Winning Space Race with Data Science

Lau Ying Yi Aaron
24 April 2023

IBM Developer
SKILLS NETWORK

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion

Executive Summary

- Space X has revolutionized the space industry by providing rocket launches at a fraction of the cost of its competitors. This is possible due to its ability to reuse the first stage of its Falcon 9 rocket, saving millions of dollars per launch. To determine the cost of a launch, it is essential to predict if the first stage will land. This information can be used to bid against Space X for a rocket launch.

- In this project, we have created a machine learning pipeline using data extraction, exploratory data analysis, plots, machine learning, and Folium maps. The pipeline predicts if the first stage of a Falcon 9 rocket will land, given the data from preceding labs. The pipeline's accuracy is essential for determining the cost of a rocket launch and can help an alternate company bid against Space X.

- Overall, this project's goal is to leverage machine learning to enable fair competition in the rocket launch industry by accurately predicting whether the first stage of a rocket will land.

Introduction

- How Space X disrupted the rocket industry?
  - Reuse the first stage of their Falcon 9 rockets
  - Vertical Landing
  - In-house manufacturing
  - Advanced technology

- For companies looking to bid against Space X for a rocket launch, it is essential to accurately predict if the first stage of a Falcon 9 rocket will land. This information can help companies determine the cost of a launch and make informed decisions about bidding.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected through request to the SpaceX API
- Perform data wrangling
  - Cleaning is done for rows with missing values
- Perform exploratory data analysis (EDA) using visualization
- Perform interactive visual analytics using Folium and Plotly Dashboard
- Perform predictive analysis using classification models
  - Finally, the building, tuning, and evaluation of classification models

Data Collection

- Data sets were collected through Space X API
- Requests allows me to make HTTP requests which I used to get data from the API
- Below is an example of how I obtained information on the outcome of the landing as well as other details such as the number of flights based on certain cores:

```
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
                Block.append(response['block'])
                ReusedCount.append(response['reuse_count'])
                Serial.append(response['serial'])
            else:
                Block.append(None)
                ReusedCount.append(None)
                Serial.append(None)
            Outcome.append(str(core['landing_success'])+' '+str(core['landing_type']))
            Flights.append(core['flight'])
            GridFins.append(core['gridfins'])
            Reused.append(core['reused'])
            Legs.append(core['legs'])
            LandingPad.append(core['landpad'])
```

Data Collection – SpaceX API

- Firstly, create multiple functions such as the one shown previously
- Essentially each function/ API call will give us information such as the Booster name/ name and location of launch site used/ landing outcomes/ type of landing/ etc.

- Link below for more info/ reference:

https://github.com/aaron-lyy/IBM_capstone_datascience/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

```python
# Requests allows us to make HTTP requests which we will use to get data from an API
import requests
# Pandas is a software library written for the Python programming language for data manipulation and analysis
import pandas as pd
# NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays
import numpy as np
# Datetime is a library that allows us to represent dates
import datetime
```

```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'])
```

Data Wrangling

- Exploratory Data Analysis (EDA) was conducted and converted the success outcomes into 1 (Successfully Landed) and 0 (Unsuccessfully Landed)

- The analysis also includes looking into the number of launches for each site, the number and occurrence of each orbit and many more.

- Link below for more info/ reference:

https://github.com/aaron-lyy/IBM_capstone_datascience/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- Visualizations of various relationships between variables in the dataset were explored
- It provided some preliminary insights about how certain variables would affect the success rate
- Feature engineering was carried out such as converting important categorical variables into dummy variables
- Link below for more info/ reference:

https://github.com/aaron-lyy/IBM_capstone_datascience/blob/main/jupyter-labs-eda-dataviz.ipynb

EDA with SQL

- Queries done included:
  - The 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 date when the first successful landing outcome in ground pad was achieved
  - The total number of successful and failure mission outcomes
  - The number of successful landing outcomes between the date 04-06-2010 and 20-03-2017
  - Etc.
- Link below for more info/ reference:

https://github.com/aaron-lyy/IBM_capstone_datascience/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Multiple map objects such as markers, circles, lines, etc. were created and added to a folium map
- This objects provide better visualizations on the Launch Sites and the proximities around it.
- Using colored markers, it provides easier identification of the launch sites with relatively higher success rate
- Link below for more info/ reference:

https://github.com/aaron-lyy/IBM_capstone_datascience/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- A pie chart which shows the proportion of success among launch sites as well as the success rate of each launch site
- A scatter plot showing the correlation between success rate and payload mass is also added to the dashboard.
- These plots provide useful insights for us to understand which launch sites are doing well and whether the payload mass affects our success rate
- For more info/ reference of the dashboard, please refer to: https://github.com/aaron-lyy/IBM_capstone_datascience/blob/main/spacex_dash_app.py
- Dataset: https://github.com/aaron-lyy/IBM_capstone_datascience/blob/main/spacex_launch_dash.csv

Predictive Analysis (Classification)

- The model was built, evaluated, improved, and found the best using Grid Search CV as well as each model's respective parameters
- Using accuracy as the performance metric, we got the best classification model by comparing the scores.
- Link below for more info/ reference:

https://github.com/aaron-lyy/IBM_capstone_datascience/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

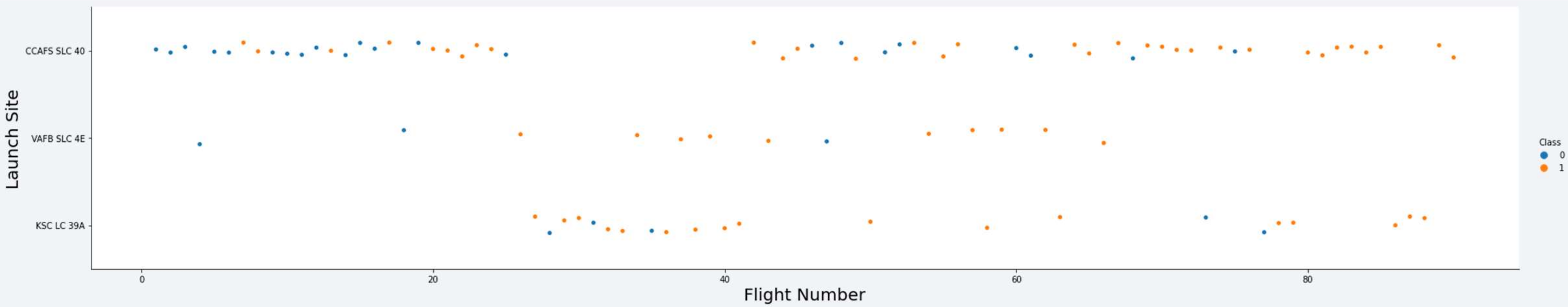
Results

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

Section 2

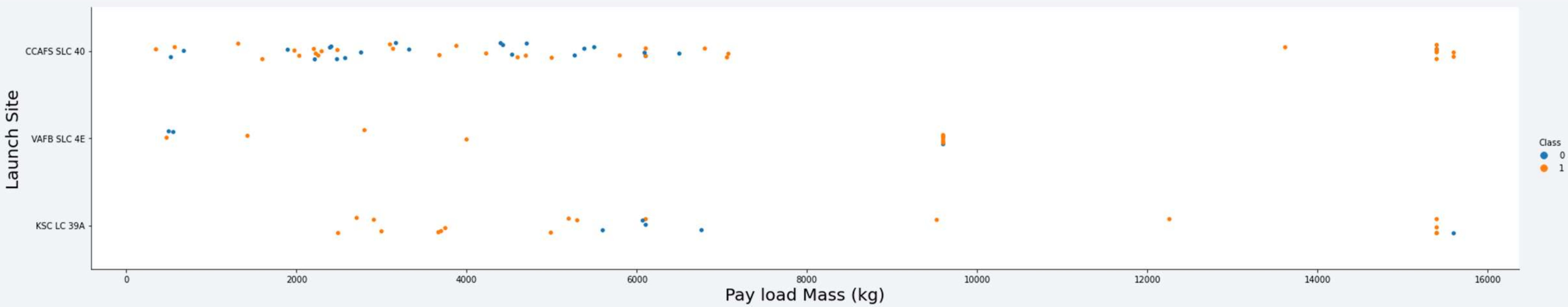
Insights drawn from EDA

Flight Number vs. Launch Site



- The above plot suggests that as the number of flights increase, the greater the success rate at each launch site.

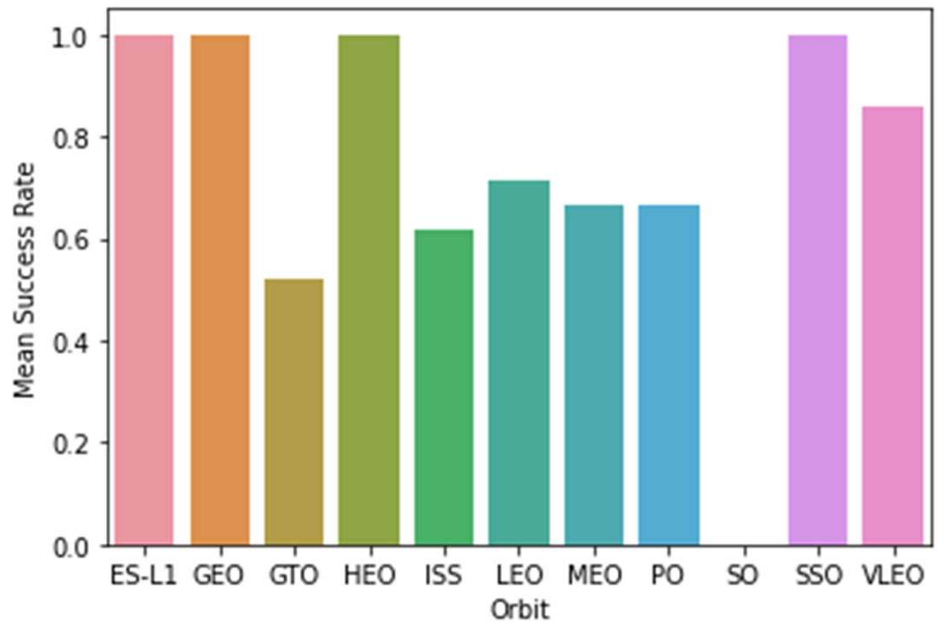
Payload vs. Launch Site



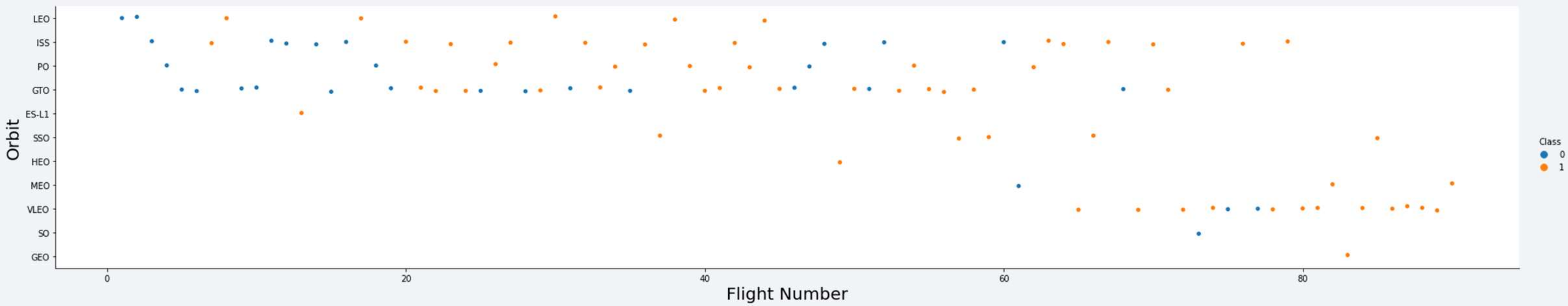
- The plot above shows that for the VAFB-SLC launch site there are no rockets launched for heavy payload mass(greater than 10000)

Success Rate vs. Orbit Type

- Here we can see that ES-L1, GEO, HEO, and SSO has the greatest success Rate

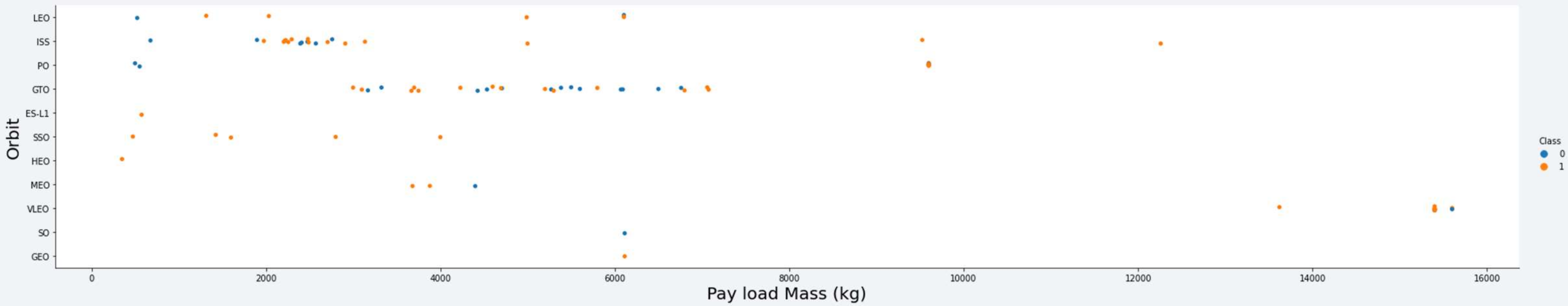


Flight Number vs. Orbit Type



- We can see that in the 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.

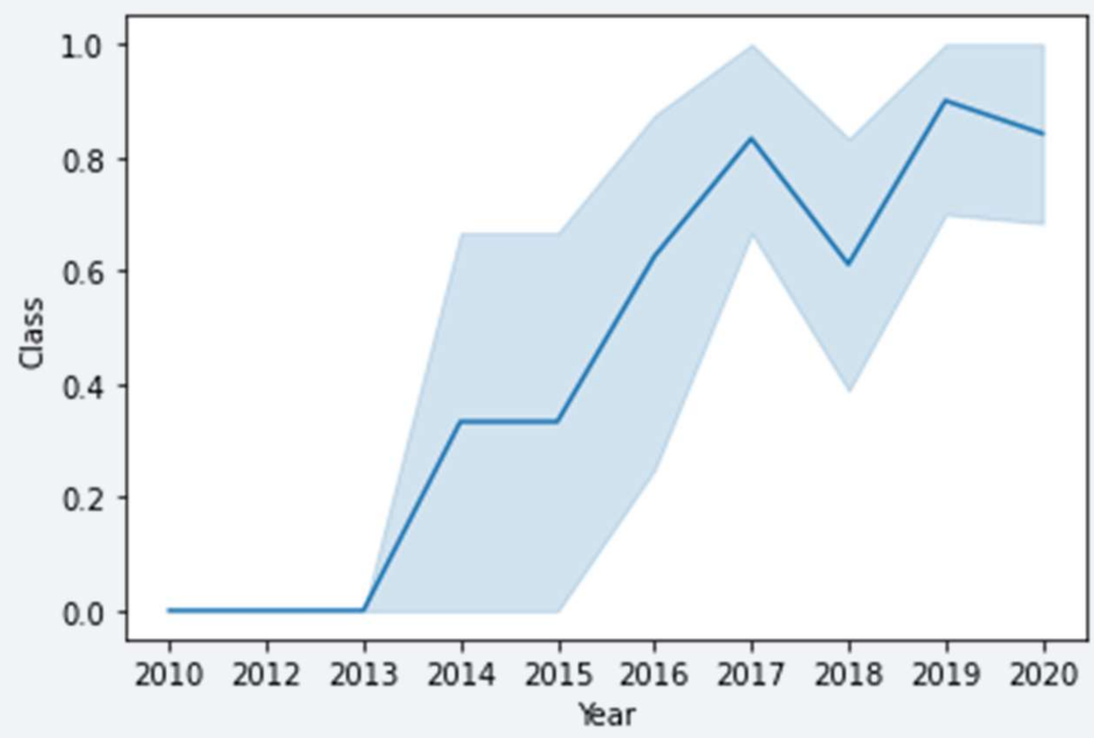
Payload vs. Orbit Type



- With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.
- However, for GTO, both positive landing rate and negative landing(unsuccessful mission) cannot be distinguished well as they are both there here.

Launch Success Yearly Trend

- Success rate since 2013 continue to increase till 2020



All Launch Site Names

- DISTINCT was used in the query to get the unique launch site names
- Here a total of 4 Launch site names were queried

```
query='''
SELECT DISTINCT Launch_Site
FROM SPACEXTBL
'''

df = pd.read_sql_query(query, con)
df
```

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

Launch Site Names Begin with 'CCA'

- Query Result below shows 5 records with launch site names beginning with ‘CCA’

```
query='''
SELECT *
FROM SPACEXTBL
WHERE Launch_Site like 'CCA%'
LIMIT 5
'''

df = pd.read_sql_query(query, con)
df
```

	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
0	04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	08-12-2010	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	22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
3	08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	01-03-2013	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 carried by boosters from NASA (including collabs) is 48,213 kg
- SUM() function is used to get the total payload mass
- LIKE is used to get all possible boosters from 'NASA (CRS)'. Note that this includes any collaborations done with other customers.

```
query='''
SELECT SUM(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE CUSTOMER like '%NASA (CRS)%'

'''

df = pd.read_sql_query(query, con)
df
```

	SUM(PAYLOAD_MASS__KG_)
0	48213

Average Payload Mass by F9 v1.1

- Average payload mass carried by booster version F9 v1.1 is about 2,535 kg
- AVG() function is used to get the average payload mass
- LIKE is used to get all possible booster versions from ‘F9 v1.1’

```
query='''
SELECT AVG(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE Booster_Version like '%F9 v1.1%'

'''

df = pd.read_sql_query(query, con)
df
```

	AVG(PAYLOAD_MASS__KG_)
0	2534.666667

First Successful Ground Landing Date

```
query='''
SELECT min(new_date) as First_Date
FROM (SELECT (substr(Date,7,4) || "-" ||substr(Date,4,2)||"-"||substr(Date,1,2)) as new_date, *
FROM SPACEXTBL)
WHERE "Landing _Outcome" like 'Success (ground pad)'

'''

df = pd.read_sql_query(query, con)
df
```

	First_Date
0	2015-12-22

- First Landing Date is on 22nd December 2015
- Some data manipulation was required for the date column to get it in the right format. (See the sub query)
- The min() function was used to get the earliest date

Successful Drone Ship Landing with Payload between 4000 and 6000

```
query='''
SELECT Booster_Version
FROM SPACEXTBL
WHERE "Landing _Outcome" = 'Success (drone ship)' AND PAYLOAD_MASS__KG_>4000 AND PAYLOAD_MASS__KG_<6000
'''

df = pd.read_sql_query(query, con)
df
```

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

- As shown above, the required conditions are placed in the query to get the list of Booster versions above

Total Number of Successful and Failure Mission Outcomes

- “CASE WHEN” is used to add in the condition for success / Failure
- “GROUP BY” is then used to group based on the created condition
- As shown, the number of successful mission outcomes is 100. The number of failed mission outcome is 1.

```
query='''
SELECT CASE WHEN Mission_Outcome like '%Failure%' THEN "Failure"
            WHEN  Mission_Outcome like '%Success%' THEN "Success"
            ELSE "Others"
        END AS mission_outcomes,
COUNT(*)
FROM SPACEXTBL
GROUP BY(mission_outcomes)
'''

df = pd.read_sql_query(query, con)
df
```

	mission_outcomes	COUNT(*)
0	Failure	1
1	Success	100

Boosters Carried Maximum Payload

```
query='''
SELECT Booster_Version, PAYLOAD_MASS__KG_ as payload_mass_kg
FROM SPACEXTBL
WHERE payload_mass_kg = (
                        SELECT MAX(PAYLOAD_MASS__KG_)
                        FROM SPACEXTBL
                        )
ORDER BY Booster_Version
'''

df = pd.read_sql_query(query, con)
df
```

	Booster_Version	payload_mass_kg
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

- Subquery was used to get the maximum payload mass
- This will allow us to get the list of boosters carried by the maximum payload mass

2015 Launch Records

```
query='''
SELECT substr(Date,7,4) As Year, substr(Date,4,2) As Month, Date, "Landing _Outcome", Booster_Version, Launch_Site
FROM SPACEXTBL
WHERE "Landing _Outcome" like '%Failure%' AND Year = '2015'
'''

df = pd.read_sql_query(query, con)
df
```

	Year	Month	Date	Landing _Outcome	Booster_Version	Launch_Site
0	2015	01	10-01-2015	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
1	2015	04	14-04-2015	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

- As mentioned previously, the date format is not in the standard format.
- Substr() is used to get the Year which allows us to get the 2015 Launch Records

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

```
query='''
SELECT "Landing _Outcome",COUNT(*)
FROM (SELECT (substr(Date,7,4) || "-" ||substr(Date,4,2)||"-"||substr(Date,1,2)) as new_date, *
FROM SPACEXTBL WHERE (new_date BETWEEN "2010-06-04" AND "2017-03-20"))
WHERE "Landing _Outcome" like '%Success%'
GROUP BY "Landing _Outcome"
ORDER BY COUNT(*) DESC
'''

df = pd.read_sql_query(query, con)
df
```

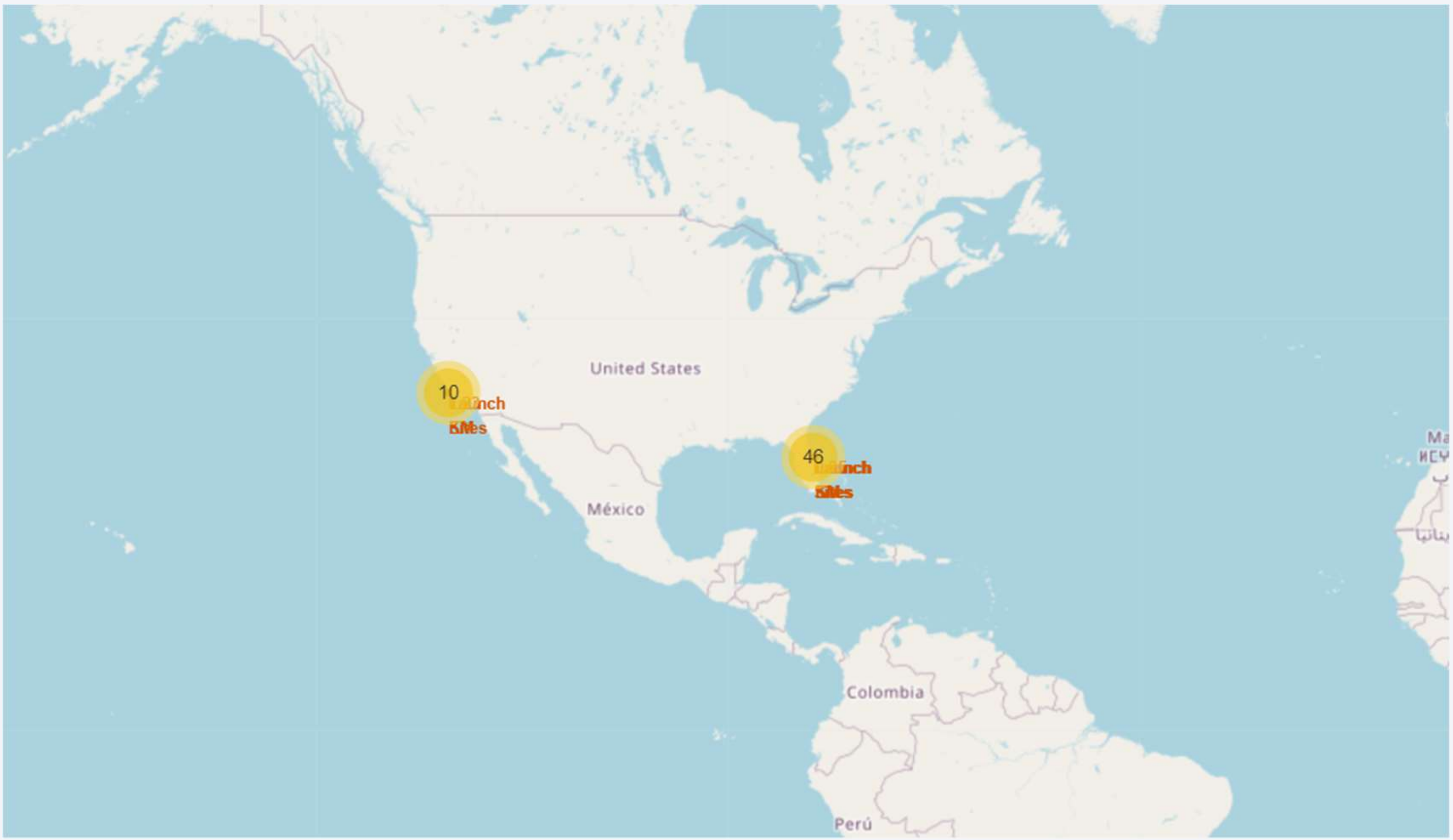
	Landing _Outcome	COUNT(*)
0	Success (drone ship)	5
1	Success (ground pad)	3

- Used sub query to get the correct date format and date range.
- Then get the successful landing outcomes and do a group by
- Above shows the successful landing outcomes from 2010-06-04 to 2017-03-20

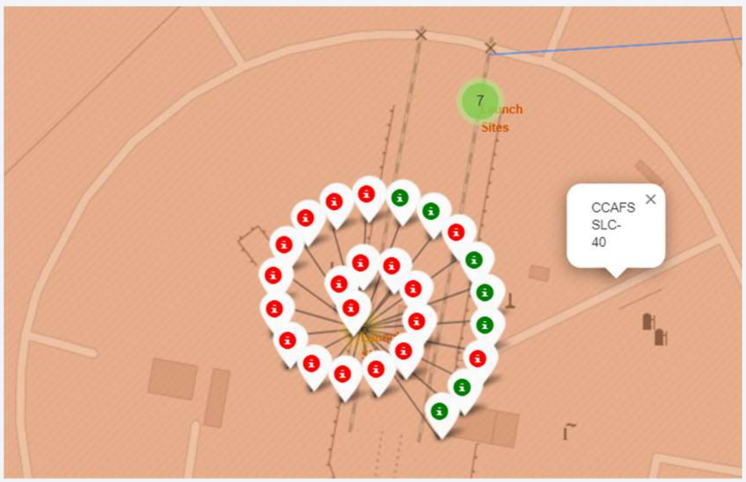
Section 3

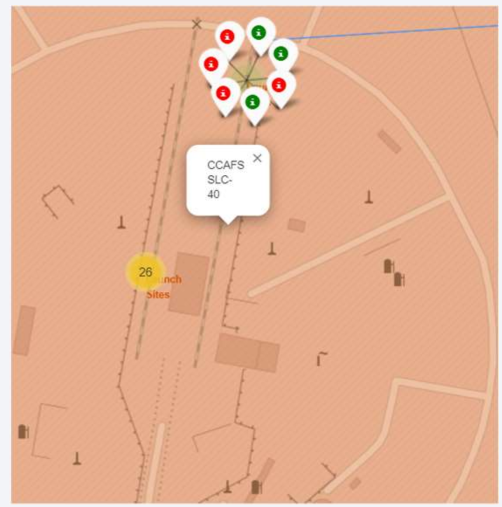
Launch Sites Proximities Analysis

General Overview of Launch Sites in Follium Map



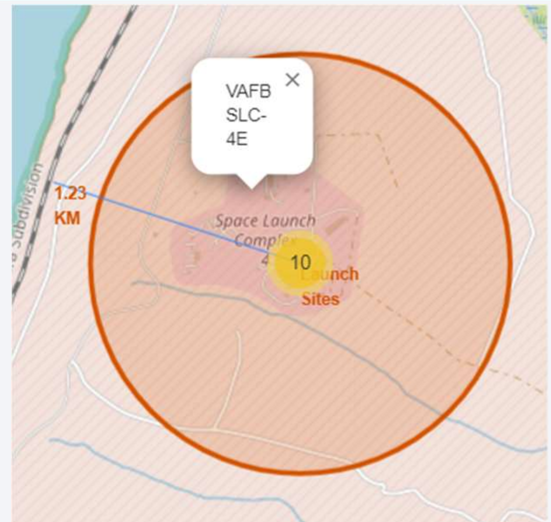
Coloured Markers showing Launch Site and Labels





- Green Markers above shows success while Red Markers show failure

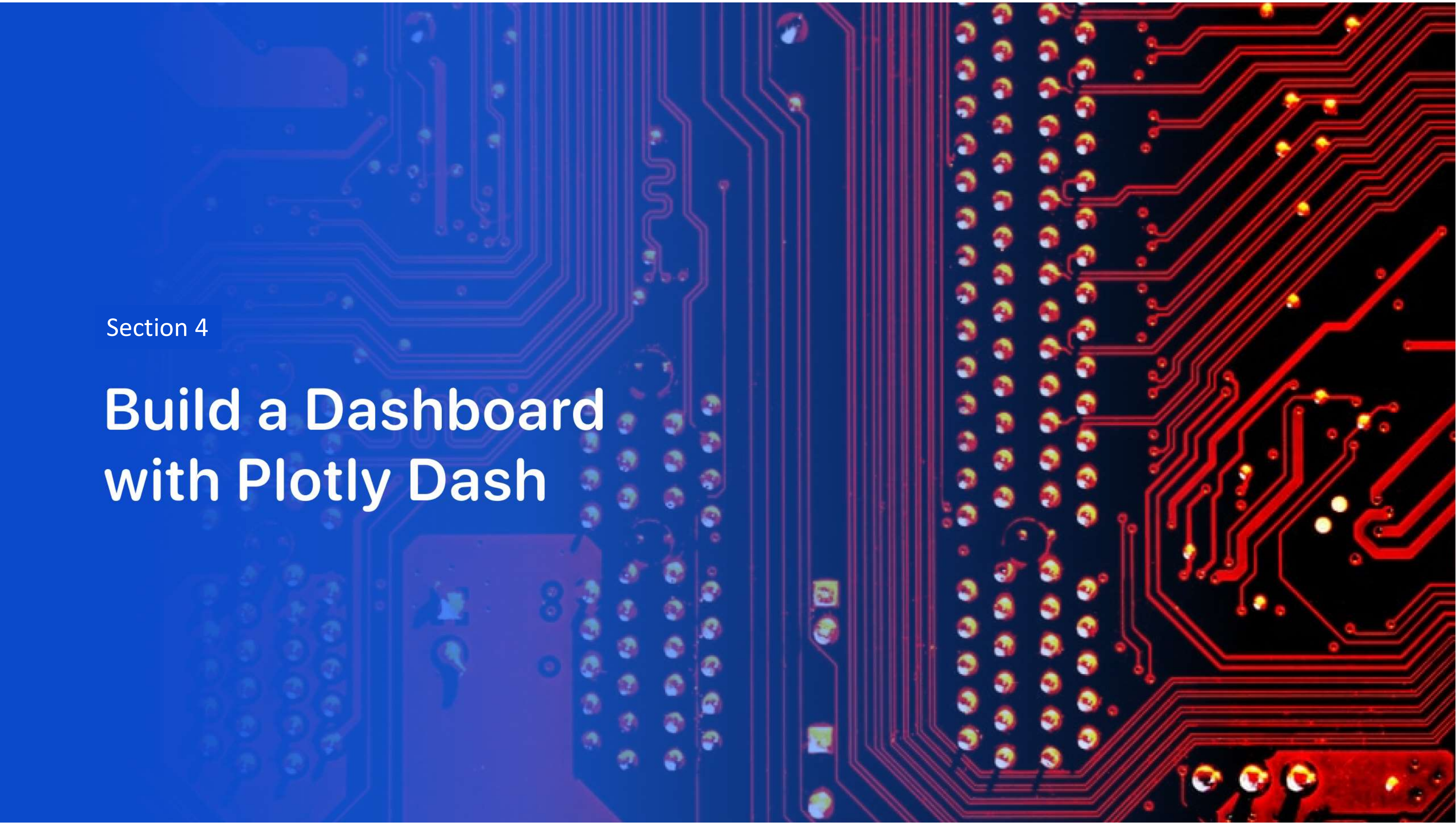
Proximities to Railway and Coastline



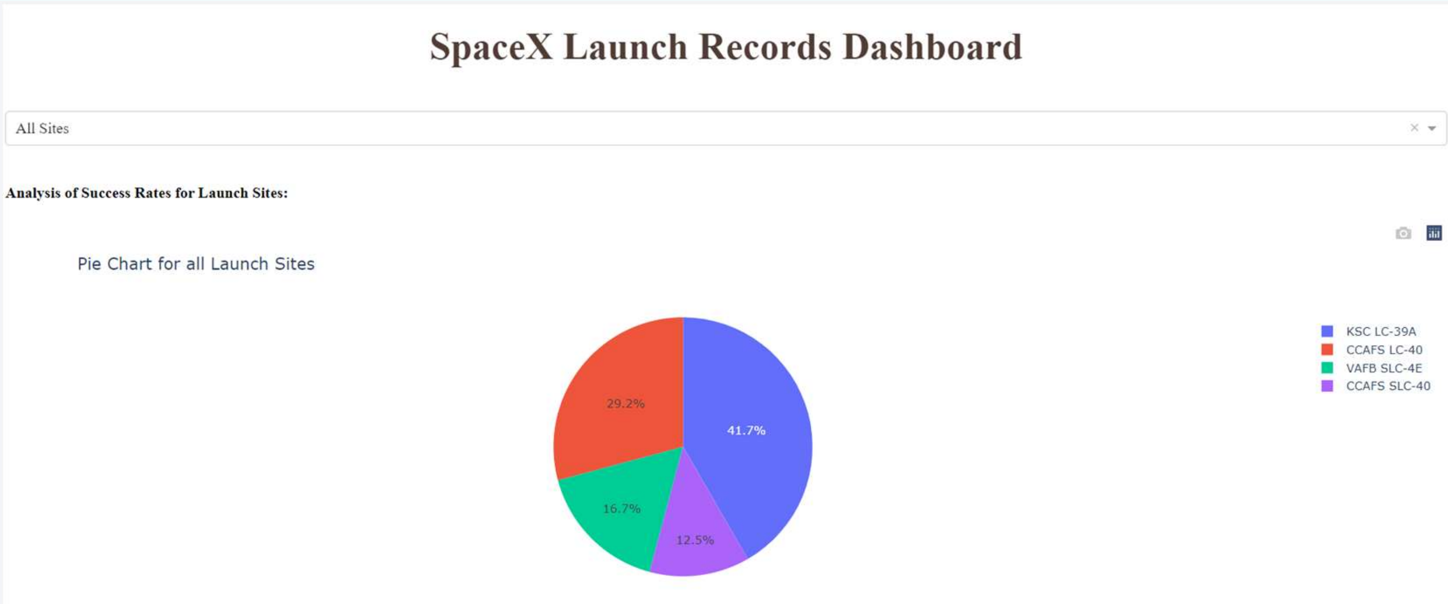
- Above shows the selected launch site to its proximities such as railway, or coastline, with distance calculated and displayed
- As shown, a line is drawn from the launch site to the respective proximities

Section 4

Build a Dashboard with Plotly Dash

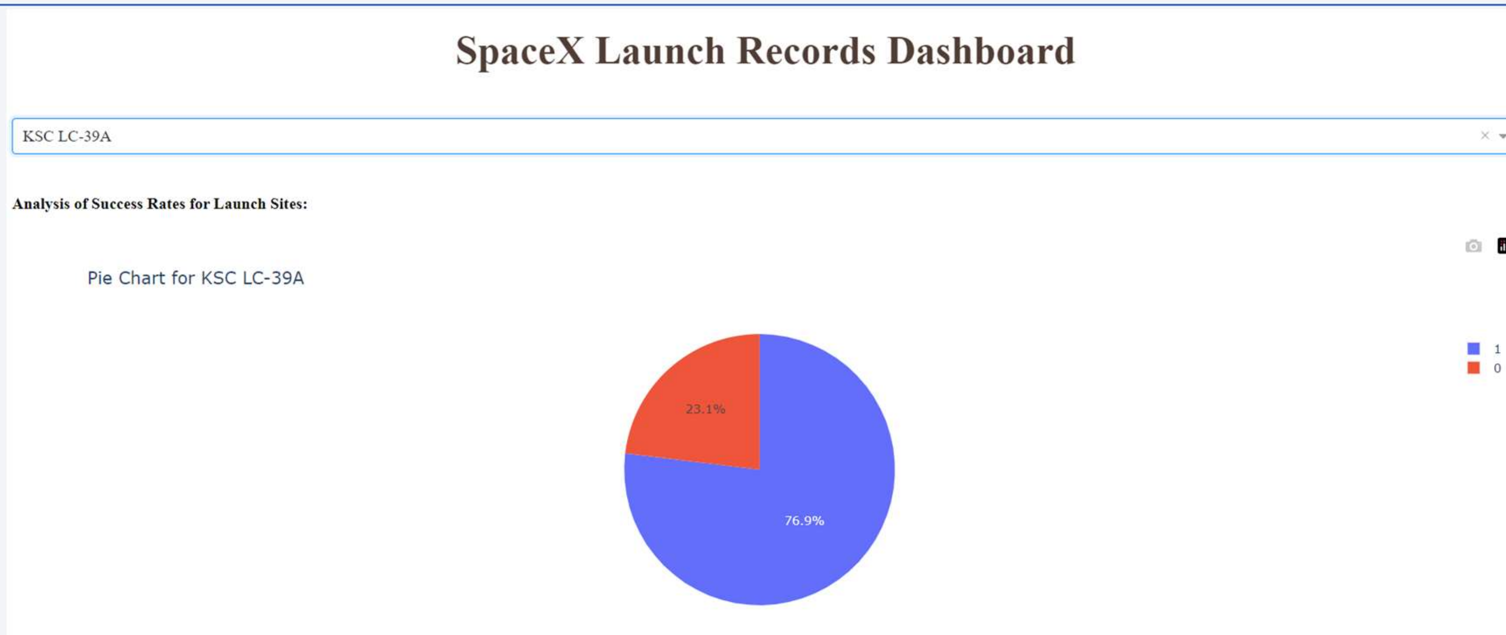


Success ratio among the Launch sites



- The pie chart above shows the percentage of Success among the launch sites
- KSC LC-39A shows the highest success rate.

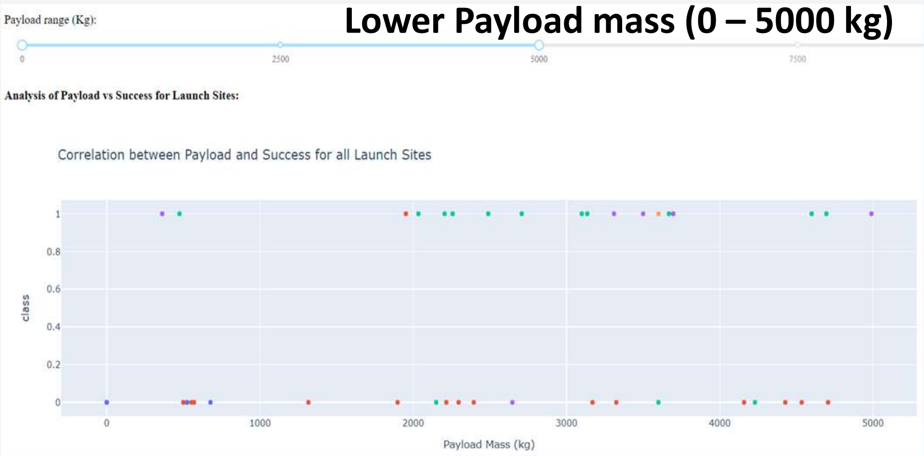
Success Rate for KSC LC-39A



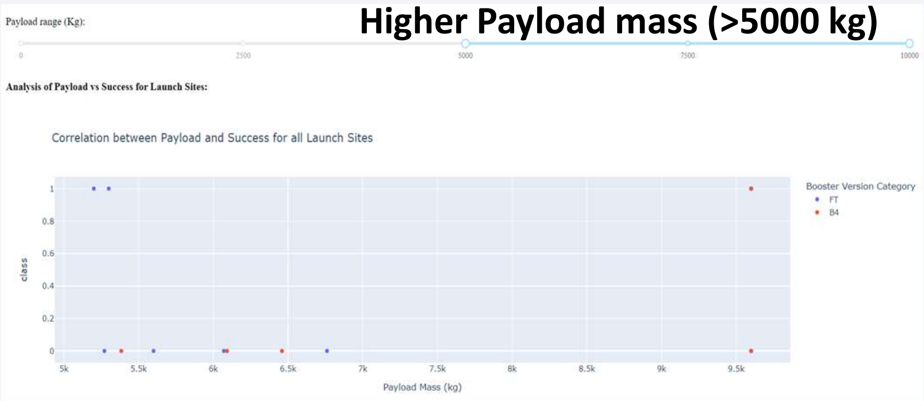
- The dashboard can also show us a pie chart of the success rate for a particular launch site
- Here, we can see that the success rate for KSC LC-39A is about 77%

Correlation between Payload and Launch Outcome

Lower Payload mass (0 – 5000 kg)



Higher Payload mass (>5000 kg)



- The dashboard has a range slider which allows users to select the range of interest for the payload mass

- We can see that in general, lower payload mass gives a higher success rate

Section 5

Predictive Analysis (Classification)

Classification Accuracy

Find the method performs best:

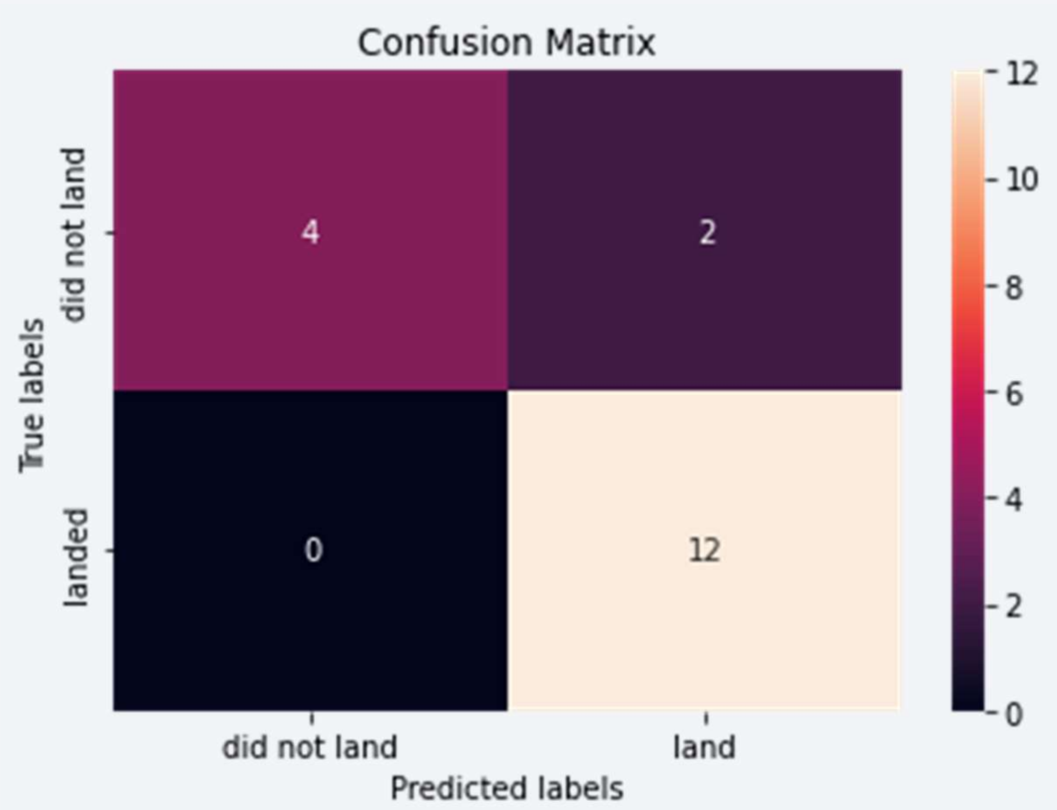
```
model = {'K Nearest Neighbors':knn_cv.best_score_,'Decision Tree':tree_cv.best_score_ ,'Support Vector Machine':svm_cv.best_score_,'Logistic Regressio

best_method = max(model, key = model.get)

print(f"The best method is {best_method}: {model[best_method]}")
```

```
The best method is Decision Tree: 0.9035714285714287
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 4, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2,
'splitter': 'best'}
```

- Out of all the built classification models, Decision Tree provided the highest classification accuracy

Confusion Matrix



- On the right shows the confusion matrix for the Decision Tree Model
- The only issue shown from the matrix is the False Positives with 2 counts predicted to land but in actual fact did not land.

Conclusions

- A higher payload mass suggests a lower success rate
- A greater flight number suggests a higher success rate for each launch site
- Some orbits have very high success rates, perhaps more analysis can be done on those orbits to gain insights on success rates factors
- We can also look into KSC LC-39A which has the greatest success rate among the launch sites.
- Overall, the decision tree classifier is the best model among all the classifiers explored


Thank you!