IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Pelluru Pavan Kumar Reddy
10th August 2024

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 Analytics result
- Interactive insights (screenshots)
- Predictive Analytics result

Introduction

- Project background and context
  - Space X 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. Therefore, if we can determine if the first stage will land, we can determine the cost of the launch. This info can be used if an alternative company wants to bid against space X for a project launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

- Problems you want to find answers
  - What factors determine if the rocket will land successfully?
  - The interaction amongst various features that determine the success rate of a successful landing.
  - What operating conditions needs to be in place to ensure a successful landing program.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Collected using SpaceX API and web scraping from Wikipedia
- Perform data wrangling
  - Usage of One-hot encoding
- 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 collection process involved a combination of API requests from SpaceX REST API and Web Scraping data from a table in SpaceX's Wikipedia entry. Both these methods were utilized in order to get complete info about the launches.
  - Data columns like the FlightNumber, Date, BoosterVersion, PayloadMass, Orbit, LaunchSite, Outcome, Flights, GridFins, Reused, Legs, LandingPad, Block etc were obtained using SpaceX REST API.
  - Data columns like the Flight No., Launch Site, Payload, Payload Mass, Orbit, Customer, launch outcome were obtained by using Wikipedia Web Scraping.

Data Collection – SpaceX API

- The API used is https://api.spacexdata.com/v4/rockets/.
- The API provides data about many types of rocket launches done by SpaceX, the data is therefore filtered to include only Falcon 9 launches.
- Every missing value in the data is replaced the mean the column that the missing value belongs to.
- Notebook Link: https://github.com/Pavanreddy-2003/IBM-Data-Science-Capstone/blob/main/%5B1%5D%20Data%20Collection%20API.ipynb

Out[50]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	
4	1	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	
5	2	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False	
6	3	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False	
7	4	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False	
8	5	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False	
...	...	...	...	...	...	...	...	...	...	...	...	
89	86	2020-09-03	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	2	True	True	True	5e9e3032383

Data Collection - Scraping

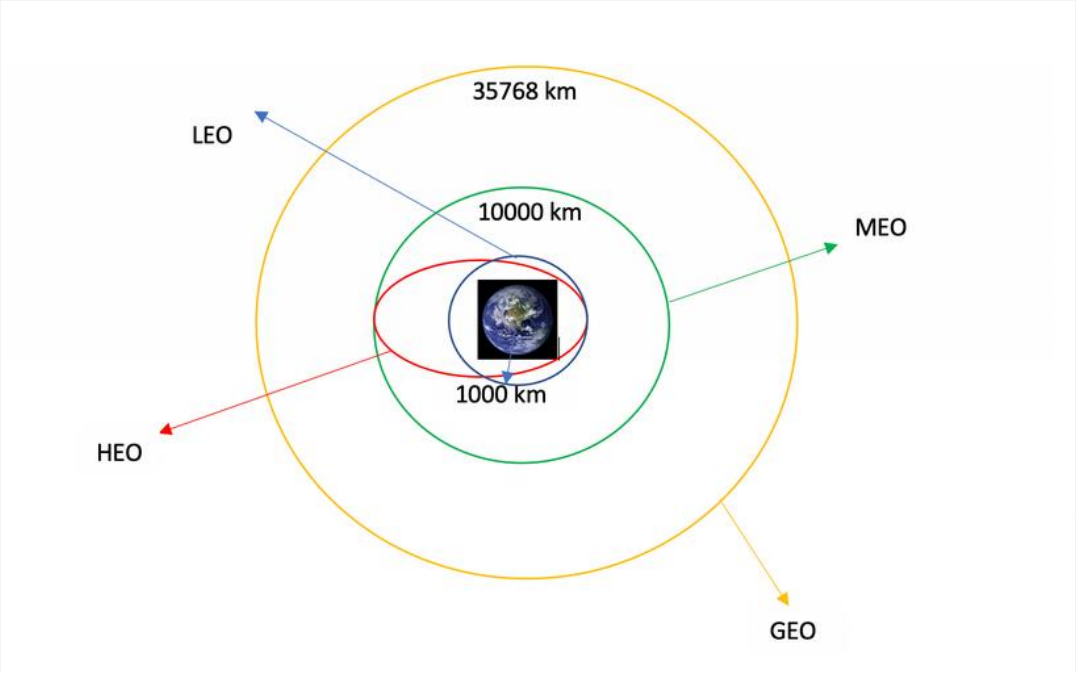
- Applied Web Scraping to webscrap Falcon 9 launch records with BeautifulSoup.
- Parsed the table and converted it into a pandas dataframe.
- Notebook Link: https://github.com/Pavanreddy-2003/IBM-Data-Science-Capstone/blob/main/%5B2%5D%20Data%20Collection%20with%20Web%20Scraping%20.ipynb

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsib
   # 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])

            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            date = datatimelist[0].strip(',')
            #print(date)

            # Time value
            # TODO: Append the time into launch_dict with key `Time`
            time = datatimelist[1]
            #print(time)
```

Data Wrangling

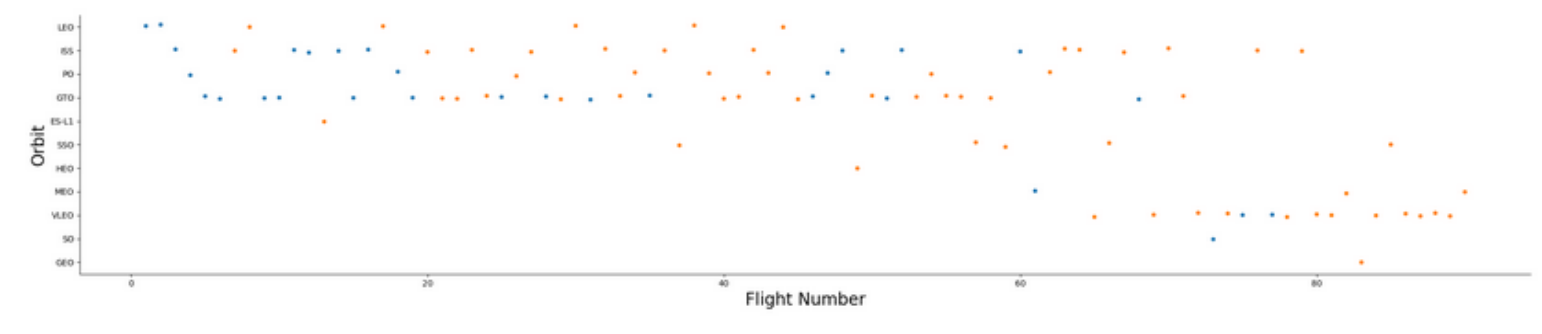
- Calculated number of launches at each site, and the number and occurrence of each orbits and exported the results into csv file.

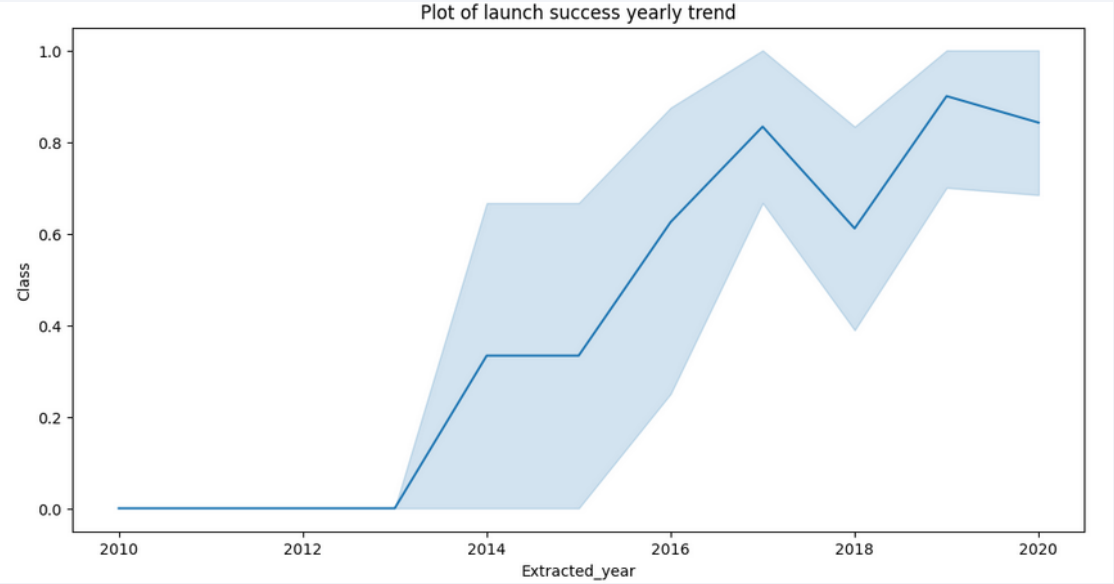
- Notebook link: https://github.com/Pavanreddy-2003/IBM-Data-Science-Capstone/blob/main/%5B3%5D%20Data%20Wrangling.ipynb



EDA with Data Visualization

- Explored data by visualizing relationship between columns like 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.
- Notebook link: https://github.com/Pavanreddy-2003/IBM-Data-Science-Capstone/blob/main/%5B5%5D%20EDA%20Visualization.ipynb





EDA with SQL

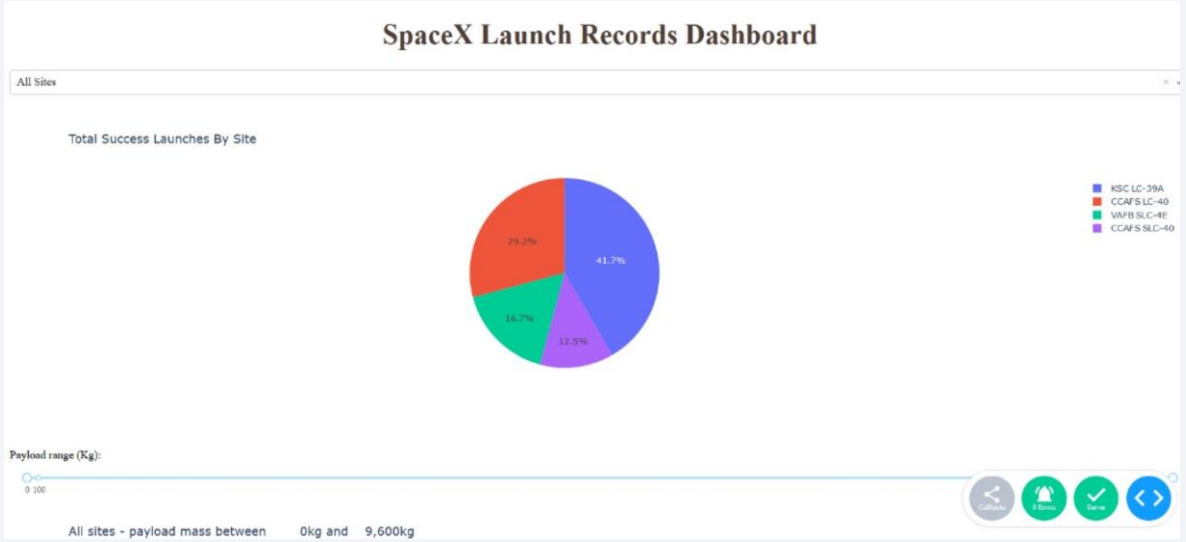
- Loaded the SpaceX dataset into PostgreSQL database without leaving the jupyter notebook.
- Then applied EDA with SQL to get insight from the data. Wrote queries to find out for instance:
- The names of unique launch sites
- Total payload mass carried by boosters launched by NASA(CRS)
- Average payload mass carried by booster version F9 v 1.1
- Notebook Link: https://github.com/Pavanreddy-2003/IBM-Data-Science-Capstone/blob/main/%5B4%5D%20EDA%20With%20SQL.ipynb

Build an Interactive Map with Folium

- We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class O and 1.i.e., O for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the distances between a launch site to its proximities. We answered some question for instance:
- Are launch sites near railways, highways and coastlines.
- Do launch sites keep certain distance away from cities.

Build a Dashboard with Plotly Dash

- Built an interactive dashboard with Plotty dash
- Plotted pie charts showing total launches by a certain sites
- Plotted scatter graph showing relationship with outcome and payload mass(Kg) for different booster version
- Notebook link: https://github.com/Pavanreddy-2003/IBM-Data-Science-Capstone/blob/main/%5B7%5D%20spacex_dash_app.py

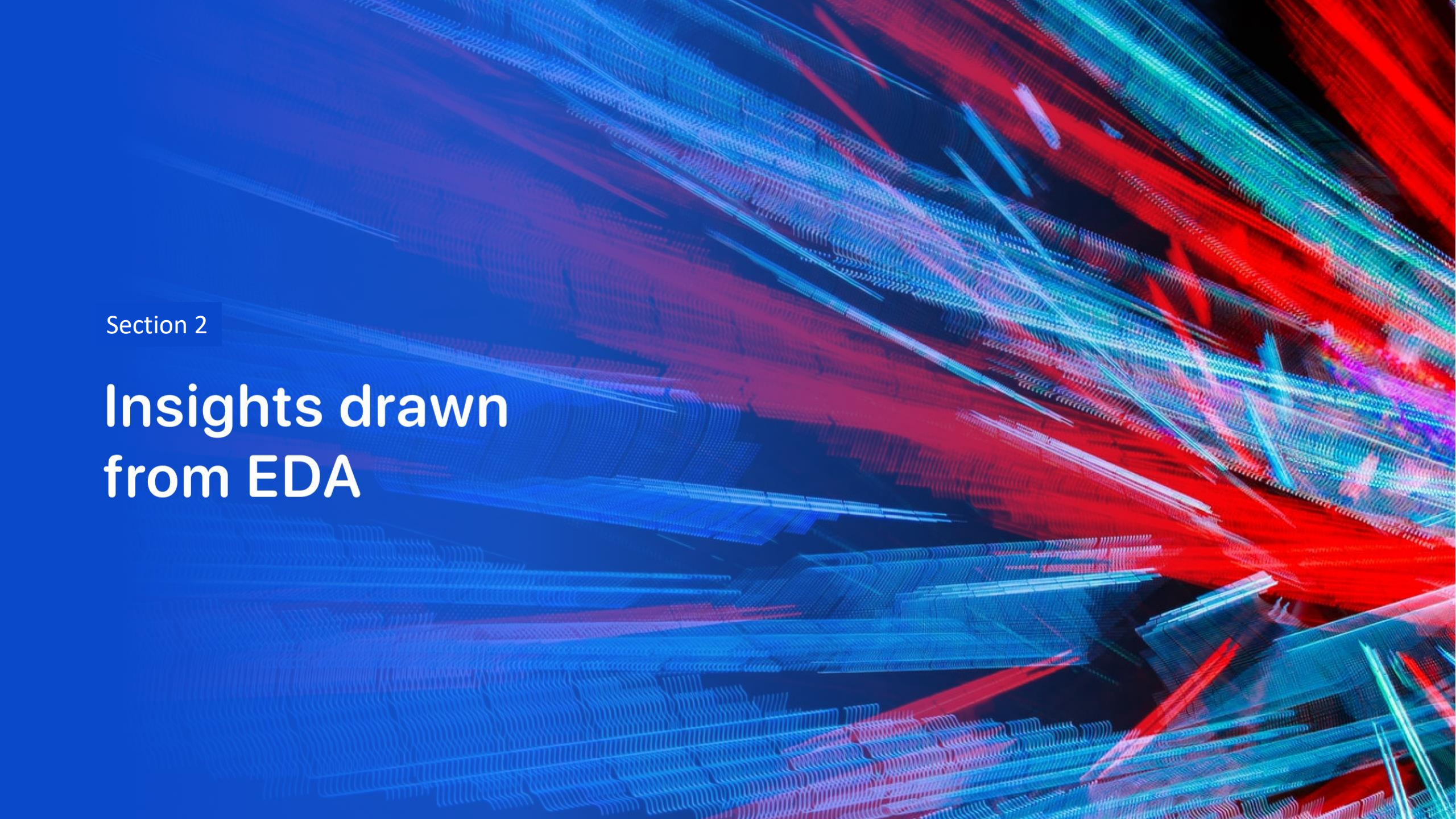


Predictive Analysis (Classification)

- Loaded the data using numpy and pandas, transformed the data, split out data into training and testing
- Created a GridSearch CV object with cv = 10 to find the best parameters
- Applying GridSearchCV on LogReg, SVM, Decision Tree and KNN models
- Calculating the accuracy on the test data using the method.score() method
- Examining the confusion matrix for all models
- Finding the method that performs the best by examining the Jaccard_score and F1_score metrics
- Notebook Link: https://github.com/Pavanreddy-2003/IBM-Data-Science-Capstone/blob/main/%5B8%5D%20Machine%20Learning%20Pediction.ipynb

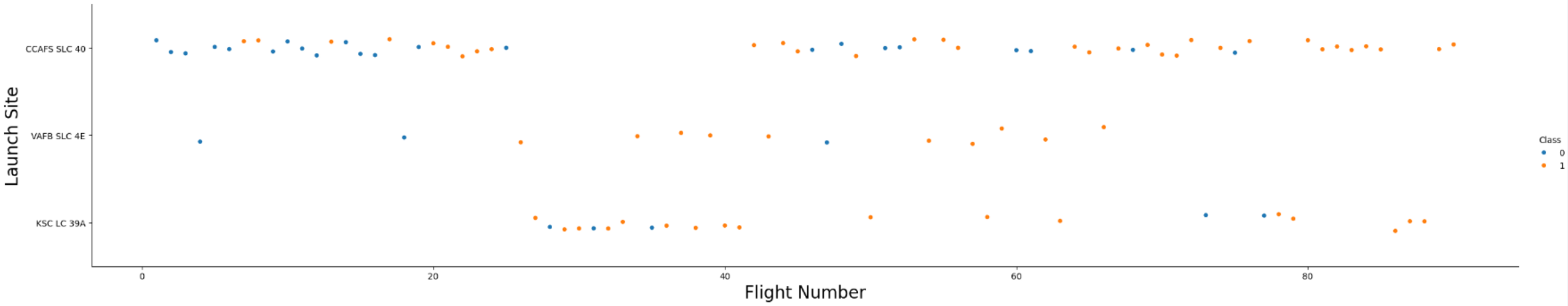
Results

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


Section 2
Insights drawn from EDA

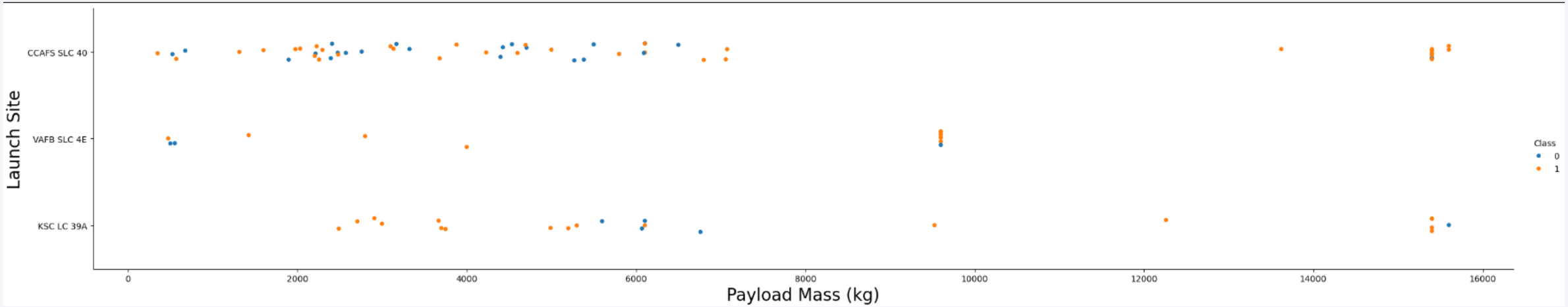
Flight Number vs. Launch Site

- We can infer from the plot that larger the flight amount at a launch site, greater the success rate at a launch site.



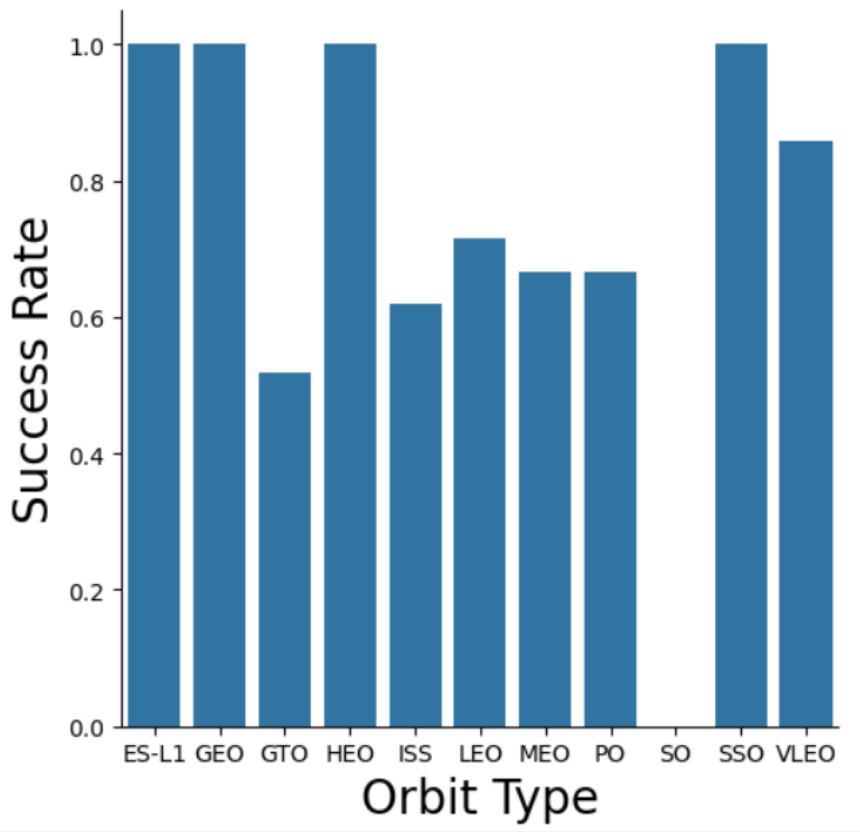
Payload vs. Launch Site

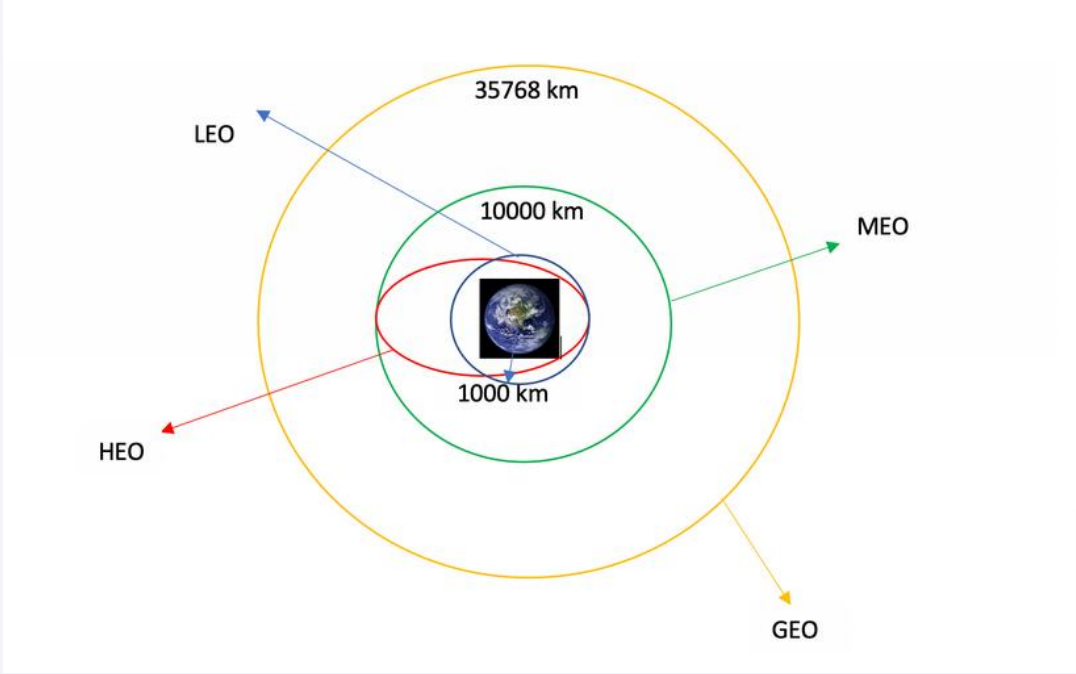
- Greater the payload mass for launch site CCAFS SLC 40, higher the success rate.



Success Rate vs. Orbit Type

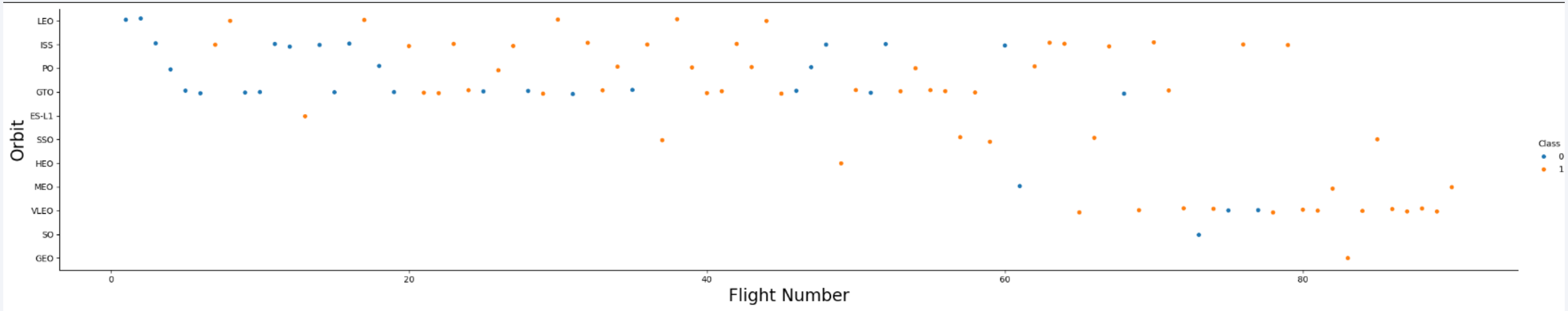
- ES-L1, GEO, HEO, SSO, VLEO have the most success rate.





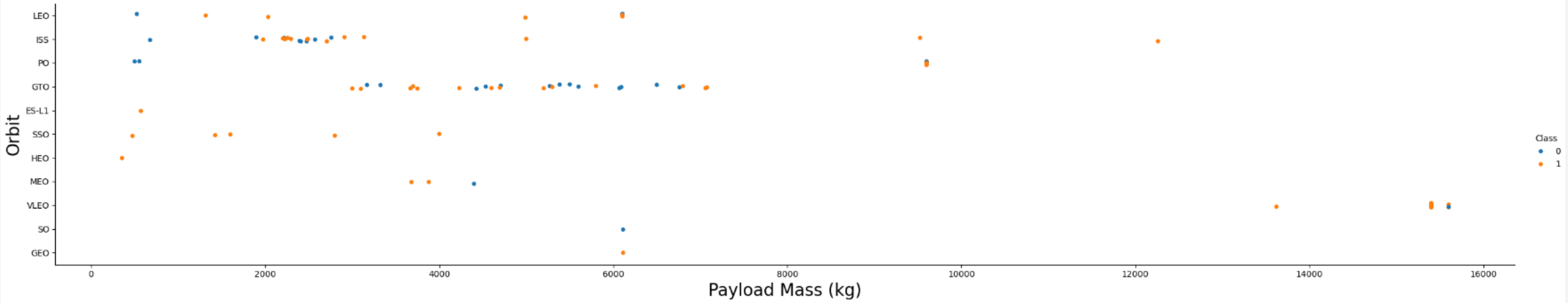
Flight Number vs. Orbit Type

- In LEO orbit, success is related to number of flights whereas in the GTO orbit, there is no relationship between flight number and orbit



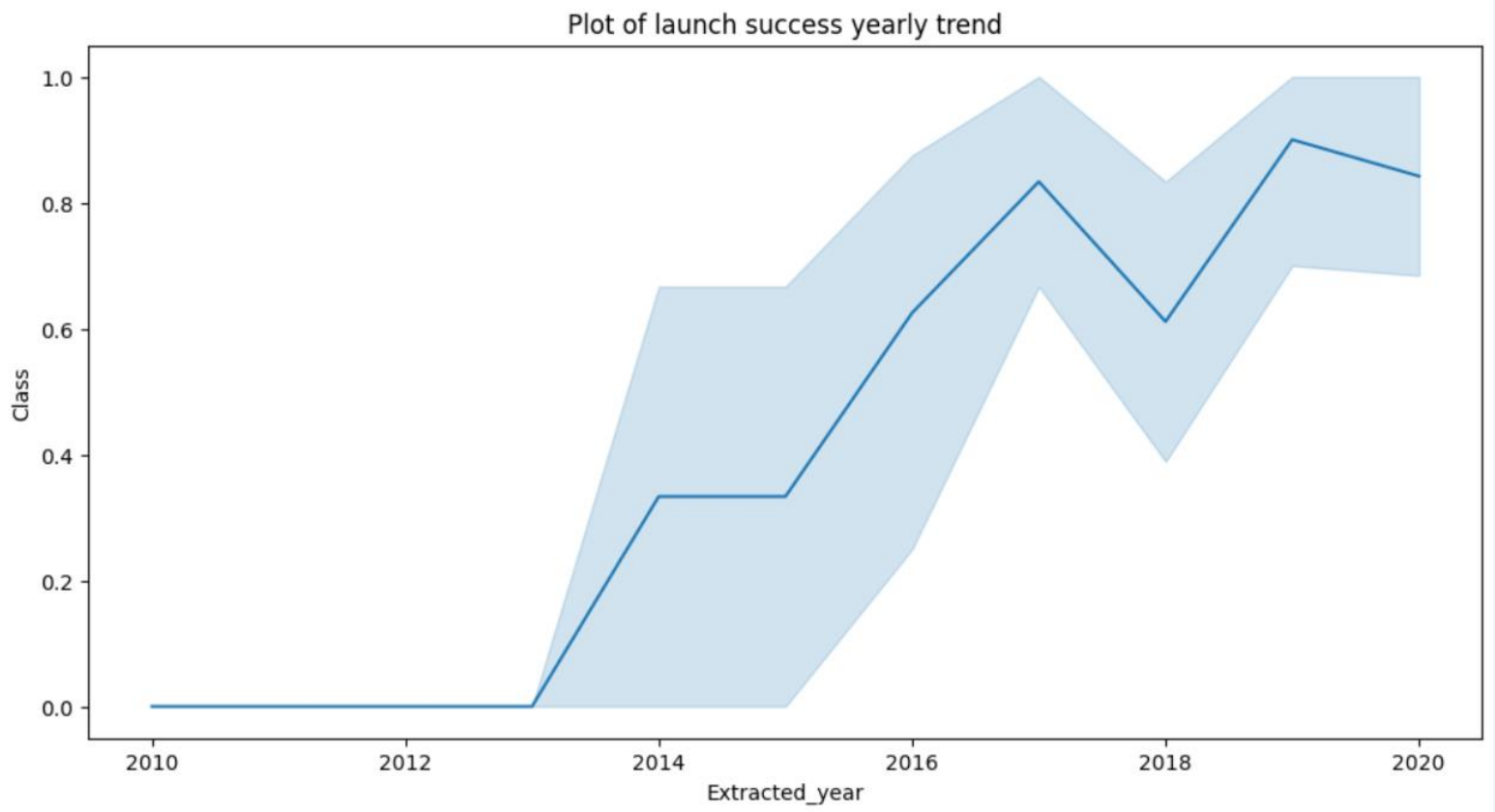
Payload vs. Orbit Type

- Successful landing is more for PO, LEO and ISS orbits



Launch Success Yearly Trend

- Success rate has been increasing from 2013 to 2020



All Launch Site Names

- Keyword **DISTINCT** is used to show only unique launch sites form the SpaceX data.

```
%sql select distinct launch_site from SPACEXTABLE;
```

```
 * sqlite:///my_data1.db
Done.
Out[9]:
```

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

Launch Site Names Begin with 'KSC'

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

In [10]:

```
%sql select * from SPACEXTABLE where launch_site like 'KSC%' limit 5;
```

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

Out[10]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer
2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)
2017-03-16	6:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	5600	GTO	EchoStar
2017-03-30	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES
2017-05-01	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300	LEO	NRO
2017-05-15	23:21:00	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070	GTO	Inmarsat

Total Payload Mass

Task 3

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

In [11]:
```
%sql select sum(payload_mass__kg_) as total_payload_mass from SPACEXTABLE where customer = 'NASA (CRS)';
```

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

Out[11]:

total_payload_mass
45596

Average Payload Mass by F9 v1.1

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

In [12]:
```
%sql select avg(payload_mass__kg_) as average_payload_mass from SPACEXTABLE where booster_version like '%F9 v1.1%';
```

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

Out[12]:

average_payload_mass
2534.6666666666665

First Successful Ground Landing Date

- As we can see, the first successful landing outcome on drone ship was 8th April 2016.

List the date where the succesful landing outcome in drone ship was acheived.

Hint:Use min function

In [17]:
```
%sql select min(date) as first_successful_landing from SPACEXTABLE where landing_outcome = 'Success (drone ship)';
```

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

Out[17]:

first_successful_landing
2016-04-08

Successful Drone Ship Landing with Payload between 4000 and 6000

- We 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 ground pad and have payload mass greater than 4000 but less than 6000

In [18]:
```
%sql select booster_version from SPACEXTABLE where landing_outcome = 'Success (ground pad)' and payload_mass__kg_ between 4
```

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

Out[18]:

Booster_Version
F9 FT B1032.1
F9 B4 B1040.1
F9 B4 B1043.1

Total Number of Successful and Failure Mission Outcomes

In [20]:
```
%sql select mission_outcome, count(*) as total_number from SPACEXTABLE group by mission_outcome;
```

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

Out[20]:

Mission_Outcome	total_number
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

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

In [22]:
```
%sql select booster_version from SPACEXTABLE where payload_mass__kg_ = (select max(payload_mass__kg_) from SPACEXTABLE);
```

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

Out[22]:

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

2015 Launch Records

- List the records which will display the month names, succesful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017

List the records which will display the month names, succesful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017

Note: SQLLite does not support monthnames. So you need to use substr(Date,6,2) for month, substr(Date,9,2) for date, substr(Date,0,5),='2017' for year.

In [31]:
```
%sql SELECT substr(Date,4,2) as month, DATE,BOOSTER_VERSION, LAUNCH_SITE, [landing_outcome] \
FROM SPACEXTABLE \
where [landing_outcome] = 'Success (ground pad)' and substr(Date,7,4)='2017';
```

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

Out[31]:

month	Date	Booster_Version	Launch_Site	Landing_Outcome

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

- Selected landing outcomes and the **COUNT** of landing outcomes form the data and used the **WHERE** clause to filter for landing outcomes **BTEWEEN** the dates
- Then applied **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome in descending order.

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

In [26]:
```
%%sql select landing_outcome, count(*) as count_outcomes from SPACEXTABLE
      where date between '2010-06-04' and '2017-03-20'
      group by landing_outcome
      order by count_outcomes desc;
```

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

Out[26]:

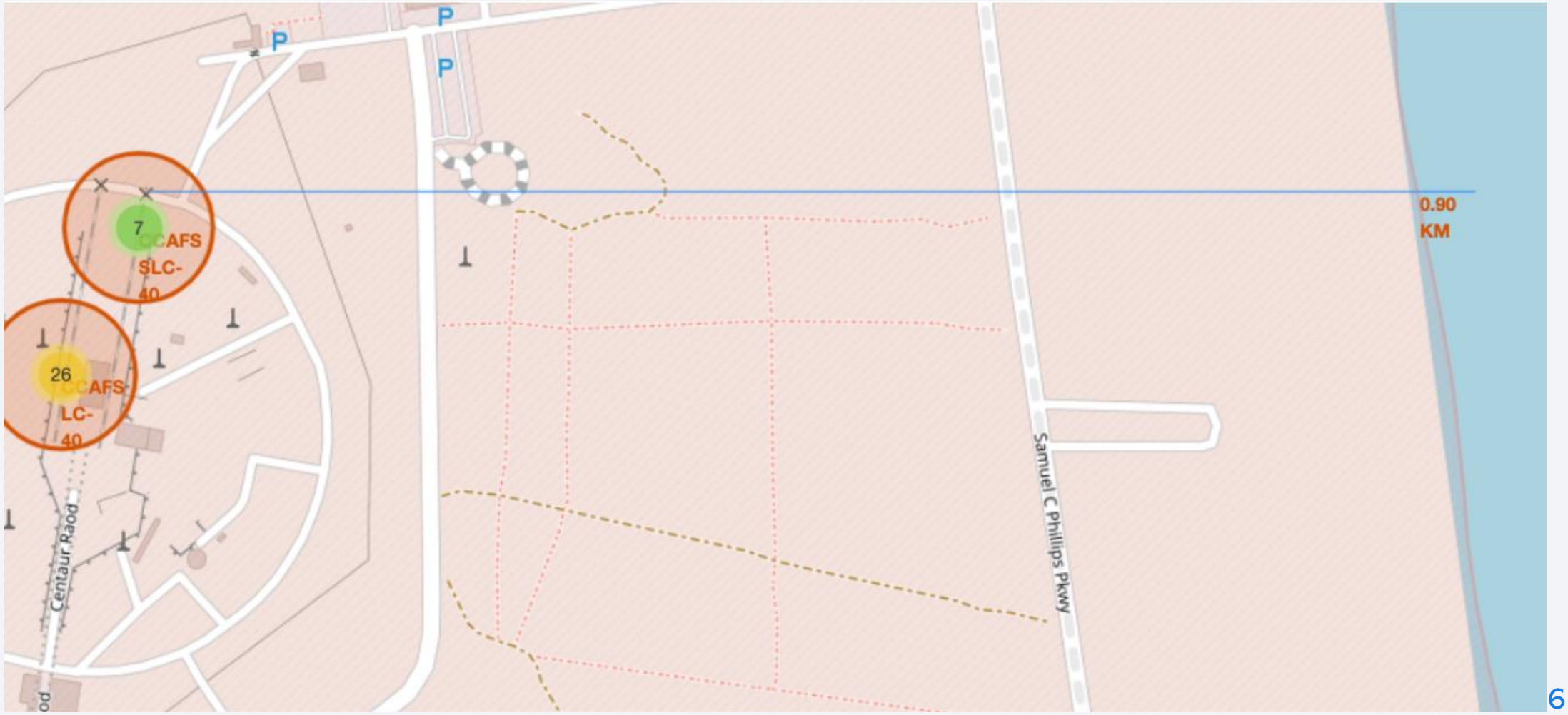
Landing_Outcome	count_outcomes
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

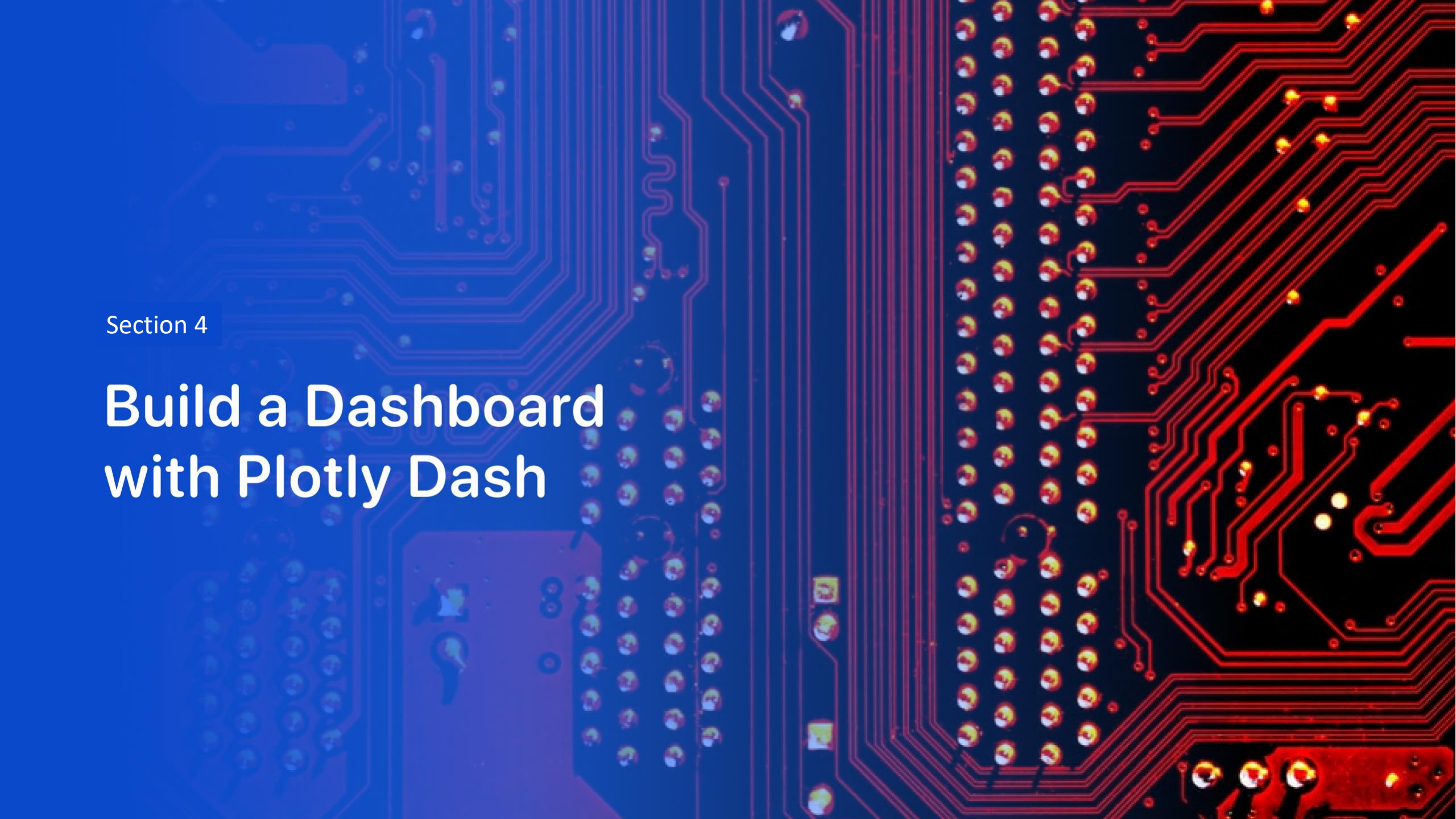
Section 3

Launch Sites Proximities Analysis

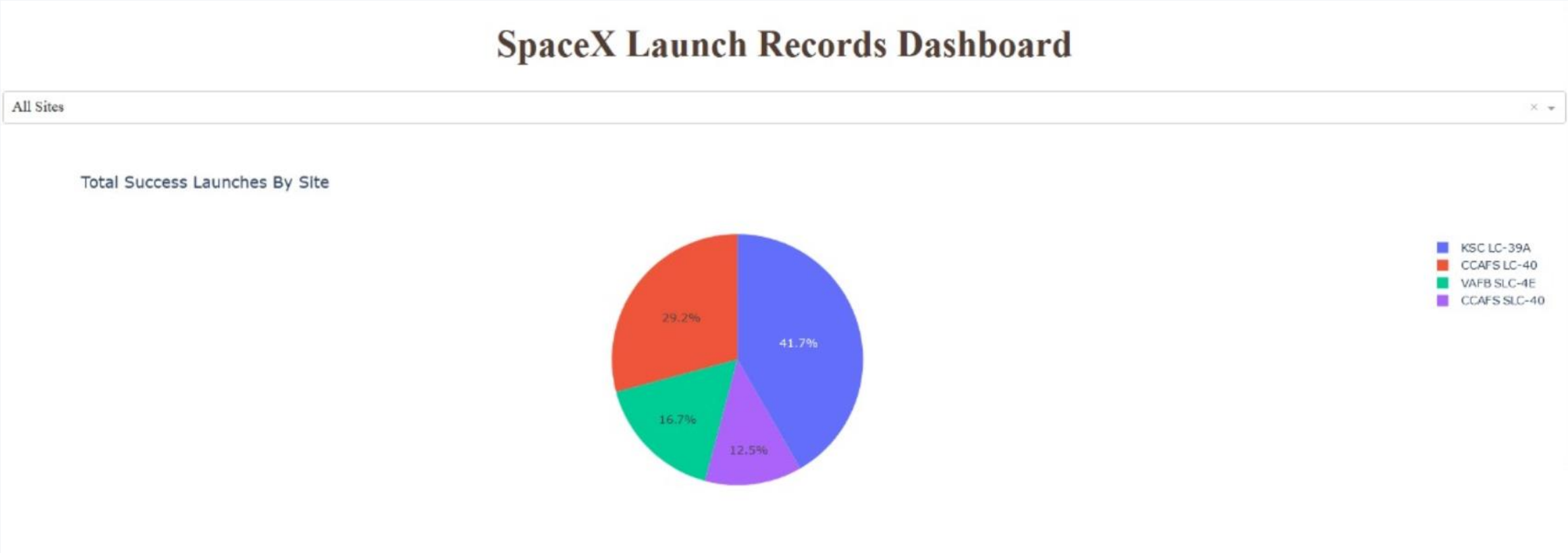
Launch sites Global Map Markers

Launch Site distance to landmarks

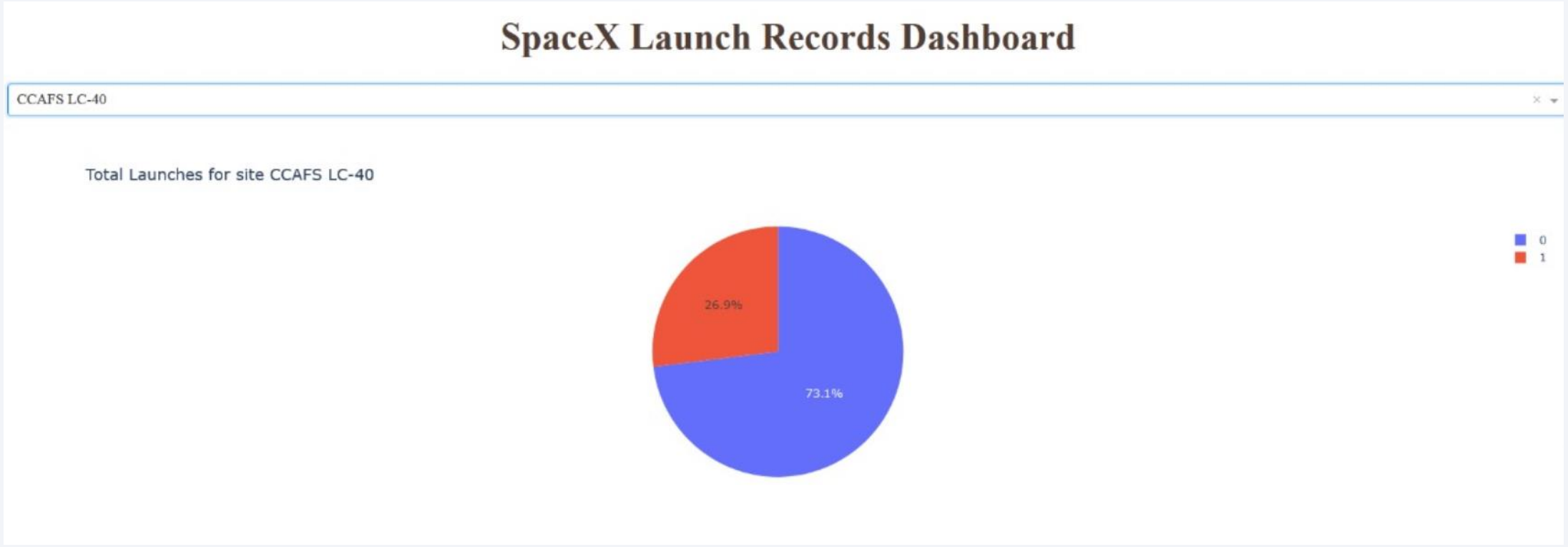



Section 4
Build a Dashboard with Plotly Dash

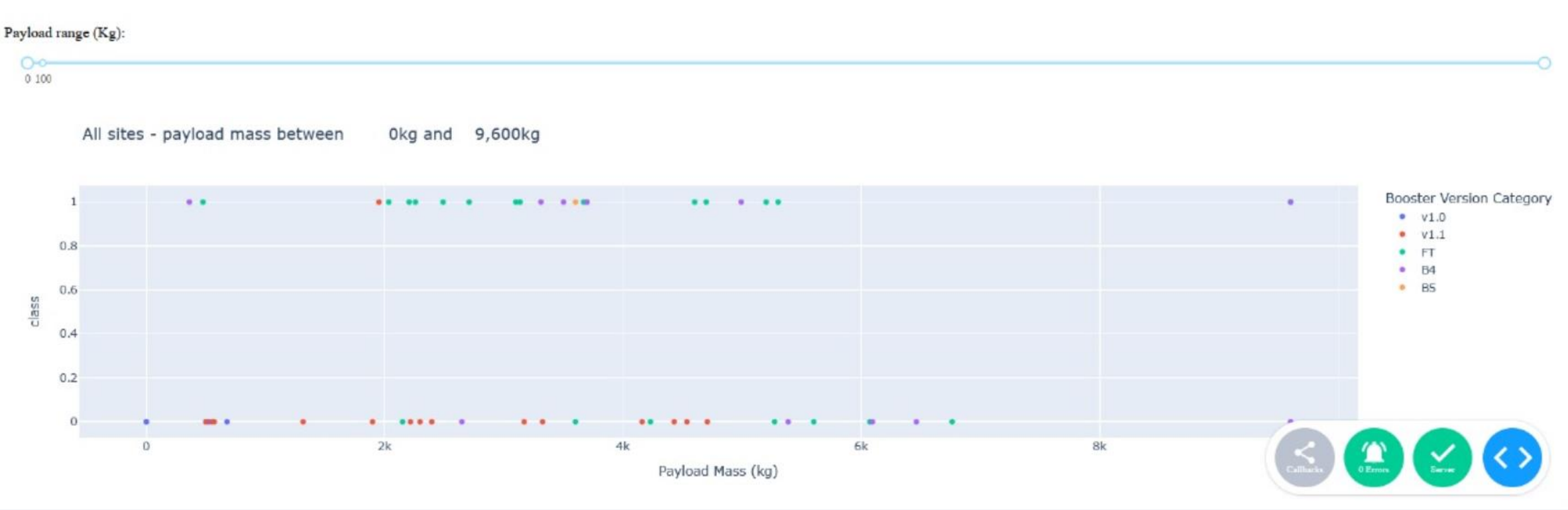
Pie chart showing success % achieved by each launch site



Pie chart showing launch site (CCAFS LC-40) success ratio



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



Section 5

Predictive Analysis (Classification)

Classification Accuracy

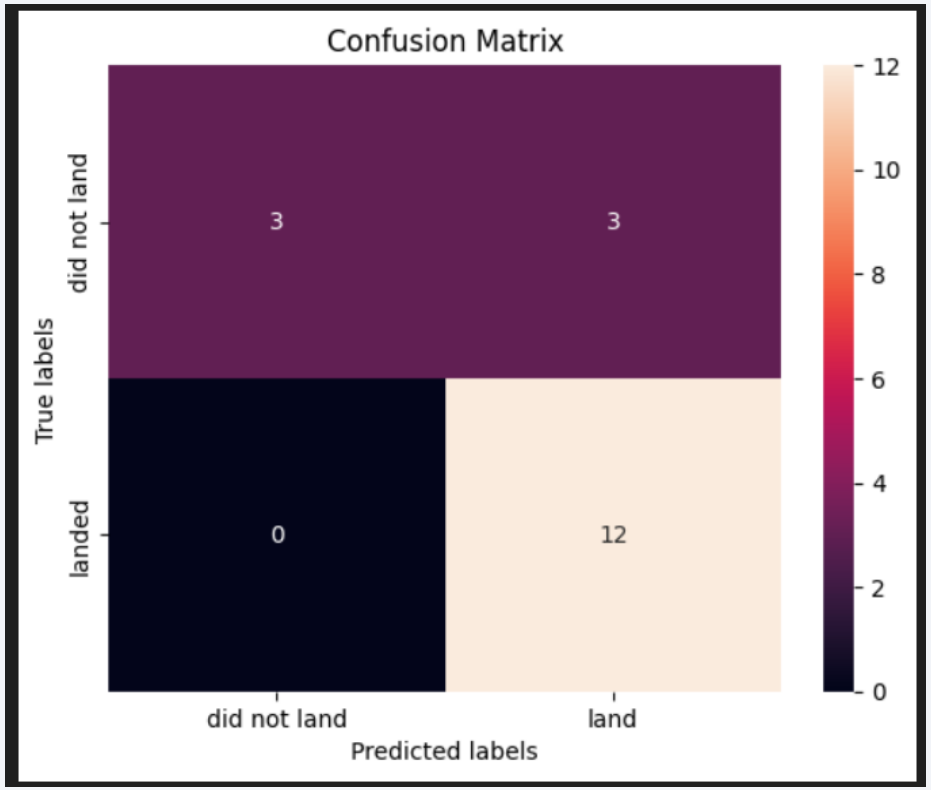
- We could infer that the decision tree classifier is the model with the highest classification accuracy.

```
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': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

Confusion Matrix

- The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives i.e., unsuccessful landing marked as successful landing by the classifier.



Conclusions

- The larger the flight amount at a launch site, greater the success rate at a launch site.
- Launch success rate started to increase from 2013 to 2020.
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- KSC LC-39A had the most successful launches of any sites.
- The Decision tree classifier is the best machine learning algorithm for this task.

Appendix

```
# Read the airline data into pandas dataframe
spacex_df = pd.read_csv("spacex_launch_dash.csv")
max_payload = spacex_df['Payload Mass (kg)'].max()
min_payload = spacex_df['Payload Mass (kg)'].min()

#print(spacex_df['Launch Site'].unique())

# Create a dash application
app = dash.Dash(__name__)

# Create an app layout
app.layout = html.Div(children=[html.H1('SpaceX Launch Records Dashboard',
                                        style={'textAlign': 'center', 'color': '#503D36',
                                               'font-size': 40}),
                                # TASK 1: Add a dropdown list to enable Launch Site selection
                                # The default select value is for ALL sites
                                dcc.Dropdown(id='site-dropdown',
                                             options=[
                                                    {'label': 'All Sites', 'value': 'ALL'},
                                                    {'label': 'CCAFS LC-40', 'value': 'CCAFS LC-40'},
                                                    {'label': 'VAFB SLC-4E', 'value': 'VAFB SLC-4E'},
                                                    {'label': 'KSC LC-39A', 'value': 'KSC LC-39A'},
                                                    {'label': 'CCAFS SLC-40', 'value': 'CCAFS SLC-40'}],
                                             value='ALL',
                                             placeholder="State",
                                             searchable=True
                                            ),
                                html.Br(),
```

```
# TASK 2:
# Add a callback function for `site-dropdown` as input, `success-pie-chart` as output
# Function decorator to specify function input and output
@app.callback(Output(component_id='success-pie-chart', component_property='figure'),
              Input(component_id='site-dropdown', component_property='value'))
def get_pie_chart(entered_site):
    if entered_site == 'ALL':
        fig = px.pie(spacex_df,
                     values='class',
                     names='Launch Site',
                     title='Total Success Launches By Site')
    else:
        filtered_df = spacex_df[spacex_df['Launch Site'] == entered_site]
        filtered_df = filtered_df.groupby('class').count().reset_index()
        fig = px.pie(filtered_df,
                     values='Unnamed: 0',
                     names='class',
                     title='Total Launches for site {}'.format(entered_site))
        # return the outcomes piechart for a selected site
    return fig
```

Thank you!