Winning Space Race with Data Science

Ignacio Enrique García Medel

October 28th 2021

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection methodology
  - Perform data wrangling
  - Perform exploratory data analysis (EDA) using visualization and SQL
  - Perform interactive visual analytics using Folium and Plotly Dash
  - Perform predictive analysis using classification models
- Summary of all results
  - Exploratory data analysis results
  - Interactive dashboards
  - Classification models
  - Predictions analysis

Introduction

- Project background and context
  - The idea of the projects is to predict if the Falcon 9 first stage will land successfully. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch. In this lab, you will collect and make sure the data is in the correct format from an API. The following is an example of a successful and launch.
- Problems you want to find answers
  - What are the factors that determines if land will be successfully.

Section 1

Methodology

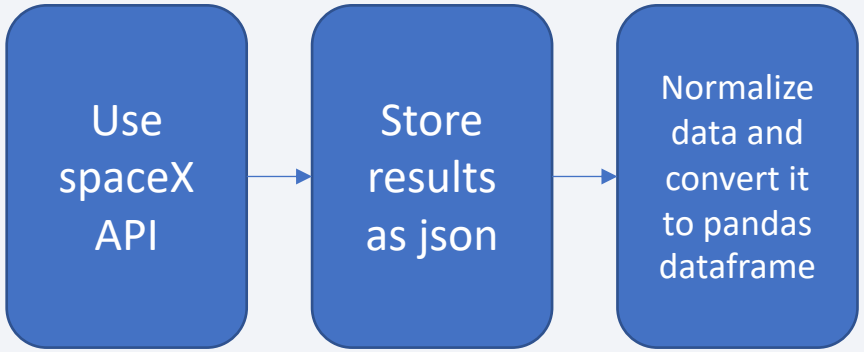
Methodology

Executive Summary

- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- 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

To collect the data, we create a get request from the Spacex API and after that we parsed to a pandas dataset.



Data Collection – SpaceX API

- Github

1. Making the request and getting results.

```
response = requests.get(static_json_url).json()
```

2. Normalize data

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

3. Apply functions to clean data

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

4. Create dataframe

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

5. Filter dataframe to only include falcon 9 launches

```
data_falcon9 = df.loc[df['BoosterVersion']!="Falcon 1"]
```

6. Dealing with missing values

```
# Calculate the mean value of PayloadMass column
mean = data_falcon9['PayloadMass'].mean()
mean
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].fillna(mean)
```

Data Collection - Scraping

1. We perform a HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

```
page = requests.get(static_url)
page.status_code
```

2. Create a BeautifulSoup object from the HTML response.

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

3. Extract tables and find our target table

```
html_tables = soup.find_all('table')
```

```
first_launch_table = html_tables[2]
print(first_launch_table)
```

4. Get column names

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

5. Create a dictionary with keys from previous step

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

Data Collection - Scraping

6. Fill dictionary with launch records extracted from table rows.

```
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])

            # 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)
```

```
            # Booster version
            # TODO: Append the bv into launch_dict with key `Version Booster`
            bv=booster_version(row[1])
            if not(bv):
                bv=row[1].a.string
            print(bv)

            # Launch Site
            # TODO: Append the bv into launch_dict with key `Launch Site`
            launch_site = row[2].a.string
            #print(launch_site)

            # Payload
            # TODO: Append the payload into launch_dict with key `Payload`
            payload = row[3].a.string
            #print(payload)

            # Payload Mass
            # TODO: Append the payload_mass into launch_dict with key `Payload mass`
            payload_mass = get_mass(row[4])
            #print(payload)

            # Orbit
            # TODO: Append the orbit into launch_dict with key `Orbit`
            orbit = row[5].a.string
            #print(orbit)

            # Customer
            # TODO: Append the customer into launch_dict with key `Customer`
            customer = row[6].a.string
            #print(customer)
```

```
            # Launch outcome
            # TODO: Append the launch_outcome into launch_dict with key `Launch outcome`
            launch_outcome = list(row[7].strings)[0]
            #print(launch_outcome)

            # Booster landing
            # TODO: Append the launch_outcome into launch_dict with key `Booster landing`
            booster_landing = landing_status(row[8])
            #print(booster_landing)
```

7. Create a DataFrame from dictionary

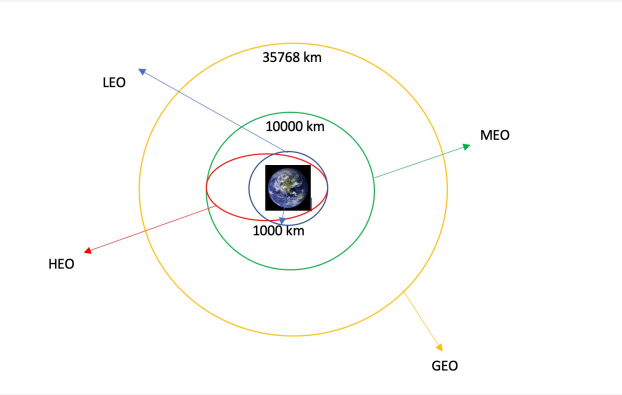
```
df=pd.DataFrame(launch_dict)
```

[Github]

Data Wrangling

In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident; for example, True Ocean means the mission outcome was successfully landed to a specific region of the ocean while False Ocean means the mission outcome was unsuccessfully landed to a specific region of the ocean. True RTLS means the mission outcome was successfully landed to a ground pad False RTLS means the mission outcome was unsuccessfully landed to a ground pad.True ASDS means the mission outcome was successfully landed on a drone ship False ASDS means the mission outcome was unsuccessfully landed on a drone ship.

Each launch aims to an dedicated orbit, and here are some common orbit types:



Calculate the number of launches on each site → Calculate the number and occurrence of each orbit → Calculate the number and occurence of mission outcome per orbit type → Create a landing outcome label from Outcome column

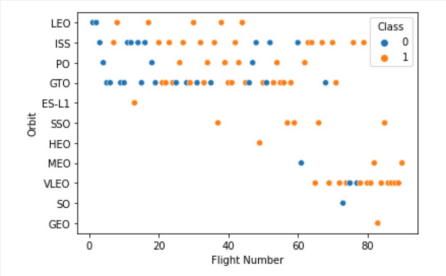
Github

EDA with Data Visualization

We used the following type of charts:

Scatter charts:

- Flight Number and Launch Site
- Payload and Launch Site
- FlightNumber and Orbit type
- Payload and Orbit type

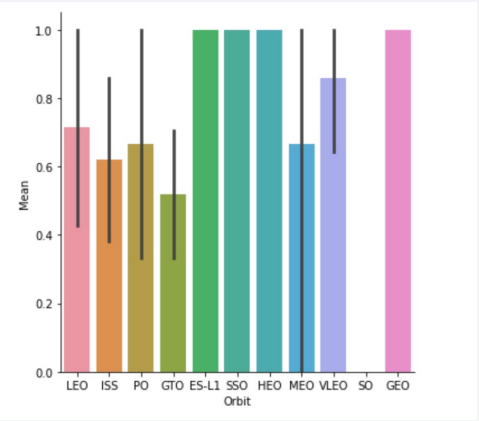


Scatter plots are **used** to plot data points on a horizontal and a vertical axis in the attempt to show how much one variable is affected by another.

Bar charts:

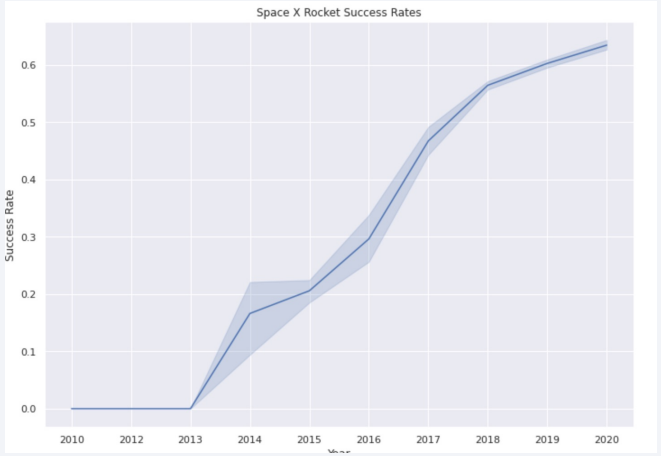
- Success rate of each orbit type

A bar chart is used when **you want to show a distribution of data points or perform a comparison of metric values across different subgroups of your data**.



Line charts:

- Launch success yearly trend



Line graphs are used **to track changes over short and long periods of time**.

[Github]

EDA with SQL

The summarize of the queries used is the following:

- 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 successful landing outcome in ground pad was achieved
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

Github

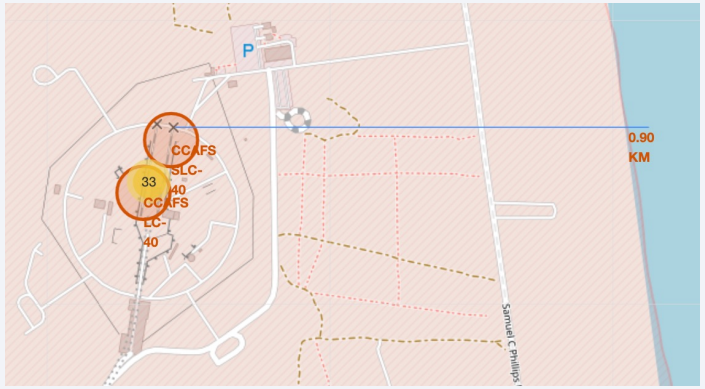
Build an Interactive Map with Folium

To visualize the Launch Data into an interactive map. We took the Latitude and Longitude Coordinates at each launch site and added a Circle Marker around each launch site with a label of the name of the launch site.

We assigned the dataframe launch _outcomes(failures, success) to classes 0 and 1 with Green and Red markers on the map in a MarkerCluster()

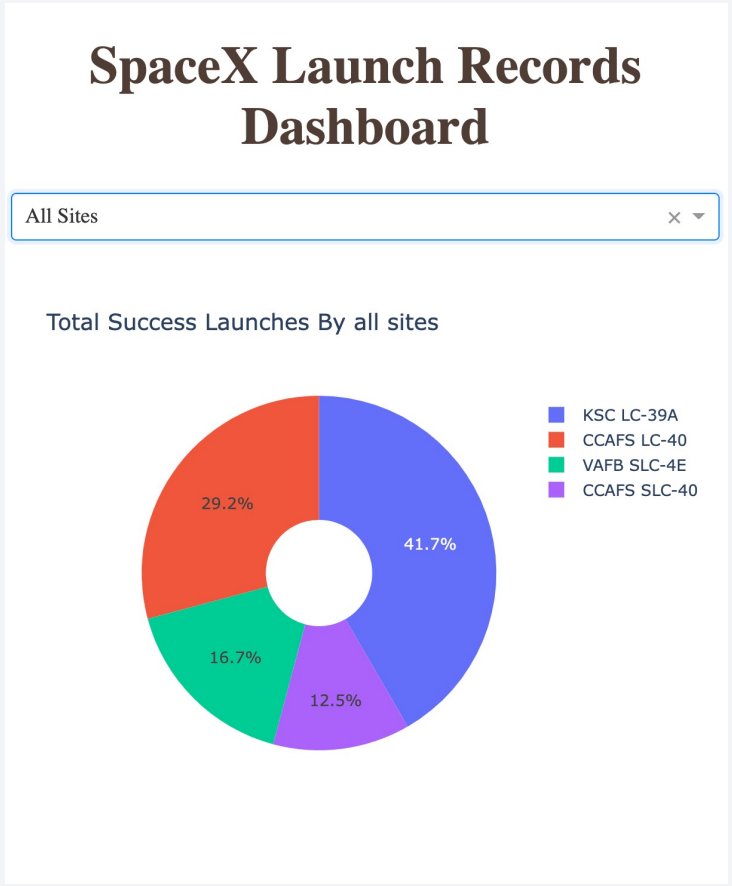
Using Haversine's formula we calculated the distance from the Launch Site to various landmarks to find various trends about what is around the Launch Site to measure patterns. Lines are drawn on the map to measure distance to landmarks

	Launch Site	Lat	Long	class	marker_color
46	KSC LC-39A	28.573255	-80.646895	1	green
47	KSC LC-39A	28.573255	-80.646895	1	green
48	KSC LC-39A	28.573255	-80.646895	1	green
49	CCAFS SLC-40	28.563197	-80.576820	1	green
50	CCAFS SLC-40	28.563197	-80.576820	1	green
51	CCAFS SLC-40	28.563197	-80.576820	0	red
52	CCAFS SLC-40	28.563197	-80.576820	0	red
53	CCAFS SLC-40	28.563197	-80.576820	0	red
54	CCAFS SLC-40	28.563197	-80.576820	1	green
55	CCAFS SLC-40	28.563197	-80.576820	0	red



Github

Build a Dashboard with Plotly Dash



We use pie charts to see the comparassion of the success rates of all the sites. Also,we used a scatter graph to see the relationship of the payload mass with the class(if succeed or fails).

Predictive Analysis (Classification)

- Building the model
  - Load dataset into numpy and pandas.
  - Standardize the data
  - Split data into training and test data
  - Create a model object and fit the object to find the best parameters from the dictionary parameters

  Note: This steps are for all the different models. (Logistic regression, SVM, Decision Tree and K nearest)

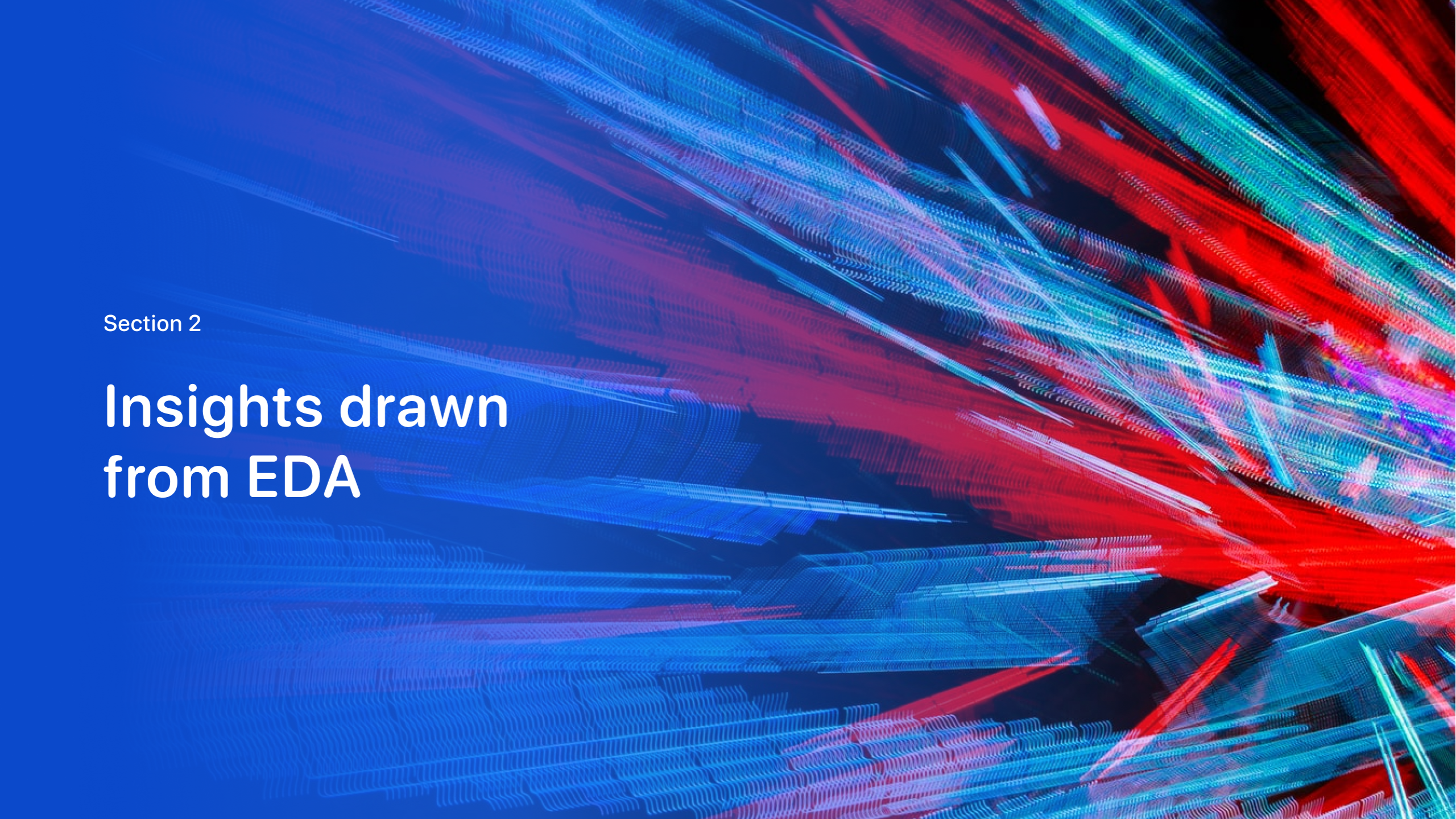
- Model evaluation
  - Get the best parameters and accuracy of the model by using best_score_ and best_params_ functions.
  - Get the accuracy of the model using test data.
  - Create and examine confusion matrix.

- Model improvement
  - Treat missing and Outlier values

- Finding the best performing classification model
  - To find the best performing classification we compared all the results from best_score_ function of all the models and select the best performing.

Results

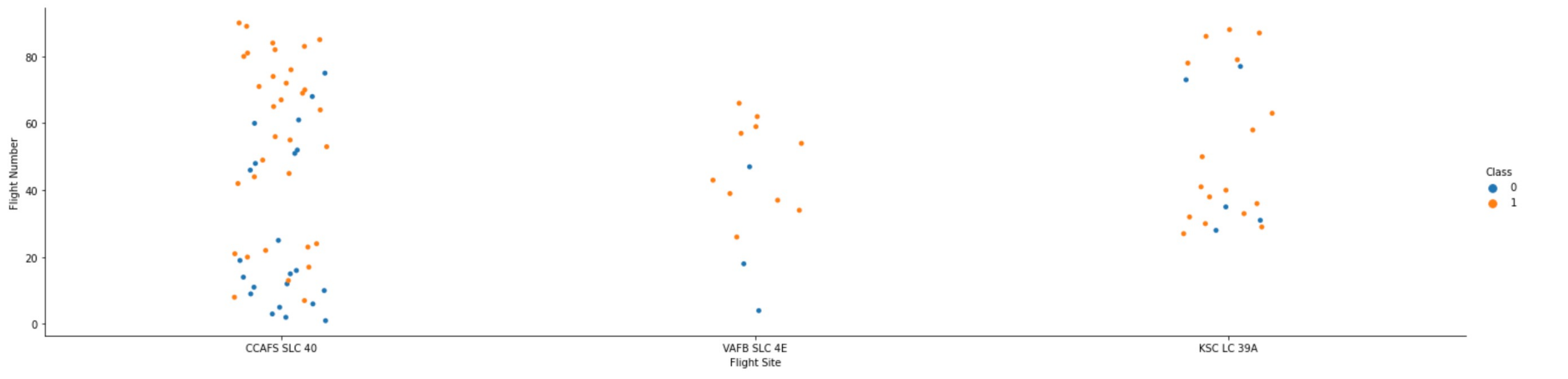
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results



Section 2

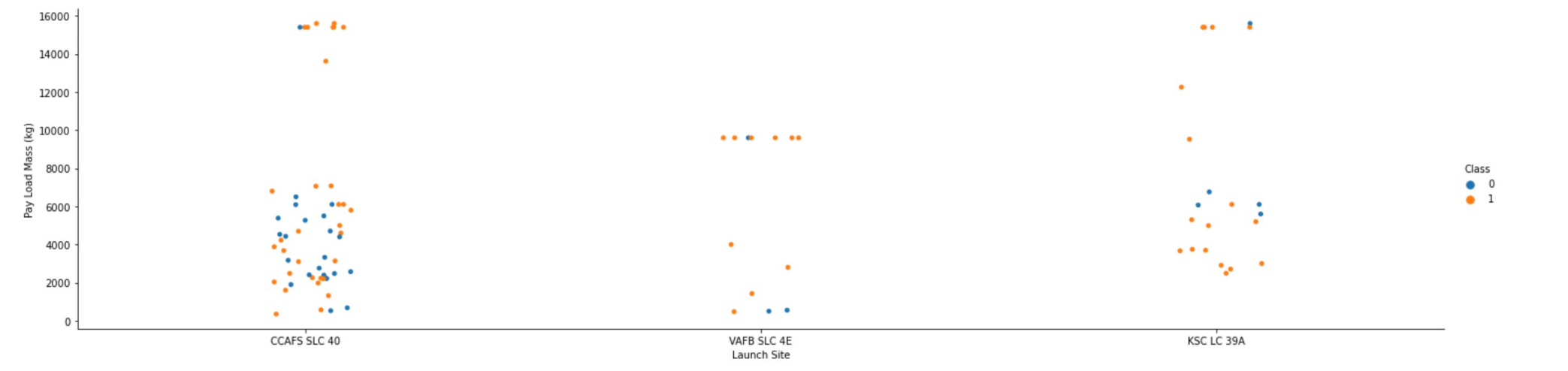
Insights drawn from EDA

Flight Number vs. Launch Site



The success rate is higher when there are more flights at a launch site.

Payload vs. Launch Site



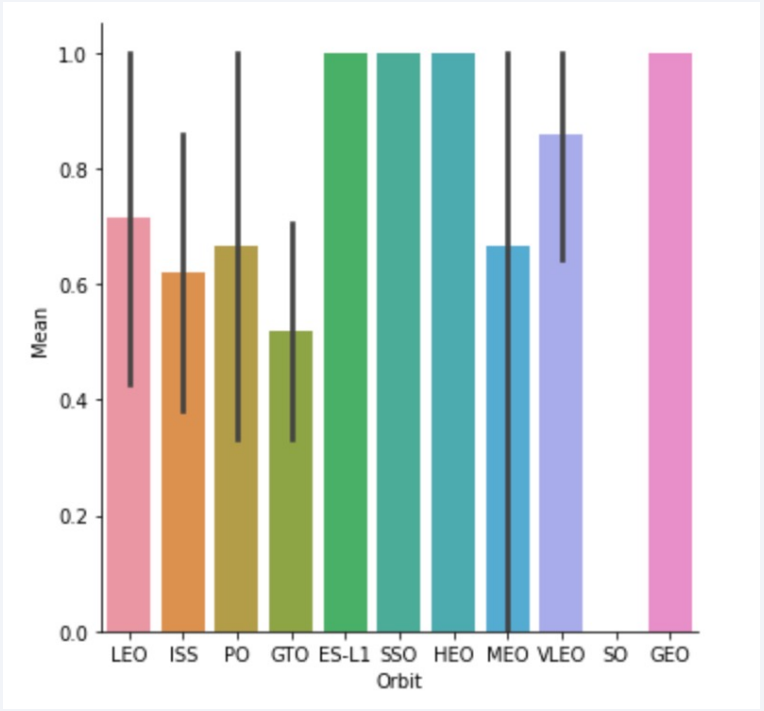
The greater payload mass the higher success rate for CCAFS SLC 40

For the VAFB-SLC launchsite there are no rockets launched for heavy payload mass(greater than 10000)

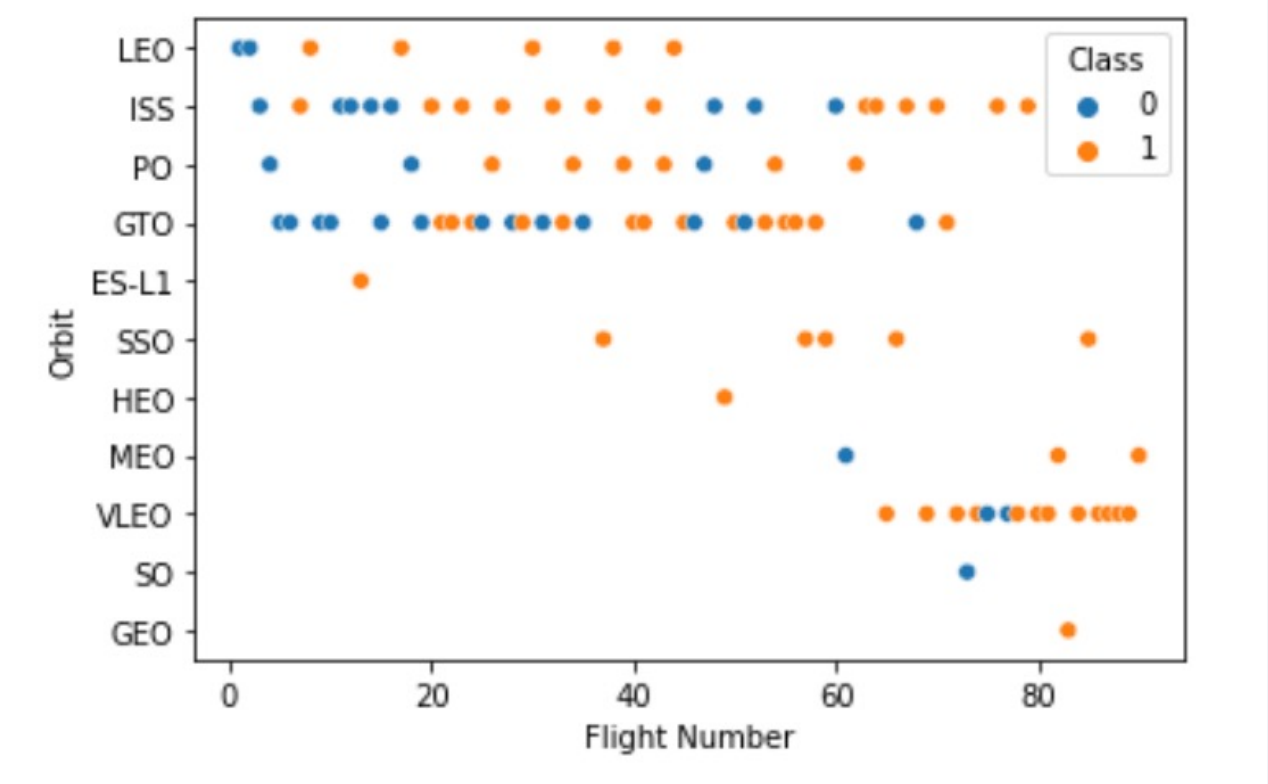
Success Rate vs. Orbit Type

	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCount	Longitude	Latitude	Class
Orbit											
ES-L1	13.000000	570.000000	1.000000	1.000000	0.000000	1.000000	1.000000	0.000000	-80.577366	28.561857	1.000000
GEO	83.000000	6104.959412	2.000000	1.000000	1.000000	1.000000	5.000000	2.000000	-80.577366	28.561857	1.000000
GTO	35.037037	5011.994444	1.407407	0.629630	0.333333	0.629630	3.037037	0.962963	-80.586229	28.577258	0.518519
HEO	49.000000	350.000000	1.000000	1.000000	0.000000	1.000000	4.000000	1.000000	-80.577366	28.561857	1.000000
ISS	39.142857	3279.938095	1.238095	0.809524	0.238095	0.857143	3.142857	1.285714	-80.583697	28.572857	0.619048
LEO	20.000000	3882.839748	1.000000	0.571429	0.000000	0.714286	2.142857	0.428571	-80.584963	28.575058	0.714286
MEO	77.666667	3987.000000	1.000000	0.666667	0.000000	0.666667	5.000000	0.666667	-80.577366	28.561857	0.666667
PO	36.333333	7583.666667	1.333333	0.888889	0.333333	0.777778	3.222222	1.555556	-120.610829	34.632093	0.666667
SO	73.000000	6104.959412	4.000000	0.000000	1.000000	0.000000	5.000000	3.000000	-80.603956	28.608058	0.000000
SSO	60.800000	2060.000000	2.400000	1.000000	0.800000	1.000000	4.600000	3.200000	-112.604136	33.418046	1.000000
VLEO	78.928571	15315.714286	3.928571	1.000000	1.000000	1.000000	5.000000	3.928571	-80.586862	28.578358	0.857143

ES-L1, SSO, HEO and GEO has the best success rate

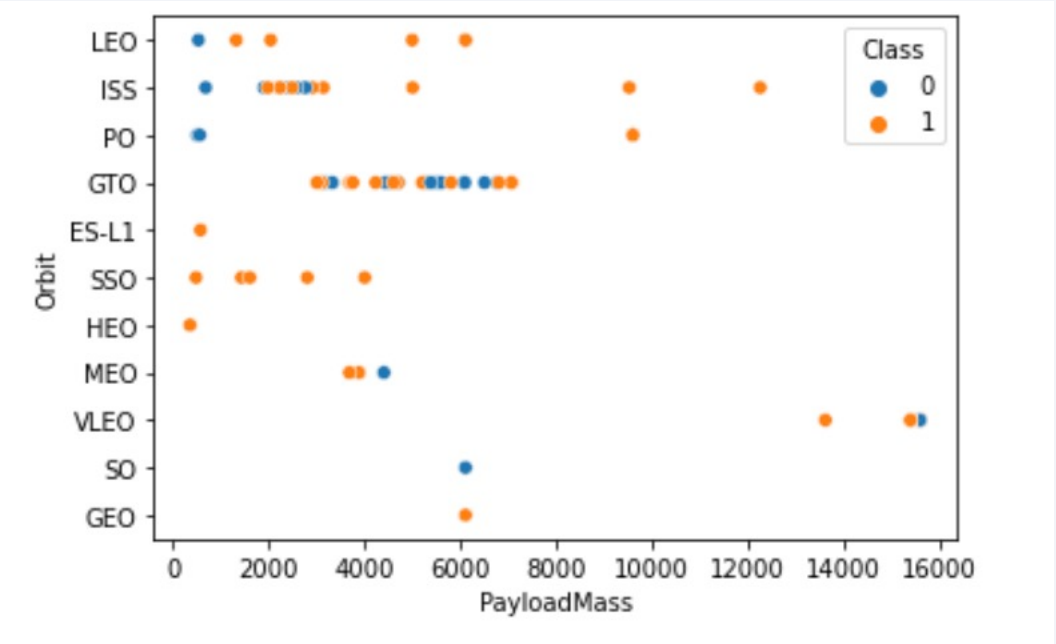


Flight Number vs. Orbit Type



We see that in the LEO orbit the success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

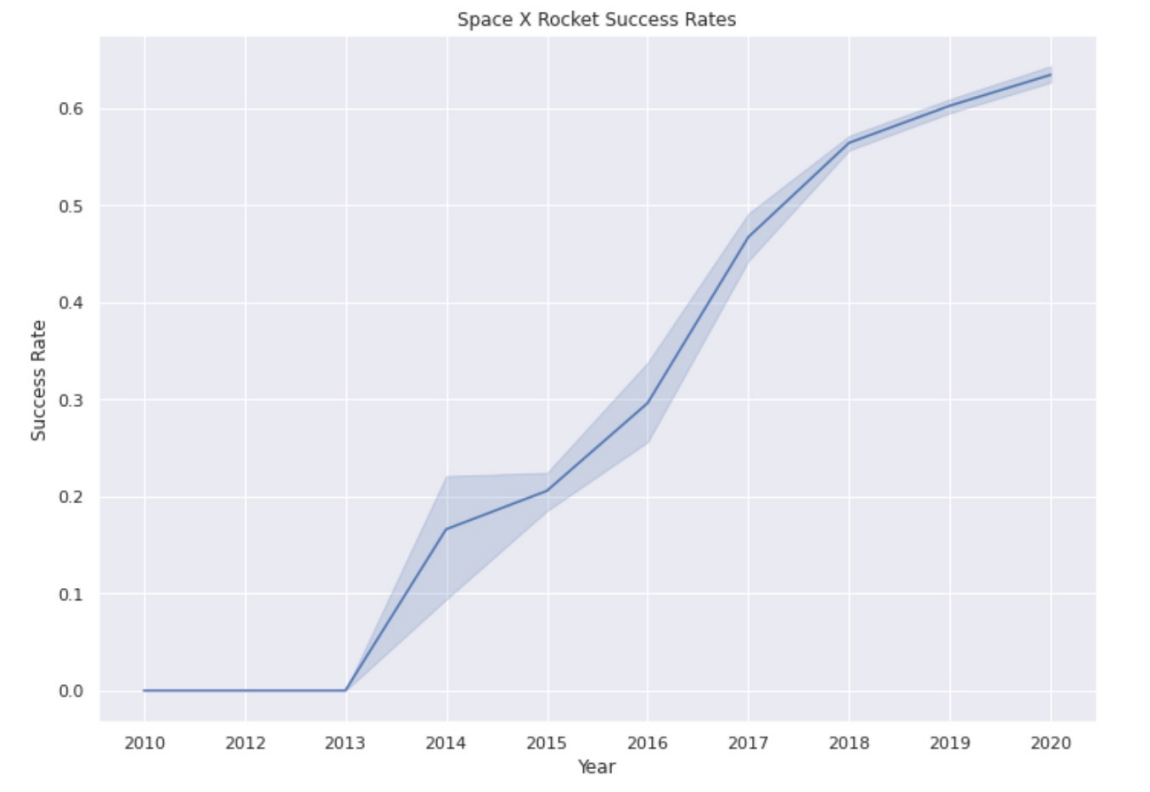
Payload vs. Orbit Type



We see that with heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.

However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.

Launch Success Yearly Trend



We observe that the success rate since 2013 kept increasing till 2020

All Launch Site Names

```
: %sql select distinct launch_site from SPACEXTBL
```

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

Query explanation

SQL DISTINCT clause is used to remove the duplicates columns from the result set.
It means the result will be the unique sites names.

Launch Site Names Begin with 'CCA'

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

```
In [7]: %sql select * from SPACEXTBL where launch_site like 'CCA%' limit 5
```

```
 * ibm_db_sa://mtv60411:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

Out[7]:

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Query explanation

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column. In this case we used it to get the launch site names begin with 'CCA’.

Total Payload Mass

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

```
In [21]: %sql select sum(payload_mass__kg_) totalPayloadMassKG from SPACEXTBL where customer = 'NASA (CRS)'
```

```
 * ibm_db_sa://mtv60411:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

Out[21]:

totalpayloadmasskg
45596

Query explanation

The SUM() function returns the total sum of a numeric column. In this case we used it to get the total payload mass carried by boosters launched by NASA(CRS).

Average Payload Mass by F9 v1.1

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

```
In [22]: %sql select avg(payload_mass__kg_) avg from spacextbl where booster_version = 'F9 v1.1'
 * ibm_db_sa://mtv60411:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

Out[22]:

AVG
2928

Query explanation

The AVG() function returns the average value of a numeric column. We used it to get the average payload mass carried by booster version F9 v1.1.

First Successful Ground Landing Date

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

Hint:Use min function

In [26]:
```
%sql select min(date) fsdate from spacextbl where landing__outcome = 'Success (drone ship)'
```

```
 * ibm_db_sa://mtv60411:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

Out[26]:

fsdate
2016-04-08

Query explanation

The MIN() function returns the smallest value of the selected column. In this case is used to get the date of the first successful landing outcome in ground pad.

Successful Drone Ship Landing with Payload between 4000 and 6000

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

In [28]:
```
%sql select booster_version from spacextbl where landing__outcome = 'Success (drone ship)' and (payload_mass__kg_ > 4000 and p
ayload_mass__kg_ < 6000)
```

```
 * ibm_db_sa://mtv60411:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

Out[28]:

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

Query explanation

The WHERE clause filters the dataset to Landing_outcome = Success (drone ship) and the other condition executed by the statement "and" specifies the filters for when payload mass is greater than 4000 and less than 6000.

Total Number of Successful and Failure Mission Outcomes

```
%sql select sum(case when mission_outcome like '%Success%' then 1 else 0 END) as succeed, sum(case when mission_outcome like
'%Failure%' then 1 else 0 END) as failure from spacextbl
```

```
 * ibm_db_sa://mtv60411:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

succeed	failure
100	1

Query explanation

We used subqueries to join the results. First, we used the SUM function to sum the rows that are a success mission outcome and then we do the same for the ones that failed.

Boosters Carried Maximum Payload

```
%sql select booster_version, payload_mass__kg_ as max_payload_mass from spacextbl where payload_mass__kg_ = (select max(payloa
d_mass__kg_) from spacextbl)
```

booster_version	max_payload_mass
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

Query explanation

We select booster version and payload mass with the condition that their payload mass matches with the maximum payload mass of the records. To get the maximum payload mass of the dataset we use the “MAX” function.

2015 Launch Records

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

```
%sql select booster_version, launch_site, date from spacextbl where landing__outcome like '%Failure (drone ship)%' and year(DA
TE) = year('2015-01-01')
```

```
 * ibm_db_sa://mtv60411:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

booster_version	launch_site	DATE
F9 v1.1 B1012	CCAFS LC-40	2015-01-10
F9 v1.1 B1015	CCAFS LC-40	2015-04-14

Query explanation

To List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015 we used 2 conditions. The first is using the “LIKE” operator to get the rows that their landing outcome is failed and the second condition is to just get the records that are on 2015.

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

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

```
%sql select count(landing__outcome) as "Landing outcomes between 2010-06-04 and 2017-03-20" from spacextbl where landing__outcome like '%Failure (drone ship)%' or landing__outcome like '%Success (ground pad)%' and (date > '2010-06-04') AND (date < '2017-03-20')
```

```
 * ibm_db_sa://mtv60411:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

Landing outcomes between 2010-06-04 and 2017-03-20
8

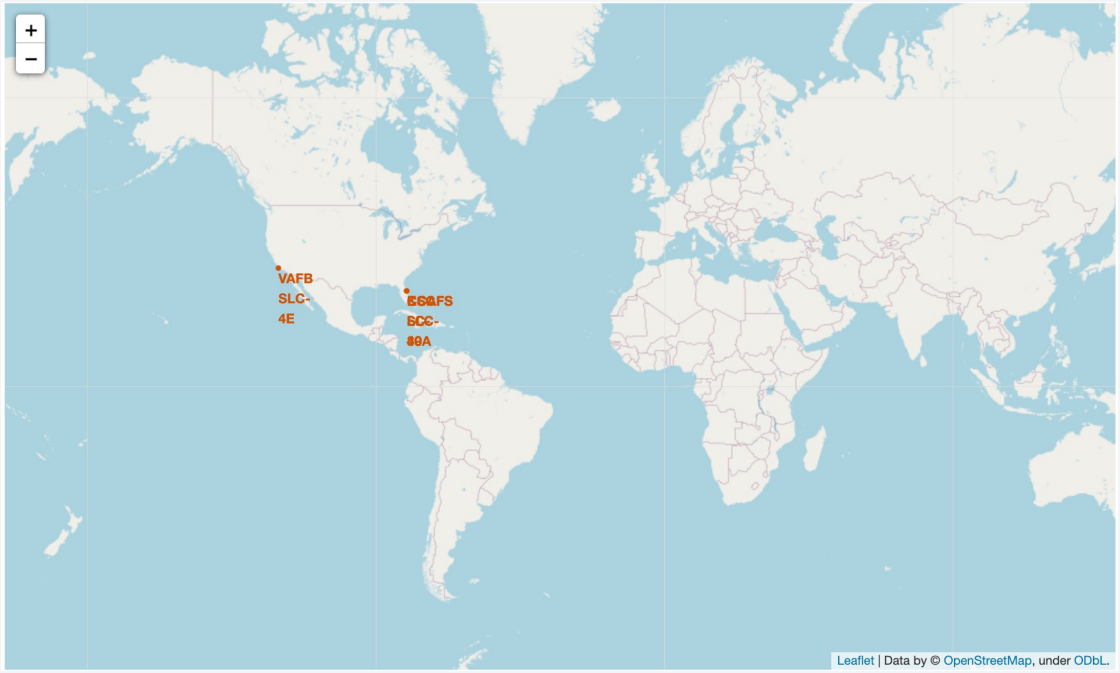
Query explanation

We use the COUNT() function to count the totat of the landing outcomes between the date and we use WHERE, AND & OR functions to filter the information.

Section 4

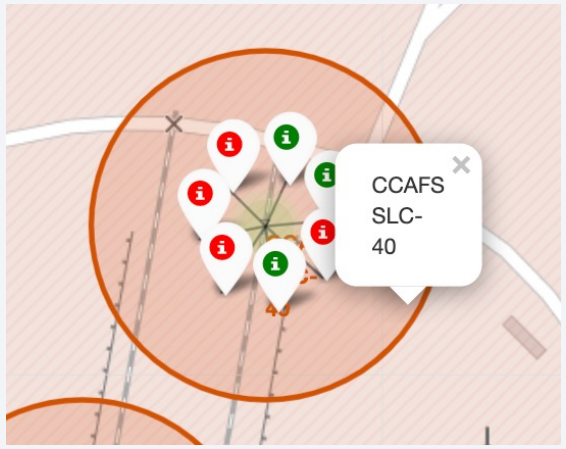
Launch Sites Proximities Analysis

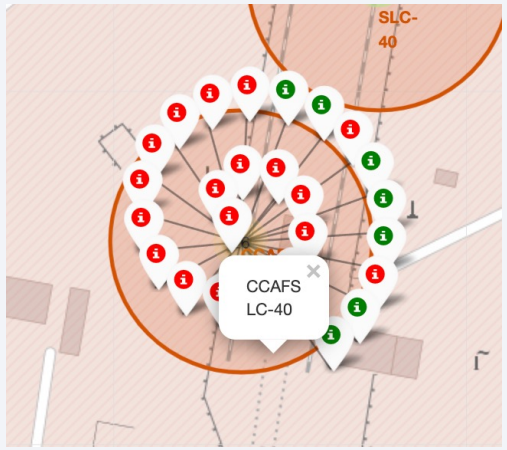
All launch sites

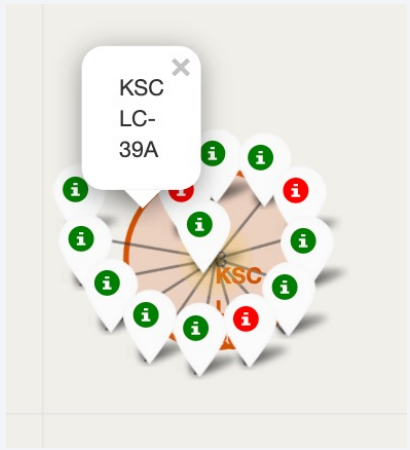


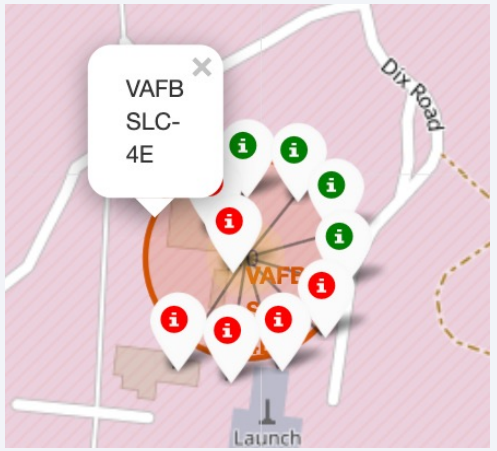
We can see in the image that all the locations of SpaceX are on the United States of America but in the opposite coasts.

Success/failed launches for each site





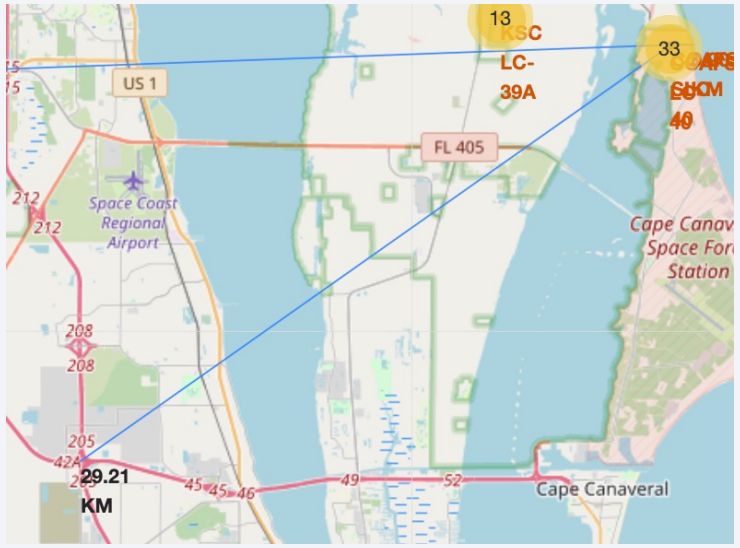




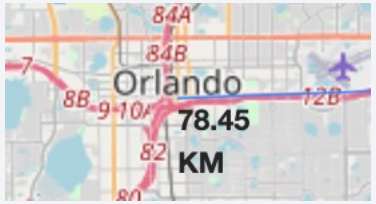
The Green flags represent the succesfull launches and the red flag the failed. We can see that the location that fails less is KSC LC-39A.

Distances between a launch site to its proximities

Distance to coast



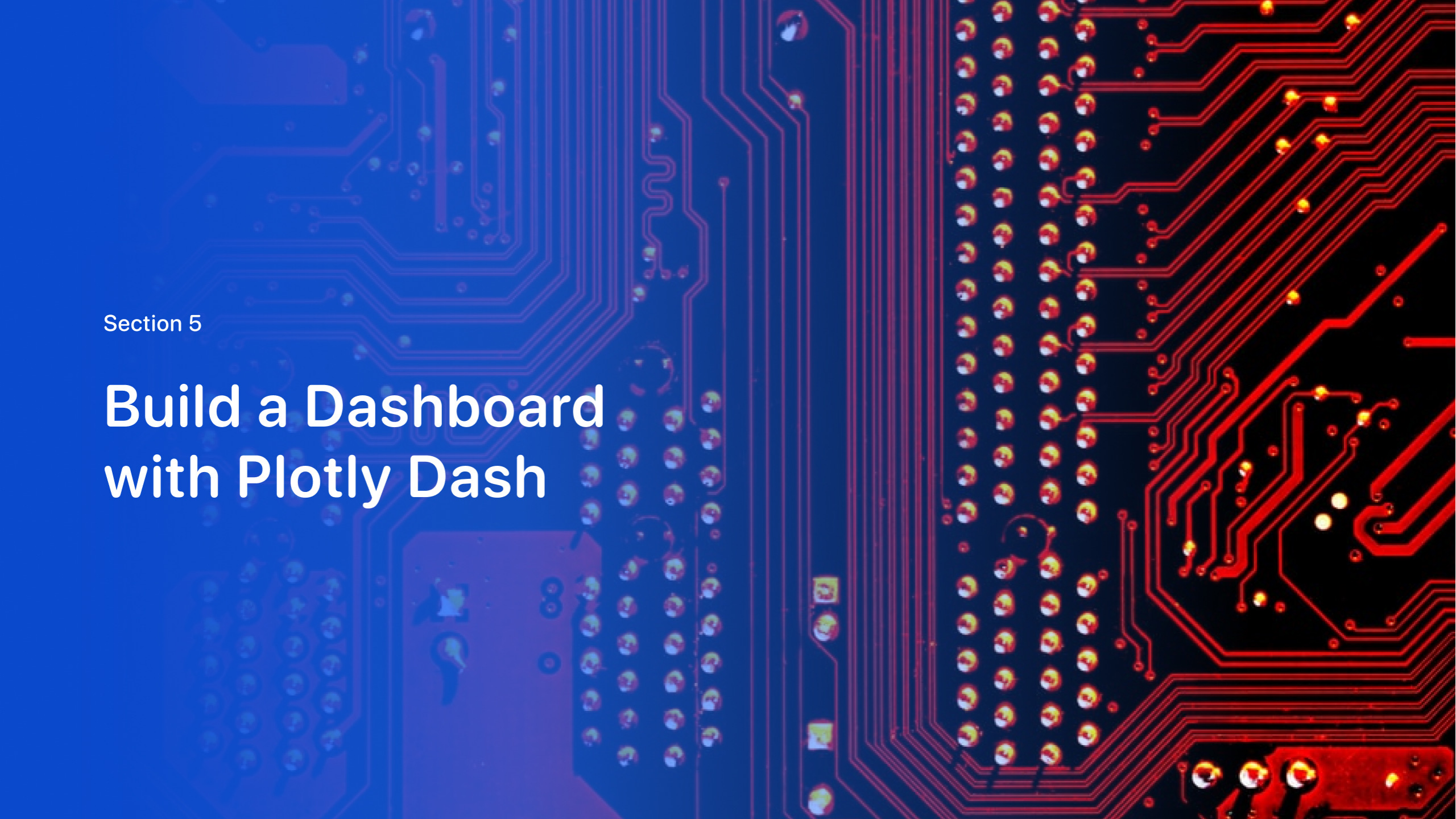
Distance to highway



Distance to city

After you plot distance lines to the proximities, you can answer the following questions easily:

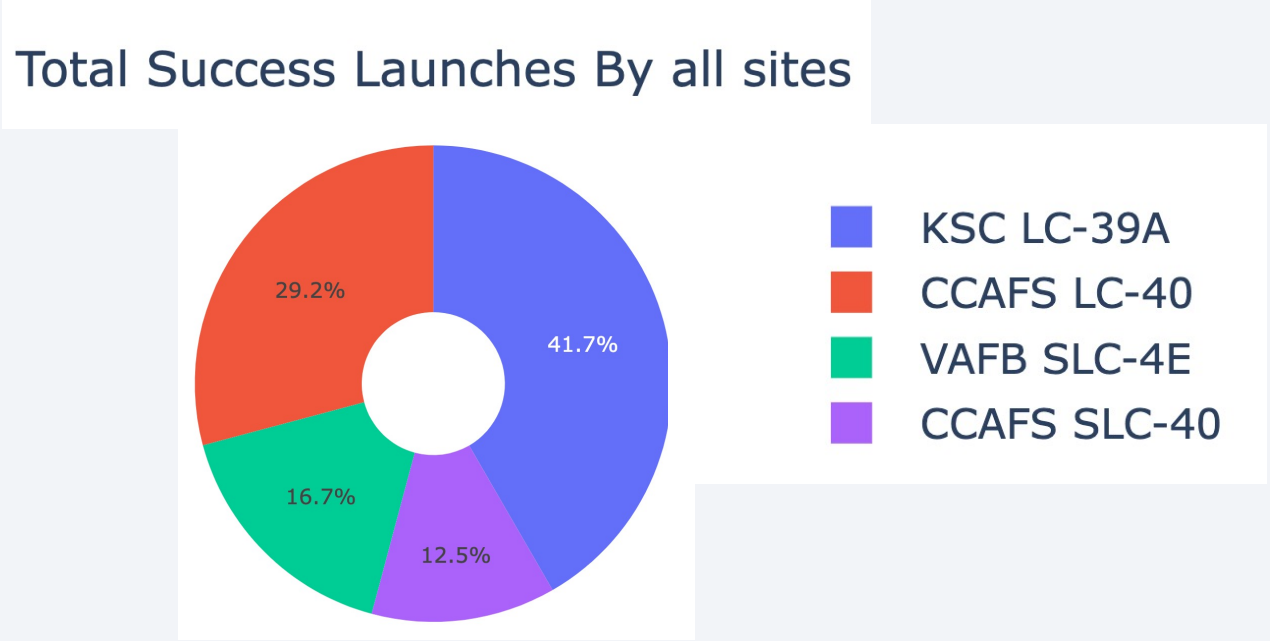
- Are launch sites in close proximity to railways?no
- Are launch sites in close proximity to highways? no
- Are launch sites in close proximity to coastline? yes
- Do launch sites keep certain distance away from cities? yes



Section 5

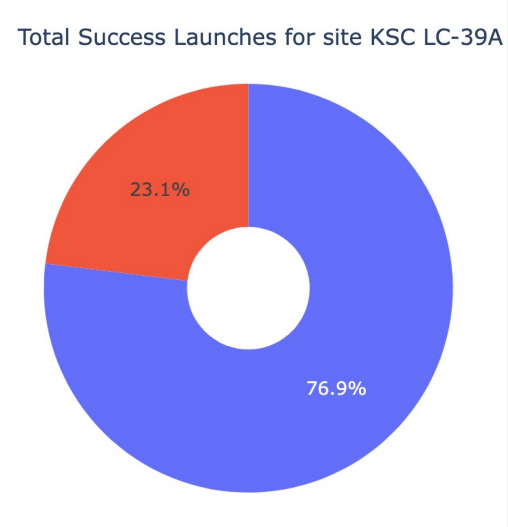
Build a Dashboard with Plotly Dash

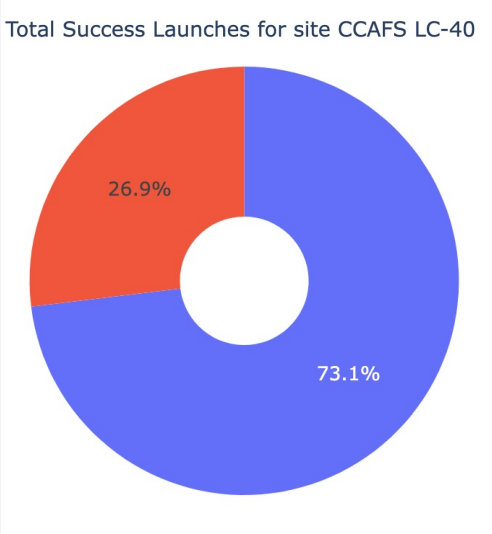
Launch success count for all sites

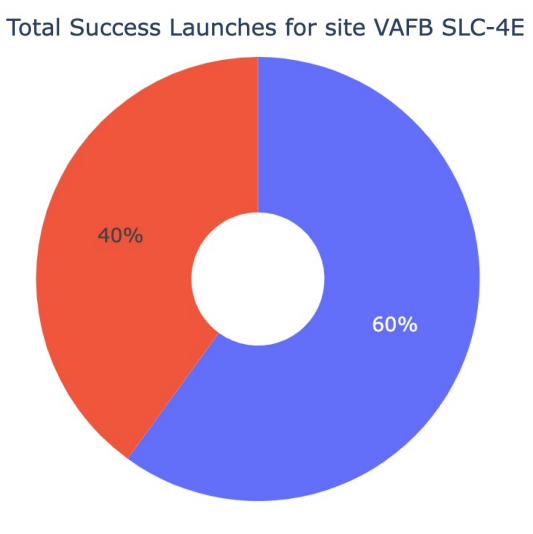


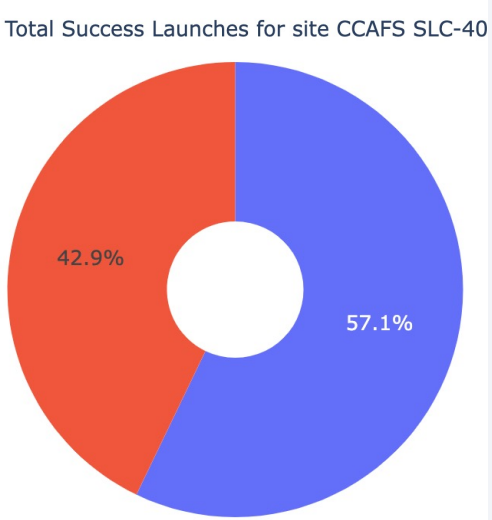
With this pie chart we can notice that the most rate of succesfull launches are from the KSC LC-39A site.

Highest launch success ratio





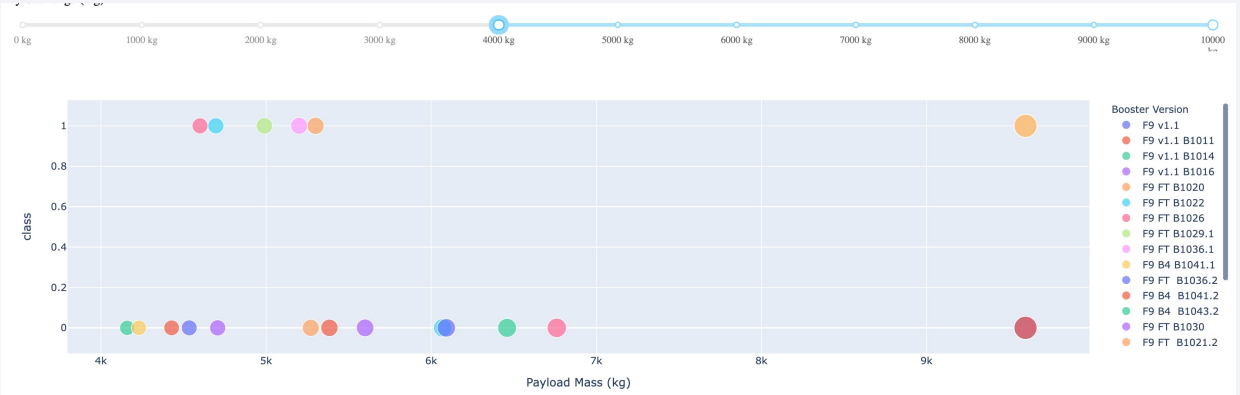




KSC LC-39A has a 76.9% success rate and 23.1% failure rate. Is the best location

Payload vs. Launch Outcome

Success rate is better for low weighted payloads.

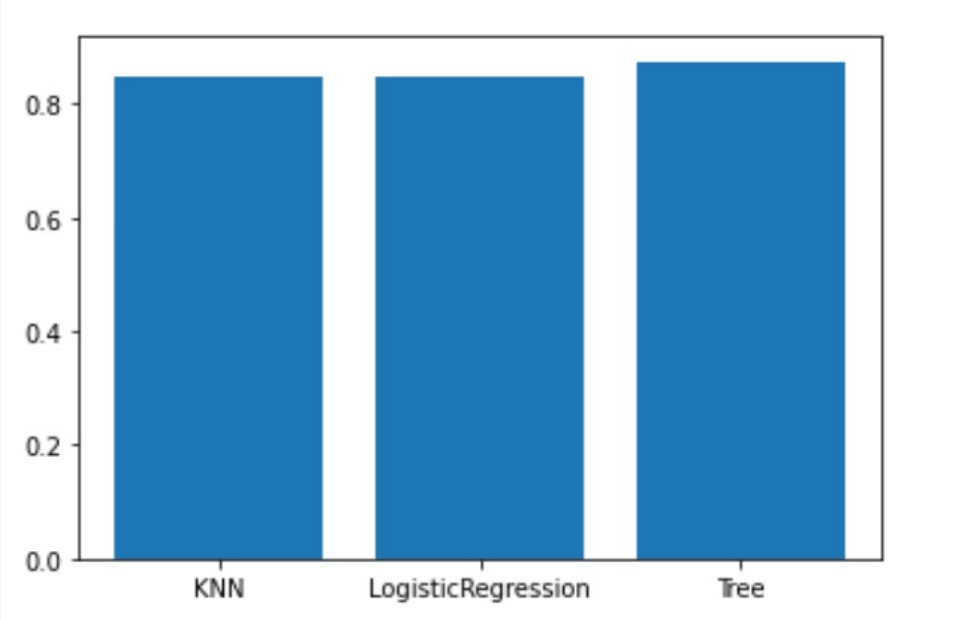


Section 6

Predictive Analysis (Classification)

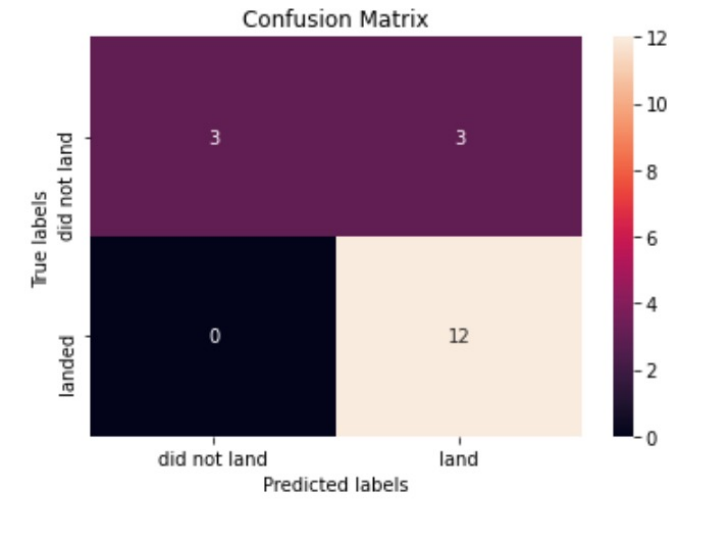
Classification Accuracy

We can see that all of the models have a similar accuracy but the highest accuracy is for the model Tree with 87.5%.




```
{'LogisticRegression': 0.8464285714285713, 'Tree': 0.875, 'KNN': 0.8482142857142858}
```

Confusion Matrix for Tree Model



We can see that the model is pretty good, but it contains some false positives. It is saying that there were 3 lanches marked as land when te reality is that it didn’t land.

Conclusions

- The success rate is higher when there are more flights at a launch site.
- ES-L1, SSO, HEO and GEO has the best success rate
- The success rate is directly proportional to time, so it means the success rate will increase.
- The most rate of successful launches are from the KSC LC-39A site.
- Success rate is better for low weighted payloads.
- The Tree classifier algorithm is the best for this project because it has a score of 87.5%

Thank you!