IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Afnan A. Yaqub
September 3, 2022

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection using SpaceX REST API
  - Data Collection by Web Scraping (HTML)
  - Data Wrangling (Occurrences vs Outcomes, Feature Engineering)
  - Exploratory Data Analysis (EDA) with SQL
  - EDA with Visualizations
  - Interactive Visual Analytics with Folium Maps
  - Interactive Dashboard with Plotly Dash
  - Machine Learning Prediction (Logistic Regression, SVM, Decision Tree and KNN classifiers)
- Summary of all results
  - EDA Results
  - Interactive maps and dashboards
  - Prediction Results

Introduction

- Project background and context
  - SpaceX is ahead of the game with $62M cost per launch of its Falcon 9 rocket compared to $165M by its competitors.
  - Primary reason for SpaceX low launch cost is recovery of the rocket's first stage.
  - Successful bidding as a competitor requires predicting the likelihood of first stage landing based on understanding of conditions that contribute to this outcome.
- Problems you want to find answers
  - Identify successful / failed landing characteristics
  - Identify relationship between mission features and first stage landing outcomes
  - Identify conditions that contribute to a successful launch result

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
    - SpaceX REST API
    - Web scraping from Wikipedia
- Perform data wrangling
    - Sorting and filtering the necessary features
    - Feature engineering using one-hot encoding for categorical variables
- 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 Logistic Regression, SVM, Decision Tree and K-Nearest Neighbors Classifiers
    - Tune classifiers using GridSearchCV and train on the training data
    - Evaluate models based on training and testing scores

Data Collection

- Datasets are collected by
  - Using SpaceX REST API from following endpoints
    - URL endpoint: https://api.spacexdata.com/v4
    - Booster Version: https://api.spacexdata.com/v4/rockets/
    - Launch Pads: https://api.spacexdata.com/v4/launchpads/
    - Launch Pads: https://api.spacexdata.com/v4/payloads/
  - Using Web scraping from following Wikipedia page
    - URL: https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922

Data Collection – SpaceX API

Github Notebook

1. Call SpaceX REST API and receive response

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

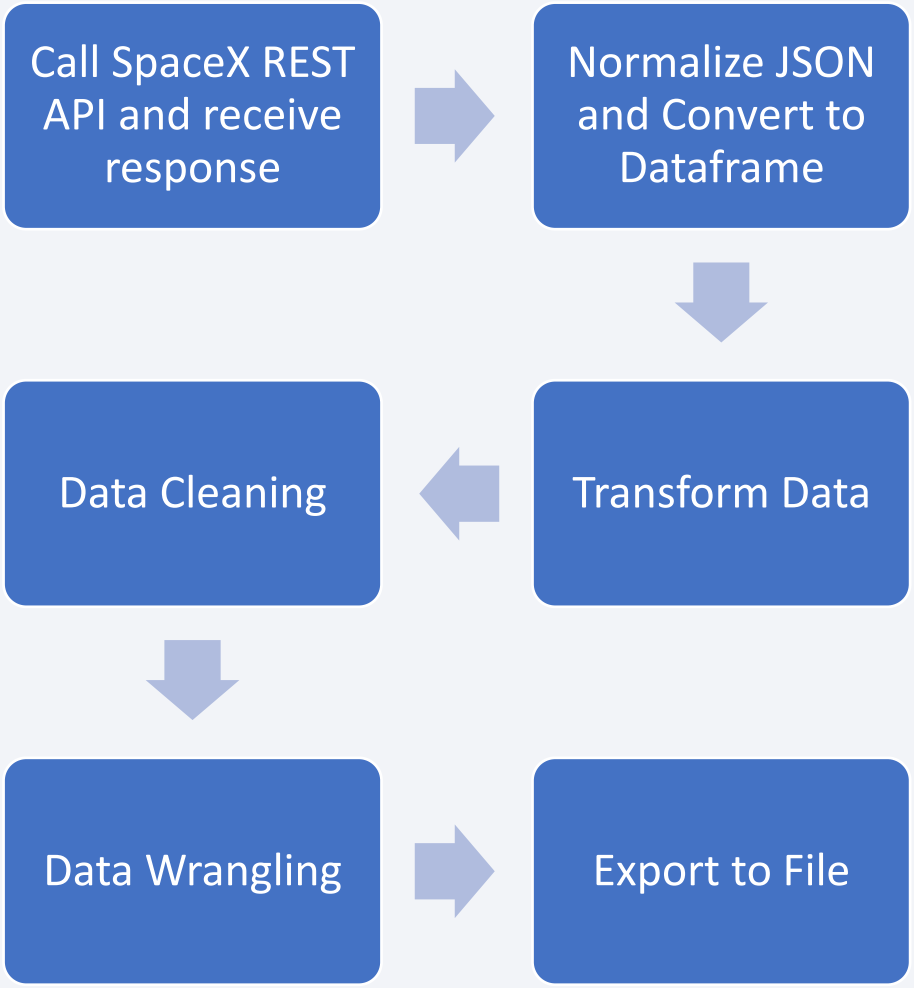
2. Normalize as JSON and convert to dataframe

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

3. Subset dataframe to keep only necessary columns

```
# Lets take a subset of our dataframe keeping only the features we want
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# remove rows with multiple cores
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# extract the single value in the list and replace the feature
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# convert the date_utc to a datetime datatype
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

Continued



Data Collection – SpaceX API

4. Extract and transform necessary features

```
# Call getBoosterVersion
getBoosterVersion(data)

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

5. Create dictionary for 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}
```

6. Filter data for Falcon 9 launches

```
# Hint data['BoosterVersion']!='Falcon 1'
filter = df_launch['BoosterVersion']!='Falcon 1'
data_falcon9 = df_launch.where(filter)
```

7. Data wrangling: delete or replace missing values

```
# drop rows that have all NaN values
data_falcon9=data_falcon9.dropna(how='all')

# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].fillna(value=data_falcon9['PayloadMass'].mean(), inplace=True)
```

8. Save data to file

```
data_falcon9.to_csv('dataset_part\_1.csv', index=False)
```


Github Notebook

Data Collection – Scraping

[Github Notebook]

1. Get HTML from Wikipedia to HTML Object

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launc
# use requests.get() method with the provided static_url
# assign the response to a object
html_data = requests.get(static_url).text
```

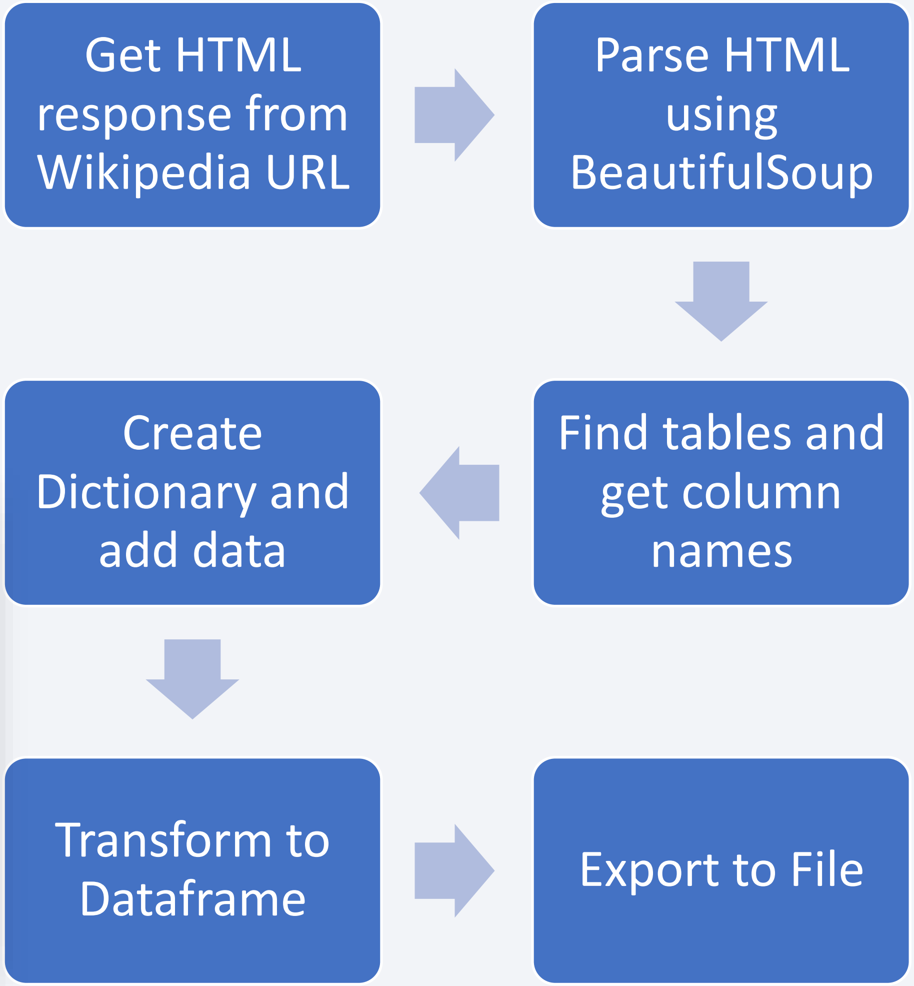
2. Parse HTML using BeautifulSoup

```
# Use BeautifulSoup() to create a BeautifulSoup object
soup = BeautifulSoup(html_data, "html.parser")
```

3. Find tables and get column names

```
# 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')

# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)

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
# Apply find_all() function with `th` element on first_launch_table
table_head = first_launch_table.find_all('th')
table_head[1]
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
for tbh in table_head:
    name = extract_column_from_header(tbh)
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
    if name is not None and len(name) > 0:
        column_names.append(name)
```

Continued



Data Collection – Scraping

Github Notebook

4. Create Dictionary

```
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']=[]
```

5. Add Data to Dictionary

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            launch_dict['Flight No.'].append(flight_number)
            #print(flight_number)
```

6. Transform to dataframe

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

7. Export to file

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

Data Wrangling

[Github Notebook]

- The aim of data wrangling activity is to perform exploratory data analysis and convert landing outcomes into labels for success and failure class for predictive algorithm
- First we calculate the number of missing values
- Then we move on to EDA for following measures
  - Number of launches for each site
  - Number and occurrence of each orbit
  - Number and occurrence of mission outcome per orbit type
    - True ASDS, True RTLS and True Ocean means the mission has been successful
    - False ASDS, None ASDS, False RTLS, False Ocean, None None all represent failures
- Create landing outcome labels and evaluate success rates
  - 1 means success
  - 0 means failure

Data Wrangling

[Github Notebook]

1. Missing value percentage

```
df.isnull().sum()/df.count()*100

FlightNumber       0.000
Date               0.000
BoosterVersion     0.000
PayloadMass        0.000
Orbit              0.000
LaunchSite         0.000
Outcome            0.000
Flights            0.000
GridFins           0.000
Reused             0.000
Legs               0.000
LandingPad        40.625
Block              0.000
ReusedCount        0.000
Serial             0.000
Longitude          0.000
Latitude           0.000
dtype: float64
```

2. Number of launches for each site

```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()

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

3. Number and occurrence of each orbit type

```
# Apply value_counts on Orbit column
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
```

4. Number and occurrence of each landing outcome

```
# landing_outcomes = values on Outcome column
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
```

5. Create outcome labels as class

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for oc in df['Outcome']:
    # print(oc)
    if oc in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(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

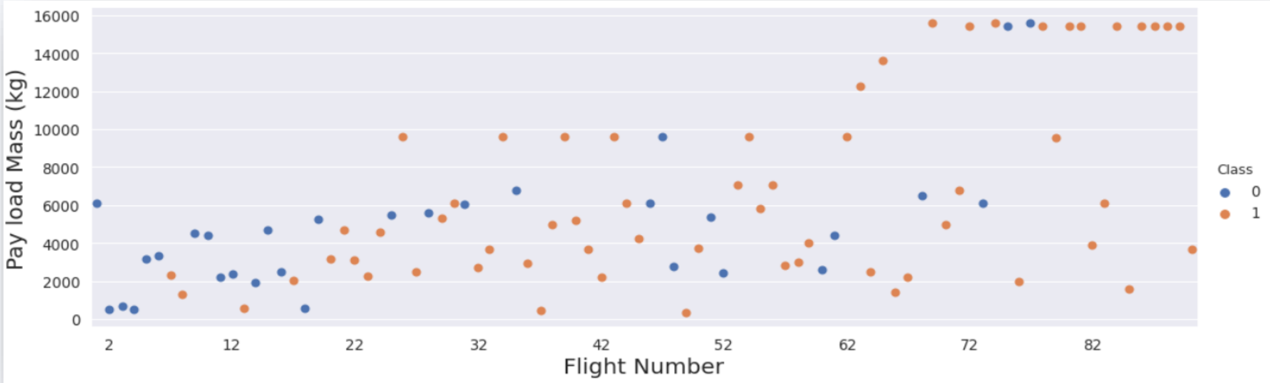
6. Export the results to file.

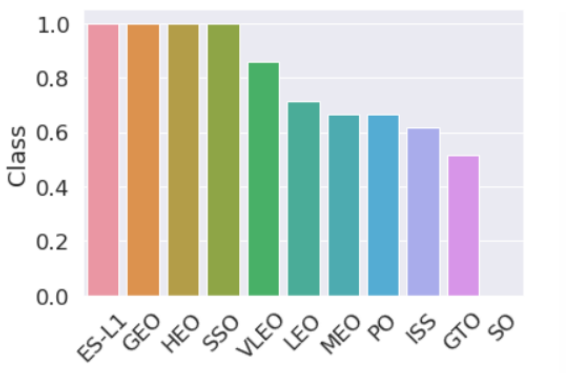
```
df.to_csv("dataset_part_2.csv", index=False)
```

EDA with Data Visualization

[Github Notebook](#)

- After data wrangling, EDA with visualization was performed
  - Scatter Charts were created for Success Analysis against following correlations
    - Flight Number vs Payload Mass
    - Flight Number vs Launch Site
    - Payload Mass vs Launch Site
    - Flight Number vs Orbit
    - Payload Mass vs Orbit
  - Bar Chart was created for
    - Success rate vs Orbit type
  - Line Chart was created for
    - Success rate over years
  - Feature Engineering was done using one-hot-encoding





EDA with SQL

Github Notebook

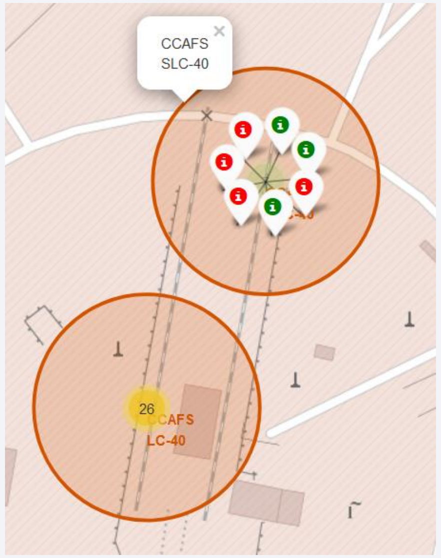
- EDA with SQL was performed inline using SQLite. Following queries were executed
  - The unique site names for missions
  - 5 records for sites with names containing ‘CCA’
  - Total payload mass for booster launched for NASA (CRS) missions
  - Average payload mass carried by F9 v1.1 booster
  - The date for first successful ground pad landing
  - Missions with payload mass between 4000 and 6000 having successful landing outcomes for drone ship
  - Total number of successful and failed missions
  - Names and versions of boosters that carried maximum payload
  - Launch sites, booster versions and month names for year 2015 missions that had failures for drone ship landing
  - Rank the count of successful landing outcomes between the date 04-06-2010 and 20-03-2017 in descending order

Build an Interactive Map with Folium

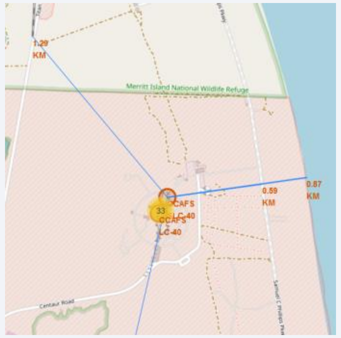
[Github Notebook]

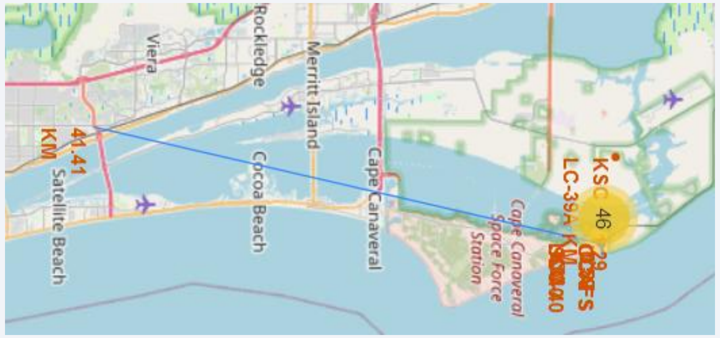
- Interactive maps were created for SpaceX data to mark all launch sites location using Folium. Following markers were created,

  - Circle with name popup for indicating NASA Johnson Space Center location
  - Circle for location of each launch site coordinates along with its name
  - Marker clusters based on mission outcome for each launch site using outcome class,
    - Success:1, indicated by green marker
    - Failure: 0, indicated by red marker



  - Markers indicating distance of launch sites from following landmarks
    - Coastline
    - Highways
    - Railways
    - Cities

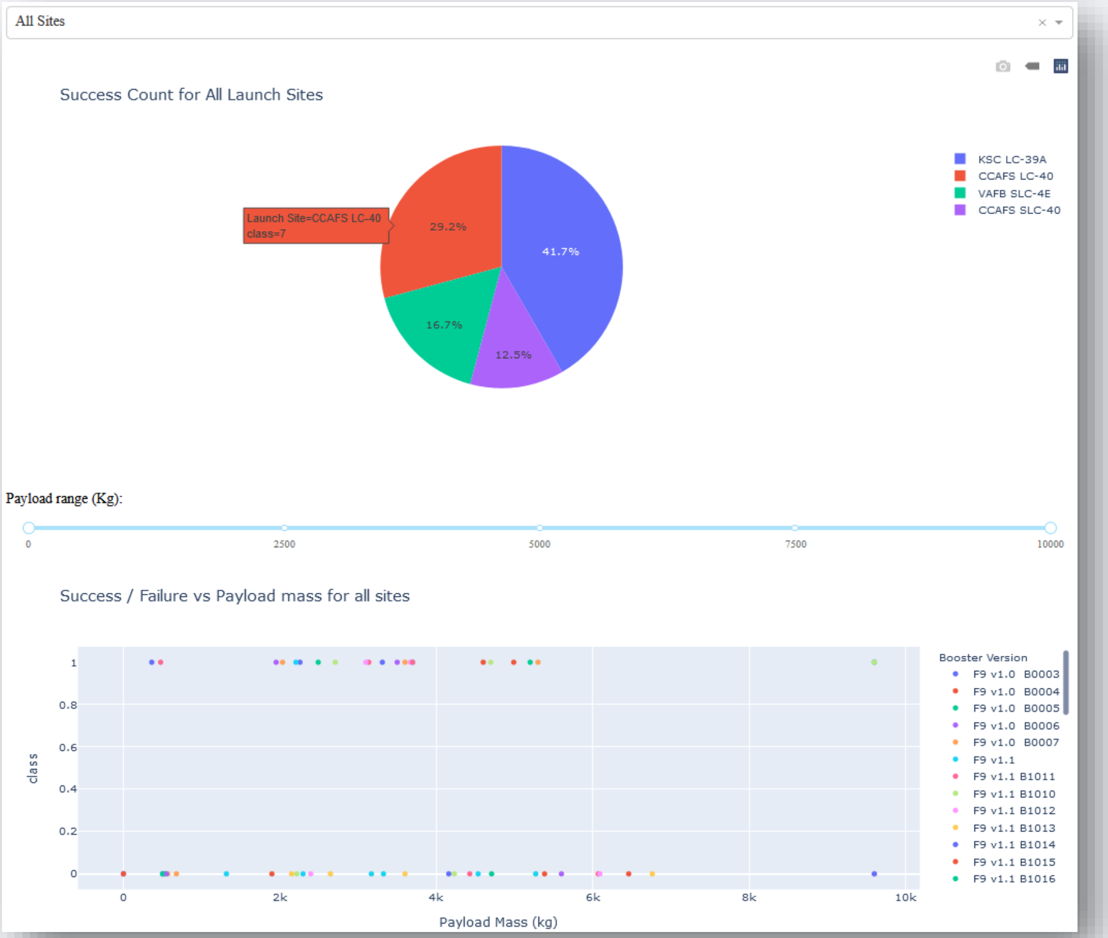




Build a Dashboard with Plotly Dash

Github Notebook

- An HTML server interactive dashboard application with following features was created using Plotly Dash
  - Drop down menu for user to select all sites or one
  - Pie chart for showing the success count based on success outcome for each site or all as selected by the user
  - A payload mass slider for selection of payload range
  - Scatter chart of Payload Mass vs Flight Number highlighted with color based on mission outcome



Predictive Analysis (Classification)

Github Notebook

- Data Preparation
  - Loading Data
  - Standardize Data (Normalize)
  - Train-Test Split
- Model Preparation
  - Select Classifier/Predictor
    - Logistic Regression, Support Vector Machine , Decision Tree, K-Nearest Neighbours)
  - Tune to select best parameters using GridSearchCV
  - Train model using training set

Predictive Analysis (Classification)

Github Notebook

- Model Evaluation
  - Training score
  - Test accuracy
  - Confusion Matrix
- Model Comparison
  - Test Scores

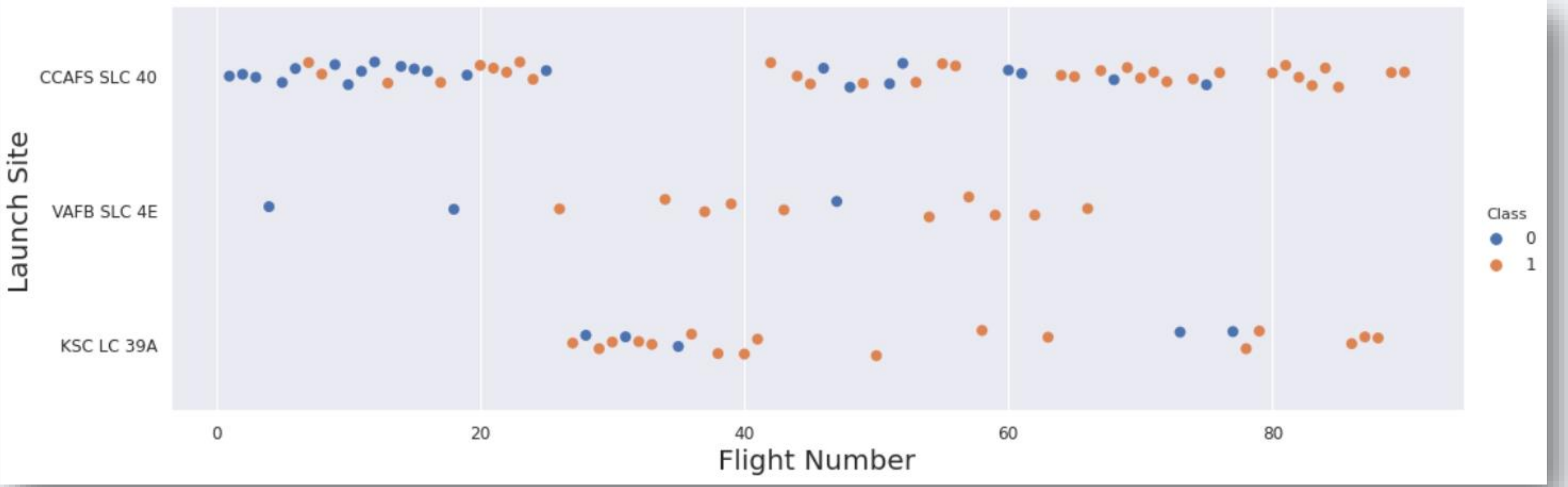
Results

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

Section 2

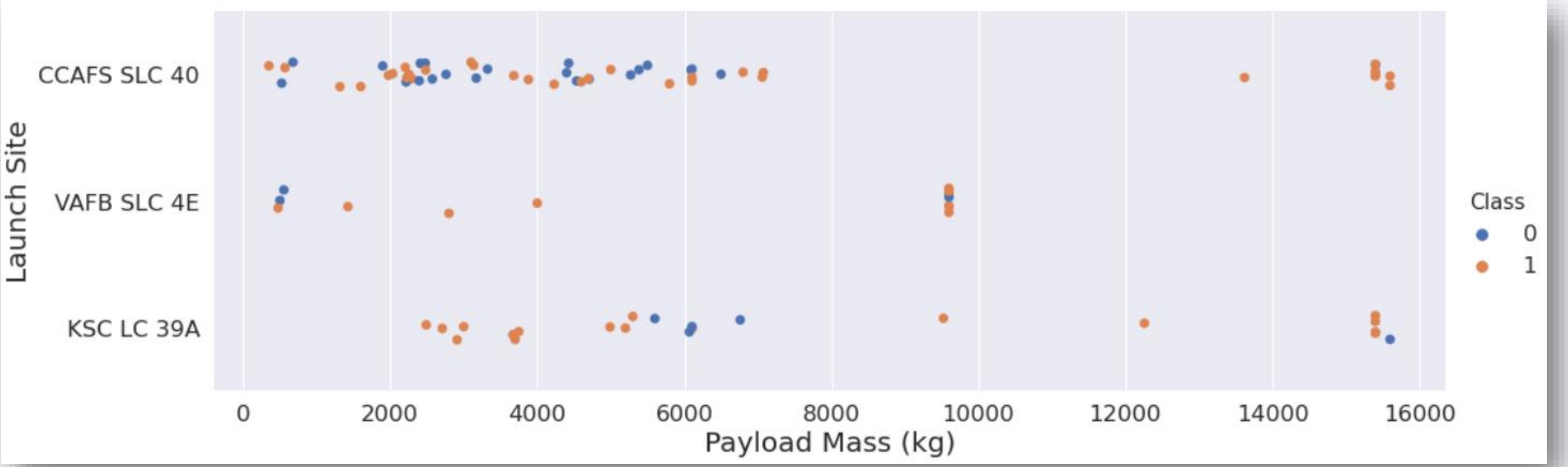
Insights drawn from EDA

Flight Number vs. Launch Site



- We see as more and more flights were carried out the landing success rate for each site has increased due to experience and learning from failures. CCAFS SLC 40 has highest number of launches.

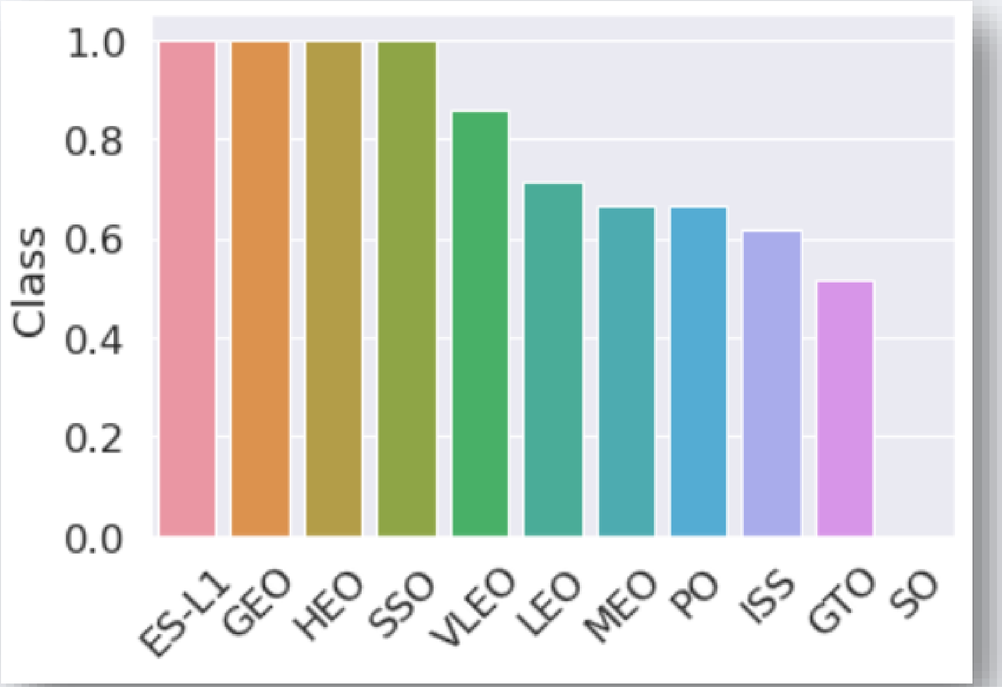
Payload vs. Launch Site



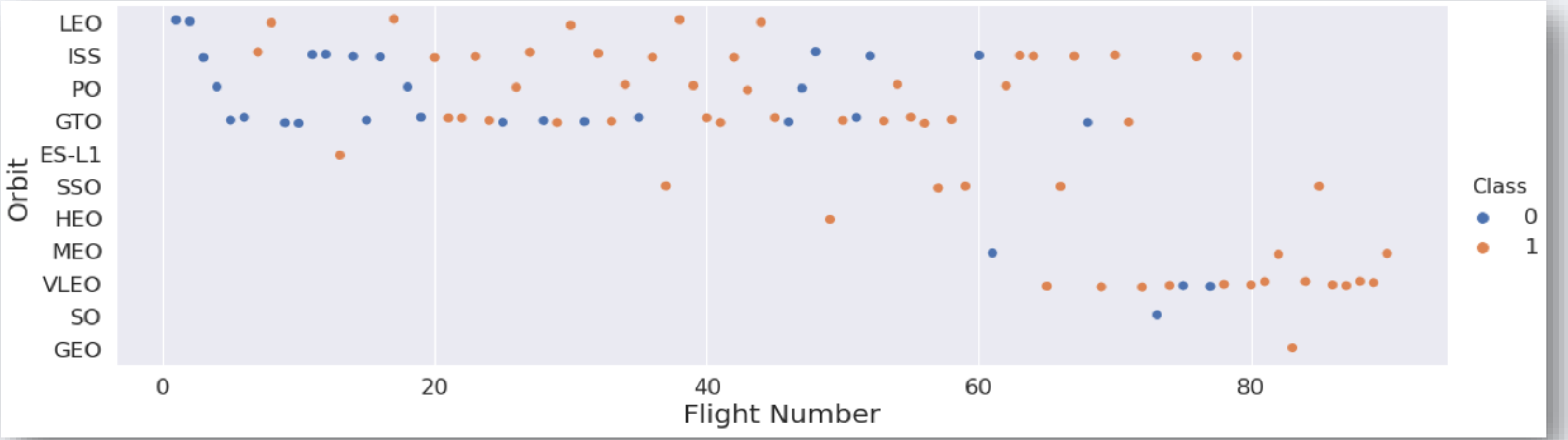
- Boosters with heavier payload have a better shot at landing success. KSC LC 39A is very successful for payload range 2000-5000

Success Rate vs. Orbit Type

- ES-L1, GEO, HEO, SSO have 100% success rate.
- VLEO has above 80% rate.

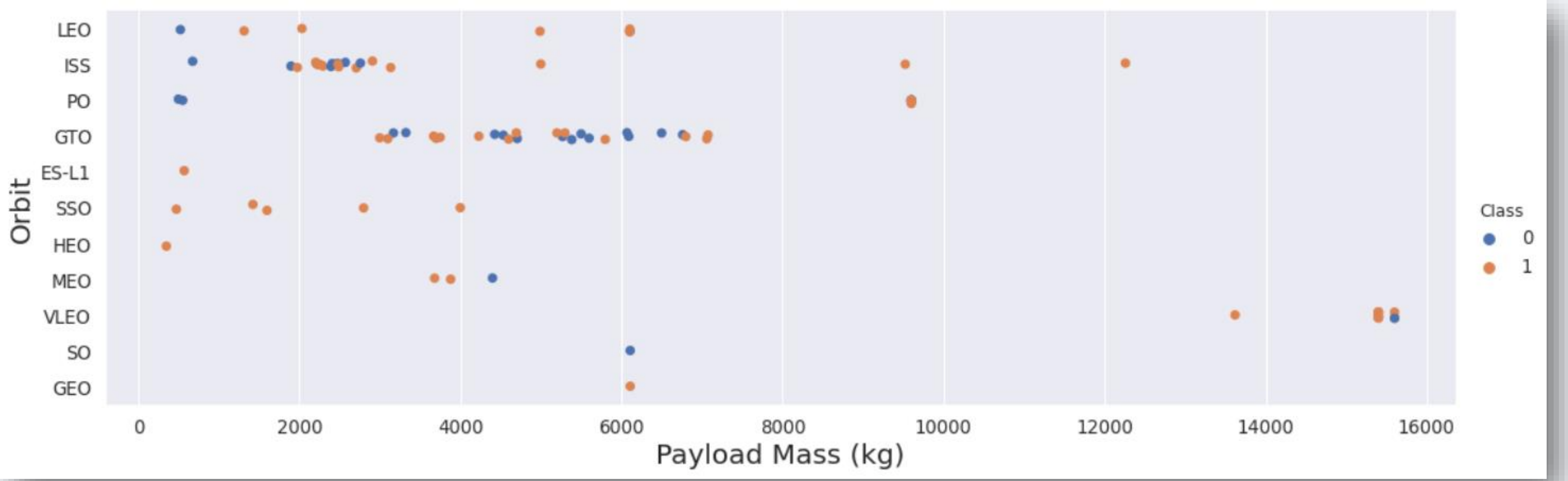


Flight Number vs. Orbit Type



- The scatter chart, however, shows that there has been only one flight for ES-L1.
- Most of the flights have been targeted for GTO and ISS. Who have a mixed rate of success rate of success for flight number.
- VLEO has only been a region of interest for recent launches.

Payload vs. Orbit Type



- Booster landing rate for heavier payloads has significantly high success rate.
- SSO has been used for lighter payload and has not experienced a failure.
- ISS and GTO have mixed rate of success against payload distribution.

Launch Success Yearly Trend

- There has been mostly a consistent consistent rise in average success rate from 2013 to 2017 and 2018
- 2017 and 2019 have seen relative rates of failure

All Launch Site Names

- Distinct attribute was used in SQL for getting unique site names from database table.
- SpaceX has used 4 launch sites for its missions.

```
%sql SELECT DISTINCT launch_site FROM SPACEXTBL
```

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

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

Launch Site Names Begin with 'CCA'

- 5 records where launch sites begin with `CCA` were queried using LIKE and LIMIT attributes
- LIKE is used to query a text containing a given subtext. LIMIT restricts the number of records in the result.

```
%sql SELECT * FROM SPACEXTBL where launch_site LIKE "%CCA%" LIMIT 5
```

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

Date	Time	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS_KG	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- Total payload mass carried by Falcon 9 for NASA CRS missions has been queried using aggregate function SUM() and attribute LIKE.
- Above 48000 kgs have been transported to orbit for NASA CRS missions.

```
%sql SELECT sum(PAYLOAD_MASS_KG) AS 'Total Payload  Mass' FROM SPACEXTBL where Customer LIKE "%NASA (CRS)%"
```

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

Total Payload Mass
48213

Average Payload Mass by F9 v1.1

- Average payload mass carried by F9 v1.1 has been queried using AVG() function and LIKE attribute.
- Above 2500 kgs of average payload mass has been placed into orbit by F9 v1.1.

```
%sql SELECT AVG(PAYLOAD_MASS_KG) AS 'Average Payload Mass'\
FROM SPACEXTBL\
WHERE Booster_Version LIKE "%F9 v1.1%"
```

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

Average Payload Mass
2534.6666666666665

First Successful Ground Landing Date

- 2015 was the year of first successful ground pad landing by 1 stage.
- Query was made using MIN() function on the Landing_Outcome column converted to datetime type using STRFTIME()
- LIKE attribute for substring "Success (ground pad)" was used on the Landing_Outcome column

```
%sql SELECT STRFTIME(Date) AS Date, \
min(Landing_Outcome) AS 'Landing Outcome' \
FROM SPACEXTBL \
WHERE Landing_Outcome LIKE "%Success (ground pad)%"
```

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

Date	Landing Outcome
2015-12-22	Success (ground pad)

Successful Drone Ship Landing with Payload between 4000 and 6000

- Boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000 were queried.
- The query uses WHERE and LIKE attributes for conditional output.

```
%sql SELECT Booster_Version AS 'Booster Version', \
Landing_Outcome AS 'Landing Outcome' \
FROM SPACEXTBL \
WHERE Landing_Outcome like "%success (drone ship)%"
```

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

Booster Version	Landing Outcome
F9 FT B1021.1	Success (drone ship)
F9 FT B1022	Success (drone ship)
F9 FT B1023.1	Success (drone ship)
F9 FT B1026	Success (drone ship)
F9 FT B1029.1	Success (drone ship)
F9 FT B1021.2	Success (drone ship)
F9 FT B1029.2	Success (drone ship)
F9 FT B1036.1	Success (drone ship)
F9 FT B1038.1	Success (drone ship)
F9 B4 B1041.1	Success (drone ship)
F9 FT B1031.2	Success (drone ship)
F9 B4 B1042.1	Success (drone ship)
F9 B4 B1045.1	Success (drone ship)
F9 B5 B1046.1	Success (drone ship)

Total Number of Successful and Failure Mission Outcomes

- WHERE and LIKE attributes are used in query for total number of successful and failed mission outcomes.

```
%sql SELECT (SELECT COUNT(Mission_Outcome) FROM SPACEXTBL \
             WHERE Mission_Outcome like "%success%") AS 'Successes', \
            (SELECT COUNT(Mission_Outcome) FROM SPACEXTBL \
            WHERE Mission_Outcome like"%fail%") AS 'Failures'
```

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

Successes	Failures
100	1

Boosters Carried Maximum Payload

- Nested queries used along with WHERE and IN to get booster versions carrying maximum payloads
- MAX() function was used on Payload_Mass_KG column

```
%sql SELECT   Booster_Version AS 'Booster Version', \
              Payload_Mass_KG AS 'Payload Mass (kg)' \
              FROM SPACEXTBL \
              WHERE Payload_Mass_KG \
              IN (SELECT MAX(Payload_Mass_KG) FROM SPACEXTBL)
```

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

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

- List for failed Landing_Outcomes in drone ship, their booster versions, and launch site names for in year 2015 was queried as shown.
- SUBSTR() function was used to extract year and month from the string type entries in date column and WHERE was used to get results for year 2015 and LIKE for the Landing_Outcome.

```
%sql SELECT substr(Date,1,4) AS Year, \
            substr(Date, 6, 2) AS Month, \
            Booster_Version AS 'Booster Version', \
            Launch_Site AS 'Launch Site',\
            Landing_Outcome AS 'Landing Outcome' \
            FROM SPACEXTBL \
            WHERE Year = "2015" \
            AND Landing_Outcome like "%failure (drone ship)%"
```

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

Year	Month	Booster Version	Launch Site	Landing Outcome
2015	01	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
2015	04	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

- Count of successful landing outcomes between the date 2010-06-04 and 2017-03-20, was ranked in descending order
- The query uses
  - WHERE, BETWEEN, AND for dates
  - LIKE for success substring in outcomes
  - GROUP BY for grouping outcomes
  - ORDER BY, DESC for ranking

```
%sql SELECT Landing_Outcome AS 'Landing Outcomes', \
COUNT(Landing_Outcome) AS 'Outcome Count' FROM SPACEXTBL \
WHERE Date between '2010-06-04' AND '2017-03-20' \
AND Landing_Outcome LIKE "%success%" \
GROUP BY Landing_Outcome \
ORDER BY COUNT(Landing_Outcome) DESC
```

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

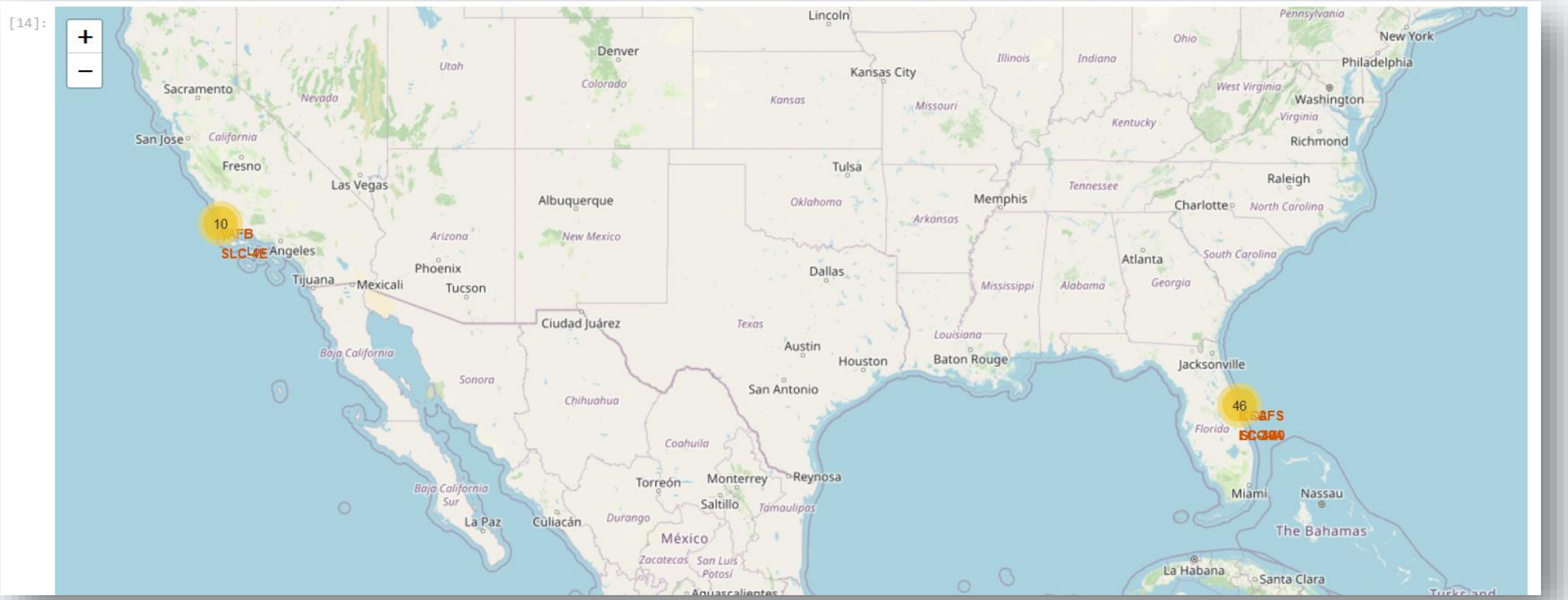
Landing Outcomes	Outcome Count
Success (drone ship)	5
Success (ground pad)	3

Section 3

Launch Sites Proximities Analysis

Folium Map – Launch Site Locations

- All site locations have been marked on the interactive map.
- Sites are in close proximity to US coastline.

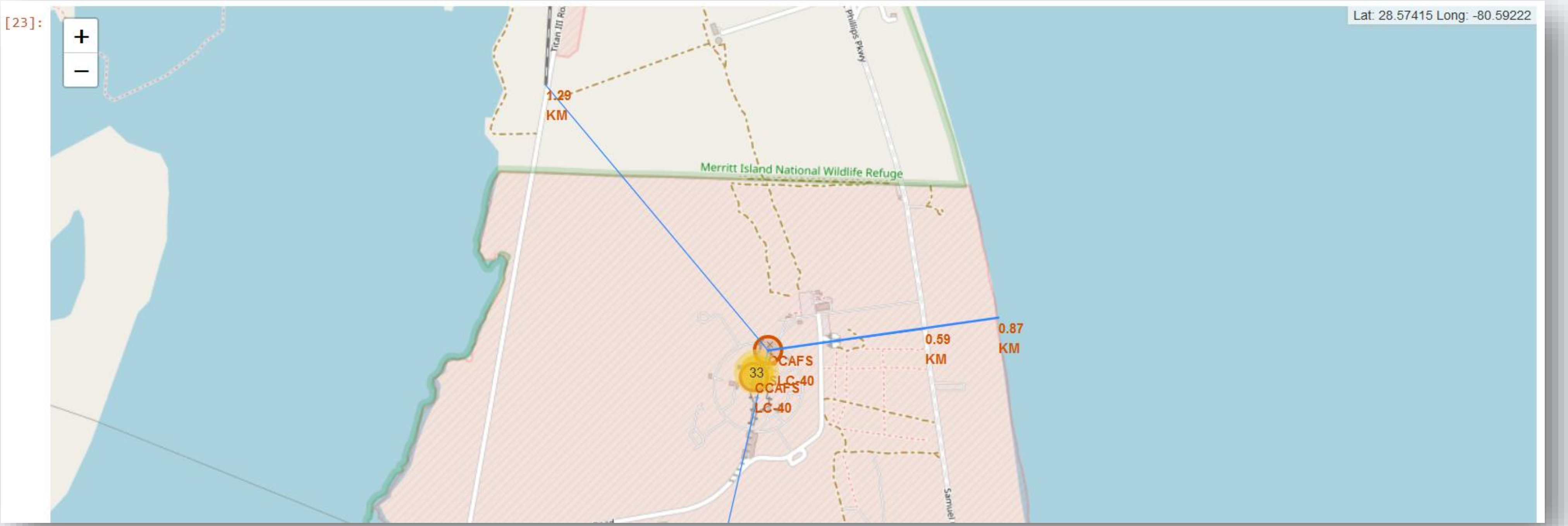


Folium Map – Mission Outcome Labels

- Marker clusters have been used to color label the launch sites based on launch outcomes

Folium Map – Proximity Indicators

- The generated folium map has been explored and the screenshot of a selected launch site to its proximities such as railway, highway, coastline, with distances calculated and displayed.
- Circles indicate sites, lines are distance markers

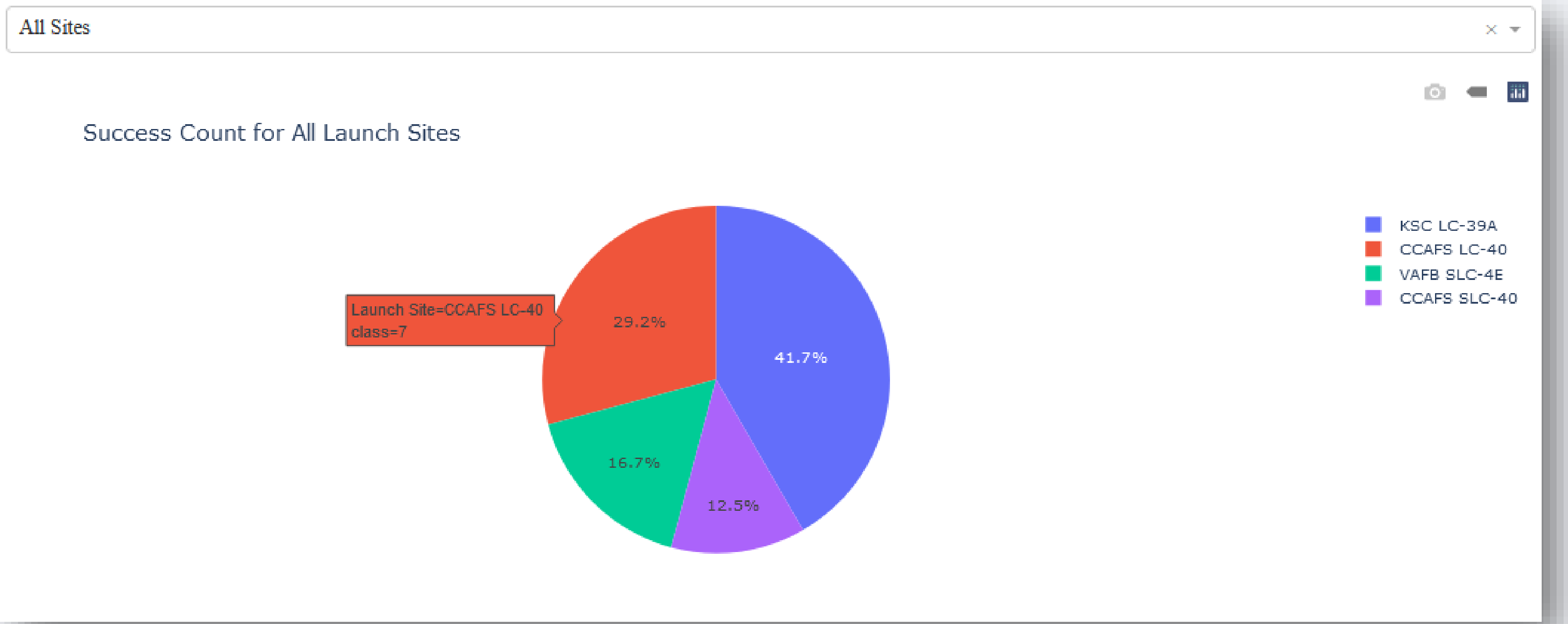


Section 4

Build a Dashboard with Plotly Dash

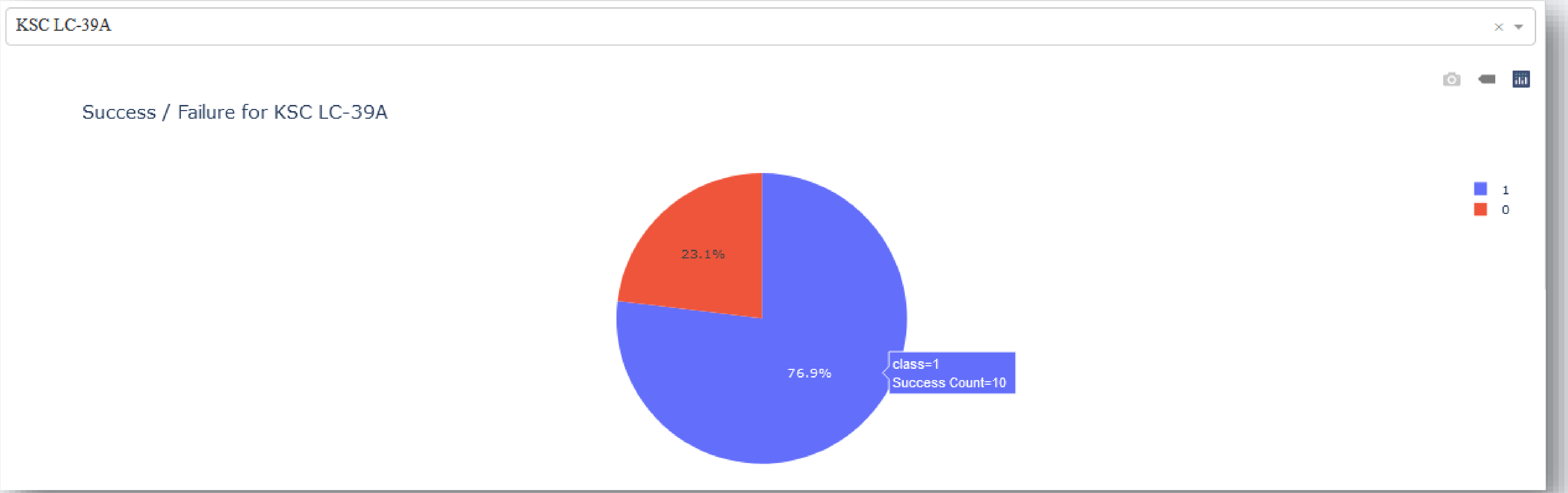
Dashboard – Success Count: All Sites

- A drop down menu is available for site selection.
- Success count for all or single site is visible as a pie chart.
- KSC LC-39A has the highest success rate of 41.7%.



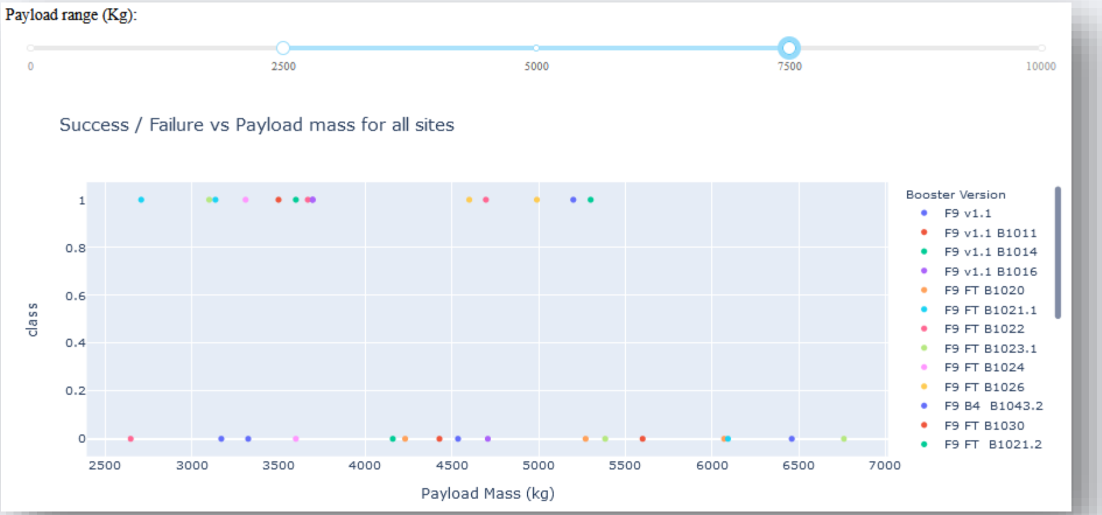
Dashboard – Most Successful Launch Site

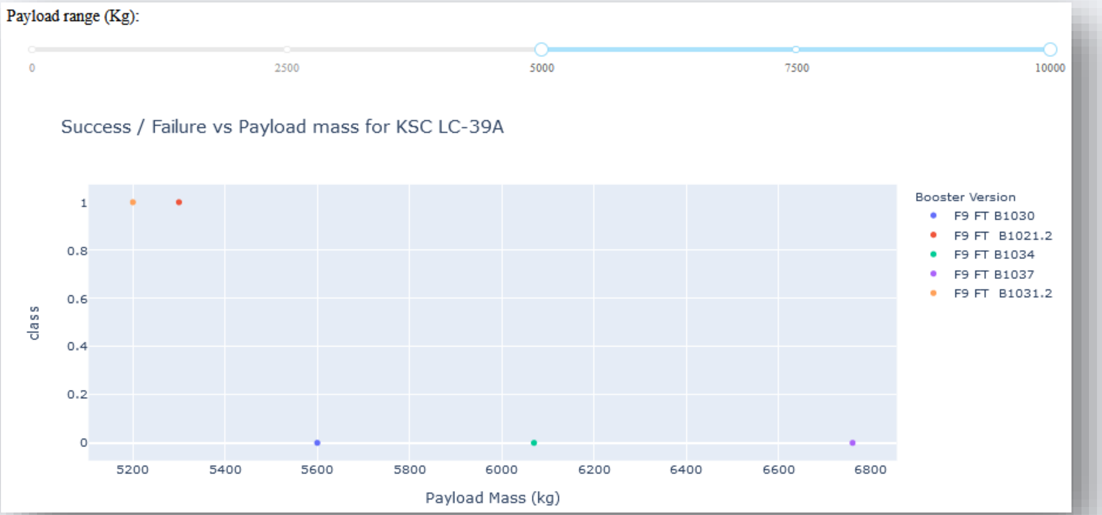
- KSC LC-39A is the Launch site with highest success count.
- Pie chart shows the success to failure count ratio for KSC LC-39A with 76.9% success.



Dashboard – Payload vs Launch Outcome

- A range slider is available for payload mass selection by user.
- Launch outcome for payload range 2500-7500 Kg for all sites is shown on left.
- Launch outcome for payload range 5000 to 10000 Kg for KSC LC-39A is on right.





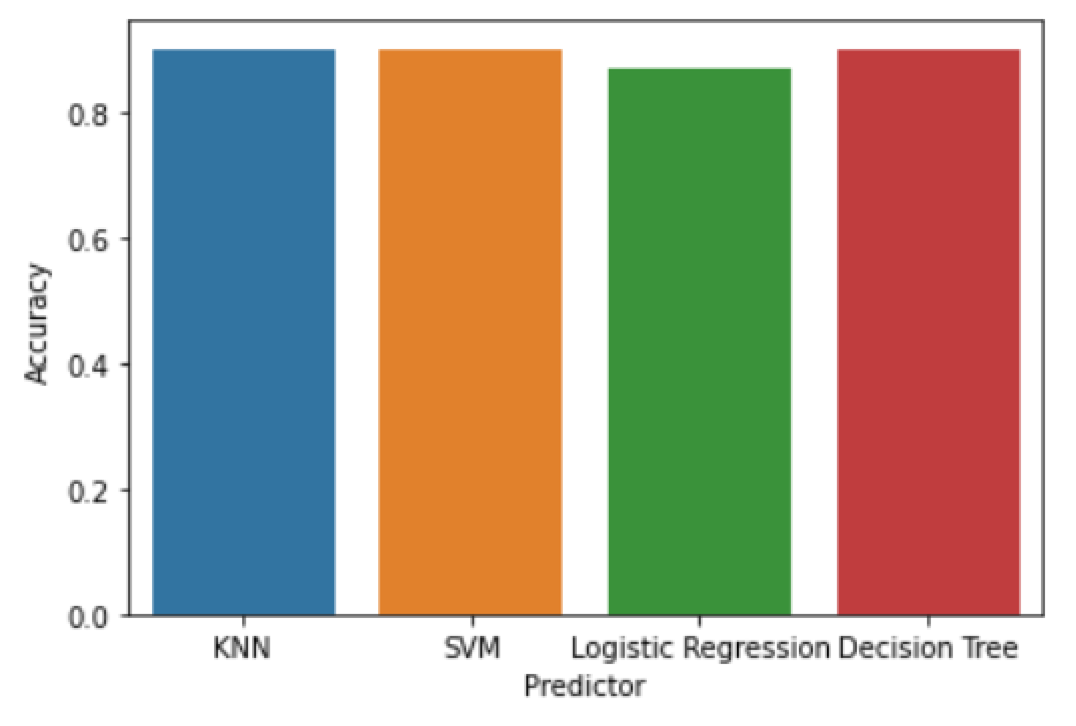
- Lower payload range has high success rate

Section 5

Predictive Analysis (Classification)

Classification Accuracy

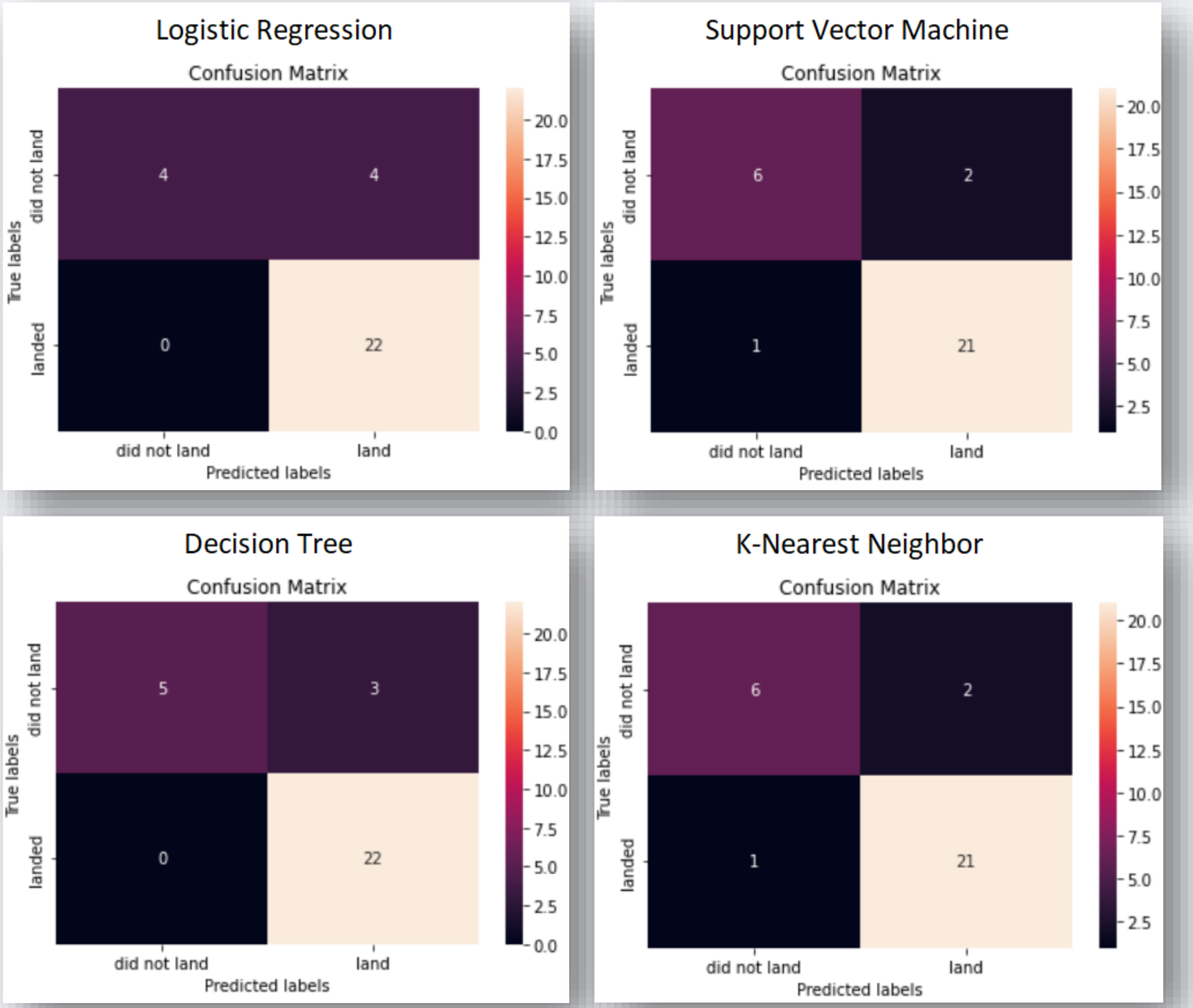
- Four predictors were evaluated for SpaceX mission success
  - K-Nearest Neighbors (KNN)
  - Support Vector Machine (SVM)
  - Logistic Regression
  - Decision Tree
- KNN performed the best with accuracy of 0.9 with the parameters shown below




```
tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 6, 'p': 1}
```

Confusion Matrix

- Confusion Matrix were created for evaluation of each type of predictor
    - The problem of false positives is clearly observable
- KNN was found to have the least number of false positives.
- KNN scored 90% on accuracy



Conclusions

- The success of launch has been studied for several factors such as launch site, orbit type especially the flight number which leads to the conclusion that success has been achieved by experience from failure.
- Launches for ES-L1, GEO, HEO, SSO orbits have been the most successful.
- Successes have increased consistently over the years since 2013
- Payload mass seems to be playing a role depending on orbit type. However in general boosters with lower payload ranges are more successful.
- SVM serves as the best predictor for the launch success evaluation.

Thank you!