Winning Space Race with Data Science

Nik Syareena Aida Binti Nik Ahmad Faizul

<31 January 2023>

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection via SpaceX REST API, Web Scrapping from Wikipedia
  - Data Wrangling
  - Exploratory Data Analysis (EDA) with SQL and visualization
  - Interactive Visual Analytics with Folium and Plotly Dash
  - Predictive Analysis
- Summary of all results
  - EDA results
  - Interactive Visual Analytics screenshots
  - Predictive Analysis results

Introduction

- Project background and context
  - SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars each whereas other providers cost upward of 165 million dollars each. This saving is caused by the fact that SpaceX can reuse the first stage of the launch. Therefore, this project's purpose is to predict the landing outcome of the first stage of the Falcon 9 for future times. From here we can also find out the cost of a launch. This information is useful if another company wants to bid against SpaceX for a rocket launch.

- Problems you want to find answers
  - Will the Falcon 9 first stage land successfully?
  - What are the characteristics that affect the success or failure of a landing?
  - What are the best conditions that result in a successful landing?

Section 1

Methodology

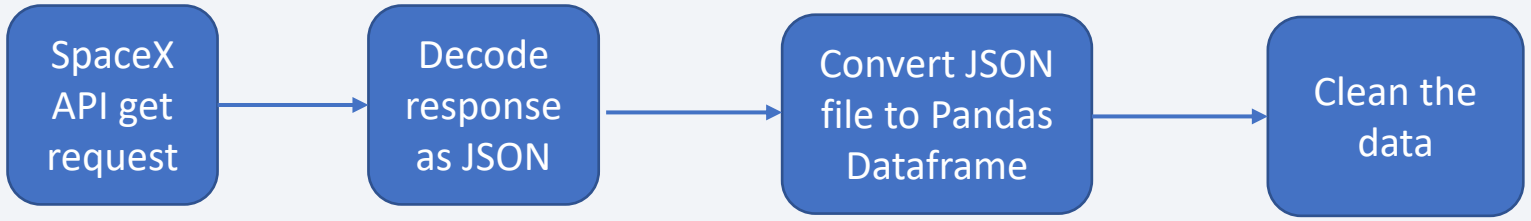
Methodology

Executive Summary

- Data collection methodology:
  - SpaceX REST API
  - Web Scrapping from Wikipedia
- Perform data wrangling
  - Filter unnecessary column/values
  - One hot encoding on 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
  - LR, SVM, TREE, KNN models were built and evaluated for the best model

Data Collection

1. **SpaceX REST API**
   - This API gives us data about launches, information about the rocket, payload delivered, launch and landing specifications , and landing outcome.
   - The API URL starts with api.spacexdata.com/v4/



2. **Web scrapping from wikipedia**
   - Python BeautifulSoup package used to scrape HTML tables that contain Falcon 9 launch records.
   - The URL is https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922

Data Collection – SpaceX API

SpaceX API GET Request

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

response = requests.get(spacex_url)
```

Make response as JSON and convert into a dataframe using json_normalize() function

```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

Data cleaning

```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that ha
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the feature
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

Github link: https://github.com/spdynns/IBM-Applied-Data-Science-Capstone/blob/main/DataCollection-api.ipynb

Data Collection - Scraping

Get HTML response

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

response = requests.get(static_url)
```

Create BeautifulSoup object

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content

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

Extract all column

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            #print(flight_number)
            datatimelist=date_time(row[0])
            launch_dict['Flight No.'].append(flight_number)
```

Create dataframe

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

Github link:
https://github.com/spdynns/IBM-Applied-Data-Science-Capstone/blob/main/DataCollection-webscraping.ipynb

Data Wrangling

- There were several cases where the booster did not land successfully.
  - True Ocean, True RTLS, True ASDS means successful outcomes.
  - False Ocean, False RTLS, False ASDS means failure.
- We convert those string outcomes into Training Labels with 1 means the booster successfully landed and 0 means it was unsuccessful.

Calculate the no. of launches on each site

```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

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

Calculate the no. of launches on each orbit

```
# Apply value_counts on Orbit colum
df['Orbit'].value_counts()
```

```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
```

Calculate the no. and occurrence of mission outcome per orbit type

```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
```

Create a landing outcome label from Outcome column

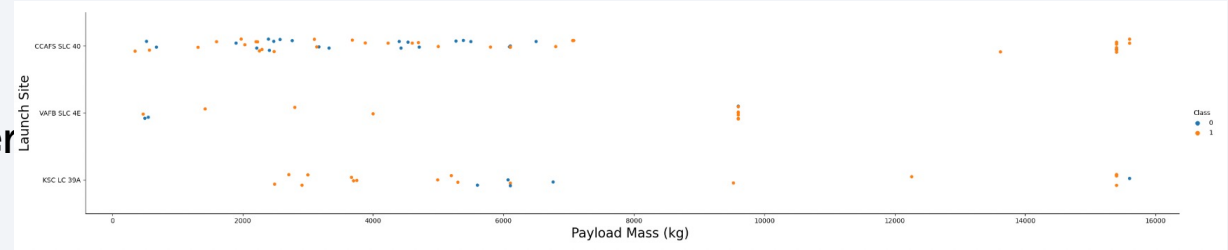
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []

for key,value in df['Outcome'].items():
    if value in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

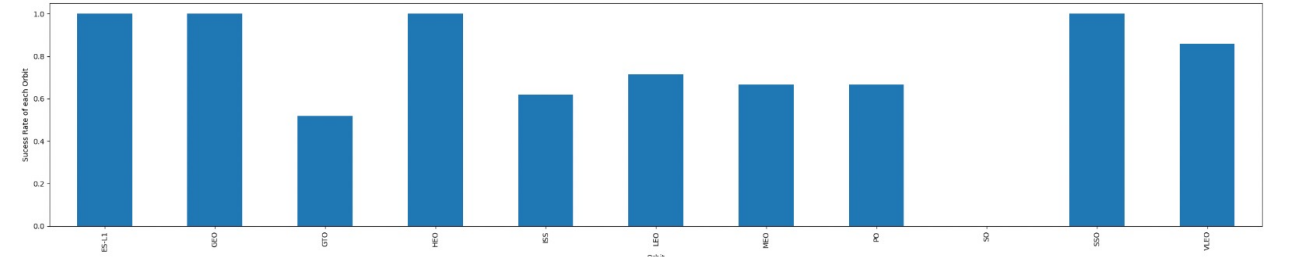
Github link: https://github.com/spdynns/IBM-Applied-Data-Science-Capstone/blob/main/DataWrangling.ipynb

EDA with Data Visualization

- With scatter plots we are able to find the relationship between multiple variables and identify correlational relationships. Such as:
  - Launch Site and Payload Mass
  - Launch Site and Flight Number
  - Payload Mass and Flight Number
  - Flight Number and Orbit Type
  - Payload and Orbit Type



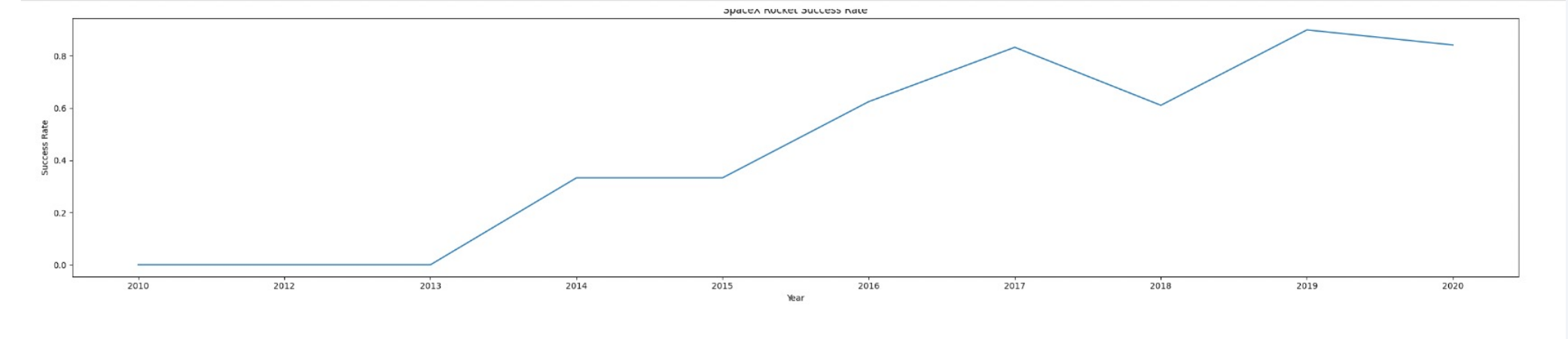
- For further analysis we also used bar chart to visualize the relationship between the success rate of each orbit type. Bar charts display the relationship between categorical and numerical variables.



Github link: https://github.com/spdynns/IBM-Applied-Data-Science-Capstone/blob/main/EDA_Visualisation.ipynb

EDA with Data Visualization

- Lastly, we plotted the launch success yearly trend using a line graph as it shows the pattern of the data over a period of time



Github link:
https://github.com/spdynns/IBM-Applied-Data-Science-Capstone/blob/main/EDA_Visualisation.ipynb

EDA with SQL

- **We performed many SQL queries to understand the data set better.**
  - Display the names of the unique launch sites in the space mission.
  - Display 5 records where launch sites begin with the string 'CCA'.
  - Display the total payload mass carried by boosters launched by NASA (CRS).
  - Display average payload mass carried by booster version F9 v1.1.
  - List the date when the first successul 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 booster_versions which have carried the maximum payload mass by using a subquery.
  - List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.
  - Rank the count of successful landing_outcome between the date 04-06-2010 and 20-03-2017 in descending order.

Github link:

https://github.com/spdynns/IBM-Applied-Data-Science-Capstone/blob/main/EDA_SQL.ipynb

Build an Interactive Map with Folium

- **We used Markers, Circles, MarkerCluster and Lines**
  - Markers – to mark points
    - Mark launch site coordinates.
    - Mark successful and unsuccessful landings (Green(1) for successful, Red(0) for unsuccessful).
    - Mark points between launch sites and key points (railway, highway, city, coastline).
  - Circles – to highlight areas
    - Red circle at each launch site along with its launch site name.
    - Red circle at NASA John Space Centre's coordinates also with its name as the label.
  - MarkerCluster - to group the events in a cluster in each coordinate
  - Lines – to show distance between 2 points
    - Plot a line between launch sites and key points (railway, highway, city, coastline).

Github link:
https://github.com/spdynns/IBM-Applied-Data-Science-Capstone/blob/main/Folium_Visualisation.ipynb

Build a Dashboard with Plotly Dash

- The dashboard contains a dropdown menu, range slider, pie chart and a scatter plot
  - A dropdown menu allows user to choose all launch sites or a specific launch site.
  - A range slider allows user to choose a payload mass in a fixed range.
  - A pie chart is to show the successful launches by total or by a specific launch site.
  - A scatter plot is to show the correlation between payload and success.

Github link: https://github.com/spdynns/IBM-Applied-Data-Science-Capstone/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

Model Building

- Load data set
- Create a column for the class
- Standardize the data
- Split data into training and testing sets
- Set the parameters to GridSearchCV and fit the data

Model Evaluation

- Calculate the accuracies for each model
- Obtain the best parameters for each model
- Plot the confusion matrix for respective models

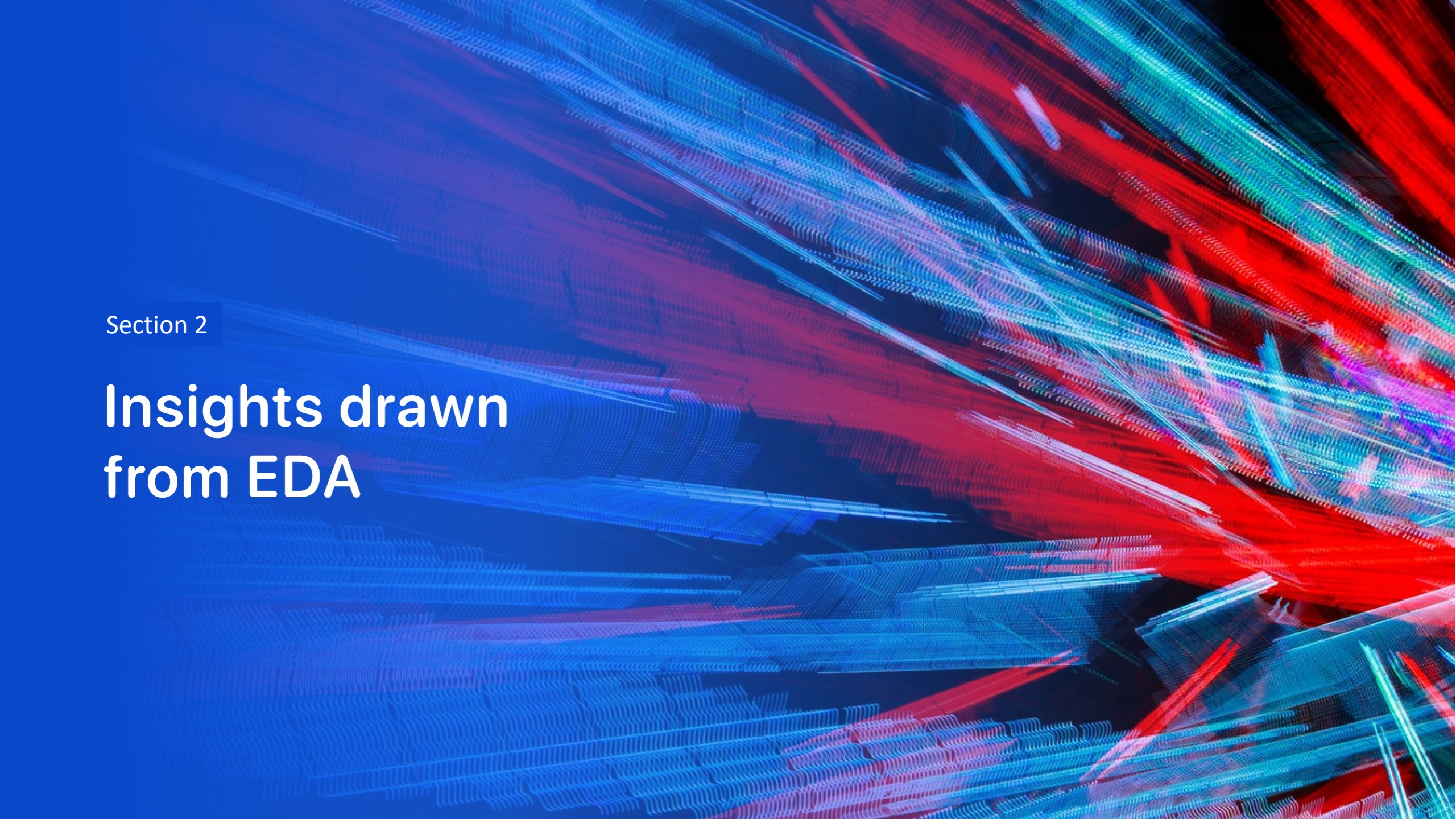
Model Comparison

- Find the best hyperparameters for the models
- Determine the model with the highest accuracy
- The best model is the one with highest accuracy

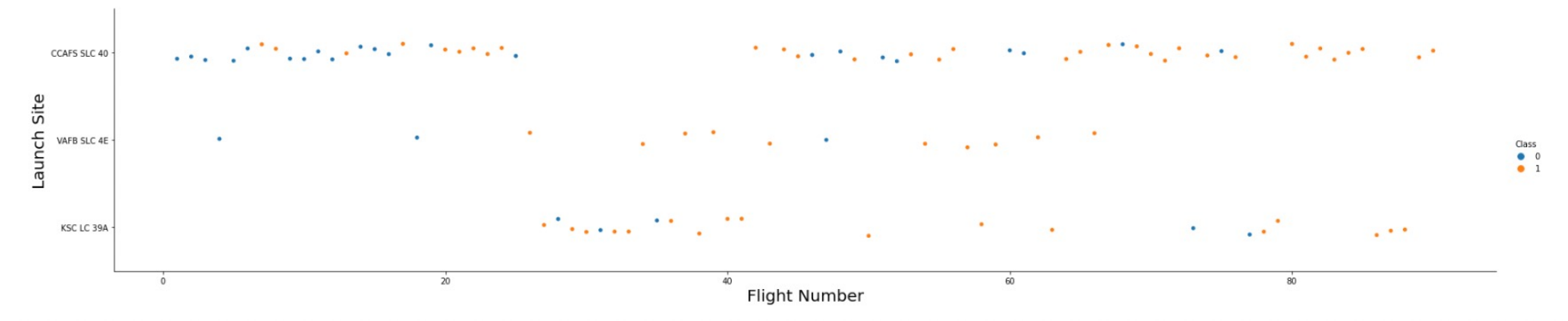
Github link:
https://github.com/spdynns/IBM-Applied-Data-Science-Capstone/blob/main/Machine Learning Prediction.ipynb

Results

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

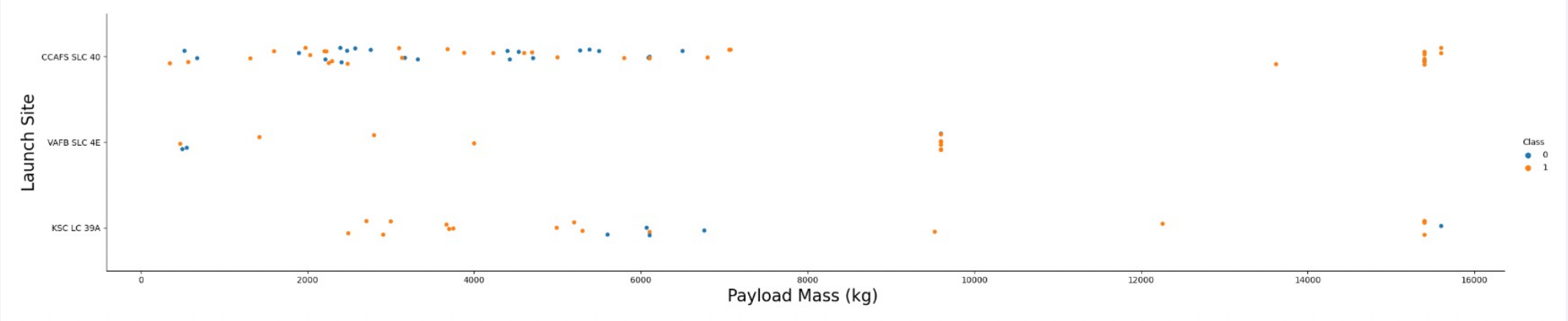

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



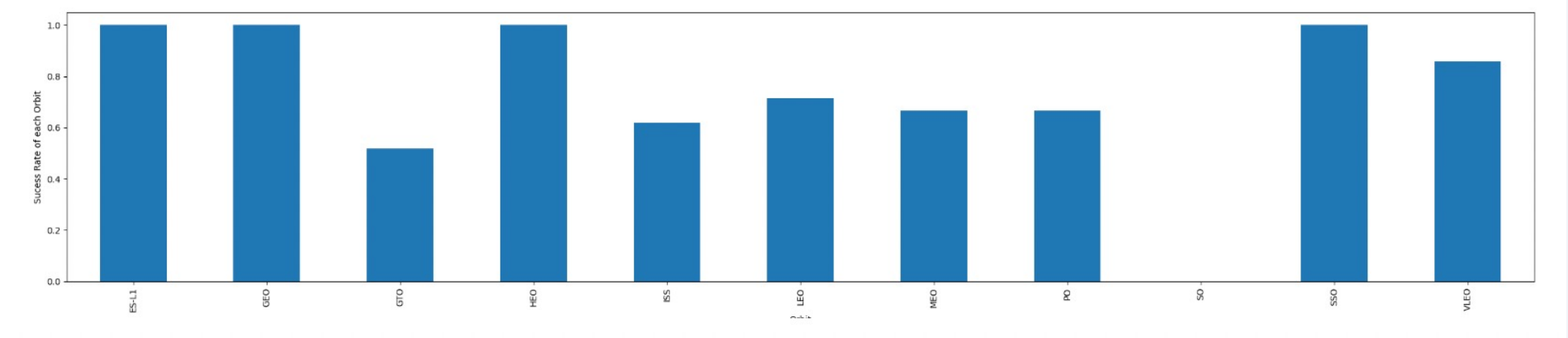
- From the scatter plot above, we can say that the best launch site is CCAFS SLC 40 as it has the most success of launches.
- Generally, we can also observe that the larger the flight number, the higher the success rate.

Payload vs. Launch Site



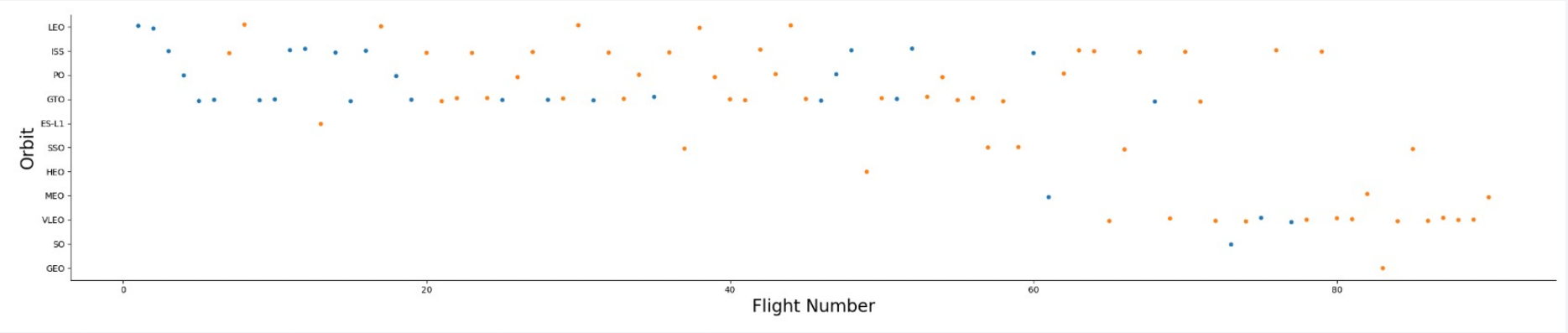
- Payloads above 8000 kg seem to have a high success rate
- However, with heavier payload (above 14000kg) seem to only be unsuccessful on launch site VAFB SLC 4E.

Success Rate vs. Orbit Type



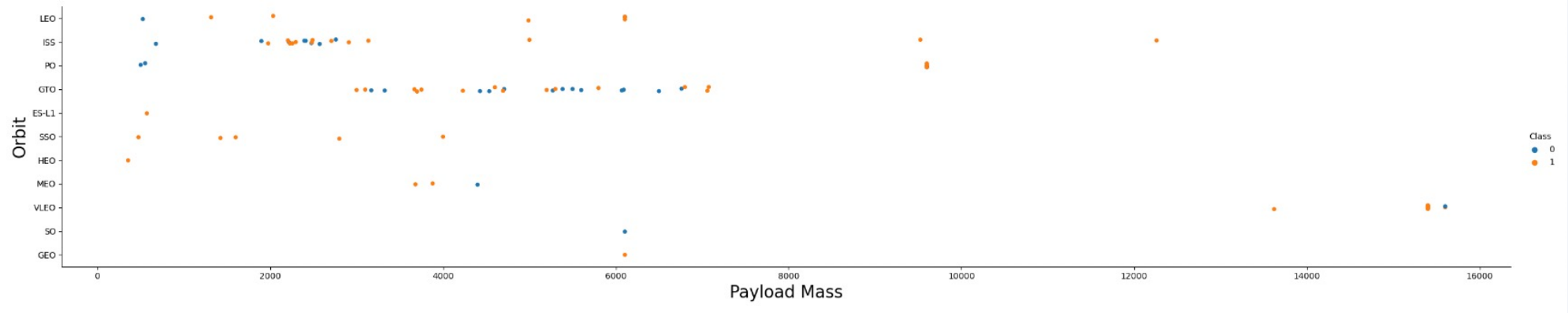
- Every orbit type have very distinctive success rates.
- The highest success rates only occur on ES-L1, GEO, HEO, and SSO
- SO on the other hand, have 0 success rate

Flight Number vs. Orbit Type



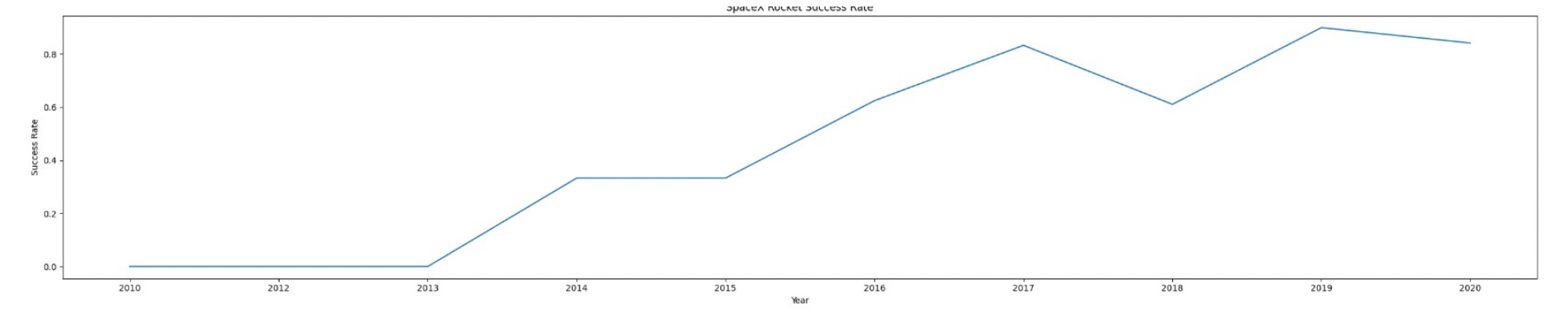
- For LEO orbit, the success rate seems to be related to the flight number.
- It also appears that there is no relationship between flight number and GTO orbit.

Payload vs. Orbit Type



- Heavy payloads are only effective on LEO, ISS, Polar.
- There seem to be zero relationship between payload and GTO .
- HEO, SO, GEO, need further analyzation with other variables.

Launch Success Yearly Trend



- The success rate kept on increasing from the year 2013 until 2020.

All Launch Site Names

- Launch site names:

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

- SQL Query:

```
%sql SELECT DISTINCT "LAUNCH_SITE" FROM SPACEXTBL
```

  - SELECT DISTINCT was used to remove and duplicate values of the launch sites.

Launch Site Names Begin with 'CCA'

- Result:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	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)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- SQL Query:

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

  - WHERE clause specifies the column name.
  - LIKE clause specifies the substring ‘CCA’.
  - LIMIT limits the result to only 5.

Total Payload Mass

- Result:

TOTAL PAYLOAD LAUNCHED BY NASA (CRS)
45596

- SQL Query:

```
%sql SELECT SUM("PAYLOAD_MASS__KG_") AS "TOTAL PAYLOAD LAUNCHED BY NASA (CRS)" FROM SPACEXTBL WHERE "CUSTOMER" = "NASA (CRS)"
```

  - SUM function calculates the sum of the column selected.
  - WHERE clause filters the rows where customer = ‘NASA (CRS)’.

Average Payload Mass by F9 v1.1

- Result:

AVG("PAYLOAD_MASS__KG_")
2534.6666666666665

- SQL Query:

```
%sql SELECT AVG("PAYLOAD_MASS__KG_") FROM SPACEXTBL WHERE "BOOSTER_VERSION" LIKE "F9 v1.1%"
```

  - SELECT AVG was used to calculate the average payload mass.
  - WHERE clause is used to specifiy the booster version.

First Successful Ground Landing Date

- Result:

MIN("DATE")
01-05-2017

- SQL Query:

```
%sql SELECT MIN("DATE") FROM SPACEXTBL WHERE "LANDING _OUTCOME" LIKE '%Success%'
```

  - SELECT MIN was used to obtain the earliest ground landing date.
  - WHERE clause is used to only return the landing_outcome with the substring ‘Success’.

Successful Drone Ship Landing with Payload between 4000 and 6000

- Result:

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

- SQL Query:

```
%sql SELECT "BOOSTER_VERSION" FROM SPACEXTBL WHERE "LANDING _OUTCOME" = 'Success (drone ship)' \
AND "PAYLOAD_MASS__KG_" > 4000 AND "PAYLOAD_MASS__KG_" < 6000;
```

  - This returns the Successful Drone Ship Landing with Payload between 4000 and 6000.

Total Number of Successful and Failure Mission Outcomes

- Result:

SUCCESS	FAILURE
100	1

- SQL Query:

```
%sql SELECT (SELECT COUNT("MISSION_OUTCOME") FROM SPACEXTBL WHERE "MISSION_OUTCOME" LIKE '%Success%') AS 'SUCCESS', \
(SELECT COUNT("MISSION_OUTCOME") FROM SPACEXTBL WHERE "MISSION_OUTCOME" LIKE '%Failure%') AS 'FAILURE'
```

  - Subquery is used to select the total success of mission_outcomes and the total failure of mission_outcomes.

Boosters Carried Maximum Payload

- Result:

Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

- SQL Query:

```
%sql SELECT DISTINCT "BOOSTER_VERSION" FROM SPACEXTBL\
WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTBL)
```

  - Subquery is used to select boosters with the maximum payload mass.

2015 Launch Records

- Result:

MONTH	Booster_Version	Launch_Site
01	F9 v1.1 B1012	CCAFS LC-40
04	F9 v1.1 B1015	CCAFS LC-40

- SQL Query:

```
%sql SELECT substr("DATE", 4, 2) AS MONTH, "BOOSTER_VERSION", "LAUNCH_SITE" FROM SPACEXTBL\
WHERE "LANDING _OUTCOME" = 'Failure (drone ship)' and substr("DATE",7,4) = '2015'
```

  - This query returns month, booster_version, and launch_site where failed landings took place in 2015.

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

- Result:

Landing _Outcome	COUNT("LANDING _OUTCOME")
Success	20
Success (drone ship)	8
Success (ground pad)	6

- SQL Query:

```
%sql SELECT "LANDING _OUTCOME", COUNT("LANDING _OUTCOME") FROM SPACEXTBL\
WHERE "LANDING _OUTCOME" LIKE '%Success%' AND "DATE" >= '04-06-2010' AND "DATE" <= '20-03-2017'\
GROUP BY "LANDING _OUTCOME" \
ORDER BY COUNT("LANDING _OUTCOME") DESC ;
```

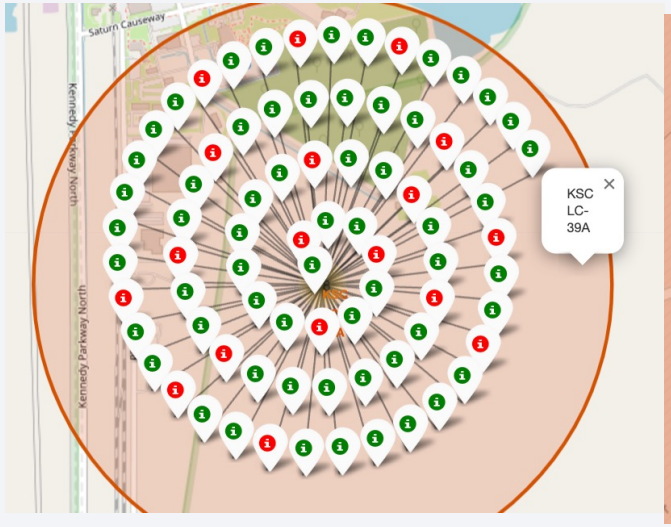
- This query returns the successful landing outcomes along with its count between 04-06-2010 and 20-03-2017.
- GROUP BY clause groups the results by landing_outcome.
- ORDER BY clause displays the result in descending order of the landing outcome.

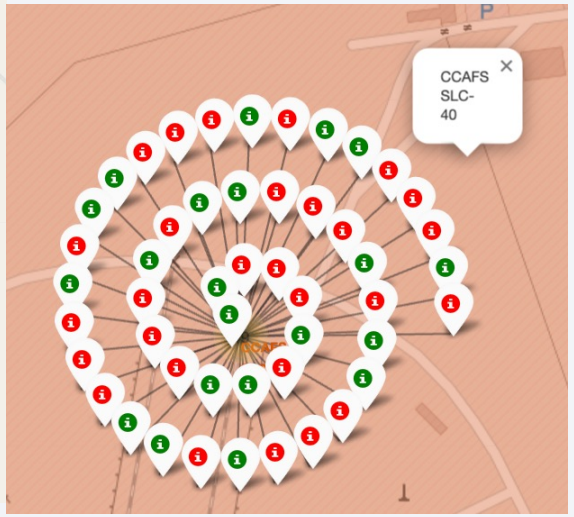

Section 3
Launch Sites
Proximities Analysis

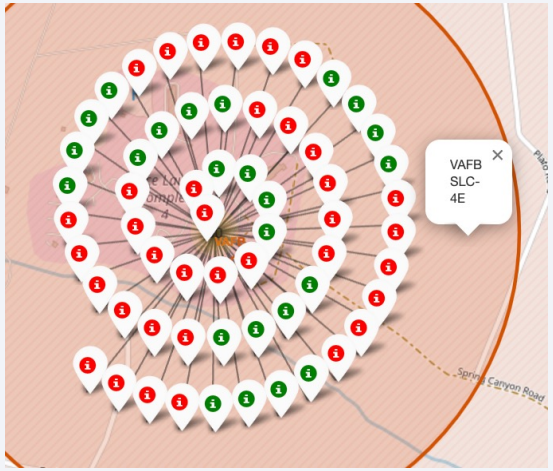
All launch site locations

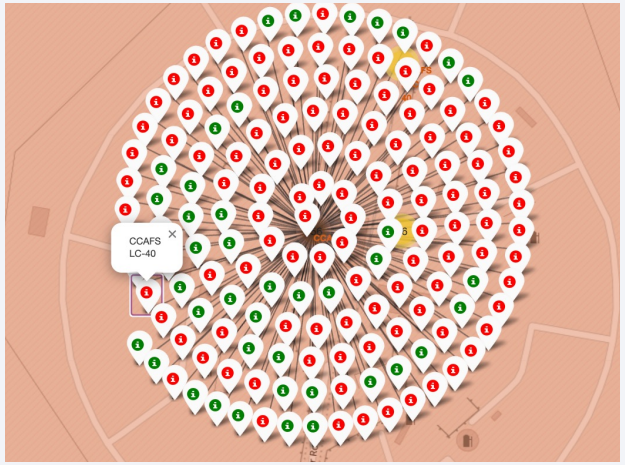
- All launch sites are located at coastlines most probably due to safety measures.
- All launch sites are located in the United States.

Color labelled launch sites



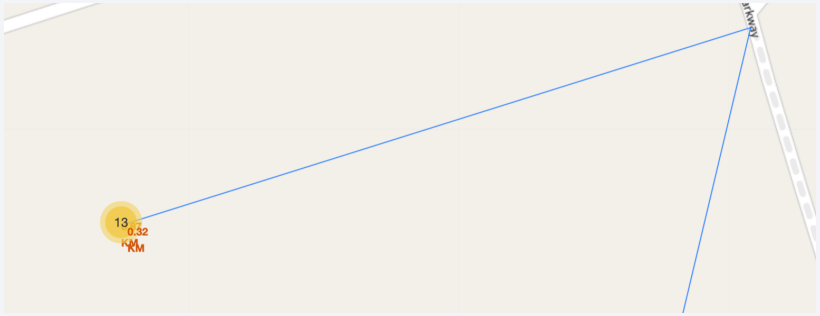


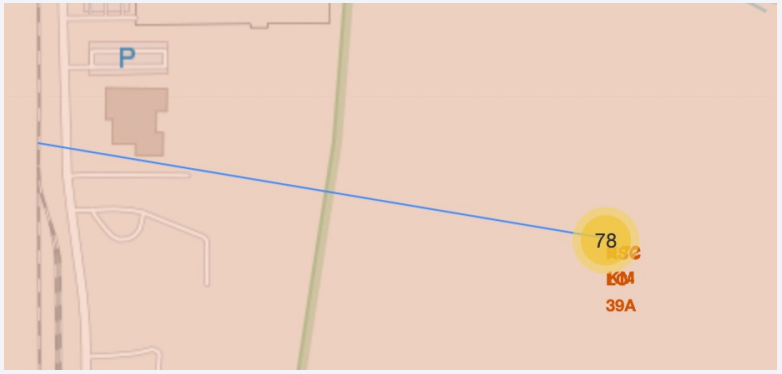




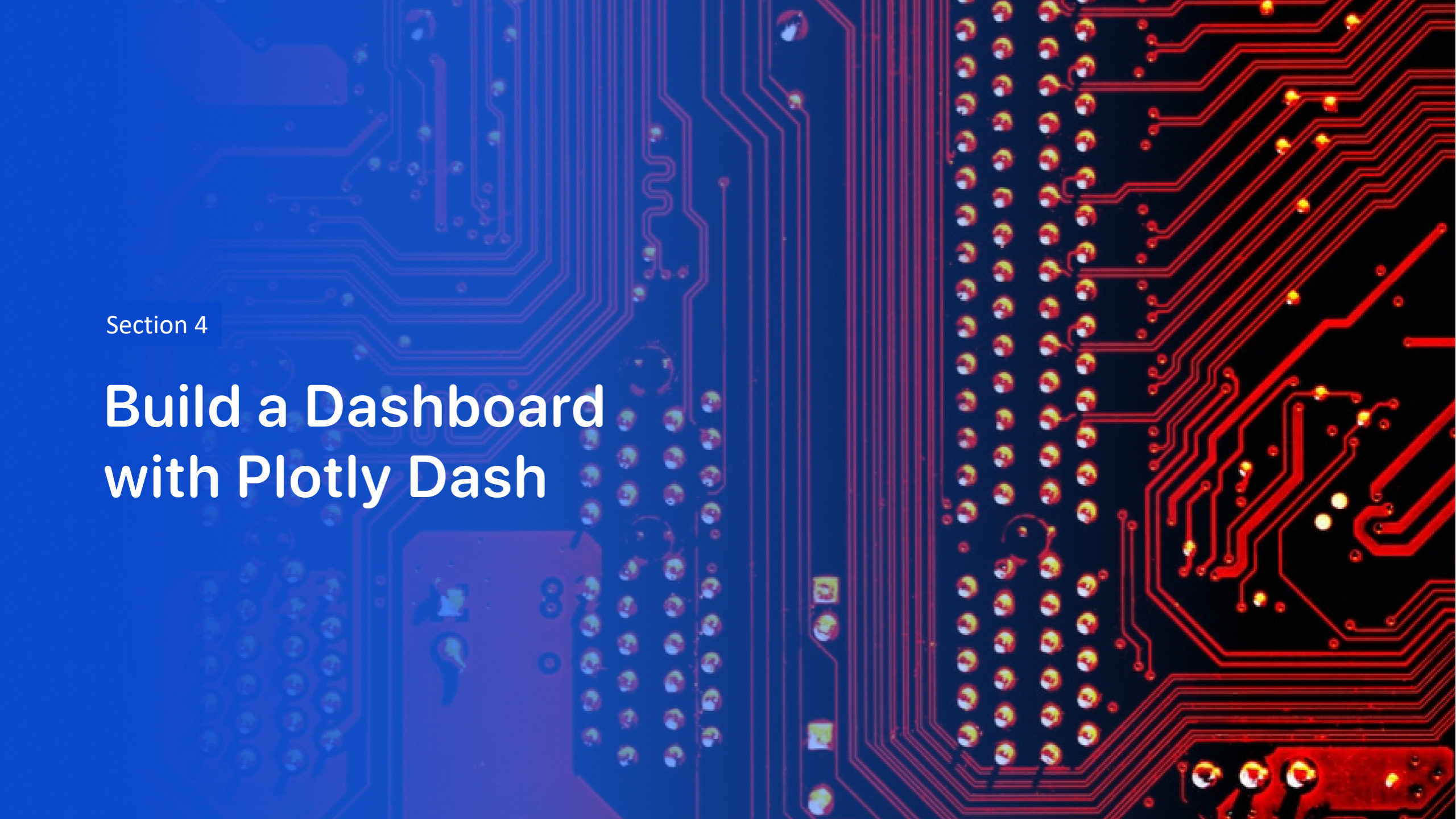
- Green markers show successful launches.
- Red markers show unsuccessful launches.

Launch site distance with its proximities

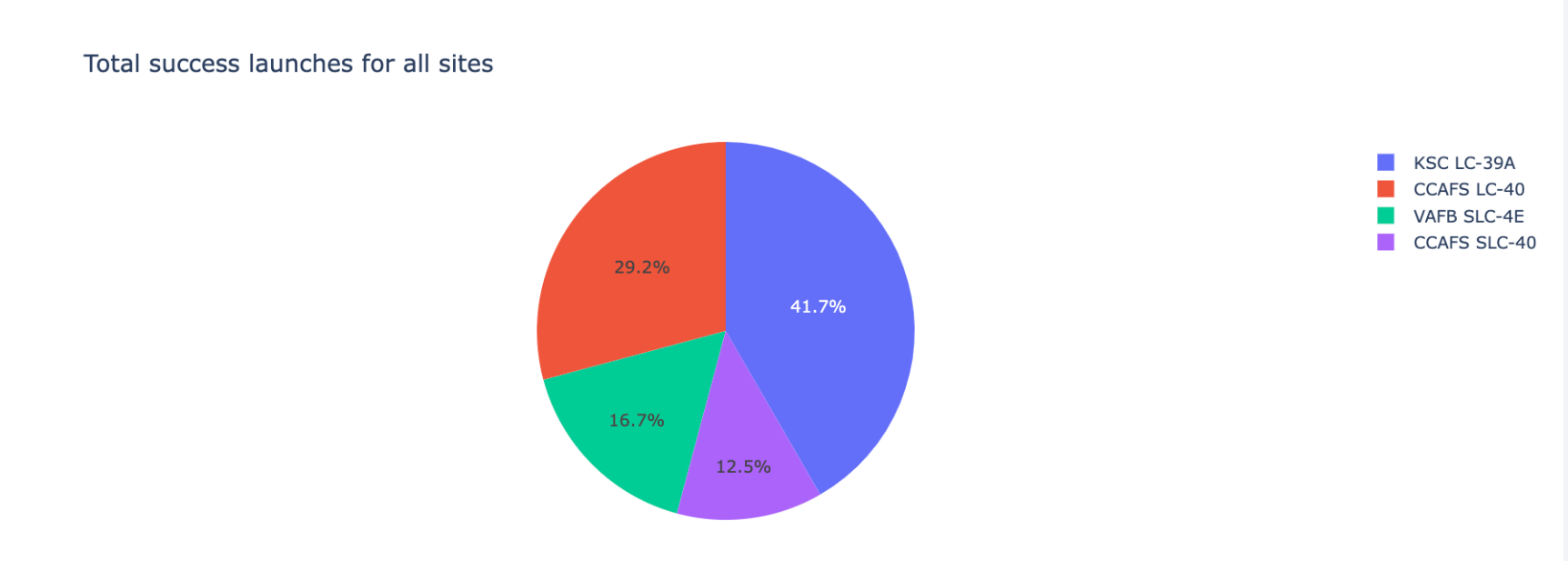




- Launch sites are in close proximity to railways.
- Launch sites are in close proximity to highways.
- Launch sites are in close proximity to coastline.
- Launch sites do keep a certain distance away from cities.

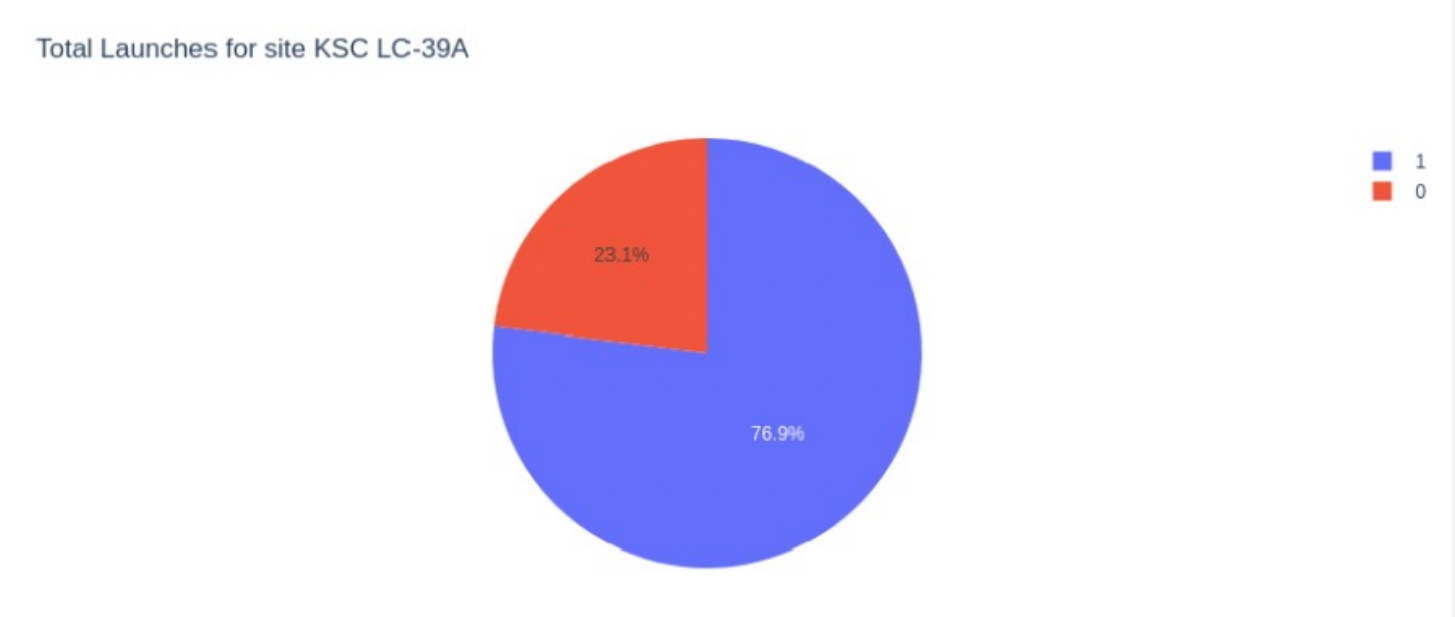

Section 4
Build a Dashboard with Plotly Dash

Total success launches for all sites



- Launch site CAFS SLC-40 has the lowest success.
- Launch site KSC LC-39A has the highest success.

Total launches for site KSC LC-39A



- The site has a success rate of 76.9% with a failure rate of 23.1%

Payload Mass Vs Launch Outcome Scatter Plot

Heavy weighted payload (5000kg -10000kg)

- Success rate with heavy weighted payload is low

Payload Mass Vs Launch Outcome Scatter Plot

Low weighted payload (0kg-5000kg)

- Success rate with low weighted payload is high

Section 5

Predictive Analysis (Classification)

Classification Accuracy

```
print('Accuracy on test data:', logreg_cv.score(X_test, Y_test)
```

```
Accuracy on test data: 0.8333333333333334
```

```
print('Accuracy on test data:', svm_cv.score(X_test, Y_test))
```

```
Accuracy on test data: 0.8333333333333334
```

```
print('Accuracy on test data:', tree_cv.score(X_test, Y_test))
```

```
Accuracy on test data: 0.8333333333333334
```

```
print('Accuracy on test data:', knn_cv.score(X_test, Y_test))
```

```
Accuracy on test data: 0.8333333333333334
```

- Logistic Regression model, SVM model, Decision Tree model and KNN model were built.
- All models have the same accuracy on the test data.

Classification Accuracy

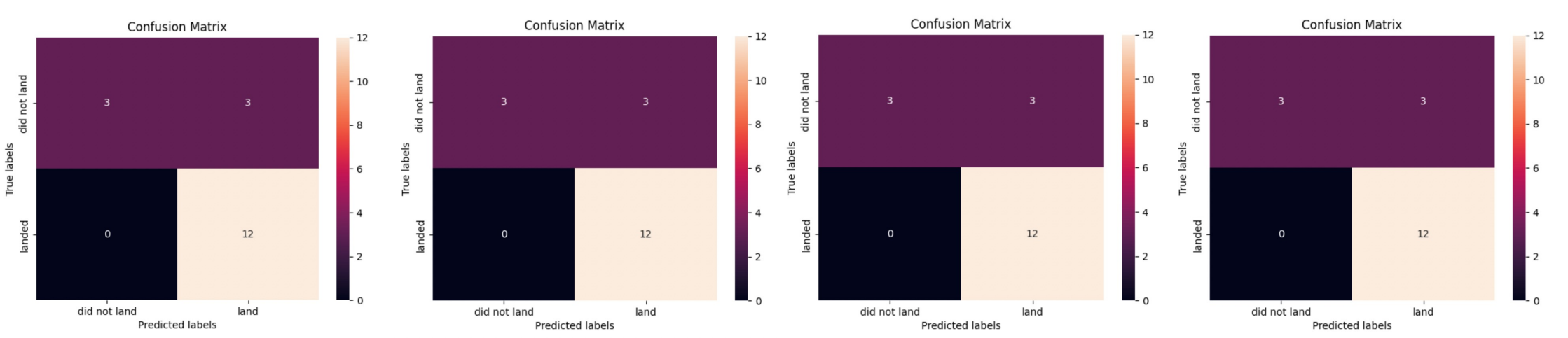
```
methods = {'LogisticRegression': logreg_cv.best_score_, 'SVM': svm_cv.best_score_, 'Tree':tree_cv.best_score_, 'KNN':knn_cv.best_score_}
bestmethod= max(methods, key=methods.get)
print('The method that performs best is:', bestmethod, 'with score', methods[bestmethod])
if bestmethod == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestmethod == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestmethod == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
```

```
The method that performs best is: Tree with score 0.8892857142857142
Best Params is : {'criterion': 'entropy', 'max_depth': 8, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 10, 'splitte
r': 'random'}
```

- However, the Decision Tree model has the best accuracy with the training data. Therefore, the best model will be the Decision Tree.

Confusion Matrix

Logistic Regression SVM Decision Tree KNN



- The confusion matrix for all classification models are equal as they all have the same accuracy on the test data.

Conclusions

- The success of a launch can be studied with the relationship of multiple factors such as the orbit, flight number, launch site and payload.
- ES-L1, GEO, HEO, and SSO orbits have the highest success rate.
- Launch site KSC LC-39A has the highest success rate than other launches.
- Low weighted payload performed better than heavy weighted payload.
- Best classification model is Decision Tree as it has higher accuracy on the training data.

Thank you!