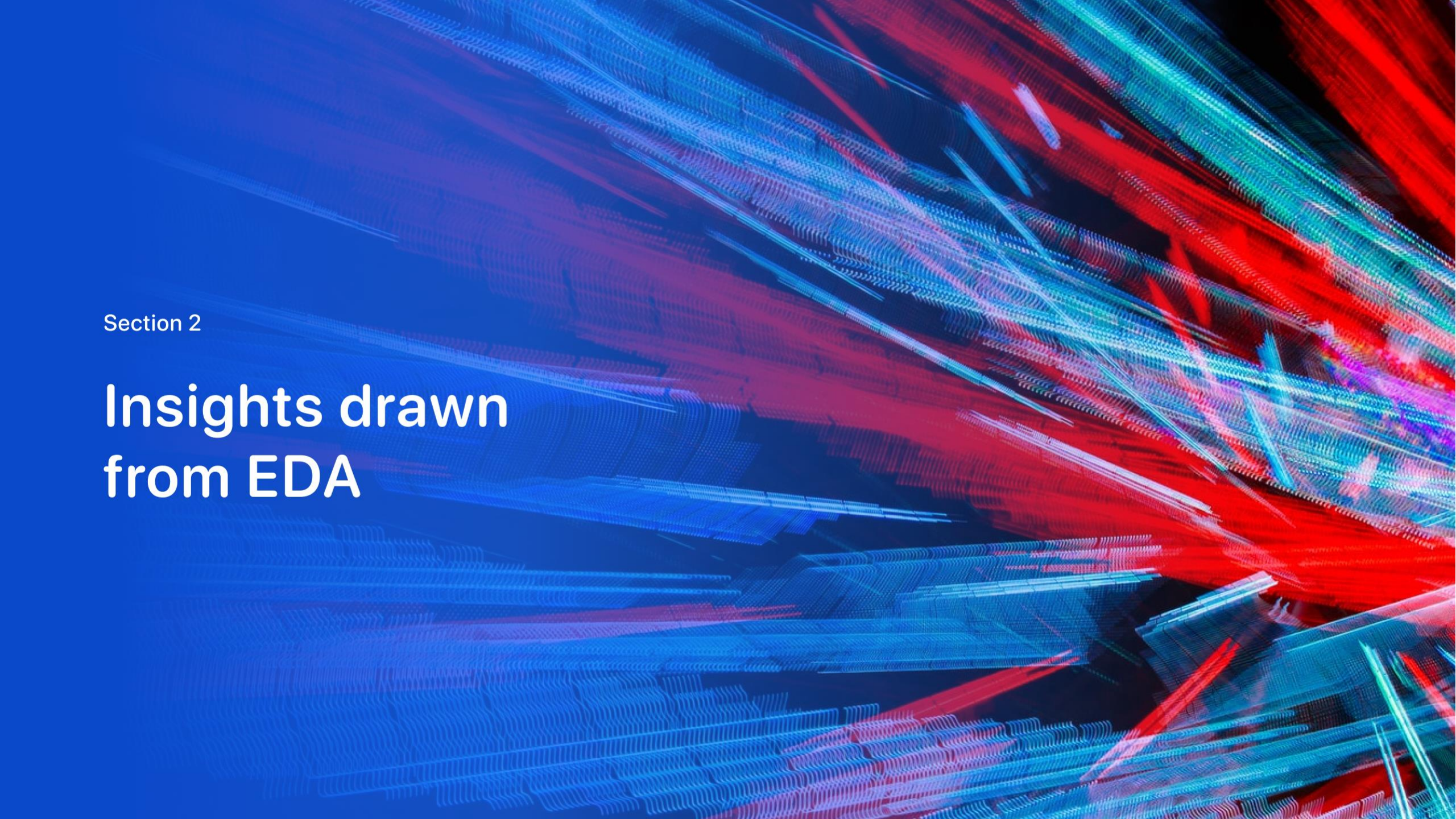

NB_Link

# Results

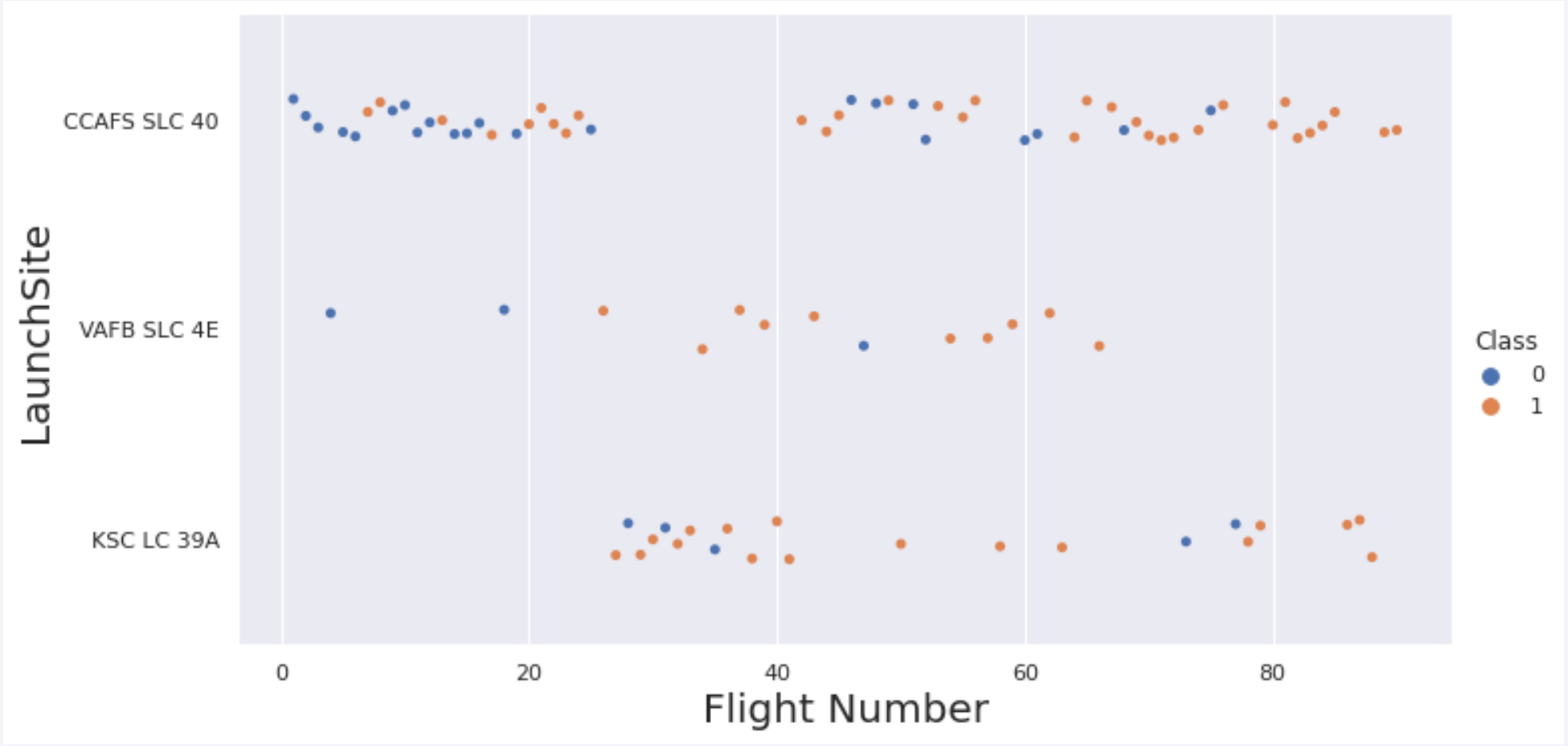
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

# Insights drawn from EDA

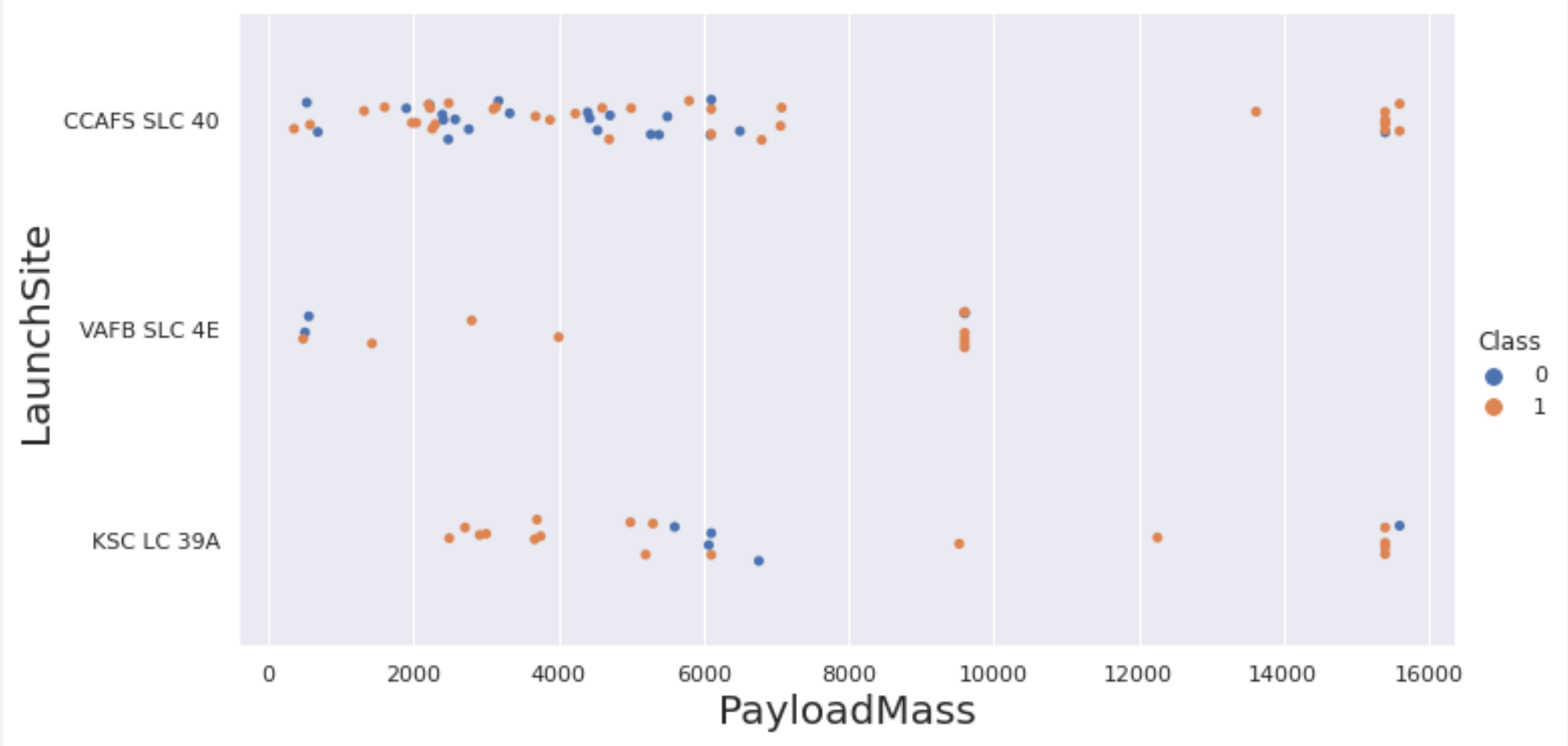


# Flight Number vs. Launch Site



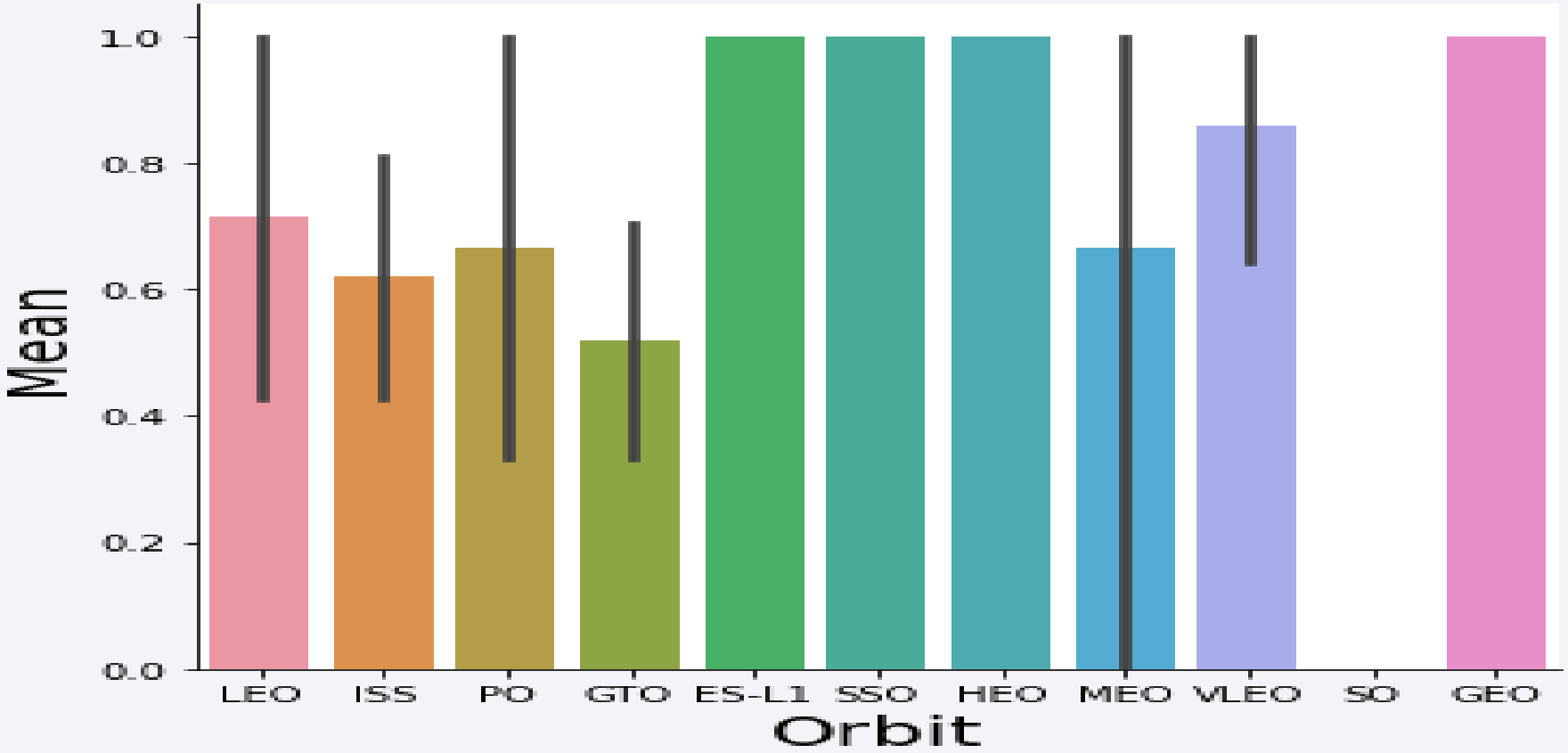

Success Rate is Consistent on Higher Flight Numbers

# Payload vs. Launch Site



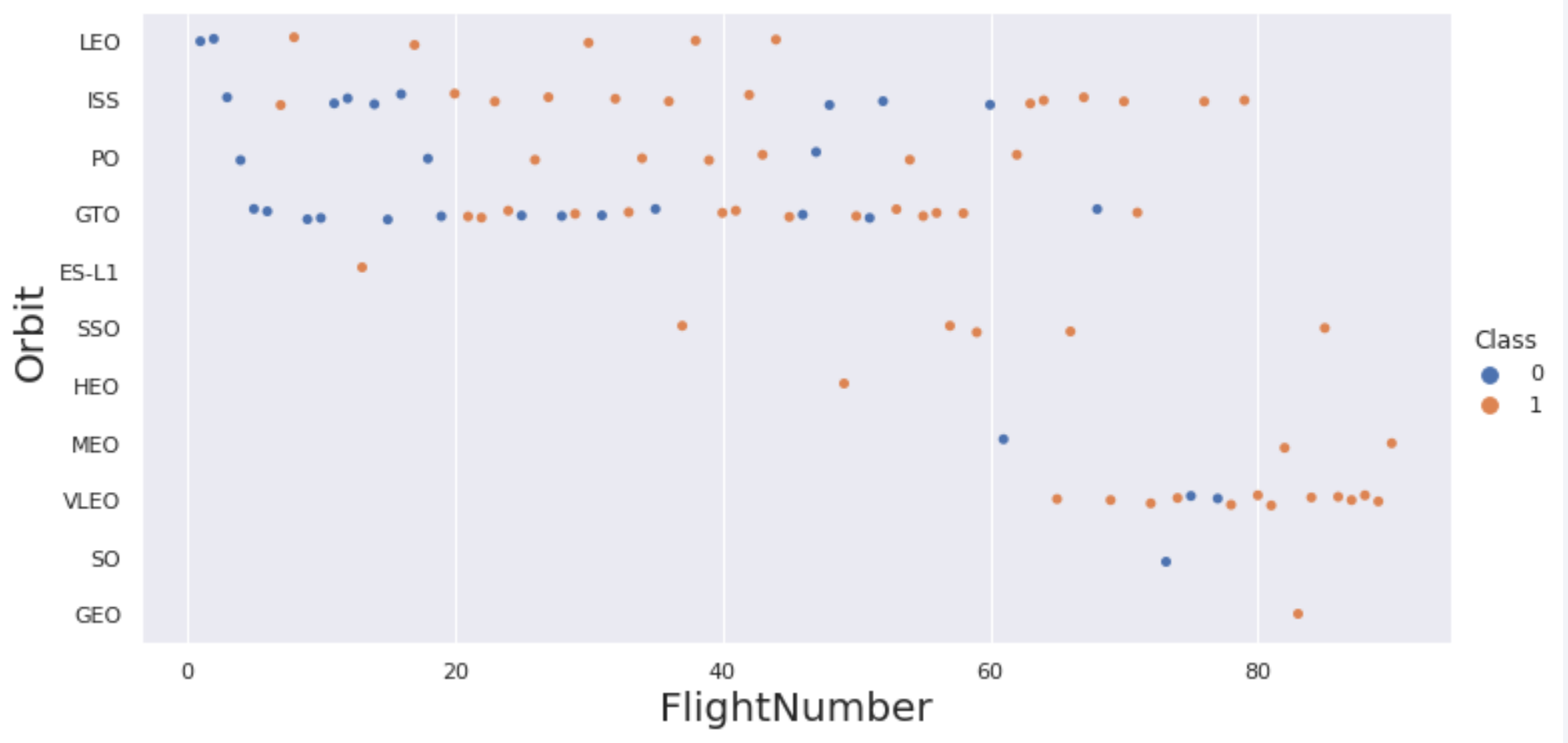

- Payload Mass of more than 15000 kg has higher success rate at CCAFS SLC 40 & KSC LC 39A
- Payload Mass of about 9000 kg has higher success rate at VAFB SLC 4E

# Success Rate vs. Orbit Type



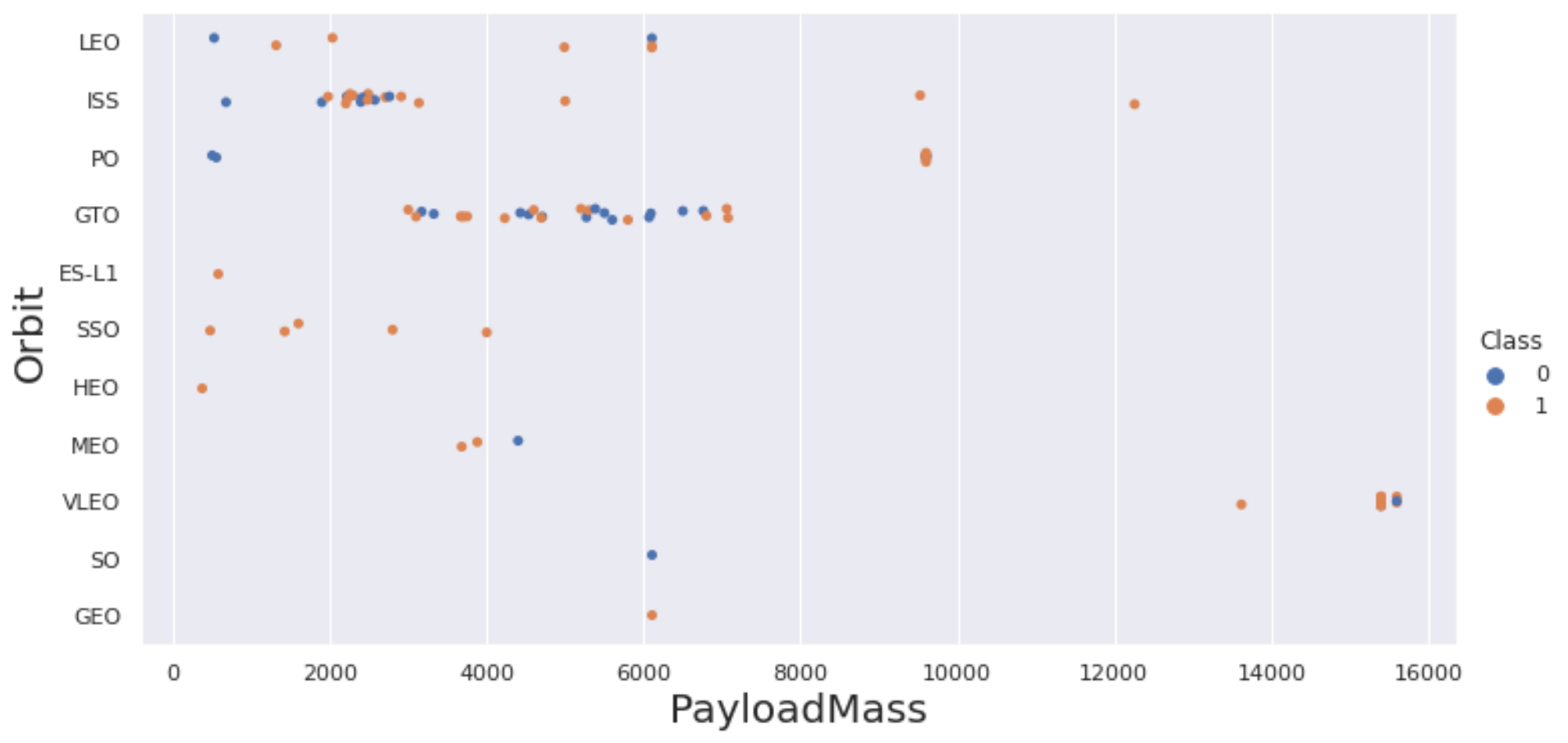

Orbits ES-L1, SSO, HEO and GEO has higher success rates

# Flight Number vs. Orbit Type



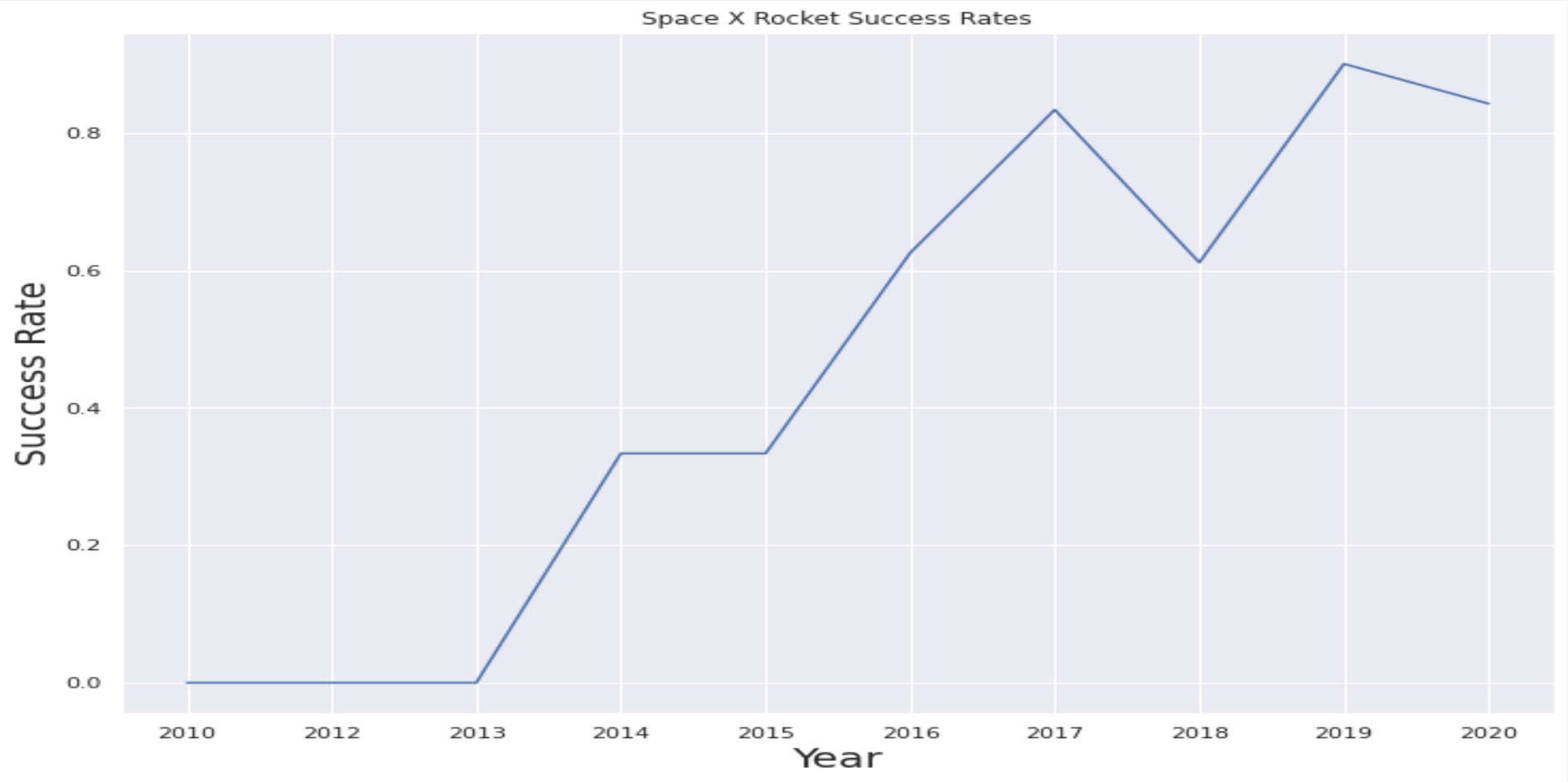

LEO orbit has better success rate at Low Number of Flights

# Payload vs. Orbit Type




With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.

# Launch Success Yearly Trend




Success rate has been consistently increasing since 2013

# All Launch Site Names

```
%sql SELECT DISTINCT(launch_site) from SPACEXTBL
```

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

The outcome of this query lists the launch site names distinctively using DISTINCT syntax.

# Launch Site Names Begin with 'CCA'

- Finding 5 records where launch sites begin with `CCA`

```
%sql select * from SPACEXTBL where launch_site like 'CCA%' limit(5)
```

| DATE | time__utc_ | booster_version | launch_site | payload | payload_mass__kg_ |
|---|---|---|---|---|---|
| 2010-06-04 | 18:45:00 | F9 v1.0 B0003 | CCAFS LC-40 | Dragon Spacecraft Qualification Unit | 0 |
| 2010-12-08 | 15:43:00 | F9 v1.0 B0004 | CCAFS LC-40 | Dragon demo flight C1, two CubeSats, barrel of Brouere cheese | 0 |
| 2012-05-22 | 07:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525 |
| 2012-10-08 | 00:35:00 | F9 v1.0 B0006 | CCAFS LC-40 | SpaceX CRS-1 | 500 |
| 2013-03-01 | 15:10:00 | F9 v1.0 B0007 | CCAFS LC-40 | SpaceX CRS-2 | 677 |

The outcome of this query lists the launch site records which begins with 'CCA' and limited to 5 using LIMIT syntax.

# Total Payload Mass

- Calculating the total payload carried by boosters from NASA

```
%sql select sum(payload_mass__kg_) as total_payload_by_NASACRS from SPACEXT
BL where customer like 'NASA (CRS)'
```

| total_payload_by_nasacrs |
| --- |
| 45596 |

The outcome of this query sums up the total payload carried from NASA CRS using the SUM sytax

# Average Payload Mass by F9 v1.1

- Calculating the average payload mass carried by booster version F9 v1.1

```
%sql select avg(payload_mass__kg_) as average_payload_mass_F9V11 from SPACE
XTBL where booster_version = 'F9 v1.1';
```

| average_payload_mass_f9v11 |
| --- |
| 2928 |

The outcome of this calculates the Average payload carried by booster version F9 v1.1 using the AVG syntax

# First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad

```
%sql select min(DATE) from SPACEXTBL where landing__outcome = 'Success (gro
und pad)'
```

| 2015-12-22 |
|---|

The outcome of this query identifies the first successful landing outcome on ground pad using the MIN function on Date.

# Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql select booster_version,payload_mass__kg_,landing__outcome from SPACEXT
BL where payload_mass__kg_ >4000 and payload_mass__kg_<6000 and landing__ou
tcome = 'Success (drone ship)';
```

| booster_version | payload_mass__kg_ | landing__outcome |
|---|---|---|
| F9 FT B1022 | 4696 | Success (drone ship) |
| F9 FT B1026 | 4600 | Success (drone ship) |
| F9 FT B1021.2 | 5300 | Success (drone ship) |
| F9 FT B1031.2 | 5200 | Success (drone ship) |

This query lists the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000.This query uses “and” operator to apply multiple conditions.

# Total Number of Successful and Failure Mission Outcomes

- Calculating the total number of successful and failure mission outcomes

```
#%sql select sum(mission_outcome) from SPACEXTBL group by mission_outcome
%sql SELECT mission_outcome,COUNT(*) as total from SPACEXTBL group by missi
on_outcome
```

| mission_outcome | total |
|---|---|
| Failure (in flight) | 1 |
| Success | 99 |
| Success (payload status unclear) | 1 |

This query is to calculate the total number of successful and failure mission outcomes. This query uses groupby syntax to group the column categories.

# Boosters Carried Maximum Payload

Tthe names of the booster which have carried the maximum payload mass

```
%sql select booster_version,payload_mass__kg_,landing__outcome from SPACEXT
BL where payload_mass__kg_ in (select max(payload_mass__kg_) from SPACEXTB
L)
```

| booster_version | payload_mass__kg_ | landing__outcome |
|---|---|---|
| F9 B5 B1048.4 | 15600 | Success |
| F9 B5 B1049.4 | 15600 | Success |
| F9 B5 B1051.3 | 15600 | Success |
| F9 B5 B1056.4 | 15600 | Failure |
| F9 B5 B1048.5 | 15600 | Failure |
| F9 B5 B1051.4 | 15600 | Success |
| F9 B5 B1049.5 | 15600 | Success |
| F9 B5 B1060.2 | 15600 | Success |
| F9 B5 B1058.3 | 15600 | Success |
| F9 B5 B1051.6 | 15600 | Success |
| F9 B5 B1060.3 | 15600 | Success |
| F9 B5 B1049.7 | 15600 | Success |

This query uses a subquery to extract the payload mass and lists along with the booster version used.

# 2015 Launch Records

- Failed landing outcomes in drone ship, their booster versions, and launch site names for in year 2015

```
%sql select DATE,booster_version,landing__outcome,launch_site from SPACEXTB
L where landing__outcome = 'Failure (drone ship)' and year(DATE)= '2015'
```

| DATE | booster_version | landing__outcome | launch_site |
|---|---|---|---|
| 2015-01-10 | F9 v1.1 B1012 | Failure (drone ship) | CCAFS LC-40 |
| 2015-04-14 | F9 v1.1 B1015 | Failure (drone ship) | CCAFS LC-40 |

This query uses and operator to apply multiple conditions and defined extract_year function to extract year from Date.

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

- Ranking landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

```
%sql SELECT mission_outcome,COUNT(*) as total from SPACEXTBL group by missi
on_outcome order by total desc
```

| mission_outcome | total |
|---|---|
| Success | 99 |
| Failure (in flight) | 1 |
| Success (payload status unclear) | 1 |

This query uses COUNT , GROUP BY, ORDER BY syntax to complete the query of ranking the mission outcomes.

Section 4

# Launch Sites Proximities Analysis

# Launch Sites Marked on the Map

# Colored Label Markers

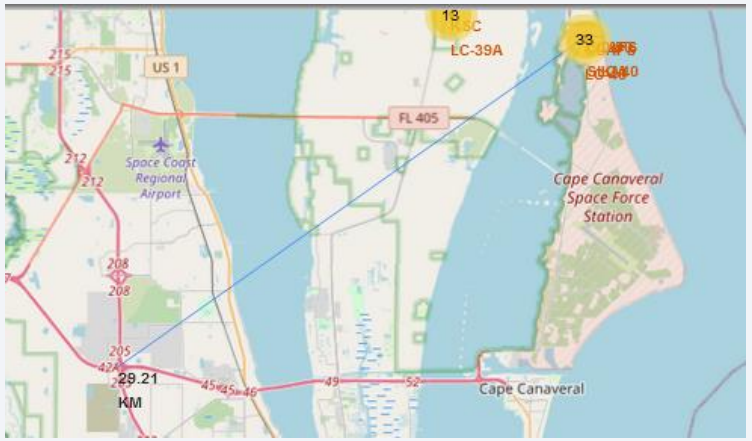
- Indication of Success and Failure using the Green and Red Markers



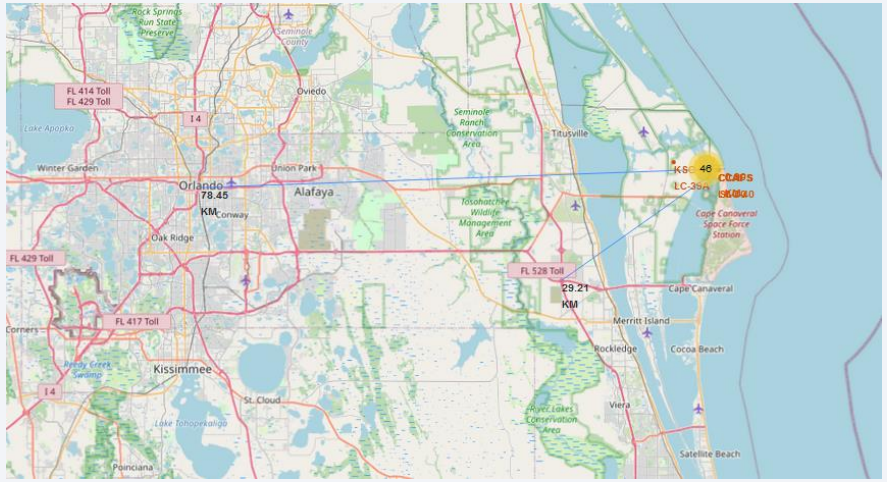
# Distances from Coastline, Highway and Railway

Coastline:


Coastline:




Railway:

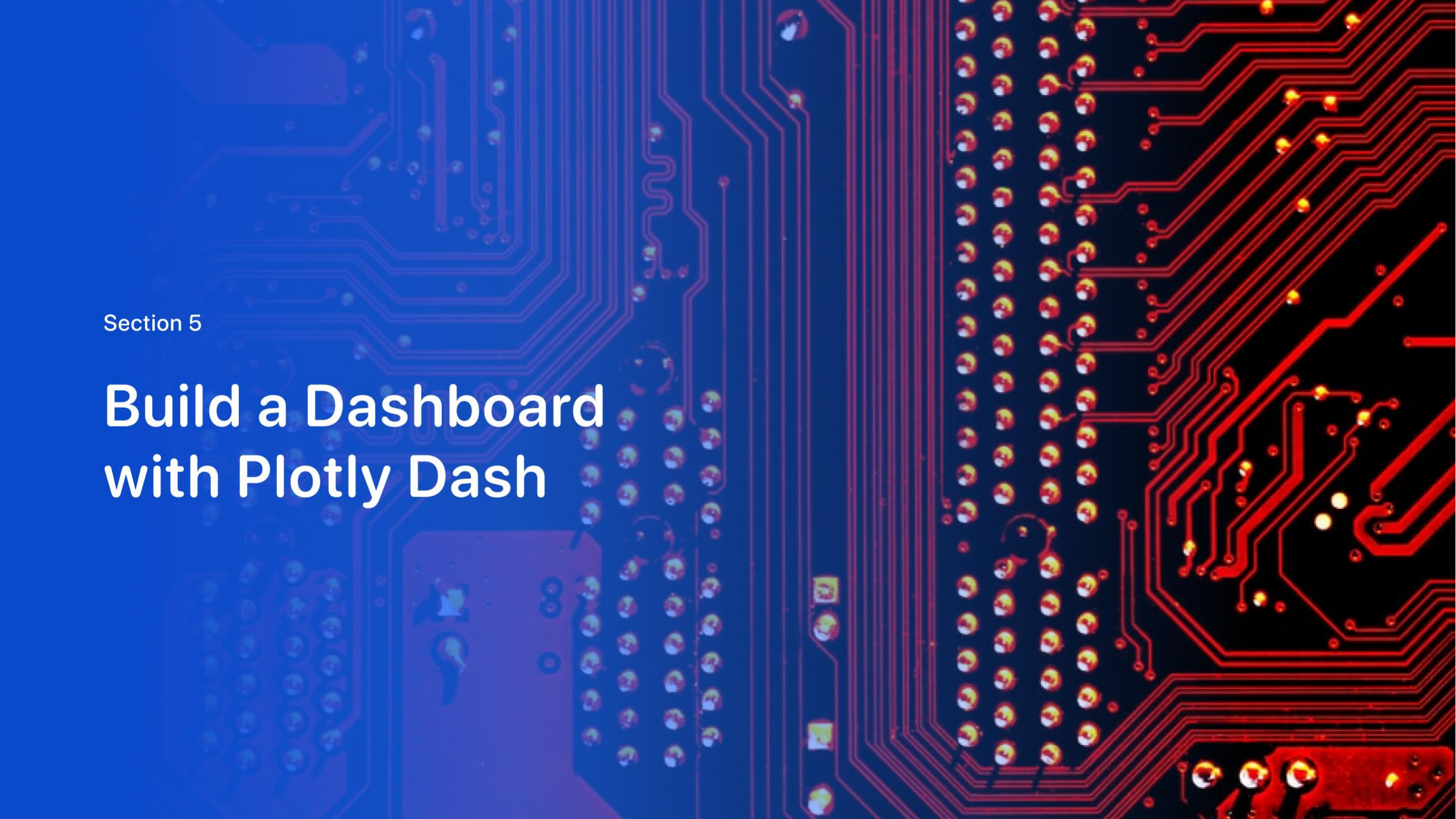



Conclusive Study:

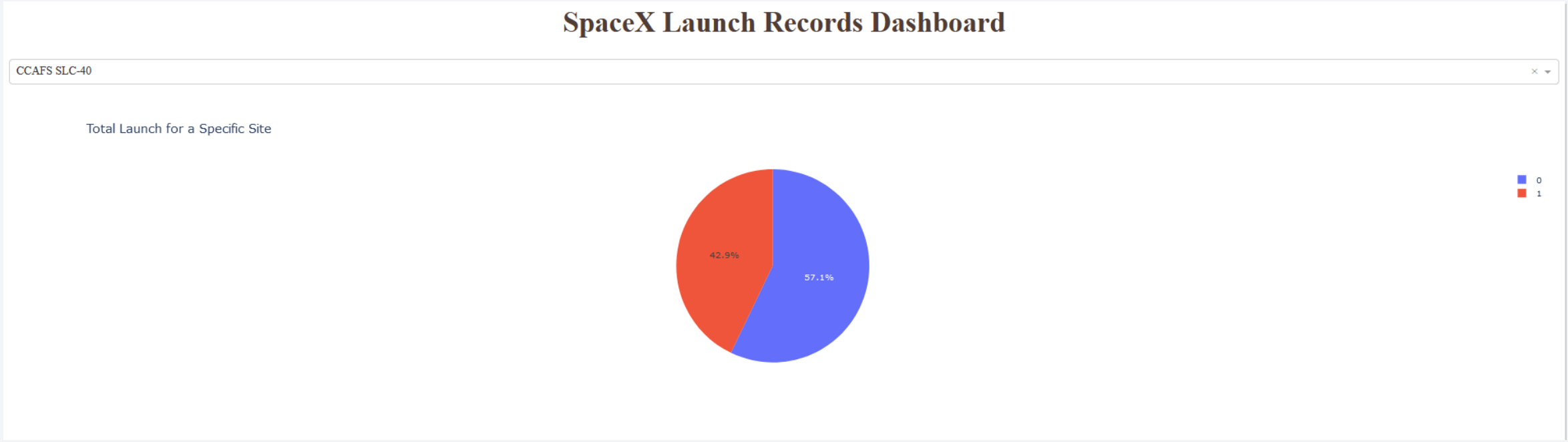
- Launch sites are NOT in close proximity to railways
- Launch sites are NOT close proximity to highways
- Launch sites are in close proximity to coastline.
- Launch sites keep certain distance away from cities.

Section 5

# Build a Dashboard with Plotly Dash

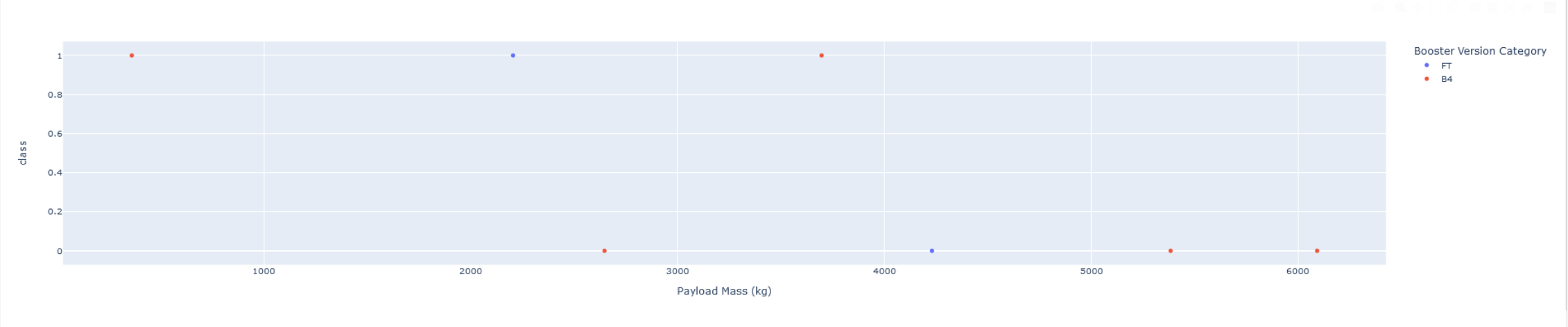


# Dashboard – Pie Chart



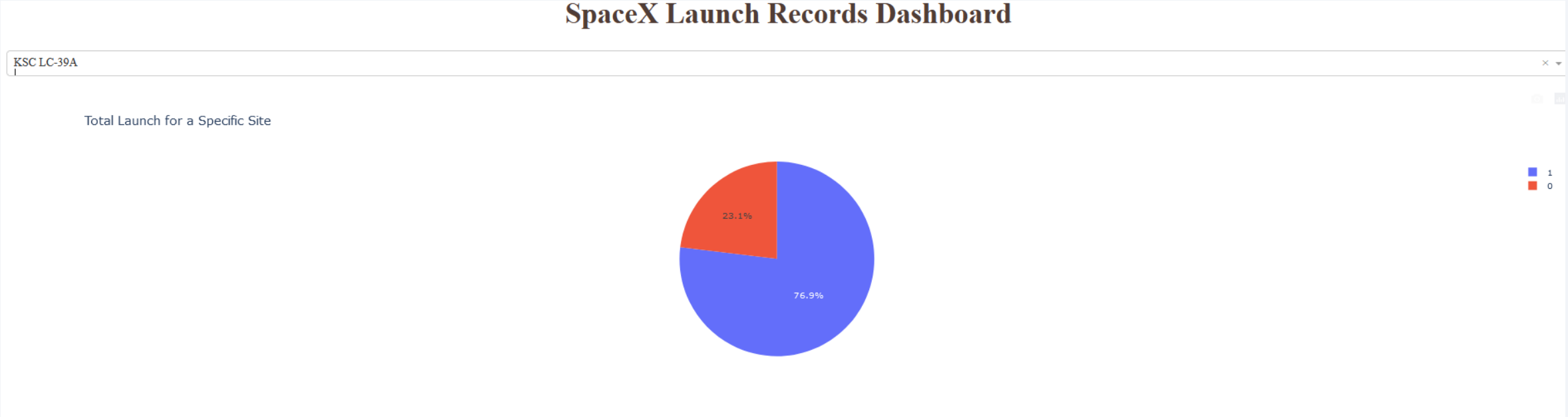

This chart shows the percentage of success and failure launch outcomes for different launch sites listed under the dropdown

# Dashboard – Scatter Chart




This chart shows the success/failure launch outcomes for different Payload Massed with respective of FT and B4 Boosters.

# Dashboard – Pie Chart




This chart shows that KSC LC 39-A Launch Site has almost 77% success rate

Section 6

# Predictive Analysis (Classification)

# Classification Accuracy

Logistic Regression:

```
print("TUNED HYPERPARAMETERS : ",logreg_cv.best_params_)
print("ACCURACY: ",logreg_cv.best_score_)
```

```
TUNED HYPERPARAMETERS :  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
ACCURACY:  0.8464285714285713
```

Support Vector Machines:

```
print("TUNED HYPERPARAMETERS: ",svm_cv.best_params_)
print("ACCURACY :",svm_cv.best_score_)
```

```
TUNED HYPERPARAMETERS:  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
ACCURACY : 0.8482142857142856
```

Decision Tree:

```
print("TUNED HYPERPARAMETERS ",tree_cv.best_params_)
print("ACCURACY",tree_cv.best_score_)
```

```
TUNED HYPERPARAMETERS  {'criterion': 'entropy', 'max_depth': 18, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_
samples_split': 5, 'splitter': 'random'}
ACCURACY 0.8892857142857142
```

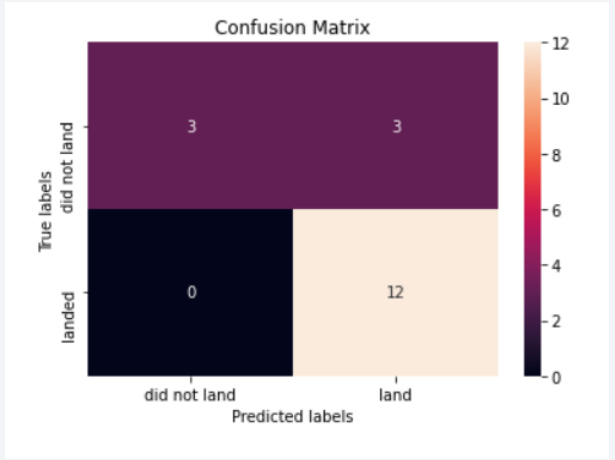
KNN:

```
print("TUNED HYPERPARAMETERS" ,knn_cv.best_params_)
print("ACCURACY",knn_cv.best_score_)
```

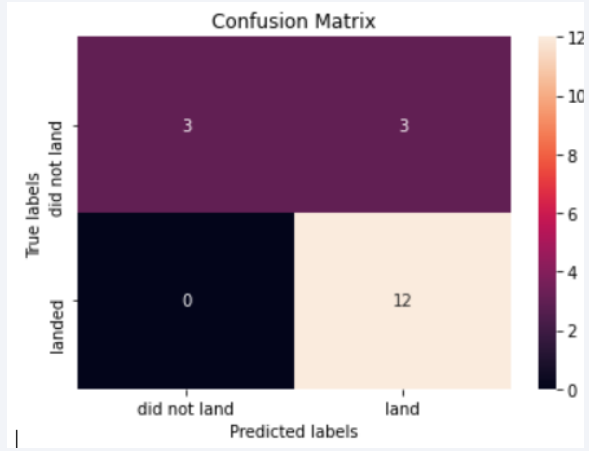
```
TUNED HYPERPARAMETERS {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
ACCURACY 0.8482142857142858
```

# Confusion Matrix

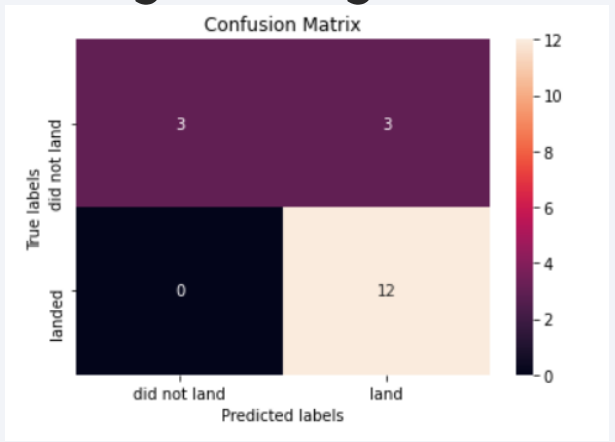
Shown here are the Confusion Matrix for 4 different classification algorithms



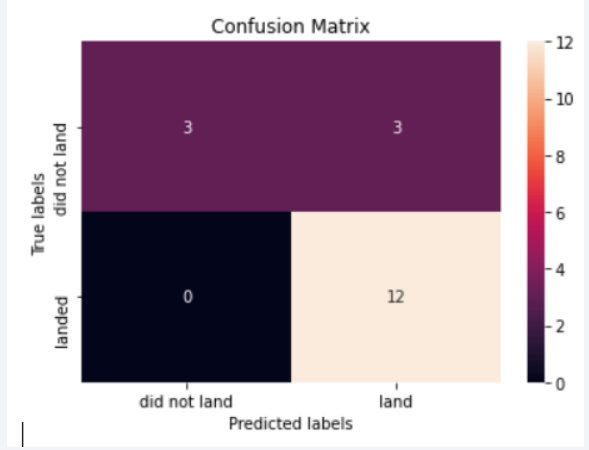

Logistic Regression




Support Vector Machine




Decision Tree




KNN

# Conclusions

- The Decision Tree  is the best Machine Learning Classifier Algorithm
- Lower Payloads have higher success rates in comparison with Higher Payloads.
- Success rates are higher in recent years
- KSC LC 39A had the most successful launches from all the sites
- Orbit GEO,HEO,SSO,ES L1 has the best Success Rate.

# Appendix

My GitHub

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

Extracting Column names from html

```
algorithms = {'knn':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
best_method = max(algorithms, key=algorithms.get)
print('BEST METHOD IS',best_method,'WITH',algorithms[best_method])
if best_method == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if best_method == 'knn':
    print('Best Params is :',knn_cv.best_params_)
if best_method == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
```

```
BEST METHOD IS Tree WITH 0.8767857142857143
Best Params is : {'criterion': 'gini', 'max_depth': 2, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_sp
lit': 5, 'splitter': 'random'}
```

Identifying best method

# Thank you!