IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Bo Wei Chien(錢柏維)
2021/01/03

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection
  - Data wrangling
  - EDA with data visualization
  - EDA with SQL
  - Building an interactive map with Folium
  - Building a Dashboard with Plotly Dash
  - Predictive analysis (Classification)
- Summary of all results
  - Exploratory data analysis results
  - Interactive analytics demo in screenshots
  - Predictive analysis results

Introduction

- Project background and context
- We will 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.

- Problems need to be researched
  - What influences if the rocket will land successfully?
  - The effect each relationship with certain rocket variables will impact in determining the success rate of a successful landing.
  - What conditions does SpaceX have to achieve to get the best results and ensure the best rocket success landing rate.

Section 1

Methodology

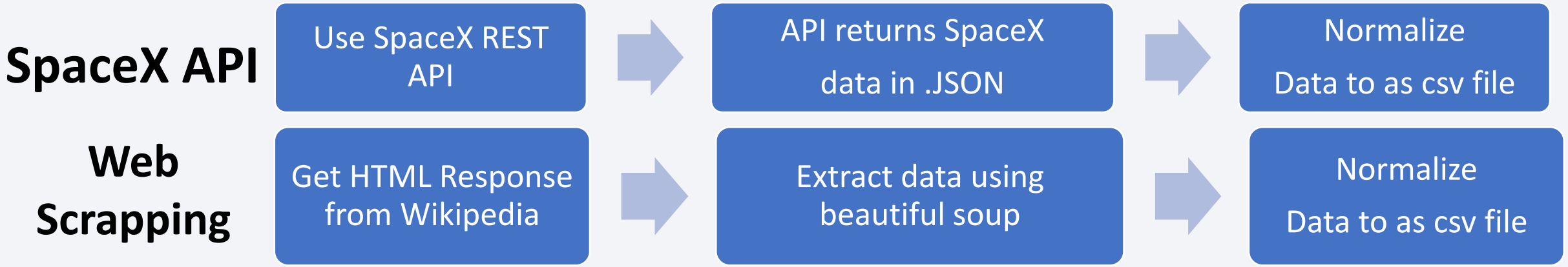
Methodology

Executive Summary

- Data collection methodology:
  - SpaceX Rest API
  - (Web Scrapping) from Wikipedia
- Perform data wrangling
  - Create a landing outcome label from Outcome column
  - One Hot Encoding data fields for Machine Learning and dropping irrelevant columns
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Find best Hyperparameter for SVM, Classification Trees and Logistic Regression
  - Find the method performs best using test data

Data Collection

- Describe how data sets were collected.
  - We worked with SpaceX launch data that is gathered from the SpaceX REST API.
  - This API will give us data about launches, including information about the rocket used, payload delivered, launch specifications, landing specifications, and landing outcome.
  - Our goal is to use this data to predict whether SpaceX will attempt to land a rocket or not.
  - The SpaceX REST API endpoints, or URL, starts with api.spacexdata.com/v4/.
  - Another popular data source for obtaining Falcon 9 Launch data is web scraping Wikipedia using BeautifulSoup



Data Collection – SpaceX API

GitHub URL to Notebook

1. Getting Response from SpaceX API
2. Converting Response to a .json file
3. Apply custom functions to extract information to multiple global variables (list) from json file
4. Construct our dataset(dictionary) from step3 global variables (list)
5. Trans dataset(list) to dataframe and filtering
6. Data Wrangling: Dealing with Missing Values
7. Export to flat file (.csv)

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url).json()
```

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

```
# Call getBoosterVersion
getBoosterVersion(data)
```

```
# Call getLaunchSite
getLaunchSite(data)
```

```
#Global variables
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

```
# Call getCoreData
getCoreData(data)
```

```
# Call getPayloadData
getPayloadData(data)
```

```
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}
```

SpaceX API → Use SpaceX REST API → API returns SpaceX data in .JSON → Normalize Data to as csv file

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

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9=dataframe[dataframe['BoosterVersion']!='Falcon 1']
# data['BoosterVersion']!='Falcon 1'
len(data_falcon9)
```

```
# Replace the np.nan values with its mean value
data_falcon9["PayloadMass"]=data_falcon9["PayloadMass"].replace(np.nan,data_falcon9["PayloadMass"].mean())
```



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

Data Collection - Scraping

GitHub URL to Notebook

1. Request the Falcon9 Launch HTML page, as an HTTP response.

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
response=requests.get(static_url)
```

2. Create BeautifulSoup object from the HTML as soup

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

3. Finding tables by find_tablesall function of bs4 object

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

4. Getting column names from html_tables and creation of dictionary and assign column names as Key of dict(launch_dict)

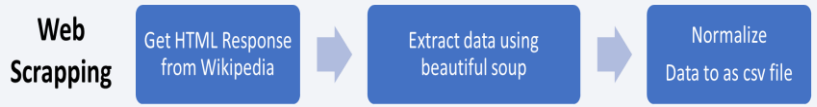
```
column_names = []
temp=soup.find_all('th')
temp
extract_column_from_header(temp[0])
for i in range(len(temp)):
    try:
        name=extract_column_from_header(temp[i])
        if (name is not None and len(name) > 0):
            column_names.append(name)
    except:
        pass
```

```
# Launch_dict= dict.fromkeys(column_na
launch_dict={}
# Remove an irrelvant column
# del launch_dict['Date and time ( )']

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

5. Appending data to relative keys of Step 4. launch_dict

```
#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
```



6. Converting dictionary to dataframe and Export to csv file

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

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

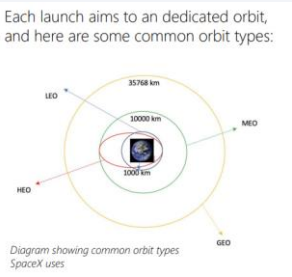
Data Wrangling

GitHub URL to Notebook

- Describe how data were processed

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;
In this lab we will mainly convert those outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful.

Outcome group	Meaning
True Ocean	means the mission outcome was successfully landed to a specific region of the ocean
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.



- Data wrangling process using key phrases and flowcharts

1. **Calculate the number of launches on each site**
2. **Calculate the number and occurrence of each orbit**
3. **Calculate the number and occurence of mission outcome per orbit type**
4. **Create a landing outcome label from Outcome column**

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

```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class=df['Outcome'].apply(lambda x: 0 if x in bad_outcomes else 1)
```

```
df['Class']=landing_class
df[['Class']].head(8)
```

	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

EDA with Data Visualization

GitHub URL to Notebook

Chart type	X ; Y ; Group
Scatter chart Hue=Class	FlightNumber vs. PayloadMass
	Flight Number vs. Launch
	Payload Vs. Launch
	FlightNumber and Orbit type
	Payload and Orbit type
Bar chart	Mean VS. Orbit
Line chart	Year VS. success rate

EDA with SQL

GitHub URL to Notebook

- Using bullet point format, summarize the SQL queries you performed

  - *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 acheived.*
  - *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*

Build an Interactive Map with Folium

GitHub URL to Notebook

- To visualize the Launch Data into an interactive map.

- 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, successes) 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.

Build a Dashboard with Plotly Dash

GitHub URL to Notebook

DashBoard HTML file

Pie Chart showing the total launches by a certain site/all sites

- display relative proportions of multiple classes of data.
- size of the circle can be made proportional to the total quantity it represents.

Scatter Graph showing the relationship with Outcome and Payload Mass (Kg) for the different Booster Versions

- It shows the relationship between two variables.
- It is the best method to show you a non-linear pattern.
- The range of data flow, i.e. maximum and minimum value, can be determined. - Observation and reading are straightforward.

Predictive Analysis (Classification)

GitHub URL to Notebook

BUILDING MODEL

- Load our dataset(After apply OneHotIncoder) into NumPy and Pandas
- Preprocessing:
  1. Data Standardization
  2. Split our data into training and test data sets
- Mode algorithms : Logistic regression, SVM(support vector machine), decision tree, KNN algorithms
- For each mode, setting the parameters set and CV(cross-validation Kflod to create GridSearchCV objects
- Setting training datasets into the GridSearchCV objects and train the model.

EVALUATING MODEL

- Check accuracy for each model
- Get each hyperparameters split_test_score by GridSearchCV for each type of algorithm model
- Choose Best Params combination(highest split_test_score) of each algorithm model
- Calculate the accuracy on the test data for each algorithm model.
- Plot Confusion Matrix

FINDING THE BEST PERFORMING CLASSIFICATION MODEL

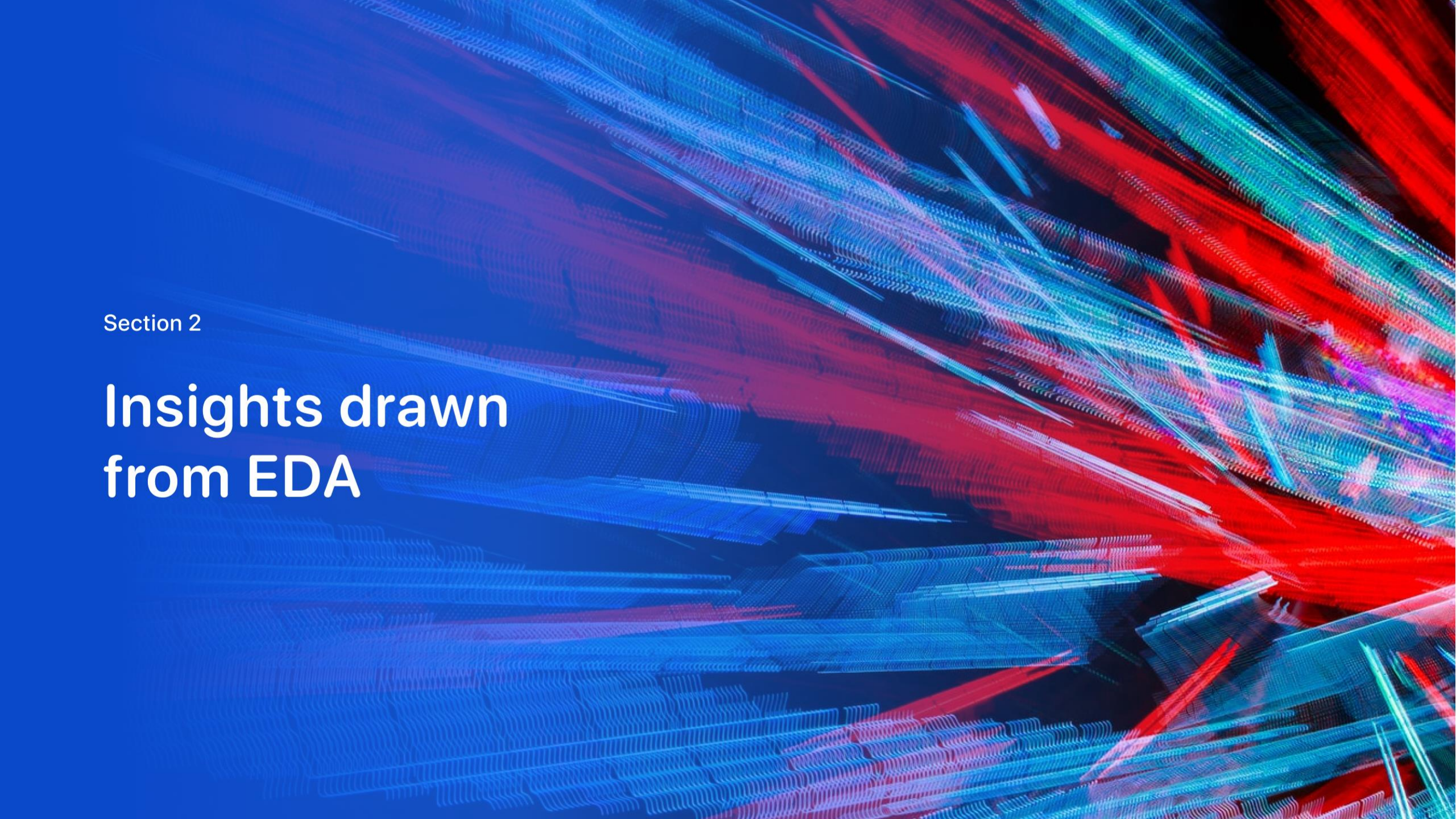
- Find the best algorithm model by highest accuracy of each algorithm on the testing dataset
- Output best Params of the best algorithm

Results

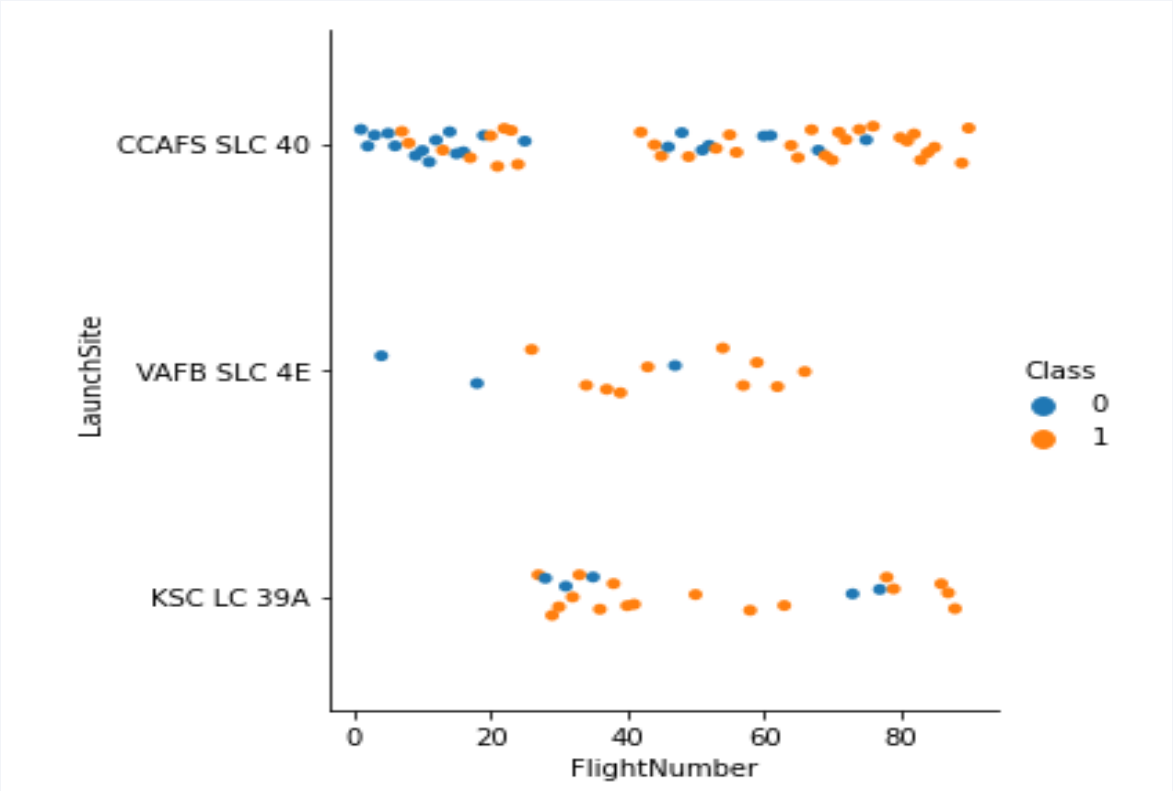
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA



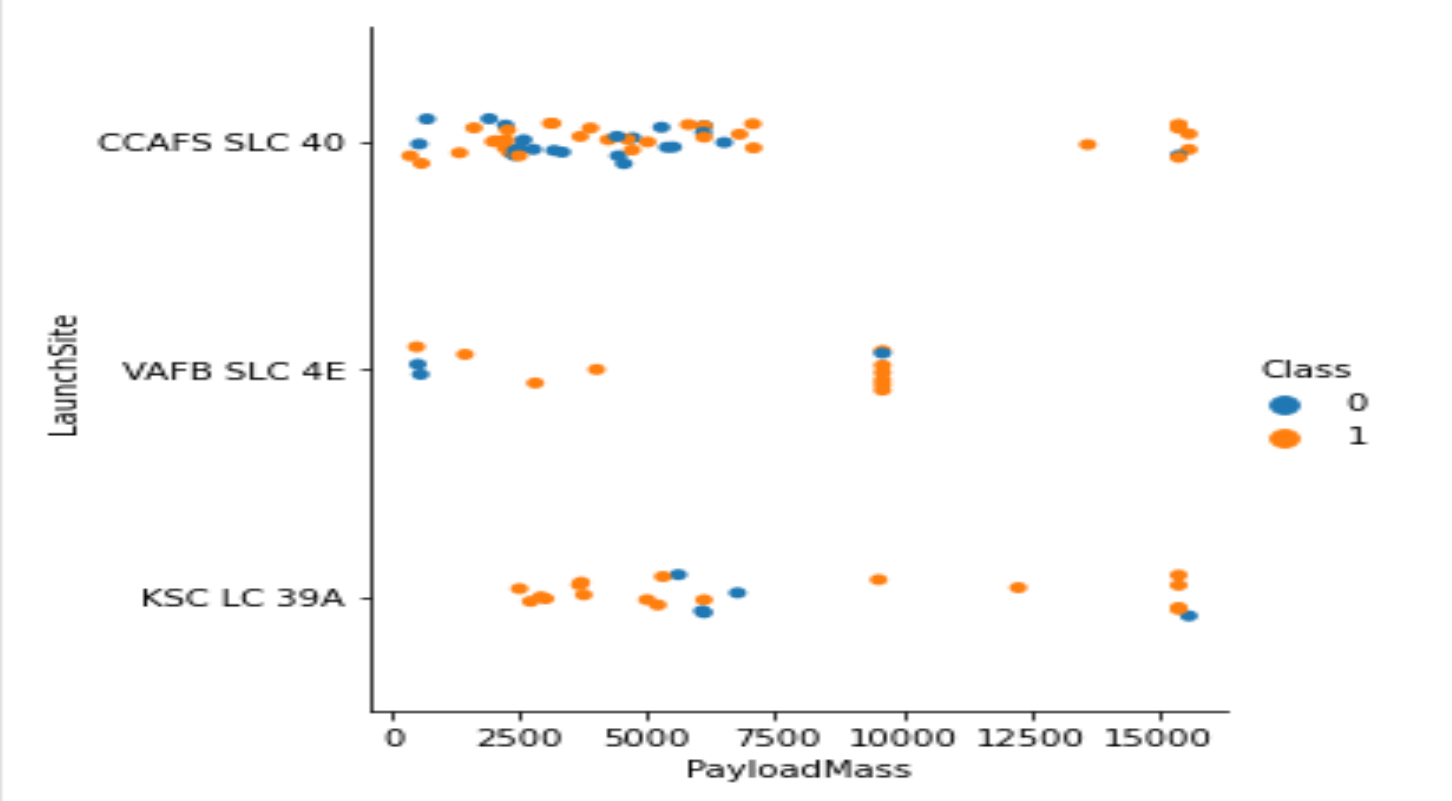
Flight Number vs. Launch Site



Result:

1. VAFB-SLC 4E launchsite there are no rockets launched for Flight Number greater than 70.
2. land successfully rate as Flight number increases on three different launchsite

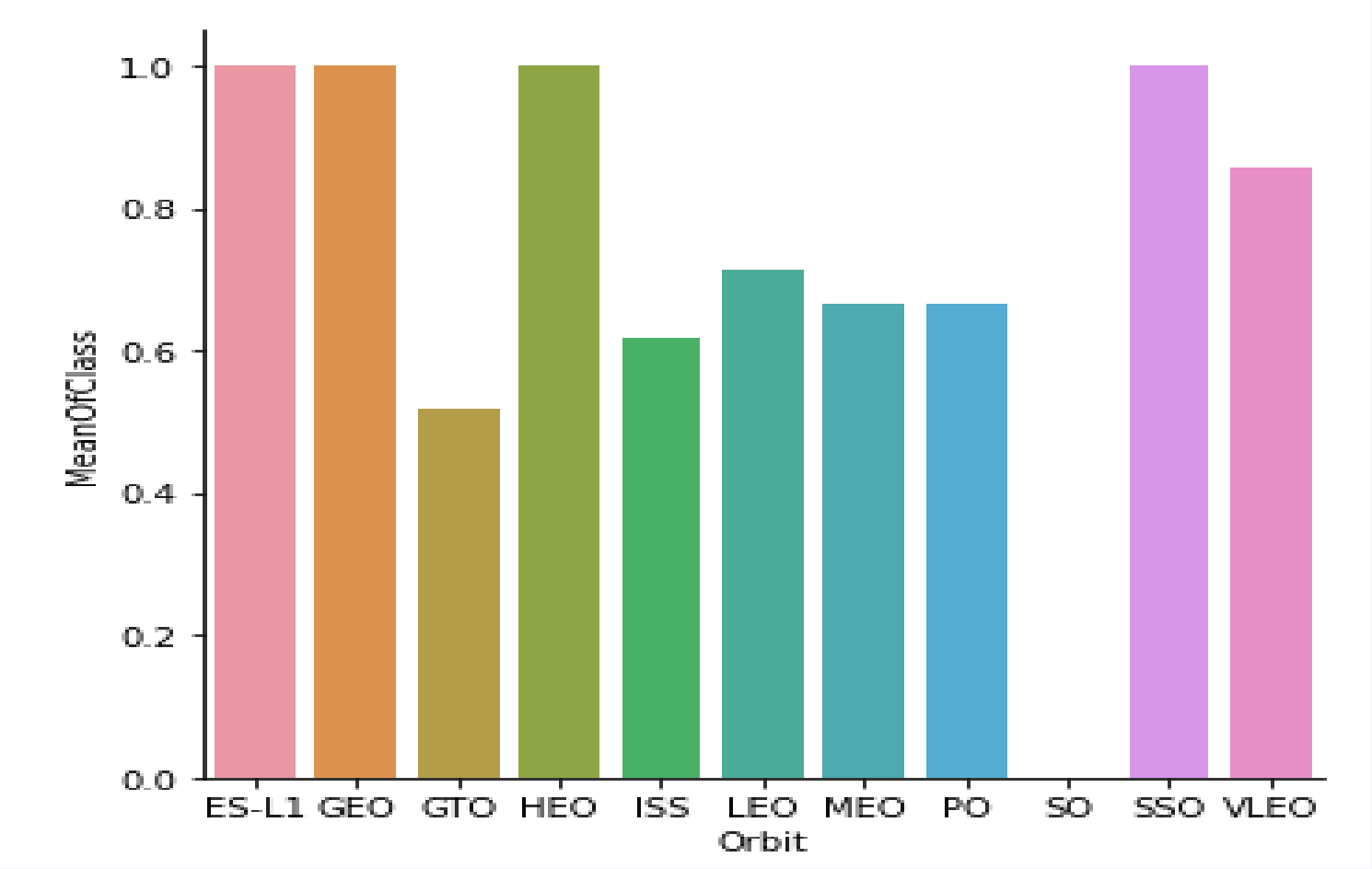
Payload vs. Launch Site



Result:

1. VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).
2. The greater the payload mass for Launch Site CCAFS SLC 40 the higher the success rate for the Rocket.

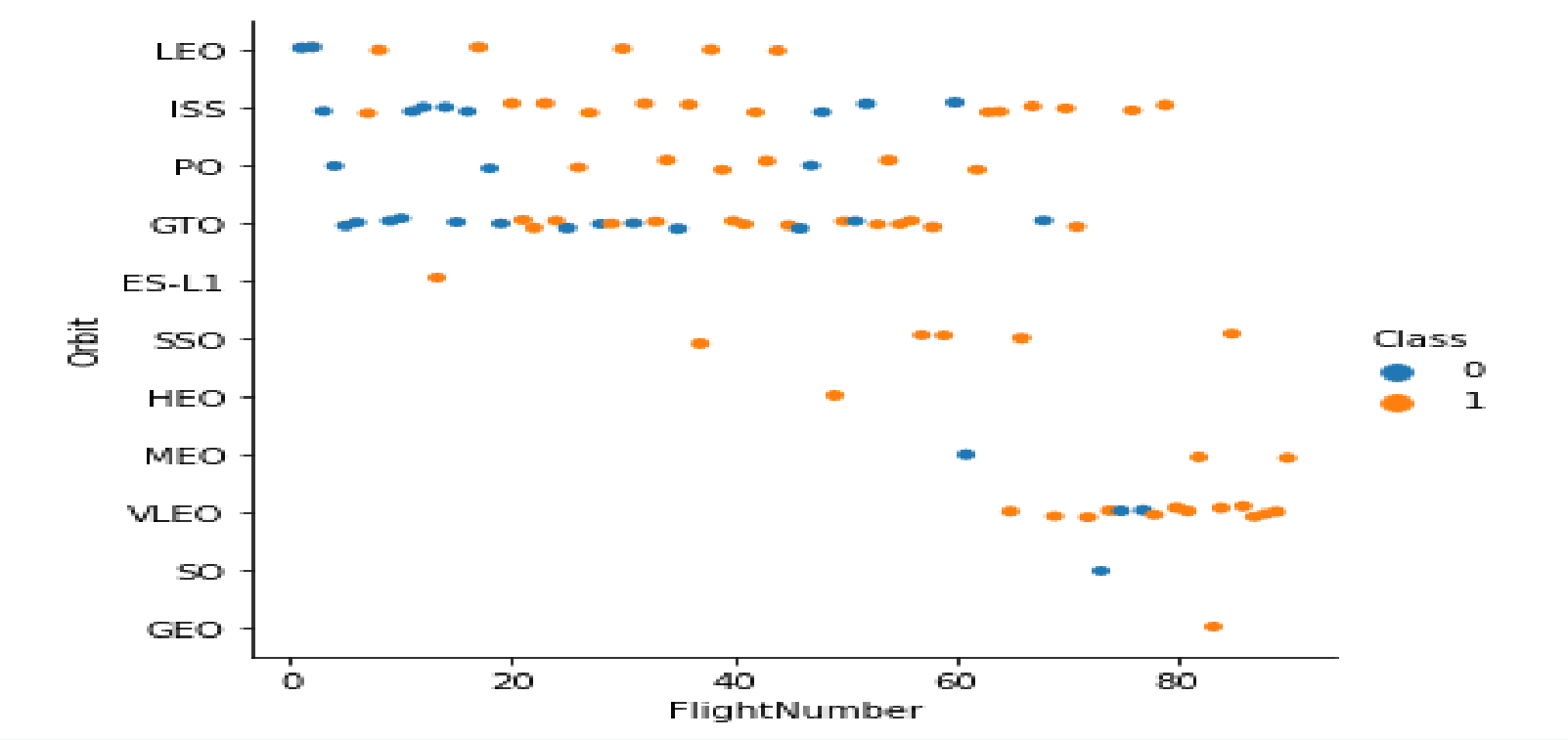
Success Rate vs. Orbit Type



Result:

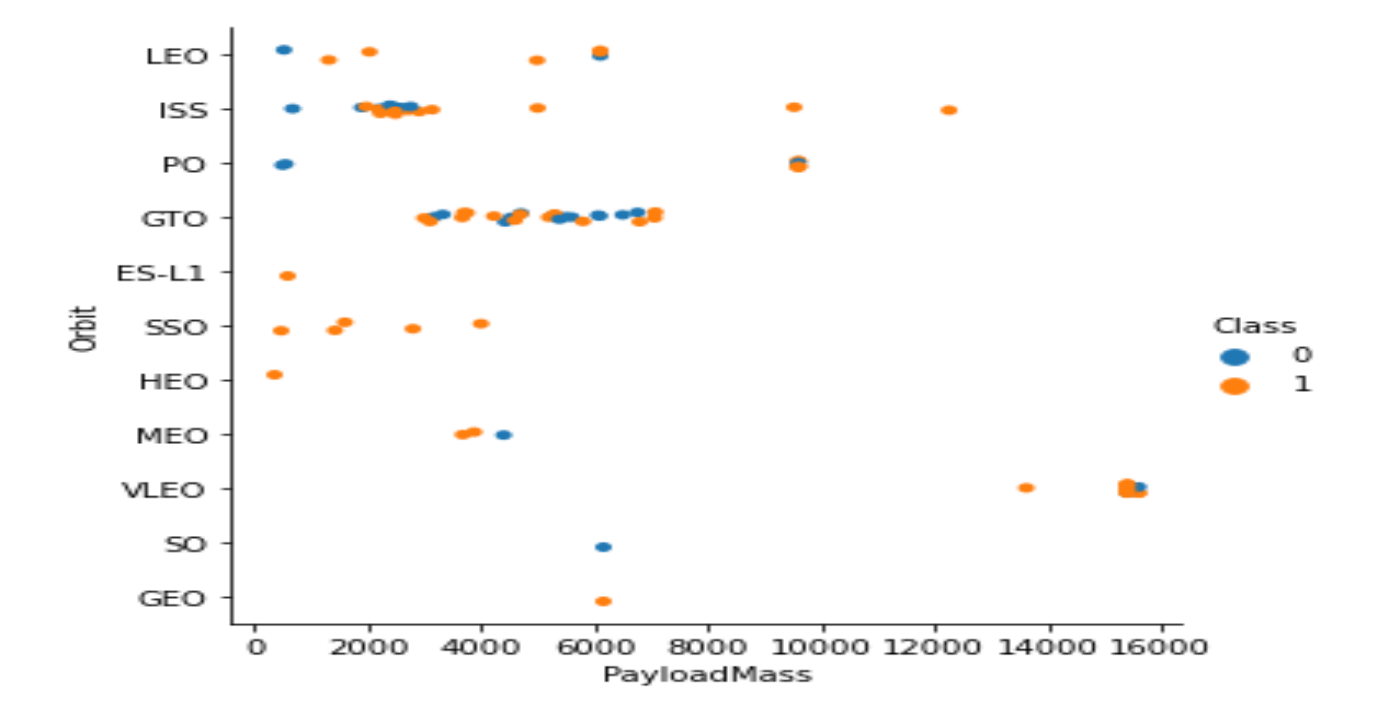
Orbits(ES-L1,GEO,HEO,SSO,VLEO) have high sucess rate.

Flight Number vs. Orbit Type



Result:
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.

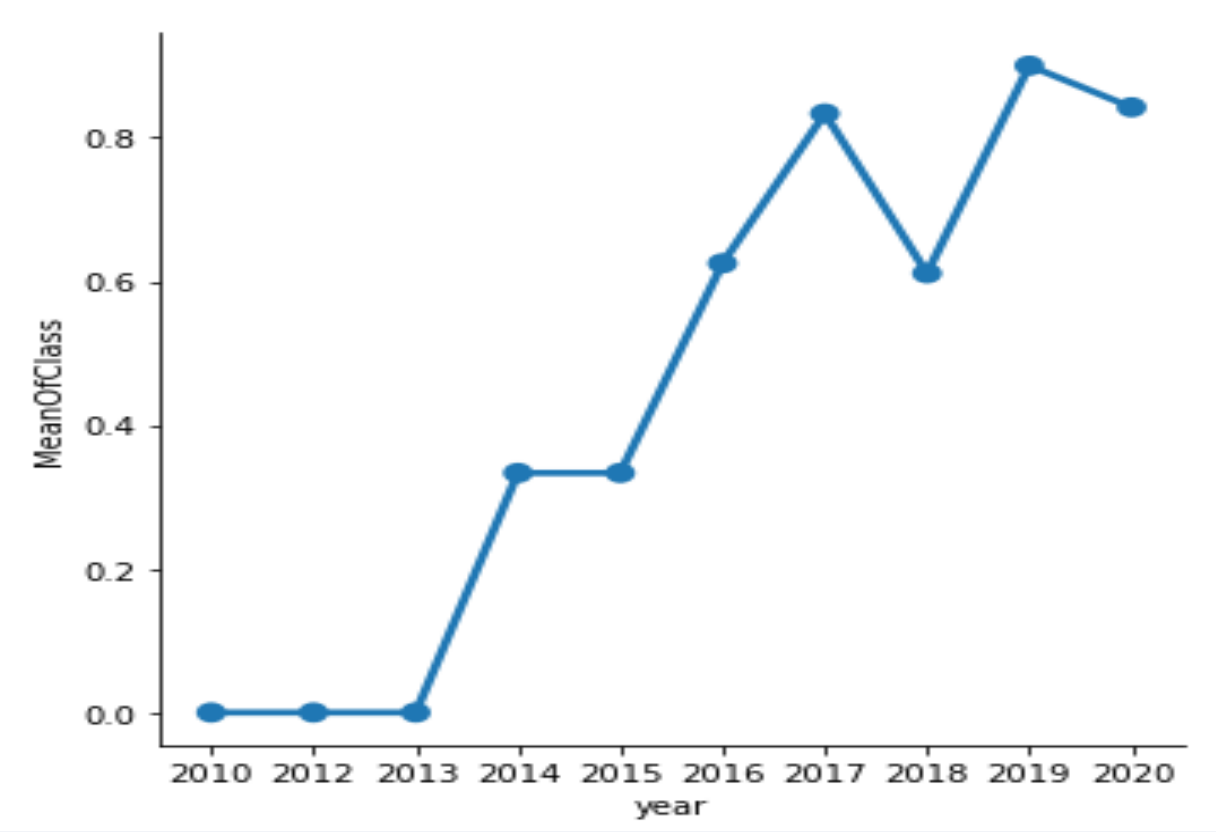
Payload vs. Orbit Type



Result:

1. With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.
2. GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.

Launch Success Yearly Trend



The sucess rate since 2013 kept increasing till 2020

All Launch Site Names

In [9]:
```
1  pd_conn=ibm_db_dbi.Connection(conn)
2  pd.read_sql("select distinct Launch_Site from SPACEX",pd_conn)
```

Out[9]:

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

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

Launch Site Names Begin with 'CCA'

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

In [28]:
```
pd_conn=ibm_db_dbi.Connection(conn)
pd.read_sql("select * from SPACEX where Launch_Site like 'CCA%' limit 5",pd_conn)
```

Out[28]:

	DATE	Time (UTC)	BOOSTER_VERSION	LAUNCH_SITE	PAYLOAD	PAYLOAD_MASS__KG_	ORBIT	CUSTOMER	MISSION_OUTCOME	LANDING_OUTCOME
0	2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
3	2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

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

Total Payload Mass

```
In [18]: 1 pd.read_sql("select sum(PAYLOAD_MASS__KG_) as totalpayloadmass from SPACEX where Customer = 'NASA (CRS)'",pd_conn)
```

Out[18]:

	TOTALPAYLOADMASS
0	45596

QUERY EXPLAINATION

Using the function SUM summates the total in the column PAYLOAD_MASS_KG_

The WHERE clause filters the dataset to only perform calculations on Customer NASA (CRS)

Average Payload Mass by F9 v1.1

```
In [20]:  1  pd.read_sql("select avg(PAYLOAD_MASS__KG_) as Average_Payload_Mass from SPACEX where Booster_Version = 'F9 v1.1'",pd_conn)
```

Out[20]:

	AVERAGE_PAYLOAD_MASS
0	2928

QUERY EXPLAINATION

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

First Successful Ground Landing Date

```
In [27]: 1 ibm_db.exec_immediate(conn, 'ALTER TABLE SPACEX RENAME COLUMN "Landing _Outcome" TO Landing_Outcome')

Out[27]: <ibm_db.IBM_DBStatement at 0x24659d31648>
```

```
In [29]: 1 pd.read_sql("select min(Date) as Min_Date from SPACEX where Landing_Outcome = 'Success (ground pad)'",pd_conn)
```

Out[29]:

	MIN_DATE
0	2015-12-22

QUERY EXPLAINATION

Using the function MIN works out the minimum date in the column Date
The WHERE clause filters the dataset to only perform
calculations on Landing_Outcome Success (ground pad)

Successful Drone Ship Landing with Payload between 4000 and 6000

In [30]:
```
pd.read_sql("select Booster_Version from SPACEX where \
            Landing_Outcome = 'Success (ground pad)' and \
            PAYLOAD_MASS__KG_ between 4000 and 6000",pd_conn)
```

Out[30]:

	BOOSTER_VERSION
0	F9 FT B1032.1
1	F9 B4 B1040.1
2	F9 B4 B1043.1

QUERY EXPLAINATION

Selecting only Booster_Version

The WHERE clause filters the dataset to Landing_Outcome = Success (ground pad)

The AND clause specifies additional filter conditions Payload_MASS_KG_ > 4000 AND Payload_MASS_KG_ < 6000

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

In [56]:
```
Task_7=pd.read_sql("select MISSION_OUTCOME,count(MISSION_OUTCOME) as count from SPACEX group by MISSION_OUTCOME",pd_conn)
Task_7["MISSION_OUTCOME_TYPE"]=Task_7["MISSION_OUTCOME"].apply(lambda x: 'Success' if 'Success' in x else 'Failure' )
Task_7
```

Out[56]:

	MISSION_OUTCOME	COUNT	MISSION_OUTCOME_TYPE
0	Failure (in flight)	1	Failure
1	Success	99	Success
2	Success (payload status unclear)	1	Success

Boosters Carried Maximum Payload

SQL QUERY

```
pd.read_sql("select Booster_Version,COUNT from (select Booster_Version,sum(PAYLOAD_MASS__KG_) as count from SPACEX group by Booster_Version) \
            where COUNT=( select max(count) from (select Booster_Version,sum(PAYLOAD_MASS__KG_) as count from SPACEX group by Booster_Version))",pd_conn)
```

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

Step1.
"select Booster_Version,COUNT from (select Booster_Version,sum(PAYLOAD_MASS__KG_) as count from SPACEX group by Booster_Version)
Step2.
where COUNT=(select max(count) from (select Booster_Version,sum(PAYLOAD_MASS__KG_) as count from SPACEX group by Booster_Version))",pd_conn)
Task_8

2015 Launch Records

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

In [69]:
```
pd.read_sql("select Landing_Outcome,Booster_Version,Launch_Site,Date from SPACEX where \
            Landing_Outcome like 'Failure (drone ship)' and \
            DATE between '2015-01-01' and '2015-12-31'",pd_conn)
```

Out[69]:

	LANDING_OUTCOME	BOOSTER_VERSION	LAUNCH_SITE	DATE
0	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40	2015-01-10
1	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40	2015-04-14

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

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

```
pd.read_sql("select Landing_Outcome,count(Landing_Outcome) as count from SPACEX \
                where DATE between '2010-06-04' and '2017-03-20' and (LANDING_OUTCOME='Failure
                (drone ship)' or  LANDING_OUTCOME='Success (ground pad)')\
                 group by Landing_Outcome order by COUNT desc  “,pd_conn)
```

	LANDING_OUTCOME	COUNT
0	Failure (drone ship)	5
1	Success (ground pad)	3

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

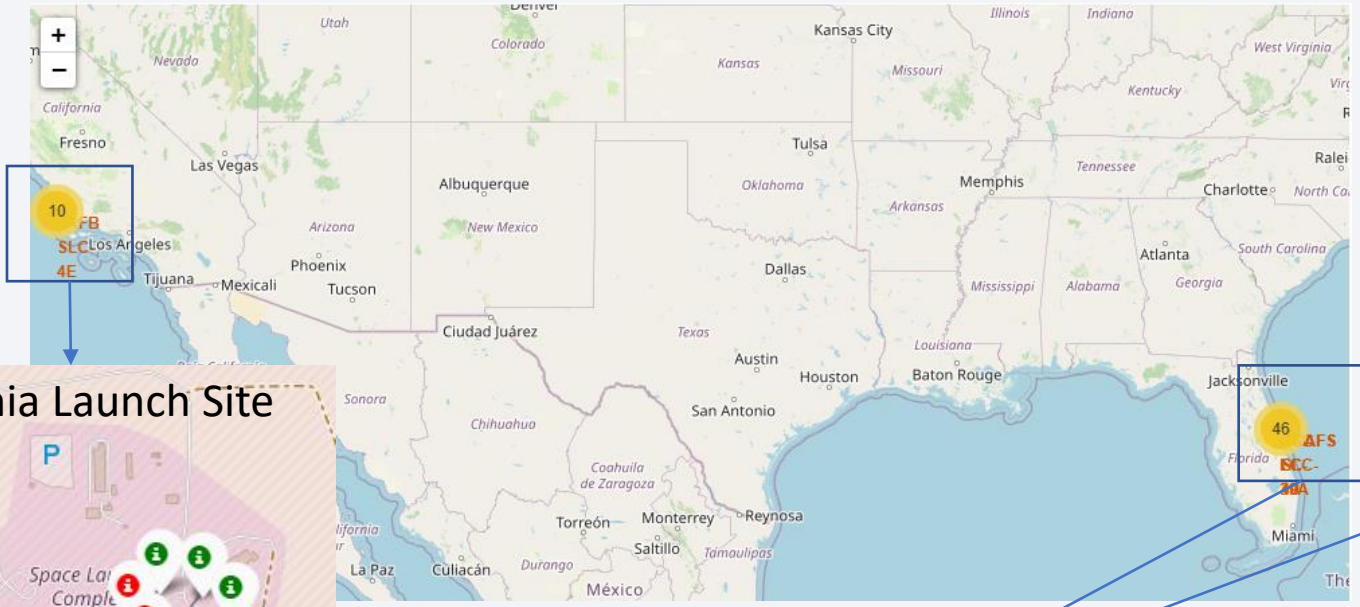
Section 4

Launch Sites Proximities Analysis

<Folium Map Screenshot 1>

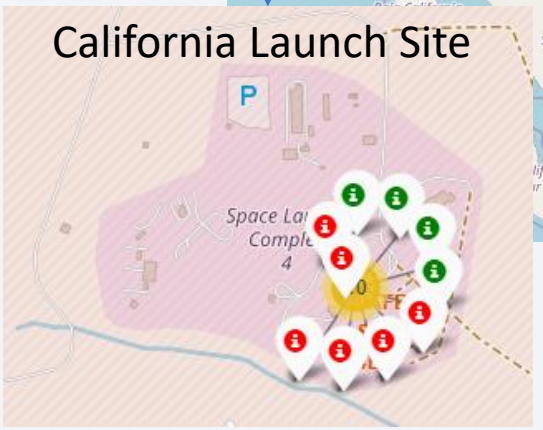
The SpaceX launch sites are in the United States of America coasts.
Florida and California

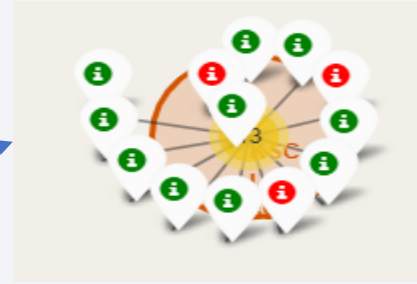
<Folium Map Screenshot 2>

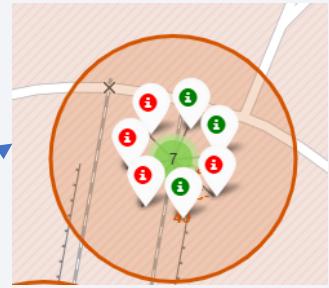


Green Marker shows successful Launches and Red Marker shows Failures

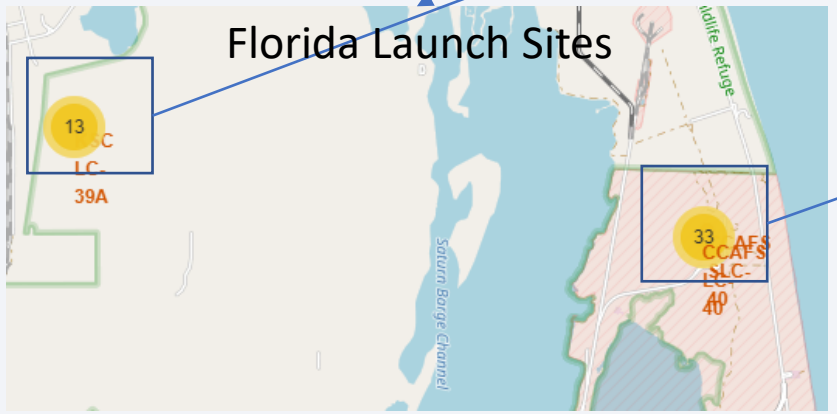
California Launch Site







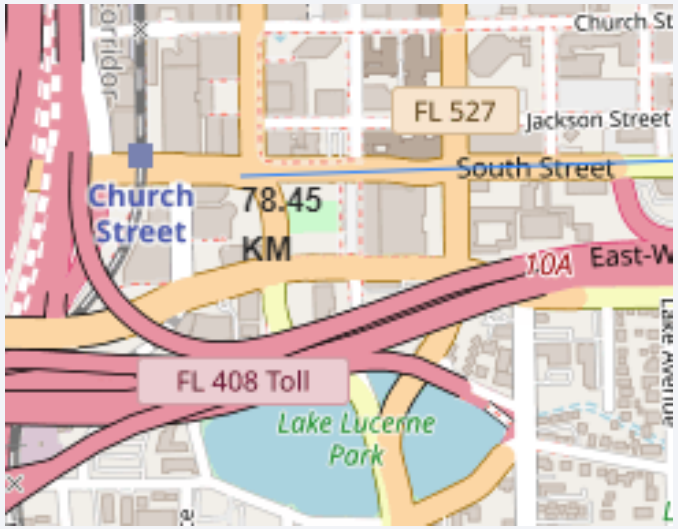
Florida Launch Sites



<Folium Map Screenshot 3>

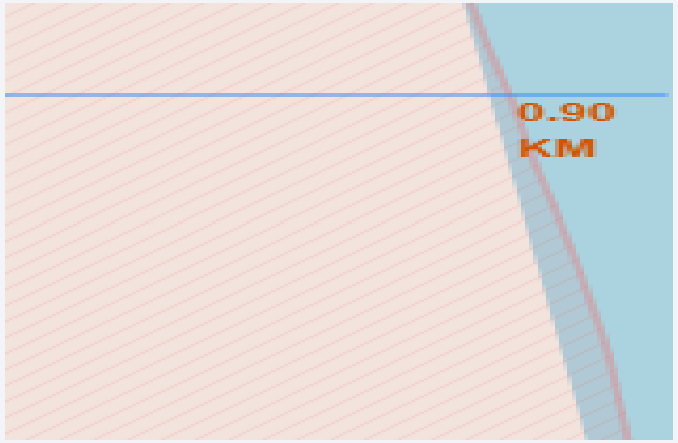
Distance to closest Highway

Distance to coast



Distance to Railway Station

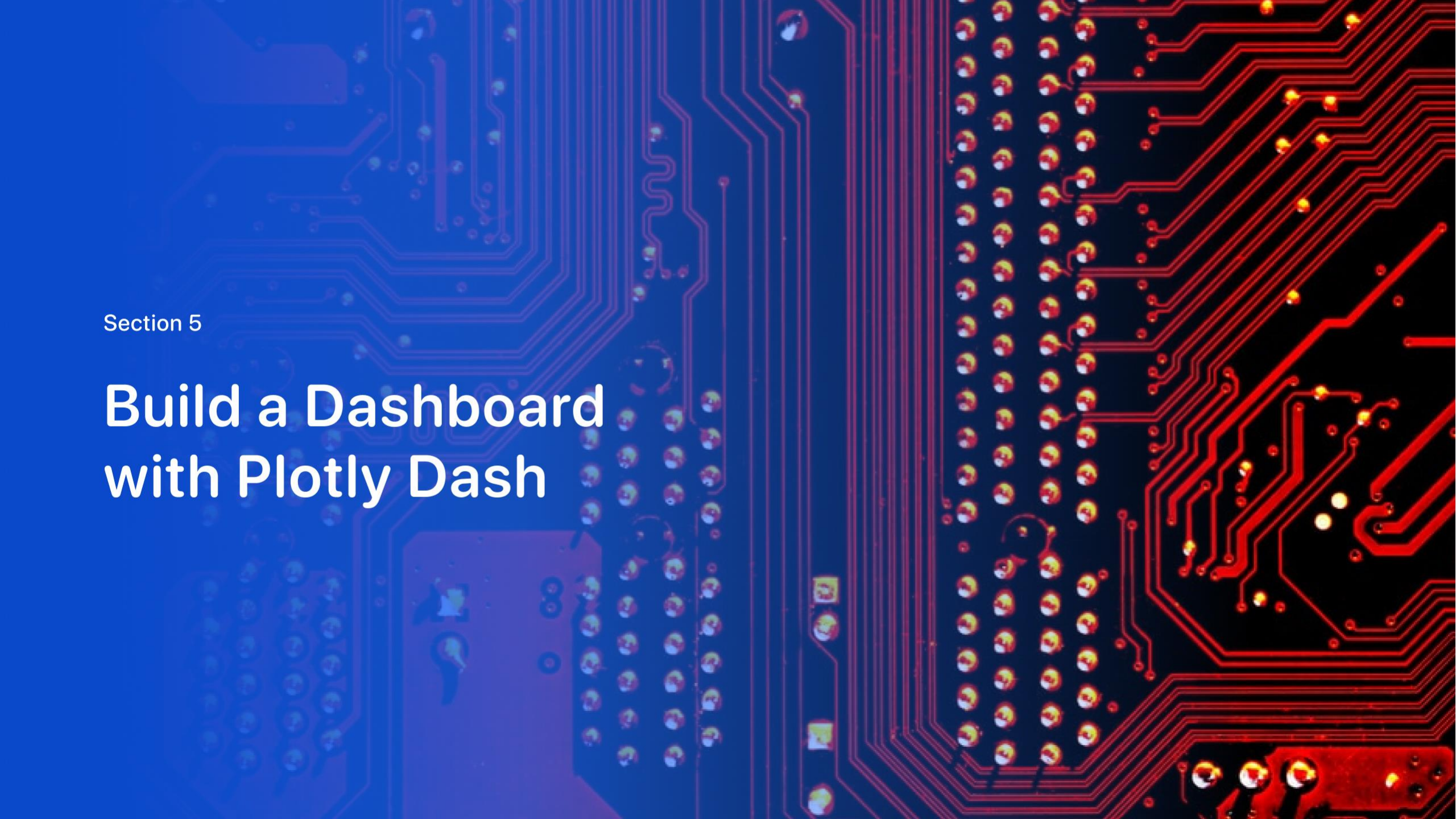
Distance to coastline



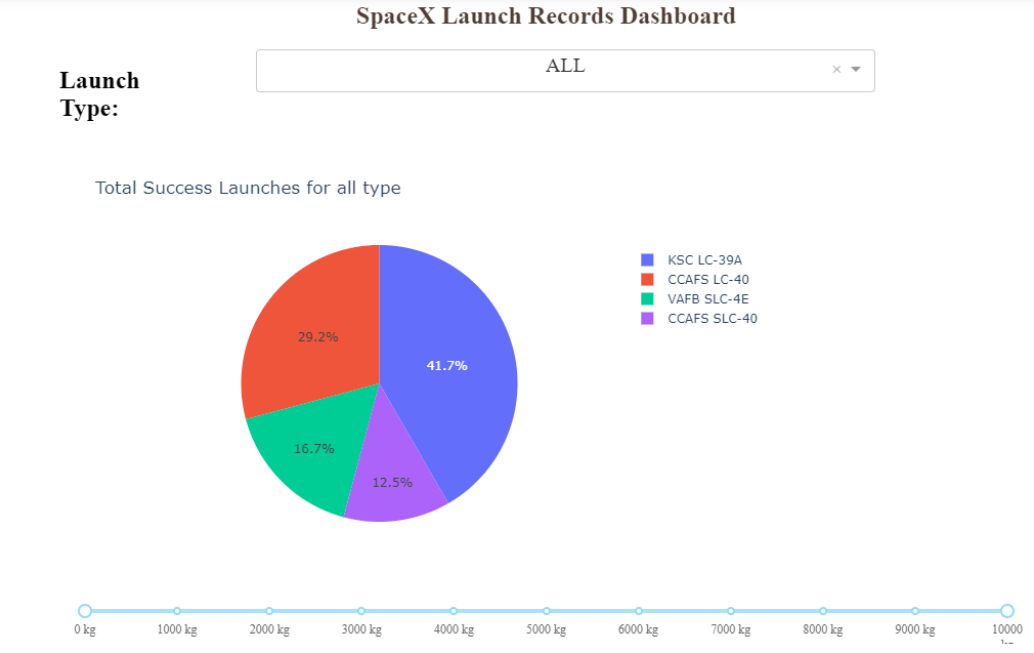
- Are launch sites in close proximity to railways?
  No
- Are launch sites in close proximity to highways?
  No
- Are launch sites in close proximity to coastline?
  Yes
- Do launch sites keep certain distance away from cities?
  Yes

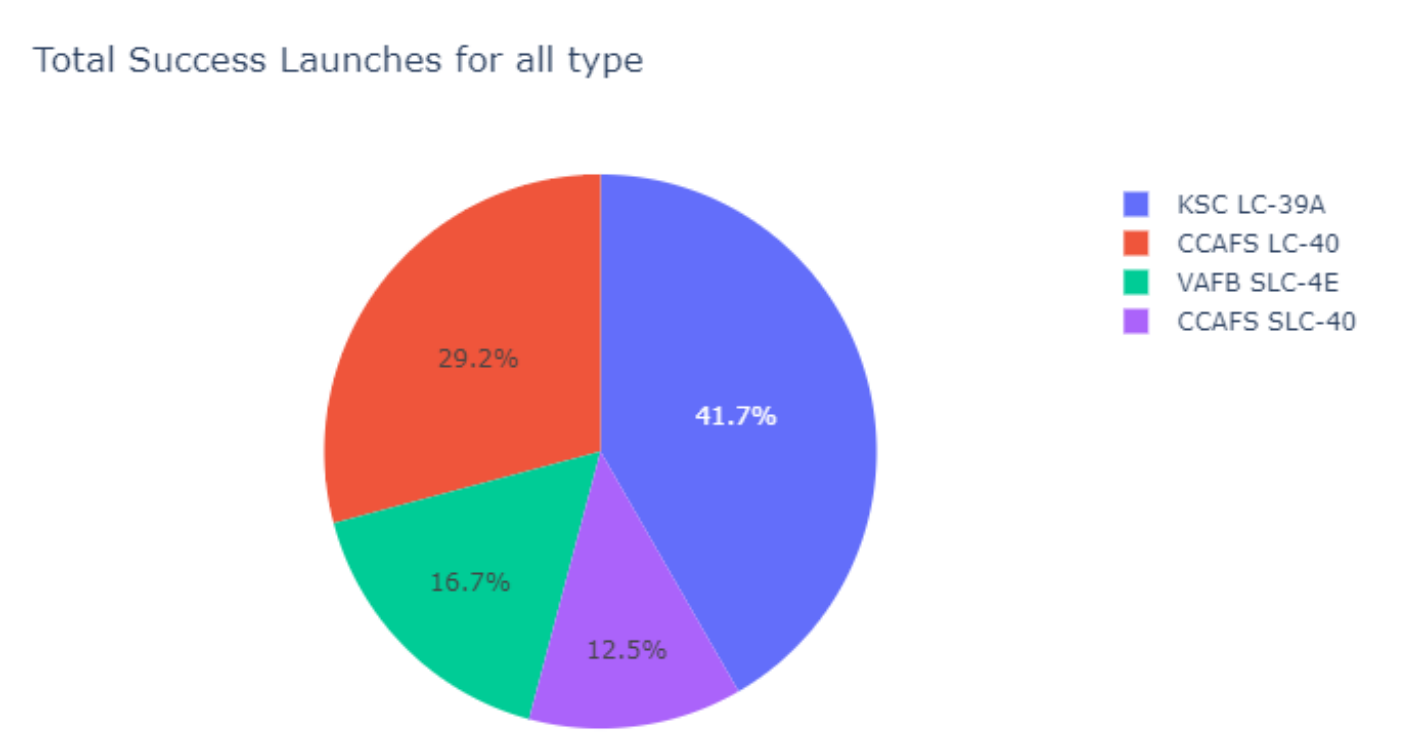
Section 5

Build a Dashboard with Plotly Dash



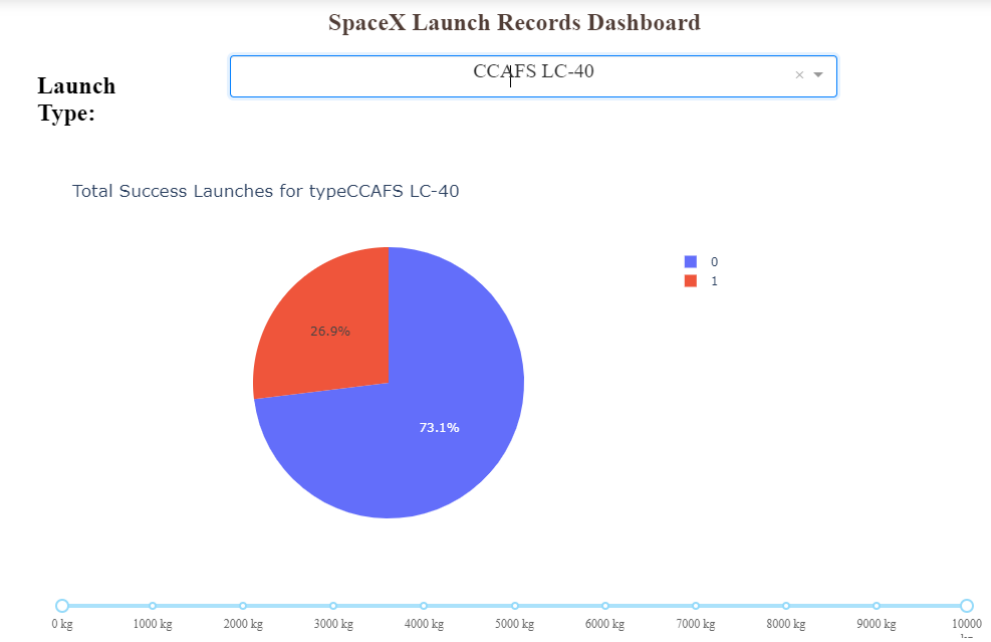
<Dashboard Screenshot 1>

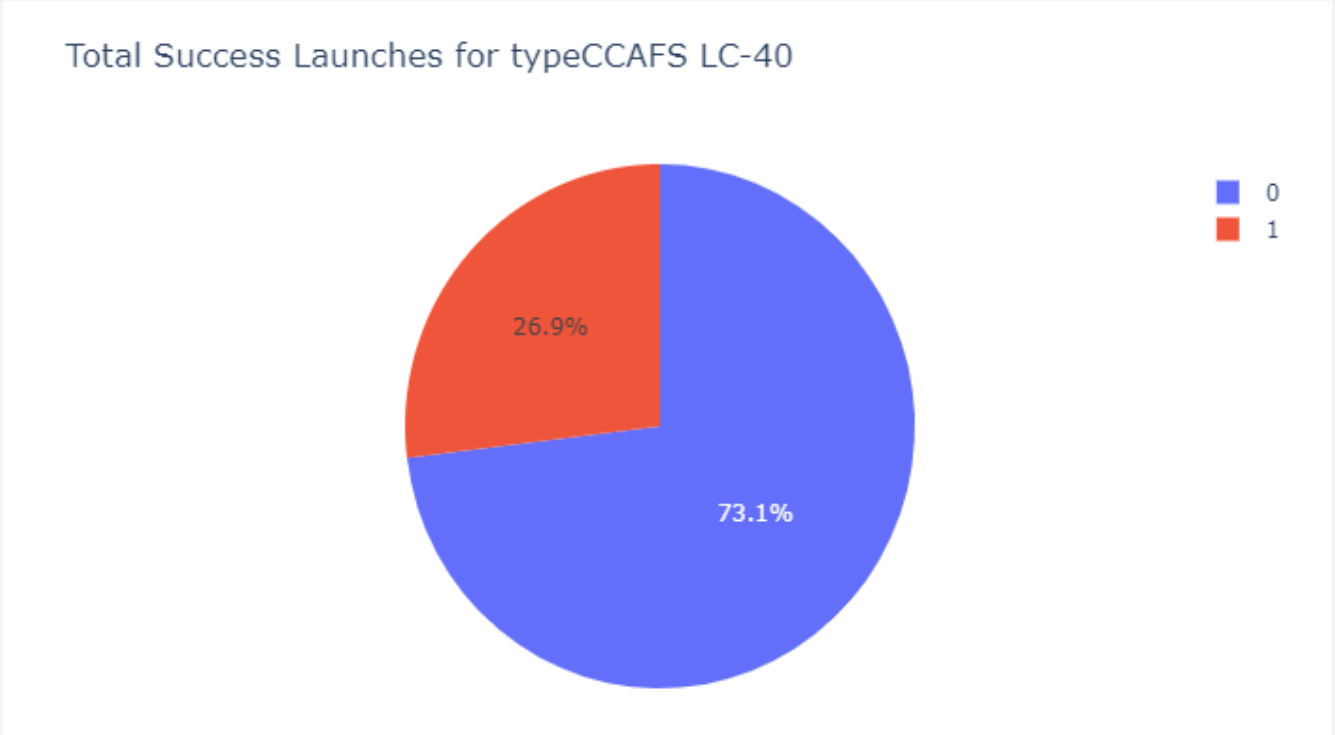




We can see that KSC LC-39A had the most successful launches from all the sites

<Dashboard Screenshot 2>



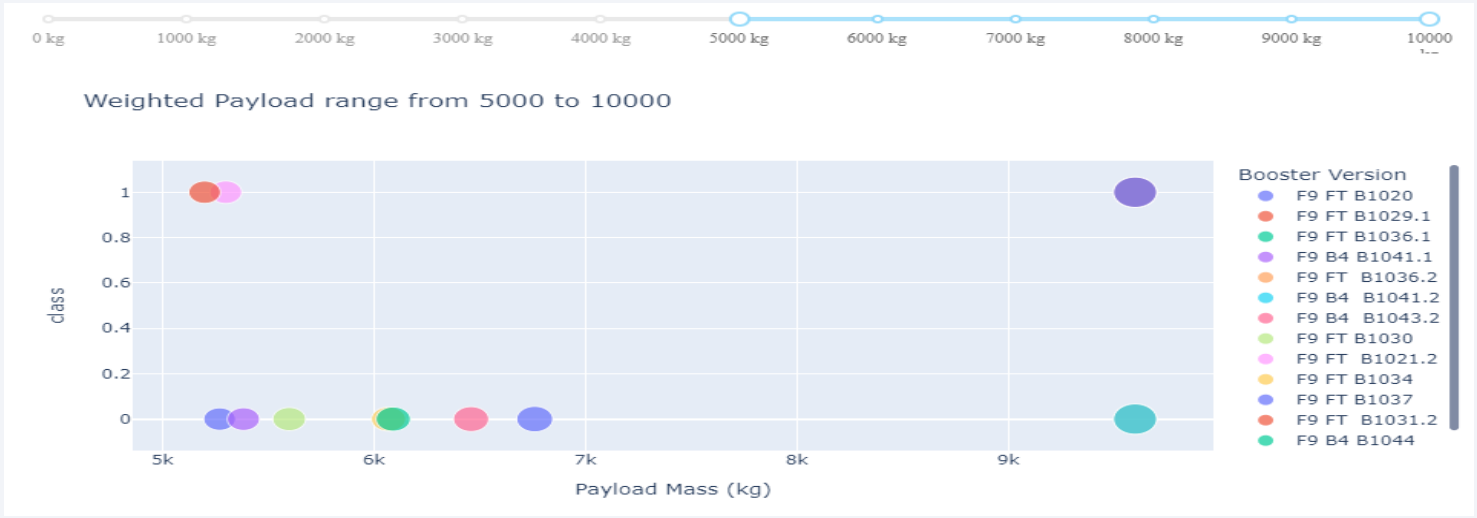


CCAFS LC-40 achieved a 73.1% success rate while getting a 26.9% failure rate

<Dashboard Screenshot 3>

Low Weighted Payload 0kg – 5000kg

High Weighted Payload 5000kg – 10000kg

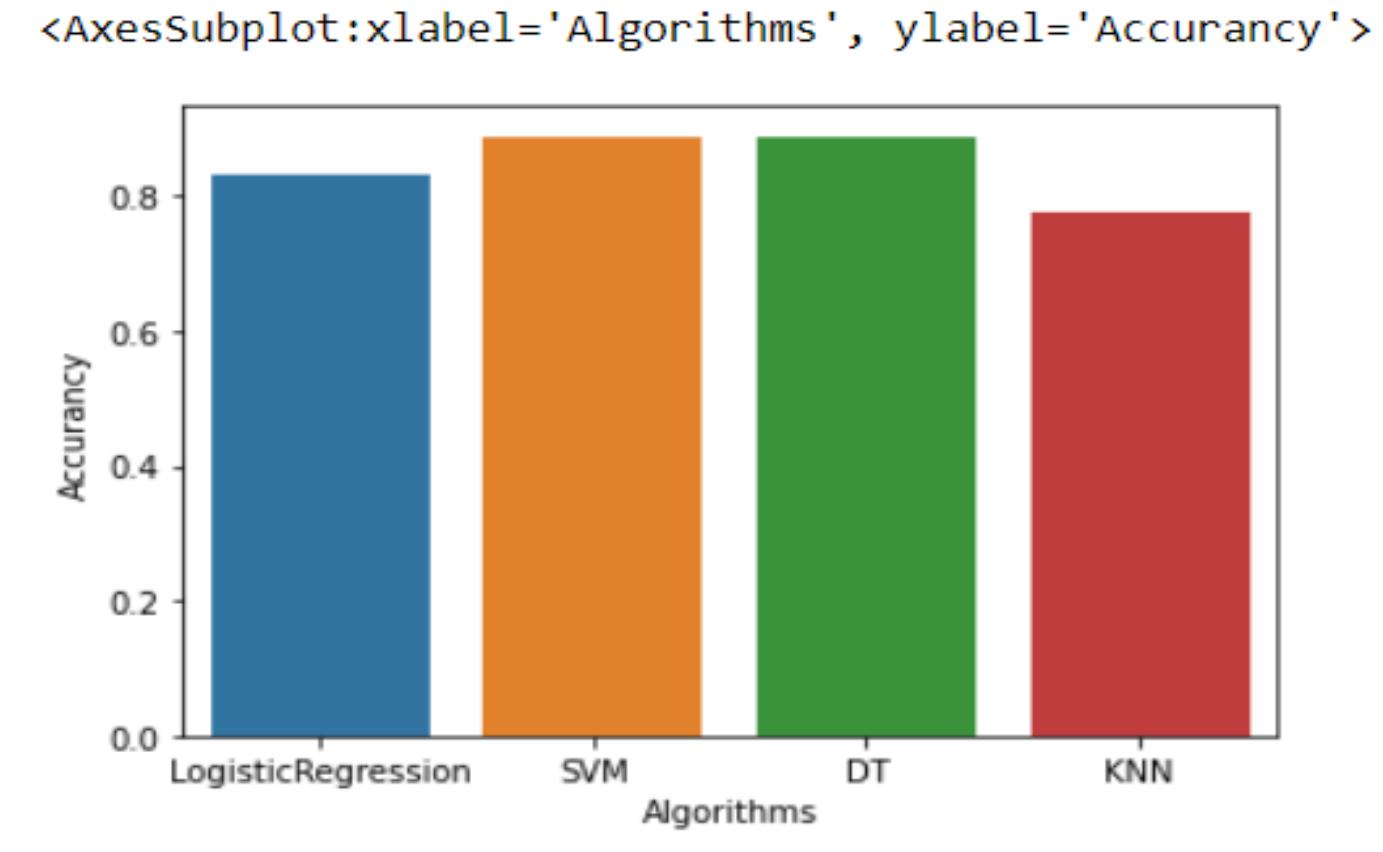


The success rates for low weighted payloads is higher than the heavy weighted payloads

Section 6

Predictive Analysis (Classification)

Classification Accuracy



	Algorithms	Accurancy
0	LogisticRegression	0.833333
1	SVM	0.888889
2	DT	0.888889
3	KNN	0.777778

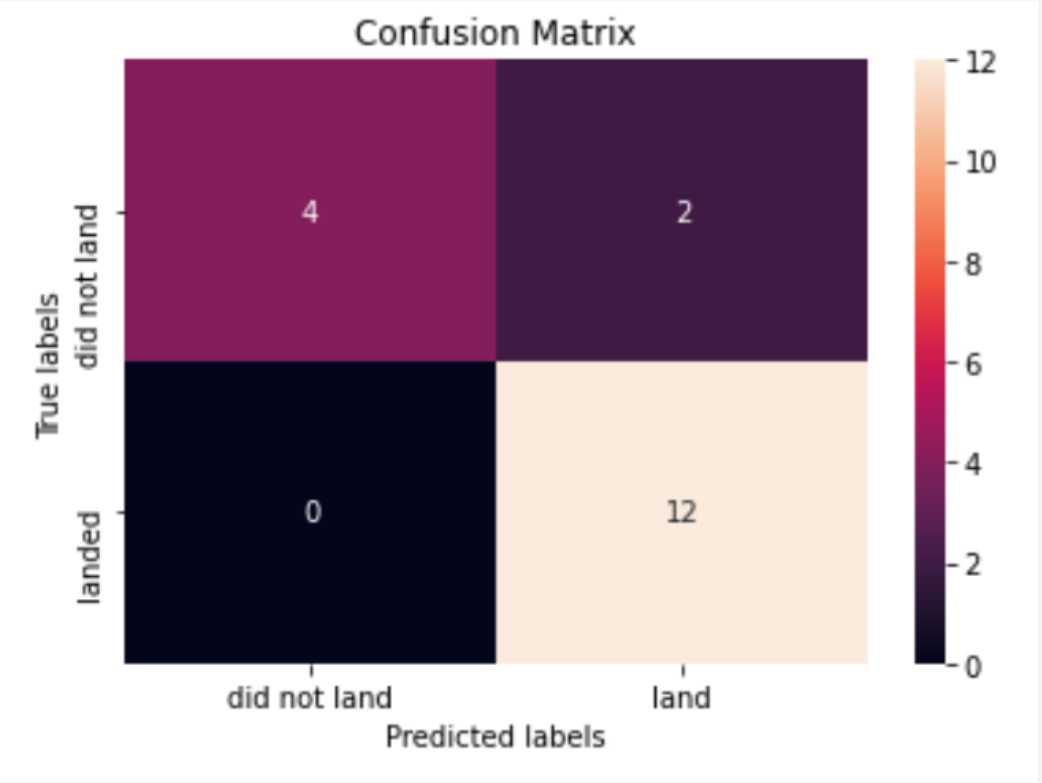
Best Algorithm is SVM with a score of 0.8888888888888888
Best Params is : {'C': 0.03162277660168379, 'gamma': 0.001, 'kernel': 'linear'}

Confusion Matrix

Best Algorithm is SVM with a score of 0.8888888888888888

#Correct(diagonal): 4+12=16
#Incorrect(Non diagonal): 2+0=2

Accuracy Score
= #Correct/(#Correct+ #Incorrect)
=16/(16+2)=0.8888



Conclusions

- Orbit GEO,HEO,SSO,ES-L1 has the best Success Rate (Page 20.)
- The success rates for SpaceX launches is directly proportional time in years they will eventually perfect the launches(Page 23.)
- We can see that KSC LC-39A had the most successful launches from all the sites(Page 39.)
- Low weighted payloads perform better than the heavier payloads(Page 41.)
- The SVM Algorithm is the best for Machine Learning for this dataset(Page 43.)

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!