IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Deepali Lalchandani
19/05/2022

Outline

URL of GitHub Repository

- 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
- Building an Interactive map with Folium
- Building a Dashboard wit Plotly Dash
- Predictive Analysis (Classification)

Summary of Results

- Exploratory Data Analysis results
- Interactive analytics in screenshots
- Predictive Analytics results from Machine Learning Lab

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 a launch. This information can be used if an alternate company wants to bid against space X for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

Problems we 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

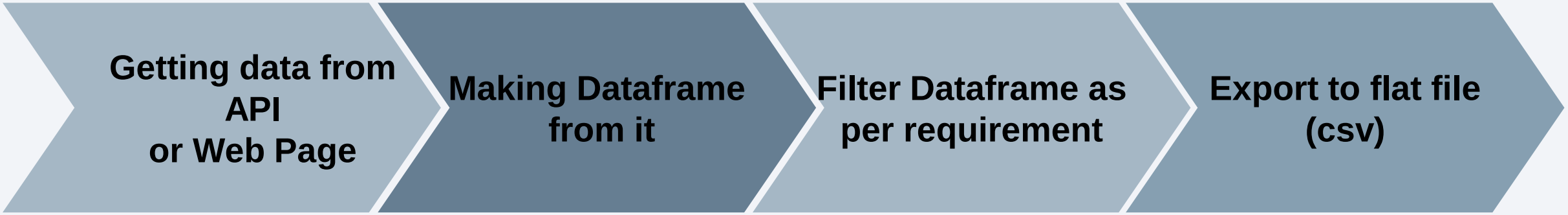
Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX API and web scraping from Wikipedia.
- Perform data wrangling
  - One hot encoding data fields for machine learning and dropping irrelevant columns (Transforming data for machine learning)
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Build, tune, evaluate classification models

Data Collection

Data collection is the process of gathering and measuring information on targeted variables in an established system, which then enables one to answer relevant questions and evaluate outcomes.

We collected data about launches, including information about rocket used, payload delivered, launch specifications, landing specifications, and landing outcome



Data Collection – SpaceX API

[GitHub URL to Notebook]

- Getting response from API
- Converting response to .json file
- Apply custom functions to clean data
- Assign list to dictionary and create dataframe
- Filter dataframe and export to csv file

1
```
spacex_url="https://api.spacexdata.com/v4/launches/past"

response = requests.get(spacex_url)
```

2
```
# Use json_normalize meethod to convert the json result into a dataframe
response = requests.get(static_json_url).json()
data = pd.json_normalize(response)
```

3
```
# Call getBoosterVersion
getBoosterVersion(data)
```
```
# Call getLaunchSite
getLaunchSite(data)
```
```
# Call getPayloadData
getPayloadData(data)
```
```
# Call getCoreData
getCoreData(data)
```

4
```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}

# Create a data from launch_dict
launch_df = pd.DataFrame.from_dict(launch_dict)
```

5
```
data_falcon9 = launch_df[launch_df['BoosterVersion']!="Falcon 1"]

data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection - Scraping

GitHub URL to Notebook

- Getting response from HTML
- Creating BeautifulSoup object
- Finding tables
- Getting column names
- Creating dictionary
- Appending data to keys
- Converting dictionary to dataframe
- Dataframe to csv

1
```
response = requests.get(static_url).text
```

2
```
soup = BeautifulSoup(response, 'html.parser')
```

3
```
html_tables = soup.find_all("table")
print(html_tables)
```

4
```
column_names = []

# Apply find_all() function with `th` element on fir
temp = soup.find_all('th')
# Iterate each th element and apply the provided ext
for x in range(len(temp)):
    try:
        name = extract_column_from_header(temp[x])
        if (name is not None and len(name)>0):
            column_names.append(name)
    except:
        pass
```

5
```
launch_dict= dict.fromkeys(column_names)

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

# Let's initial the launch_dict with each
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']=[]
```

6
```
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()
```

7
```
df=pd.DataFrame(launch_dict)
```

8
```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Data Wrangling

GitHub URL to Notebook

Data wrangling is the process of cleaning and unifying messy and complex data for easy access and Exploratory Data Analysis (EDA).

- Calculate number of launches at each site
- Calculate number and occurrence of each orbit
- Calculate the number and occurrence of mission outcome per orbit type
- Create a landing outcome label from Outcome column
- Export to csv file

1

```
df["LaunchSite"].value_counts()
```

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

2

```
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
Name: Orbit, dtype: int64
```

3

```
landing_outcomes = df["Outcome"].value_counts()
landing_outcomes
```

```
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
Name: Outcome, dtype: int64
```

4

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

5

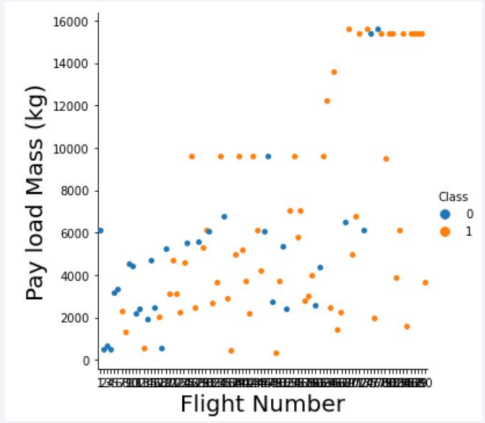
```
df.to_csv("dataset_part_2.csv", index=False)
```

EDA with Data Visualization

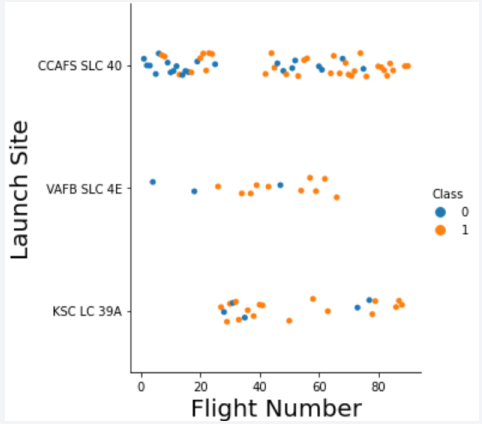
GitHub URL to Notebook

Exploratory data analysis is an approach of analyzing data sets to summarize their main characteristics, using statistical graphics and other data visualization methods

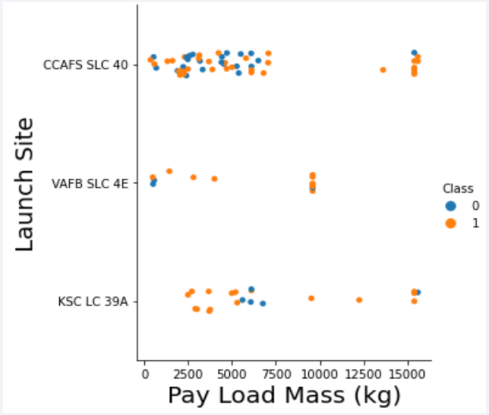
Scatter Graphs drawn



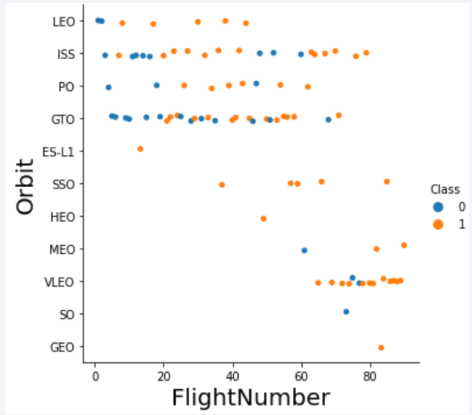
Flight Number VS. Payload Mass



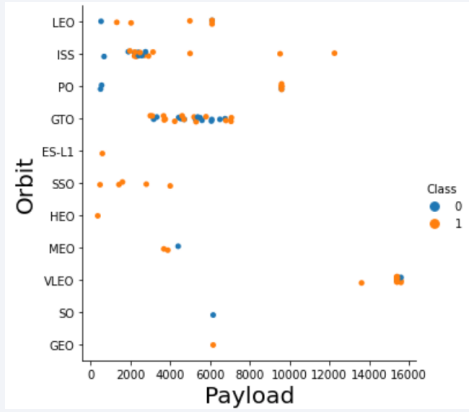
Flight Number VS. Launch Site



Payload VS. Launch Site



Orbit VS. Flight Number

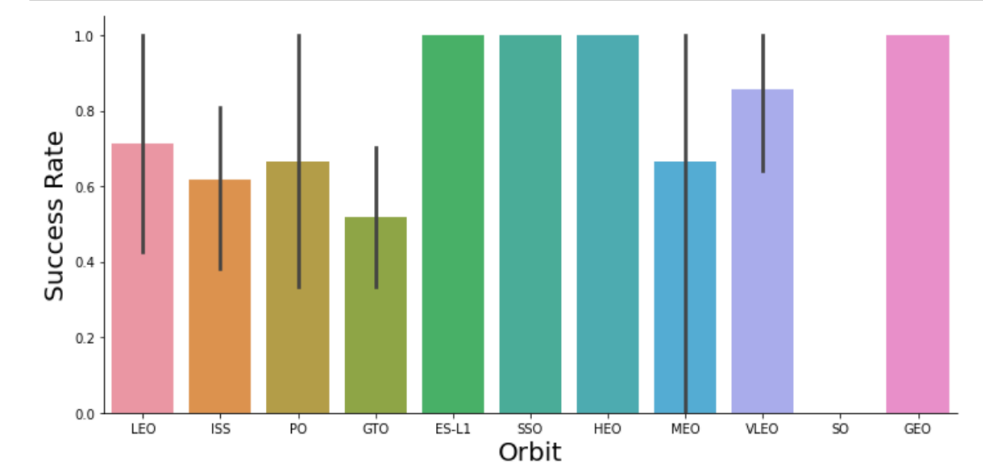


Payload VS. Orbit Type

EDA with Data Visualization

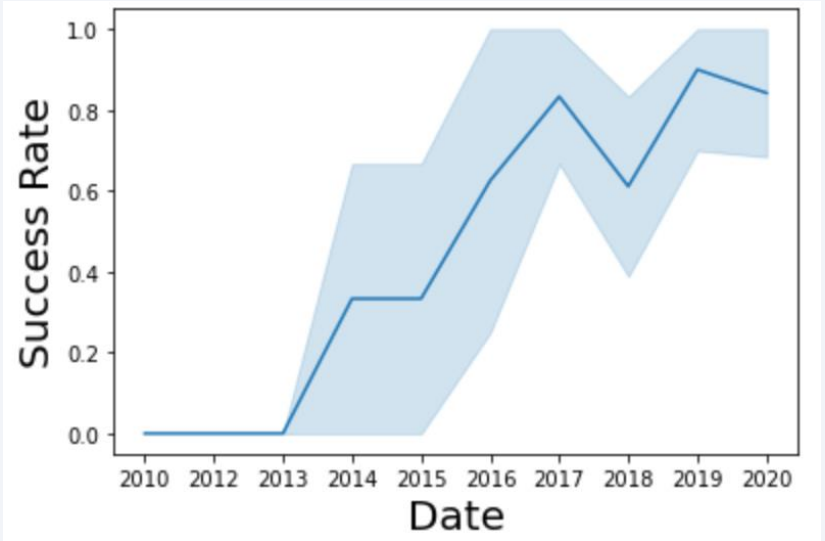
GitHub URL to Notebook

Bar Graph drawn



Success rate Vs. Orbit type

Line Graph drawn



Launch Success yearly trend

EDA with SQL

GitHub URL to Notebook

We used IBM's Db2 on cloud, which is fully managed SQL Database provided as a service. Following SQL queries were performed to gather information about the dataset:

- Displaying the names of the unique launch sites in the space mission
- Displaying 5 records where launch sites begin with the string 'KSC'
- Displaying the total payload mass carried by boosters launched by NASA (CRS)
- Displaying average payload mass carried by booster version F9 v1.1
- Listing the date where the successful landing outcome in drone ship was achieved.
- Listing the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000
- Listing the total number of successful and failure mission outcomes
- Listing the names of the booster_versions which have carried the maximum payload mass.
- Listing the records which will display the month names, successful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017
- Ranking the count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order.

Build an Interactive Map with Folium

GitHub URL to Notebook

Folium makes it easy to visualize data that is being manipulated in Python on an interactive leaflet map. Following map objects were added:

- Map marker – To mark the launch sites on map using their latitude and longitude coordinates
- Circle marker – To add a highlighted circle area
- Icon marker – To create an icon as a text label
- Marker cluster – To simplify the map containing many markers having the same coordinate. We assigned Red and Green colours to markers showing failure and success of the launch.
- PolyLine – To create a line between launch site and closest coastline, city, railway line

We calculated the distances between a launch site to its proximities using Haversine's formula . We answered some questions for instance:

-Are launch sites near railways, highways and coastlines?

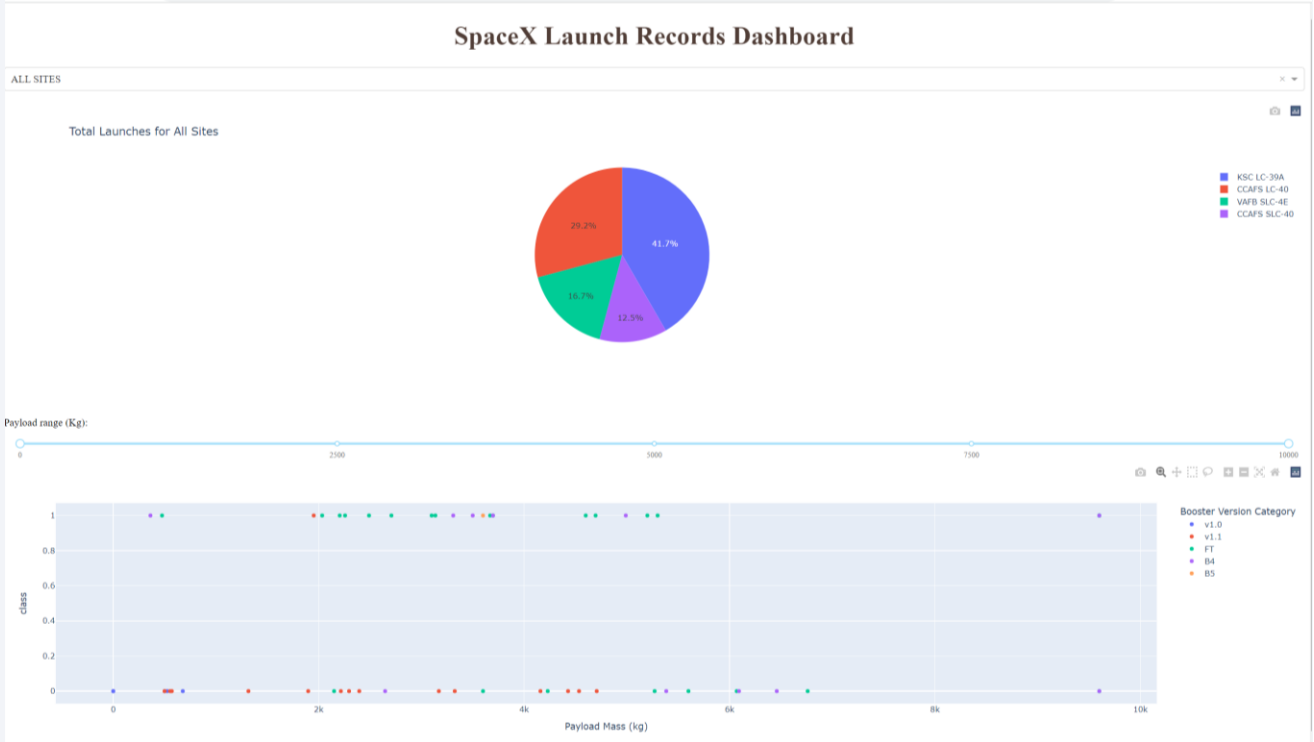
-Do launch sites keep certain distance away from cities?

Build a Dashboard with Plotly Dash

GitHub URL to Python file

Dash is a python framework created by Plotly for creating interactive web applications.
We created a dashboard application with following components:

- Dropdown list – To enable launch site selection (default is all sites)
- Pie chart - to show the total successful launches count for all sites. If a specific launch site is selected, it shows the Success vs. Failed counts for the site
- Slider – To select payload range
- Scatter chart – To show the correlation between payload and launch success



Predictive Analysis (Classification)

GitHub URL to Notebook

- Load our dataset into NumPy and Pandas
- Transform Data
- Split our data into training and test data sets
- Check how many test samples we have
- Decide which type of machine learning algorithms we want to use
- Set our parameters and algorithms to GridSearchCV
- Fit our datasets into the GridSearchCV objects and train our dataset.

- Check accuracy for each model
- Get tuned hyperparameters for each type of algorithms
- Plot Confusion Matrix

- Feature Engineering
- Algorithm Tuning

- The model with the best accuracy score is the best performing classification model

Results

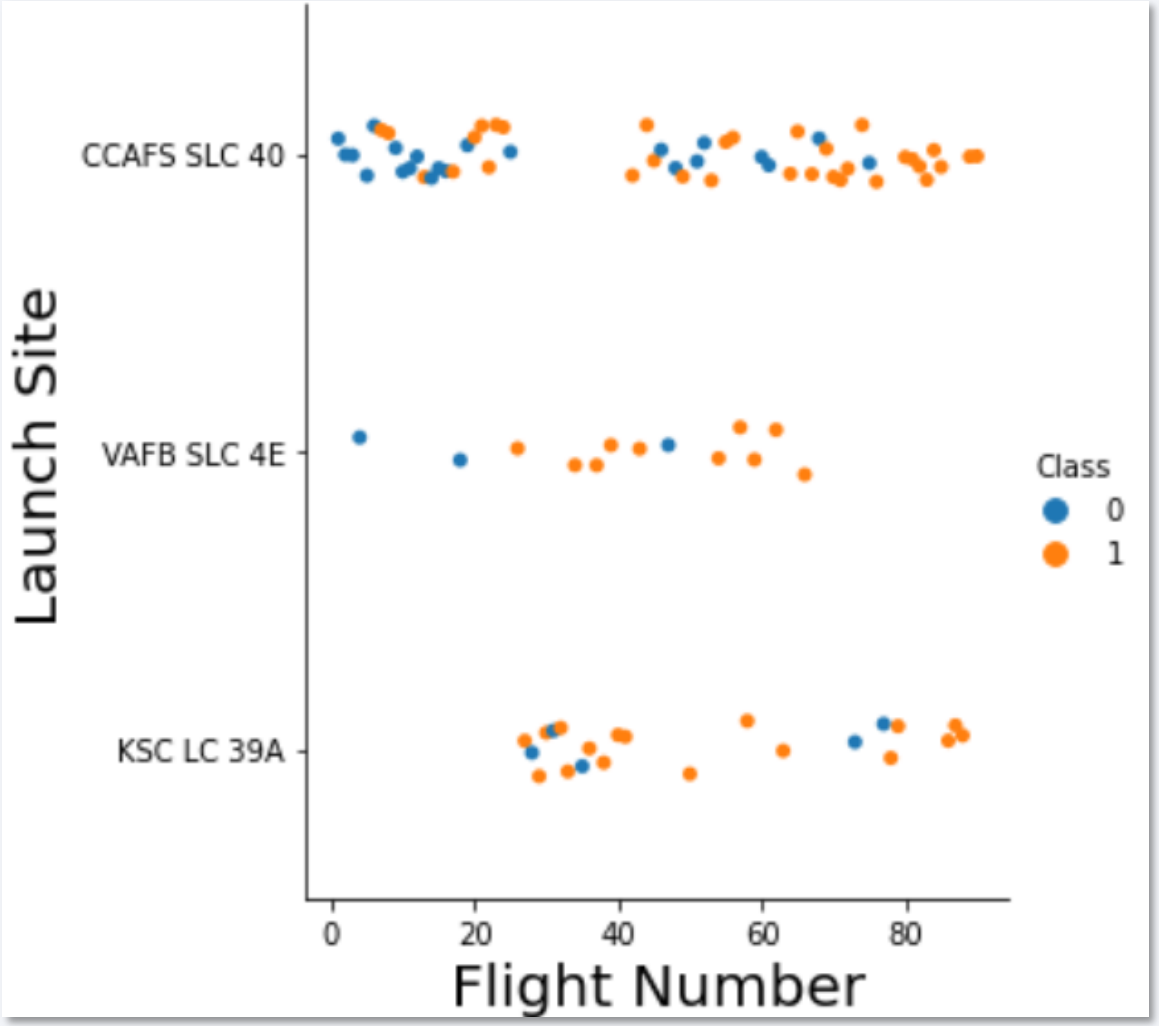
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA with Visualization

Flight Number vs. Launch Site

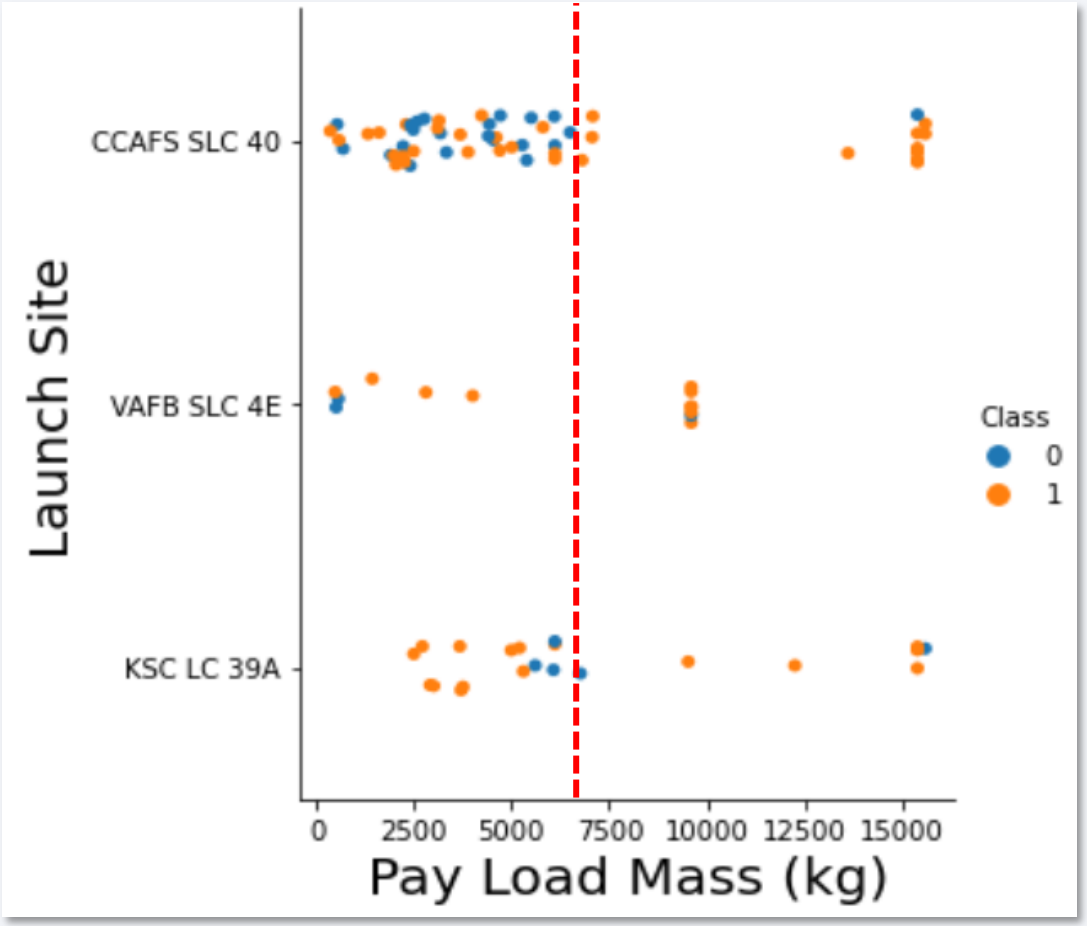
This scatter plot shows that the larger is the number of flights at a launch site, the greater is the success rate of the launch.



Payload vs. Launch Site

The plot shows once the pay load mass is greater than around 7000kg, the probability of the success rate is highly increased.

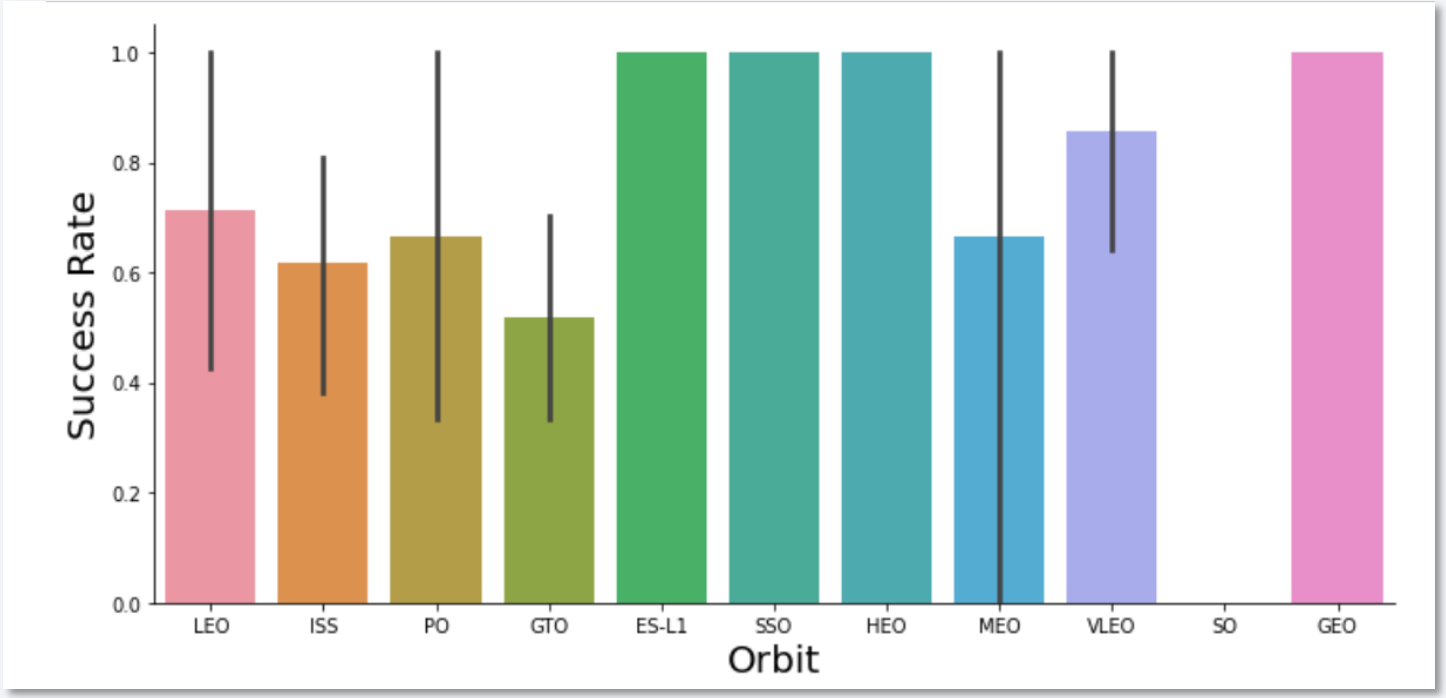
However, there is no indication of launch site being dependent on pay load mass for the success rate of the launch.



Success Rate vs. Orbit Type

The bar chart shows that the orbits ES-L1, SSO, HEO and GEO have 100% success rate while orbit SO has 0% success rate.

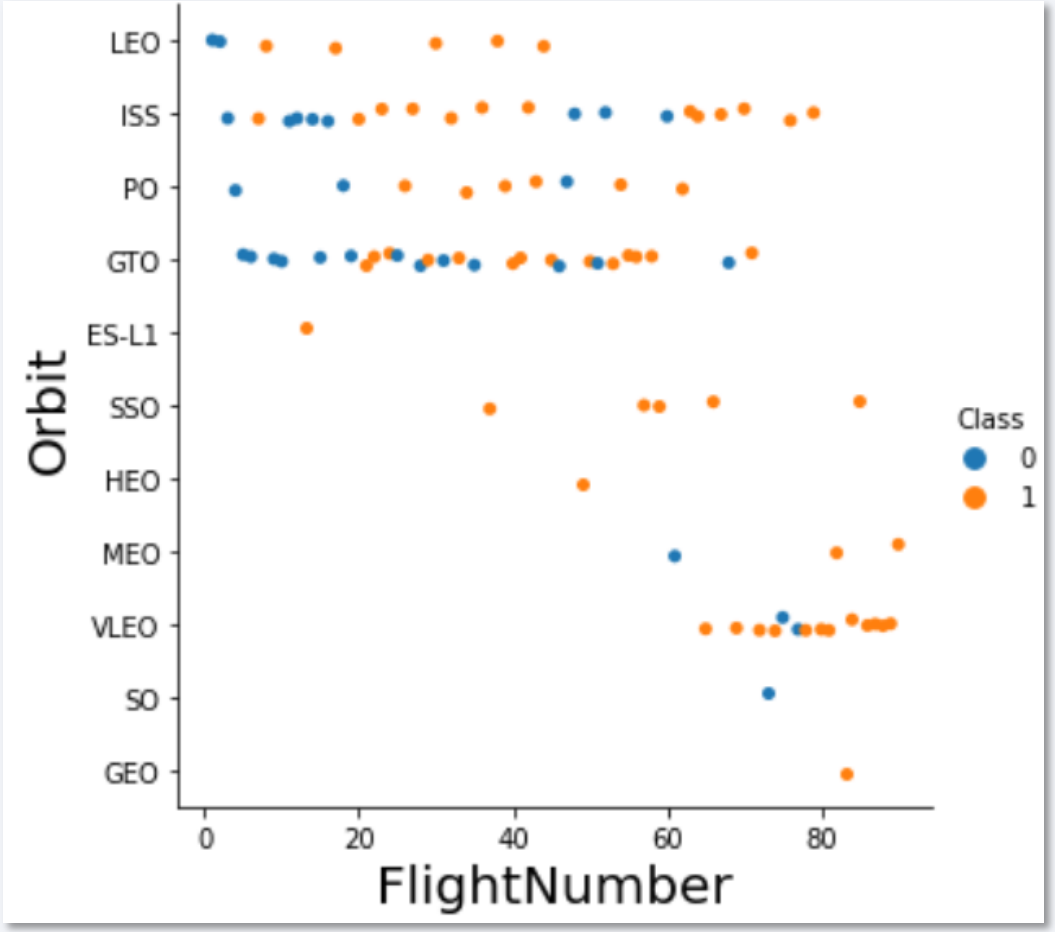
However, deeper analysis shows that some of this orbits have only 1 occurrence such as GEO, SO, HEO and ES-L1 which mean data is insufficient to draw any conclusion.



Flight Number vs. Orbit Type

This scatter plot shows that generally, the larger the flight number on each orbits, the greater the success rate. This correlation is strongest for LEO orbit and weakest for GTO orbit.

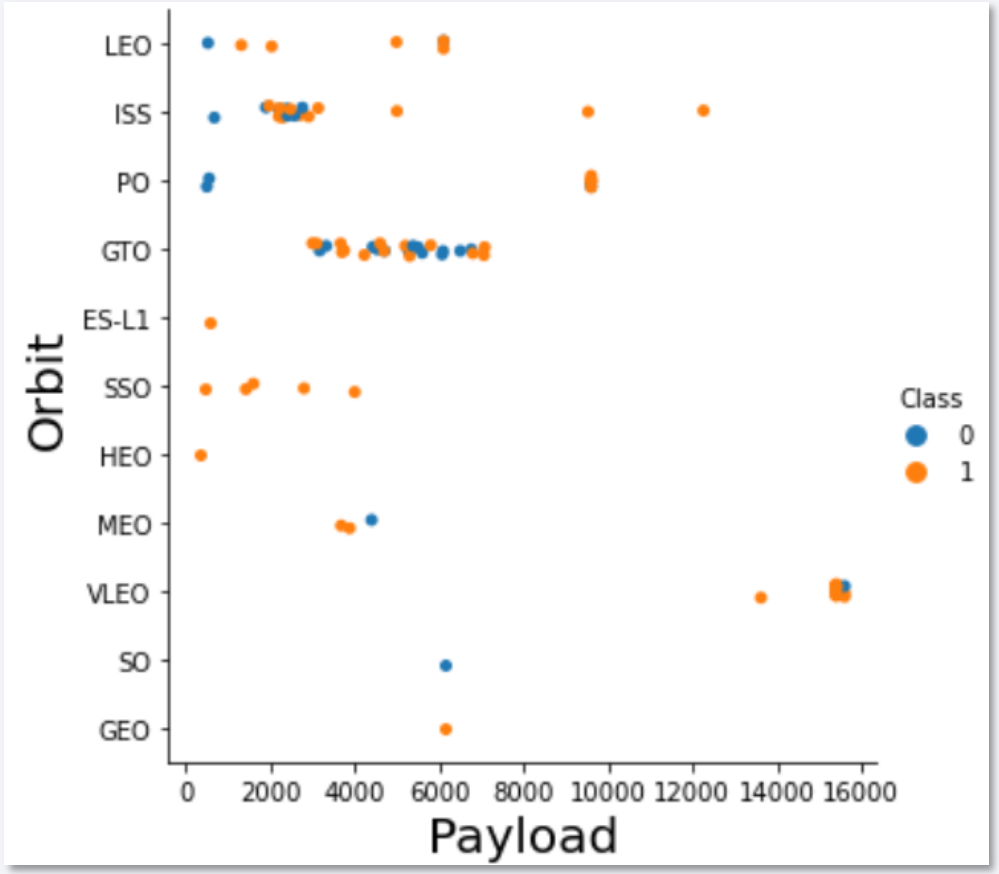
However, this cant be said about orbits that have only 1 or few occurrences.



Payload vs. Orbit Type

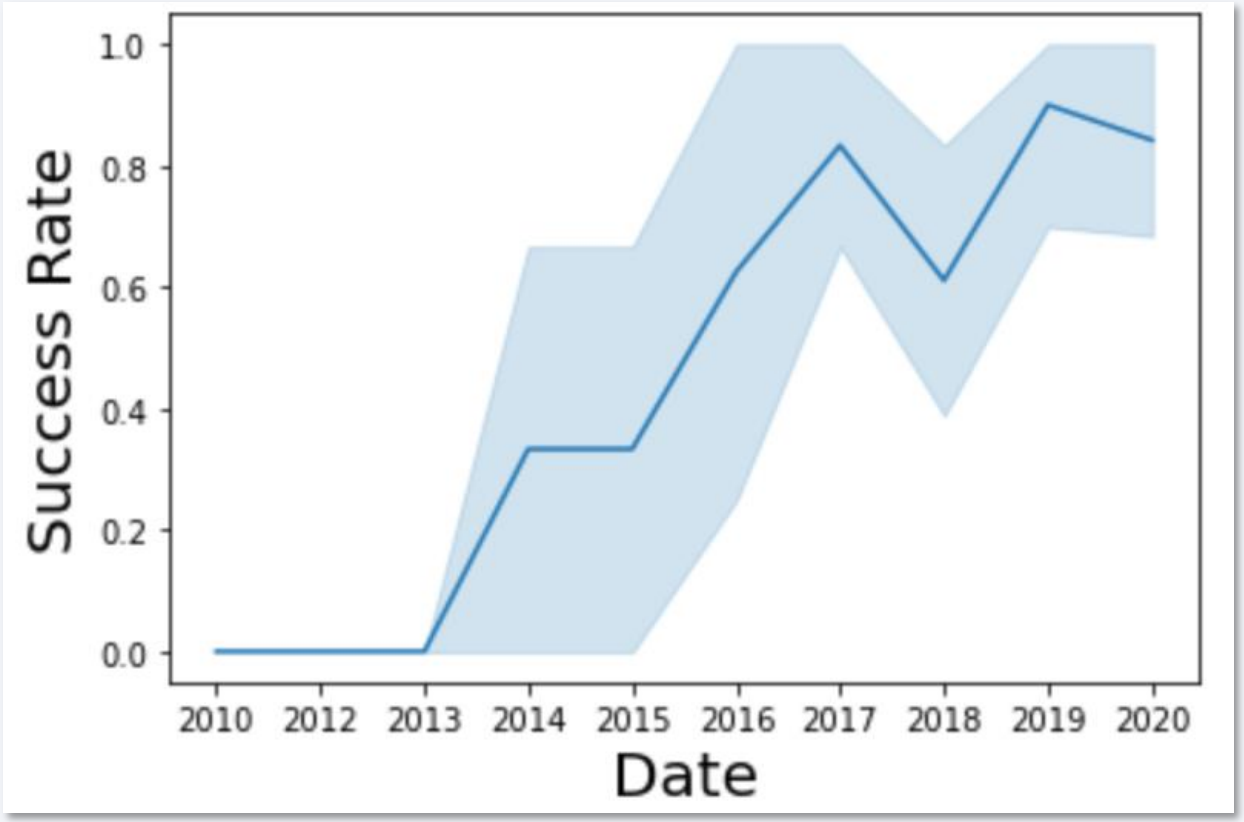
The graph shows that heavier payloads have positive impact on success rate for LEO, ISS and PO orbits and negative impact for MEO and VLEO orbits.

For GTO orbit there doesn't seem any correlation between payload mass and the success rate.



Launch Success Yearly Trend

From the timeline we can see that the success rate has been increasing since 2013 except a slight dip in 2018.



Section 2

Insights drawn from EDA with SQL

All Launch Site Names

We used the key word DISTINCT to show only unique launch sites from the SpaceX data.

In [10]:
```
%sql select distinct launch_site from SPACEX
```

```
 * ibm_db_sa://xmz96243:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
Done.
```

Out[10]:

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

Launch Site Names Begin with 'CCA'

We used the keyword LIMIT 5 to fetch five records from table Spacex and keyword LIKE for the condition that Launch_site name must start with ‘CCA’

In [14]:
```
%sql select * from Spacex where launch_site LIKE 'CCA%' limit 5
```

```
 * ibm_db_sa://xmz96243:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
Done.
```

Out[14]:

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	None	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	None	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	None	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	None	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	None	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

Using the function SUM calculates the total sum of payload mass.

The WHERE clause filters the dataset to only perform calculation on customer NASA(CRS).

```
In [35]: %sql Select sum(payload_mass__kg_) from SPACEX where customer like 'NASA%(CRS)'
 * ibm_db_sa://xmz96243:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
Done.
Out[35]:    1
45596
```

Average Payload Mass by F9 v1.1

Using the function AVG works out the average in the column payload_mass_kg_
The WHERE clause filters the dataset to only perform calculations on Booster_version F9 v1.1

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

In [30]:
```
%sql Select avg(payload_mass__kg_) from SPACEX where booster_version = 'F9 v1.1'
```

```
 * ibm_db_sa://xmz96243:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
Done.
```

Out[30]:

1
2928

First Successful Ground Landing Date

Using the function MIN works out the minimum date in the column Date

The WHERE clause filters the dataset to only perform calculation on Landing_Outcome Success with ground pad

```
In [32]: %sql select min(DATE) from spacex where mission_outcome = 'Success' and landing__outcome like '%ground pad%'

 * ibm_db_sa://xmz96243:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
Done.

Out[32]:        1

        2015-12-22
```

Successful Drone Ship Landing with Payload between 4000 and 6000

Selecting only Booster_Version

The WHERE clause filters the dataset to Landing_Outcome = Success (drone ship)

The AND clause specifies additional filter conditions Payload_MASS_KG_ BETWEEN 4000 and 6000

In [33]:
```
%%sql select booster_version from spacex
where mission_outcome = 'Success' and landing__outcome like '%drone ship%' and (payload_mass__kg_ between 4000 and 6000)
```

```
 * ibm_db_sa://xmz96243:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
Done.
```

Out[33]:

booster_version
F9 FT B1020
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

Using the GROUP BY function on mission_outcome, we calculated the total COUNT of each outcome.

In [37]:
```
%sql select mission_outcome, count(DATE) from spacex group by mission_outcome
```

```
 * ibm_db_sa://xmz96243:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
Done.
```

Out[37]:

mission_outcome	2
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

Boosters Carried Maximum Payload

Using the keyword ORDER BY and DESC in a SUB-QUERY we displayed the booster_versions which had maximum payload mass.

In [16]:
```
%%sql
select booster_version, payload_mass__kg_ from
(select * from spacex order by payload_mass__kg_ desc)
```

Out[16]:

booster_version	payload_mass__kg_
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

2015 Launch Records

We used a combinations of the keywords WHERE, LIKE, AND and YEAR() function to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

In [51]:
```
%%sql
select landing__outcome, booster_version, launch_site from spacex
where landing__outcome like 'Failure%(drone ship)' and year(DATE) = 2015
```

```
 * ibm_db_sa://xmz96243:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
Done.
```

Out[51]:

landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

We selected Landing outcomes and the COUNT of landing outcomes from the data and used the WHERE clause to filter for landing outcomes BETWEEN 2010-06-04 to 2010-03-20.

We applied the GROUP BY clause to group the landing outcomes and the ORDER BY clause to order the grouped landing outcome in descending order.

In [19]:
```
%%sql
select landing__outcome, count(landing__outcome) as "Total count" from spacex
where date between '2010-06-04' and '2017-03-20'
group by landing__outcome
order by count(landing__outcome) desc
```

```
 * ibm_db_sa://xmz96243:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
Done.
```

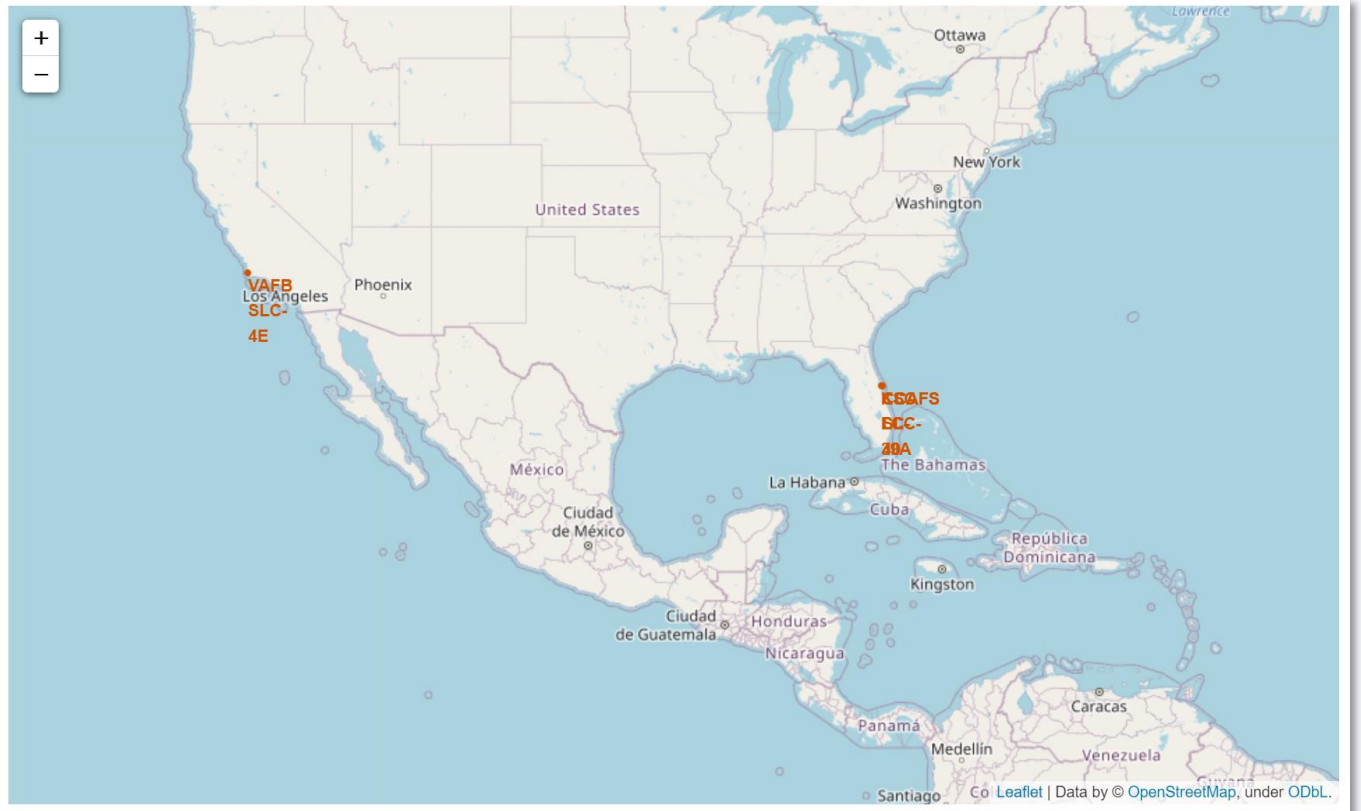
Out[19]:

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


Section 3
Launch Sites
Proximities Analysis

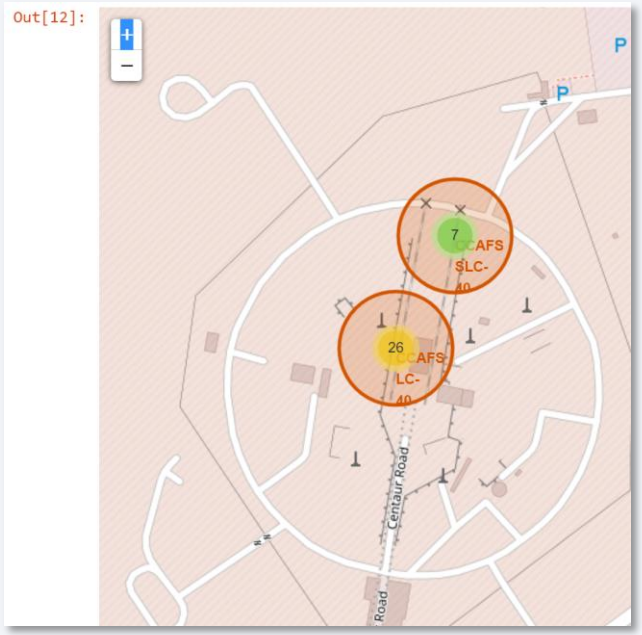
All Launch sites on Folium Map

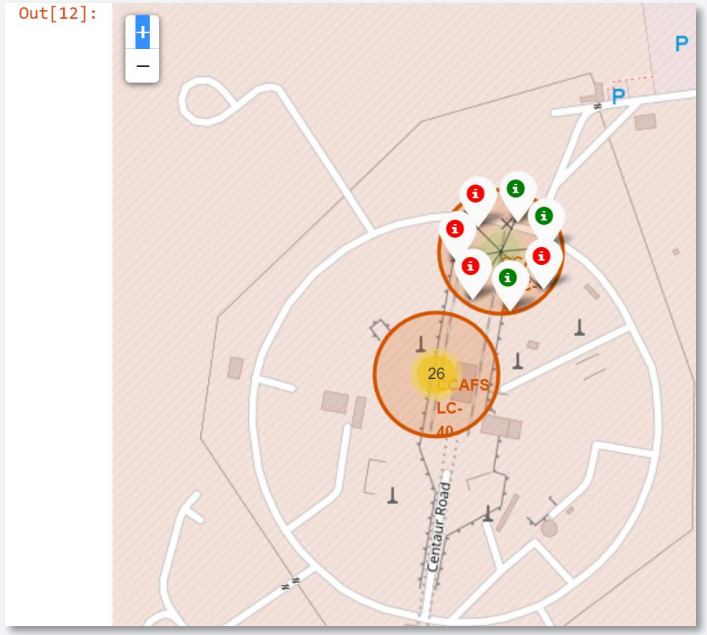
Out[8]:

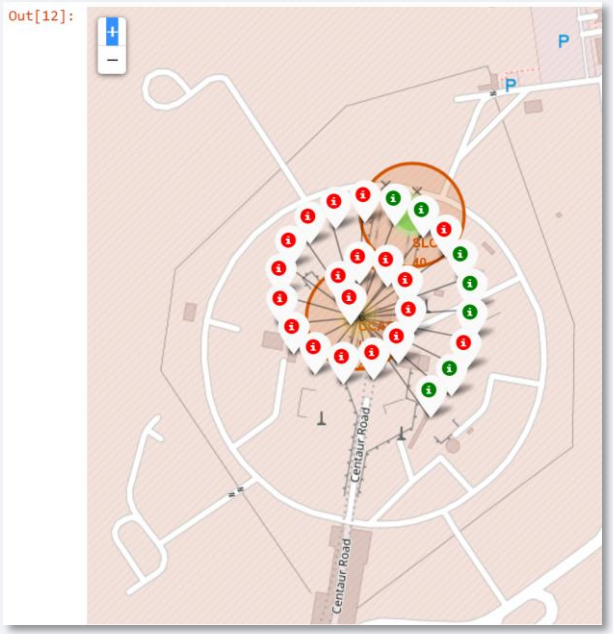


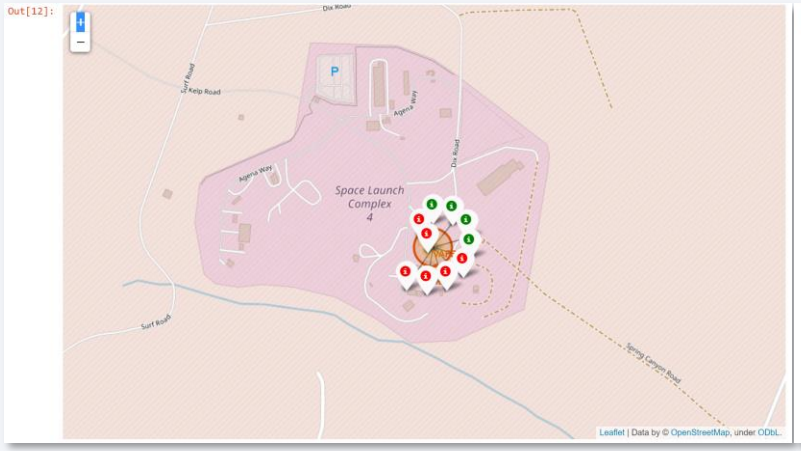
We can see that all the launch sites are in USA, Florida and California.

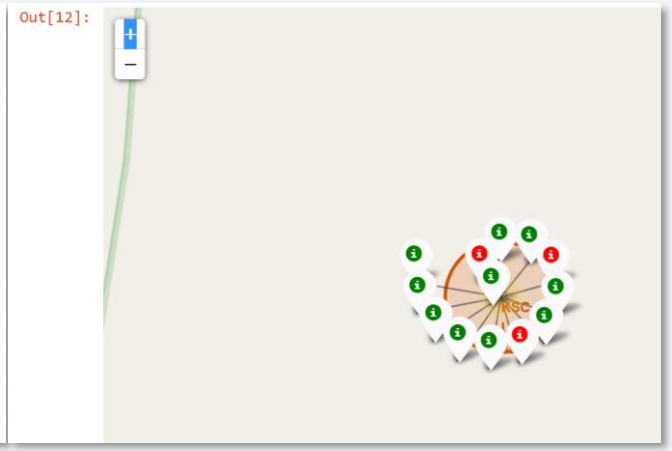
Marker showing Launch sites with Colour labels







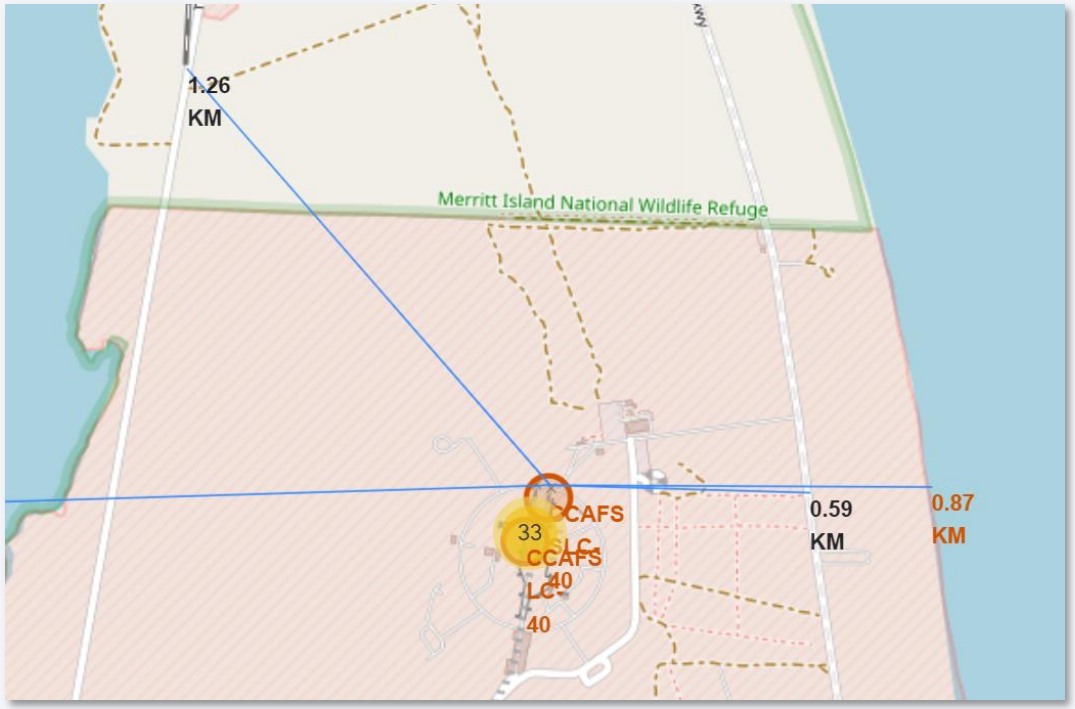


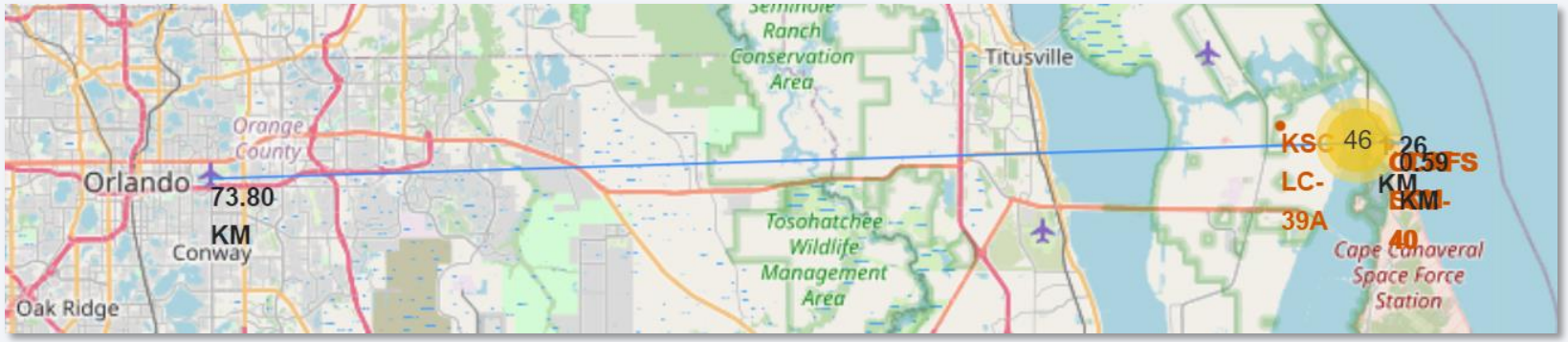


Launch sites with Green markers showing successes and Red markers showing failures

Distance of launch site CCAFS SLC-40 from its proximities

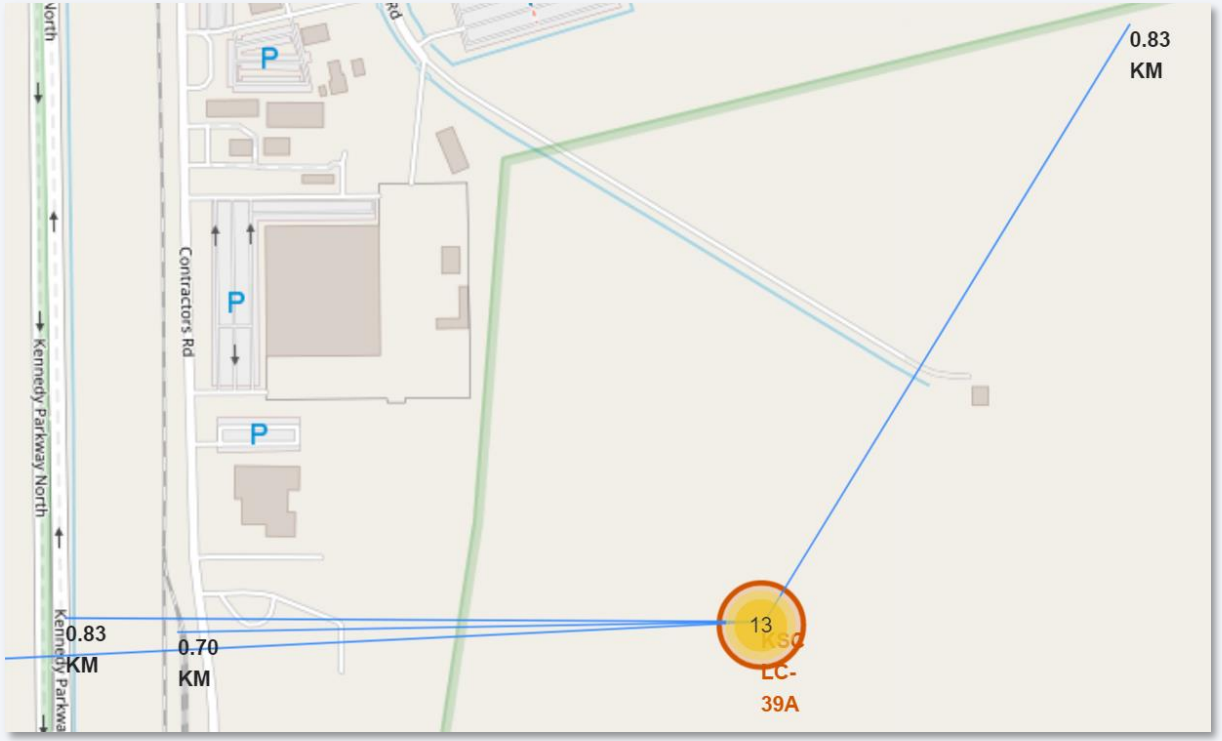
From CCAFS SLC-40	Distance (Km)
Coastline	0.87
Highway	0.59
Railway Line	1.26
Florida city	73.80





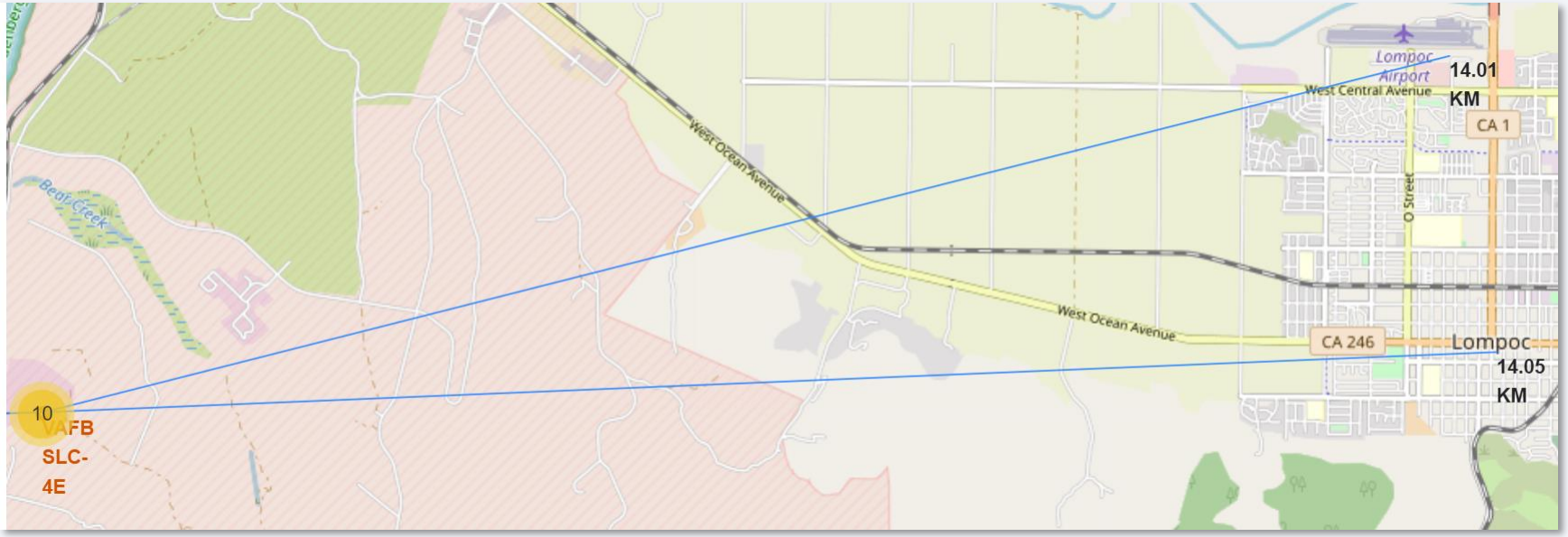
Distance of launch site KSC SLC-39A from its proximities

From KSC SLC-39A	Distance (Km)
Coastline	0.83
Highway	0.83
Railway Line	0.70
Florida city	66.93



Distance of launch site VAFB SLC-4E from its proximities

From VAFB SLC-4E	Distance (Km)
Coastline	1.43
Highway	14.01
Railway Line	1.31
Lompoc city	14.05



Answering questions using Folium Map

- Are launch sites in close proximity to railways?

  Yes, distances from nearest railway track are 0.7km, 1.31km, 1.26km for three launch sites.

- Are launch sites in close proximity to highways?

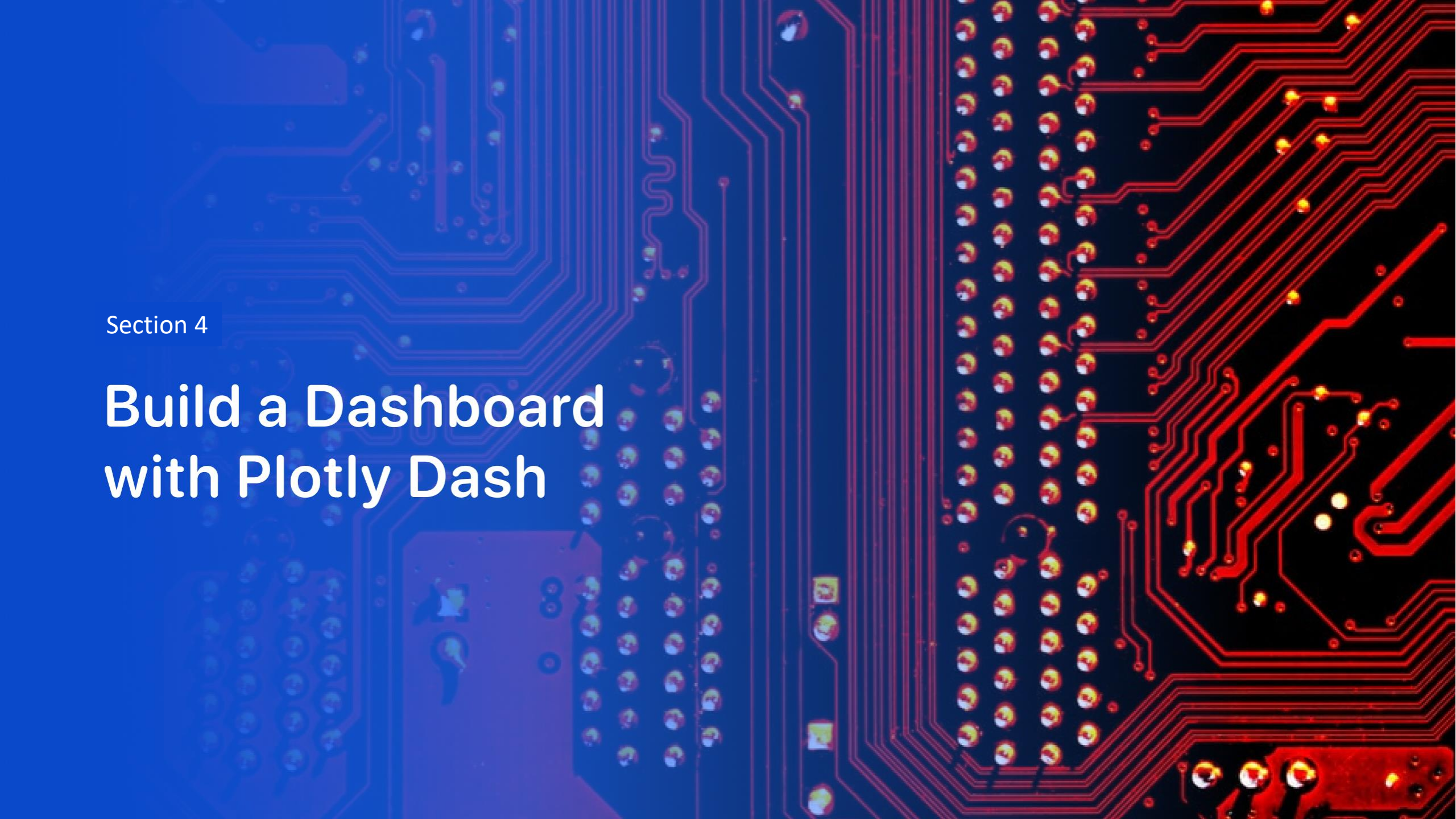
  Distances from nearest highway are 0.59km, 0.83km, 14.01km for three launch sites. So for Florida sites they are close but not for California site.

- Are launch sites in close proximity to coastline?

  Yes, distances from coastline are 0.87km, 0.83km, 1.43km for three launch sites.

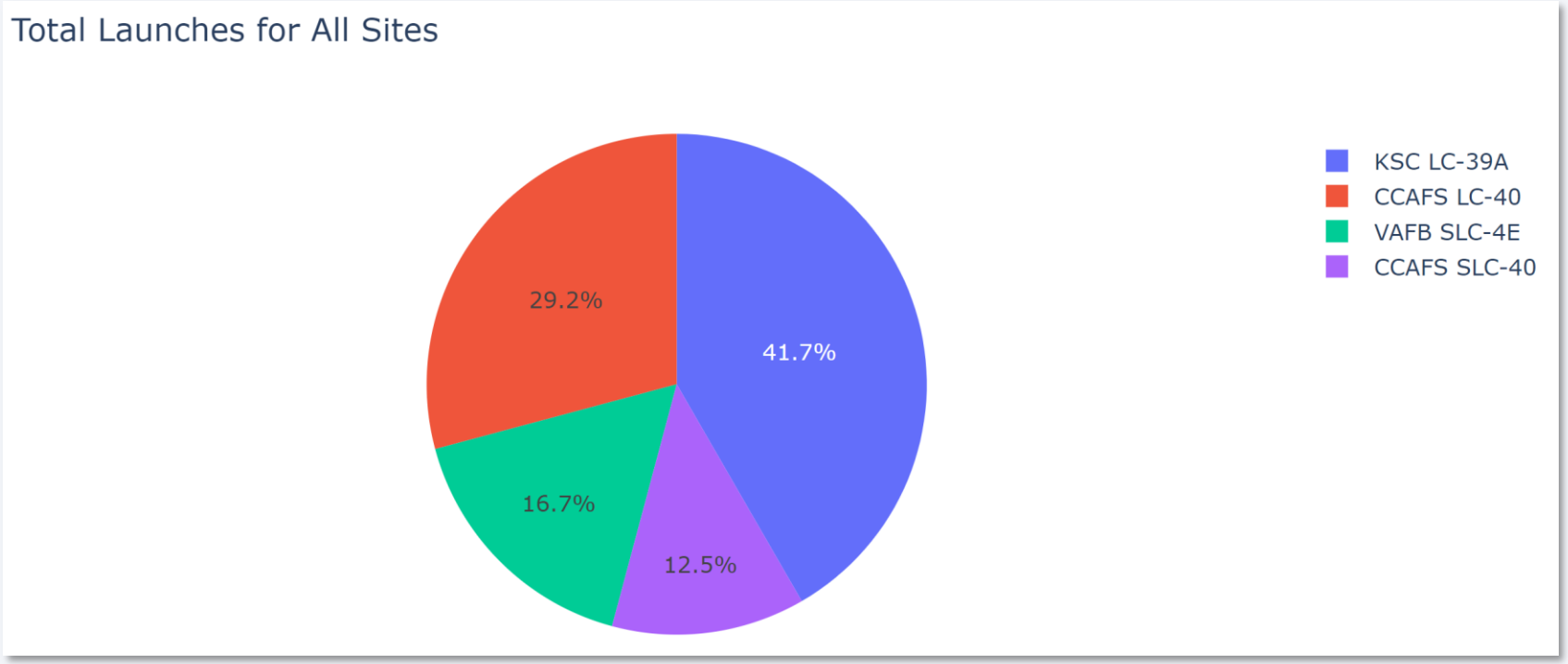
- Do launch sites keep certain distance away from cities?

  Yes, distances from nearest city are 73.80km, 66.93km and 14.05km for three launch sites.


Section 4
Build a Dashboard with Plotly Dash

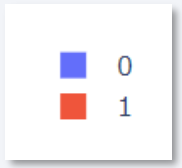
Launch Success count for all sites

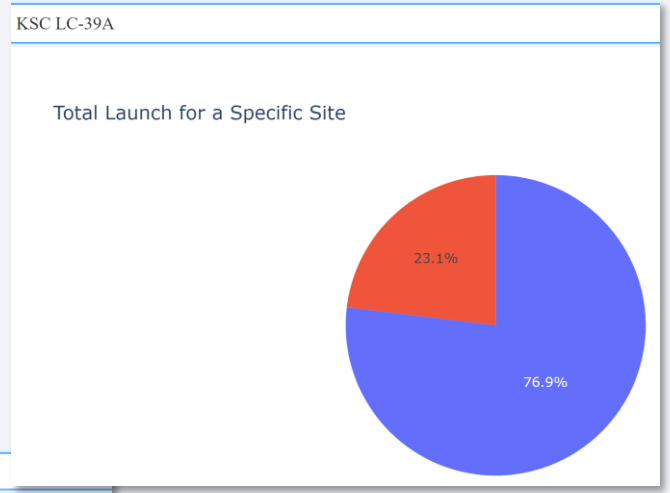
We can see from the pie chart that KSC LC-39A has the most successful launches and CCAFS SLC-40 has the least successful launches

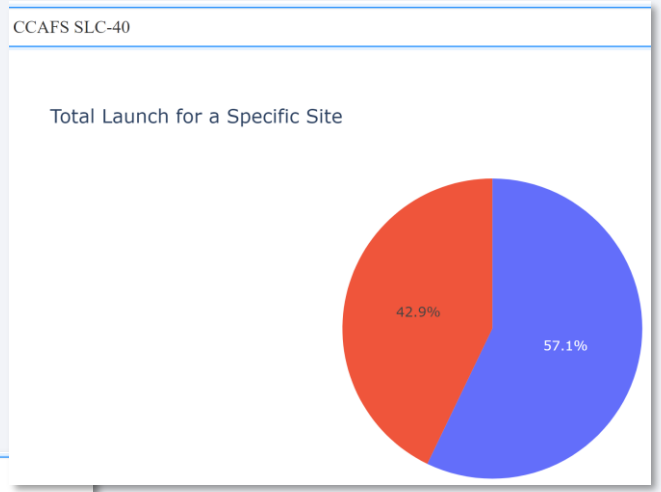


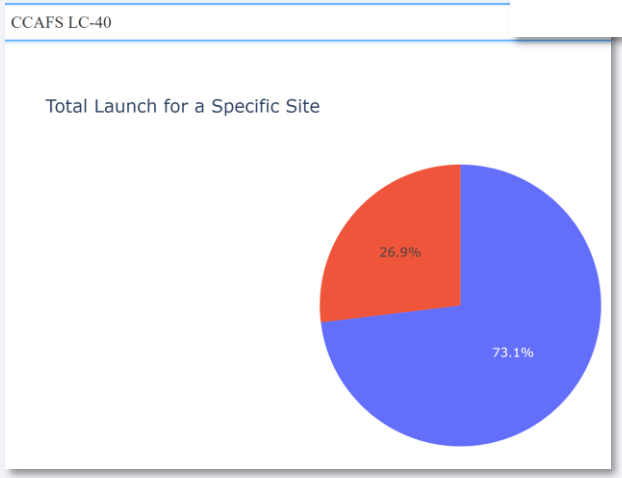
Launch success ratio for each site

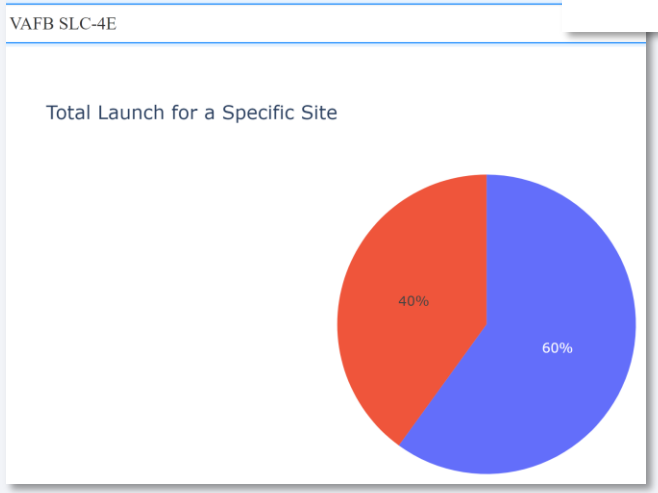
We can see that KSC LC-39A has highest launch success rate of 76.9% while CCAFS SLC-40 has the least launch success rate of 57.1%





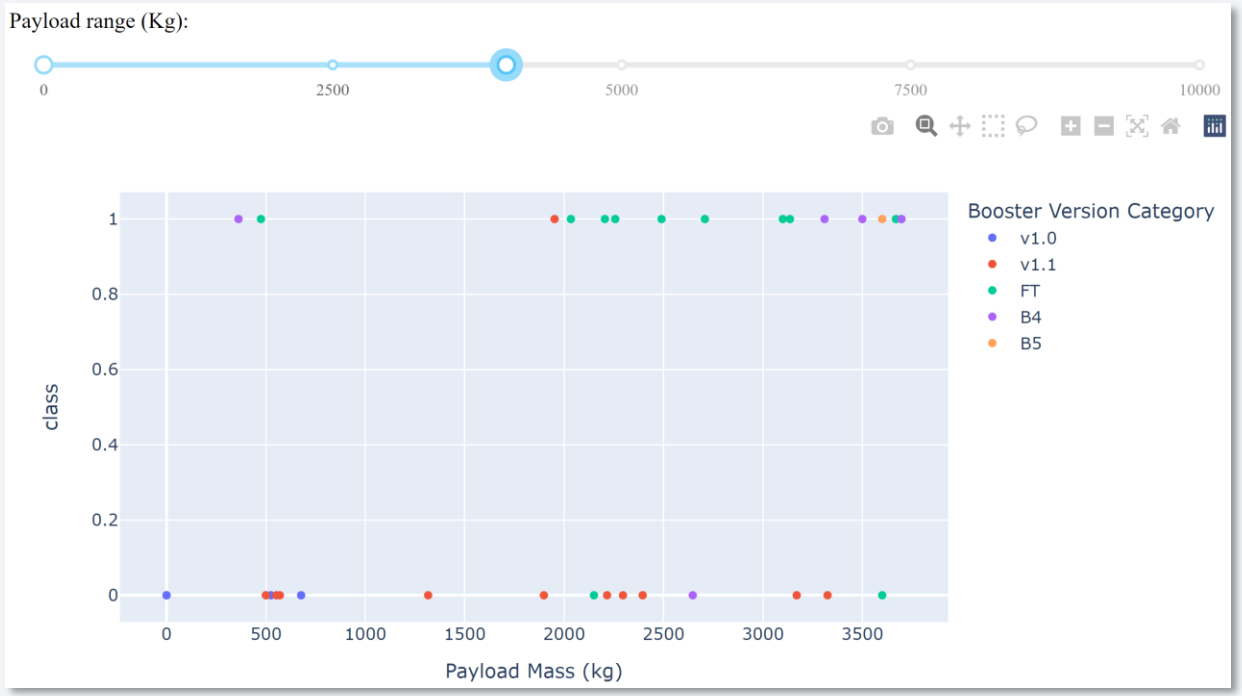


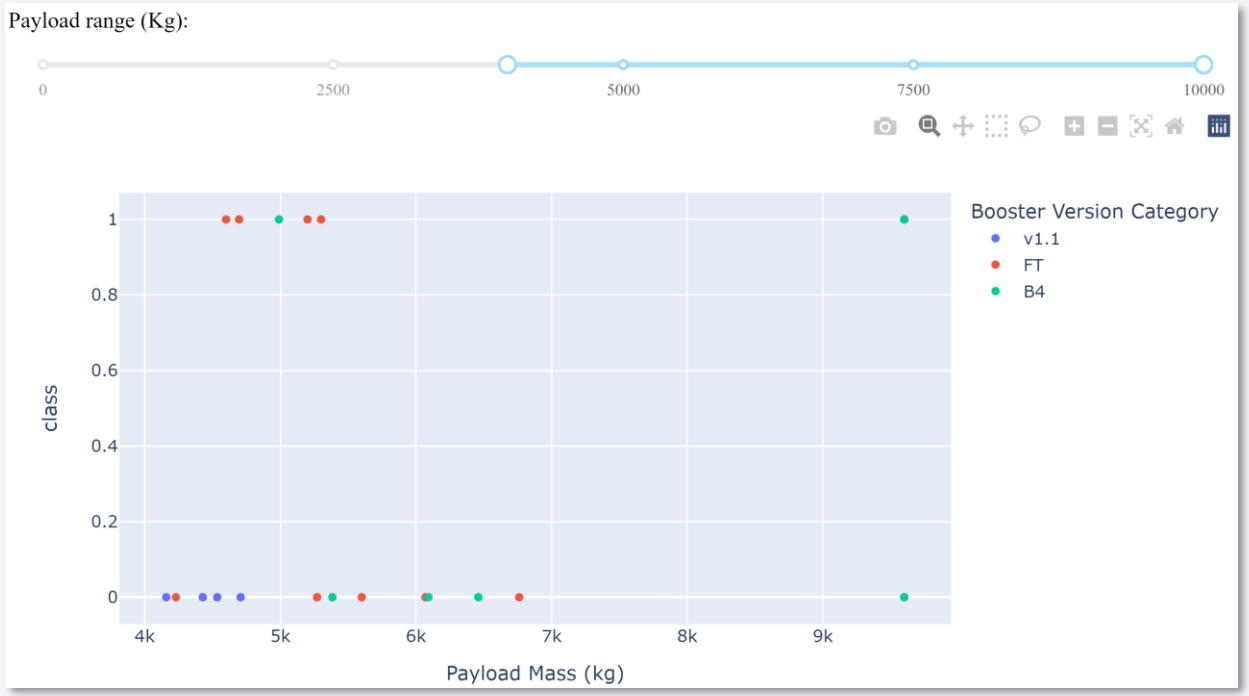




Payload mass Vs launch outcome – Scatter graph

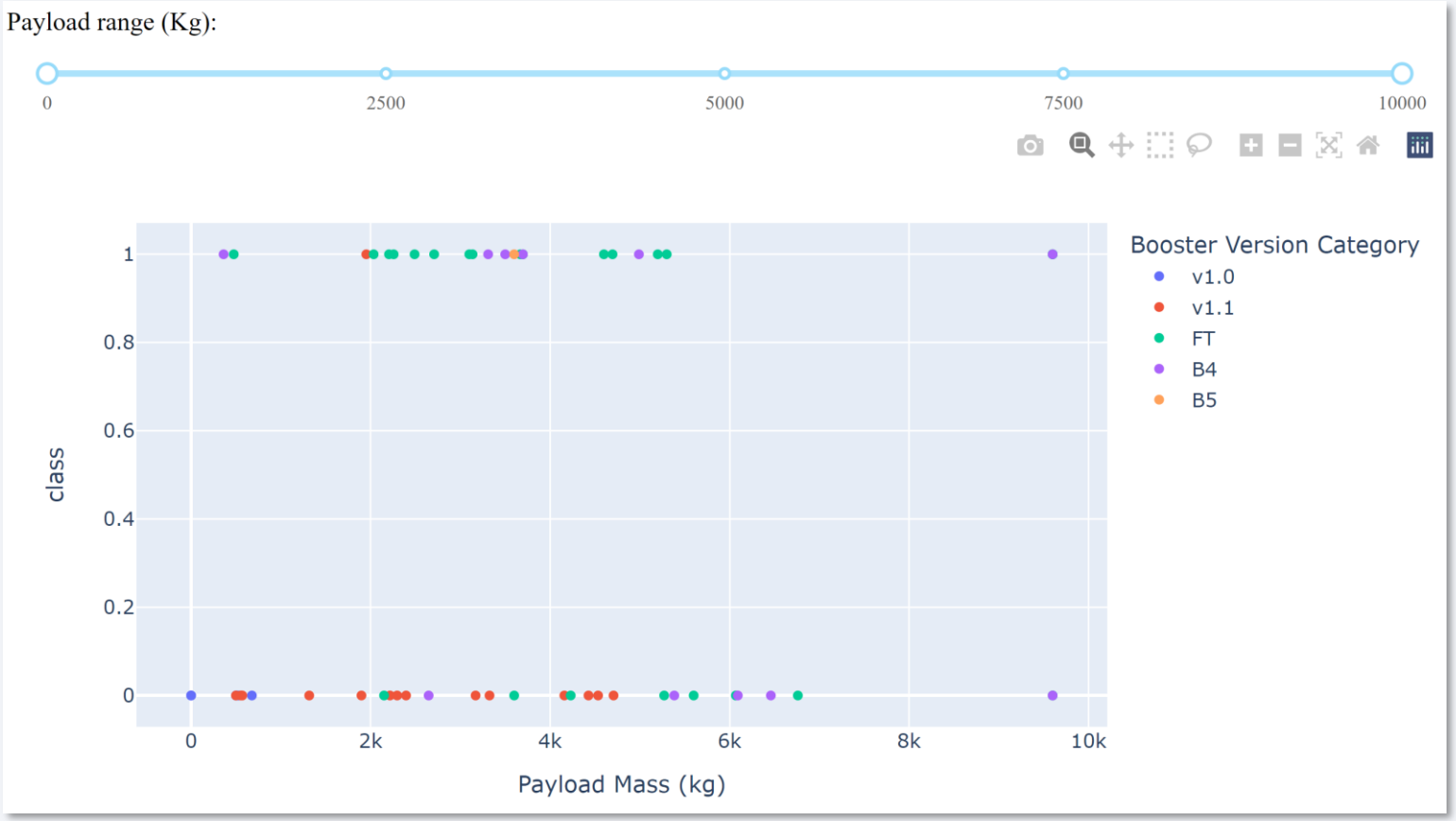
We can see from the scatter graphs that launch success rate is higher for Payload mass in the range of 0-4000kg and lower for payload mass more than 4000kg.





Success rate for all Booster Versions

We can see from the scatter graph for full payload mass range that launch success rate is highest for Booster FT



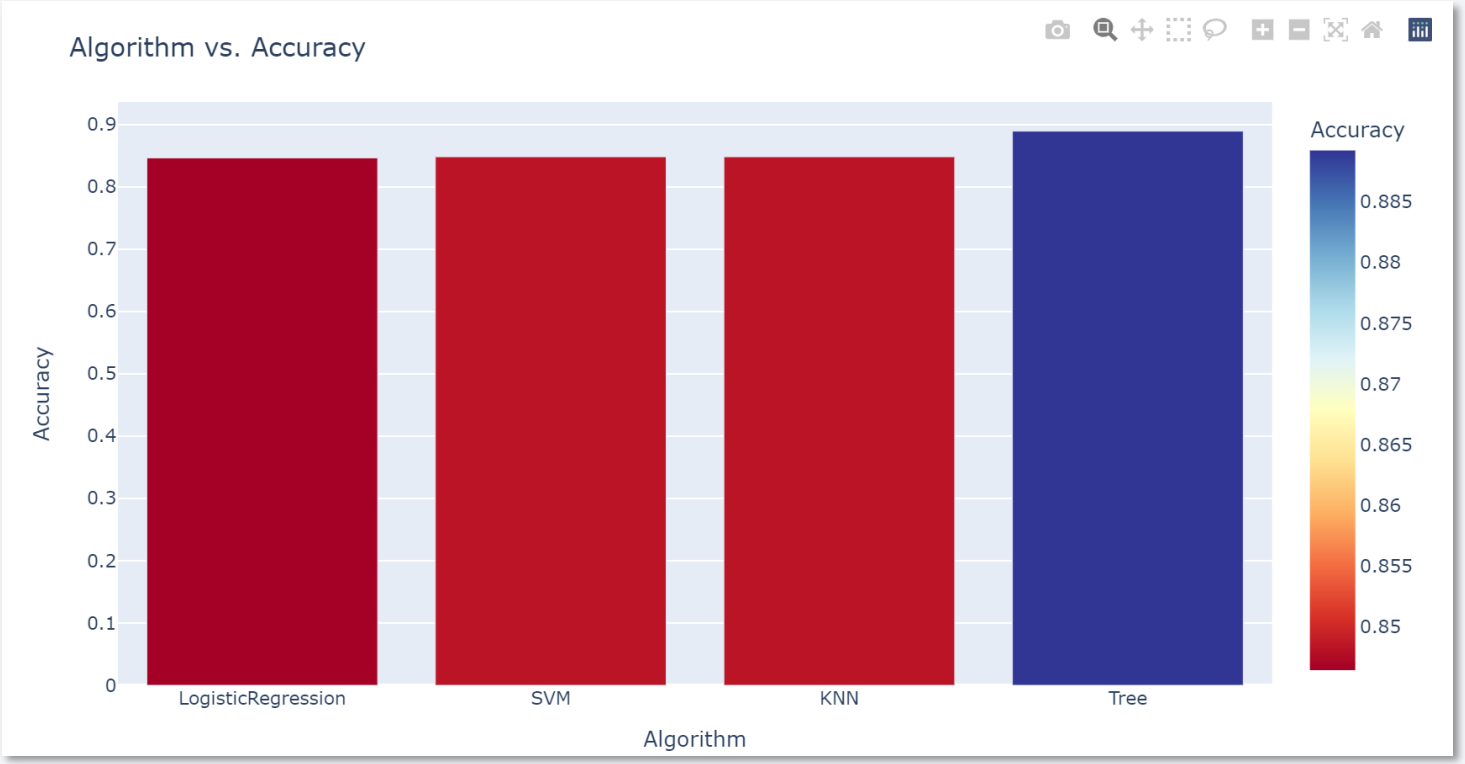
Section 5

Predictive Analysis (Classification)

Classification Accuracy

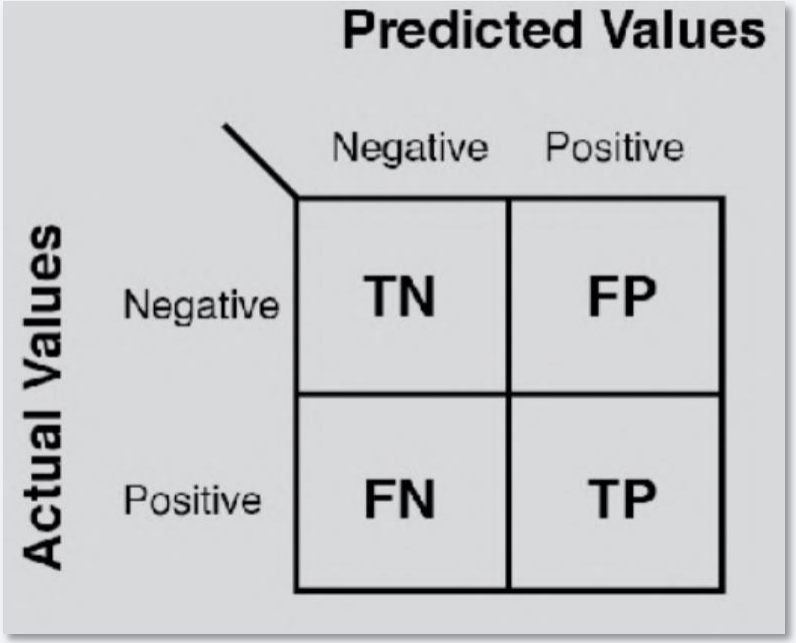
We can see that accuracy of all the models is good and quite close but the best of them all is **Decision Tree** model with 88.93% accuracy

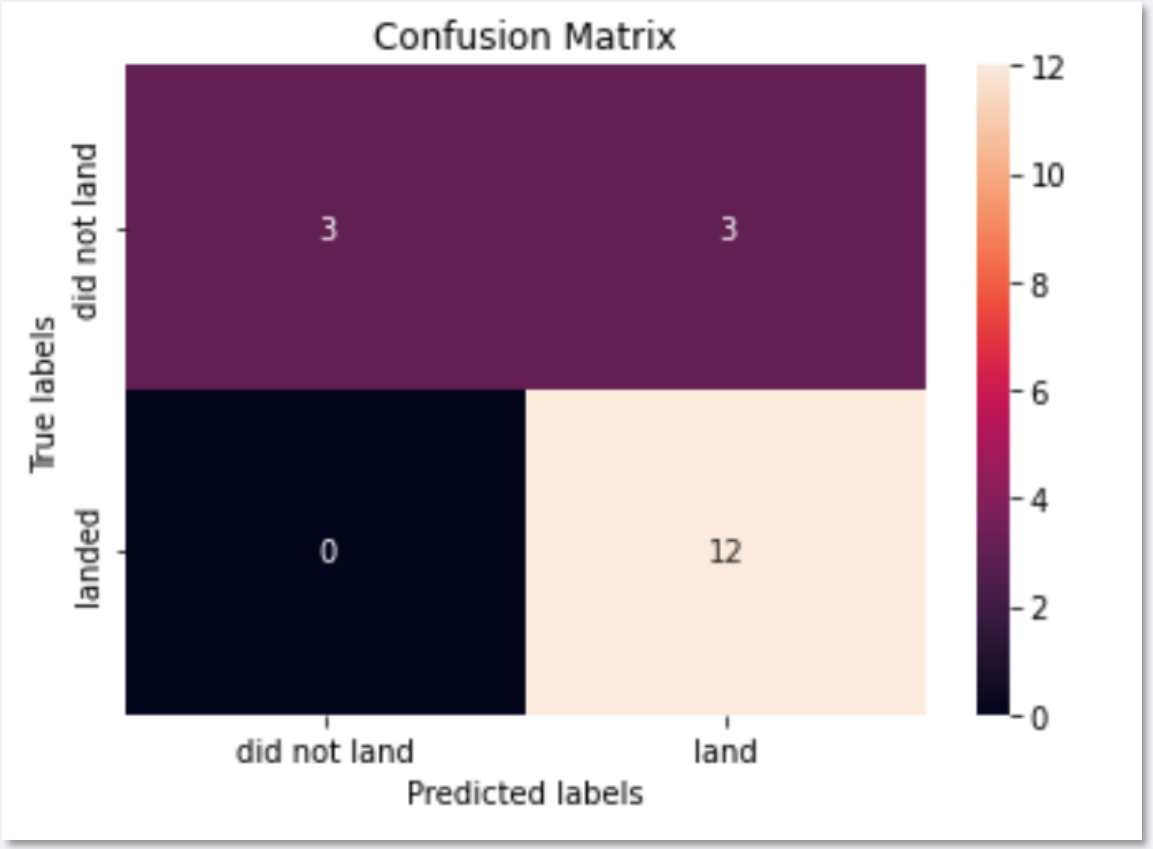
	Algorithm	Accuracy
0	LogisticRegression	0.846429
1	SVM	0.848214
2	KNN	0.848214
3	Tree	0.889286



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 Tree Classifier Algorithm is the best for Machine Learning for this dataset
- Low weighted payloads (4000kg and below) perform better than the heavier payloads
- The success rates for SpaceX launches has been increasing relatively with time and it looks like it will be eventually possible to have perfect launches.
- KSC LC-39A had the most successful launches ie. 76.9%
- Orbits GEO,HEO,SSO,ES-L1 have the best Success rate.

Thank you!