IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Name>
<Date>

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
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result

Introduction

- Project background and context
- Space X is promoting its new reusable Falcon 9 rocket. Being able to reuse the rocket significantly reduces the cost for launch from around 165 million dollars to 62 million dollars. However, the benefits of Falcon 9 can only be attained if the 'first stage' is able to successfully land. Therefore, if we can determine whether the first stage will be successful, we can then determine the cost for launch. This information can then be used by other companies to bid against Space X for a rocket launch. The goal of this project to to create an algorithm that will accurately predict whether the first stage will successfully land.

Problems you want to find answers

1. What factors determine whether the landing will be successful

2. Identify and understand the interaction between various features that determine the success rate of a successful landing

3. What do the operating conditions need to be to make sure the landing is successful.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using the SpaceX API and combined with web scraping done on Wikipedia
- Perform data wrangling
  - One-hot encoding was used on the 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
  - How to build, tune, evaluate classification models

Data Collection

- Describe how data sets were collected.
- Data was collected using a get request to the Space X API
- The response was decoded using .json() function call and turned into a panadas datafram using .json_normalize()
- The data was cleaned, and missing values were identified and filled if necessary
- The Wikipedia page for Falcon 9 launch records was web scarped using BeautifulSoup.
- This process meant that future analysis could be done easily as it was in a clean and simple format.

Data Collection – SpaceX API

The get request was used to collect data from the SpaceX API. The data given was cleaned and formatted so that it was easy to analyze.

- https://github.com/am-cuganesan/testrepo1/blob/main/Data%20Collection%20API%20Lab.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 - Scraping

- The Falcon 9 Wikipedia page was web scarped using BeautifulSoup. The table was turned into a pandas dataframe.
- https://github.com/am-cuganesan/testrepo1/blob/main/Data%20Collection%20with%20Web%20Scraping.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
```

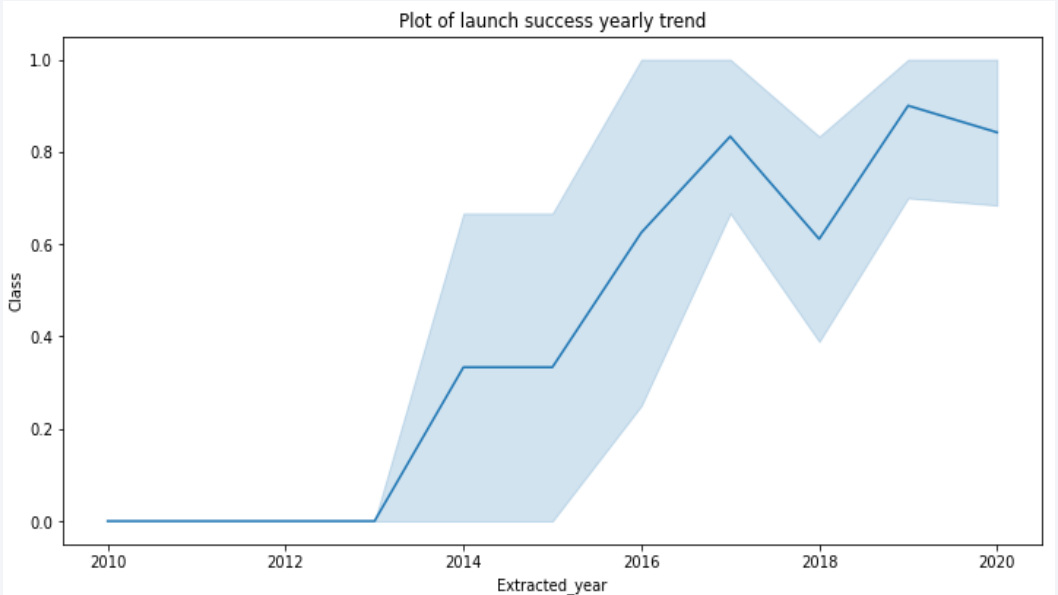
4. Create a dataframe by parsing the launch HTML tables
5. Export data to csv

Data Wrangling

- Describe how data were processed
- Firstly, the data was explored, and the training labels were determined.
- The numbed of launches at each site was calculated along with the number and occurrence of each orbit.
- A label for the landing outcome was created using the outcome column and results were exported to a csv.
- https://github.com/am-cuganesan/testrepo1/blob/main/EDA.ipynb

EDA with Data Visualization

- The several key relationships were visualized to to better understand what impacted the success rate. This included the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend.
- https://github.com/am-cuganesan/testrepo1/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_2_jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb



EDA with SQL

- The SpaceX dataset was loaded into a PostgreSQL database.
- EDA with SQL was applied to the data to aid understanding and insight. These were the questions we wanted to answer:
  - The names of the launch sites
  - The total payload mass carried by the Nasa boosters
  - The average payload mass carried by F9 v1.1 booster
  - The total number of successful and unsuccessful missions
  - The failed landing outcomes on drone ships, and their booster's version and launch site names
- https://github.com/am-cuganesan/testrepo1/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_2_jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

Build an Interactive Map with Folium

- An interactive, folium map was created to track the launch sites. Key objects were added too the map including markers, circles and lines to clearly mark either a successful or failed launch at each site.
- The launch outcomes were assigned as either class 0 for failed or 1 for successful.
- Color-labeled marker clusters were used to clearly identify which launch sites had the highest success rates.
- The distance between each launch site to its key sites was calculated and displayed on the map using the lines. The lines showed how faraway the launch site was from railways, highways and coastlines in order to see whether launch sites are required to keep a certain distance from cities.

https://github.com/am-cuganesan/testrepo1/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_3_lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

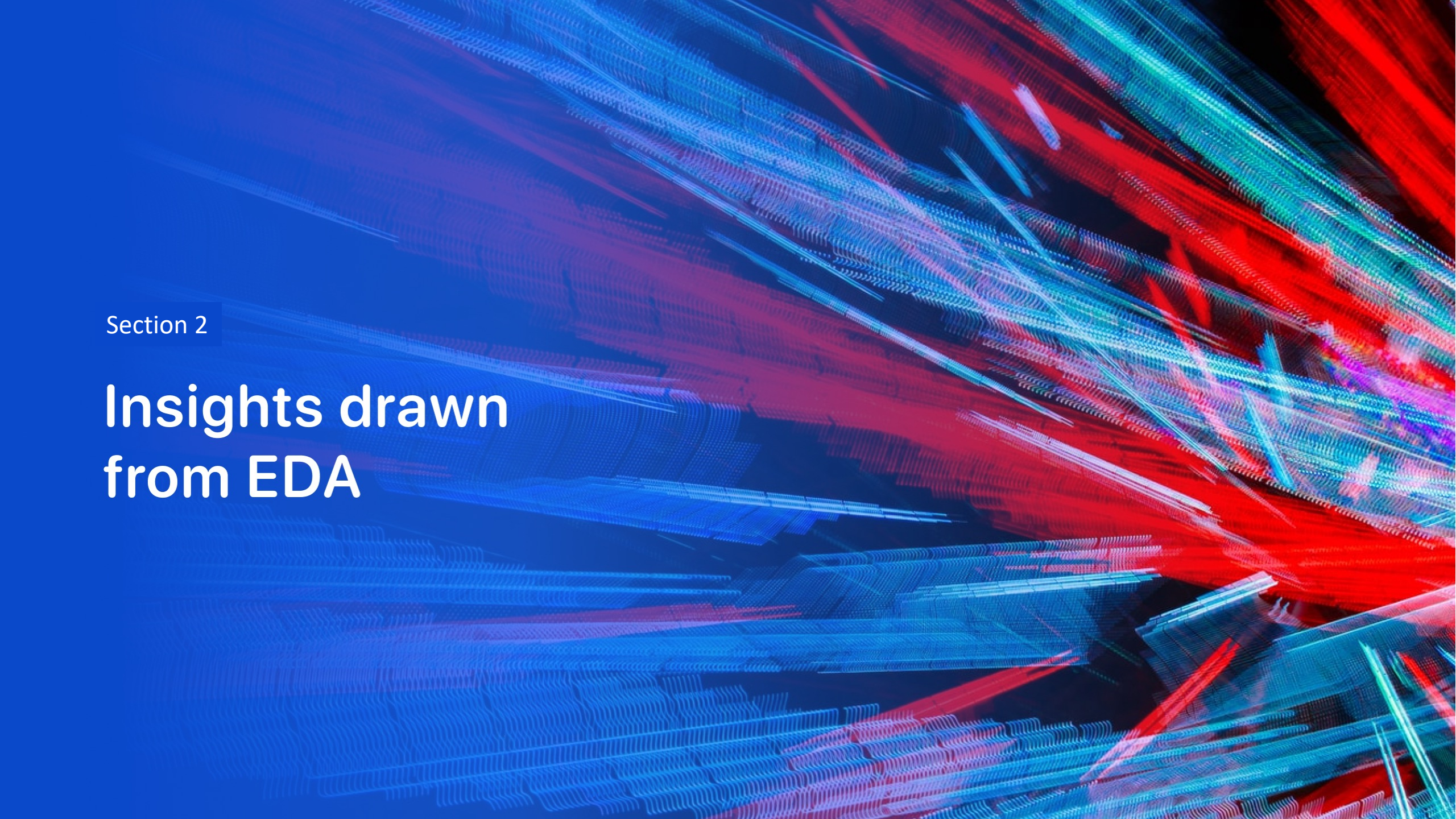
- An interactive dashboard was built using Plotly dash.
- Pie charts were included in the dashboard to visual show the total launches in certain areas.
- Scatter graphs were incorporated d into the dashboard to show the relationship between the ‘Outcome’ and Payload Mass for difference booster versions.
- https://github.com/am-cuganesan/testrepo1/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- The data was loaded using numpy and padas. The data was then transformed and split into training and testing groups.
- Difference machine learning models were built and using GridSearchCV different hyperparameters were tuned.
- Accuracy was used aa the metric for the model. The model was improved using feature engineering and algorithm tuning.
- At the end we were left with the best performing classification model/
- https://github.com/am-cuganesan/testrepo1/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

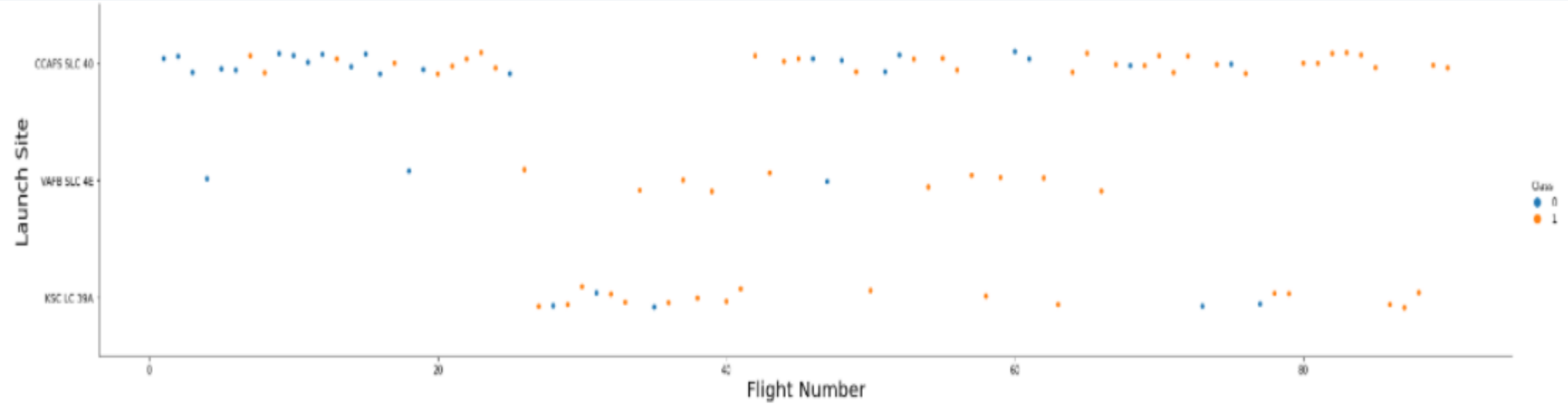
Results

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


Section 2
Insights drawn from EDA

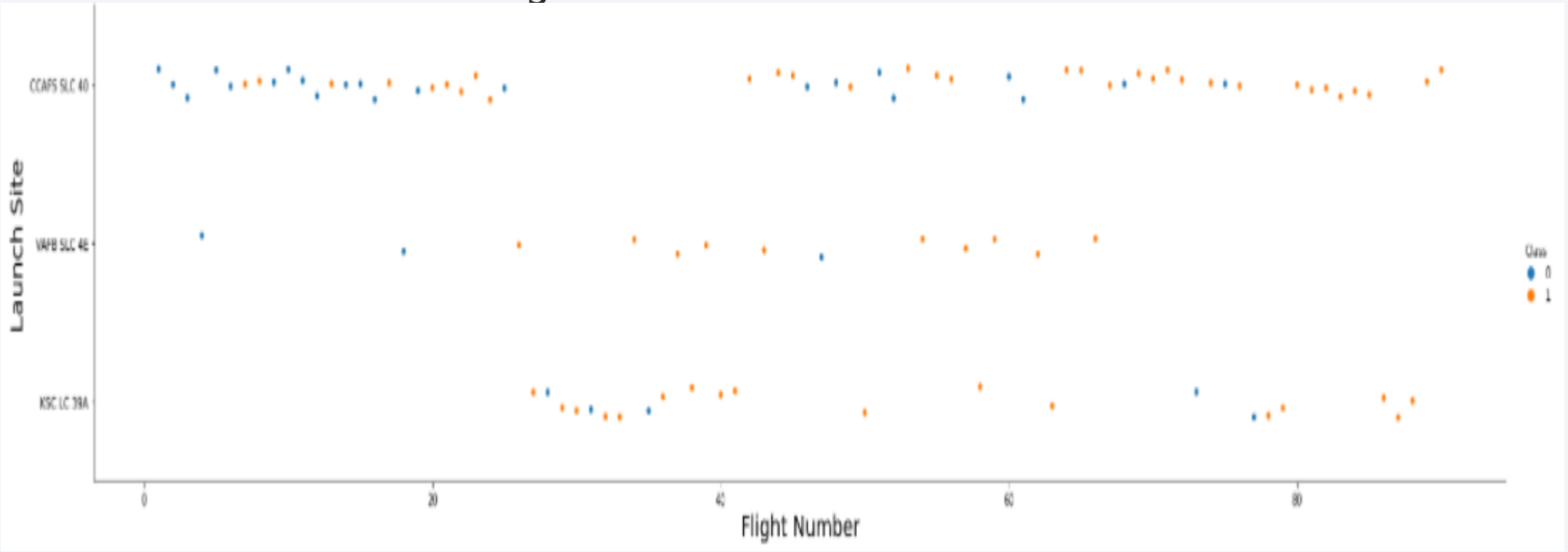
Flight Number vs. Launch Site

- Using the plot, we can see that the larger the number of flights at a launch site, the greater the overall success rate at the site.



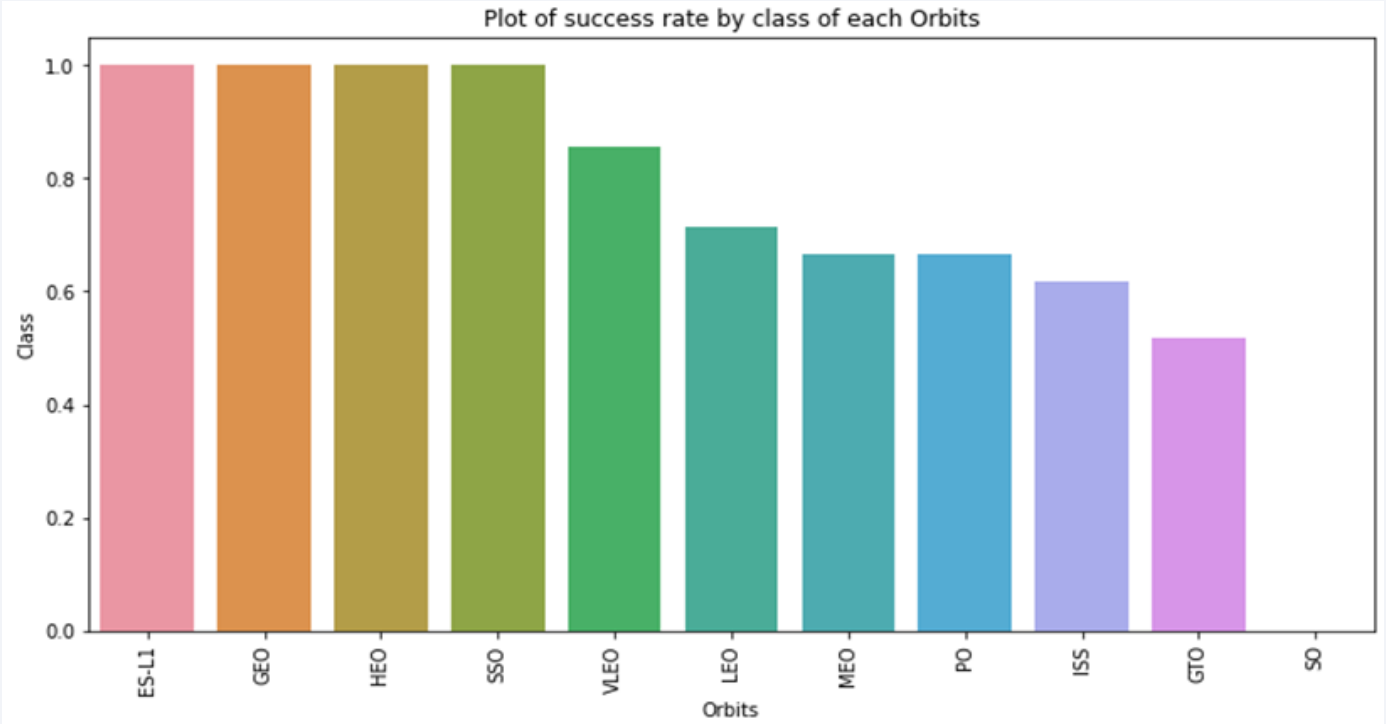
Payload vs. Launch Site

- From the scatter plot it can be seen that the greater the payload mass for the launch site CCAFS SLC40 the higher the success rate for the rocket.



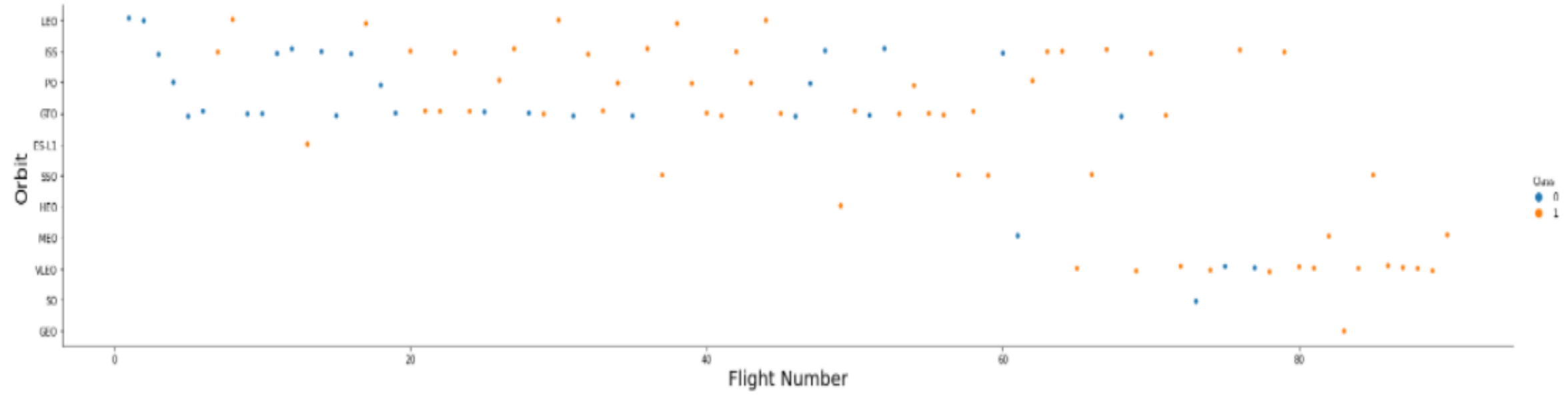
Success Rate vs. Orbit Type

- From the bar plot orbit type ES-L, GEO, HEO SSO and VLEO have the highest success rates.



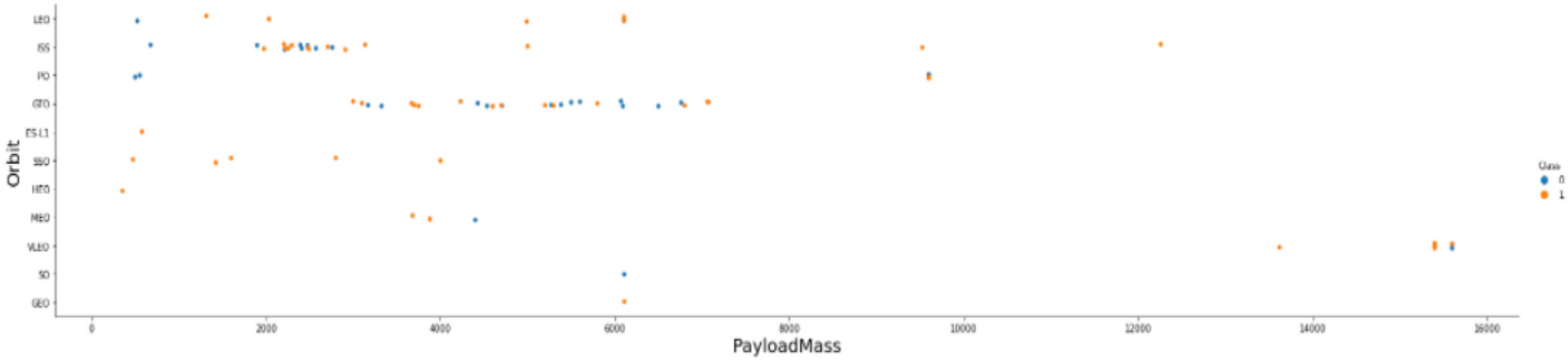
Flight Number vs. Orbit Type

- From the scatter plot we scan see that for LEO] orbits the there is a relationship between the success and number of flights performed. However, for GTO orbits there is no clear relationship between flight and success..



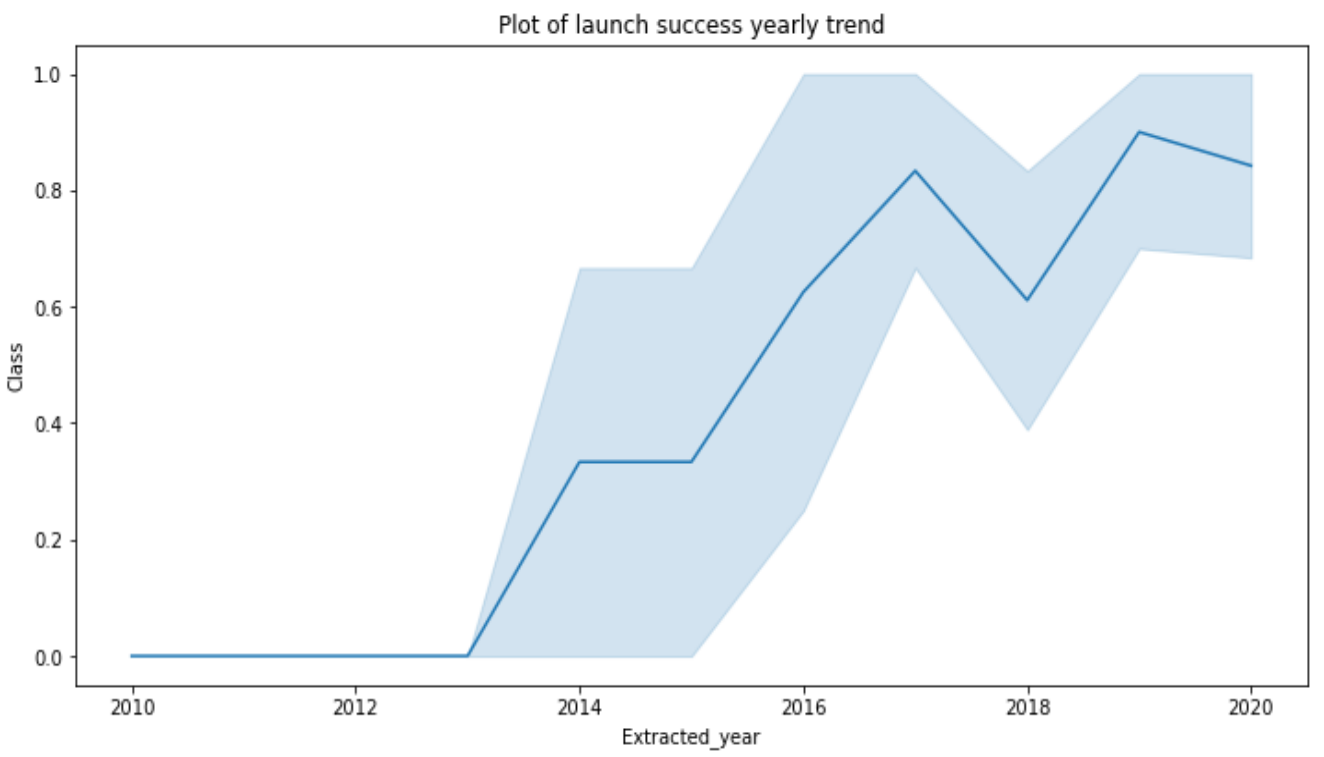
Payload vs. Orbit Type

- From the scatter plot it is shown that for heavier payloads successful landing occur in PO, LEO and ISS type orbits.



Launch Success Yearly Trend

- The chart shows that since 2013 the launch success has seen on average a continuous increase,. However, between 2017 to 2018 and2019 to maybe 2021 there was been small declines.



All Launch Site Names

- The DISTINCT keywords was used to only show the unique launch sites, taken from the SpaceX data.

Display the names of the unique launch sites in the space mission

In [10]:
```
task_1 = '''
        SELECT DISTINCT LaunchSite
        FROM SpaceX
'''
create_pandas_df(task_1, database=conn)
```

Out[10]:

	launchsite
0	KSC LC-39A
1	CCAFS LC-40
2	CCAFS SLC-40
3	VAFB SLC-4E

Launch Site Names Begin with 'CCA'

- The following query was used to extract the 5 records for launch sites that began with ‘CCA’.

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

In [11]:
```
task_2 = '''
        SELECT *
        FROM SpaceX
        WHERE LaunchSite LIKE 'CCA%'
        LIMIT 5
        '''
create_pandas_df(task_2, database=conn)
```

Out[11]:

	date	time	boosterversion	launchsite	payload	payloadmasskg	orbit	customer	missionoutcome	landingoutcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	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
3	2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- The total payload for the NASA boosters was calculated to be 45596.

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

In [12]:
```
task_3 = '''
        SELECT SUM(PayloadMassKG) AS Total_PayloadMass
        FROM SpaceX
        WHERE Customer LIKE 'NASA (CRS)'
        '''
create_pandas_df(task_3, database=conn)
```

Out[12]:

	total_payloadmass
0	45596

Average Payload Mass by F9 v1.1

- The average payload mass for the F9 v1.1 version boosters was 2928.4

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

In [13]:
```
task_4 = '''
        SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
        FROM SpaceX
        WHERE BoosterVersion = 'F9 v1.1'
        '''
create_pandas_df(task_4, database=conn)
```

Out[13]:

	avg_payloadmass
0	2928.4

First Successful Ground Landing Date

- The first successful landing on the ground pad occurred on the 22nd of December 2015.

In [14]:
```
task_5 = '''
        SELECT MIN(Date) AS FirstSuccessfull_landing_date
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Success (ground pad)'
        '''
create_pandas_df(task_5, database=conn)
```

Out[14]:

	firstsuccessfull_landing_date
0	2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- Using the WHERE clause to filter for boosters which successfully landed on the drone ship and using the AND condition to determine whether the payload mass was greater than 400 but less than 6000 resulted in the following list.

In [15]:
```
task_6 = '''
        SELECT BoosterVersion
        FROM SpaceX
        WHERE LandingOutcome = 'Success (drone ship)'
            AND PayloadMassKG > 4000
            AND PayloadMassKG < 6000
        '''
create_pandas_df(task_6, database=conn)
```

Out[15]:

	boosterversion
0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

- The wildcard % was used to filter for WHERE Mission outcome was a success or failure.

List the total number of successful and failure mission outcomes

In [16]:
```
task_7a = '''
        SELECT COUNT(MissionOutcome) AS SuccessOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Success%'
        '''

task_7b = '''
        SELECT COUNT(MissionOutcome) AS FailureOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Failure%'
        '''
print('The total number of successful mission outcome is:')
display(create_pandas_df(task_7a, database=conn))
print()
print('The total number of failed mission outcome is:')
create_pandas_df(task_7b, database=conn)
```

Out[16]:

The total number of successful mission outcome is:

	successoutcome
0	100

The total number of failed mission outcome is:

	failureoutcome
0	1

Boosters Carried Maximum Payload

- The maximum payload the boosters carried was found using a subquery in the WHERE close and the MAX() function

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

In [17]:
```
task_8 = '''
        SELECT BoosterVersion, PayloadMassKG
        FROM SpaceX
        WHERE PayloadMassKG = (
                                SELECT MAX(PayloadMassKG)
                                FROM SpaceX
                                )
        ORDER BY BoosterVersion
        '''
create_pandas_df(task_8, database=conn)
```

Out[17]:

	boosterversion	payloadmasskg
0	F9 B5 B1048.4	15600
1	F9 B5 B1048.5	15600
2	F9 B5 B1049.4	15600
3	F9 B5 B1049.5	15600
4	F9 B5 B1049.7	15600
5	F9 B5 B1051.3	15600
6	F9 B5 B1051.4	15600
7	F9 B5 B1051.6	15600
8	F9 B5 B1056.4	15600
9	F9 B5 B1058.3	15600
10	F9 B5 B1060.2	15600
11	F9 B5 B1060.3	15600

2015 Launch Records

- Using a combination of the WHERE clause, LIKE, AND and BETWEEN conditions., the filter were able to find the failed landing outcomes in the drone ship, their booster versions, and launch site names for 2015.

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

In [18]:
```
task_9 = '''
        SELECT BoosterVersion, LaunchSite, LandingOutcome
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Failure (drone ship)'
            AND Date BETWEEN '2015-01-01' AND '2015-12-31'
        '''
create_pandas_df(task_9, database=conn)
```

Out[18]:

	boosterversion	launchsite	landingoutcome
0	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
1	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

- Using the WHERE clause to filter for landing outcomes BETWEEN the two dates the necessary landing outcomes were found. The GROUP BY clause was applied to the landing outcomes and the ORDER BY clauses was used to order the results in descending order.

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad))

In [19]:
```
task_10 = '''
        SELECT LandingOutcome, COUNT(LandingOutcome)
        FROM SpaceX
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
        GROUP BY LandingOutcome
        ORDER BY COUNT(LandingOutcome) DESC
        '''
create_pandas_df(task_10, database=conn)
```

Out[19]:

	landingoutcome	count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1
7	Failure (parachute)	1

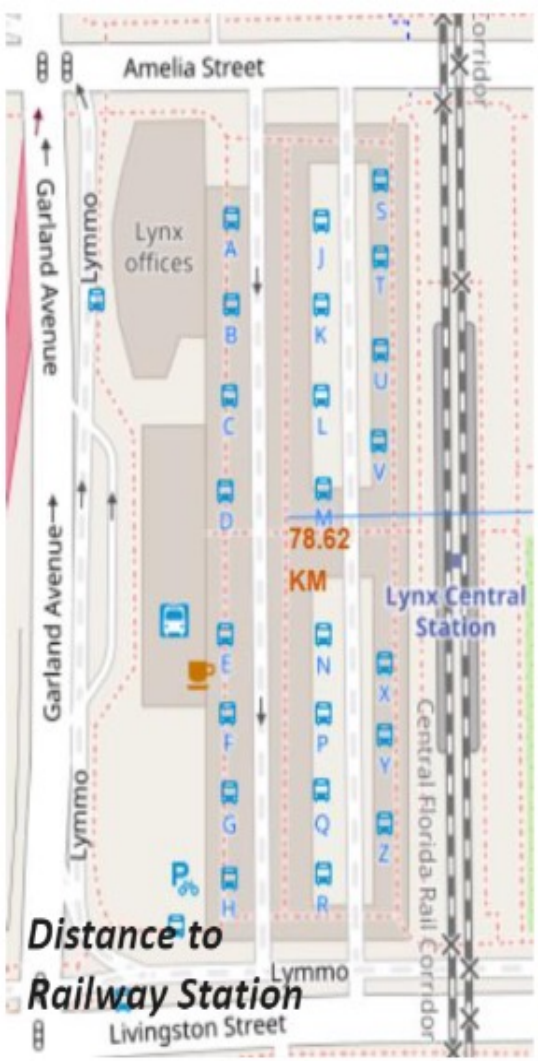

Section 3
Launch Sites
Proximities Analysis

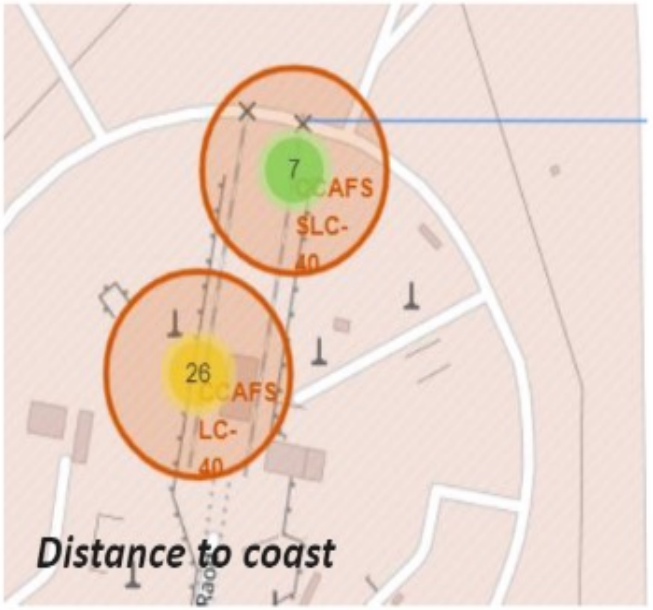
Launch Sites – Global Perspective

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

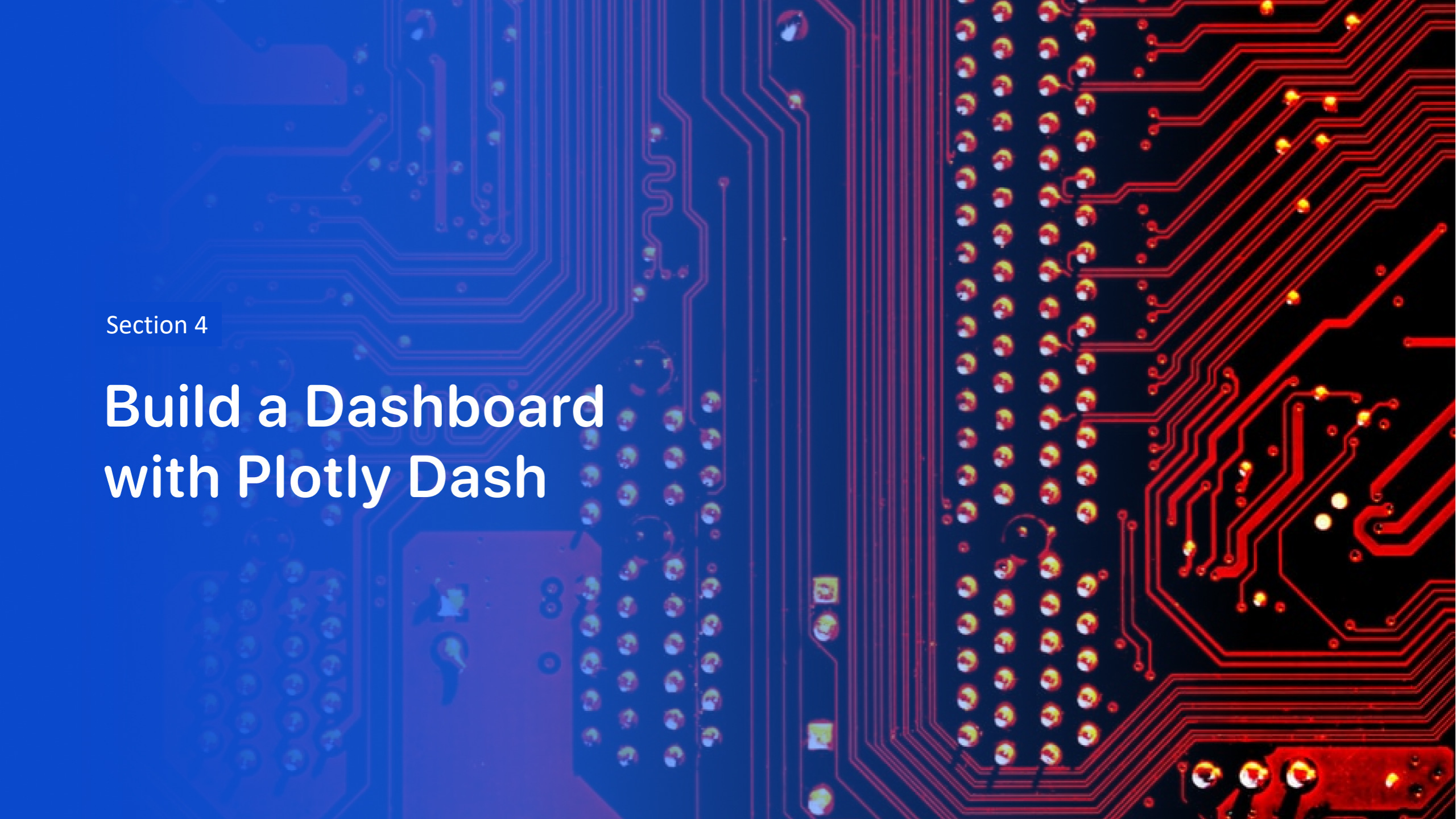
Launch Sites – Representation Using Color labelling

Launch Site – Distance from Key Landmarks

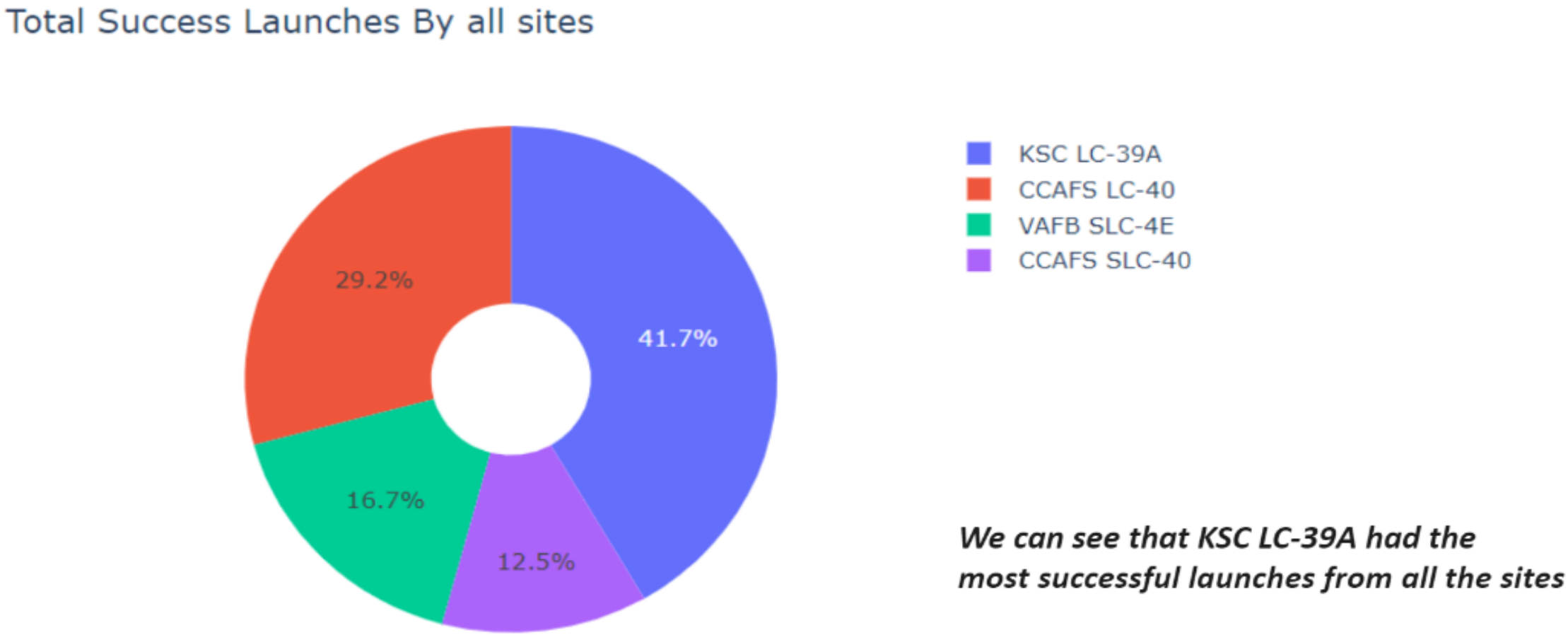




- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

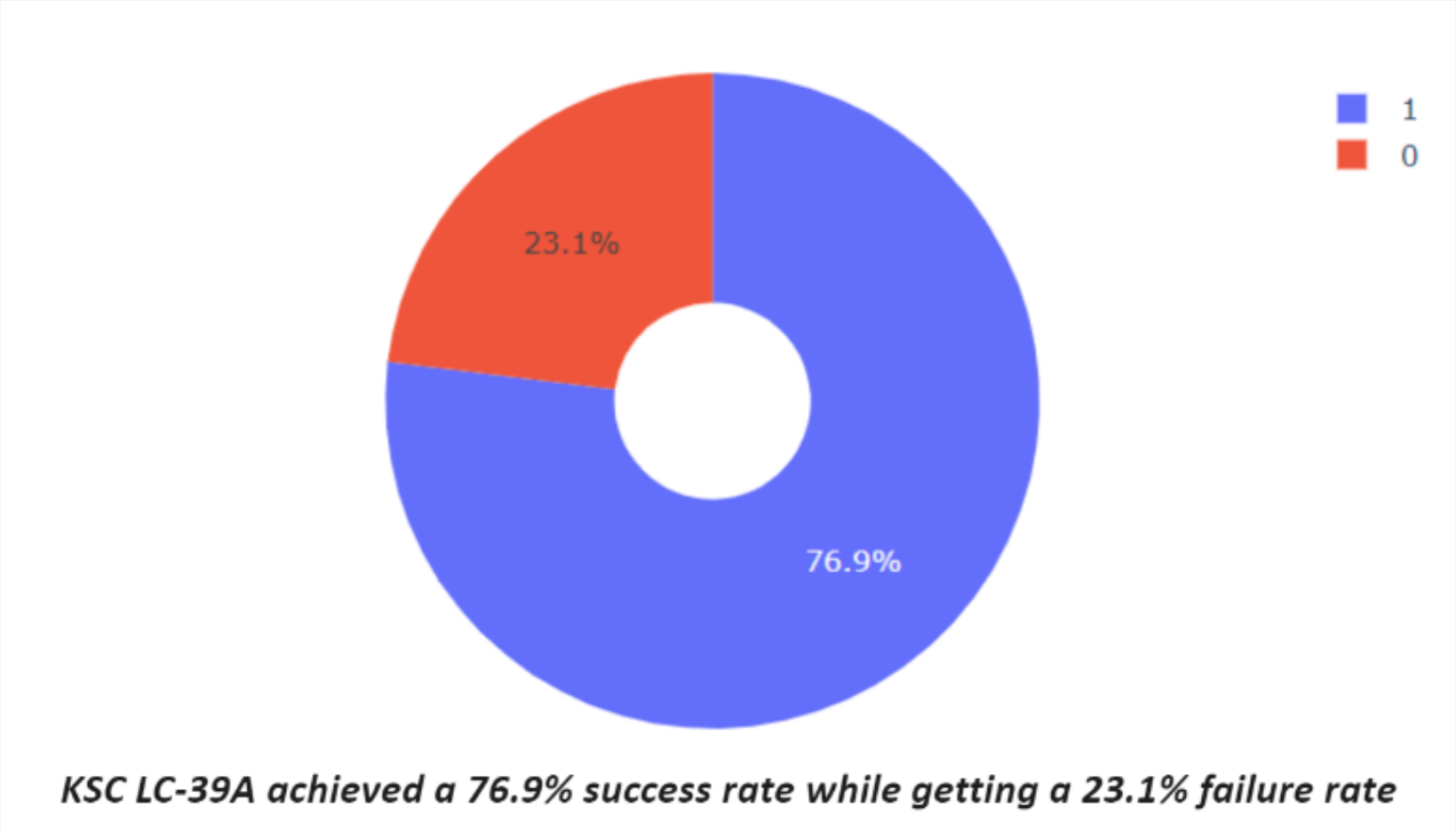

Section 4
Build a Dashboard with Plotly Dash

Pie Chart – Total Success of Each Launch Site



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

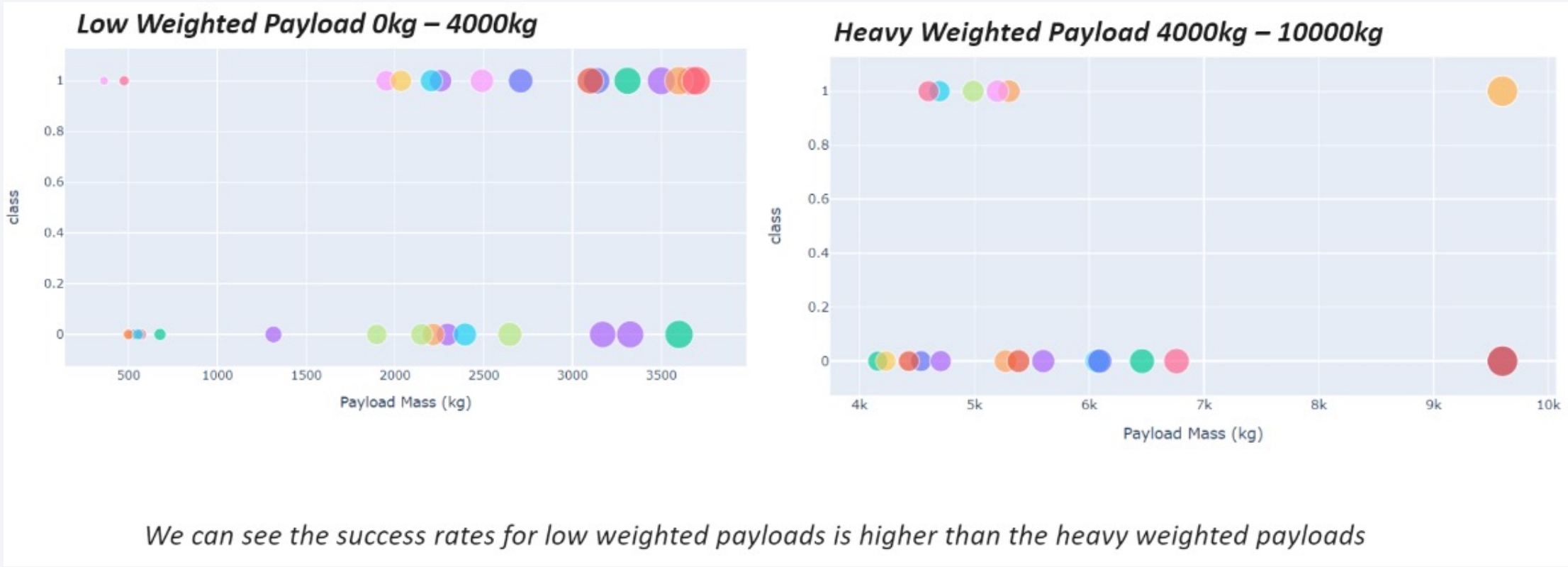
Pie Chart – Launch Site with Highest Success Ratio



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

Scatter Plot – Payload vs Launch Outcome

The scatter plot shows the relationship between payload and launch outcomes for all sites using different payloads 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

- The decision tree classifier was overall the model with the highest classification accuracy.

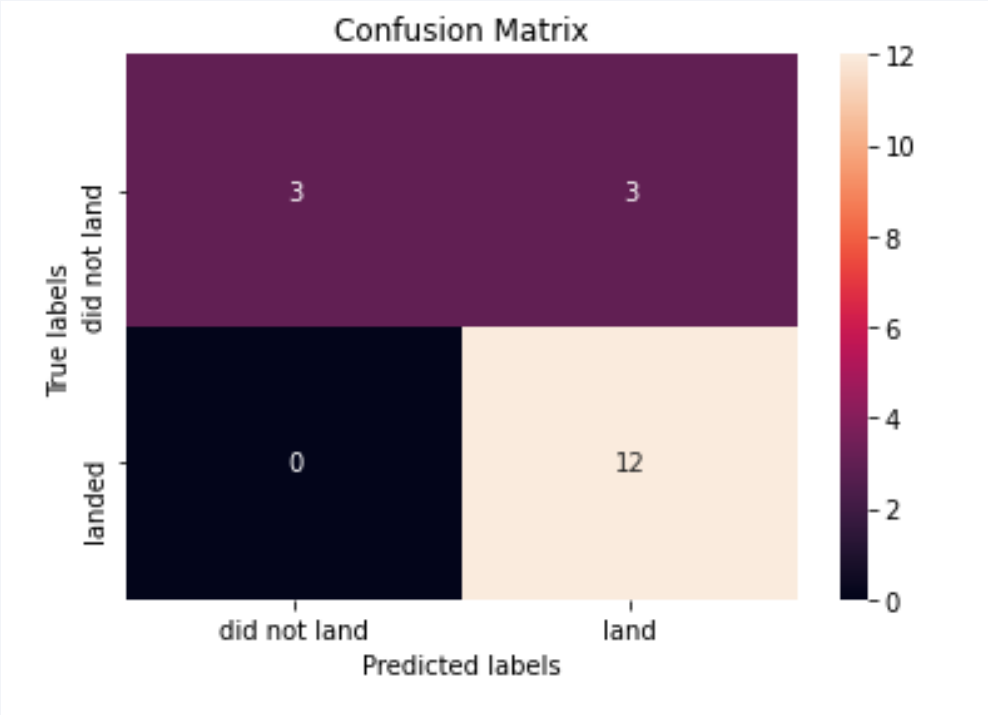
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

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

- The confusion matric for the decision tree classier shows that this classifier can distinguish between the different classes. However, a major problem is the flase positives of unsuccessful landing marked as successful landings.



Conclusions

- From this exploration the following conclusions can be made:
- The decision tree classifier had the highest accuracy and therefore the best machine learning algorithm for this task.
- The larder the number of flights at a launch sites, the greater the overall success rate for that site.
- The KSC LC-39A had overall the most successful launches of all the sites.
- The Launch success rate overall increases from 2013 to 2020, experiencing a dip between 2017 to 2018 and 2020 onwards.
- The orbit types with the best overall success were ES-L, GEO,HEO, SSO and VLEO.

Thank you!