IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Godfrey N Sanhehwe

06-07-22

Outline

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

Executive Summary

- Methodology:
  - Data Collection through API
  - Data Collection by Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis using SQL
  - Exploratory Data Analysis using Data Visualization
  - Interactive Visual Analytics using Folium
  - Prediction by Machine Learning
- Results:
  - Exploratory Data Analysis
  - Interactive analytics
  - Predictive Analytics

Introduction

- Background of Project:

  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 Space X can reuse the first stage, hence, if we can determine the landing of the first stage, we can determine the cost of launch. The goal of this project is to predict whether or not first stage will land successfully by utilizing machine learning.

- Questions to be answered:
  - What factors determine rocket landing?
  - How features interact in determining landing success rate?
  - What operating conditions ensure successful landing?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Using SpaceX API and web scraping from Wikipedia.
- Perform data wrangling
  - Applying One-hot encoding to categorical features.
- Perform exploratory data analysis by SQL and visualization
- Perform interactive visual analytics by Plotly Dash and Folium
- Perform predictive analysis using classification algorithms
  - Building, tuning, and evaluating classifiers to find the best to deploy.

Data Collection

- To collect the data, we:
  - "Get request" to SpaceX API for data collection
  - Decoded the response content as a Json using .json() function call and turned it to a pandas dataframe using .json_normalize().
  - Cleaned the data, as well checked for missing values and replaced with mean averages where necessary.
  - Also performed web scraping from Wikipedia for Falcon 9 launch records using BeautifulSoup.
  - The objective here was to extract the launch records as HTML table, parse the table, then convert it to a pandas dataframe for forthcoming analyses.

Data Collection – SpaceX API

- Used the 'get request' to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.
- The link to the notebook is https://github.com/digoznyasha/testrepo/blob/main/Data_Collection_API.ipynb

1. Get request for rocket launch data using API

In [6]:
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

In [7]:
```
response = requests.get(spacex_url)
```

2. Use json_normalize method to convert json result to dataframe

In [12]:
```
# Use json_normalize method to convert the json result into a dataframe

# decode response content as json
static_json_df = res.json()
```

In [13]:
```
# apply json_normalize
data = pd.json_normalize(static_json_df)
```

3. We then performed data cleaning and filling in the missing values

In [30]:
```
rows = data_falcon9['PayloadMass'].values.tolist()[0]

df_rows = pd.DataFrame(rows)
df_rows = df_rows.replace(np.nan, PayloadMass)

data_falcon9['PayloadMass'][0] = df_rows.values
data_falcon9
```

Data Collection – Web-Scrapping

- Applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup
- Parsed the table and converted it into a pandas dataframe.
- The link to the notebook is https://github.com/digoznyasha/testrepo/blob/main/Data_Collection_WebScraping.ipynb

1. Apply HTTP Get method to request the Falcon 9 rocket launch page

In [4]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

In [5]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
html_data = requests.get(static_url)
html_data.status_code
```

Out[5]:
```
200
```

2. Create a BeautifulSoup object from the HTML response

In [6]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html_data.text, 'html.parser')
```

Print the page title to verify if the `BeautifulSoup` object was created properly

In [7]:
```
# Use soup.title attribute
soup.title
```

Out[7]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

3. Extract all column names from the HTML table header

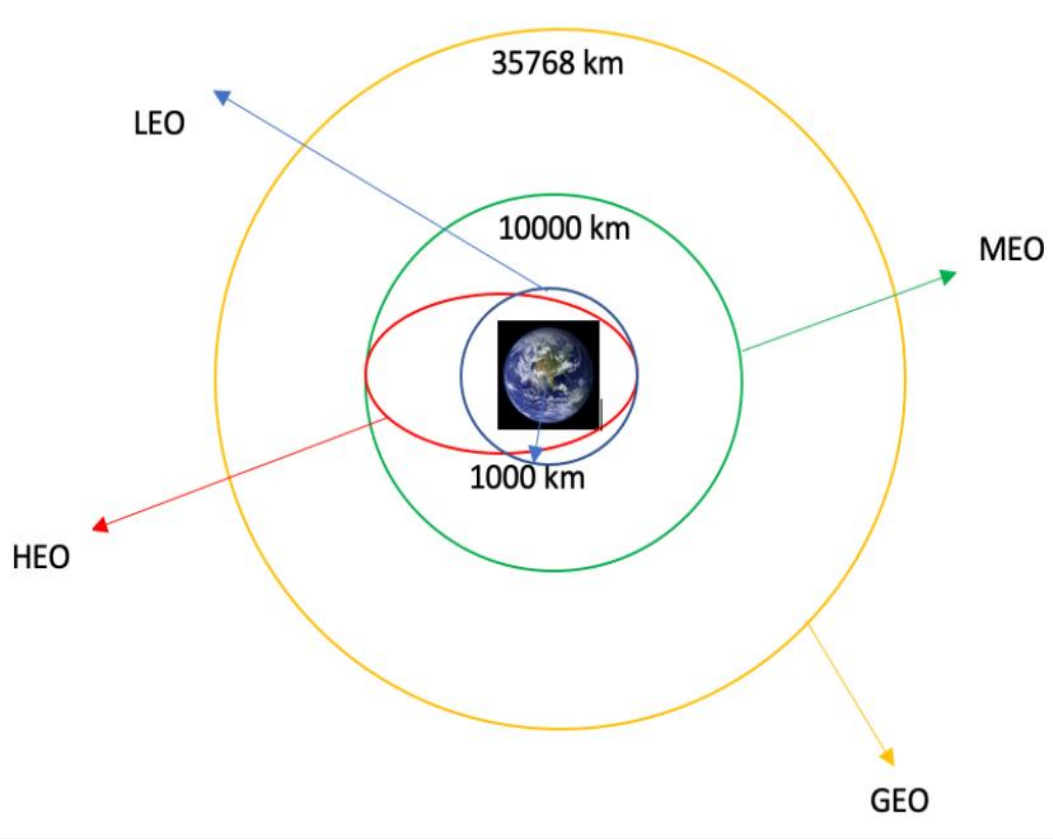
In [10]:
```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names

element = soup.find_all('th')
for row in range(len(element)):
    try:
        name = extract_column_from_header(element[row])
        if (name is not None and len(name) > 0):
            column_names.append(name)
    except:
        pass
```

4. Create a dataframe by parsing the launch HTML tables
5. Export data to csv

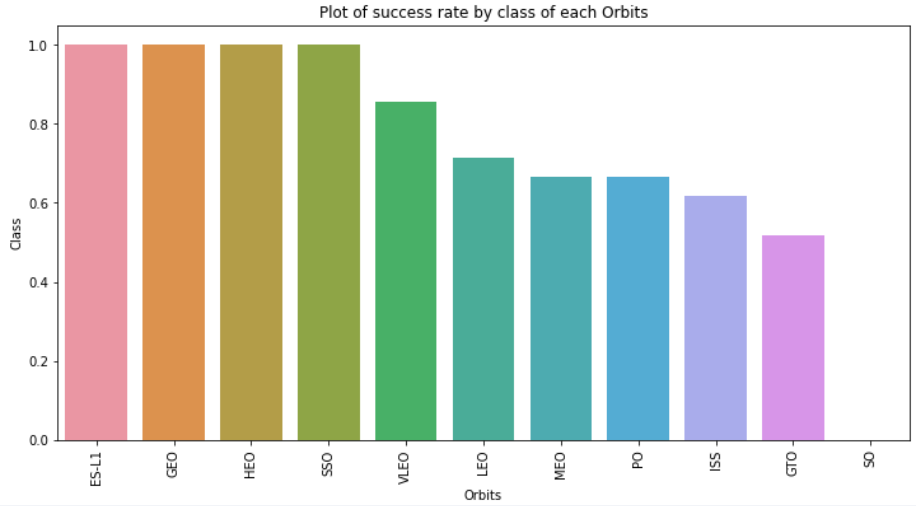
Data Wrangling

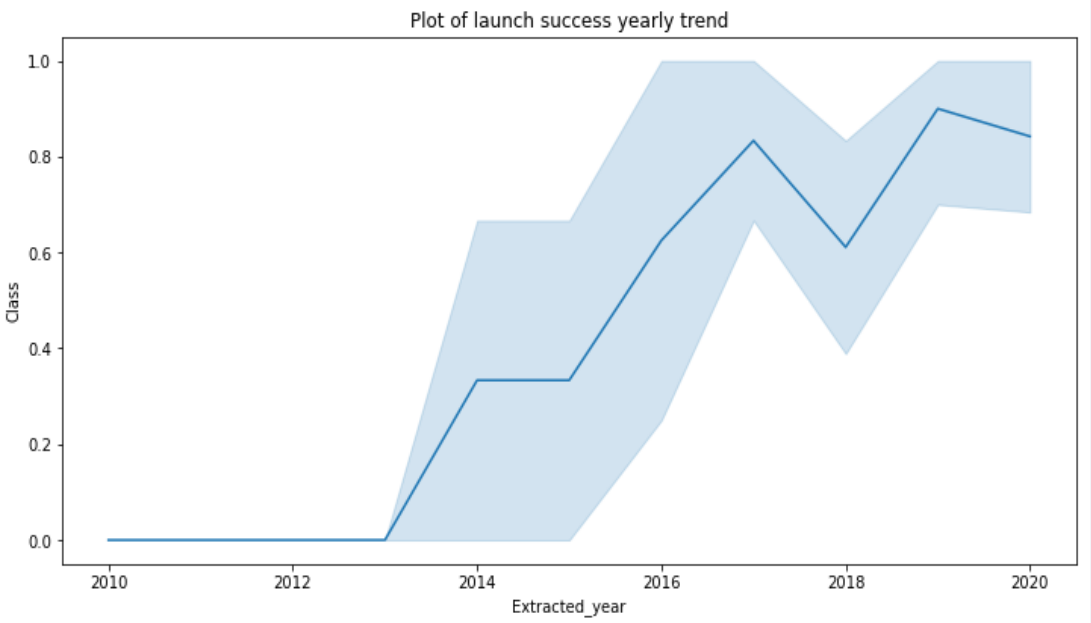


- Performed exploratory data analysis and determined the training labels.
- Calculated the number of launches at each site, and the number and occurrences of each orbits
- Created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is https://github.com/digoznyasha/testrepo/blob/main/Data_Wrangling_Final_Assignment.ipynb

EDA by Data Visualization

- Explored the data by visualizing relationships between flight number and launch site, payload and launch site, success rate of each orbit type, flight number and orbit type and finally the yearly launch success trend.





- The link to the notebook is https://github.com/digoznyasha/testrepo/blob/main/jupyter-labs-eda-dataviz.ipynb

EDA by SQL

- Loaded the SpaceX dataset into a PostgreSQL database via Jupyter notebook.
- Applied EDA with SQL to get insight from the data. Queries were performed to retrieve the following example cases:
  - Names of unique launch sites in the space mission.
  - Total payload mass carried by boosters launched by NASA.
  - Average payload mass carried by booster version F9 v1.1.
  - Total number of successful and unsuccessful mission outcomes.
  - Failed landing outcomes in drone ship, booster version and names of launch site.
  - The link to the notebook is https://github.com/digoznyasha/testrepo/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Building an Interactive Map by Folium

- Marked all launch sites and added map objects such as markers, circles and lines to mark the success or failure of launches for each site on the folium map.
- Assigned feature launch outcomes i.e. failure or success by encoding 0 for failure and 1 for success.
- Identified which launch sites have relatively high success rate using color-labeled marker clusters.
- Calculated distances between a launch site and its proximities. Some of the questions answered included:
  - Are launch sites near railways, highways and coastlines?
  - Do launch sites keep certain distances away from cities?

Build a Dashboard by Plotly Dash

- Built an interactive dashboard with PlotlyDash.
- Plotted pie charts showing the total launches by sites.
- Plotted scatter graph showing the relationship with Outcome and Payload mass for differing booster version.
- The link to the notebook is https://github.com/digoznyasha/testrepo/blob/main/Interactive_Dash_App.py

Predictive Analysis (Classification)

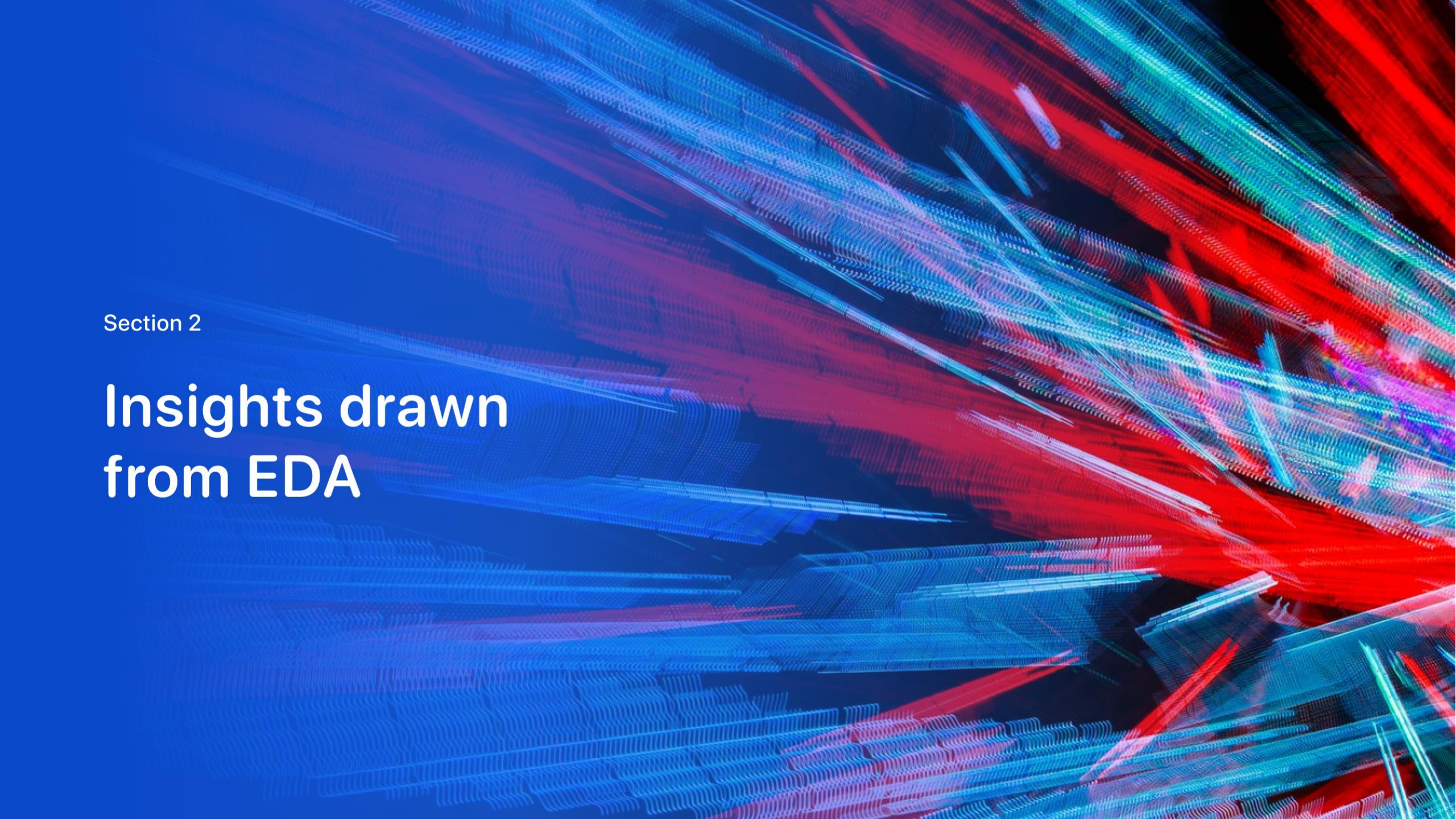
- Loaded the data using Numpy and Pandas libraries, transformed and split the data to train and test sets.
- Built different machine learning models and fine tuned hyper-parameters using GridSearchCV.
- Used accuracy as the metric for model improvement by algorithm tuning and feature engineering
- Finally selected the best performing classification model for the project.
- The link to the notebook is https://github.com/digoznyasha/testrepo/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results:

- From
  - Exploratory data analysis
  - Interactive Analytics
  - Predictive Analytics

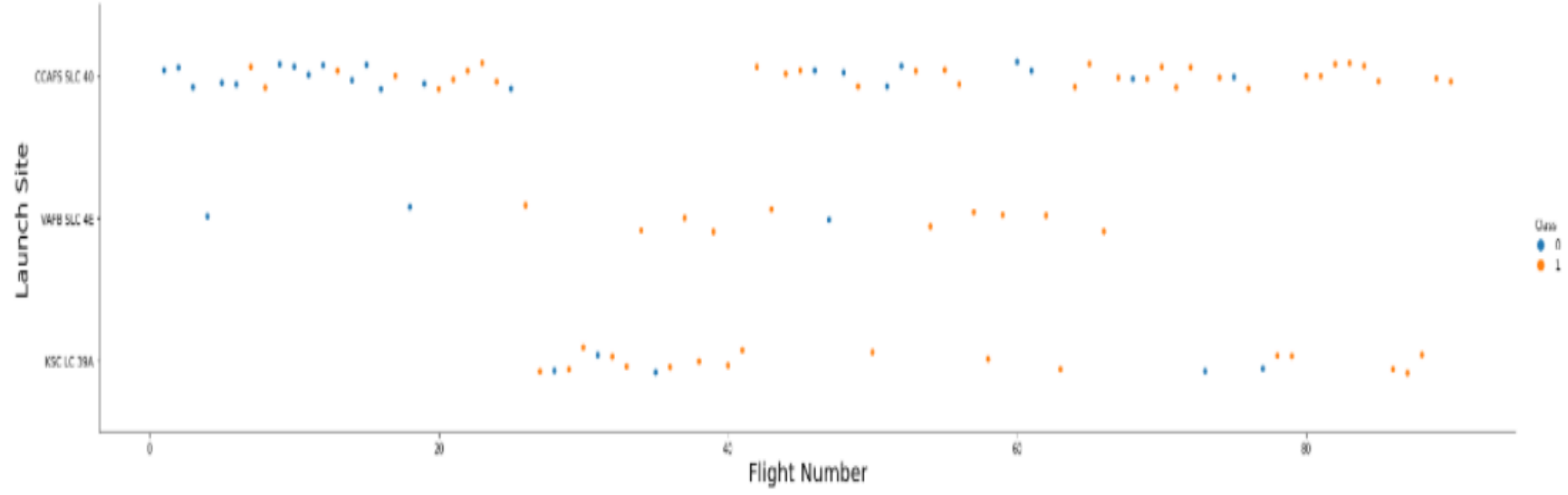
Section 2

Insights drawn from EDA



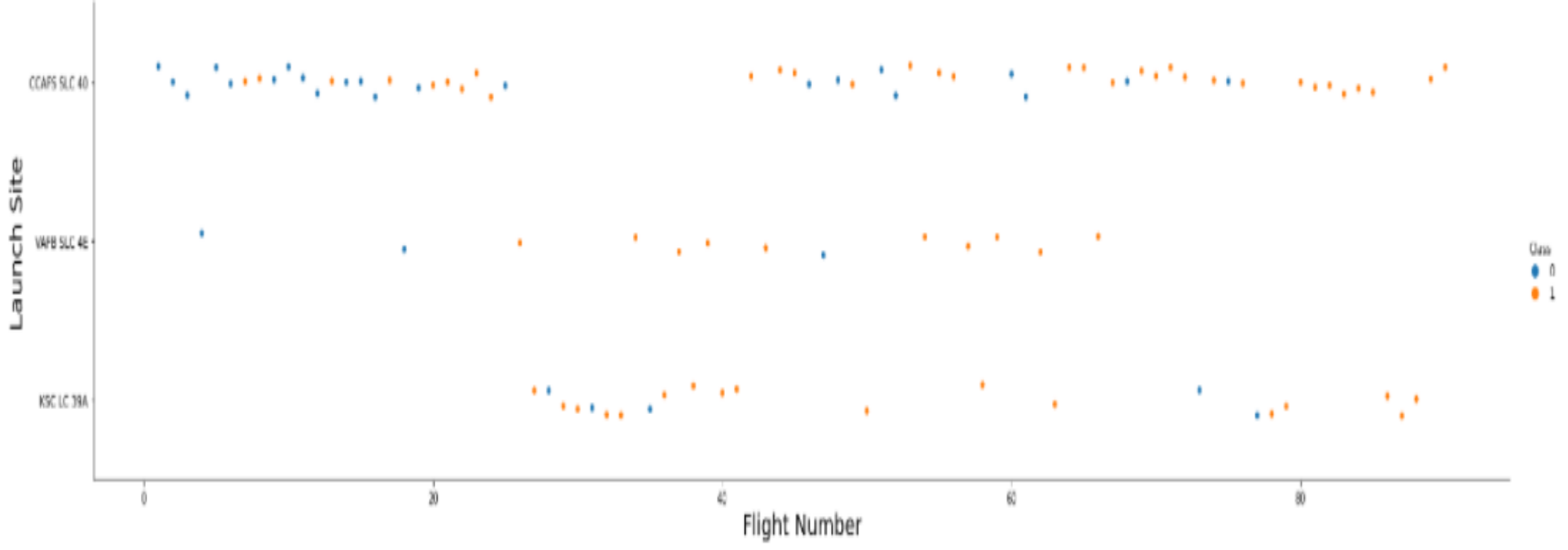
Flight Number vs. Launch Site

- The larger the launch site flight number, the greater are its chances of success.



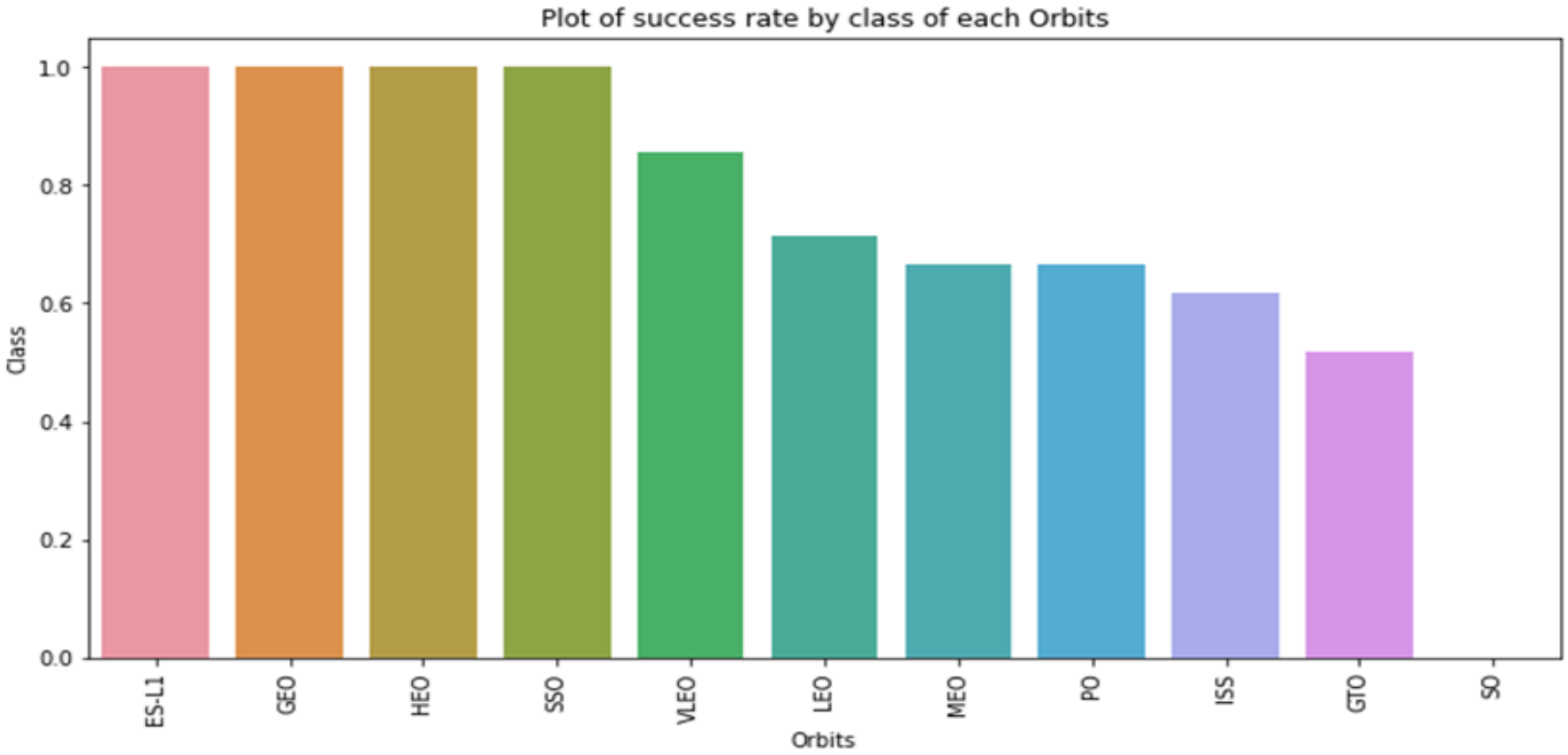
Payload vs. Launch Site

- The greater the Payload mass for launch site CCAFS SLC40, the higher the success rate for the rocket.



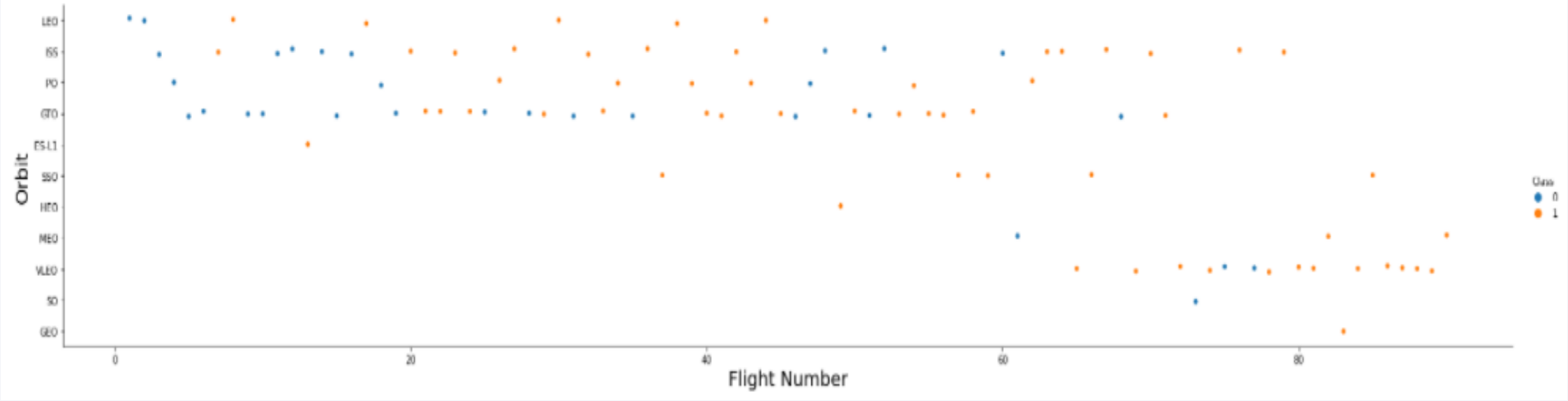
Success Rate vs. Orbit Type

- ES-L1, GEO, HEO, SSO, and VLEO have the most success rate.



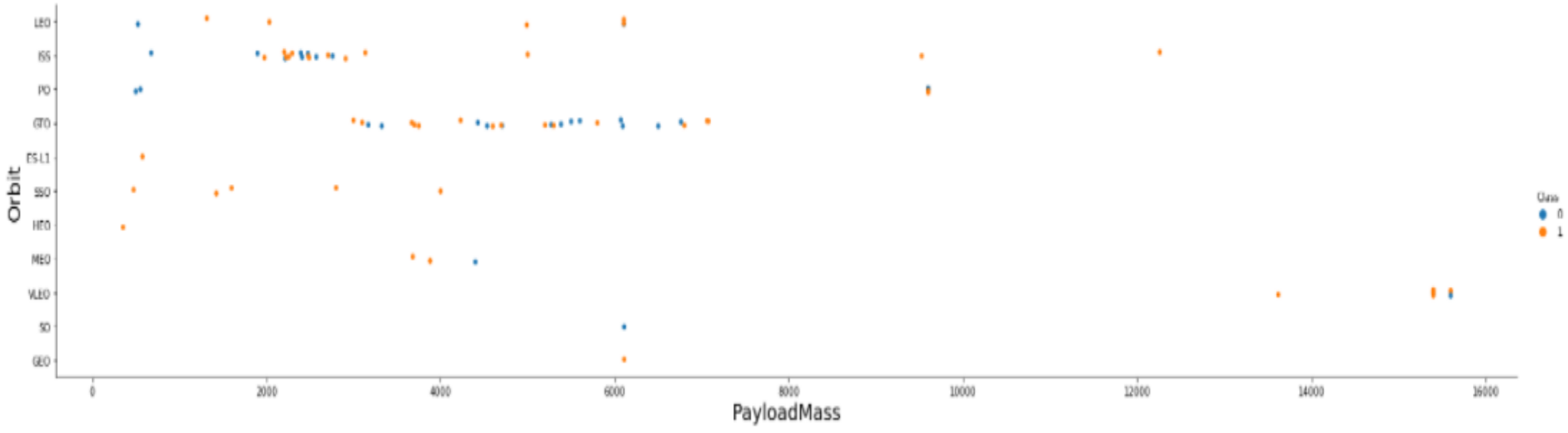
Flight Number vs. Orbit Type

- For LEO orbit, success is related to flight number.
- For GTO orbit, flight number and success are not as related.



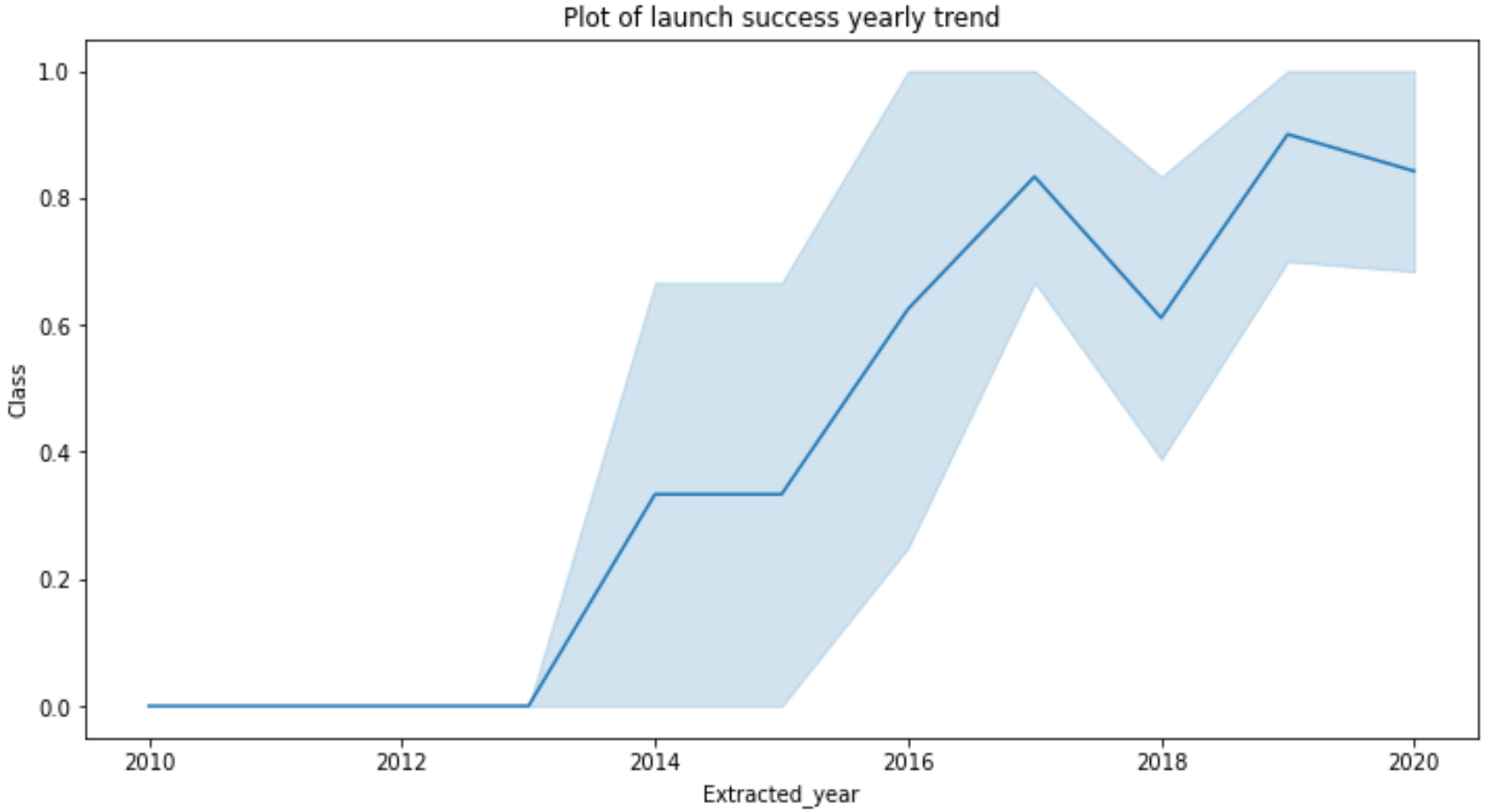
Payload vs. Orbit Type

- PO, LEO and ISS orbits have the heaviest payloads.



Launch Success Yearly Trend

- Success rate has been on the rise from 2013 until 2020.



All Launch Site Names

- Used the key word **DISTINCT** to display only unique launch sites.

```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
```

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

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

Launch Site Names Begin with 'CCA'

- Displayed 5 records.

Display 5 records where launch sites begin with the string 'CCA'

[14]:
```
%sql SELECT LAUNCH_SITE FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

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

[14]:

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Total Payload Mass

- None

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

```
[15]: %sql SELECT SUM (PAYLOAD_MASS__kg_) FROM SPACEXTBL WHERE CUSTOMER LIKE 'NASA(CRS)';
```

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

[15]:

SUM (PAYLOAD_MASS__kg_)
None

Average Payload Mass by F9 v1.1

- Average payload mass carried by booster version F9 v1.1 is 2928.4

Display average payload mass carried by booster version F9 v1.1

[16]:
```
%sql SELECT AVG (PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE BOOSTER_VERSION = 'F9 v1.1';
```

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

[16]:

AVG (PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

```
%sql SELECT MIN(Date) AS FirstSuccessfull_landing_date FROM SpaceXTBL WHERE Landing__Outcome LIKE 'Success (ground pad)';
```

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such column: LandingOutcome
[SQL: SELECT MIN(Date) AS FirstSuccessfull_landing_date FROM SpaceXTBL WHERE LandingOutcome LIKE 'Success (ground pad)';]
(Background on this error at: http://sqlalche.me/e/13/e3q8)
```

Successful Drone Ship Landing with Payload between 4000 and 6000

- Used the **WHERE** clause to filter for boosters which have successfully landed on drone ship and applied the **AND** condition to determine successful landing with payload mass greater than 4000 but less than 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
(sqlite3.OperationalError) no such column: LANDING__OUTCOME
[SQL: SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE LANDING__OUTCOME = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;]
(Background on this error at: http://sqlalche.me/e/13/e3q8)
```

Total Number of Successful and Failure Mission Outcomes

- Used wildcard like '%' to filter for **WHERE** Mission Outcome was a success or a failure.

List the total number of successful and failure mission outcomes

```
%sql SELECT COUNT MISSION_OUTCOME AS "successful mission "FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE 'Success%'
%sql SELECT COUNT MISSION_OUTCOME AS "failure mission " FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE 'Fail%'
%sql SELECT sum(case when MISSION_OUTCOME LIKE '%Success%' then 1 else 0 end) AS "Successful Mission", \
    sum(case when MISSION_OUTCOME LIKE '%Failure%' then 1 else 0 end) AS "Failure Mission" \
FROM SPACEXTBL;
```

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) near "AS": syntax error
[SQL: SELECT COUNT MISSION_OUTCOME AS "succesful mission "FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE 'Success%']
(Background on this error at: http://sqlalche.me/e/13/e3q8)
 * sqlite:///my_data1.db
(sqlite3.OperationalError) near "AS": syntax error
[SQL: SELECT COUNT MISSION_OUTCOME AS "failure mission " FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE 'Fail%']
(Background on this error at: http://sqlalche.me/e/13/e3q8)
 * sqlite:///my_data1.db
Done.
```

Successful Mission	Failure Mission
100	1

Boosters Carried Maximum Payload

- Determined the booster that carried maximum payload using a subquery in the **WHERE** clause and the **MAX()** function.

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
%sql SELECT DISTINCT BOOSTER_VERSION AS "Booster Versions which carried the Maximum Payload Mass" FROM SPACEXTBL \
WHERE PAYLOAD_MASS__KG_ =(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

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

Booster Versions which carried the Maximum Payload Mass
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

- Used a combination of the **WHERE** clause, **LIKE**, **AND**, and **BETWEEN** to filter for failed landing outcomes in drone ship, their respective booster versions, and names of launch sites for year 2015

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

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such column: LANDING__OUTCOME
[SQL: SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE DATE LIKE '2015-%' AND  LANDING__OUTCOME = 'Failure (drone ship)';]
(Background on this error at: http://sqlalche.me/e/13/e3q8)
```

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

- Selected landing outcomes and **COUNT** of landing outcomes and used **WHERE** clause to filter for landing outcomes **BETWEEN** 2010-06-04 to 2010-03-20.
- We applied the **GROUP BY** clause to group the landing outcomes and then **ORDER BY** clause for grouping by descending order.

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

```
%sql SELECT LANDING__OUTCOME as "Landing Outcome", COUNT(LANDING__OUTCOME) AS "Total Count" FROM SPACEXTBL \
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
GROUP BY  LANDING__OUTCOME \
ORDER BY COUNT(LANDING__OUTCOME) DESC ;
```

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such column: LANDING__OUTCOME
[SQL: SELECT LANDING__OUTCOME as "Landing Outcome", COUNT(LANDING__OUTCOME) AS "Total Count" FROM SPACEXTBL  WHERE DATE BETWEEN '2010-06-04' AND
'2017-03-20'  GROUP BY  LANDING__OUTCOME  ORDER BY COUNT(LANDING__OUTCOME) DESC ;]
(Background on this error at: http://sqlalche.me/e/13/e3q8)
```

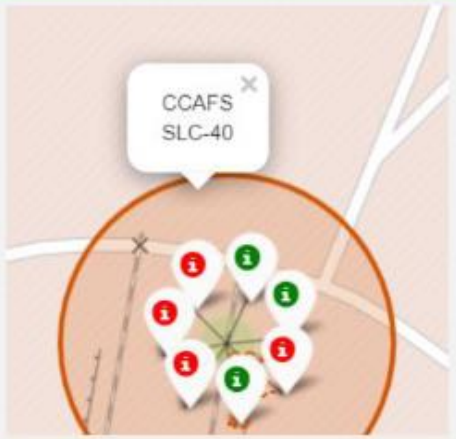
Section 3

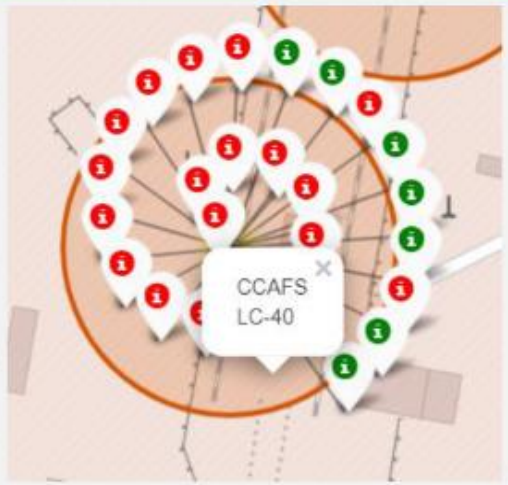
Launch Sites Proximities Analysis

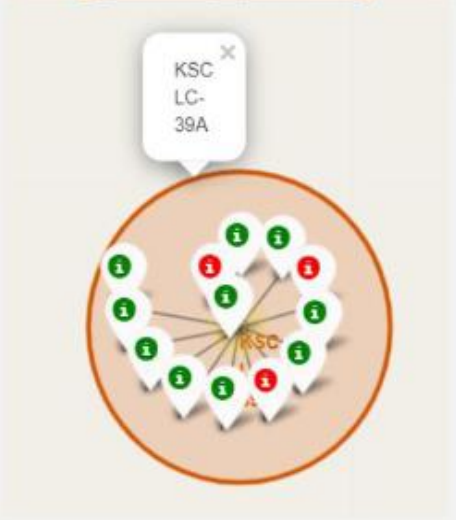
All launch sites global map markers

We can see that the SpaceX launch sites are in the United States of America coasts. Florida and California

Markers showing launch sites with color labels







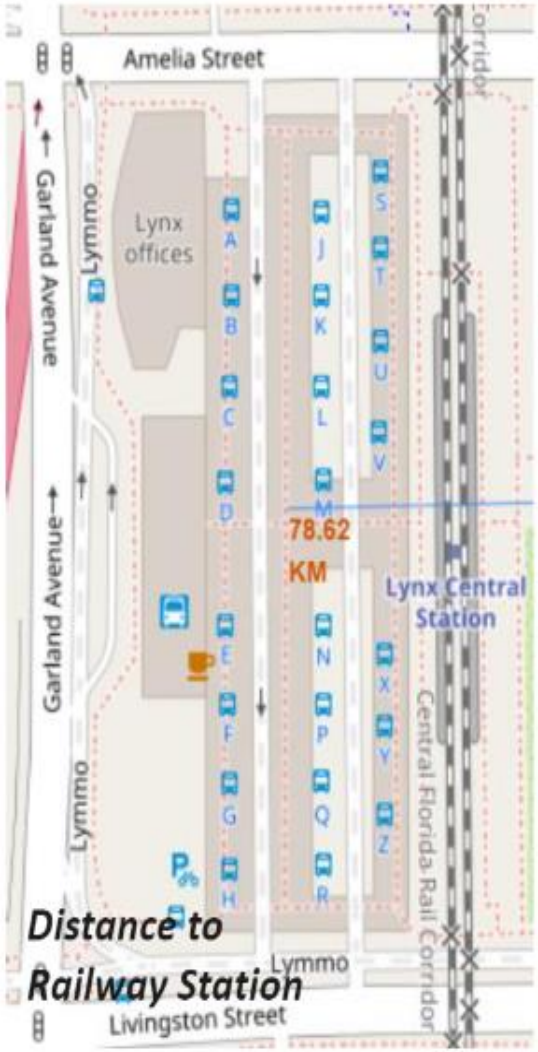
Florida Launch Sites

Green Marker shows successful Launches and Red Marker shows Failures

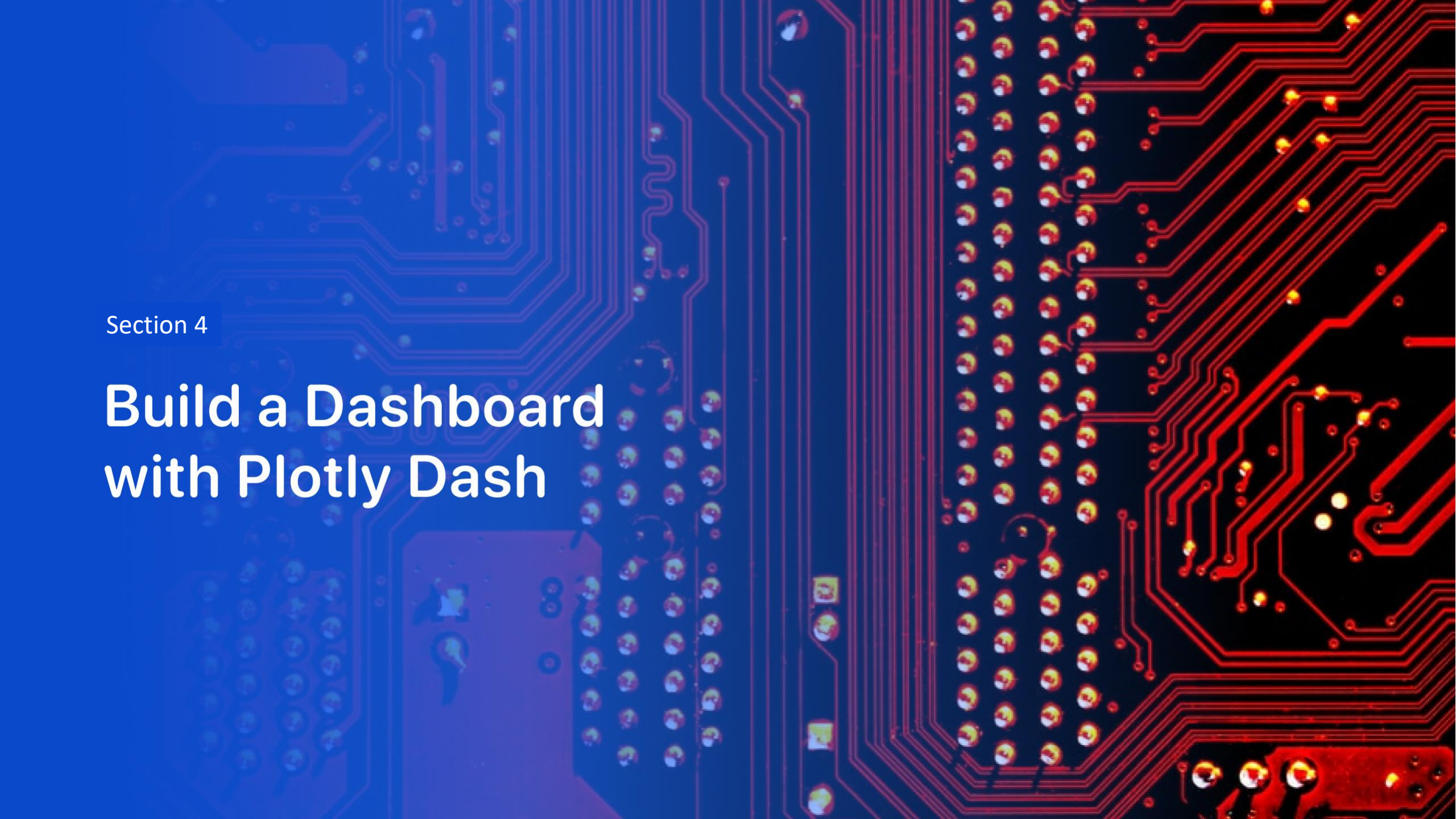
California Launch Site

37

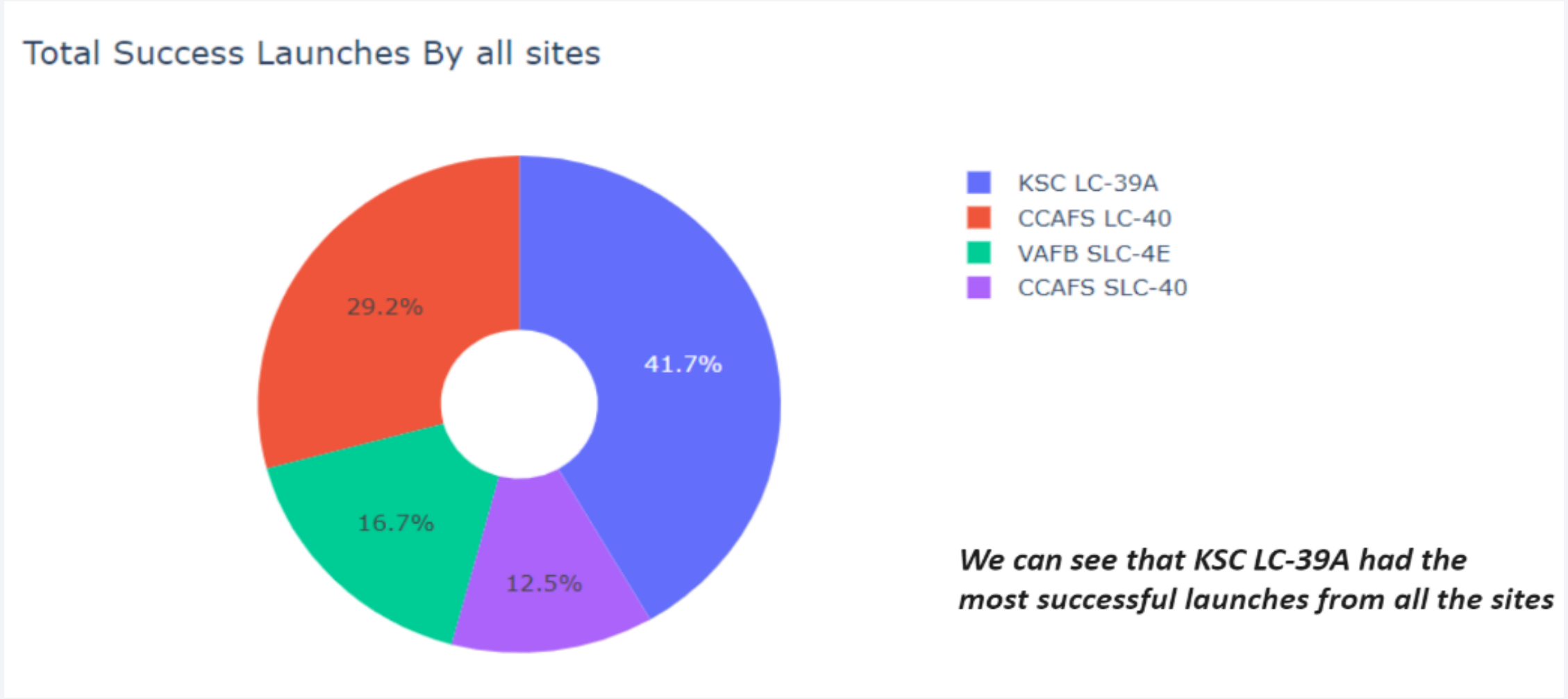
Launch Site distance to landmarks



- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

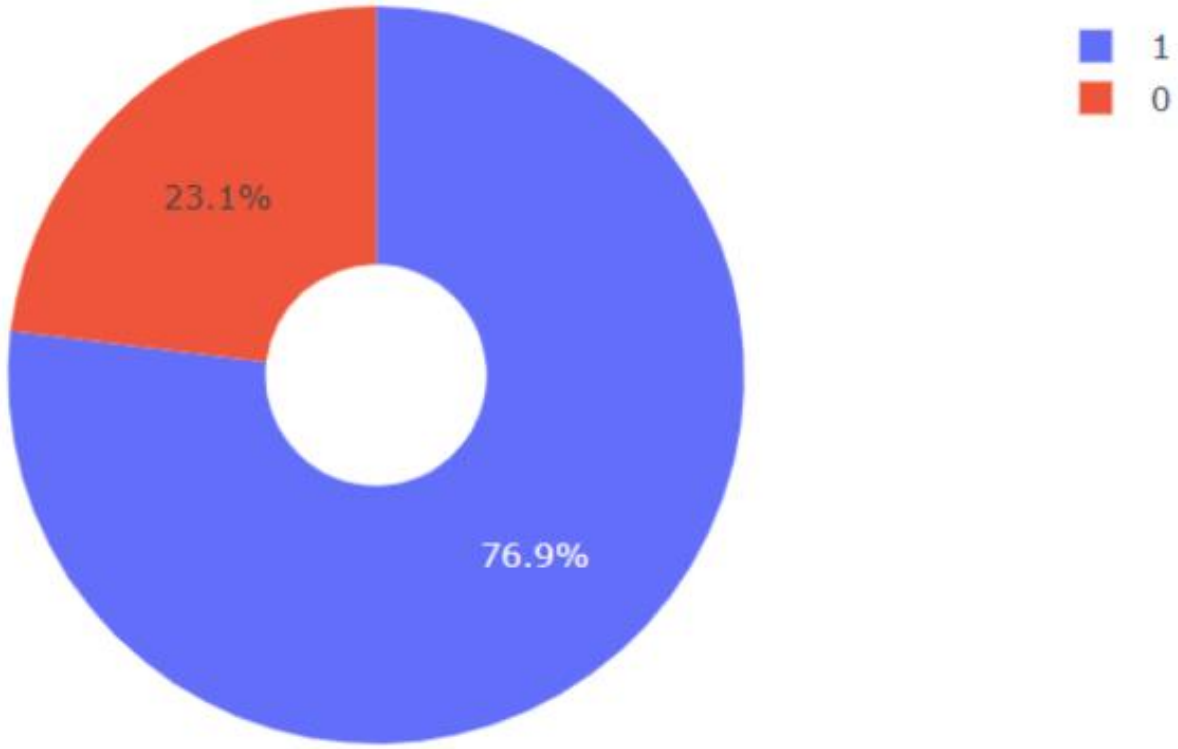

Section 4
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



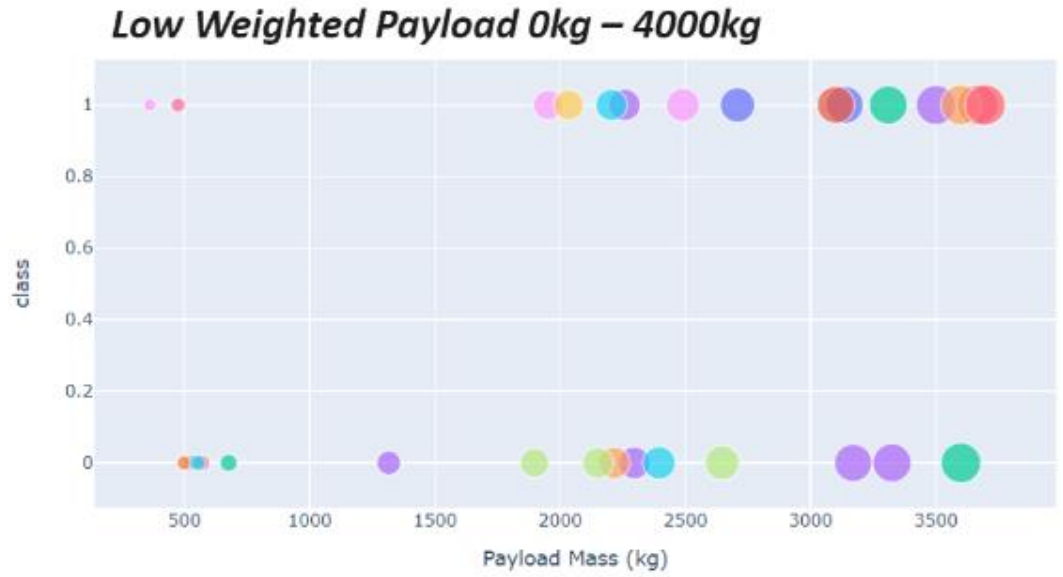
We can see that KSC LC-39A had the most successful launches from all the sites

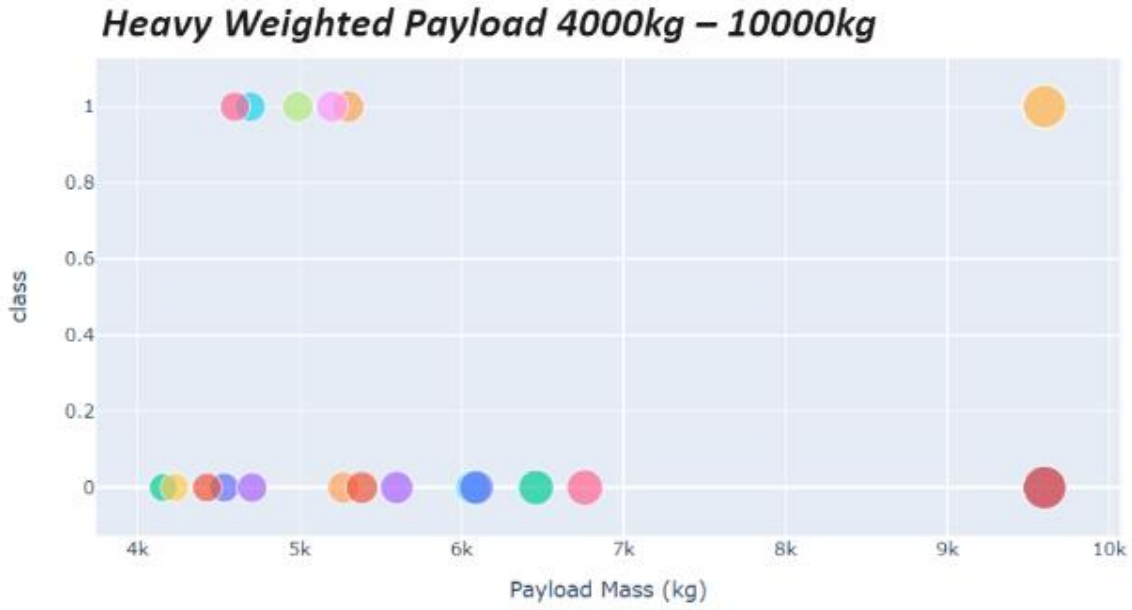
Pie chart showing the Launch site with the highest launch success ratio



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

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





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

Section 5

Predictive Analysis (Classification)

Classification Accuracy

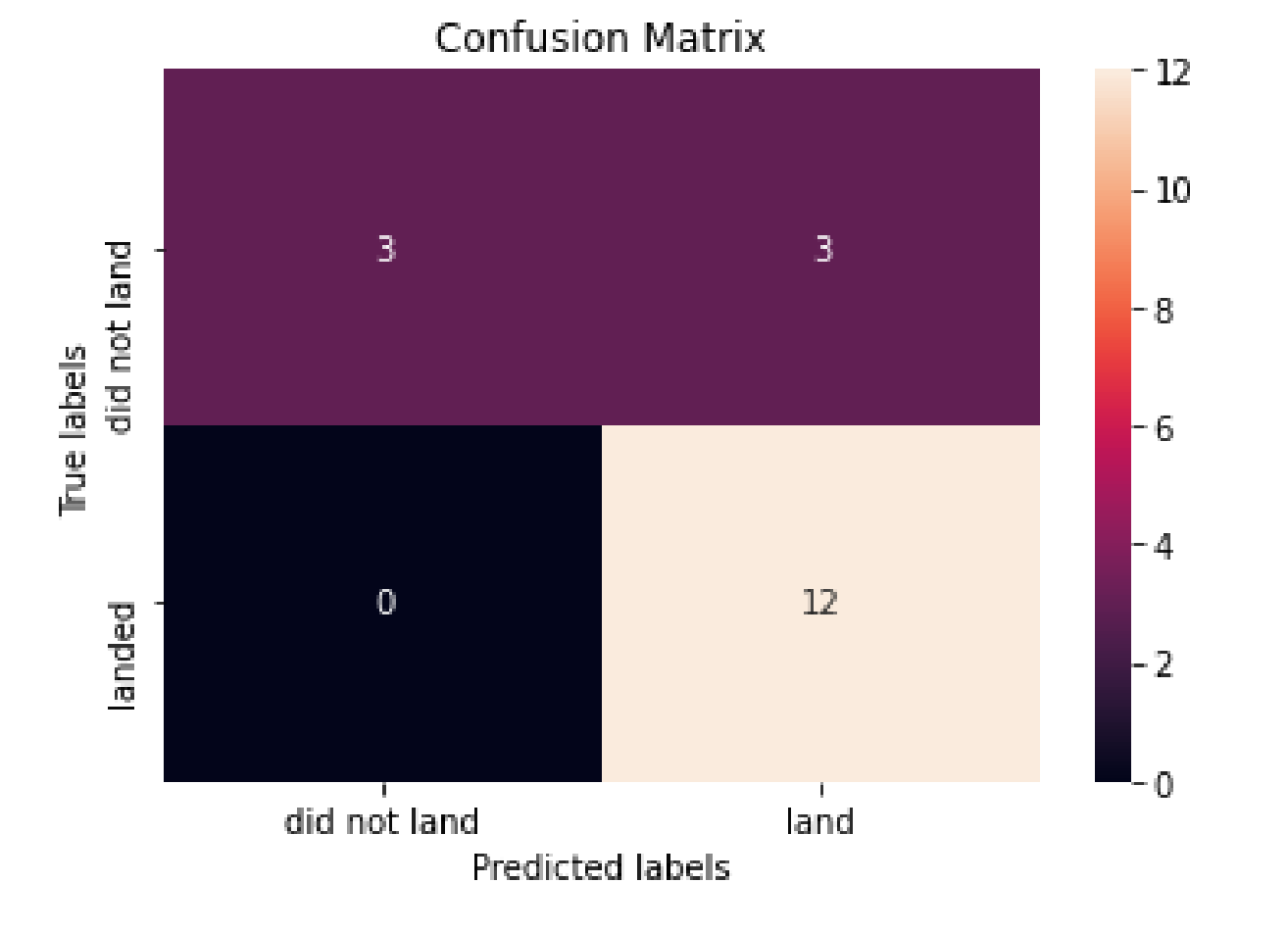
- Decision tree classifier has the highest accuracy

```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

Confusion Matrix



- Need to take note of 3 false positives (failed landing denoted by classifier as successful).
- Hence, classifier can be further improved prior to deployment.

Conclusions

- Decision Tree Classifier is the best ML algorithm for this project
- Success rate for launching was on the rise from 2013 until 2020.
- The greater the flight number at a launch site, the greater its success rate becomes at that particular site
- Orbits HEO, VLEO, ES-L1, GEO, SSO, had the most success rate.
- KSC LC-39A had the most successful launches of all sites.

Thank you!