IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Stany Devdas
20-Jan-2024

Outline

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

Executive Summary

Summary of methodologies

- **Data Collection**: Utilized APIs and Web Scraping for comprehensive data acquisition.
- **Data Wrangling**: Cleaned and transformed raw data to ensure its suitability for analysis.
- **Exploratory Data Analysis**: Conducted in-depth analysis using SQL and visualization techniques.
- **Visualization and Interaction**: Developed interactive maps using Folium and created dashboards using Plotly Dash for dynamic data exploration.
- **Machine Learning Predictions:** Implemented machine learning models for classification tasks.

Summary of all results

- Exploratory Data Analysis helped us achieve an enhanced understanding of **parameter relationships**.
- Visualizations, improved **comprehension of parameter** effects.
- Machine Learning, established an efficient **predictive model** with a minimum **accuracy of 83%**.

Introduction

- **Project background and context**

  Collaborating with **SpaceY** to analyze publicly accessible data from SpaceX, focusing on the performance of Falcon 9 stage 1 rockets. **Understanding the success and failure** of landing these rockets is crucial as they significantly impact launch expenses. The **ability to reuse** them has the potential to alter the competitive landscape in the rocket launch industry.

- **Problems to find answers**

  - **Identify** the factors, circumstances and **parameters** that influence the successful landing of stage 1 rockets after each deployment.
  - **Forecast** the outcome (**success or failure**) of a new rocket **landing** based on the collected parameters.
  - Evaluate the **accuracy** of the predictions made using the aforementioned parameters.

Section 1

Methodology

Methodology

Executive Summary

- **Data collection methodology**:
  - Data collection involved leveraging SpaceX's public APIs and implementing web scraping techniques.
- **Perform data wrangling**
  - The data underwent one-hot encoding to enhance its suitability for utilization in learning algorithms.
- **Perform exploratory data analysis** (EDA) using visualization and SQL
  - Uncover novel data patterns through the application of SQL and visualization techniques.
- **Perform interactive visual analytics** using Folium and Plotly Dash
  - Interactive methods in this analysis involved the utilization of Plotly Dash for dashboards and Folium maps.
- **Perform predictive analysis** using classification models
  - Various machine learning algorithms were assessed to determine the most effective method.

Data Collection

Using SpaceX API calls

1. Collecting Data using SpaceX API calls
2. Normalize Data using JSON functions
3. Preprocess columns and rows
4. Loading Dictionary and converting to DataFrame Object
5. Filter for Falcon 9 data

Data Wrangling & Extract data to CSV format

1. Response from Webpage
2. Create BeautifulSoup Object
3. Find Relevant Tables & Columns
4. Create dictionary object
5. Convert to DataFrame object

Using Web Scrapping

Data Collection – SpaceX API

Get Requests from API's

```
[6]: spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
[7]: response = requests.get(spacex_url)
```

```
[9]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetw
response = requests.get(static_json_url)
```

Normalizing using JSON functions

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

```
[12]: # Get the head of the dataframe
data.head()
```

[12]:

	static_fire_date_utc	static_fire_date_unix	tbd	net	window	rocket	success	details	cre
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	False	0.0	5e9d0d95eda69955f709d1eb	False	Engine failure at 33 seconds and loss of vehicle	

Pre-processing Data

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

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replac
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

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

Loading Data in Dictionary and converting to DataFrame

```
[21]: # Create a data from launch_dict
launch_pd= pd.DataFrame.from_dict(launch_dict)
```

Filtering for Falcon 9 Data

```
[23]: data_falcon9=launch_pd[launch_pd['BoosterVersion'].str.contains('Falcon 9')]
```

```
[24]: data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

[24]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Leg:
4	1	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False
5	2	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False

- Github Link: https://github.com/StanyDevdas/IBM_DS_CapstoneProjectSpaceX_StanyDevdas/blob/1d477a8b34a0632d6ed96f19d80479a96f0fc1fe/SpaceX_DataCollection_API.ipynb

Data Collection – Scraping

Response from Webpage

```
[4]: static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1
```

```
[5]: # use requests.get() method with the provided static_url
response = requests.get(static_url)
# assign the response to a object
resp_object=response.content
```

Create BeautifulSoup object

```
[6]: # Use BeautifulSoup() to create a BeautifulSoup object

soup=BeautifulSoup(resp_object, 'html5lib')
```

Find Relevant Tables and Columns

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

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

```
<table class="wikitable plainrowheaders collapsible" style="width: 100%;">
<tbody><tr>
<th scope="col">Flight No.
</th>
<th scope="col">Date and<br/>time (<a href="/wiki/Coordinated_Universal_Time" title="Coordina
Time">UTC</a>)
</th>
<th scope="col"><a href="/wiki/List_of_Falcon_9_first-stage_boosters" title="List of Falcon 9
```

```
[10]: 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_name
for row in first_launch_table.find_all('th'):
    temp_col_name = extract_column_from_header(row)
    if (temp_col_name!=None and len(temp_col_name)>0):
        column_names.append(temp_col_name)
```

```
[11]: print(column_names)

['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch
outcome']
```

Create and load Dictionary Object

```
[12]: launch_dict= dict.fromkeys(column_names)
```

```
[13]: 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)
            datatimelist=date_time(row[0])
```

```
1
4 June 2010
18:45
F9 v1.0B0003.1
CCAFS
Dragon Spacecraft Qualification Unit
Dragon Spacecraft Qualification Unit
LEO
SpaceX
Success

Failure
2
```

Convert to DataFrame Object

```
[14]: df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })

[15]: df.shape

[15]: (121, 11)
```

GitHub Link:
https://github.com/StanyDevdas/IBM_DS_CapstoneProjectSpaceX_StanyDevdas/blob/1d477a8b34a0632d6ed96f19d80479a96f0fc1fe/SpaceX_DataCollection_Scrapping.ipynb

Data Wrangling

Reading the Data

```
[2]: df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetw
     df.head(10)
```

Checking Launch Sites and Orbit Counts

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

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

```
[6]: # Apply value_counts on Orbit column
     df['Orbit'].value_counts()

[6]: 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
```

Identifying Bad Outcomes

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

[9]: {'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

Categorizing success and failure of landing into new column class

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

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

[13]:

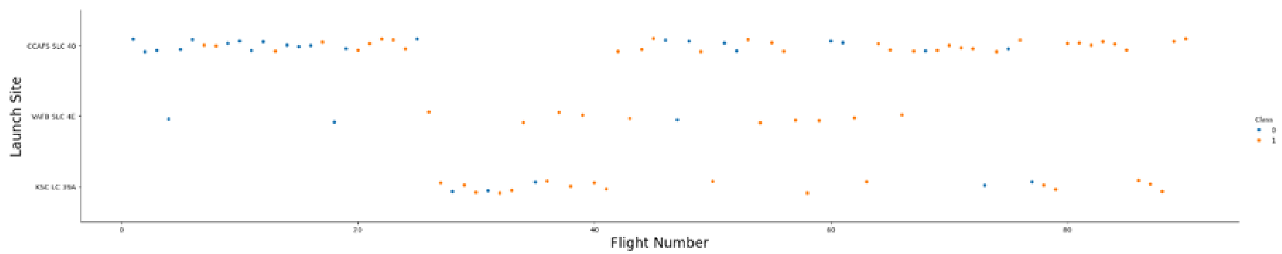
	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

GitHub Link:
https://github.com/StanyDevdas/IBM_DS_CapstoneProjectSpaceX_StanyDevdas/blob/1d477a8b34a0632d6ed96f19d80479a96f0fc1fe/SpaceX_DataWrangling.ipynb

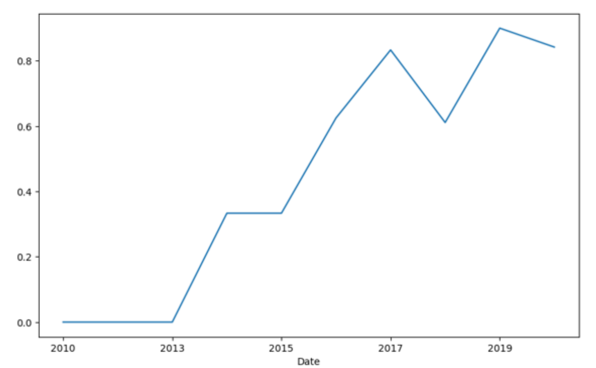
EDA with Data Visualization

Part 1

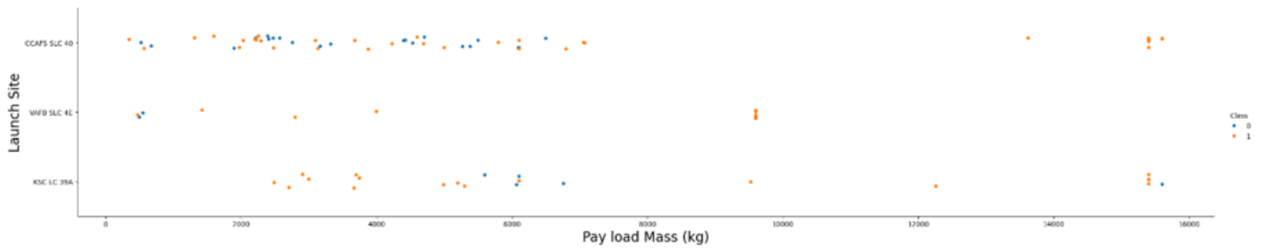
Relationship between Flight Number, Launch Site and landing success/failure class.



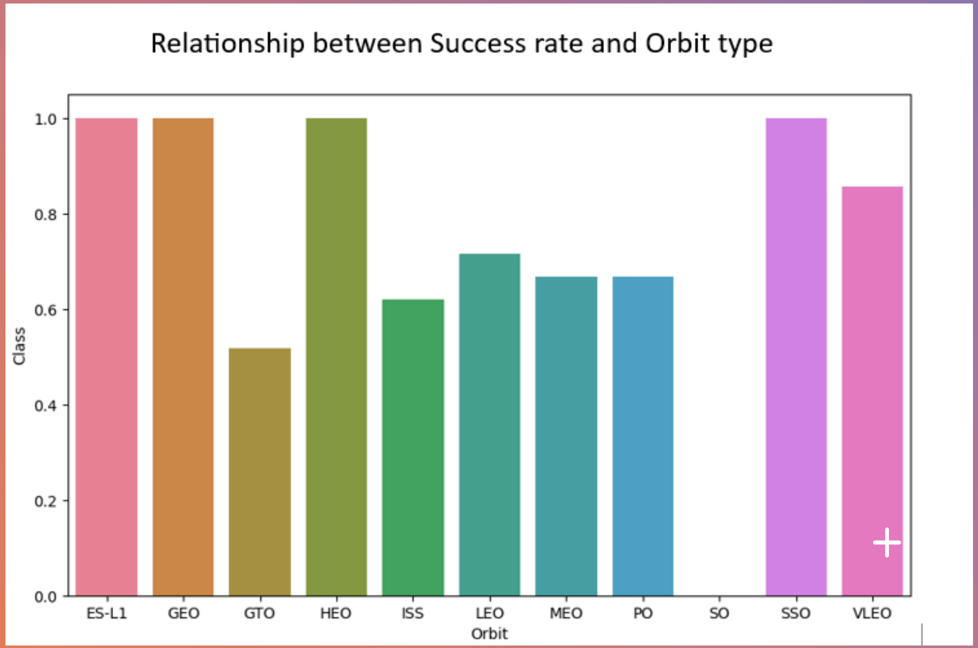
Launch success yearly trend.

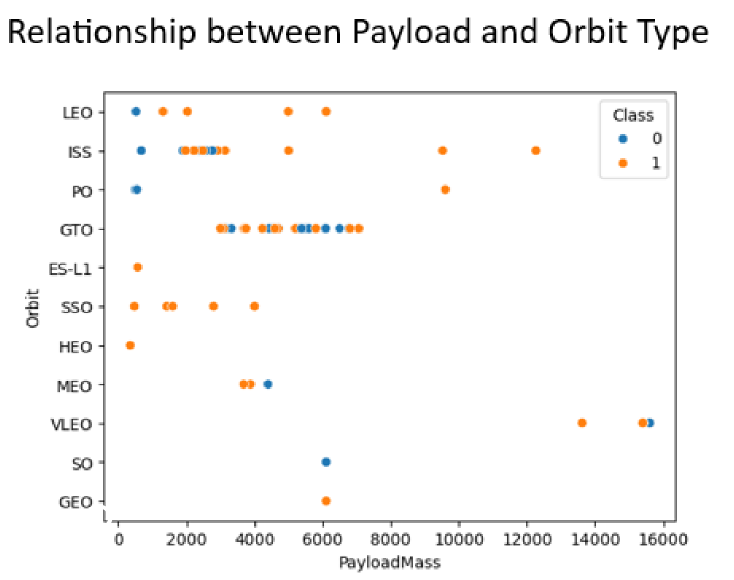


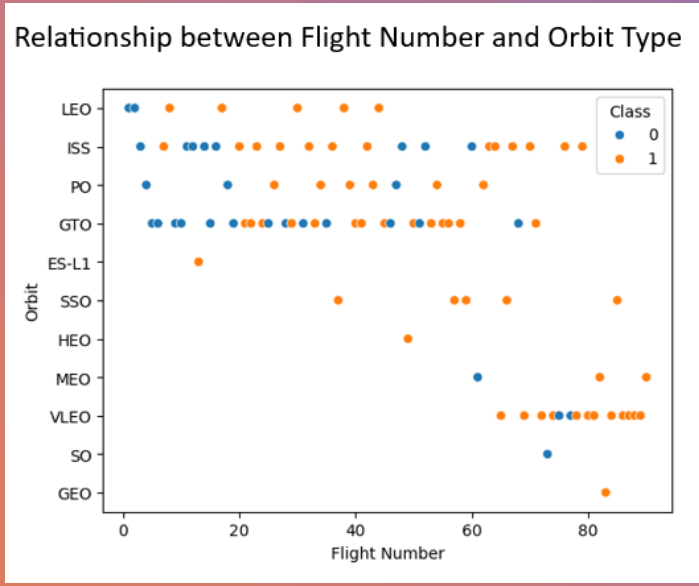
Relationship between Payload, Launch Site and landing success/failure class.



EDA with Data Visualization Part 2







GitHub Link:
https://github.com/StanyDevdas/IBM_DS_CapstoneProjectSpaceX_StanyDevdas/blob/1d477a8b34a0632d6ed96f19d80479a96f0fc1fe/SpaceX_EDA_Visualizations.ipynb

EDA with SQL

Used SQL queries for the following tasks

- Display the names of the unique launch sites in the space mission
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome in ground pad was achieved.
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster versions which have carried the maximum payload mass.
- List the records which will display the month names, failure landing outcomes in drone ship ,booster versions, launch site for the months in year 2015.
- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

GitHub Link:
https://github.com/StanyDevdas/IBM_DS_CapstoneProjectSpaceX_StanyDevdas/blob/1d477a8b34a0632d6ed96f19d80479a96f0fc1fe/SpaceX_EDA_SQLlite.ipynb

Built an Interactive Map with Folium

- 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, using folium.Circle and folium.Marker
- 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 Cluster , we identified which launch sites have relatively high success rate.
- 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/StanyDevdas/IBM_DS_CapstoneProjectSpaceX_StanyDevdas/blob/1d477a8b34a0632d6ed96f19d80479a96f0fc1fe/SpaceX_LaunchSite_MapAnalysis.ipynb

Built a Dashboard with Plotly Dash

- Built an interactive dashboard with Plotly Dash
- Plotted pie charts showing the total successful launches by All sites or selected sites (selected from dropdown).
- Plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version ranges (with dynamic payload range slider).

GitHub Link:
https://github.com/StanyDevdas/IBM_DS_CapstoneProjectSpaceX_StanyDevdas/blob/1d477a8b34a0632d6ed96f19d80479a96f0fc1fe/SpaceX_Dash.py

Predictive Analysis (Classification)

1

Building the model

- Create array Y for output class
- Standardize the data
- Split into training and testing set
- Build GridSeachCV model and fit the data

2

Evaluating the model

- Calculating the accuracies
- Calculating the confusion matrix
- Plot the results

3

Finding the optimal model

- Find the best hyperparameters for the model
- Find the best model with highest accuracy on testing data
- Confirm the optimal model

GitHub Link:
https://github.com/StanyDevdas/IBM_DS_CapstoneProjectSpaceX_StanyDevdas/blob/1d477a8b34a0632d6ed96f19d80479a96f0fc1fe/SpaceX_Machine_Learning_Prediction.ipynb

Results

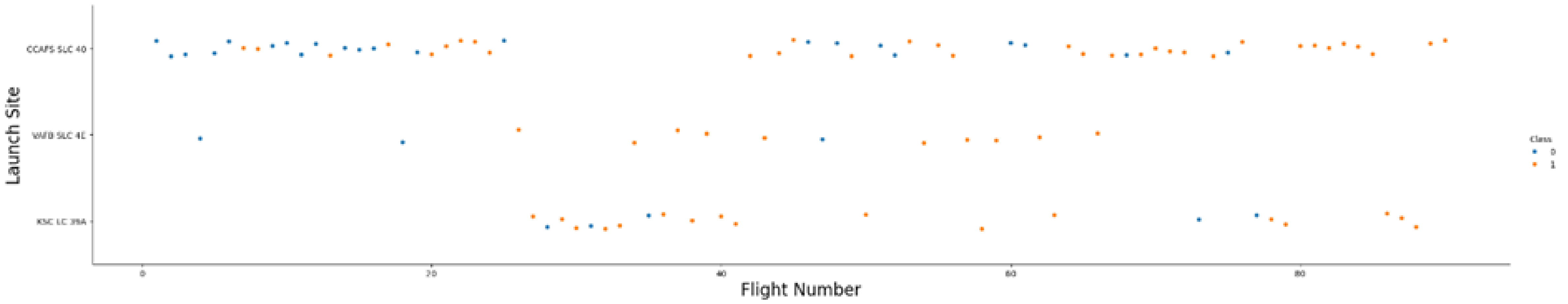
- The SVM, KNN and Logistic regression models are the best in terms of prediction accuracy
- Low weighted payloads perform better than the heavier payloads
- The success rates of SpaceX launches have significantly improved over the years
- KSC LC 39A had the most successful launches from all the sites
- Orbits GEO, HEO, SSO, ES L1 have the best success rates

Section 2

Insights drawn from EDA

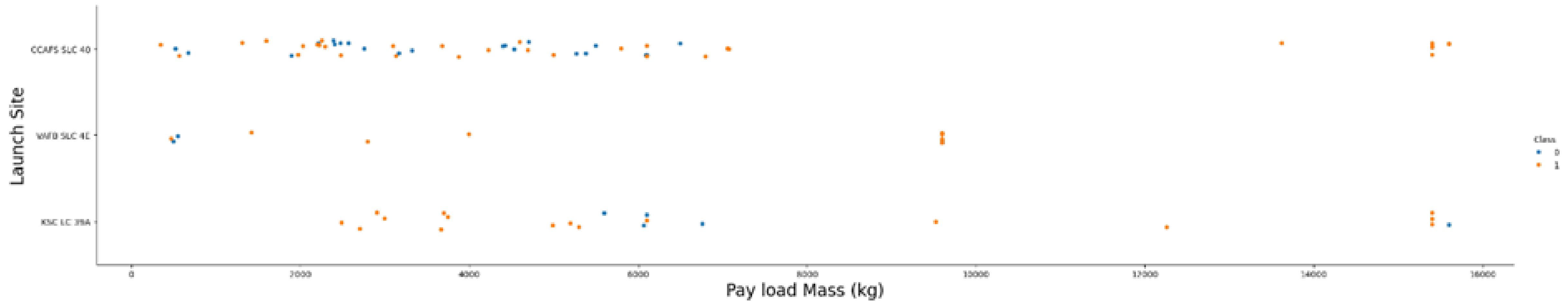
Flight Number vs. Launch Site

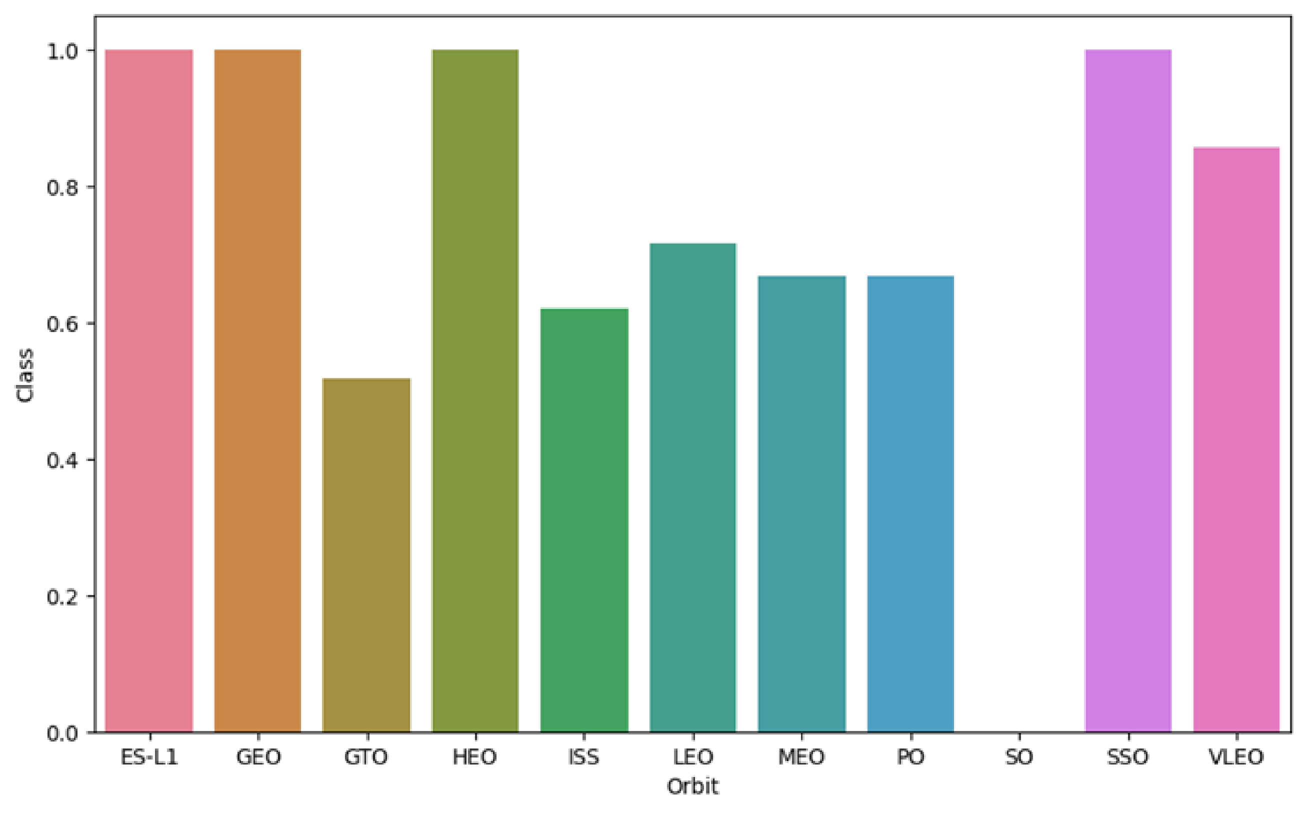
- Launches from the site of CCAFS SLC 40 are significantly higher than launches from other sites
- With the increase in flight number, the success rates also increases



Payload vs. Launch Site

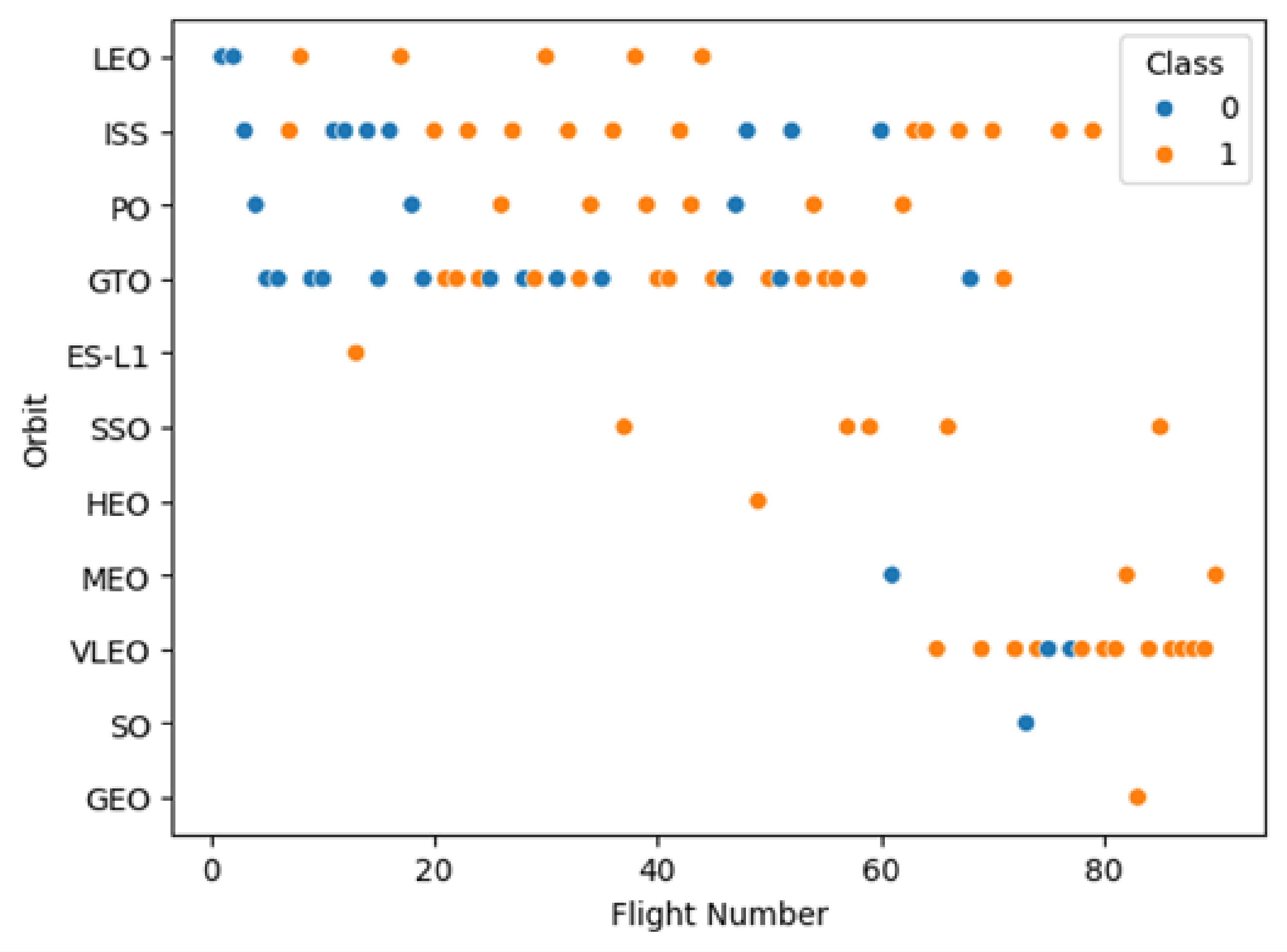
- Majority of Pay loads with lower Mass range have been launched from CCAFS SLC 40





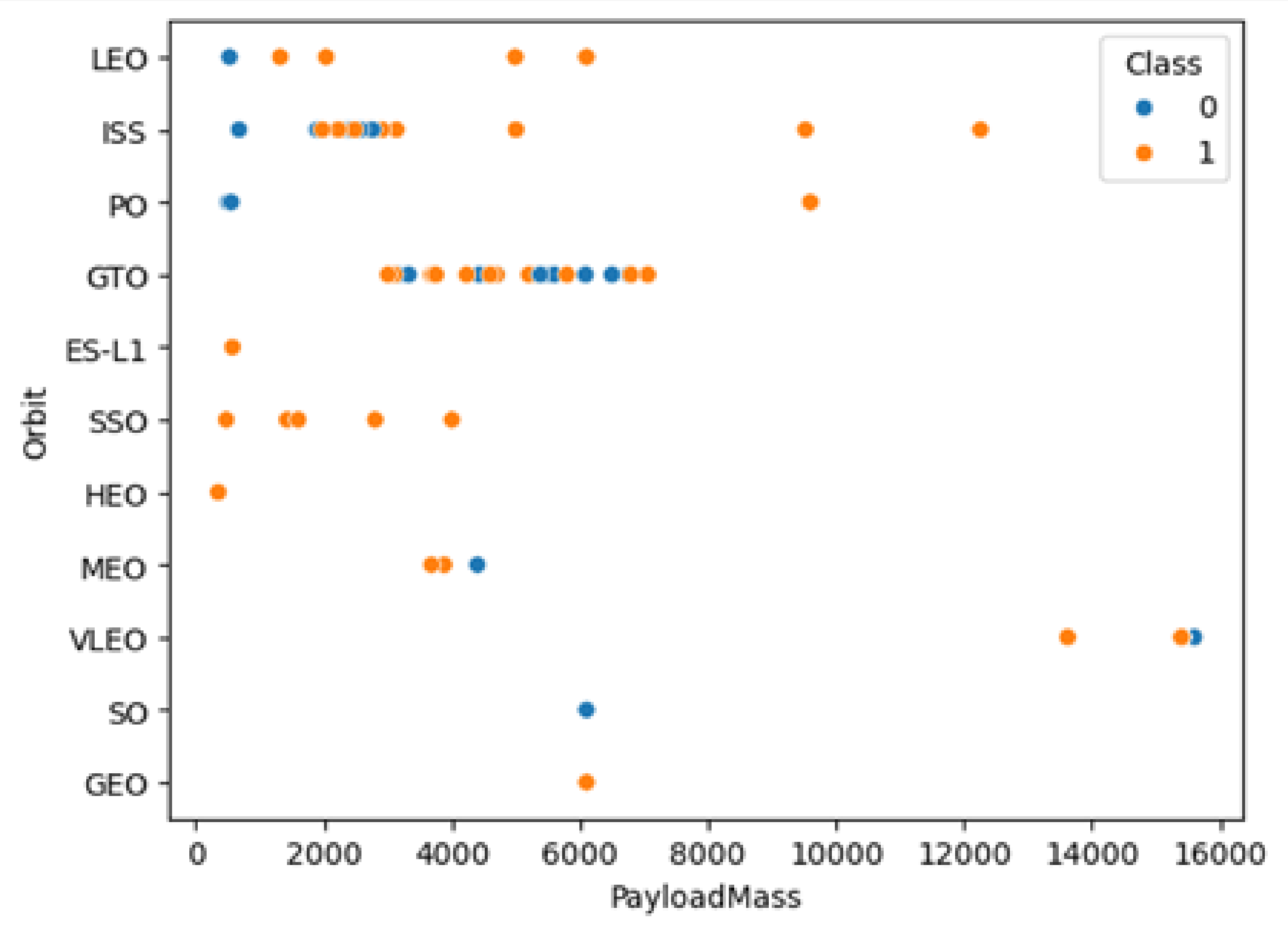
Success Rate vs. Orbit Type

- Following orbits have high success rate (100%)
  - ES-L1
  - GEO
  - HEO
  - SSO
- SO has the least success (0%)



Flight Number vs. Orbit Type

- A trend can be observed where the latest launches have been shifted to VLEO orbit with good success rates

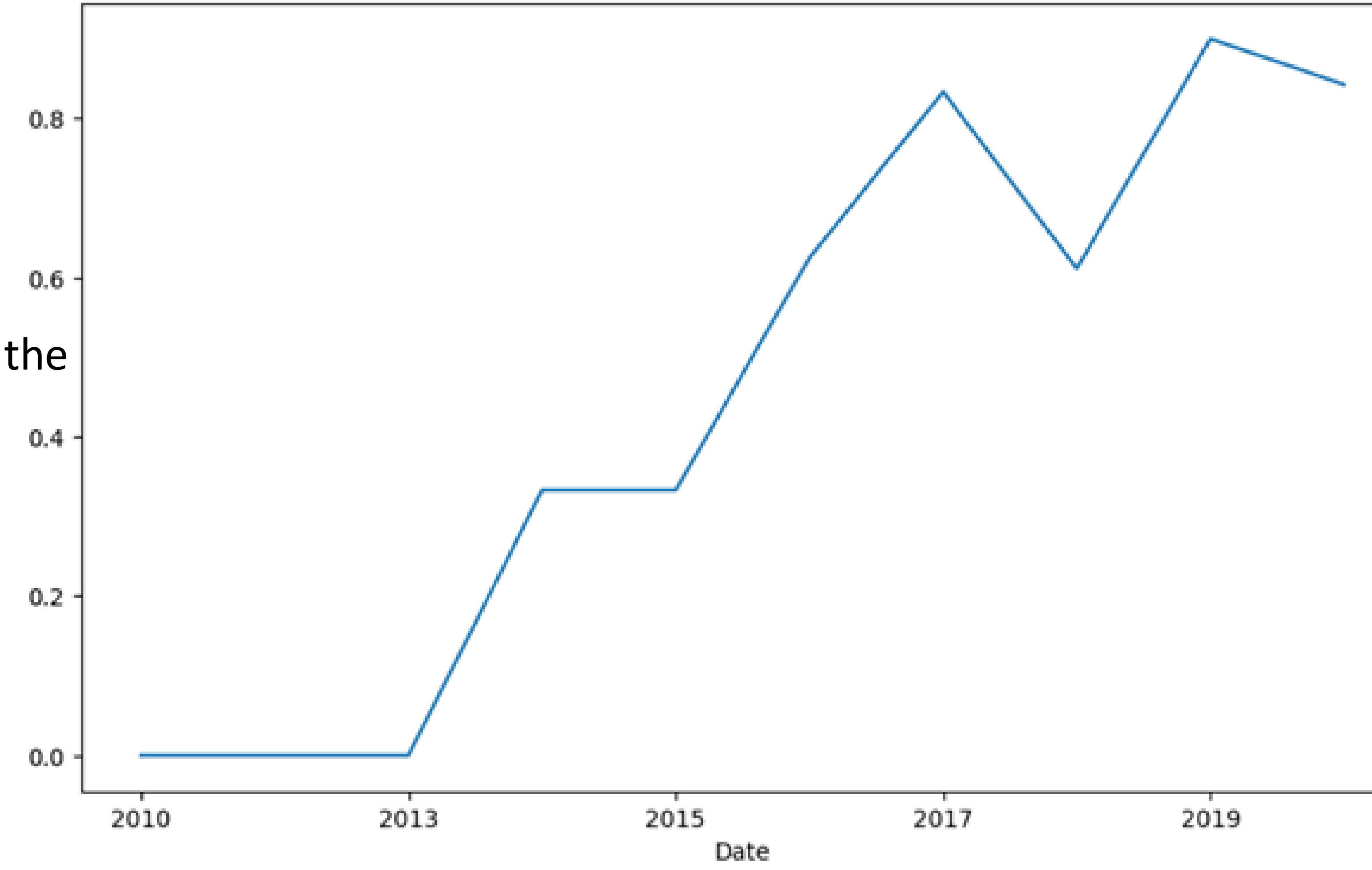


Payload vs. Orbit Type

- Co-relation between ISS orbit and payloads between range of 2000 and 4000 kgs.
- Also a relationship between the GTO orbit and payload range of 3000-7000 kgs.

Launch Success Yearly Trend

- Success rates are significantly increasing over the years, even with the dip and failures around 2018



All Launch Site Names

Using DISTINCT keyword for listing unique names

```
[8]: %sql select distinct "Launch_Site" from SPACEXTABLE;

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

[8]:

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

Launch Site Names Begin with 'CCA'

Used keyword LIMIT to display only 5 records

```
%sql select * from SPACEXTABLE where "Launch_Site" like 'CCA%' LIMIT 5
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	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

Used the keyword SUM.

```
%sql select sum(PAYLOAD_MASS__KG_) as "Total_Payload_Mass" from SPACEXTABLE where "Customer"='NASA (CRS)';
```

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

Total_Payload_Mass
45596

Average Payload Mass by F9 v1.1

Used the keyword AVG

```
%sql select avg(PAYLOAD_MASS__KG_) as "Average_Payload" from SPACEXTABLE where "Booster_Version" like 'F9 v1.1%'
```

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

Average_Payload
2534.6666666666665

First Successful Ground Landing Date

Used the keyword MIN on the date column with the appropriate condition to find the required parameter

```
%sql select min(Date) from SPACEXTABLE where "Landing_Outcome"="Success (ground pad)"
```

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

min(Date)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

Displaying list the names of boosters which have successfully landed on drone ship and had payload mass in between 4000 and 6000

```
%%sql
select distinct Booster_Version from SPACEXTABLE where "Landing_Outcome"="Success (drone ship)" and
PAYLOAD_MASS__KG_ between 4000 and 6000
```

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

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

Total Number of Successful and Failure Mission Outcomes

- Used the query with GROUP BY feature
- Results show 10 Failures and 61 Successful landing

```
%%sql
select sum(C) as "Total", Landing_Outcome from
    (select count(*) as C,SUBSTR(Landing_Outcome,1,7) as Landing_Outcome
        from SPACEXTABLE
        where (SUBSTR(Landing_Outcome,1,7) like ('Success%') or
                SUBSTR(Landing_Outcome,1,7) like ('Failure%'))
        group by Landing_Outcome)
    group by Landing_Outcome
```

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

Total	Landing_Outcome
10	Failure
61	Success

Boosters Carried Maximum Payload

- Using Sub-query to get the distinct list of booster version meeting the payload conditions

```
%%sql

select distinct Booster_Version from SPACEXTABLE
    where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACEXTABLE) order by 1
```

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

Booster_Version
F9 B5 B1048.4
F9 B5 B1048.5
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1049.7
F9 B5 B1051.3
F9 B5 B1051.4
F9 B5 B1051.6
F9 B5 B1056.4
F9 B5 B1058.3
F9 B5 B1060.2
F9 B5 B1060.3

2015 Launch Records

- Matched the 2015 records using SUBSTR function in the where conditions

```
%%sql
select REPLACE(REPLACE(substr(Date,6,2),'01','January'),'04','April') as Month_Names,
       Landing_Outcome, Booster_Version, Launch_Site
       from SPACEXTABLE where substr(Date,0,5)='2015' and Landing_Outcome='Failure (drone ship)'
```

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

Month_Names	Landing_Outcome	Booster_Version	Launch_Site
January	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
April	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

- Using the RANK function on the descending order of count function used to count number of landing outcomes in the given date range.

```
%%sql
select RANK () OVER (
    ORDER BY count(*) desc) Rank, count(*) Count, Landing_Outcome from SPACEXTABLE
    where Date between '2010-06-04' and '2017-03-20'
    group by Landing_Outcome order by 1
```

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

Rank	Count	Landing_Outcome
1	10	No attempt
2	5	Success (drone ship)
2	5	Failure (drone ship)
4	3	Success (ground pad)
4	3	Controlled (ocean)
6	2	Uncontrolled (ocean)
6	2	Failure (parachute)
8	1	Precluded (drone ship)

Section 3

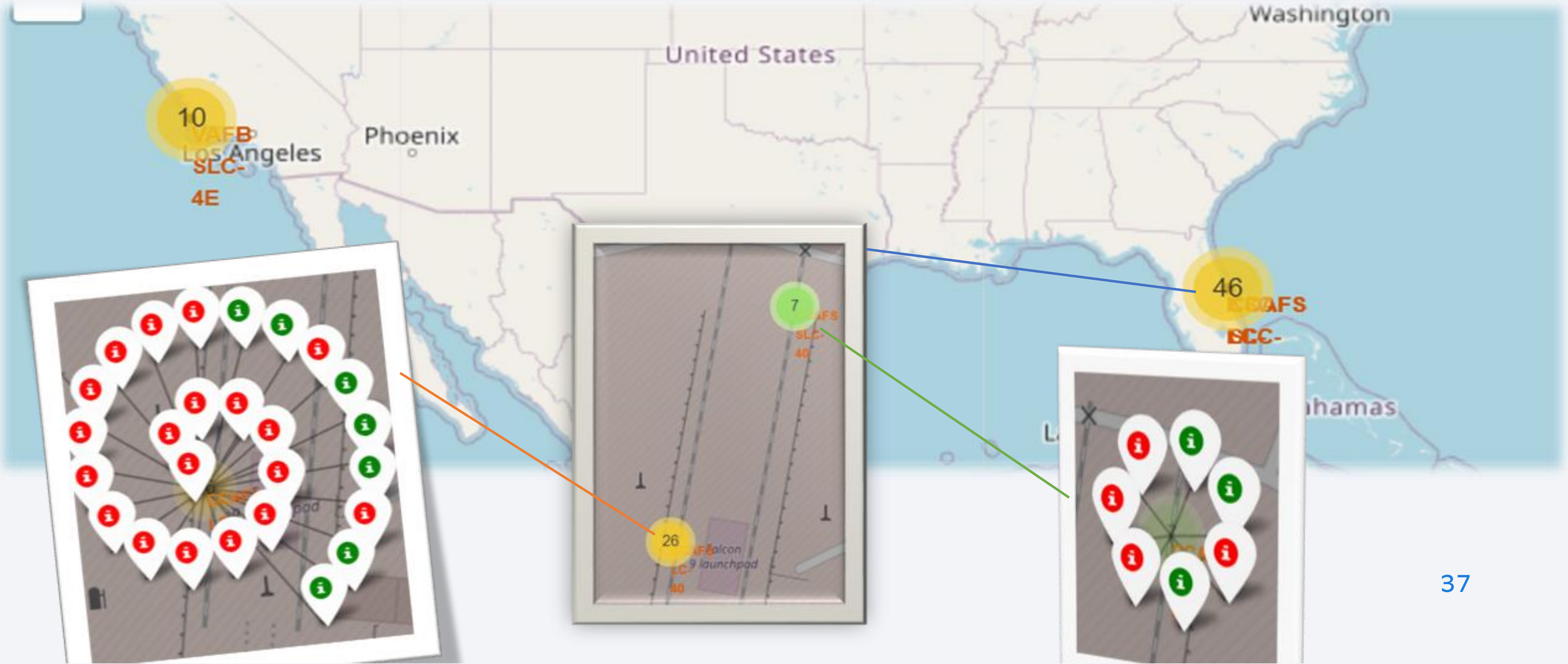
Launch Sites Proximities Analysis

Launch Sites for SpaceX

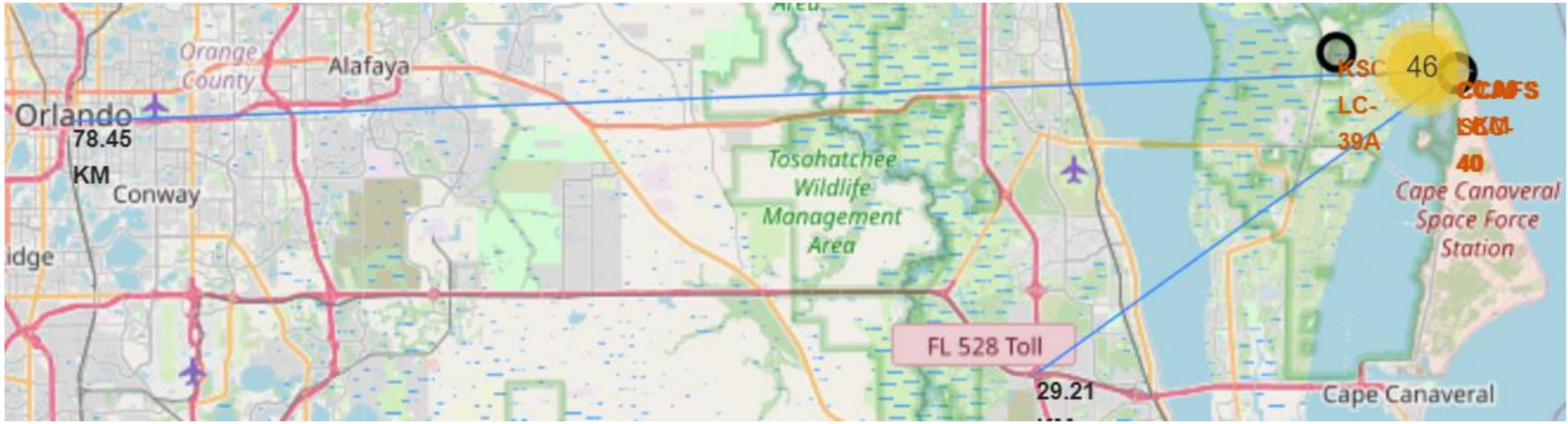
- All SpaceX rockets are launched from the coastal lines of Florida and California in the United States of America

Markers showing launch sites with color labels

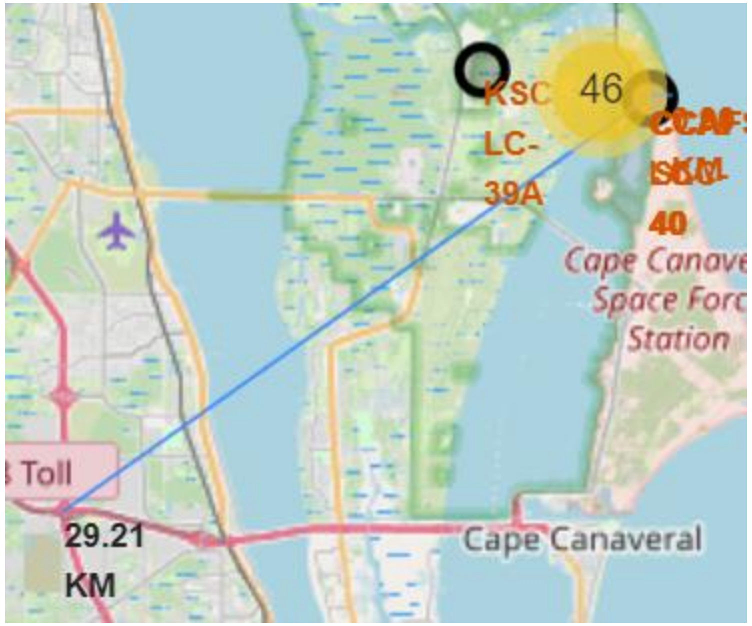
- Green label show successful launch



Distances between a launch sites to its proximities



- All infrastructures like, railway line, highway, coastal line and nearest city are in close proximities

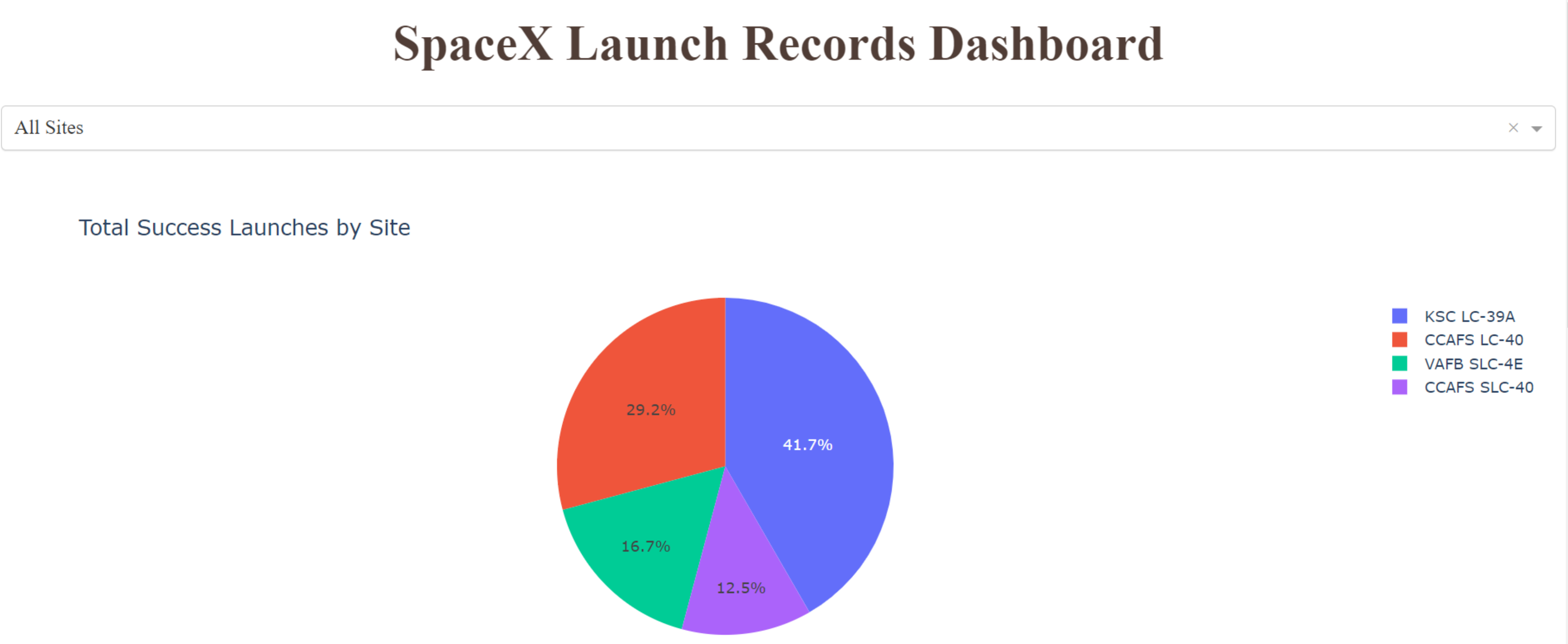


Section 4

Build a Dashboard with Plotly Dash

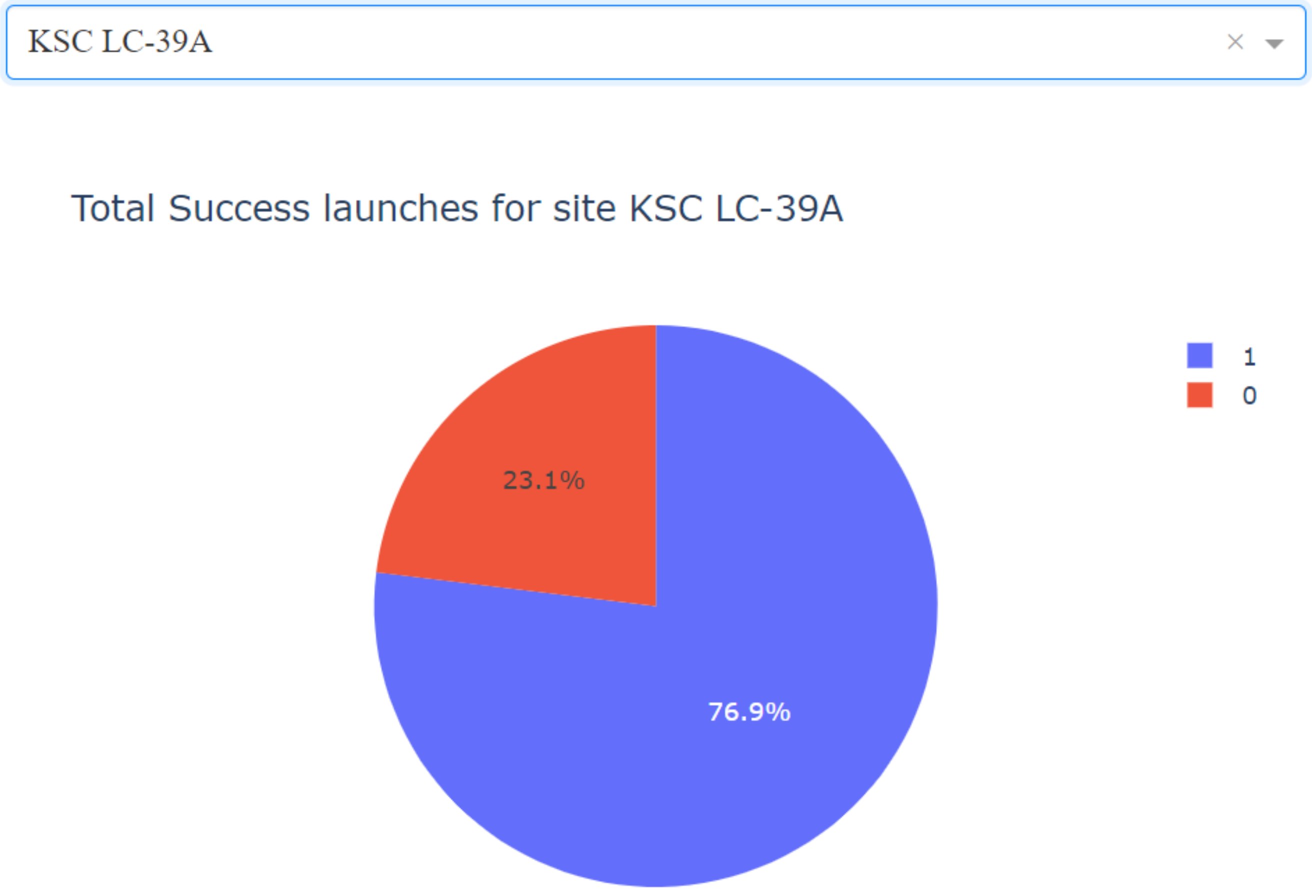
Total success Launches by all sites

- KSC LC-39A has had the most successful launches in comparison to all the launch sites.



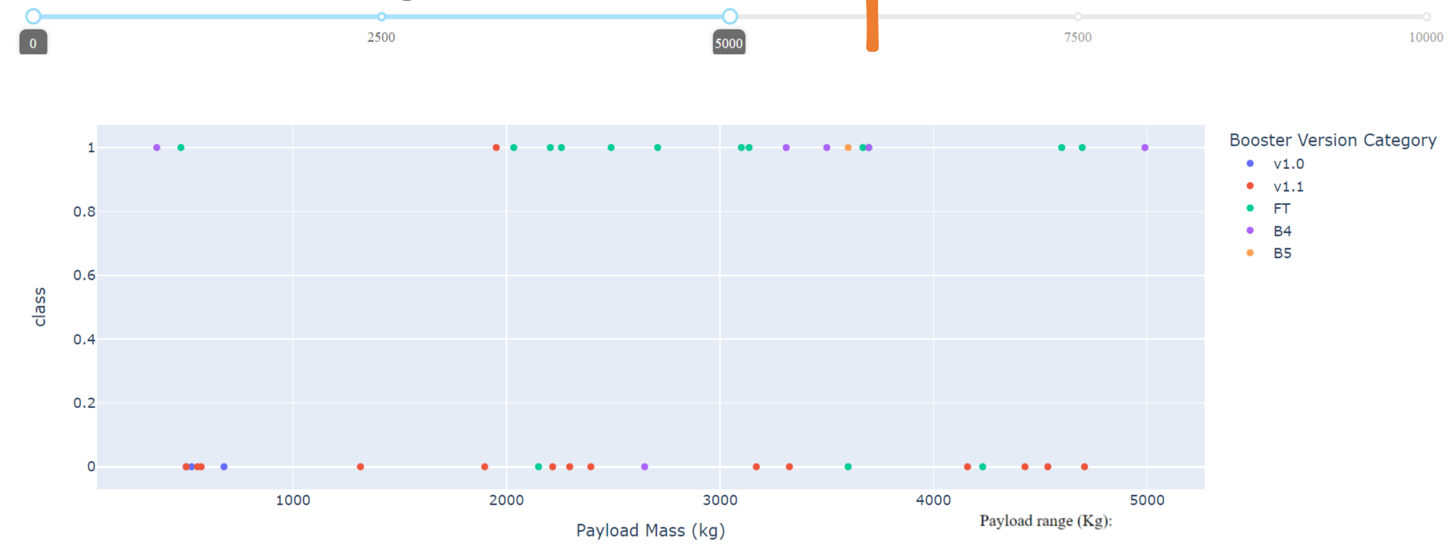
Success Rate by Site (KSC LC-39A)

- KSC LC-39A has a 76.9% success rate as a launch site

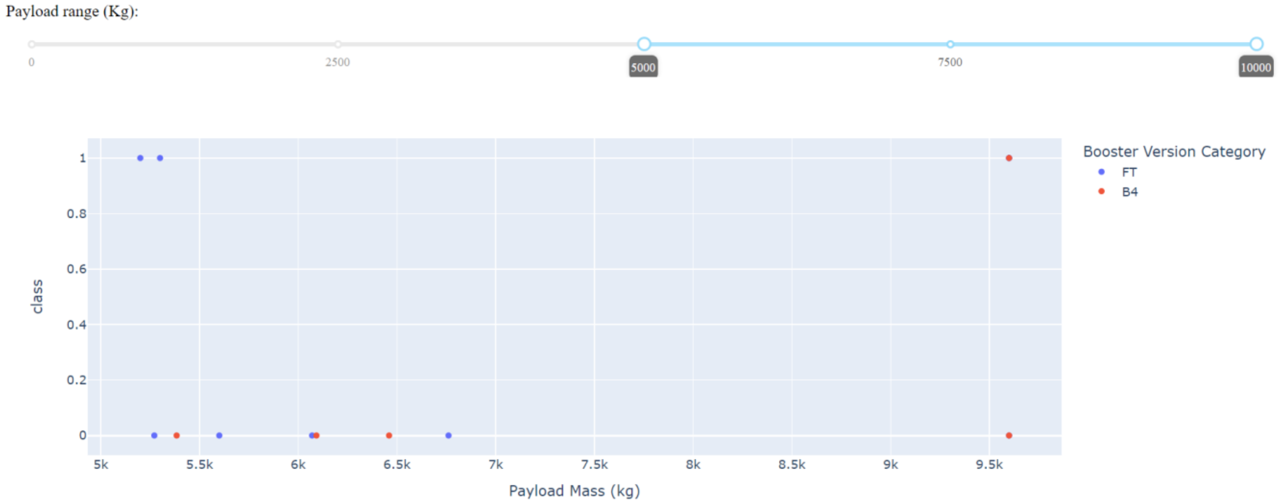


Success rates on variable payload using Dash Rangeslider

Payload range (Kg):



- Payload (0-5000kg) range for all sites



Payload (5000-10000kg) range for all sites

Section 5

Predictive Analysis (Classification)

Classification Accuracy

As seen in screenshot, all algorithms have similar accuracy

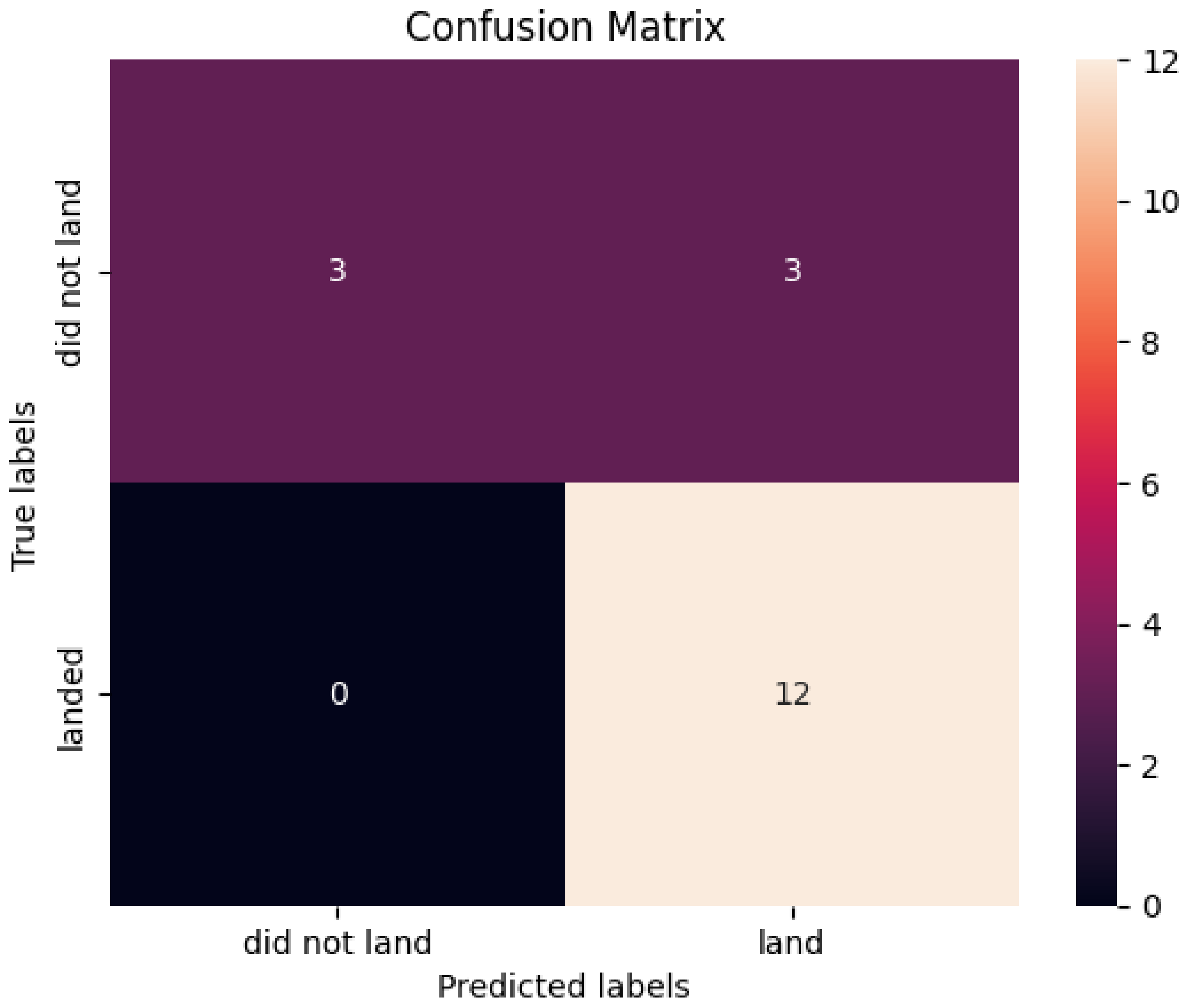
We select K-Nearest Neighbours as the preferred algorithm

```
print("Logistic Regression:",logreg_cv.score(X_test,Y_test))
print("SVM:",svm_cv.score(X_test,Y_test))
print("Decision Tree Classifier:",tree_cv.score(X_test,Y_test))
print("K-Nearest Neighbours:",knn_cv.score(X_test,Y_test))
```

```
Logistic Regression: 0.8333333333333334
SVM: 0.8333333333333334
Decision Tree Classifier: 0.8333333333333334
K-Nearest Neighbours: 0.8333333333333334
```

Confusion Matrix Of KNN Model

Matrix shows that inaccuracies in the model result in displaying False negatives, but never result in False positives



Conclusions

In summary:

- The launch success rate exhibited a consistent increase from 2013 to 2020.
- Lighter payloads demonstrated higher success rates compared to heavier payloads.
- Orbits such as ES-L1, GEO, HEO, SSO, and VLEO achieved the highest success rates.
- KSC LC-39A emerged as the site with the highest number of successful launches.
- The K-Nearest Neighbor algorithm proved to be the most effective for this specific task.

Thank you!