Winning Space Race with Data Science

Vaibhav Pavtekar
20 Sep 2024

Outline

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

Executive Summary

- Summary of methodologies
  - Project data collection
    - API
    - Web Scraping
  - Data Wrangling
  - Exploratory Analysis
    - Using SQL
    - Using Data Visualization
  - Machin Learning Predictions
- Summary of all results
  - Exploratory Analysis Result
  - Predictive Analysis Results

Introduction

- Project background and context

  Space X advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because Space X can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against space X for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

- Problems you want to find answers
  - What factors determine if the rocket will land successfully?
  - The interaction amongst various features that determine the success rate of a landing.
  - Identifying operating conditions to ensure a successful landing program.

Section 1

Methodology

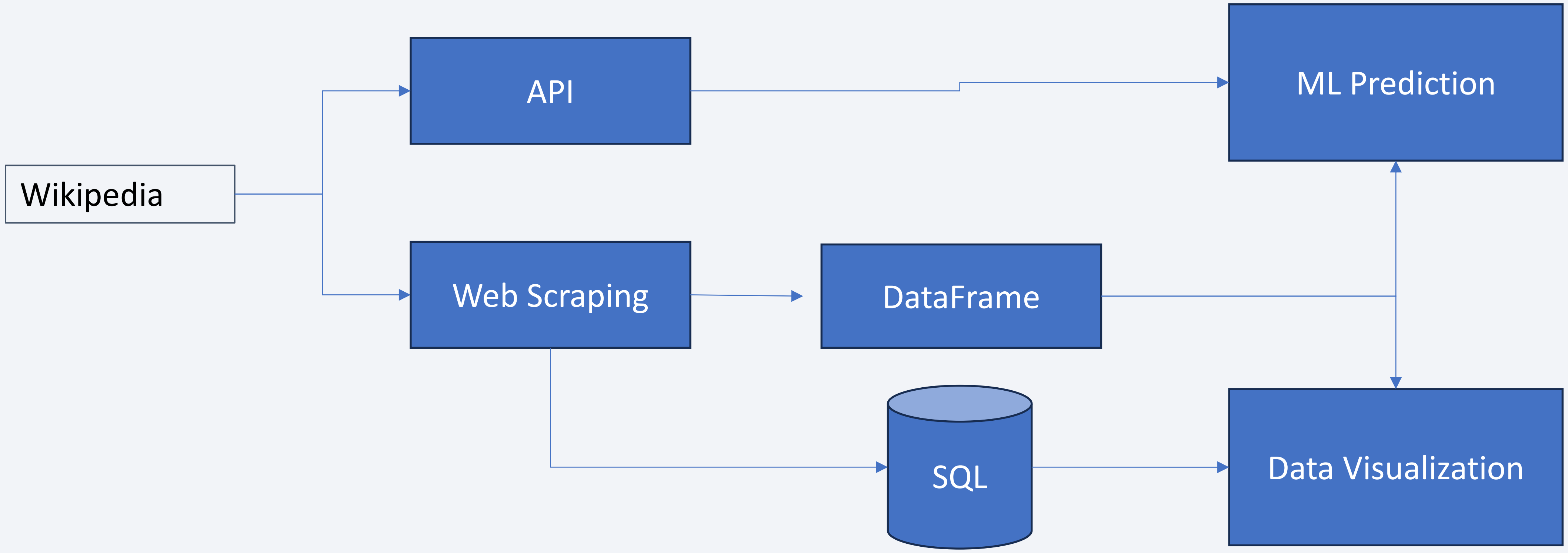
Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX API and web scraping from Wikipedia.
- Perform data wrangling
  - One-hot encoding was applied to categorical features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- Data collection was done using get request to the SpaceX API and Web Scraping method in some cases.



Data Collection – SpaceX API

- Data collected using Rest API and using function and transformed to DataFrame for Presentation
- GitHub URL :

https://github.com/maxh8086/Data-Science-Project/blob/main/jupyter-labs-spacex-data-collection-api%20(1).ipynb

From the `rocket` column we would like to learn the booster name.

In [19]:
```python
# Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
       if x:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

From the `launchpad` we would like to know the name of the launch site being used, the logitude, and the latitude.

In [20]:
```python
# Takes the dataset and uses the launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['launchpad']:
       if x:
         response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
         Longitude.append(response['longitude'])
         Latitude.append(response['latitude'])
         LaunchSite.append(response['name'])
```

From the `payload` we would like to learn the mass of the payload and the orbit that it is going to.

In [21]:
```python
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
       if load:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

From `cores` we would like to learn the outcome of the landing, the type of the landing, number of flights with that core, whether gridfins were used, wheter the core is reused, wheter legs were used, the landing pad used, the block of the core which is a number used to seperate version of cores, the number of times this specific core has been reused, and the serial of the core.

In [22]:
```python
# Takes the dataset and uses the cores column to call the API and append the data to the lists
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
```

Now, let's apply `getBoosterVersion` function method to get the booster version

In [33]:
```python
# Call getBoosterVersion
getBoosterVersion(data)
```

the list has now been update

In [34]:
```python
BoosterVersion[0:5]
```

Out[34]:
```
['Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 9']
```

we can apply the rest of the functions here:

In [35]:
```python
# Call getLaunchSite
getLaunchSite(data)
```

In [36]:
```python
# Call getPayloadData
getPayloadData(data)
```

In [37]:
```python
# Call getCoreData
getCoreData(data)
```

Data Collection - Scraping

- We collected data using Web Scraping to Cloud object, Transform collected data in to necessary Format and Performed data wrangling and presentation.
- GitHub URL of Capstone Project :

https://github.com/maxh8086/Data-Science-Project

Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

In [26]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/AF
```

We should see that the request was successfull with the 200 status response code

In [27]:
```
response.status_code
```

Out[27]: 200

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

In [28]:
```
# Use json_normalize meethod to convert the json result into a dataframe
import pandas as pd
from pandas import json_normalize

response_json = response.json()
data = json_normalize(response_json)
```

Using the dataframe `data` print the first 5 rows

In [29]:
```
# Get the head of the dataframe
data.head()
```

Out[29]:

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships	c
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 33, 'altitude': None, 'reason': 'merlin engine	Engine failure at 33 seconds and loss of	[]	[]	

Data Wrangling

- Data was collected through Web scraping, Performed Data wrangling to remove Null values and missing values.

- GitHub Project URL :

  https://github.com/maxh8086/Data-Science-Project/blob/main/jupyter-labs-spacex-data-collection-api%20(1).ipynb

Data Wrangling

We can see below that some of the rows are missing values in our dataset.

In [64]:
```
data_falcon9.isnull().sum()
```

Out[64]:
```
FlightNumber       0
Date               0
BoosterVersion     0
PayloadMass        5
Orbit              0
LaunchSite         0
Outcome            0
Flights            0
GridFins           0
Reused             0
Legs               0
LandingPad        26
Block              0
ReusedCount        0
Serial             0
Longitude          0
Latitude           0
dtype: int64
```

Before we can continue we must deal with these missing values. The `LandingPad` column will retain None values to represent when landing pads were not used.

Task 3: Dealing with Missing Values

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace `np.nan` values in the data with the mean you calculated.

In [69]:
```
# Calculate the mean value of PayloadMass column
mean_payload_mass = data_falcon9['PayloadMass'].mean()

print(mean_payload_mass)
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].fillna(mean_payload_mass, inplace=True)
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

```
6123.547647058824
```

EDA with Data Visualization

- Scatter point graph was plotted to visualize the relationship between Payload and Launch Site
- GitHub URL :

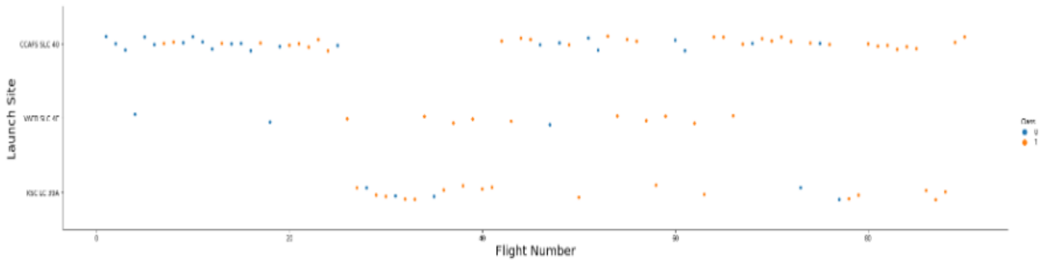
https://github.com/maxh8086/Data-Science-Project/blob/main/EDA%20with%20Data%20Visualization.ipynb

TASK 2: Visualize the relationship between Payload and Launch Site

We also want to observe if there is any relationship between launch sites and their payload mass.

In [5]:

```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the cla
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```

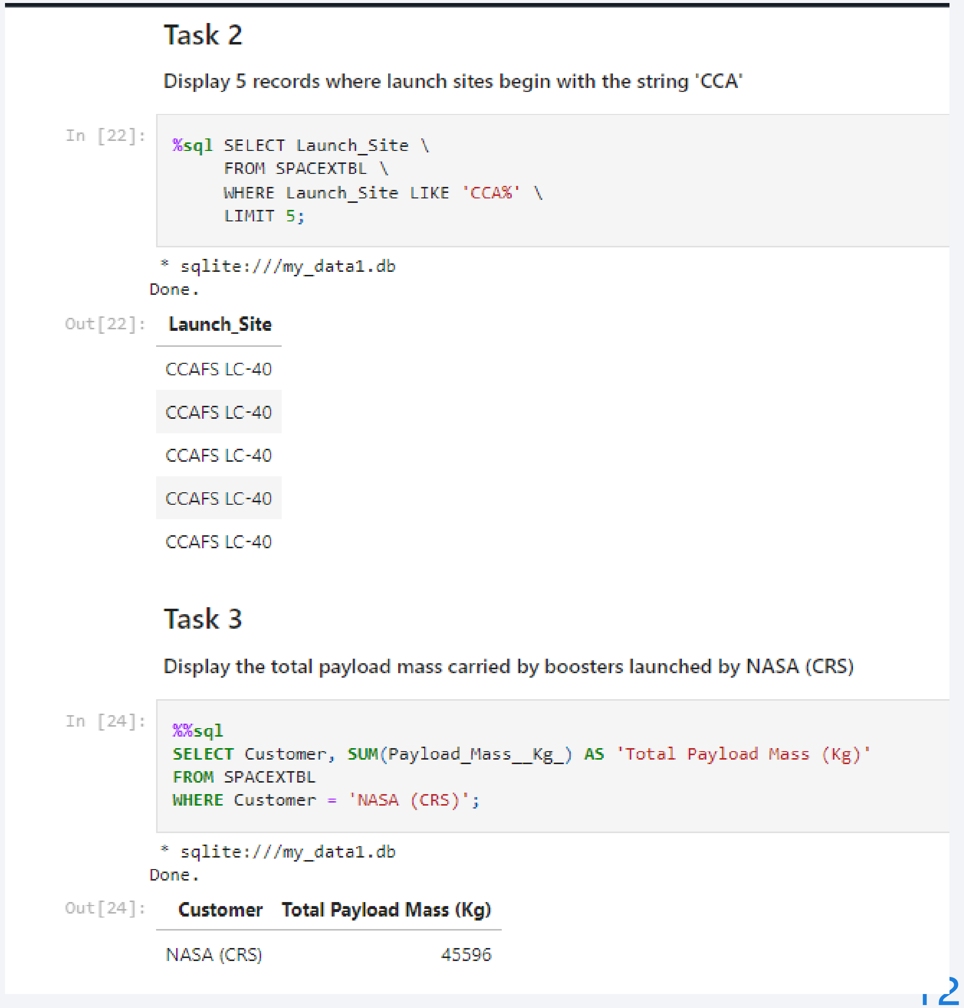


Now if you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

EDA with SQL

- SQL Queries used to perform Exploratory analysis
  - Select Distinct Launch_Site from SPACEXTBL
  - SELECT Launch_Site FROM SPACEXTBL WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
  - SELECT Customer, SUM(Payload_Mass__Kg_) AS 'Total Payload Mass (Kg)' FROM SPACEXTBL WHERE Customer = 'NASA (CRS)';
- GitHub URL :

  https://github.com/maxh8086/Data-Science-Project/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb



Task 2

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

In [22]:
```
%sql SELECT Launch_Site \
     FROM SPACEXTBL \
     WHERE Launch_Site LIKE 'CCA%' \
     LIMIT 5;
```

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

Out[22]:

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Task 3

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

In [24]:
```
%%sql
SELECT Customer, SUM(Payload_Mass__Kg_) AS 'Total Payload Mass (Kg)'
FROM SPACEXTBL
WHERE Customer = 'NASA (CRS)';
```

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

Out[24]:

Customer	Total Payload Mass (Kg)
NASA (CRS)	45596

Build an Interactive Map with Folium

- All launch sites was marked and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- Assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- identified which launch sites have relative high success rate using the color-labeled marker clusters.
- Calculated the distances between a launch site to its proximities.
- GitHub URL: https://github.com/maxh8086/Data-Science-Project/blob/main/lab-jupyter-launch-site-location-v2.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 URL : https://github.com/maxh8086/Data-Science-Project/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- To perform and analyse predictive analysis performance data Was collected using Blob and converted to DataFrame, Data was Transformed for Analysis using Machine Learning Model sklearn, Numpy. Data was split for training and testing multiple model for comparison of performance. Hypothecation analysis was performed using Logistic Regression, Support Vector Machine, Decision Tree classifier, Kneighbors Classifier where Logistic Regression and Decision Tree classifier was able to predict outcome most accurately.
- GitHub URL : https://github.com/maxh8086/Data-Science-Project/blob/main/SpaceX-Machine-Learning-Prediction-Part-5-v1.ipynb

Results

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

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

- Show a scatter plot of Flight Number vs. Launch Site

- Show the screenshot of the scatter plot with explanations

Payload vs. Launch Site

- Show a scatter plot
  of Payload vs. Launch Site

- Show the screenshot of the
  scatter plot with explanations

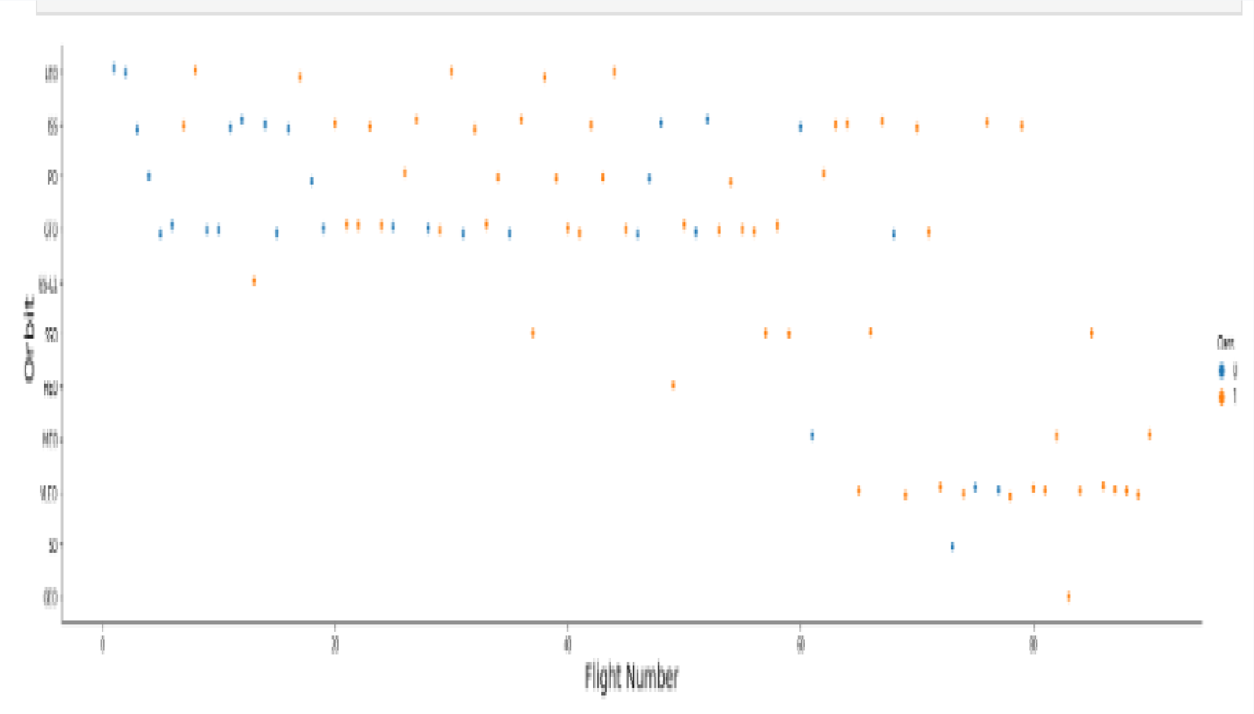
Success Rate vs. Orbit Type

- Show a bar chart for the success rate of each orbit type
- Show the screenshot of the scatter plot with explanations

Flight Number vs. Orbit Type

- scatter point of Flight number vs. Orbit type

- 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.

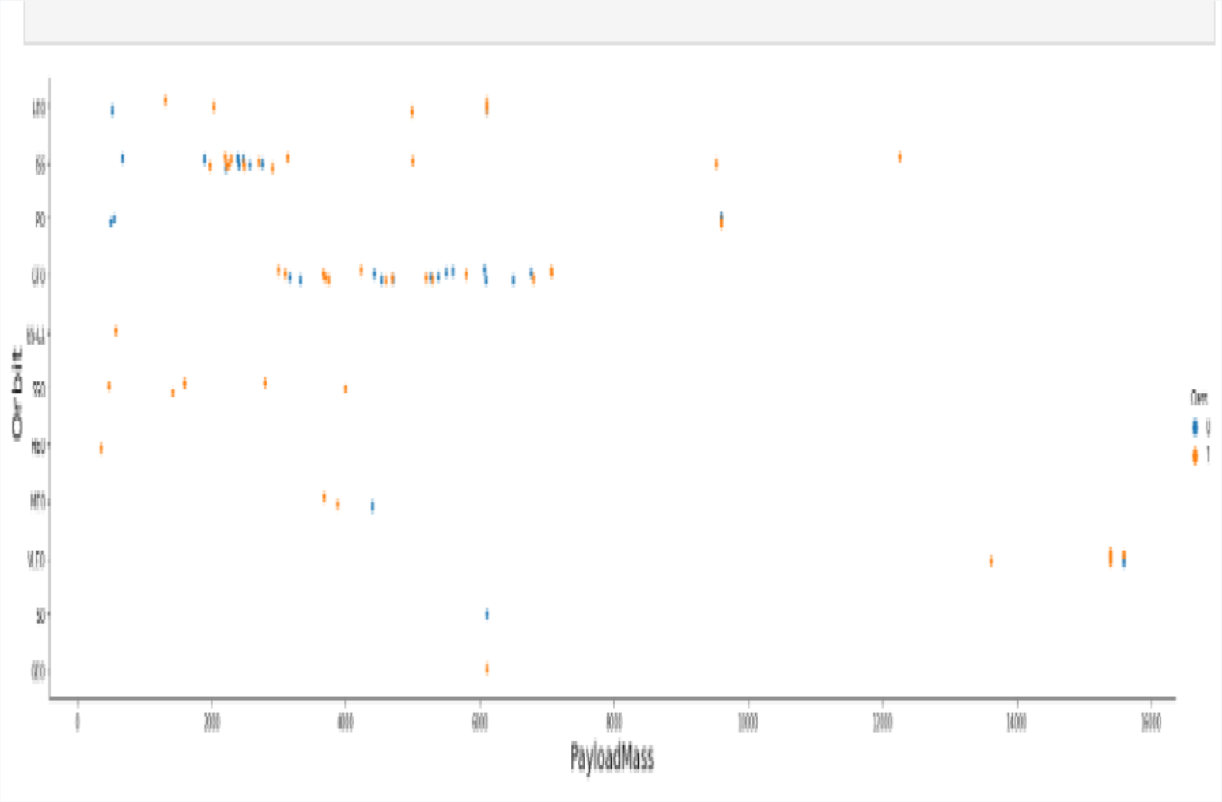


Payload vs. Orbit Type

- scatter point of payload vs. orbit type

- With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

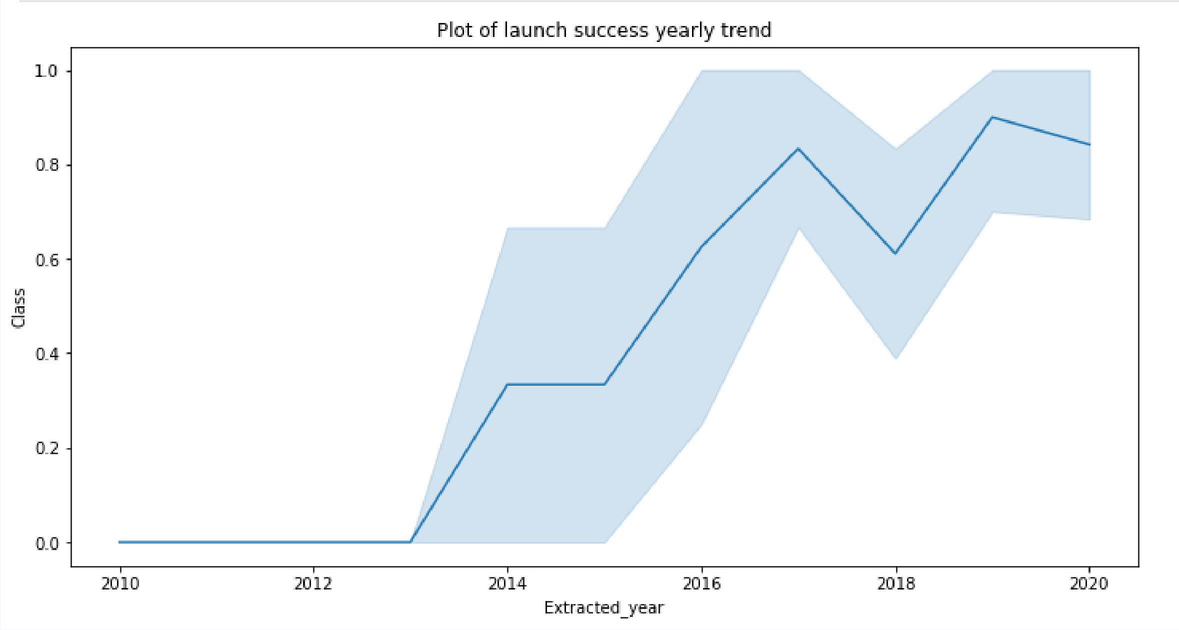
- However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.



Launch Success Yearly Trend

- Lchart of yearly average success rate

- you can observe that the success rate since 2013 kept increasing till 2020



All Launch Site Names

- Total 4 Sites were used in all Missions CCAFS LC-40, VAFB SLC-4E, KSC LC-39A, CCAFS SLC-40 where total 100 successful and 1 Failed mission was recorded.

Launch Site Names Begin with 'CCA'

- 5 records where launch sites begin with `CCA`

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

In [22]:
```
%sql SELECT Launch_Site \
     FROM SPACEXTBL \
     WHERE Launch_Site LIKE 'CCA%' \
     LIMIT 5;
```

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

Out[22]:

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Total Payload Mass

- Total 45596 payload was carried by boosters from NASA (CRS) throughout 101 missions.

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

In [24]:

```
%%sql
SELECT Customer, SUM(Payload_Mass__Kg_) AS 'Total Payload Mass (Kg)'
FROM SPACEXTBL
WHERE Customer = 'NASA (CRS)';
```

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

Out[24]:

Customer	Total Payload Mass (Kg)
NASA (CRS)	45596

Average Payload Mass by F9 v1.1

- The average payload mass carried by booster version F9 v1.1 was 2928.4

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

In [25]:

```
%%sql
SELECT Booster_Version, AVG(Payload_Mass__Kg_) AS 'AVG Payload Mass (Kg)'
FROM SPACEXTBL
WHERE Booster_Version = 'F9 v1.1';
```

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

Out[25]:

Booster_Version	AVG Payload Mass (Kg)
F9 v1.1	2928.4

First Successful Ground Landing Date

- The first successful landing outcome on ground pad was recorded on 2018-07-22.

```
In [30]: %%sql
         SELECT MIN(Date) AS 'First Successful Landing Date'
         FROM SPACEXTBL
         WHERE Landing_Outcome = 'Success';

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

Out[30]:

First Successful Landing Date
2018-07-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- List of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

Task 6

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

In [31]:

```
%%sql
SELECT Booster_Version
FROM SPACEXTBL
WHERE Landing_Outcome = 'Success (drone ship)'
  AND Payload_Mass__Kg_ BETWEEN 4000 AND 6000;
```

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

Out[31]:

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

Total Number of Successful and Failure Mission Outcomes

- Calculate the total 100 of successful and 1 failure mission was recorded.

List the total number of successful and failure mission outcomes

In [41]:
```
%%sql
SELECT
    SUM(CASE WHEN Mission_Outcome LIKE 'Success%' THEN 1 ELSE 0 END) AS Successful_Missions,
    SUM(CASE WHEN Mission_Outcome LIKE 'Failure%' THEN 1 ELSE 0 END) AS Failed_Missions
FROM SPACEXTBL;
```

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

Out[41]:

Successful_Missions	Failed_Missions
100	1

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass

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

In [43]:
```
%%sql
SELECT Booster_Version
FROM SPACEXTBL
WHERE Payload_Mass__Kg_ = (
    SELECT MAX(Payload_Mass__Kg_)
    FROM SPACEXTBL
);
```

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

Out[43]:

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

2015 Launch Records

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

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.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

In [44]:

```
%%sql
SELECT
    CASE substr(Date, 6, 2)
        WHEN '01' THEN 'January'
        WHEN '02' THEN 'February'
        WHEN '03' THEN 'March'
        WHEN '04' THEN 'April'
        WHEN '05' THEN 'May'
        WHEN '06' THEN 'June'
        WHEN '07' THEN 'July'
        WHEN '08' THEN 'August'
        WHEN '09' THEN 'September'
        WHEN '10' THEN 'October'
        WHEN '11' THEN 'November'
        WHEN '12' THEN 'December'
    END AS Month,
    Booster_Version,
    Launch_Site,
    Landing_Outcome
FROM SPACEXTBL
WHERE substr(Date, 0, 5) = '2015'
    AND Landing_Outcome LIKE '%Failure (drone ship)%';
```

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

Out[44]:

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

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

- 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

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.

In [45]:

```
%%sql
SELECT Landing_Outcome, COUNT(*) AS Outcome_Count
FROM SPACEXTBL
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 Outcome_Count DESC;
```

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

Out[45]:

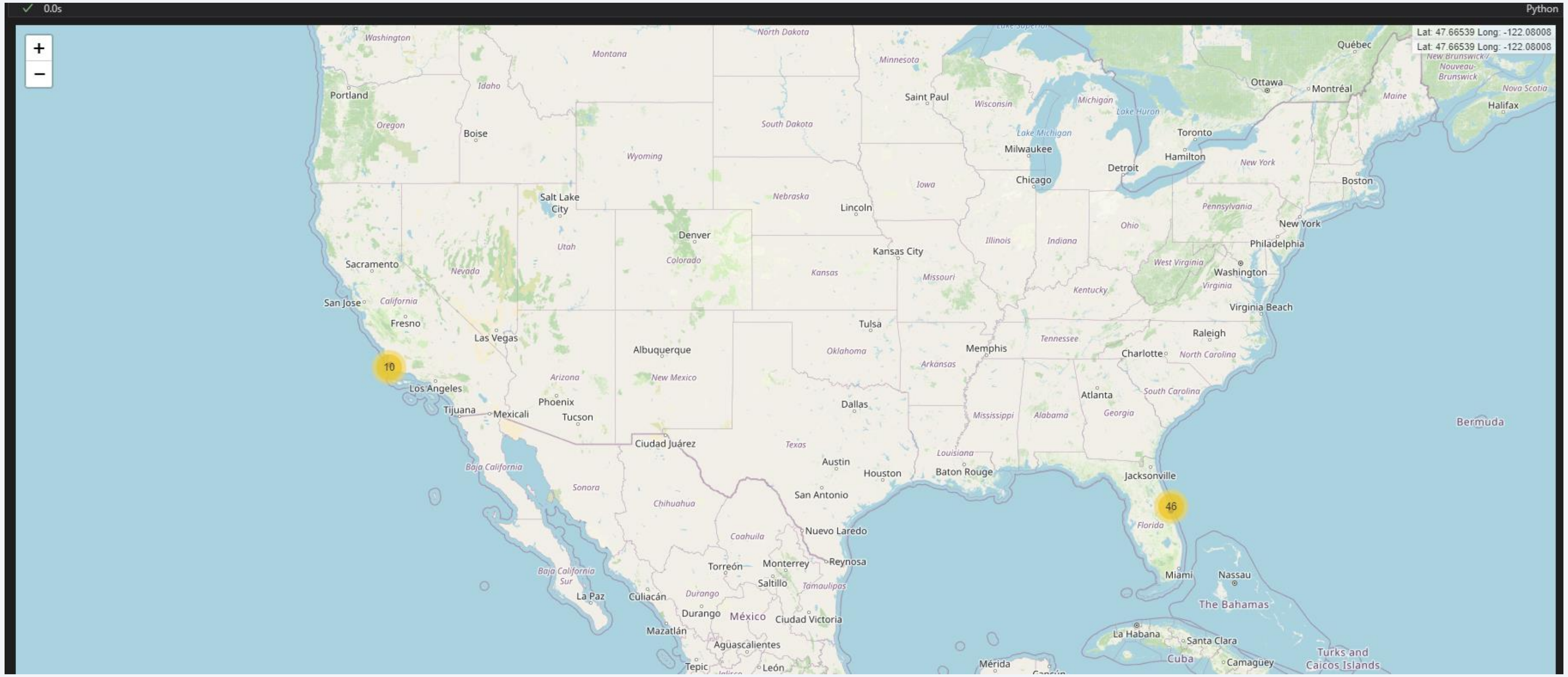
Landing_Outcome	Outcome_Count
Failure (drone ship)	5
Success (ground pad)	3

Section 3

Launch Sites Proximities Analysis

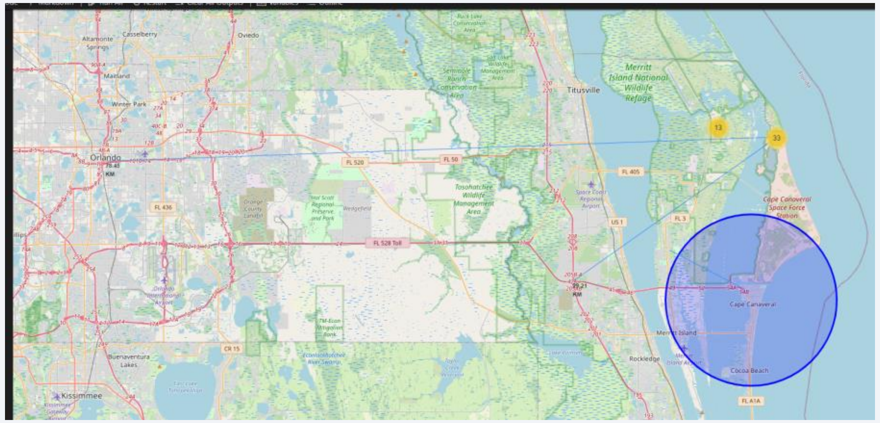
All launch sites global map markers

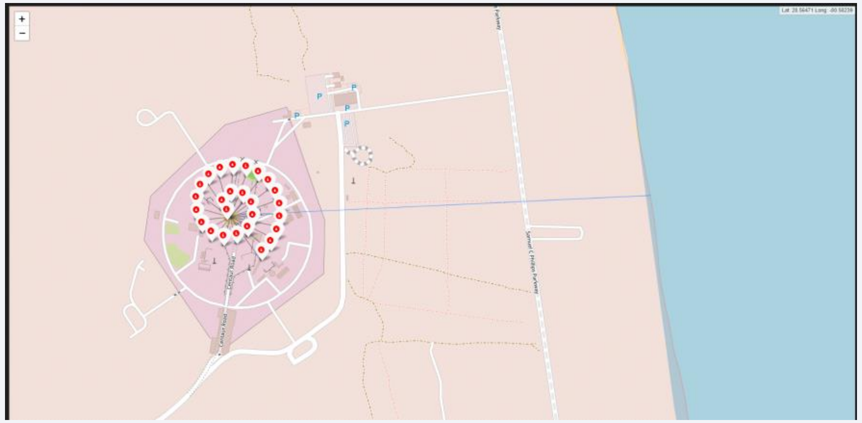
- Total 46 Launch operations was performed from Cape Canaveral Space Station and 10 Launch operations was performed from Vandenberg Space Station



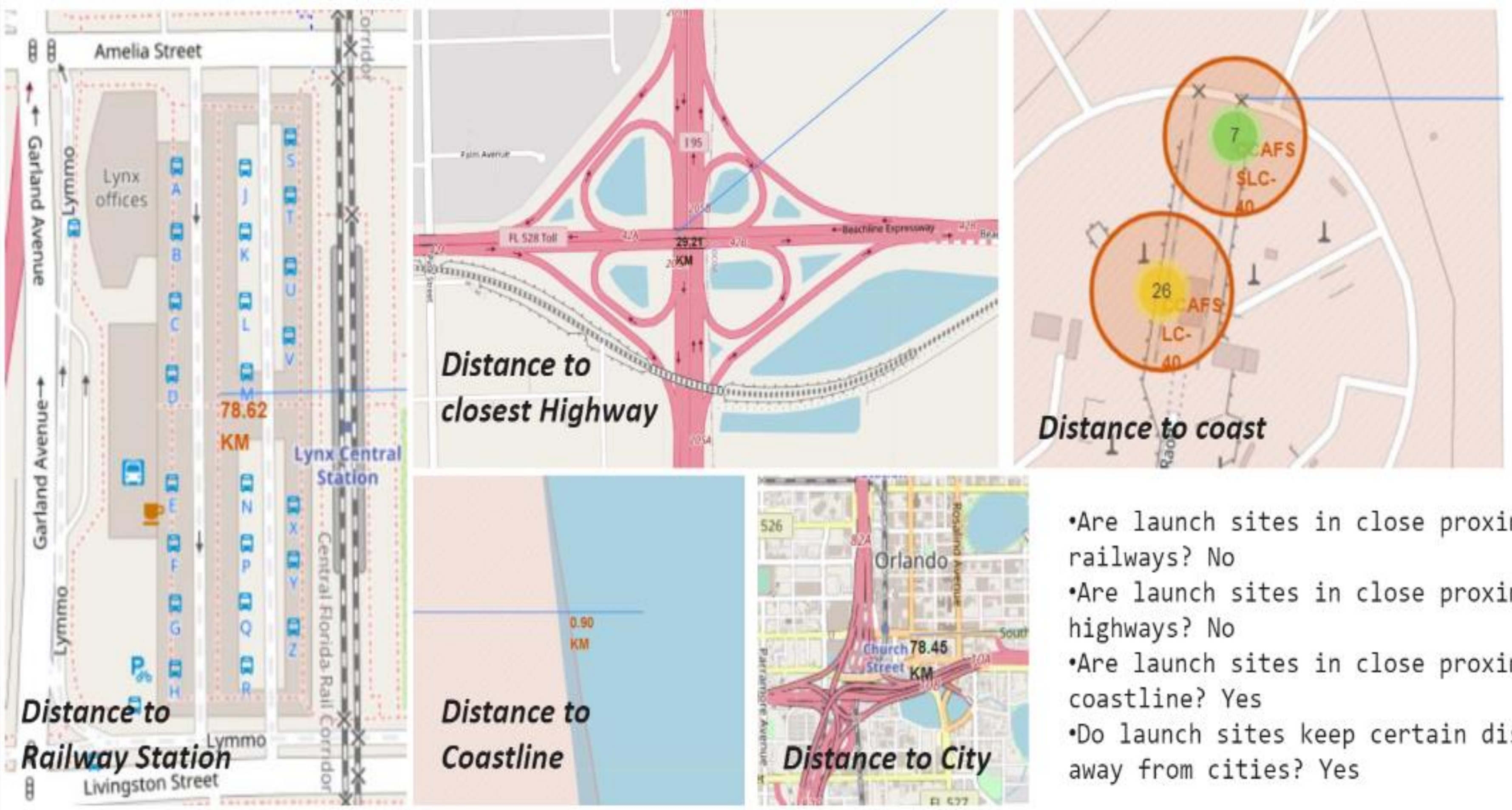
Cape Canaveral space station

- Falcon 9 Launchpad





Launch Site distance to landmarks



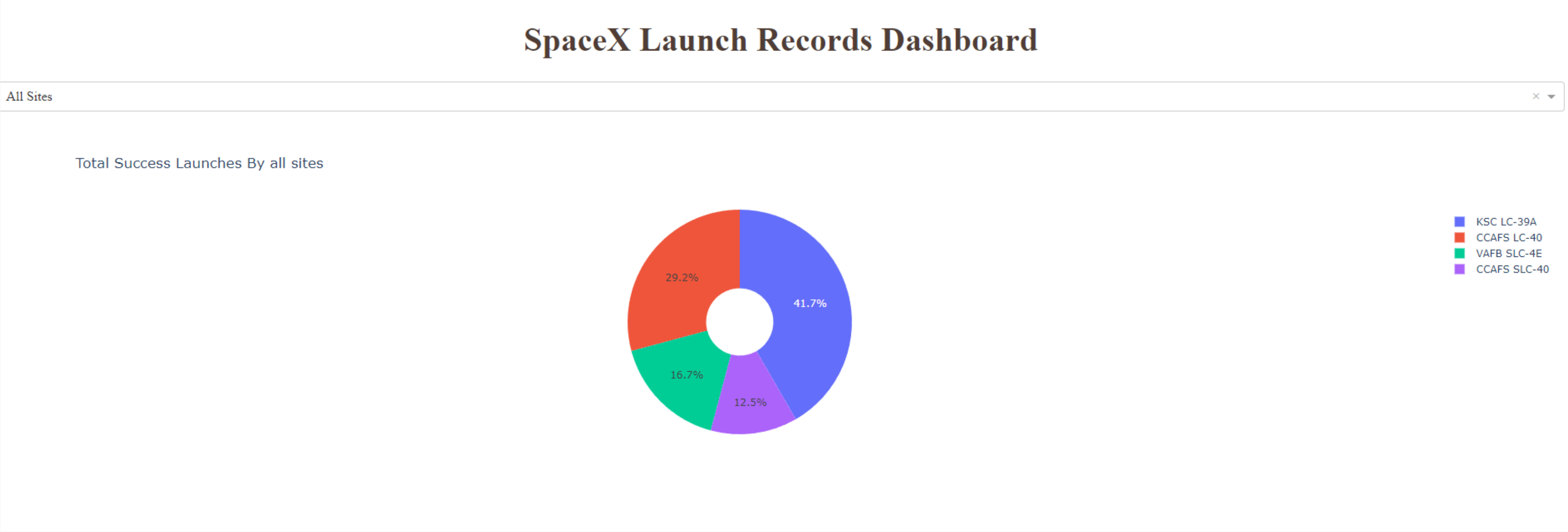
•Are launch sites in close proximity to railways? No
•Are launch sites in close proximity to highways? No
•Are launch sites in close proximity to coastline? Yes
•Do launch sites keep certain distance away from cities? Yes

Section 4

Build a Dashboard with Plotly Dash

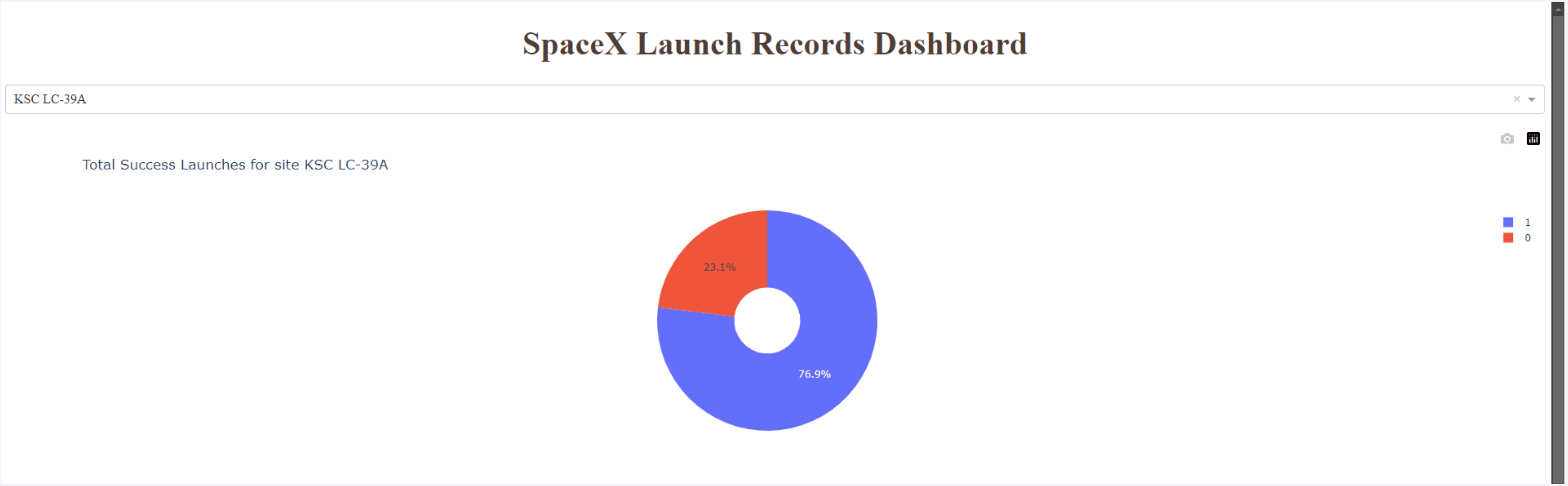
Dashboard Application with Plotly Dash

Out of 4 Sites, KSC LC 39A has most successful Launch



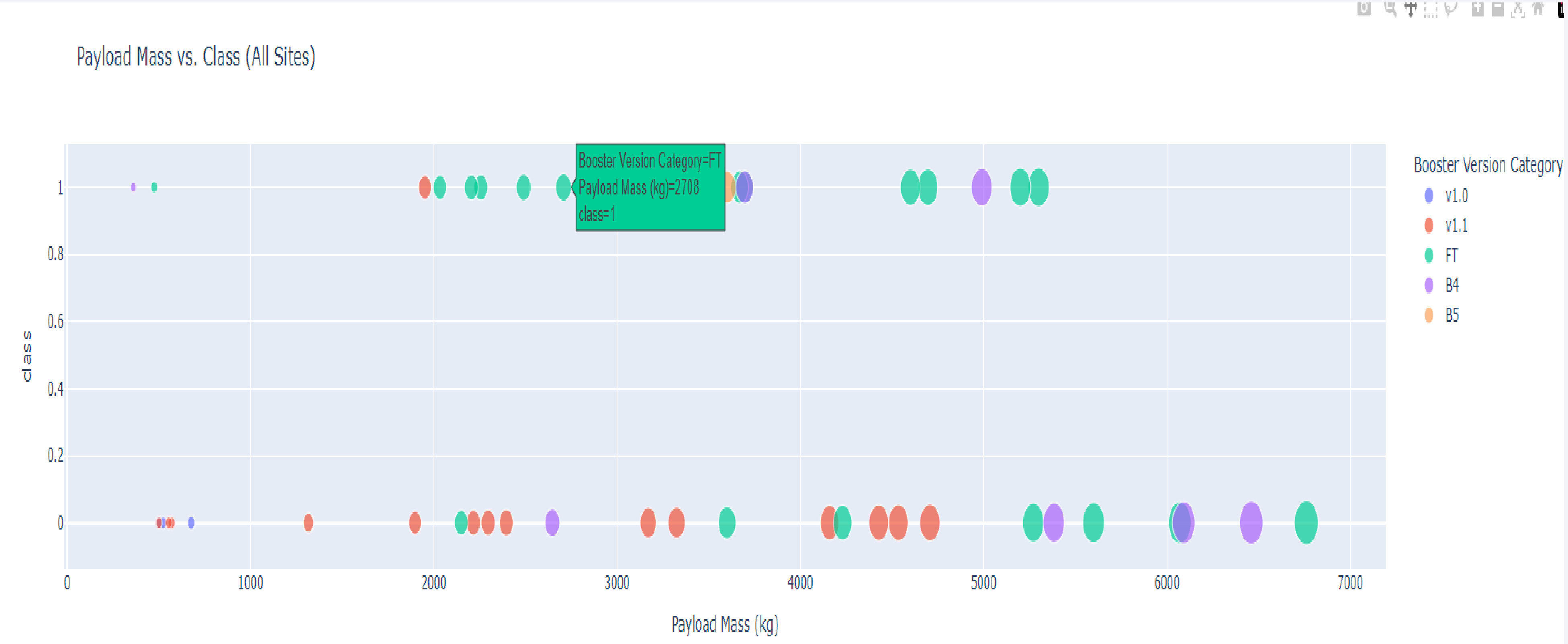
Launch site with the highest Success ratio

- KSC LS 39A has 76.9% Launch success ratio



Payload vs. Launch Outcome

- FT Booster version have the largest success rate across range.

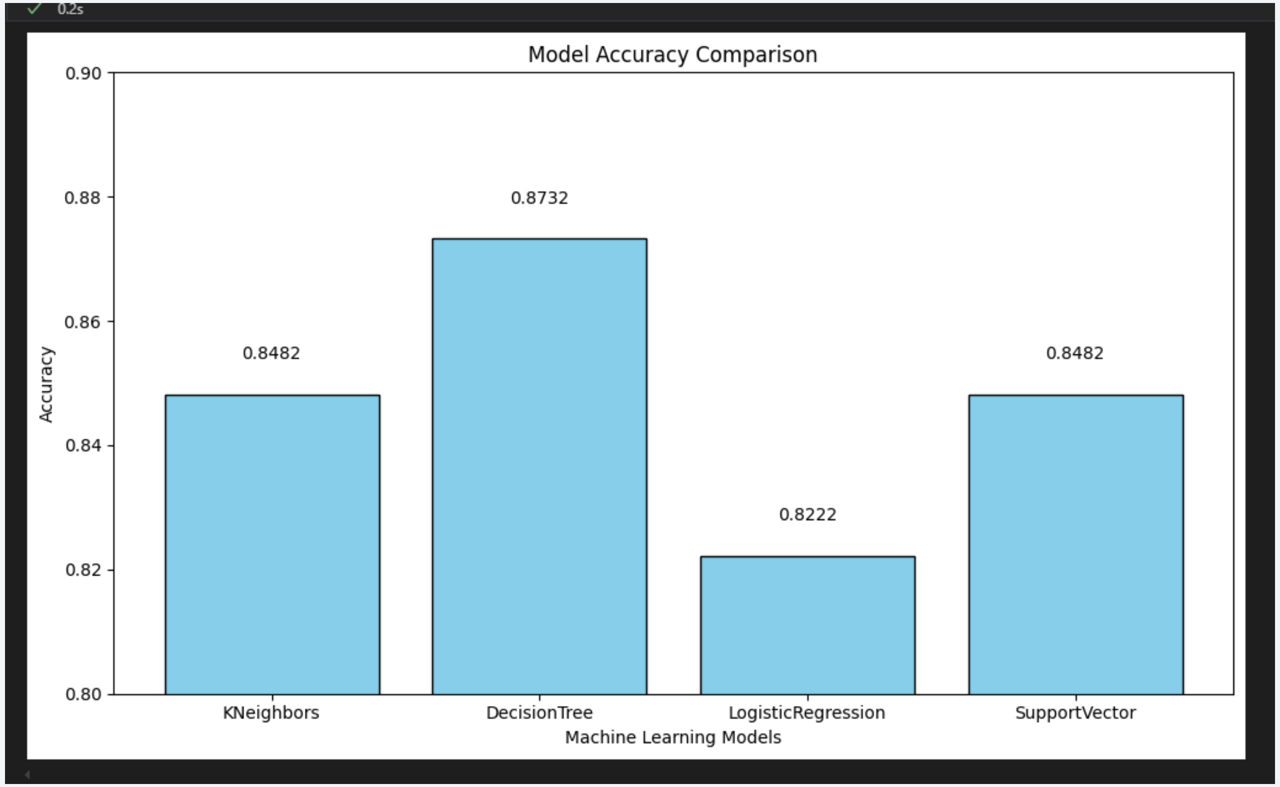


Section 5

Predictive Analysis (Classification)

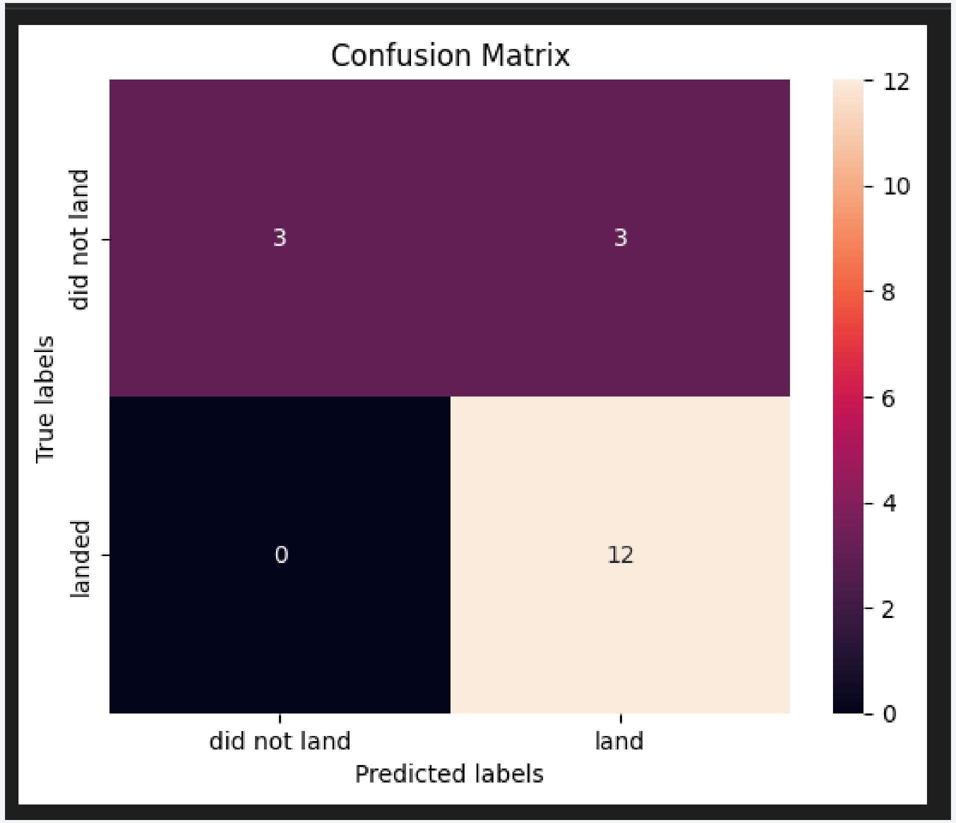
Classification Accuracy

- DecisionTree model has the highest classification accuracy with a score of 0.8732



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

We can conclude that:

- 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!