
# Winning Space Race with Data Science

Akash Sharma
11/10/2021

# Outline

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

# Executive Summary

- Summary of methodologies
  - Data collection
  - Data wrangling
  - EDA with Data visualization
  - EDA with SQL
  - Interactive visual analytics with Folium
  - Building interactive dashboard with Dash
  - Predictive analysis
- Summary of all results
  - Data visualization result
  - Model result

# Introduction

- Project background and context

  We have predicted if the Falcon 9 first stage will land successfully. 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.

- Problems you want to find answers
  - SpaceX will reuse its first stage of falcon 9?
  - What are the factor affecting the successful landing of rocket?

Section 1

# Methodology

Click to add text

# Methodology

## Executive Summary

- Data collection methodology:
  - SpaceX data using API
  - Gathering data from Wikipedia
- Perform data wrangling
  - Preprocessing data, features selection for model development
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Scikit-learn framework for model building and evaluation

# Data Collection

# Data Collection – SpaceX API

[Github URL to Notebook](#)
*(click here)*

**1. Fetching response form API**

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

**2. Create response dataframe using json response**

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

**3. Getting data**

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

**4. Create data structure and then dataframe**

```
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}

launch_data = pd.DataFrame(launch_dict)
```

**5. Filter required data**

```
data_falcon9 = launch_data[launch_data['BoosterVersion']=='Falcon 9']
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
```

**6. Save the final data as '.csv' file**

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

# Data Collection - Scraping

[Github URL to Notebook](#)
*(click here)*

**1. Fetching response form static url**

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

**2. Creating a Soup Object**

```
soup = BeautifulSoup(response.text, "html.parser")
```

**3. Finding table in response**

```
html_tables = soup.find_all('table')
first_launch_table = html_tables[2]
```

**4. Getting name of the columns**

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

**5. Creating a dictionary**

```
launch_dict= dict.fromkeys(column_names)

del launch_dict['Date and time ( )']

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

**6. Store the data into dictionary**
*(For full code refer code cell 13 in Notebook)*

```
extracted_row = 0

for table_number,table in enumerate(soup
    for rows in table.find_all("tr"):
        if rows.th:
            if rows.th.string:
```

**7. Creating dataframe using dictionary**

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

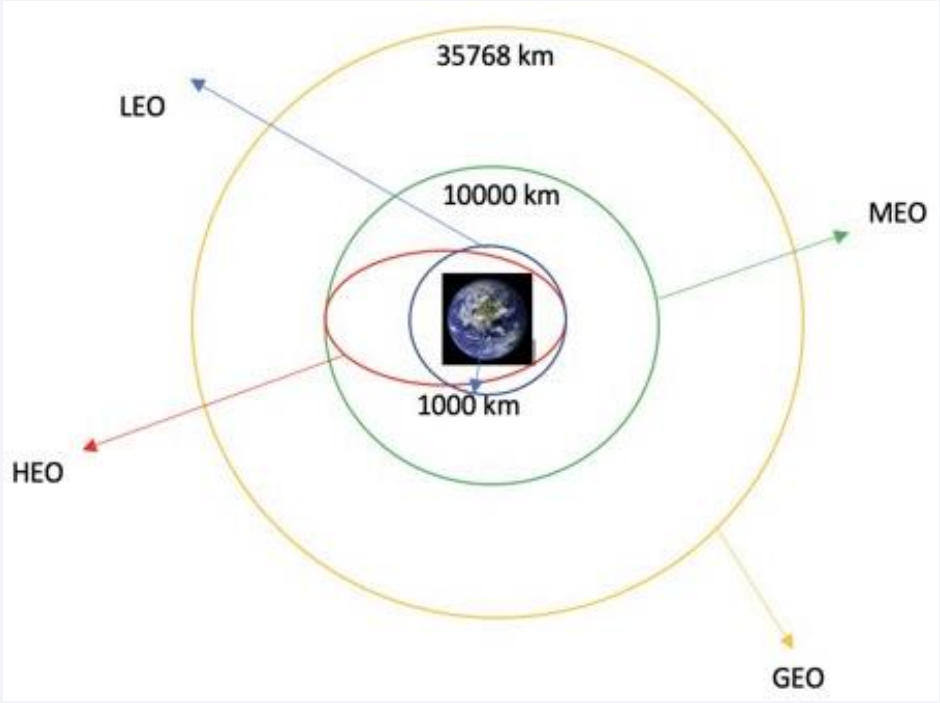
**8. Save the final data as '.csv' file**

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

# Data Wrangling

**Introduction**

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` means the mission outcome was successfully landed to a specific region of the ocean while `False Ocean` means the mission outcome was unsuccessfully landed to a specific region of the ocean. `True RTLS` means the mission outcome was successfully landed to a ground pad `False RTLS` means the mission outcome was unsuccessfully landed to a ground pad.`True ASDS` means the mission outcome was successfully landed on a drone ship `False ASDS` means the mission outcome was unsuccessfully landed on a drone ship.


[Github URL to Notebook](#)
*(click here)*

# EDA with Data Visualization

## Scatter Graphs :

- Flight Number V/s Payload Mass
- Flight Number V/s Launch Site
- Payload Mass V/s Launch Site
- Flight Number V/s Orbit Type
- Payload V/s Orbit Type

A scatter plot is a type of plot or mathematical diagram using Cartesian coordinates to display values for typically two variables for a set of data

[Github URL to Notebook](#)
*(click here)*

## Bar Graph :

- Success Rate V/s Orbit Type

A bar chart or bar graph is a chart or graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent.

## Line Graph :

- Launch success yearly trend

A line graph is a type of chart used to show information that changes over time. We plot line graphs using several points connected by straight lines.

# EDA with SQL

## SQL query used to gather information from the dataset.

- *Display the name of unique site launch 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 achieved.*
- *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.*
- *List the failed `landing_outcomes` in drone ship, their booster versions, and launch site names for in year 2015.*
- *Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.*

Github URL to Notebook
*(click here)*

# Build an Interactive Map with Folium

The launch success rate may depend on many factors such as payload mass, orbit type, and so on. It may also depend on the location and proximities of a launch site, i.e., the initial position of rocket trajectories. Finding an optimal location for building a launch site certainly involves many factors and hopefully we could discover some of the factors by analyzing the existing launch site locations.

Github URL to Notebook
*(click here)*

- **TASK 1:** Mark all launch sites on a map
- **TASK 2:** Mark the success/failed launches for each site on the map
- **TASK 3:** Calculate the distances between a launch site to its proximities

**Function to calculate the distance**

```
from math import sin, cos, sqrt, atan2, radians

def calculate_distance(lat1, lon1, lat2, lon2):
    # approximate radius of earth in km
    R = 6373.0

    lat1 = radians(lat1)
    lon1 = radians(lon1)
    lat2 = radians(lat2)
    lon2 = radians(lon2)

    dlon = lon2 - lon1
    dlat = lat2 - lat1

    a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
    c = 2 * atan2(sqrt(a), sqrt(1 - a))

    distance = R * c
    return distance
```

# Build a Dashboard with Plotly Dash

**Building interactive dashboard with Ploty and Dash Framework**

**Pie chart showing the success rate at different launch sites.**

- Success rate comparison between all launch sites.
- Success vs Failure ratio for each launch site.

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

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

Github URL to Notebook
*(click here)*

# Predictive Analysis (Classification)

## Model Building

- Pandas to Load and manipulate data
- Numpy to create an array
- Standardize the data
- Splitting data for training and testing model
- Initializing ML Algorithm objects
- Load the object with parameters in GridSearchCV

## Model Evaluation

- Best parameter
- Calculating accuracy of model
- Creating confusion matrix

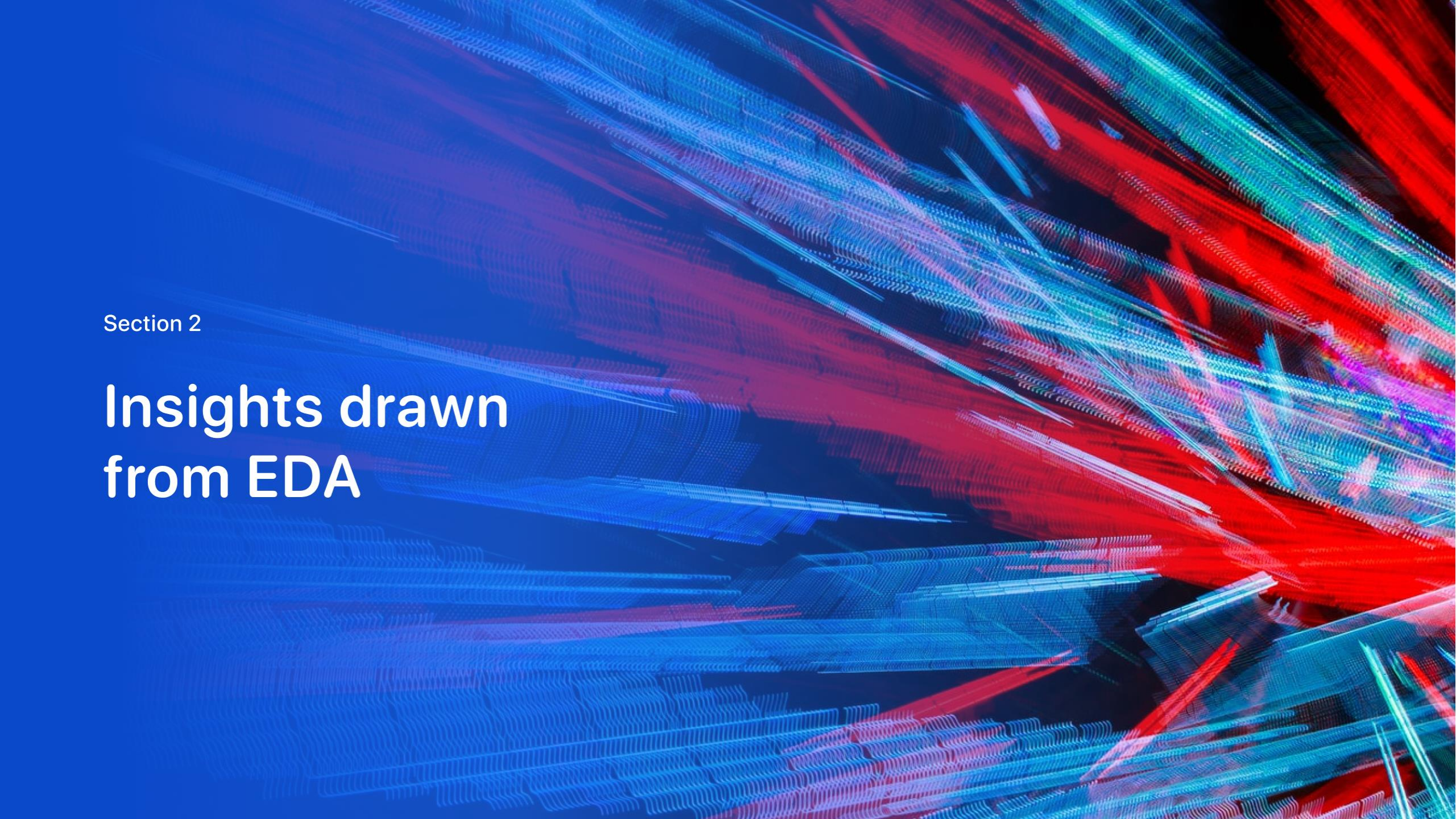
## Finding the best model

- Comparing the score of all the model and finding the best accuracy.
- The model report is created at the end of the notebook.

Github URL to Notebook
*(click here)*

# Results

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

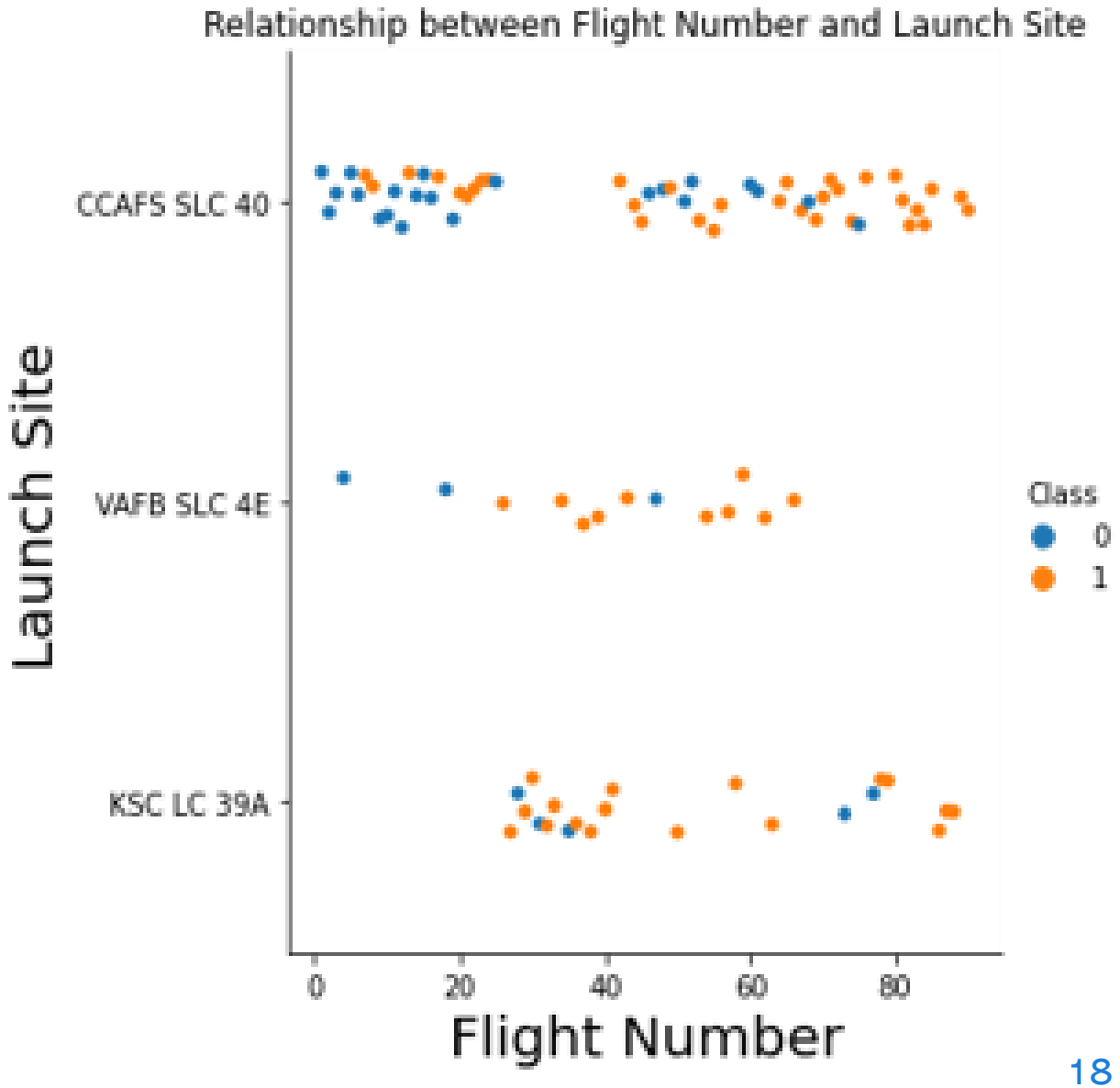


Section 2

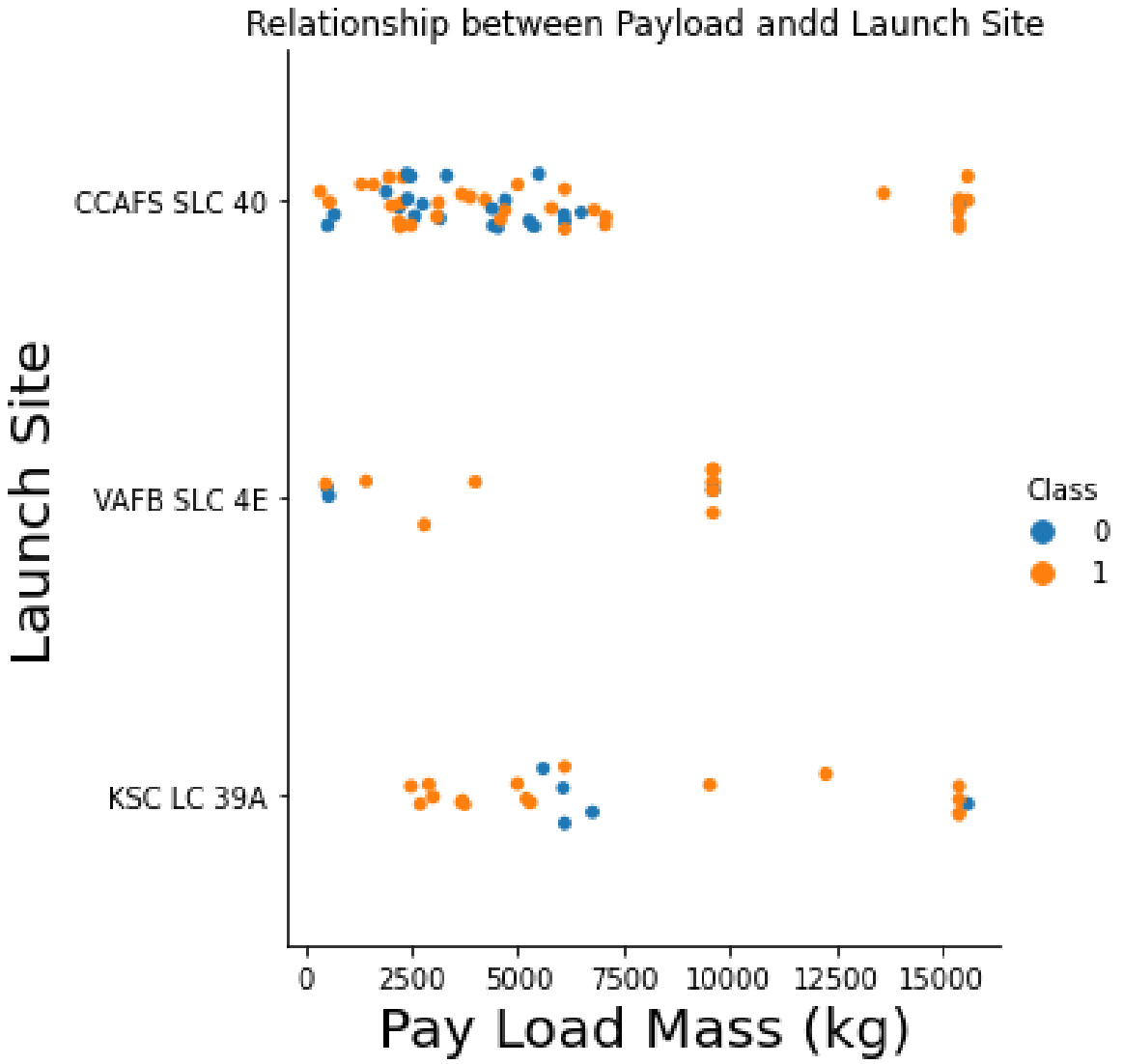
# Insights drawn from EDA

# Flight Number vs Launch Site

Higher amount of flight has success rate at different Launch Site.

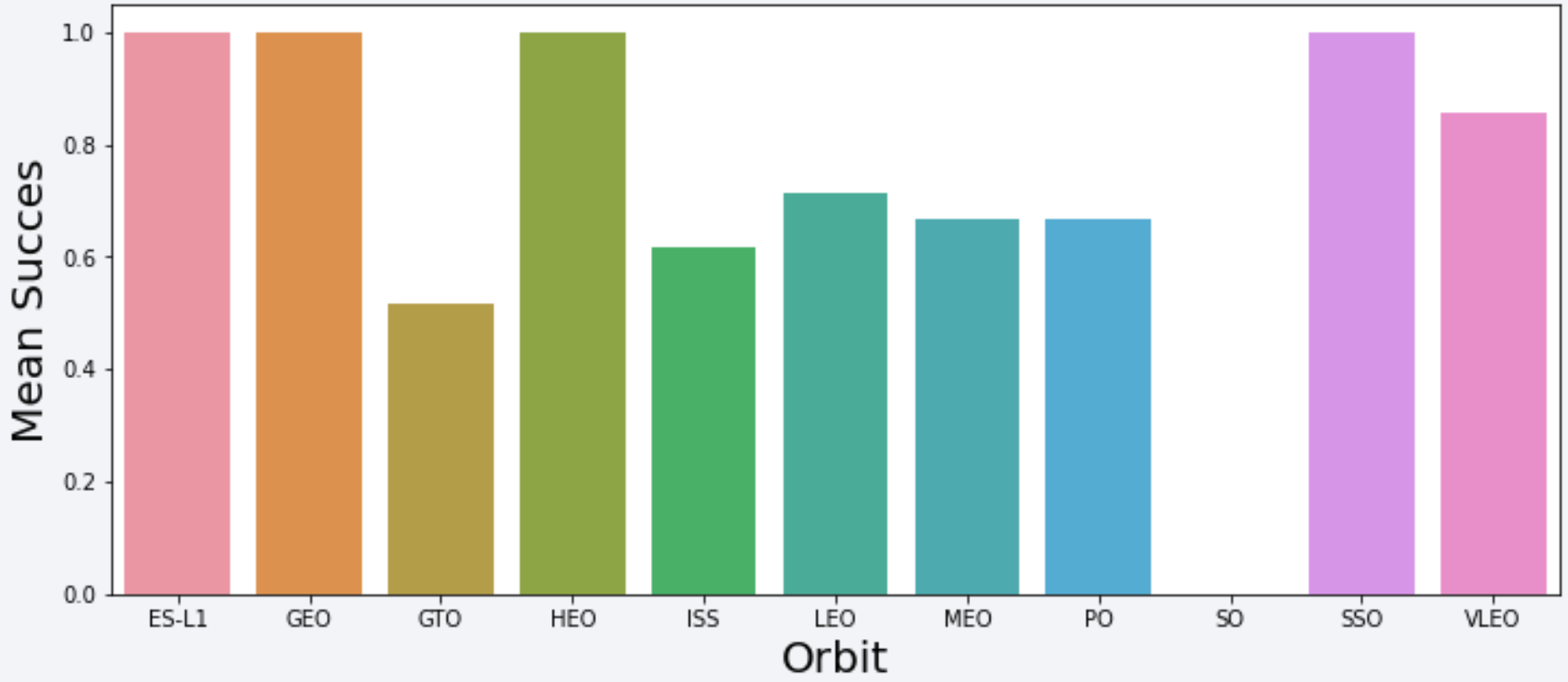


# Payload Mass vs Launch Site



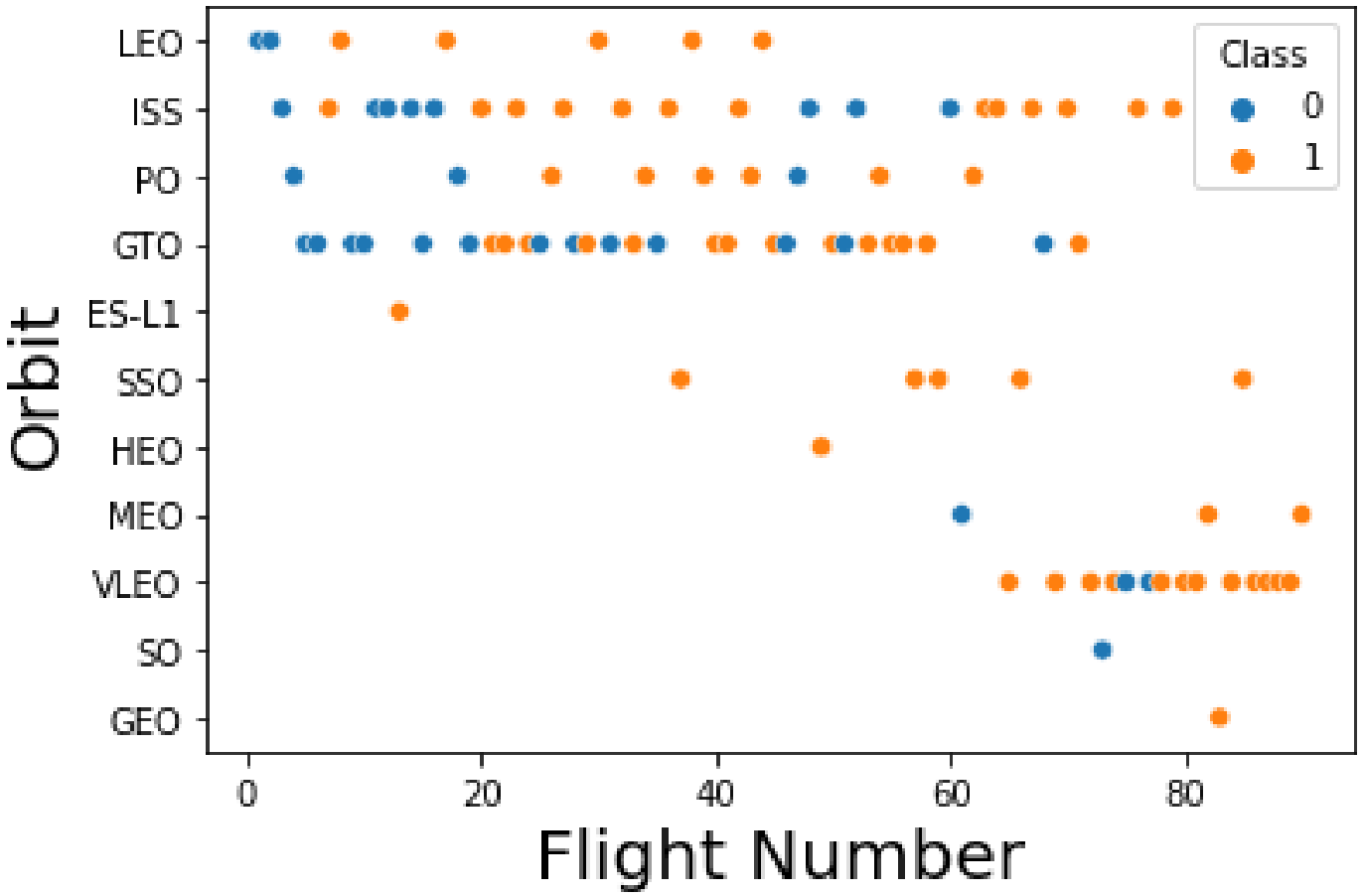

Higher payload mass seems to have high success rate.

# Success Rate vs. Orbit Type



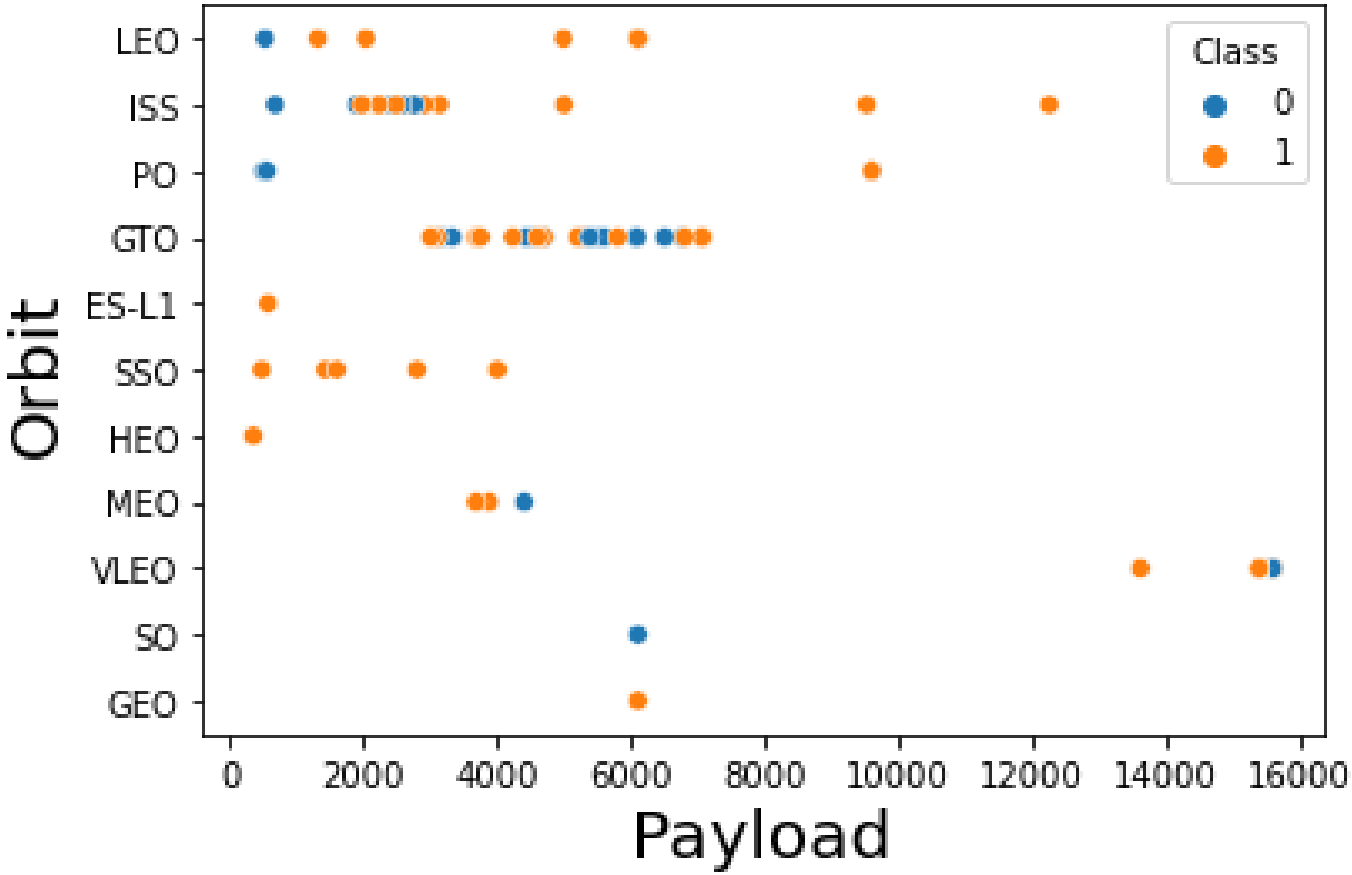

ES-L1, GEO, HEO, SS0 Orbit has the highest success rate.

# Flight Number vs Orbit Type



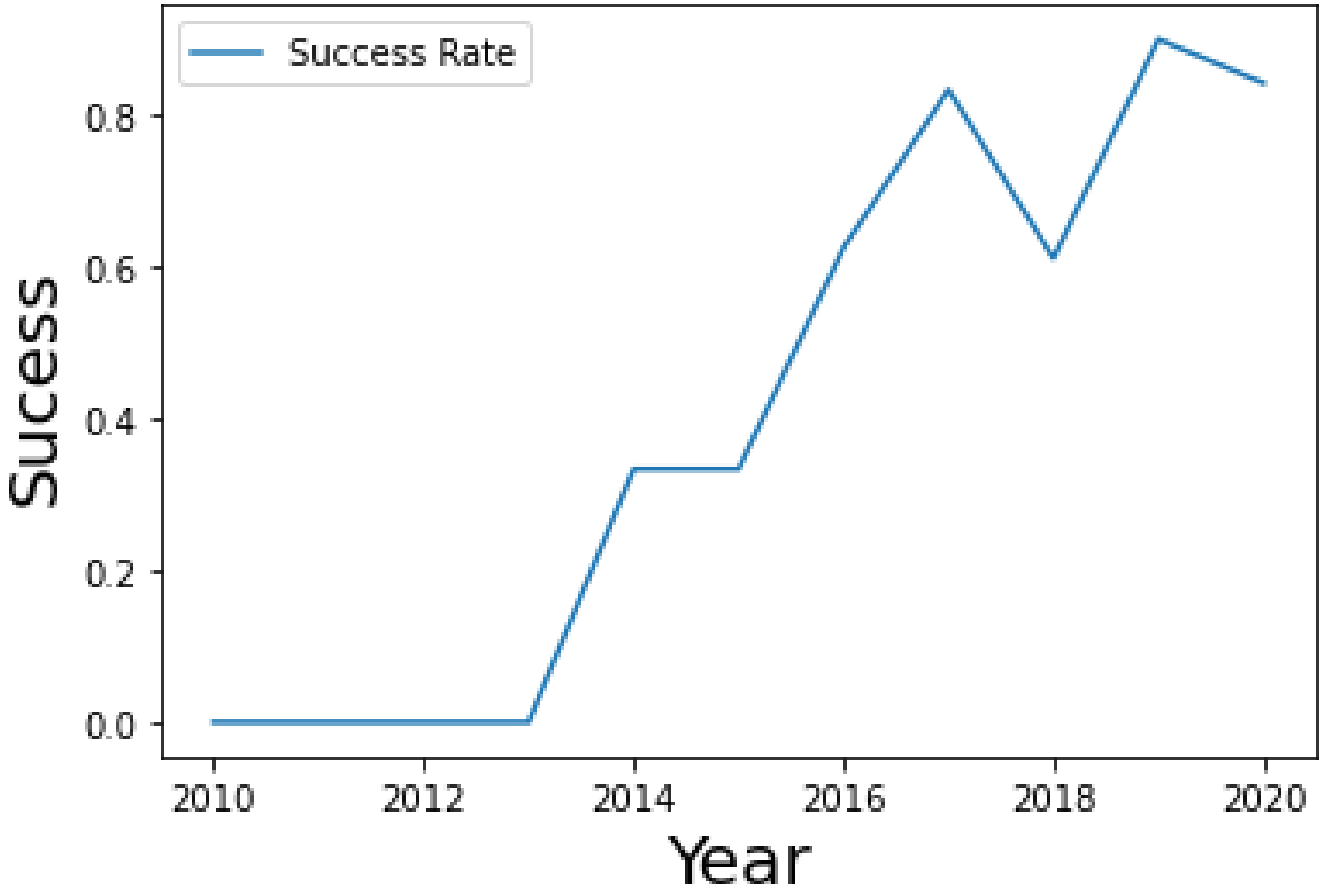

This plot doesn't show any confident relationship.

# Payload Mass vs Orbit Type




There are positive outcome at high payload mass for orbit type other tha GTO. So again we don't have strong evidence of orbit and payload relation.

# Launch Success Yearly Trend




Success rate increased with the coming year

# All Launch Site Names

***Output :***

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

## SQL Query:

***SELECT DISTINCT LAUNCH_SITE from SPACEXTBL***

*In above query we are selecting launch_site from SPACEXTBL and distinct keyword in sql query is used to select only unique launch_site (no repeatation).*

# Launch Site Names Begin with 'CCA'

## SQL Query:

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

*In above query we are selecting 5 record where launch_site starts with CCA.*

***Output :***

| DATE | time__utc_ | booster_version | launch_site | payload | payload_mass__kg_ | orbit | customer | mission_outcome | landing__outcome |
|---|---|---|---|---|---|---|---|---|---|
| 2010-06-04 | 18:45:00 | F9 v1.0 B0003 | CCAFS LC-40 | Dragon Spacecraft Qualification Unit | 0 | LEO | SpaceX | Success | Failure (parachute) |
| 2010-12-08 | 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) |
| 2012-05-22 | 07:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525 | LEO (ISS) | NASA (COTS) | Success | No attempt |
| 2012-10-08 | 00:35:00 | F9 v1.0 B0006 | CCAFS LC-40 | SpaceX CRS-1 | 500 | LEO (ISS) | NASA (CRS) | Success | No attempt |
| 2013-03-01 | 15:10:00 | F9 v1.0 B0007 | CCAFS LC-40 | SpaceX CRS-2 | 677 | LEO (ISS) | NASA (CRS) | Success | No attempt |

# Total Payload Mass

## SQL Query:

***SELECT SUM(payload_mass__kg_) as "Total Payload Mass" FROM SPACEXTBL WHERE CUSTOMER LIKE 'NASA (CRS)'***

*In above query we are calculating total payload mass carried by boosters launched by NASA (CRS).*

***Output :***

| Total Payload Mass |
| --- |
| 45596 |

# Average Payload Mass by F9 v1.1

## SQL Query:

***SELECT AVG(payload_mass__kg_) as "Average Payload Mass" FROM SPACEXTBL WHERE BOOSTER_VERSION LIKE 'F9 v1.1'***

*In above query we are calculating average payload mass for booster version 'F9 v1.1'.*

***Output :***

| Average Payload Mass |
| --- |
| 2928 |

# First Successful Ground Landing Date

## SQL Query:

***SELECT MIN(DATE) FROM SPACEXTBL WHERE LANDING__OUTCOME LIKE 'Success%'***

*In above query we are finding the date where the landing was successful and we are selecting the very first date using MIN() function.*

***Output :*** 1

2015-12-22

# Successful Drone Ship Landing with Payload between 4000 and 6000

## SQL Query:

***SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE LANDING__OUTCOME='Success (drone ship)' AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000***

*In above query we are selecting booster_version from spacextbl which had success drone ship landing and payload mass in range 4000 to 6000.*

***Output :***

| booster_version |
|---|
| F9 FT B1022 |
| F9 FT B1026 |
| F9 FT B1021.2 |
| F9 FT B1031.2 |

# Total Number of Successful and Failure Mission Outcomes

## SQL Query:

***SELECT MISSION_OUTCOME, COUNT(DATE) as COUNT FROM SPACEXTBL GROUP BY MISSION_OUTCOME***

*In above query we are selecting the total number of successful and failure mission outcomes.*

***Output :***

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

# Boosters Carried Maximum Payload

## SQL Query:

```
SELECT BOOSTER_VERSION, PAYLOAD_MASS__KG_ FROM (
   SELECT BOOSTER_VERSION, SUM(PAYLOAD_MASS__KG_) AS PAYLOAD_MASS__KG_
   FROM SPACEXTBL GROUP BY BOOSTER_VERSION
) ORDER BY PAYLOAD_MASS__KG_ DESC LIMIT 1
```

*In above query we are selecting the names of the booster_versions which have carried the maximum payload mass.*

***Output :***

| booster_version | payload_mass__kg_ |
|---|---|
| F9 B5 B1048.4 | 15600 |

# 2015 Launch Records

## SQL Query:

```
SELECT YEAR(DATE), LAUNCH_SITE, BOOSTER_VERSION, LANDING__OUTCOME
FROM SPACEXTBL
WHERE LANDING__OUTCOME LIKE 'Failure (drone ship)' AND YEAR(DATE)=2015
```

*In above query we are selecting the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015.*

***Output :***

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

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

## SQL Query:

```
SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) FROM SPACEXTBL
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY LANDING__OUTCOME
ORDER BY COUNT(LANDING__OUTCOME) DESC
```

*In above query we are selecting the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.*

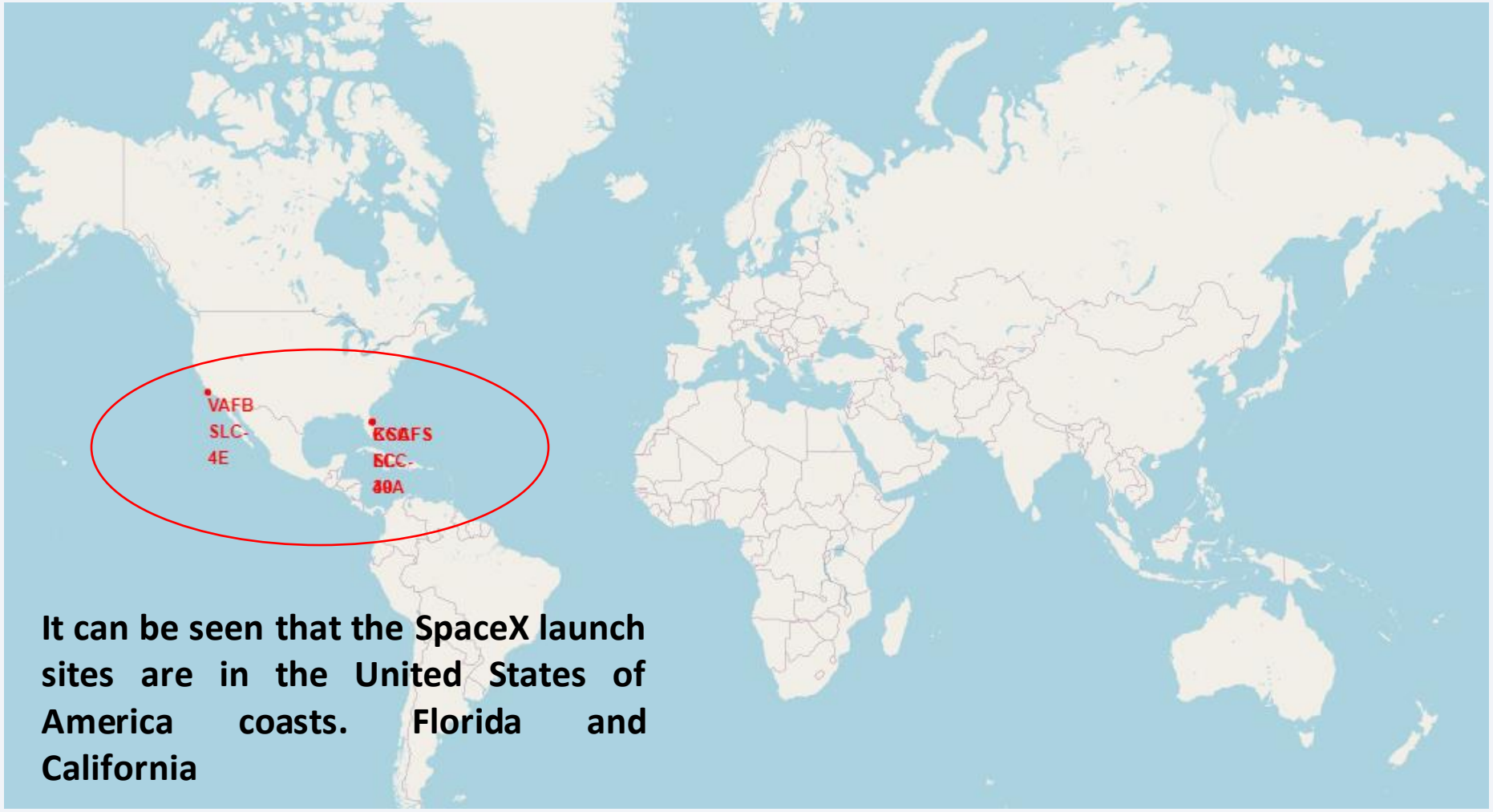
***Output :***

| landing__outcome | 2 |
|---|---|
| No attempt | 10 |
| Failure (drone ship) | 5 |
| Success (drone ship) | 5 |
| Controlled (ocean) | 3 |
| Success (ground pad) | 3 |
| Failure (parachute) | 2 |
| Uncontrolled (ocean) | 2 |
| Precluded (drone ship) | 1 |

Section 4

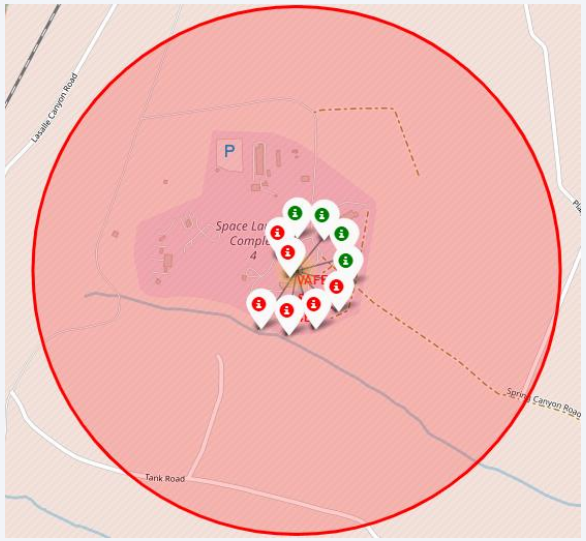
# Launch Sites Proximities Analysis

# All Launch Site on Map




**It can be seen that the SpaceX launch sites are in the United States of America coasts. Florida and California**

# Success/Failed Launch for each site on map

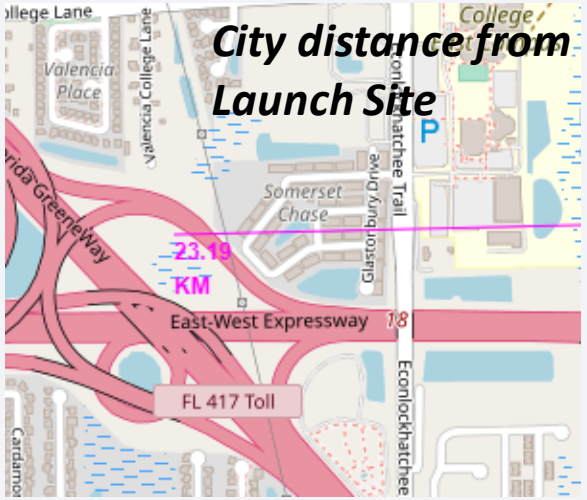



California Launch Site

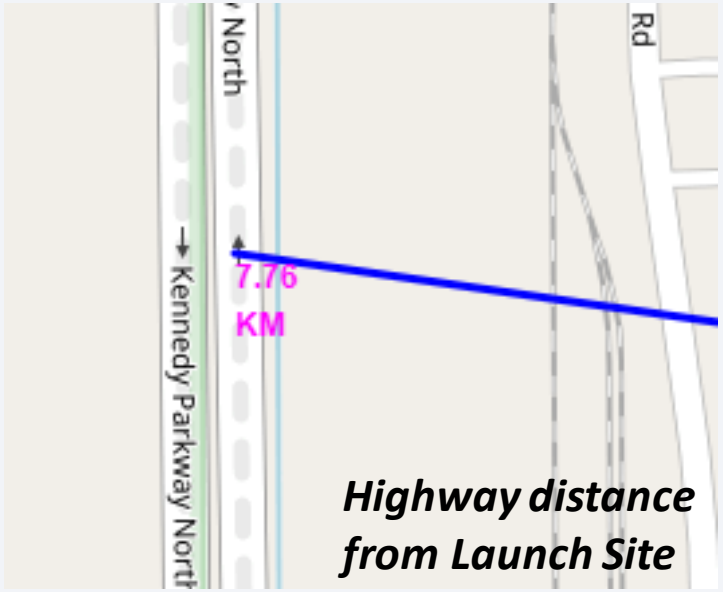

Florida Launch Site


Markers for all launch records. If a launch was successful (`class=1`), then we use a green marker and if a launch was failed, we use a red marker (`class=0`)

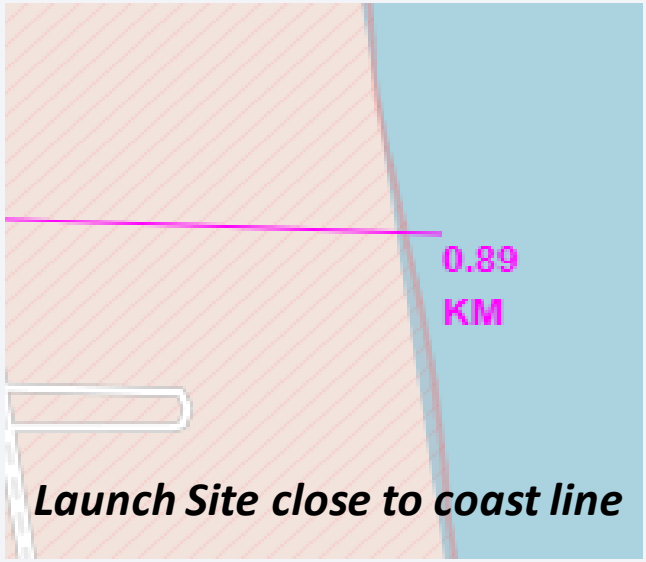
# Distance between Launch Site and its Proximities



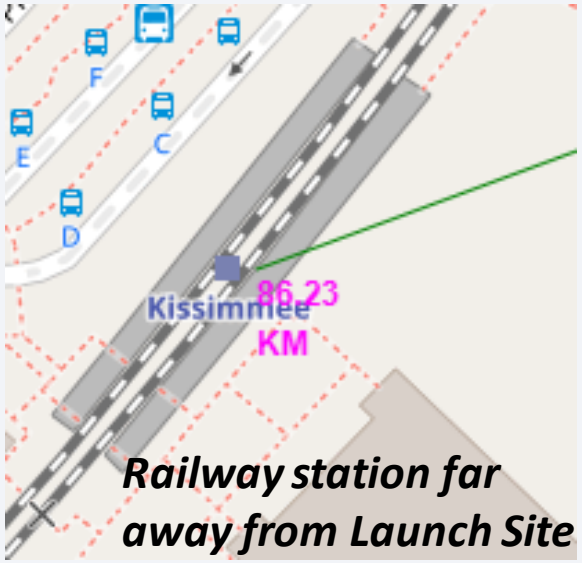

*City distance from Launch Site*




*Highway distance from Launch Site*

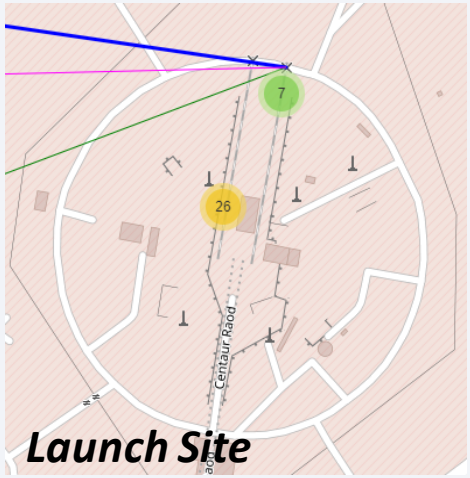



*Launch Site close to coast line*

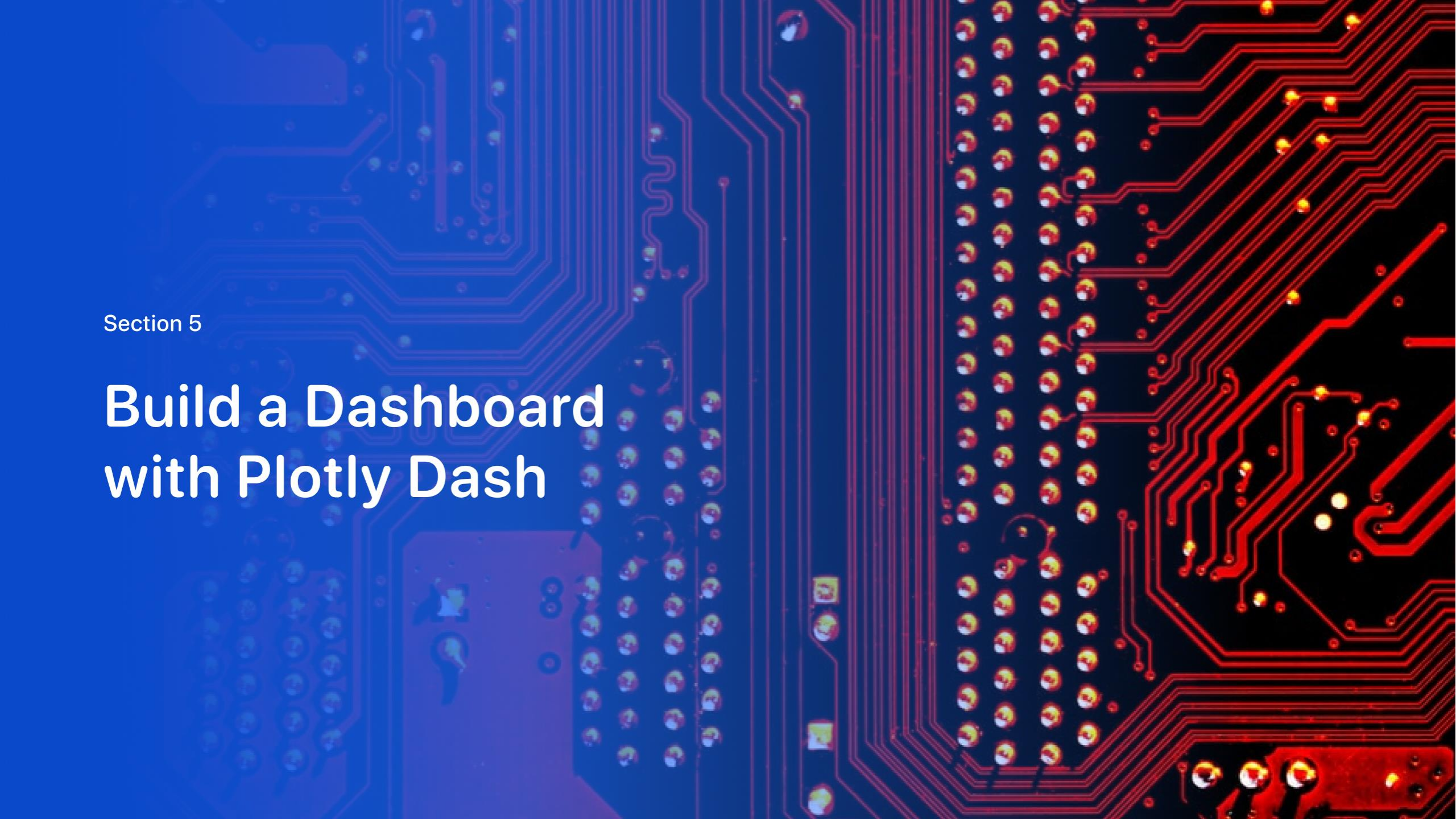



*Railway station far away from Launch Site*

- launch sites is in close proximity to coastline.
- Launch sites are far away from city.
- There is no launch sites in close proximity to highways.
- There's no launch sites in close proximity to railways.



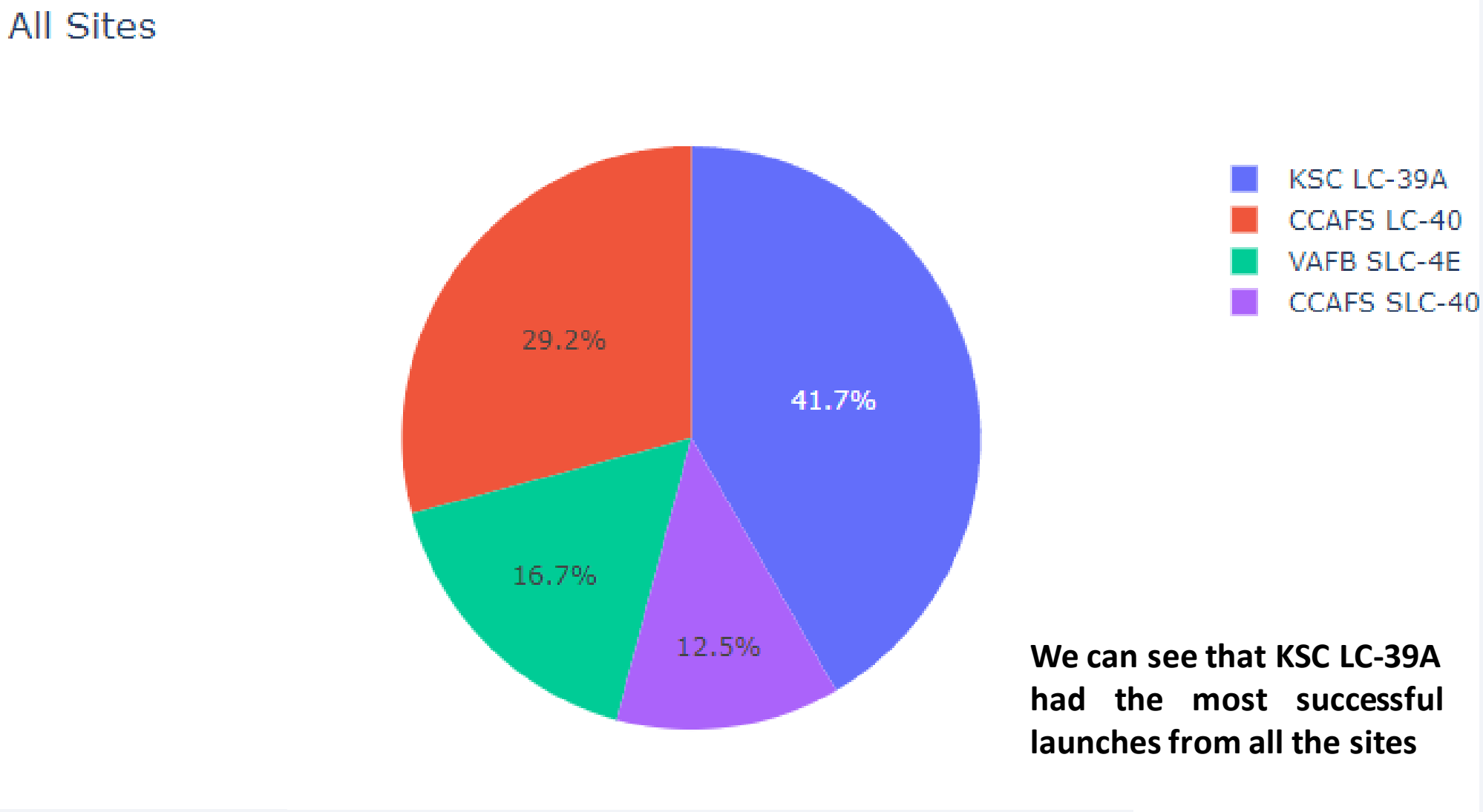

*Launch Site*



Section 5

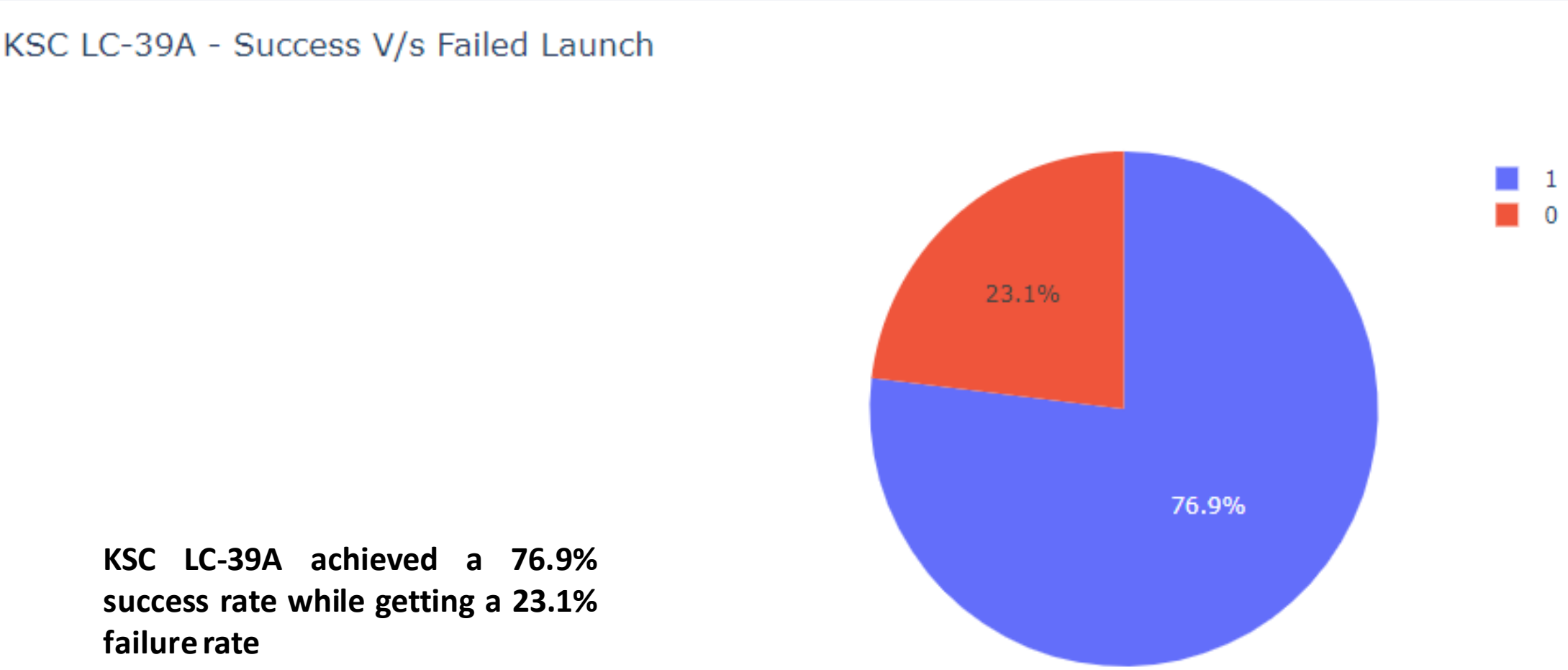
# Build a Dashboard with Plotly Dash

# Launch Success Percent for All Launch Site



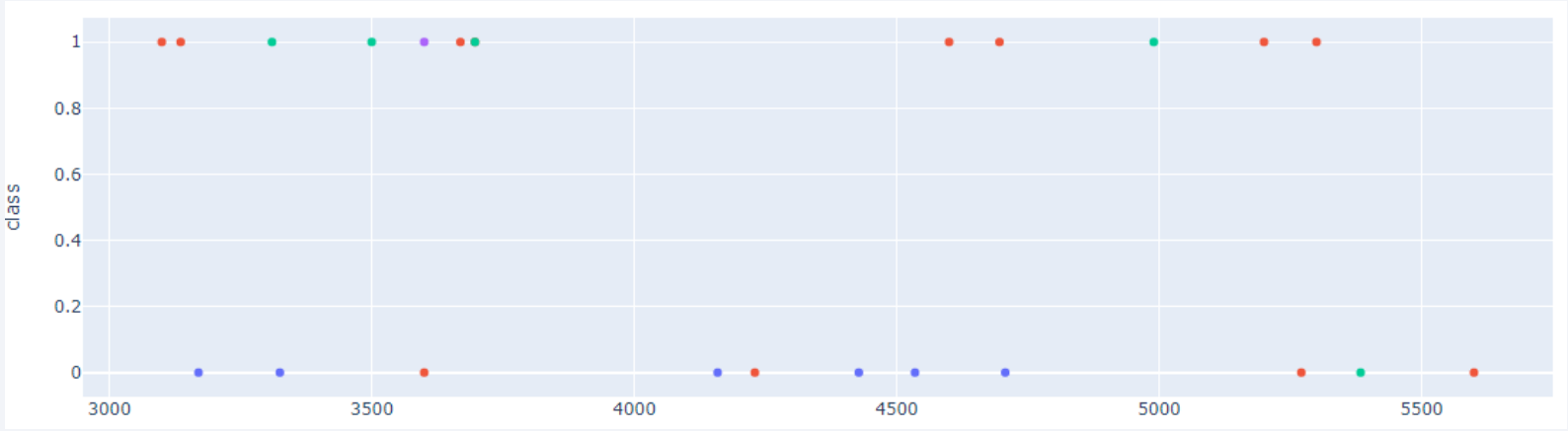

**We can see that KSC LC-39A had the most successful launches from all the sites**

# Pie chart for Launch Site with Highest Success Ratio

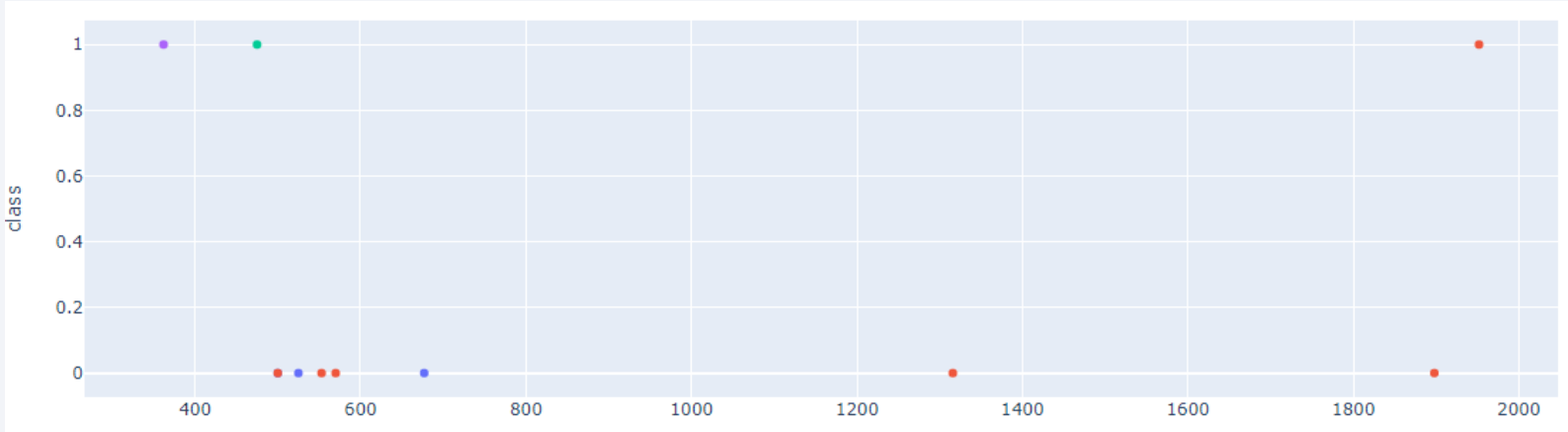



**KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate**

# Payload vs Success Outcome scatter plot




Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider




**Result :**

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

Section 6

# Predictive Analysis (Classification)

# Classification Accuracy

- Visualize the built model accuracy for all built classification models, in a bar chart
- Find which model has the highest classification accuracy

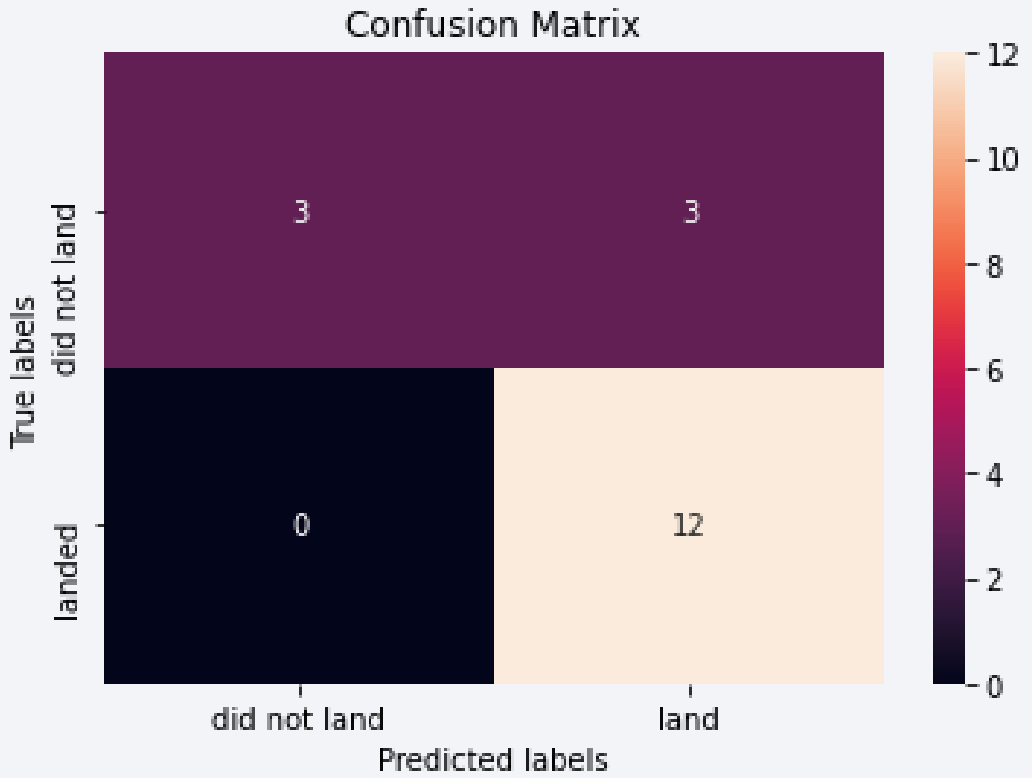
| | F1-score | Accuracy |
|---|---|---|
| Algorithm | | |
| Logistic Regression | 0.888889 | 0.833333 |
| SVM | 0.888889 | 0.833333 |
| Decision Tree | 0.916667 | 0.888889 |
| K Nearest Neighbor | 0.888889 | 0.833333 |

```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 18, 'max_features': 'sqrt', 'min_samples_leaf': 4,
'min_samples_split': 5, 'splitter': 'best'}
accuracy : 0.8875000000000002
```

# Confusion Matrix

We conclude that the Decision Tree was the best classifier which was able to predict the difference between two classes and also we have problem with false positive.

| | Predicted 0 | Predicted 1 |
|---|---|---|
| Actual 0 | **TN** TRUE NEGATIVE | **FP** FALSE POSITIVE |
| Actual 1 | **FN** FALSE NEGATIVE | **TP** TRUE POSITIVE |



# Conclusions

- Decision Tree Algorithm is the best classifier for this classification problem.
- Low weighted payload performs better than heavy payload.
- ES-L1, GEO, HEO, SSO Orbit has the highest success rate.
- KSC LC-29A launch site has highest success rate.
- Success rate increased with the coming year.

# Appendix

- ❑ Google Map to find nearest co-ordinate
- ❑ Python for Data Analysis (Notebook)
- ❑ Plotly Docs
- ❑ Folium Docs
- ❑ SQLALCHEMY

# Thank you!