Winning Space Race with Data Science

Libin Andrews
17.07.2022

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 Plolty Das
- Predictive analysis (Classification)

Summary of all results

- EDA Results
- Interactive Analysis
- Predictive Analysis

Introduction

- Project background and context

Space X advertise that it is reusing the first stage and because of that rocket launching is cheaper that the competitors.

- Problems you want to find answers

The project task is to predicting if the first stage of the SpaceX Falcon 9 rocket will land successfully.

Section 1

Methodology

Methodology

Executive Summary

Data collection methodology:

- Collected data with API using library called requests from website of spacex url.

Perform data wrangling

- Data Cleaning is done by removing the null 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
- LR,KNN, SVM, DT Models have been built and evaluated for the best classifier

Data Collection

The data was collected using various methods

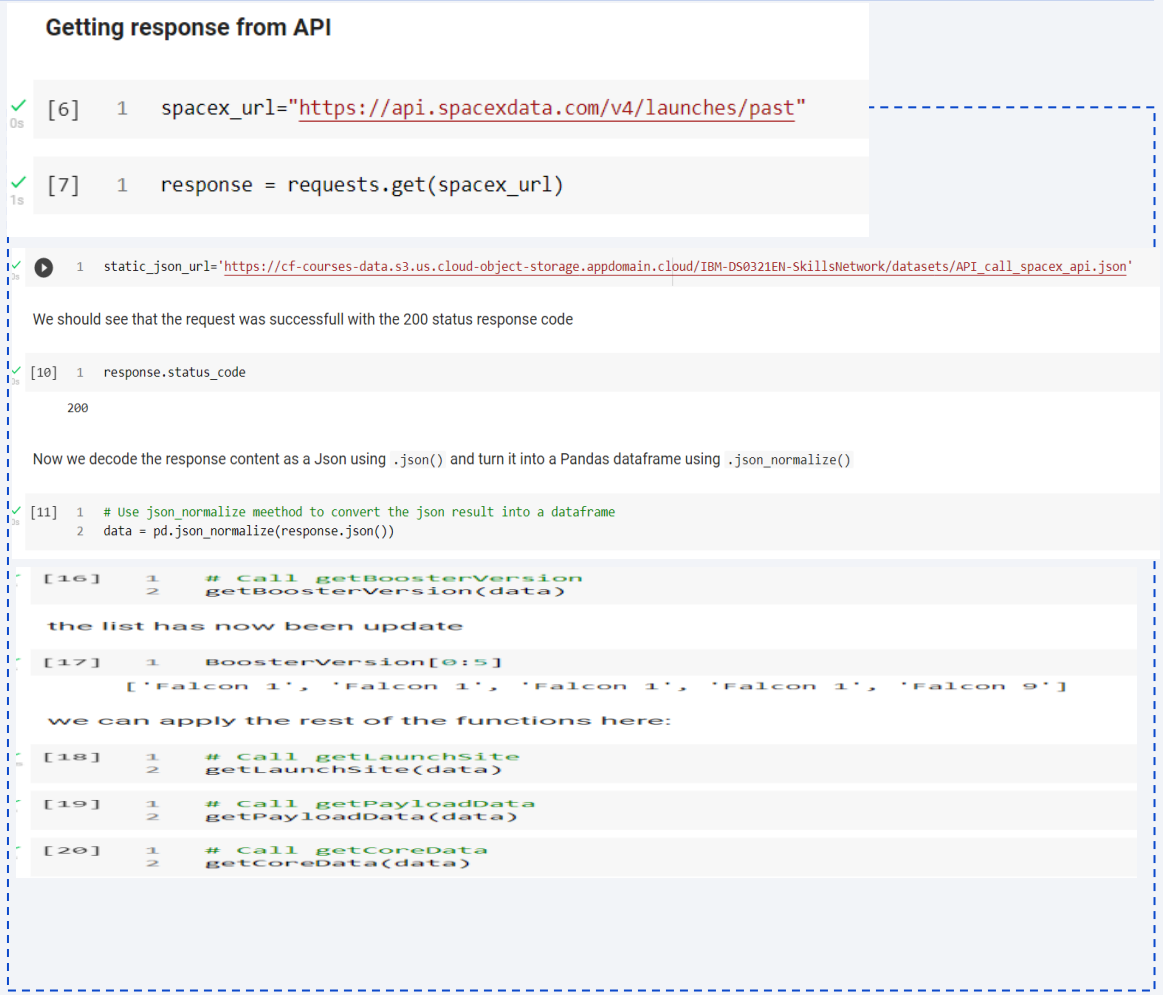
- Data collection was done using get request to the SpaceX API.
- Next, we decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().
- We then cleaned the data, checked for missing values and fill in missing values where necessary.
- In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
- The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis.

Data Collection – SpaceX API

- Data is collection with SpaceX Rest calls.

1. Getting response from API
2. Converting data to a pandas data
3. Apply Custom function to clean data
4. Assign list to dictionary then data frame.
5. Filter and export to CSV File

- Github link : https://github.com/libinandrews/Final_Project/blob/main/01_Data_Collection_API_Lab.ipynb



Data Collection – SpaceX API

- Data is collection with SpaceX Rest calls.

1. Getting response from API
2. Converting data to a pandas data
3. Apply Custom function to clean data
4. Assign list to dictionary then data frame.
5. Filter and export to CSV File

- Github link :

https://github.com/libinandrews/Final_Project/blob/main/01_Data_Collection_API_Lab.ipynb

4

```
[21]  1  launch_dict = {'FlightNumber': list(data['flight_number']),
      2  'Date': list(data['date']),
      3  'BoosterVersion':BoosterVersion,
      4  'PayloadMass':PayloadMass,
      5  'Orbit':Orbit,
      6  'LaunchSite':LaunchSite,
      7  'Outcome':Outcome,
      8  'Flights':Flights,
      9  'GridFins':GridFins,
     10  'Reused':Reused,
     11  'Legs':Legs,
     12  'LandingPad':LandingPad,
     13  'Block':Block,
     14  'ReusedCount':ReusedCount,
     15  'Serial':Serial,
     16  'Longitude': Longitude,
     17  'Latitude': Latitude}
     18
```

5

```
✓ [29]  1  data_falcon9.to_csv('dataset_part_1.csv', index=False)
0s
```

Data Collection - Scraping

- Web scrapping from Wikipedia

1. Getting response from HTML
2. Creating BeautifulSoup Object
3. Finding Tables
4. Getting column name
5. Creation of dictionary
6. Appending data to keys
7. Converting dictionary to dataframe
8. Dataframe to csv

- Github :

https://github.com/libinandrews/Final_Project/blob/main/02_Data_Collection_with_Web_Scraping_lab.ipynb

1

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

2

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response, 'html.parser')
```

3

```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all("table")
```

4

In [12]:

```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
temp = soup.find_all('th')
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
```

Check the extracted column names

In [13]:

```
print(column_names)
```

```
['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome', 'Flight No.', 'Date and time ( )',
'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome', 'Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload
mass', 'Orbit', 'Customer', 'Launch outcome', 'Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch
outcome', 'N/A', 'Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome', 'Flight No.', 'Dat
e and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome', 'Flight No.', 'Date and time ( )', 'Launch site', 'Pa
yload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome', 'FH 2', 'FH 3', 'Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload ma
ss', 'Orbit', 'Customer', 'Launch outcome', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome', 'Date
and time ( )', 'Launch site', 'Payload', 'Orbit', 'Customer', 'Date and time ( )', 'Launch site', 'Payload', 'Orbit', 'Customer', 'Date and time ( )',
'Launch site', 'Payload', 'Orbit', 'Customer', 'Date and time ( )', 'Launch site', 'Payload', 'Orbit', 'Customer', 'Demo flights', 'logistics', 'Crewed
missions', 'Commercial satellites', 'Scientific satellites', 'Military satellites', 'Rideshares', 'Current', 'In development', 'Retired', 'Cancelled',
'Spacecraft', 'Cargo', 'Crewed', 'Test vehicles', 'Current', 'Retired', 'Unflown', 'Orbital', 'Atmospheric', 'Landing sites', 'Other facilities', 'Supp
ort', 'Contracts', 'R&D programs', 'Key people', 'Related', 'General', 'General', 'People', 'Vehicles', 'Launches by rocket type', 'Launches by spacepo
rt', 'Agencies, companies and facilities', 'Other mission lists and timelines']
```

Data Collection - Scraping

- Web scrapping from Wikipedia

1. Getting response from HTML
2. Creating BeautifulSoup Object
3. Finding Tables
4. Getting column name
5. Creation of dictionary
6. Appending data to keys
7. Converting dictionary to dataframe
8. Dataframe to csv

- Github :

https://github.com/libinandrews/Final_Project/blob/main/02_Data_Collection_with_Web_Scraping_lab.ipynb

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 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']=[]
```

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

In [16]:

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

8

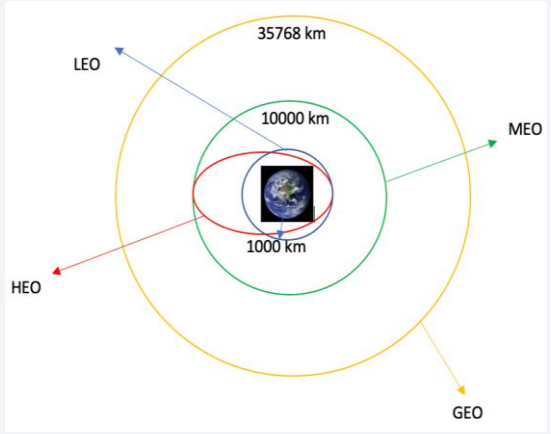
We can now export it to a **CSV** for the next section, but to make the answers consistent and in case you have difficulties finishing this lab.

Following labs will be using a provided dataset to make each lab independent.

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

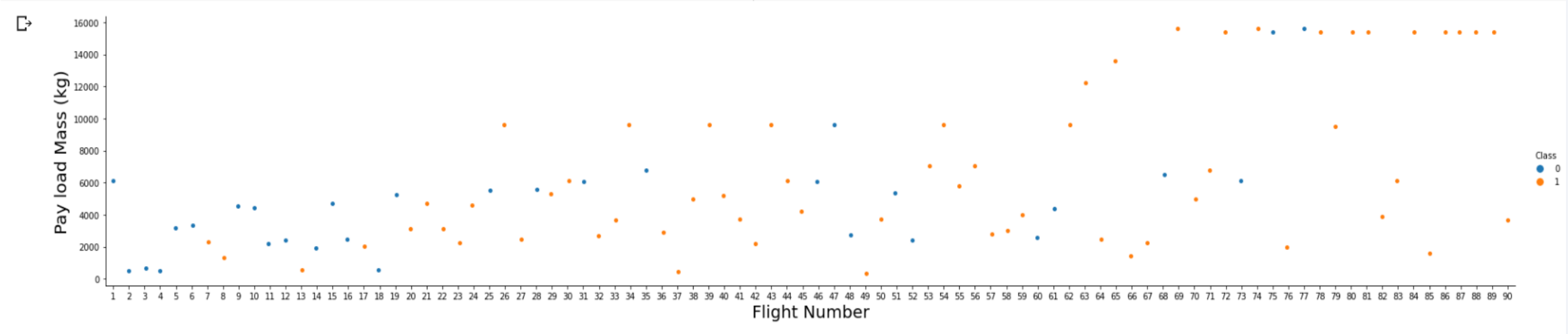
Data Wrangling

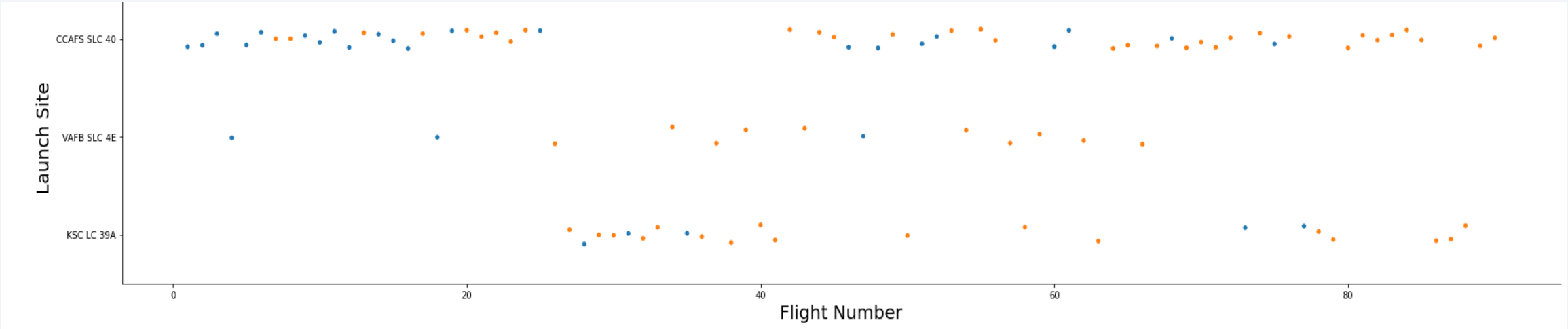
- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv



Github Link : https://github.com/libinandrews/Final_Project/blob/Master/03_Data_wrangling.ipynb

EDA with Data Visualization

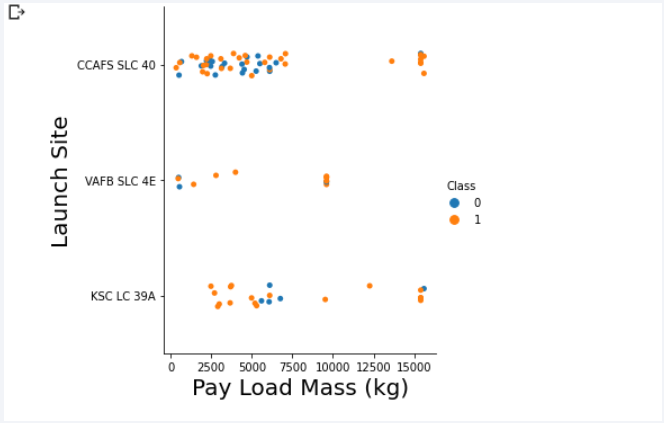


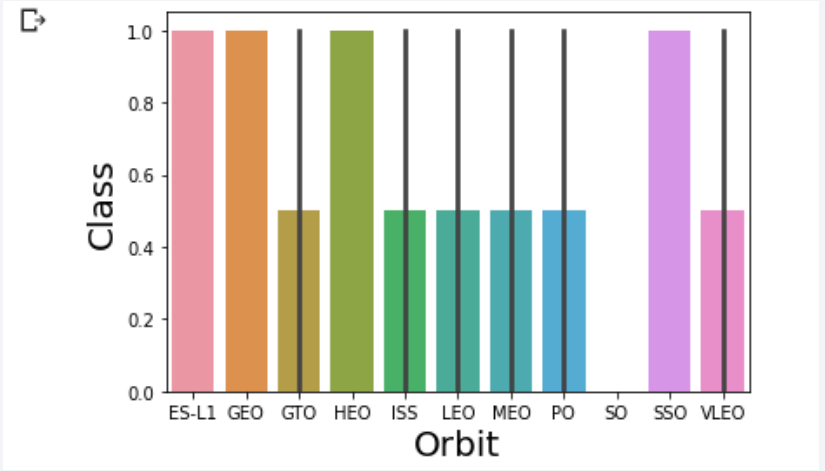


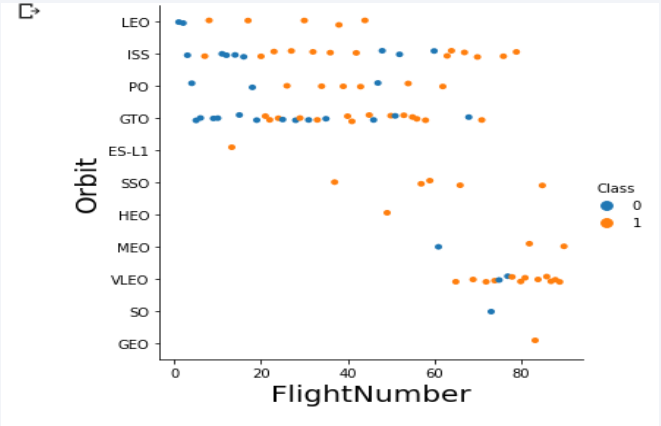
Github Link : https://github.com/libinandrews/Final_Project/blob/Master/04_Eda_With_Data_Visualization.ipynb

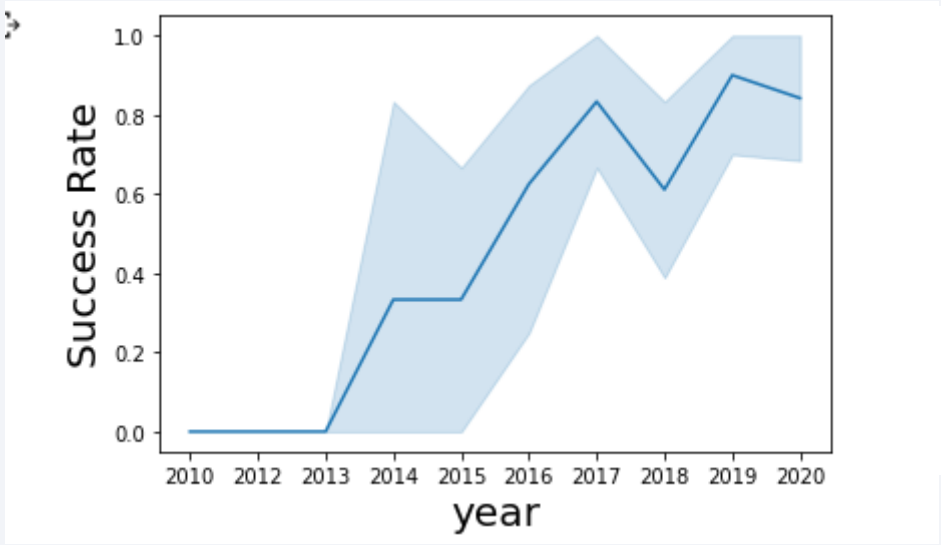
EDA with Data Visualization

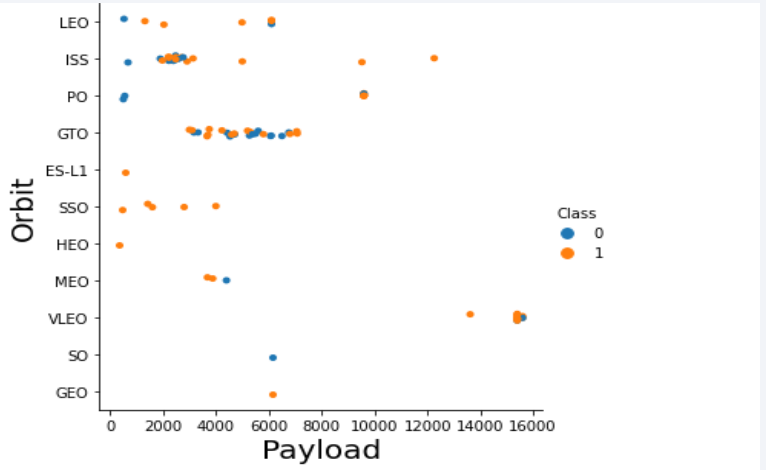
We explored the data by visualizing the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend.











Github Link : https://github.com/libinandrews/Final_Project/blob/Master/04_Eda_With_Data_Visualization.ipynb

EDA with SQL

- We loaded the SpaceX dataset into a PostgreSQL database without leaving the notebook.
- We applied EDA with SQL to get more information of the data and wrote queries to find out for instance:
  - The names of unique launch sites in the space mission.
  - The total payload mass carried by boosters launched by NASA (CRS)
  - The average payload mass carried by booster version F9 v1.1
  - The total number of successful and failure mission outcomes
  - The failed landing outcomes in drone ship, their booster version and launch site names.

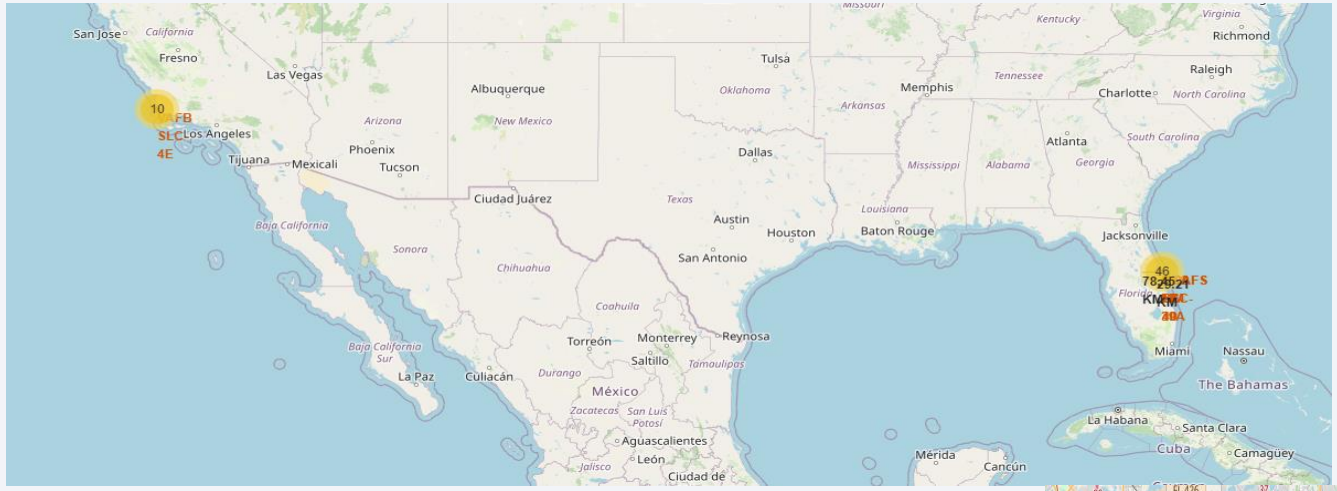
Github Link : https://github.com/libinandrews/Final_Project/blob/Master/05_SQL_Notebook_for_PeerAssignment.ipynb

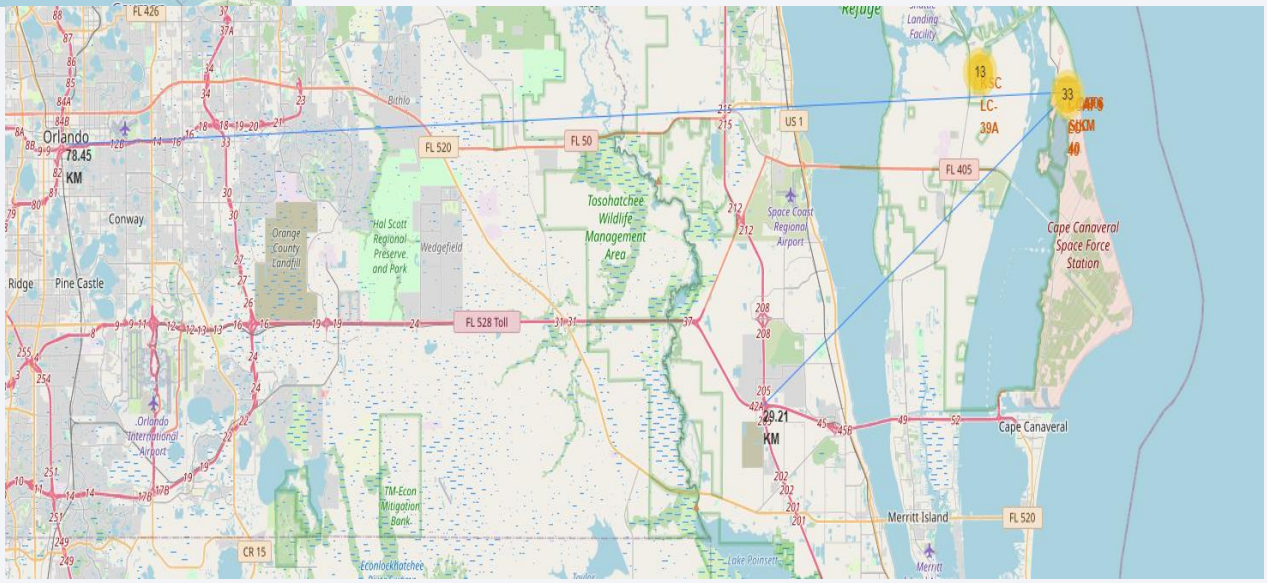
Build an Interactive Map with Folium

- We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the distances between a launch site to its proximities. We answered some question for instance:
  - Are launch sites near railways, highways and coastlines.
  - Do launch sites keep certain distance away from cities.

Github Link : https://github.com/libinandrews/Final_Project/blob/Master/06_Folium.ipynb

Build an Interactive Map with Folium

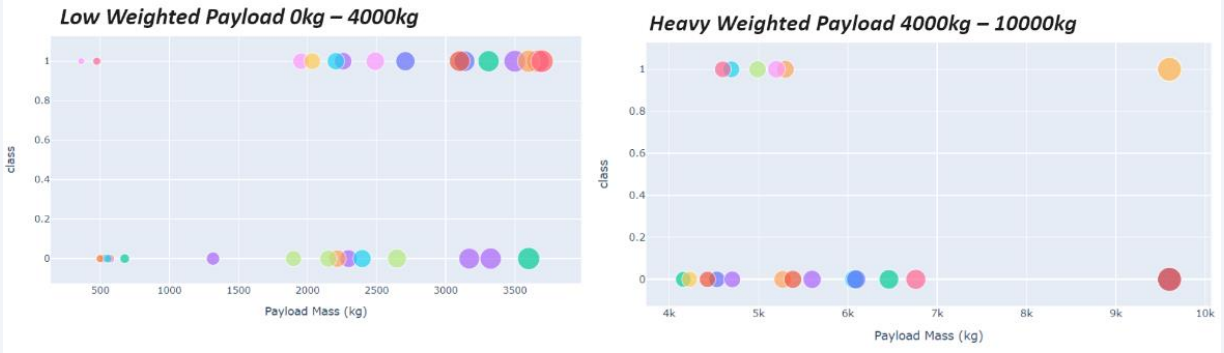




Github Link : https://github.com/libinandrews/Final_Project/blob/Master/06_Folium.ipynb

Build a Dashboard with Plotly Dash

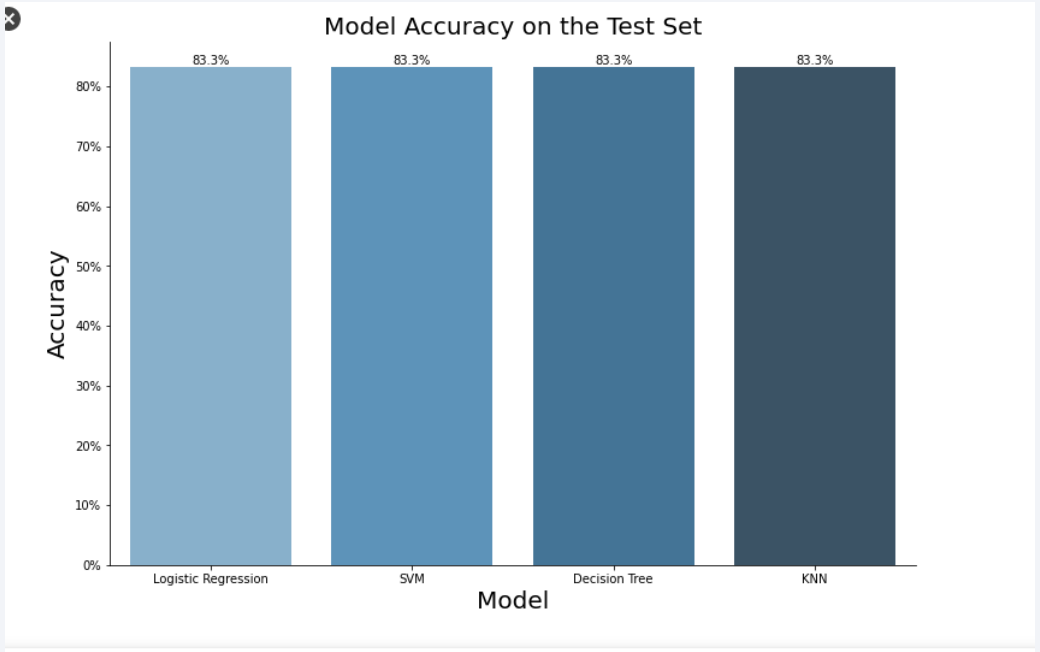
- We built an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.



Github Link : https://github.com/libinandrews/Final_Project/blob/Master/07_Machine_Learning_Prediction_lab.ipynb

Predictive Analysis (Classification)

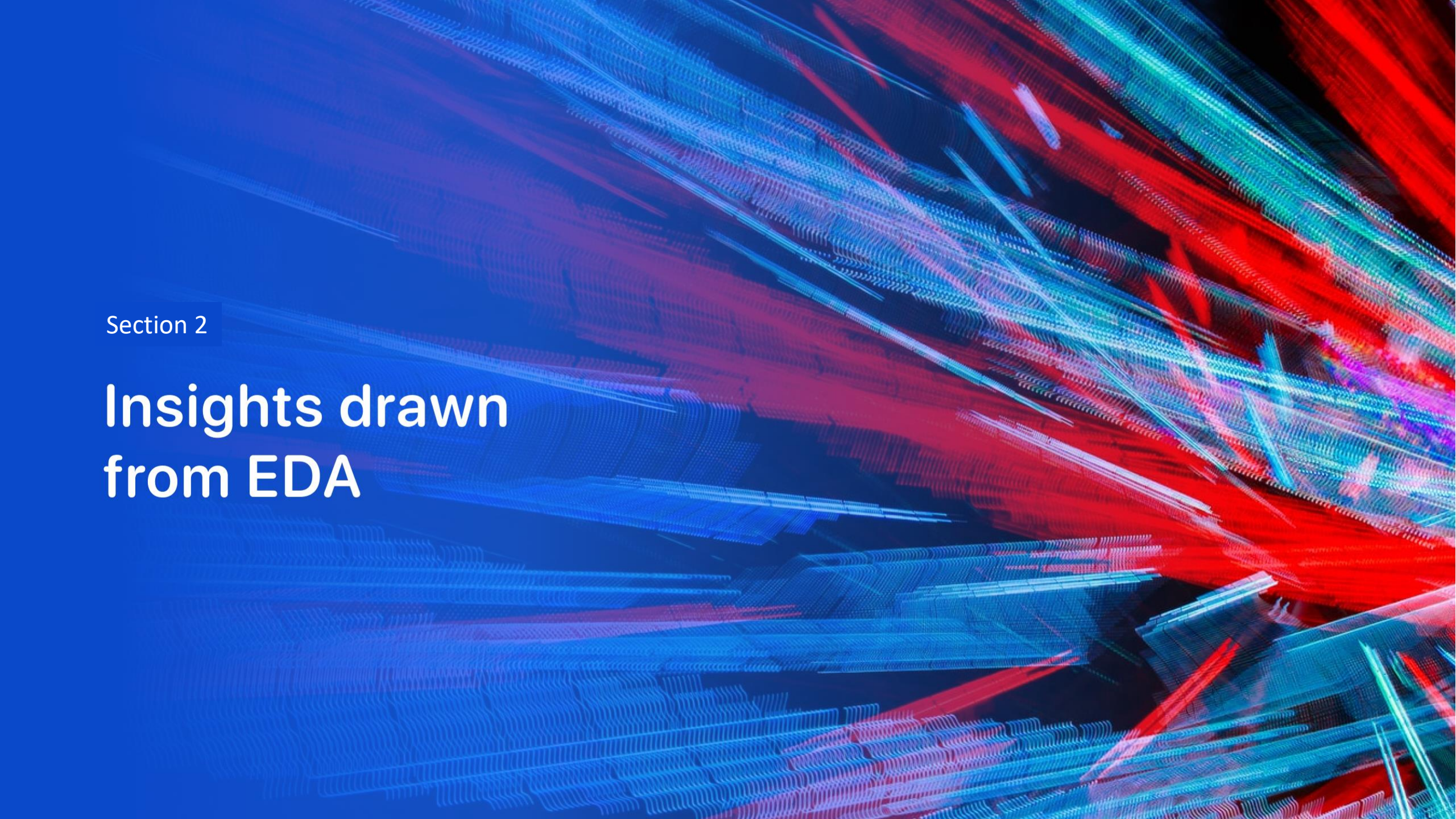
- We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- The SVM, KNN and Logistic Regression model achieved the highest accuracy while SVM perform the best in terms of Area under curve..



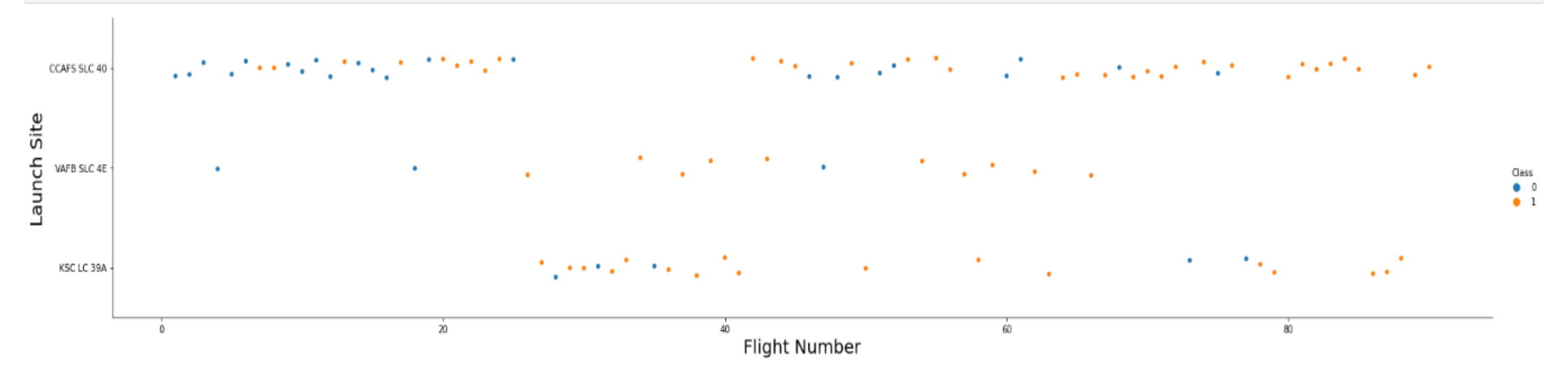
Github Link : https://github.com/libinandrews/Final_Project/blob/Master/07_Machine_Learning_Prediction_lab.ipynb

Results

- The SVM, KNN and Logicistic Regression models are the best in terms of prediction accuracy for this dataset.
- Low weight payloads perform better than the heavier payloads
- The success rates for SpaceX launches is directly proportional time in years they will eventually perfect the launches.
- KSC LC 39A had the most successful launches from all the sites.
- Orbit GEO, HEO,SSO,ES L1 ha the best success rate.


Section 2
Insights drawn from EDA

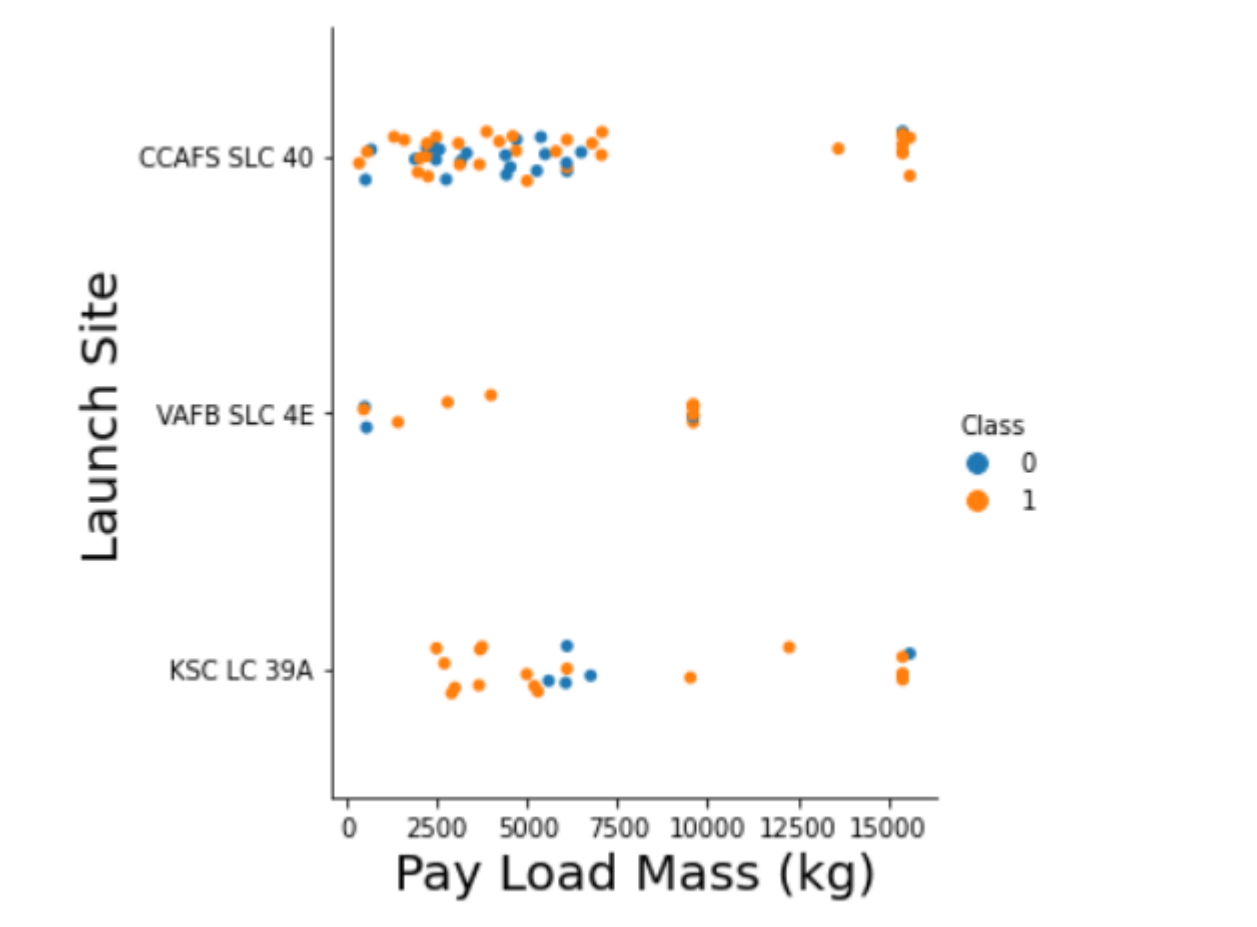
Flight Number vs. Launch Site



- From the plot, we found that the larger the flight amount at a launch site, the greater the success rate at a launch site.

Github Link : https://github.com/libinandrews/Final_Project/blob/Master/04_Eda_With_Data_Visualization.ipynb

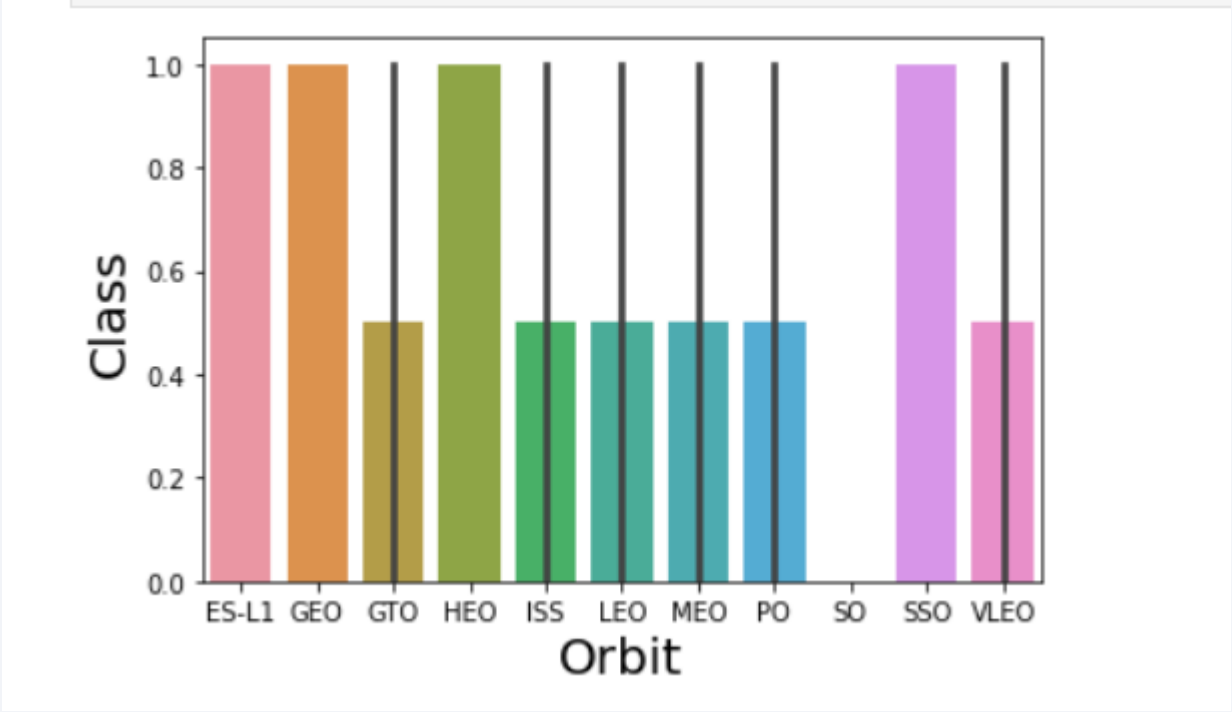
Payload vs. Launch Site



The Majority of Ipay Loads with lower Mass have been launched from CCAFS SLC40

Github Link : https://github.com/libinandrews/Final_Project/blob/Master/04_Eda_With_Data_Visualization.ipynb

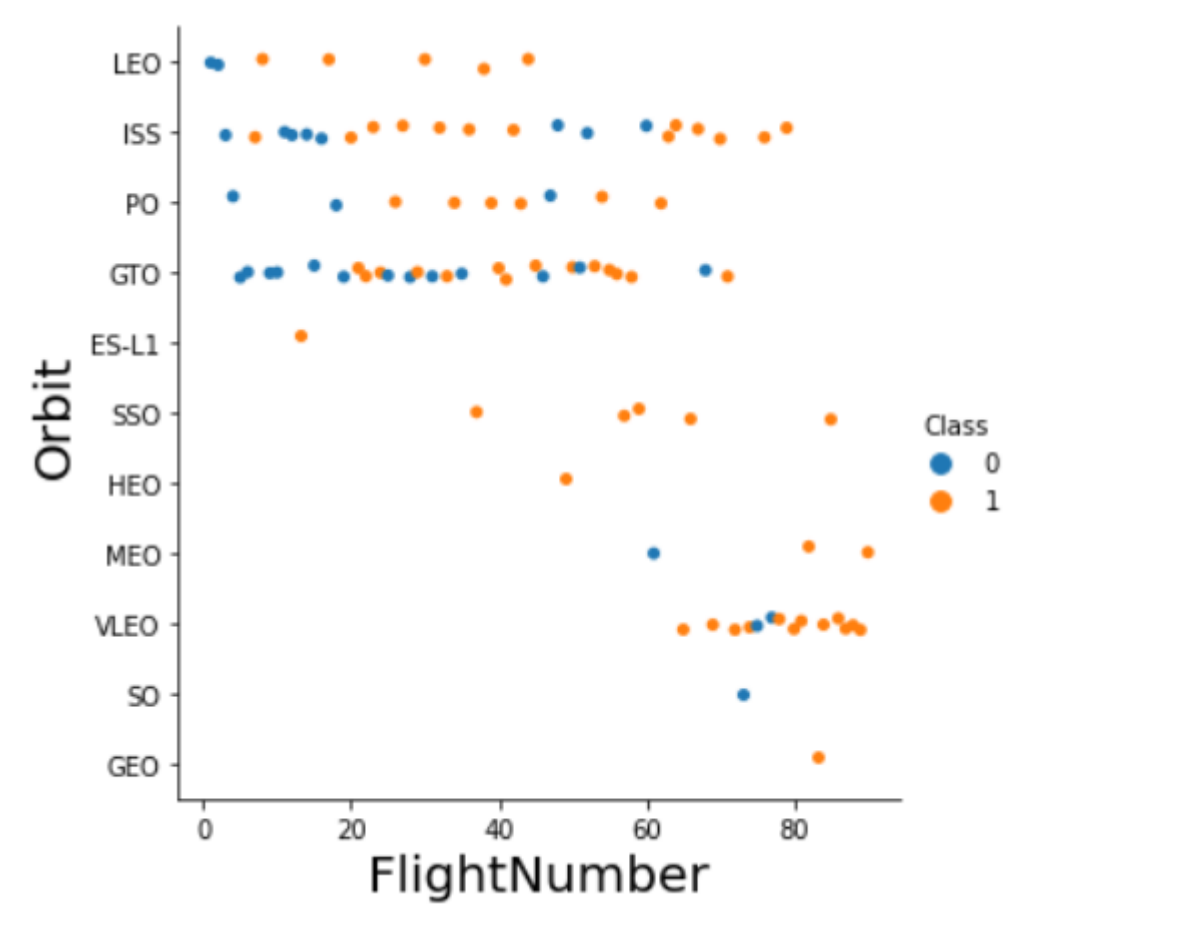
Success Rate vs. Orbit Type



Orbit of ES L1, GEO, HEO, SSO have high success rate.

Github Link : https://github.com/libinandrews/Final_Project/blob/Master/04_Eda_With_Data_Visualization.ipynb

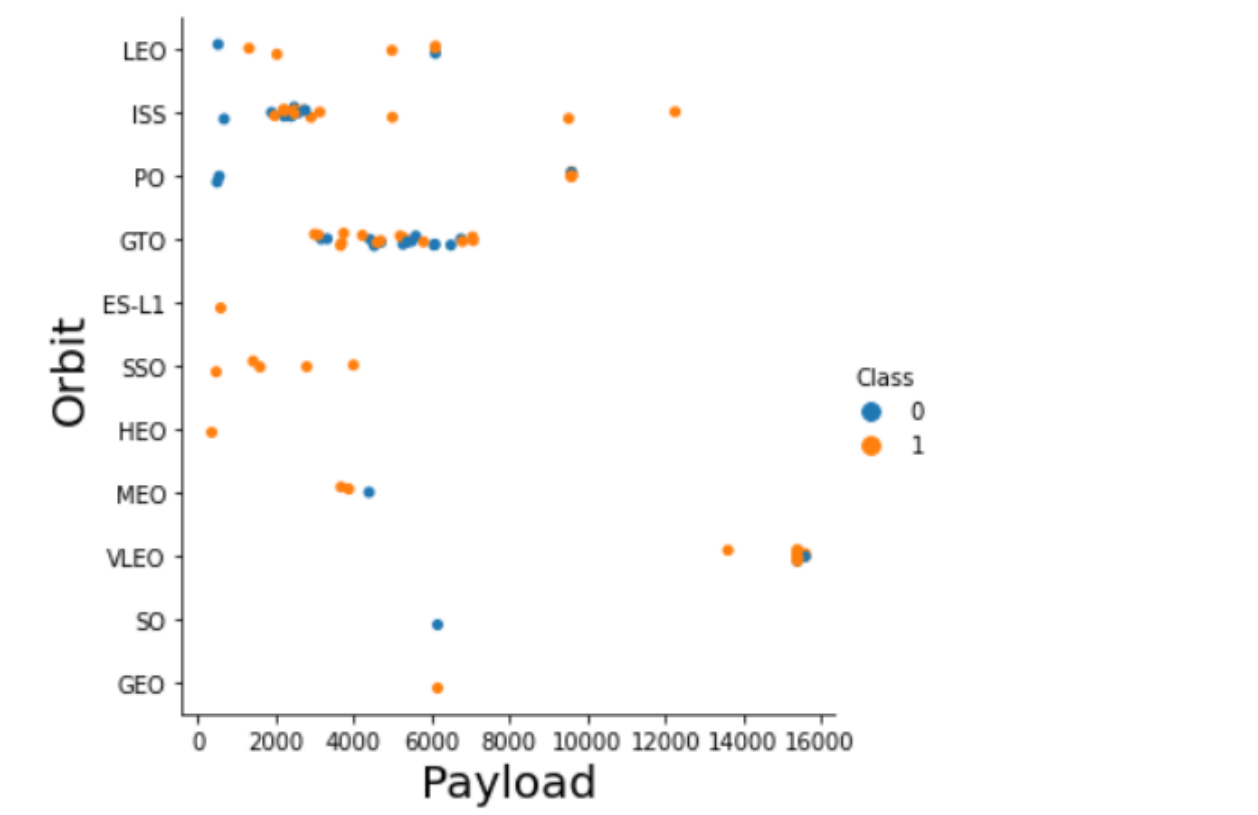
Flight Number vs. Orbit Type



A trend can be observed of shifting to VLEO in recent years

Github Link : https://github.com/libinandrews/Final_Project/blob/Master/04_Eda_With_Data_Visualization.ipynb

Payload vs. Orbit Type

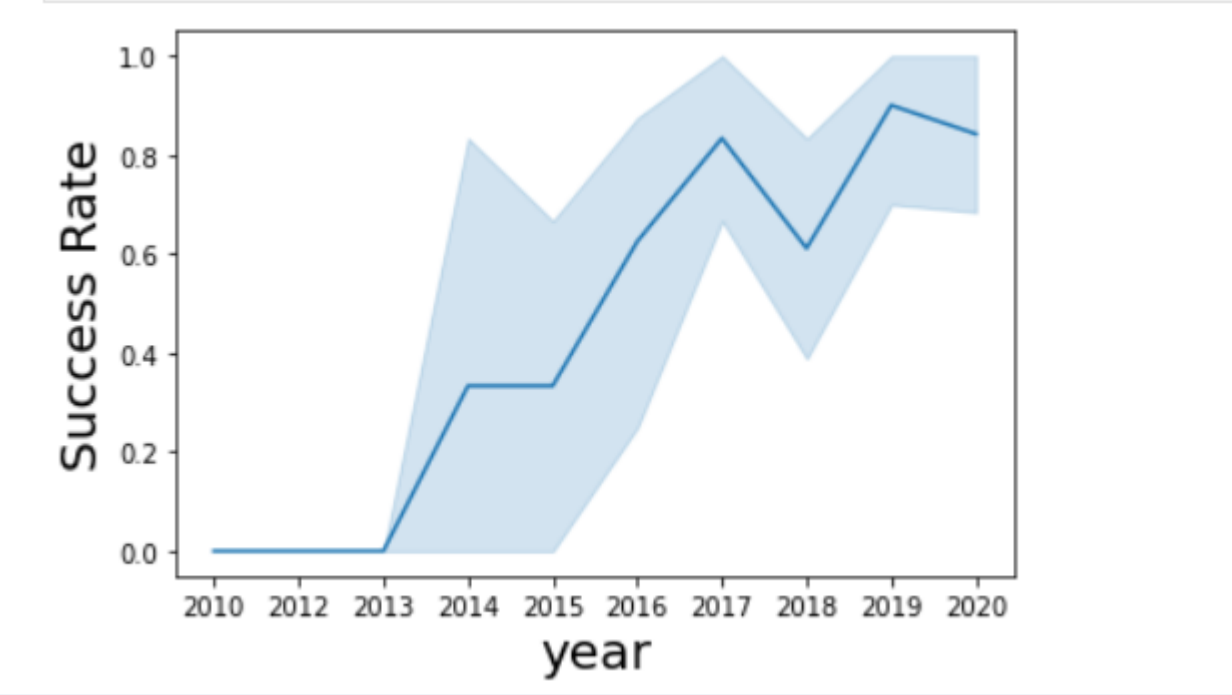


We can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.

Github Link : https://github.com/libinandrews/Final_Project/blob/Master/04_Eda_With_Data_Visualization.ipynb

Launch Success Yearly Trend

- From the plot, we can observe that success rate since 2013 kept on increasing till 2020.



Github Link : https://github.com/libinandrews/Final_Project/blob/Master/04_Eda_With_Data_Visualization.ipynb

All Launch Site Names

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

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

```
task_1 = '''
        SELECT DISTINCT LaunchSite
        FROM SpaceX
'''
create_pandas_df(task_1, database=conn)
```

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

Github Link : https://github.com/libinandrews/Final_Project/blob/Master/05_SQL_Notebook_for_PeerAssignment.ipynb

Launch Site Names Begin with 'KSC'

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

```
n [ ]:
task_2 = '''
        SELECT *
        FROM SpaceX
        WHERE LaunchSite LIKE 'CCA%'
        LIMIT 5
        '''
create_pandas_df(task_2, database=conn)
```

ut[ ]:

	date	time	boosterversion	launchsite	payload	payloadmasskg	orbit	customer	missionoutcome	landingoutcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	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	2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

We used the query above to display 5 records where launch sites begin with `CCA`

Total Payload Mass

- We calculated the total payload carried by boosters from NASA as 45596 using the query below

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

```
In [ ]:
task_3 = '''
        SELECT SUM(PayloadMassKG) AS Total_PayloadMass
        FROM SpaceX
        WHERE Customer LIKE 'NASA (CRS)'
        '''
create_pandas_df(task_3, database=conn)
```

Out[ ]:

total_payloadmass
0	45596

Average Payload Mass by F9 v1.1

We calculated the average payload carried by boosters using the query below

Task 4

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

```
task_4 = '''
        SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
        FROM SpaceX
        WHERE BoosterVersion = 'F9 v1.1'
        '''
create_pandas_df(task_4, database=conn)
```

	avg_payloadmass
0	2928.4

First Successful Ground Landing Date

- We observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015

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

Hint:Use min function

```
task_5 = '''
        SELECT MIN(Date) AS FirstSuccessfull_landing_date
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Success (ground pad)'
        '''
create_pandas_df(task_5, database=conn)
```

	firstsuccessfull_landing_date
0	2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- We used the **WHERE** clause to filter for boosters which have successfully landed on drone ship and applied the **AND** condition to determine successful landing with payload mass greater than 4000 but less than 6000

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

```
task_6 = '''
        SELECT BoosterVersion
        FROM SpaceX
        WHERE LandingOutcome = 'Success (drone ship)'
            AND PayloadMassKG > 4000
            AND PayloadMassKG < 6000
        '''
create_pandas_df(task_6, database=conn)
```

	boosterversion
0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

- We used wildcard like '%' to filter for WHERE MissionOutcome was a success or a failure.

List the total number of successful and failure mission outcomes

```
task_7a = '''
        SELECT COUNT(MissionOutcome) AS SuccessOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Success%'
        '''

task_7b = '''
        SELECT COUNT(MissionOutcome) AS FailureOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Failure%'
        '''
print('The total number of successful mission outcome is:')
display(create_pandas_df(task_7a, database=conn))
print()
print('The total number of failed mission outcome is:')
create_pandas_df(task_7b, database=conn)
```

```
The total number of successful mission outcome is:
```

	successoutcome
0	100

```
The total number of failed mission outcome is:
```

	failureoutcome
0	1

Boosters Carried Maximum Payload

- We determined the booster that have carried the maximum payload using a subquery in the WHERE clause and the MAX() function.

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

```
task_8 = '''
        SELECT BoosterVersion, PayloadMassKG
        FROM SpaceX
        WHERE PayloadMassKG = (
                                SELECT MAX(PayloadMassKG)
                                FROM SpaceX
                                )
        ORDER BY BoosterVersion
        '''
create_pandas_df(task_8, database=conn)
```

	boosterversion	payloadmasskg
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

2015 Launch Records

- We used a combinations of the WHERE clause, LIKE, AND, and BETWEEN conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

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

```
task_9 = '''
        SELECT BoosterVersion, LaunchSite, LandingOutcome
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Failure (drone ship)'
            AND Date BETWEEN '2015-01-01' AND '2015-12-31'
        '''
create_pandas_df(task_9, database=conn)
```

	boosterversion	launchsite	landingoutcome
0	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
1	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

Rank the count of landing outcomes (such as Failure (drone ship) or

```
task_10 = '''
        SELECT LandingOutcome, COUNT(LandingOutcome)
        FROM SpaceX
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
        GROUP BY LandingOutcome
        ORDER BY COUNT(LandingOutcome) DESC
        '''
create_pandas_df(task_10, database=conn)
```

	landingoutcome	count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1
7	Failure (parachute)	1

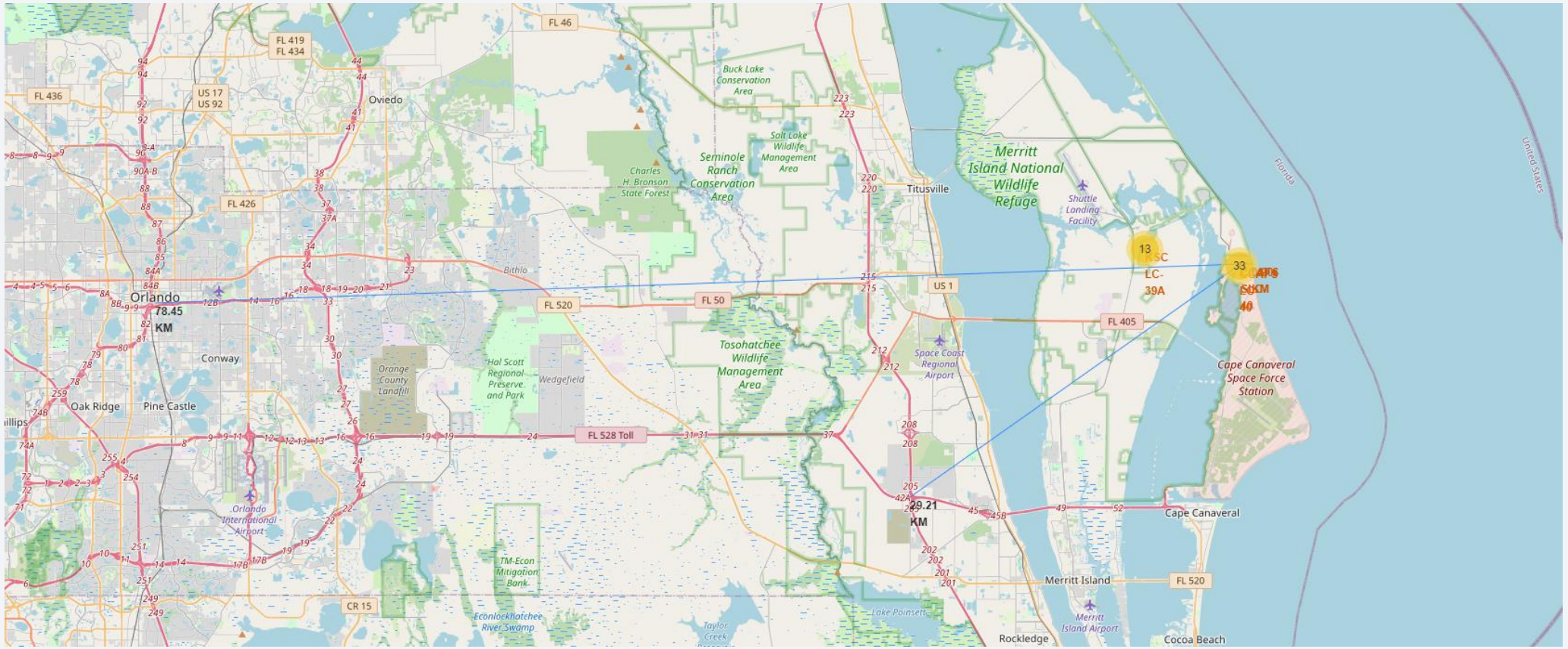
Section 3

Launch Sites Proximities Analysis

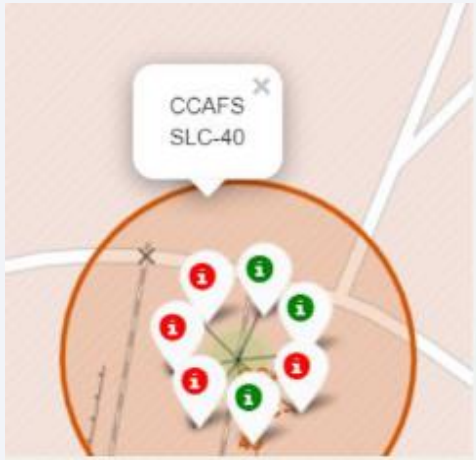
All launch sites global map markers

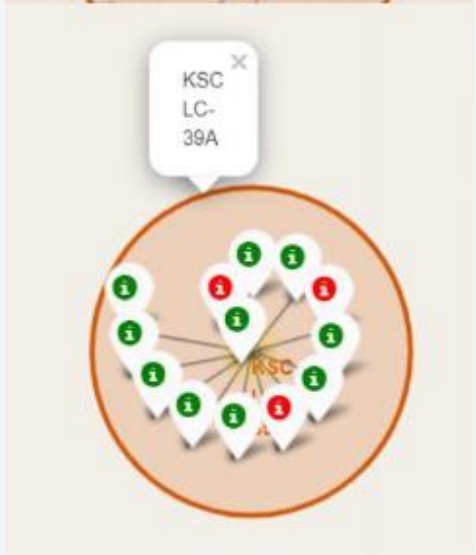
SpaceX launch sites are in US costal area

Distance to Florida City



Markers showing launch sites with color labels



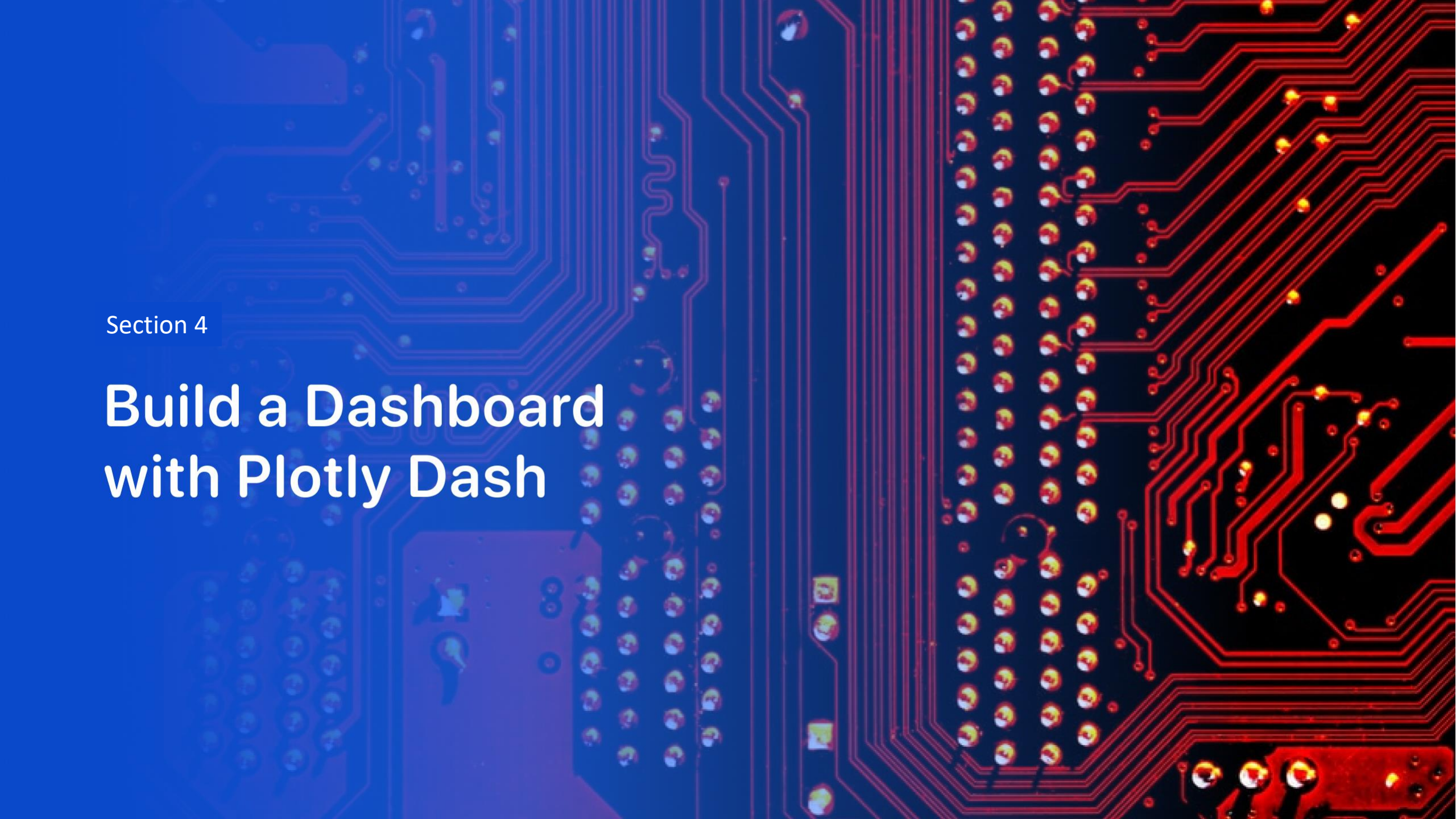


Florida Launch Sites

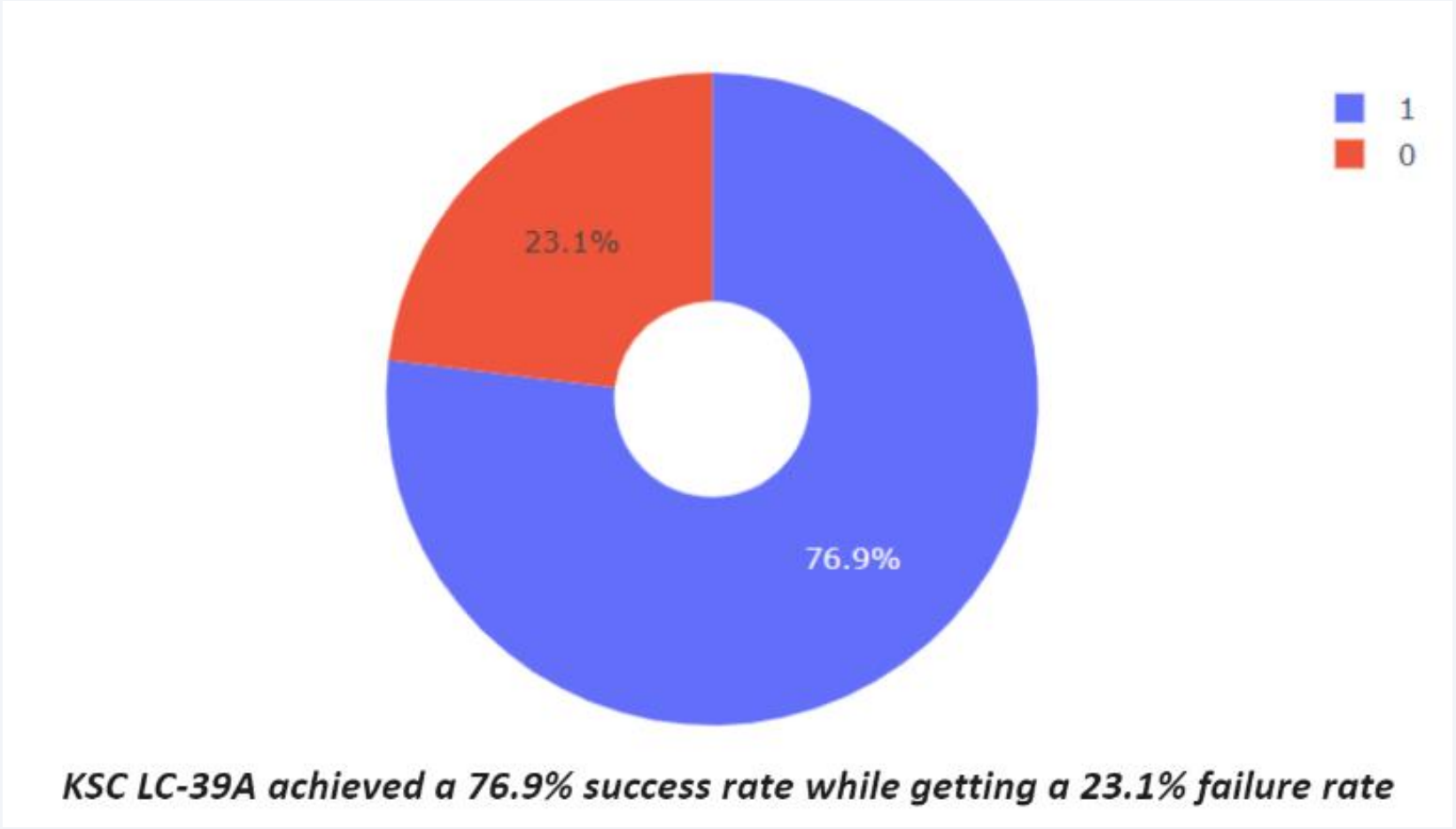
Green Marker shows successful Launches and Red Marker shows Failures

California Launch Site

37

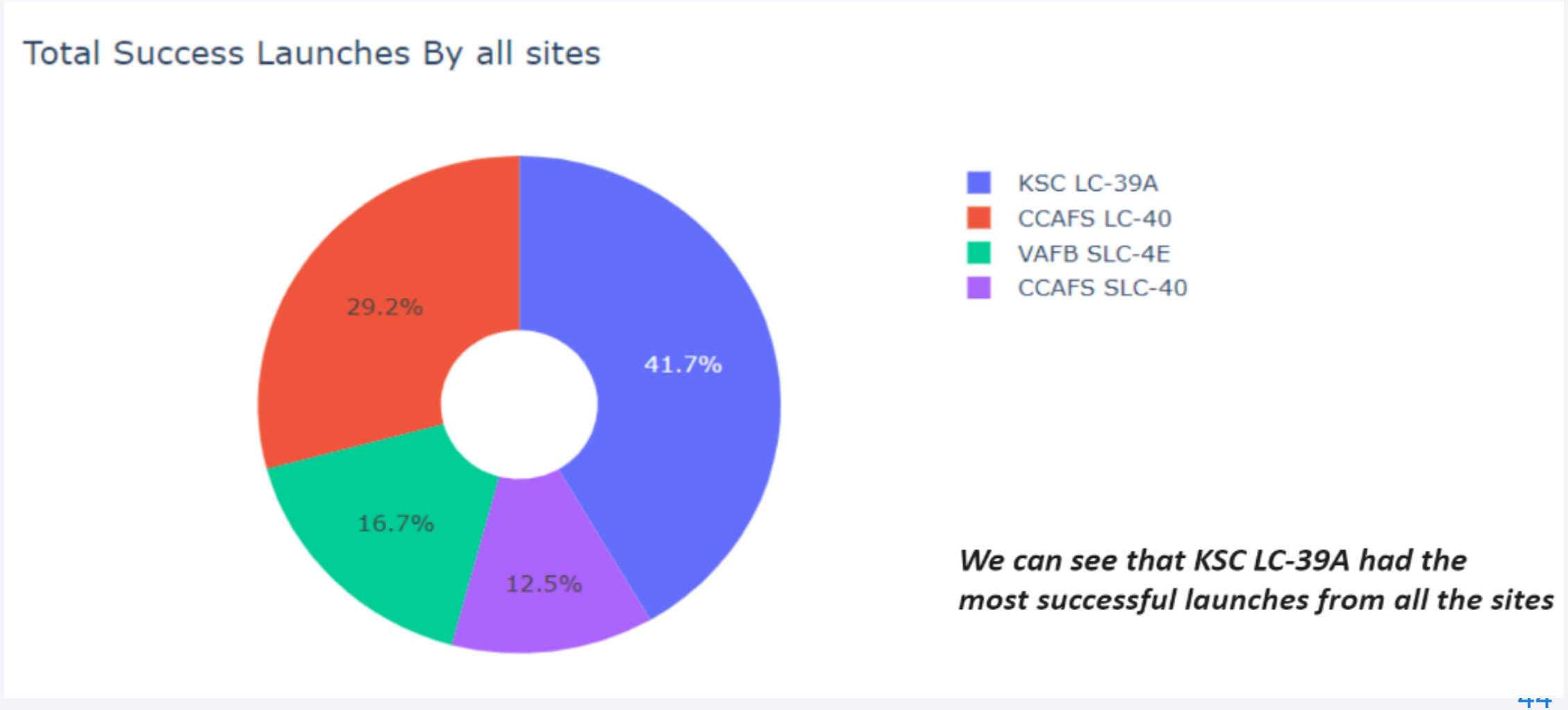

Section 4
Build a Dashboard with Plotly Dash

Pie chart showing the Launch site with the highest launch success ratio



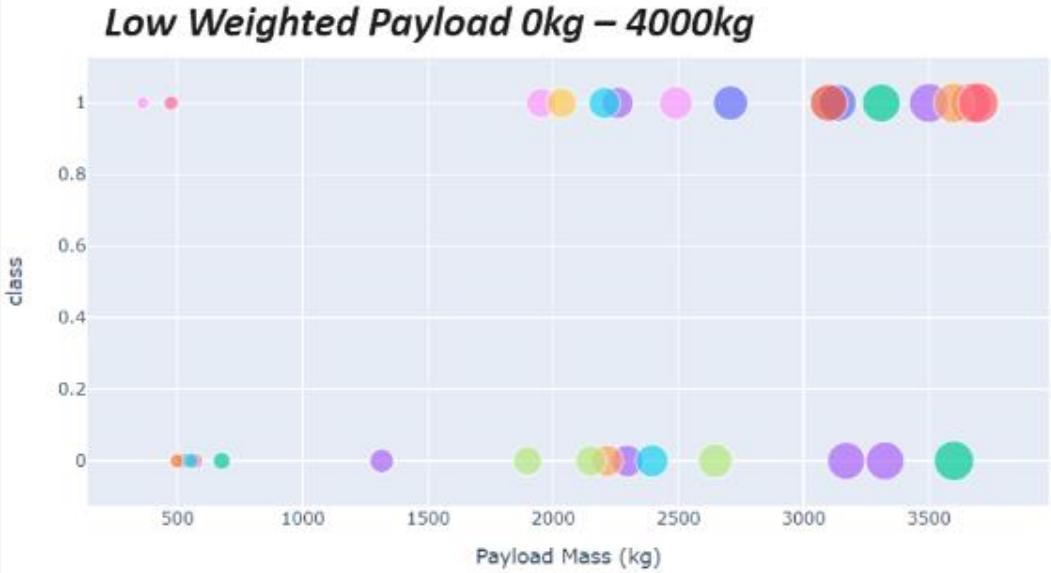
KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

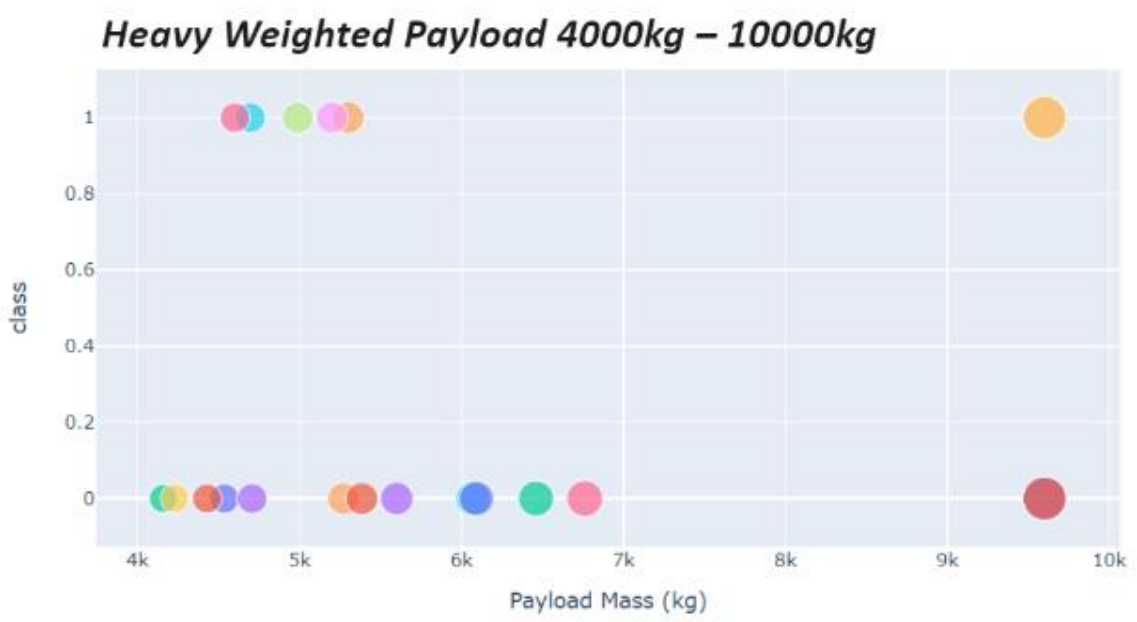
Pie chart showing the success percentage achieved by each launch site



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

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



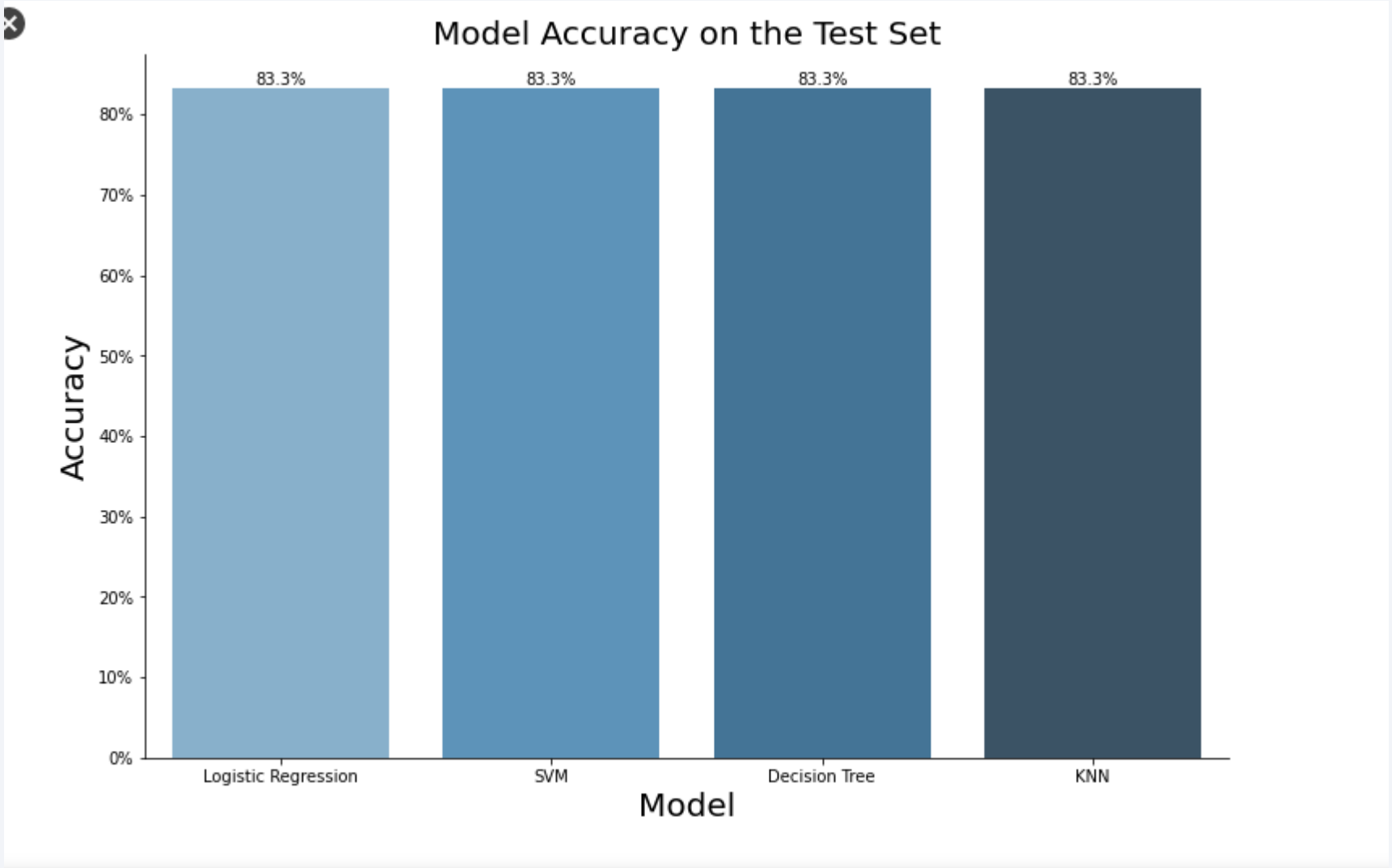


We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 5

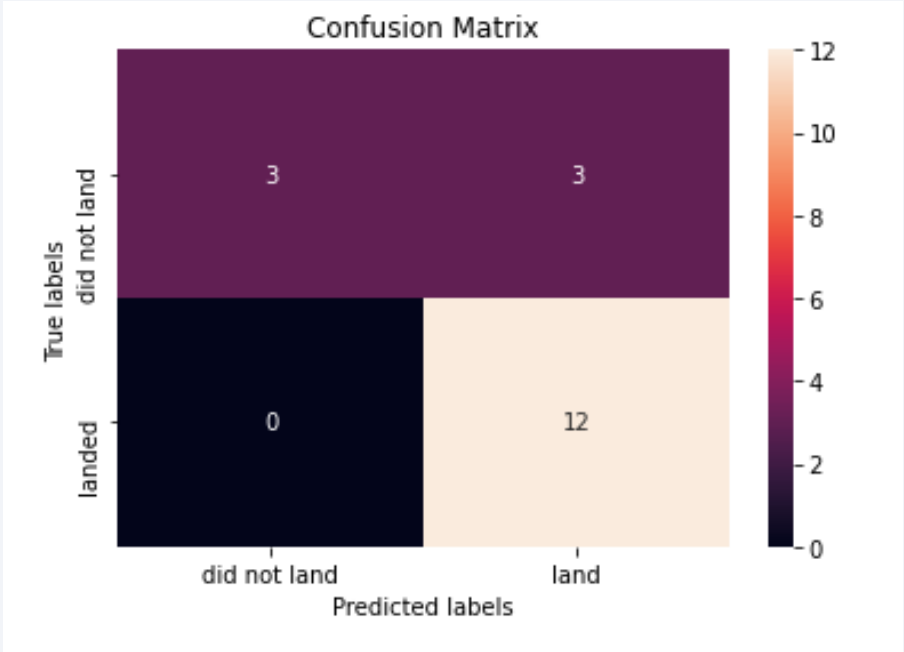
Predictive Analysis (Classification)

Classification Accuracy



Confusion Matrix

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



Conclusions

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

Thank you!