Winning Space Race with Data Science

Roberto Fragoso

Febrero 2024

https://github.com/Explrador/Aplpied-Data-Science-CAPSTONE.git

Outline

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

Executive Summary

Summary of methodologies

- Data collection
- Data wrangling
- EDA with data visualization
- EDA with SQL
- Building an interactive map with Folium
- Building a Dashboard with Plotly Dash
- Predictive analysis (Classification

- Summary of all results
  - Exploratory data analysis results
  - Interactive analytics demo in screenshots
  - Predictive analysis results

Introduction

- **Project background and context**

The era of commercial space has arrived, and there are several companies that are making space travel affordable for everyone. Perhaps the most successful of them is *SpaceX*, and one of the reasons is that their rocket launch is relatively inexpensive.
SpaceX 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 *SpaceX* can reuse the first stage.
Therefore, we will predict if the *Falcon 9* first stage will land successfully. 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 *SpaceX* for a rocket launch.

- **Problems you want to find answers**

- **Correlations** between each rocket variables and successful landing rate
- Conditions to get the best results and ensure the best successful landing rate

Section 1

Methodology

Methodology

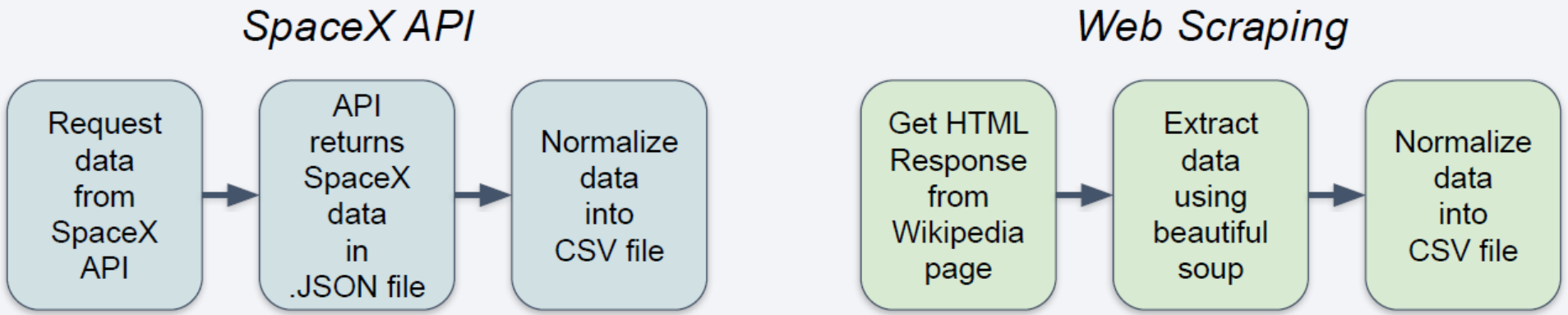
Executive Summary

- Data collection methodology:
  - SpaceX API & Web Scraping Falcon 9 and Falcon Heavy Launches Records from Wikipedia
- Perform data wrangling
  - Convert outcomes into Training Labels with the booster successfully/unsuccessful landed
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
- Find best Hyperparameter for SVM, Classification Trees and Logistic Regression

Data Collection

The data collection process includes a combination of API requests from the SpaceX API and web scraping data from a table in the Wikipedia page of SpaceX, *Falcon 9 and Falcon Heavy Launches Records*.

- SpaceX API Data Columns: `FlightNumber, Date, BoosterVersion, PayloadMass, Orbit, LaunchSite, Outcome, Flights, GridFins, Reused, Legs, LandingPad, Block, ReusedCount, Serial, Longitude, Latitude`
- Wikipedia Web Scrape Data Columns: `Flight No., Launch site, Payload, PayloadMass, Orbit, Customer, Launch outcome, Version Booster, Booster landing, Date, Time`



Data Collection – SpaceX API

1. IMPORT LIBRARIES:

```
# Requests allows us to make HTTP requests which we will i
import requests
# Pandas is a software library written for the Python prog
import pandas as pd
# NumPy is a library for the Python programming language,
import numpy as np
# Datetime is a library that allows us to represent dates
import datetime
```

2. REQUEST GET API / JSON LOADS/ NORMALIZE JSON:

```
# Use json_normalize meethod to convert the json result into a dataframe
import json
response = requests.get(static_json_url)
data_dict = json.loads(response.content)
data = pd.json_normalize(data_dict)
```

3. CONSTRUCTING DATASET USING JSON :

```
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}
```

4. FILTER DATAFRAME ONLY INCLUDE FALCON 9 LAUNCH:

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

5. DATA READY FOR WRANGLING:

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

GithubURL:
https://github.com/Explrador/Aplpied-Data-Science-CAPSTONE

Data Collection - Scraping

1. IMPORT LIBRARIES:

```
import sys

import requests
from bs4 import BeautifulSoup
import re
import unicodedata
import pandas as pd
```

2. REQUEST FALCON 9 LAUNCH WEB SCRAPING:

```
response = requests.get (static_url)
print(response.status_code)
print(response.content [0:100])
```

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

3. EXTRACT ALL COLUMN:

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

4. CONSTRUCTING DATASET USING WEB THE DATA:

```
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. Building dataframe:

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

Data Wrangling

1. IMPORT LIBRARIES AND LOADING DATASET:

```
# Pandas is a software library written for the Python programming l
import pandas as pd
#NumPy is a library for the Python programming language, adding sup
import numpy as np
from js import fetch
import io

URL = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain
resp = await fetch(URL)
dataset_part_1_csv = io.BytesIO((await resp.arrayBuffer()).to_py())
```

2. NUMBER OF LAUCHES ON EACH SITE:

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

CCAFS SLC 40	55
KSC LC 39A	22
VAFB SLC 4E	13

3. LANDING OUTCOMES IN SPECIFIC REGION:

```
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

4. FINDING THE BAD OUTCOMES TO FIND SUCCESS RATE

```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
landing_class = []
for i in df['Outcome']:
    if i in set(bad_outcomes):
        landing_class.append(0)
    else:
        landing_class.append(1)
df['Class']=landing_class
df[['Class']].head(8)
```

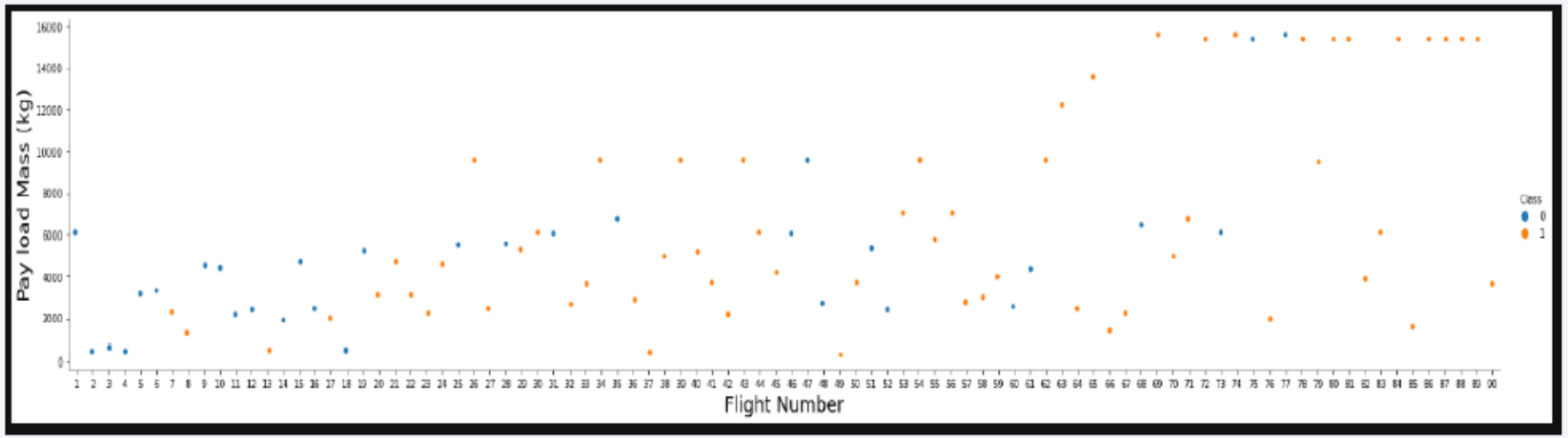
	Class
0	0
1	0
2	0
3	0

5. SUCESS RATE

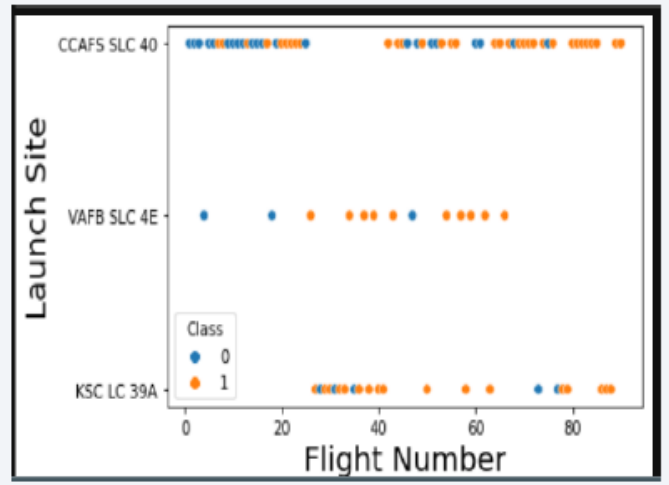
```
df["Class"].mean() * 100
```

```
66.66666666666666
```

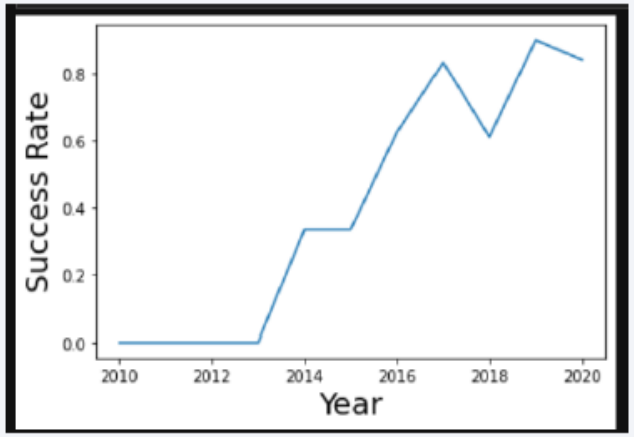
EDA with Data Visualization



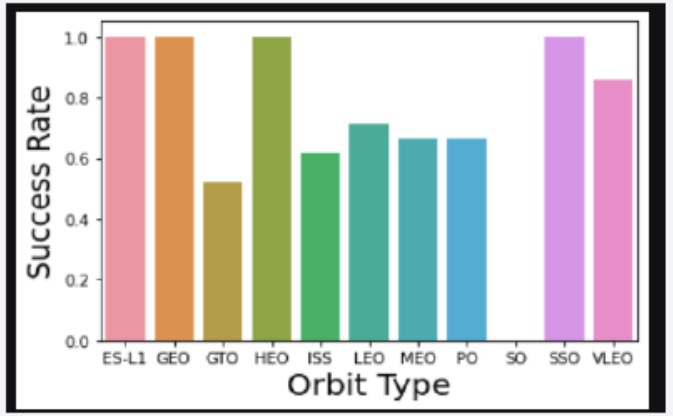
1. Plot the FlightNumber vs. PayloadMass(kg) and about the launch result



2. Relationship between Flight Number and Launch Site about the launch result



4. Launch success yearly trend



3. Success rate of each orbit type

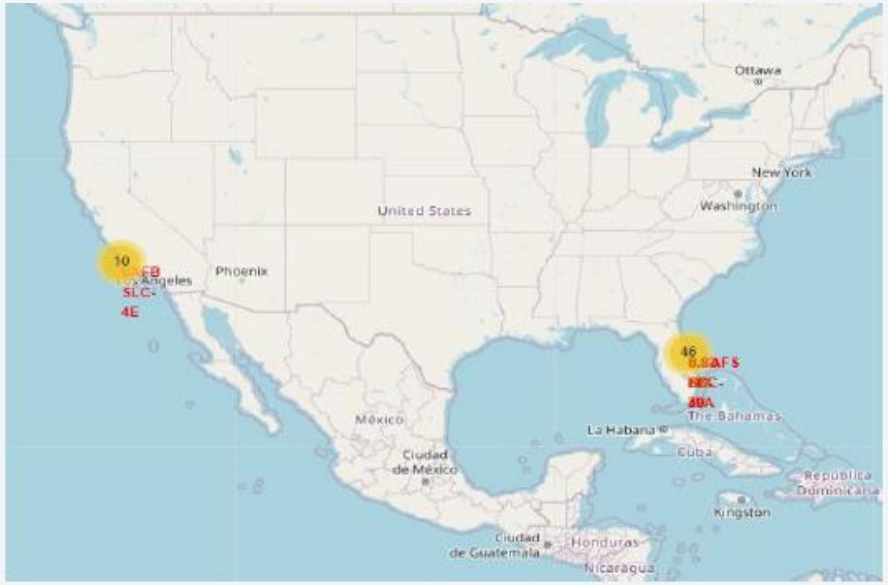
Github URL:

EDA with SQL

SQL queries to solve the assignment tasks:

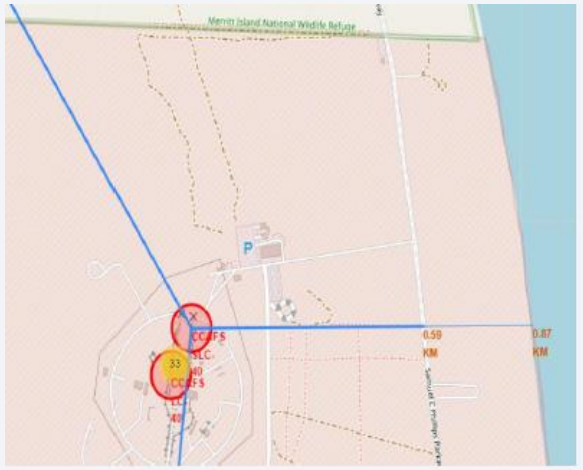
- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first succesful 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. Use a subquery
- List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.
- 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



s,

th



1- site_map = folium.Map - ADD map with informed coordinates
2- marker = folium.map.Marker - ADD markup or description
3- circle = folium.Circle - ADD circle at predefined coordinate
4- mouse_position = MousePosition - Add mouse position to get coordinate
5- lines=folium.PolyLine - ADD line on map
6- marker_cluster = MarkerCluster() - ADD grouping of markers
7- Site_map.add_child - ADD the previous items on the map

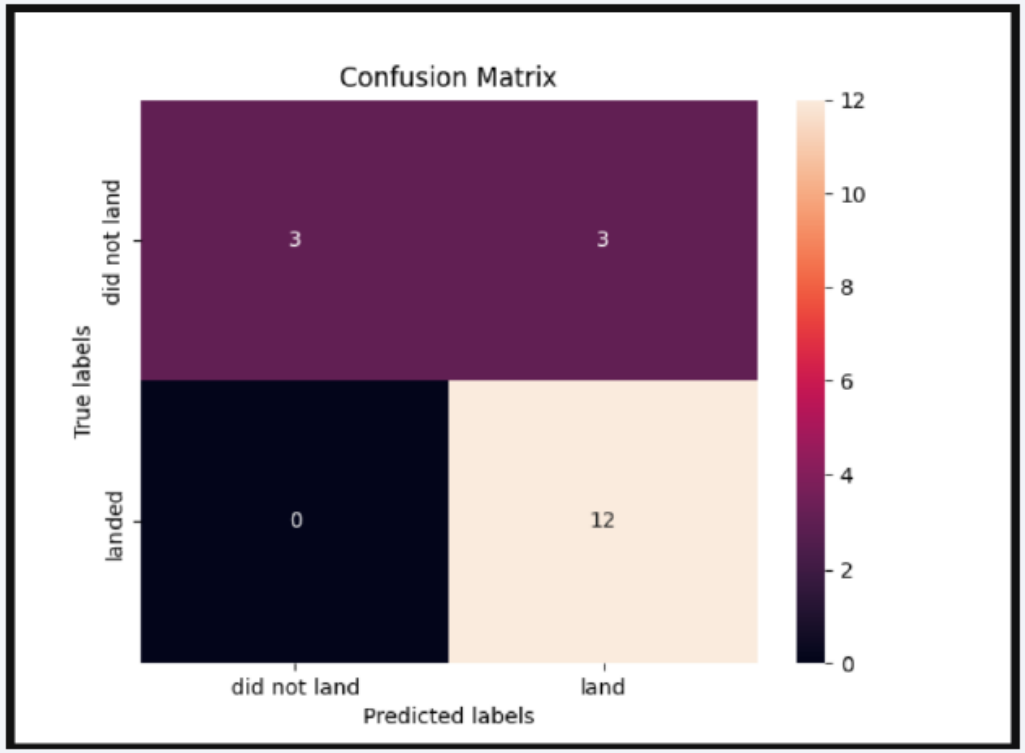
Build a Dashboard with Plotly Dash

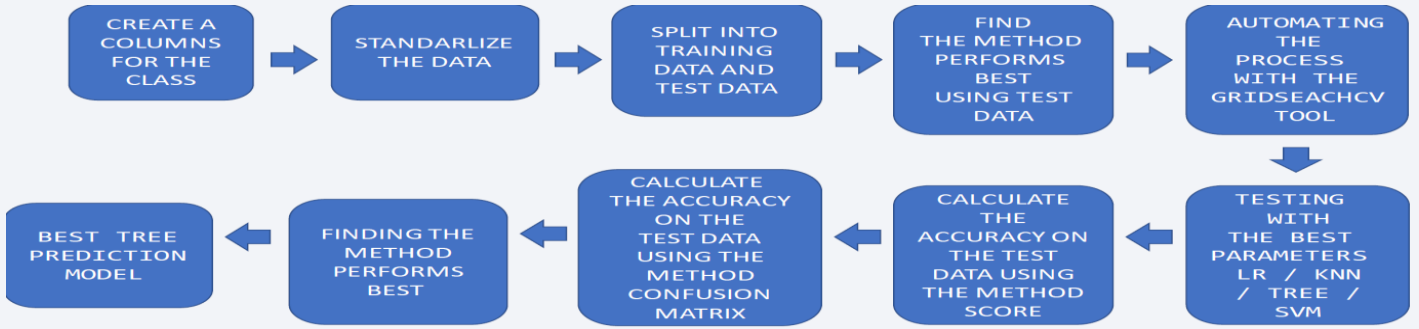
- The dashboard application contains a pie chart and a scatter point chart.
  - *Pie chart*
    - For showing total success launches by sites
    - This chart can be selected to indicate a successful landing distribution across all launch sites or to indicate the success rate of individual launch sites.
  - *Scatter chart*
    - For showing the relationship between `Outcomes` and `Payload mass (Kg)` by different boosters
    - Has 2 inputs: All sites/individual site & Payload mass on a slider between 0 and 10000 kg
    - This chart helps determine how success depends on the launch point, payload mass, and booster version categories.

Predictive Analysis (Classification)

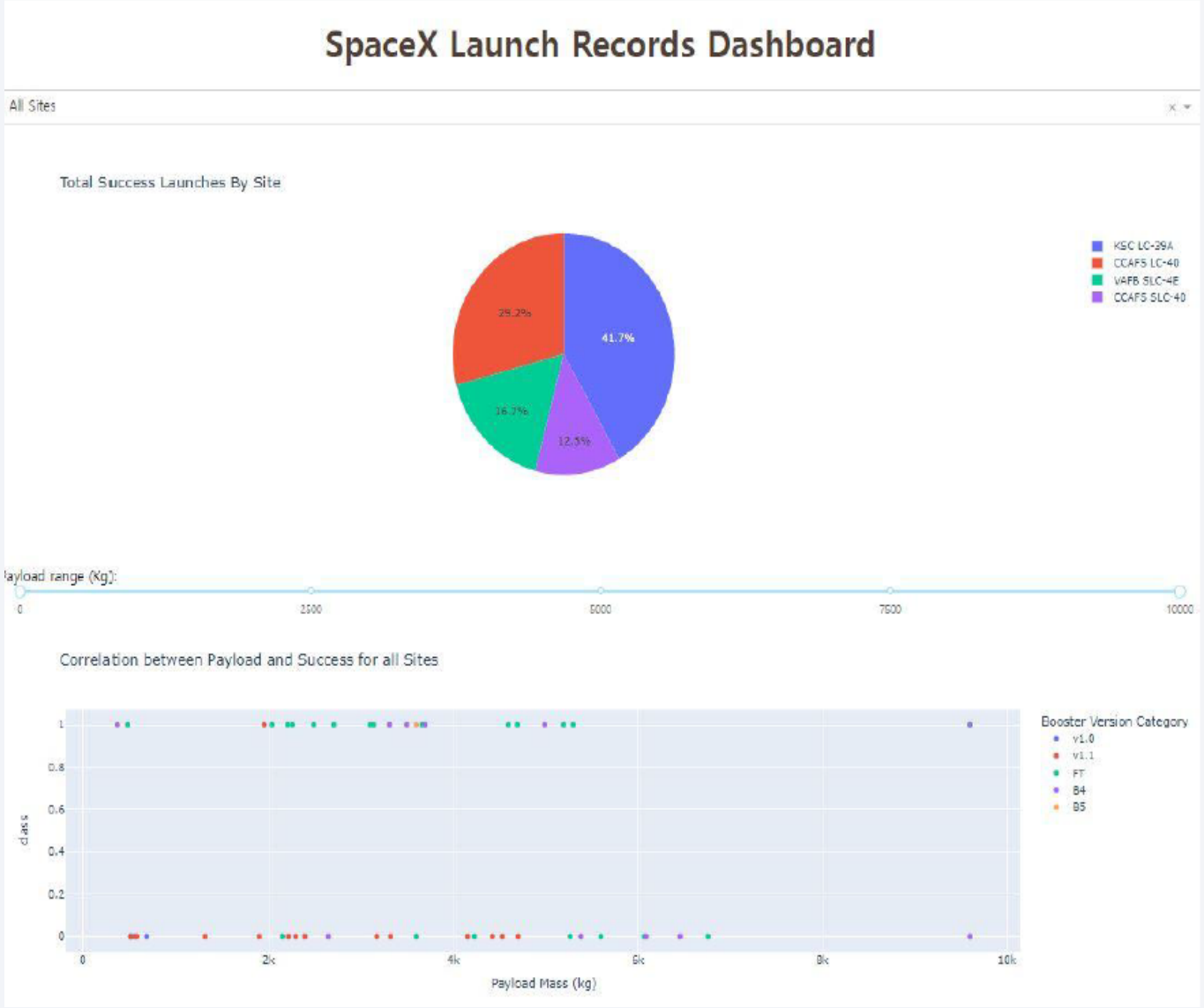
```
parameters = {'criterion': ['gini', 'entropy'],
     'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],
     'max_features': ['auto', 'sqrt'],
     'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10]}

tree = DecisionTreeClassifier()
tree_cv=GridSearchCV(tree, parameters, cv=10, scoring='accuracy')
tree_cv.fit(X_train,Y_train)
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
print('Accuracy on test data is: {:.3f}'.format(tree_cv.score(X_test, Y_test)))
```

```
Accuracy on test data is: 0.944
```

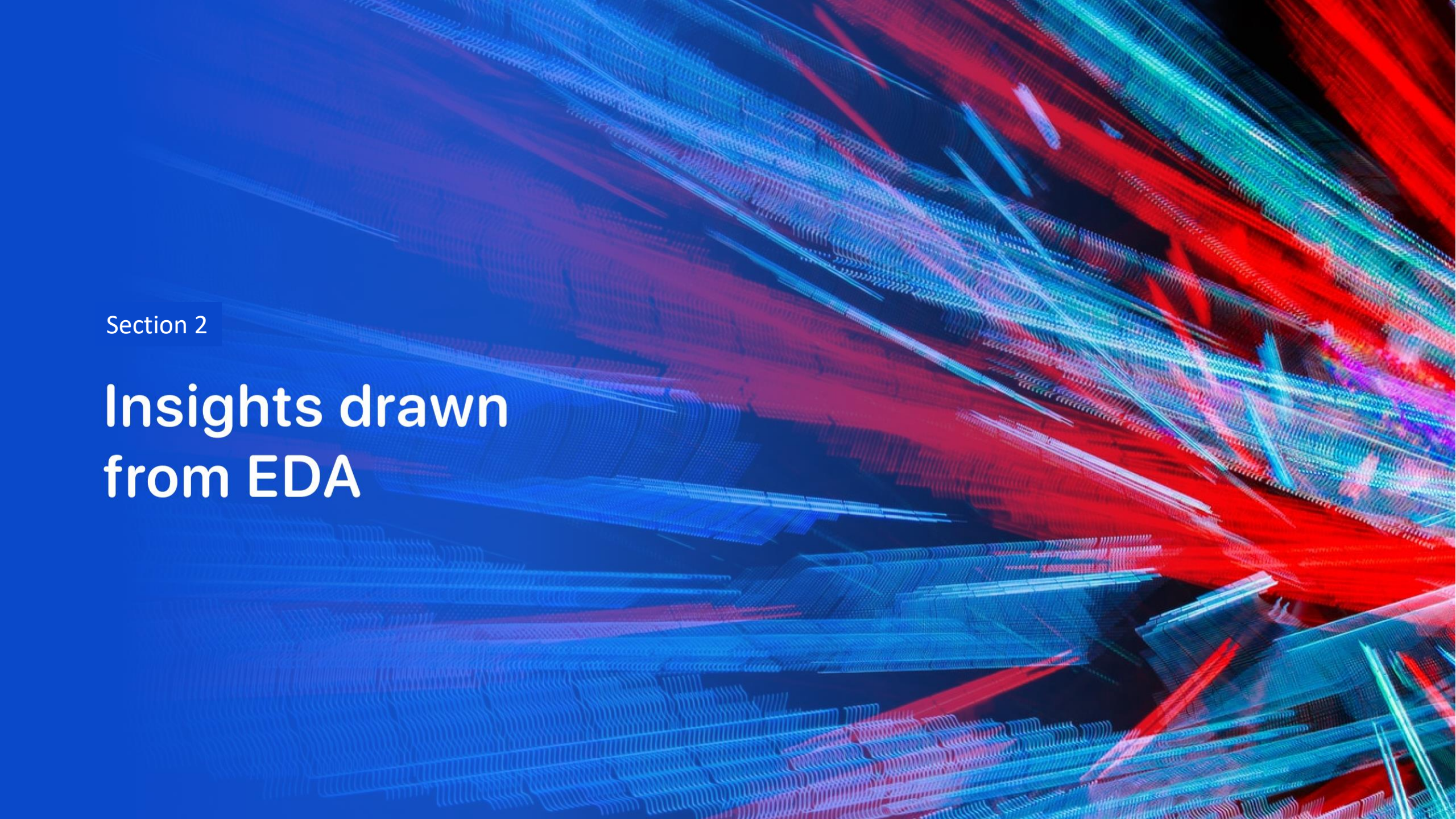




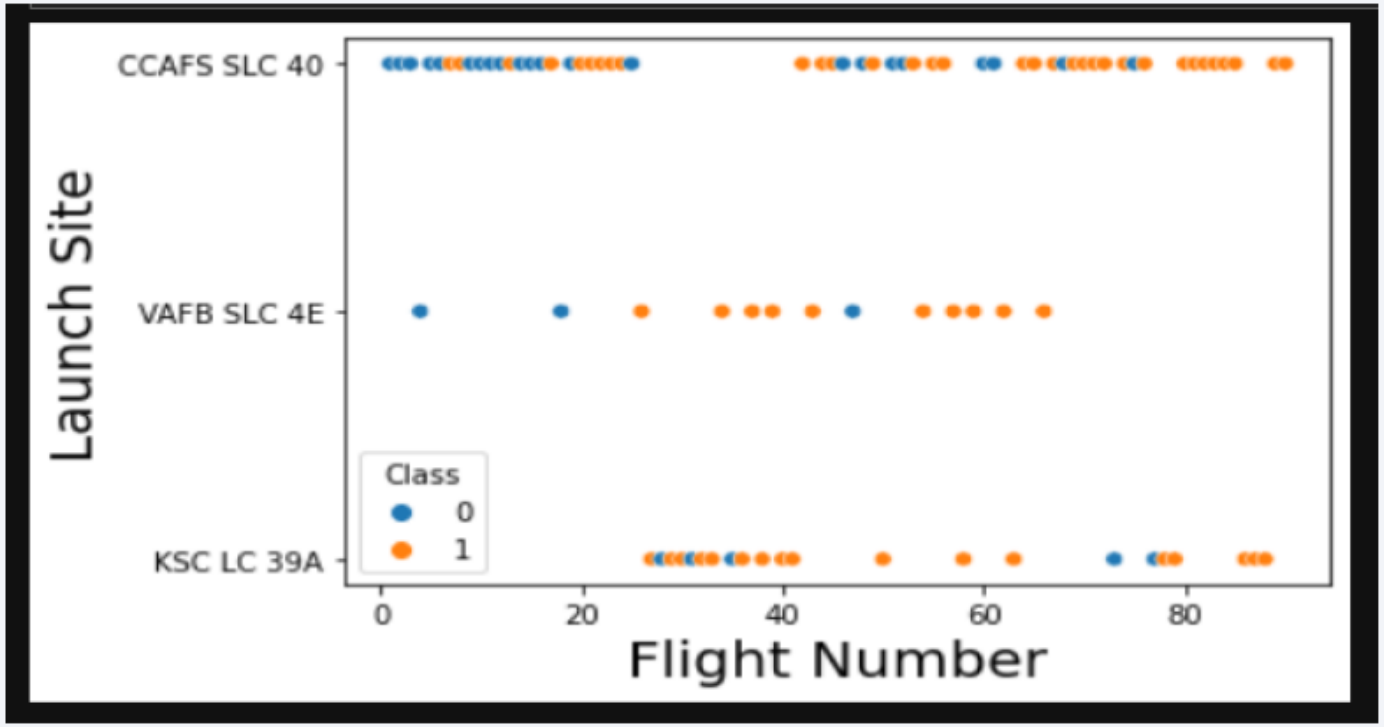
Results



- The left screenshot is a preview of the Dashboard with Plotly Dash.
- The results of EDA with visualization, EDA with SQL, Interactive Map with Folium, and Interactive Dashboard will be shown in the next slides.
- Comparing the accuracy of the four methods, all return the same accuracy of about *83%* for test data.

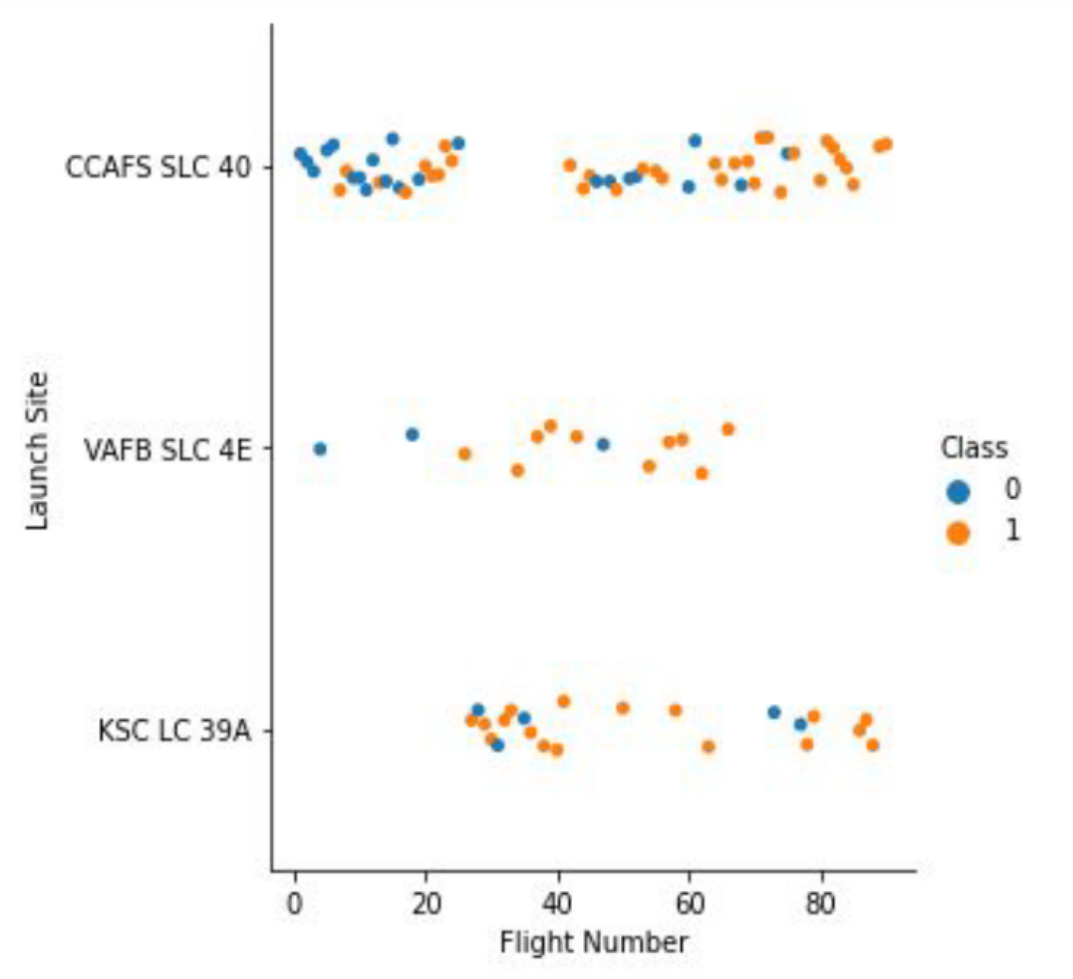

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



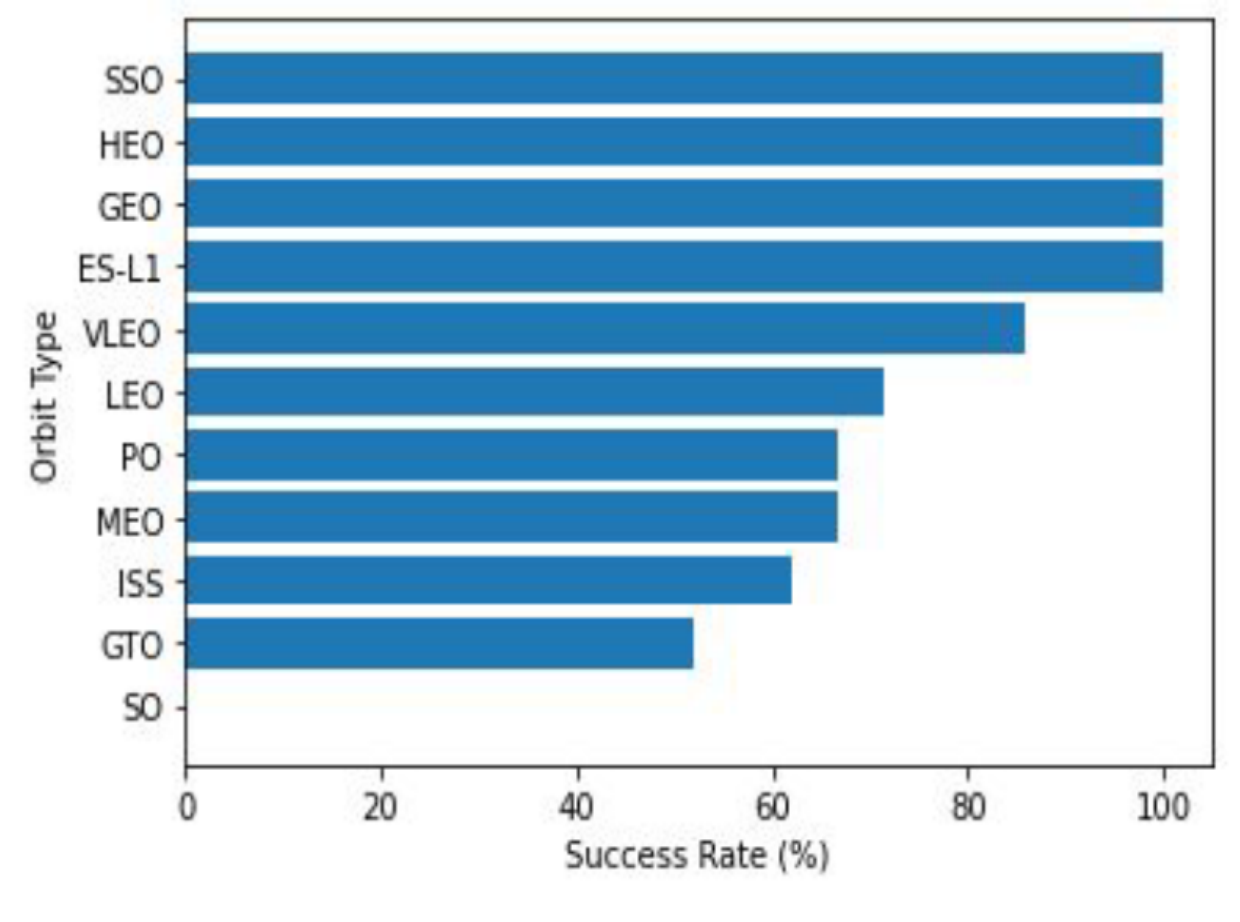
- ✓ Success rate PFB SLC 4E and KSC LC 39A is higher CCAFS SLC 40 even with lower launch amounts.
- ✓ As more experience launches on each site, the success rate increases.

Payload vs. Launch Site



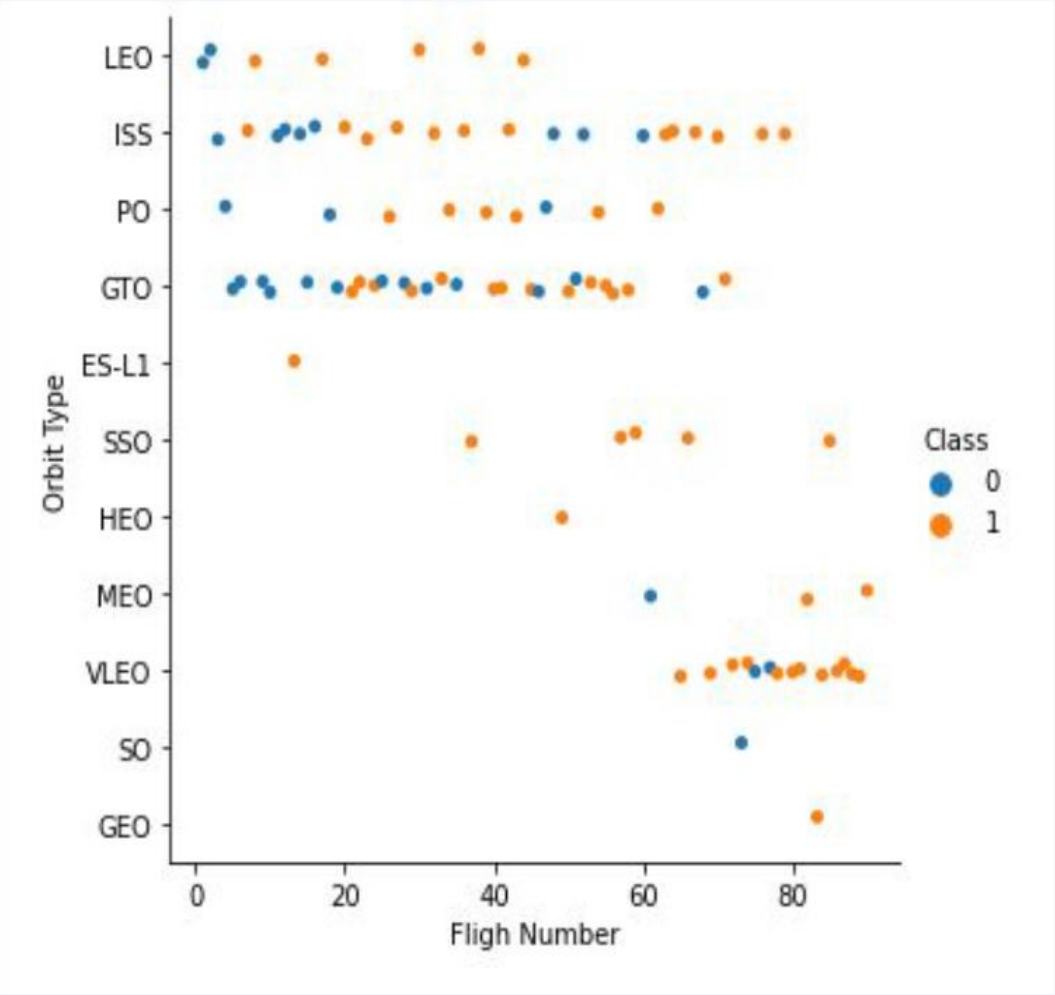
- `Class 0` (blue) represents unsuccessful launch, and `Class 1` (orange) represents successful launch.
- This figure shows that **the success rate increased as the number of flights increased**.
- As the success rate has increased considerably since the *20th* flight, this point seems to be a big breakthrough.

Success Rate vs. Orbit Type



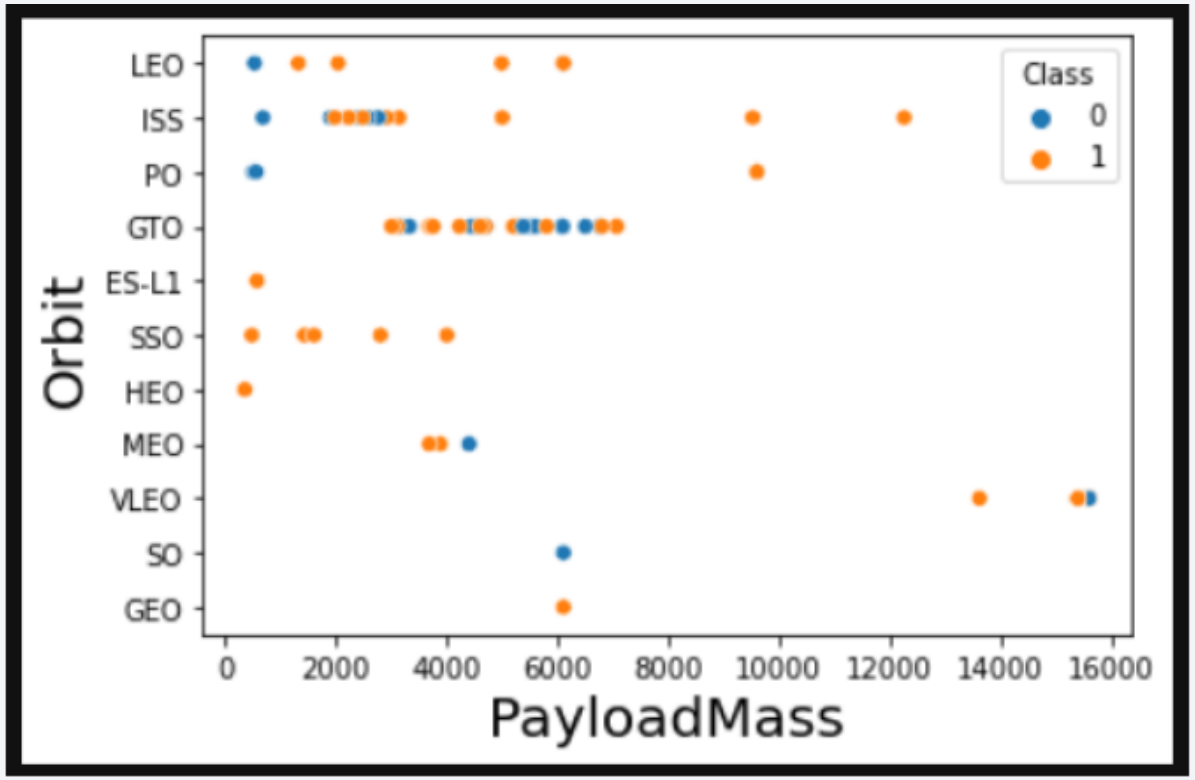
- Orbit types **SSO**, **HEO**, **GEO**, and **ES-L1** have **the highest success rates** (*100%*).
- On the other hand, the success rate of orbit type **GTO** is only *50%*, and it is the **lowest** except for type SO, which recorded failure in a single attempt.

Flight Number vs. Orbit Type



- `Class 0` (blue) represents unsuccessful launch, and `Class 1` (orange) represents successful launch.
- In most cases, the launch outcome seems to be correlated with the flight number.
- On the other hand, in **GTO** orbit, there seems to be **no** relationship between flight numbers and success rate.
- SpaceX starts with LEO with a moderate success rate, and it seems that VLEO, which has a high success rate, is used the most in recent launches.

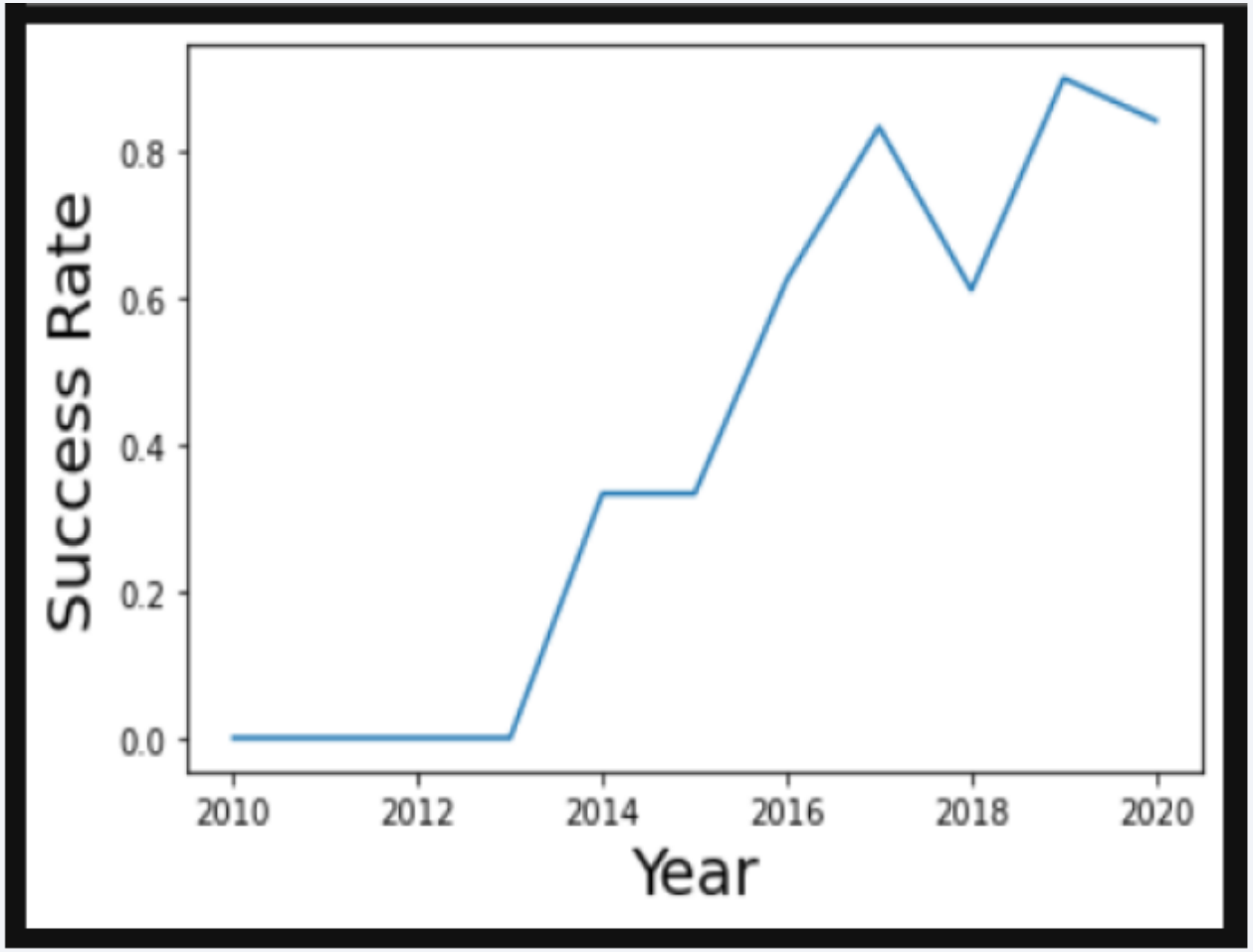
Payload vs. Orbit Type



- ✓ For the LEO, ISS and PO orbits after a payload of 4000kg the success rate was 100%.
- ✓ SSO orbit had 100% success rate with payloads up to 4000kg.
- ✓ GTO orbit has unstable success rate regardless of payload.

Launch Success Yearly Trend

The success rate from 2013 had a great growth until 2017, with the growth being interrupted in 2018 and the rate increasing again in 2019. It is possible that the experience had a great impact on success.



All Launch Site Names

```
sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL ORDER BY 1
 * sqlite:///my_data1.db
Done.
```

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

All site where used flom Launch

Launch Site Names Begin with 'CCA'

- Query

```
SELECT * FROM SPACEXTBL
WHERE LAUNCH_SITE LIKE 'CCA%'
LIMIT 5
```

- Result

- Only five records of the SpaceX table were displayed using `LIMIT 5` clause in the query.
- Using the `LIKE` operator and the percent sign (%) together, the `Launch_Site` name starting with `CAA` could be called.

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	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

```
sql SELECT SUM (PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE CUSTOMER='NASA (CRS)'
```

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

SUM (PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

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

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

AVG (PAYLOAD_MASS__KG_)
2534.6666666666665

First Successful Ground Landing Date

```
sql SELECT MIN(Date) FROM SPACEXTBL WHERE Mission_outcome LIKE 'Success%'
```

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

MIN(Date)
01-03-2013

Successful Drone Ship Landing with Payload between 4000 and 6000

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

```
 * 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

```
sql SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) FROM SPACEXTBL WHERE MISSION_OUTCOME like 'Success%'
```

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

Mission_Outcome	COUNT(MISSION_OUTCOME)
Success	100

```
sql SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) FROM SPACEXTBL WHERE MISSION_OUTCOME = 'Failure (in flight)'
```

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

Mission_Outcome	COUNT(MISSION_OUTCOME)
Failure (in flight)	1

Boosters Carried Maximum Payload

```
sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
 * sqlite:///my_data1.db
Done.
```

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

```
sql SELECT substr(Date, 4, 2) AS Month, substr(Date, 7, 4) as Year,BOOSTER_VERSION, "Landing _Outcome",launch_site FROM SPACEXTBL where substr(Date
```

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

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

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

```
SELECT LANDING__OUTCOME,
       COUNT(LANDING__OUTCOME) AS total_number
FROM SPACEXTBL
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY LANDING__OUTCOME
ORDER BY total_number DESC
```

landing__outcome	total_number
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

In the WHERE clause, filter the dataset to perform a search if the date is between 2010-06-04 and 2017-03-20.

Using the ORDER BY keyword to sort the records by total number of landing, and using DESC keyword to sort the records in descending order.

According to the results, the number of successes and failures between 2010-06-04 and 2017-03-20 was similar.

Section 3

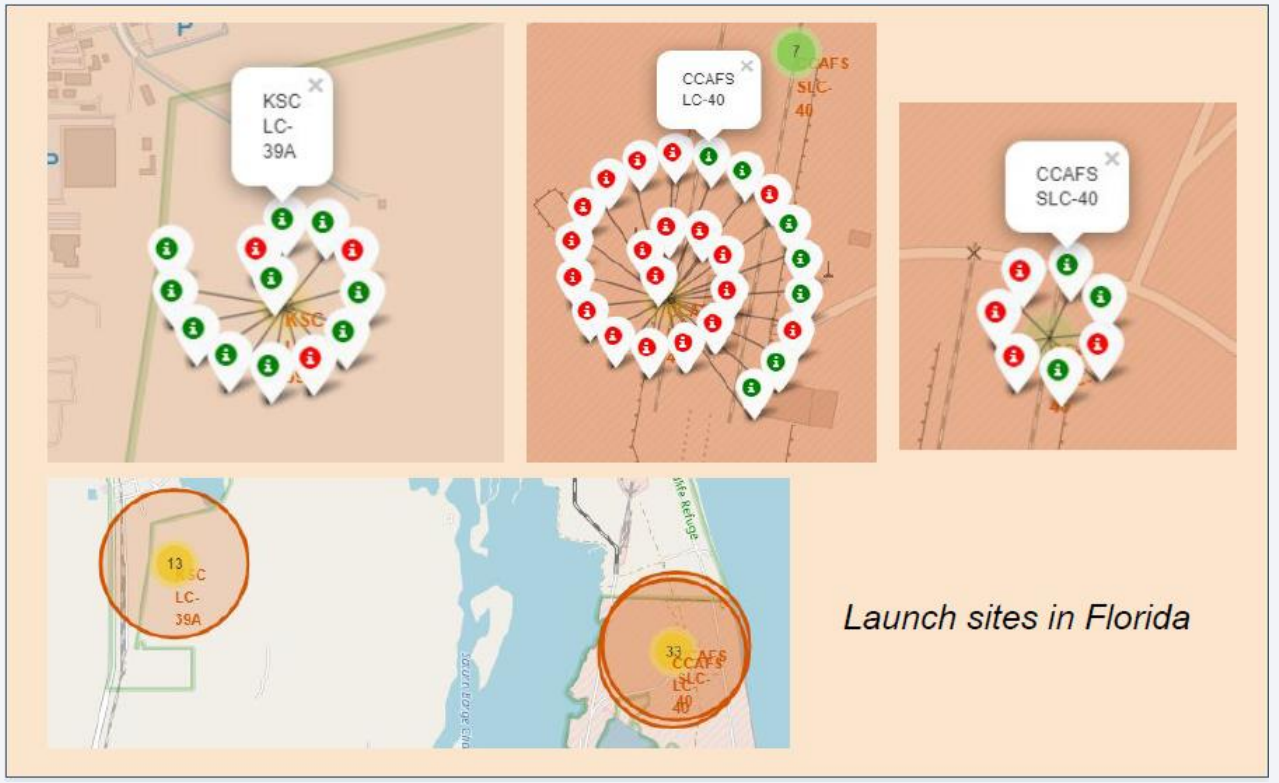
Launch Sites Proximities Analysis

<Folium Map Screenshot 1>

✓ ALL LAUNCH LOCATIONS ARE NEAR THE SEA

<Folium Map Screenshot 2>

Launch site in California

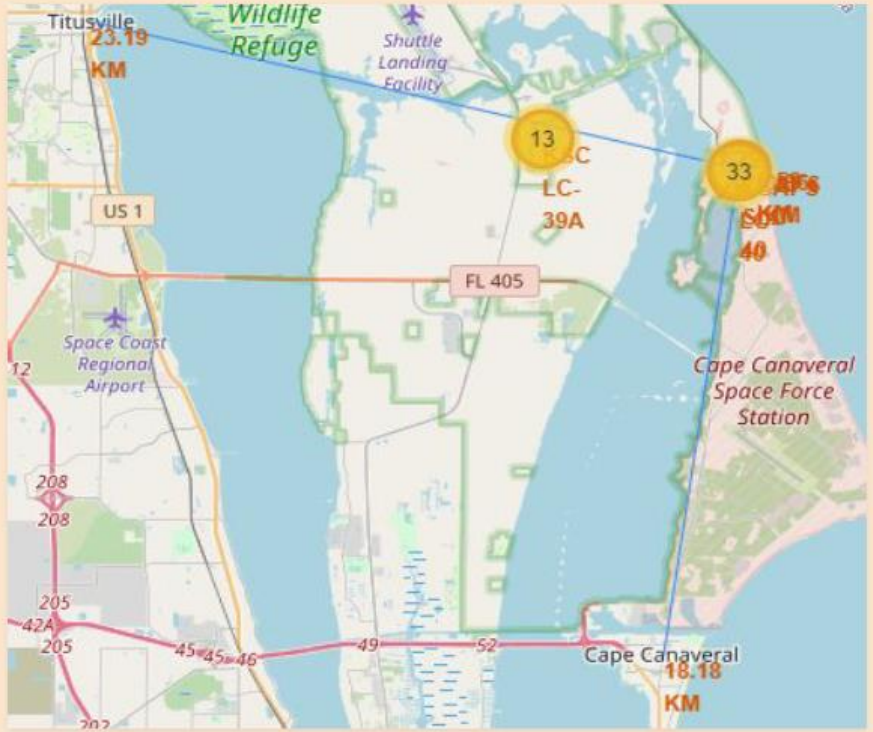


Launch sites in Florida

- By clicking on the marker clusters,
  successful landing (green) or failed landing (red) are displayed.

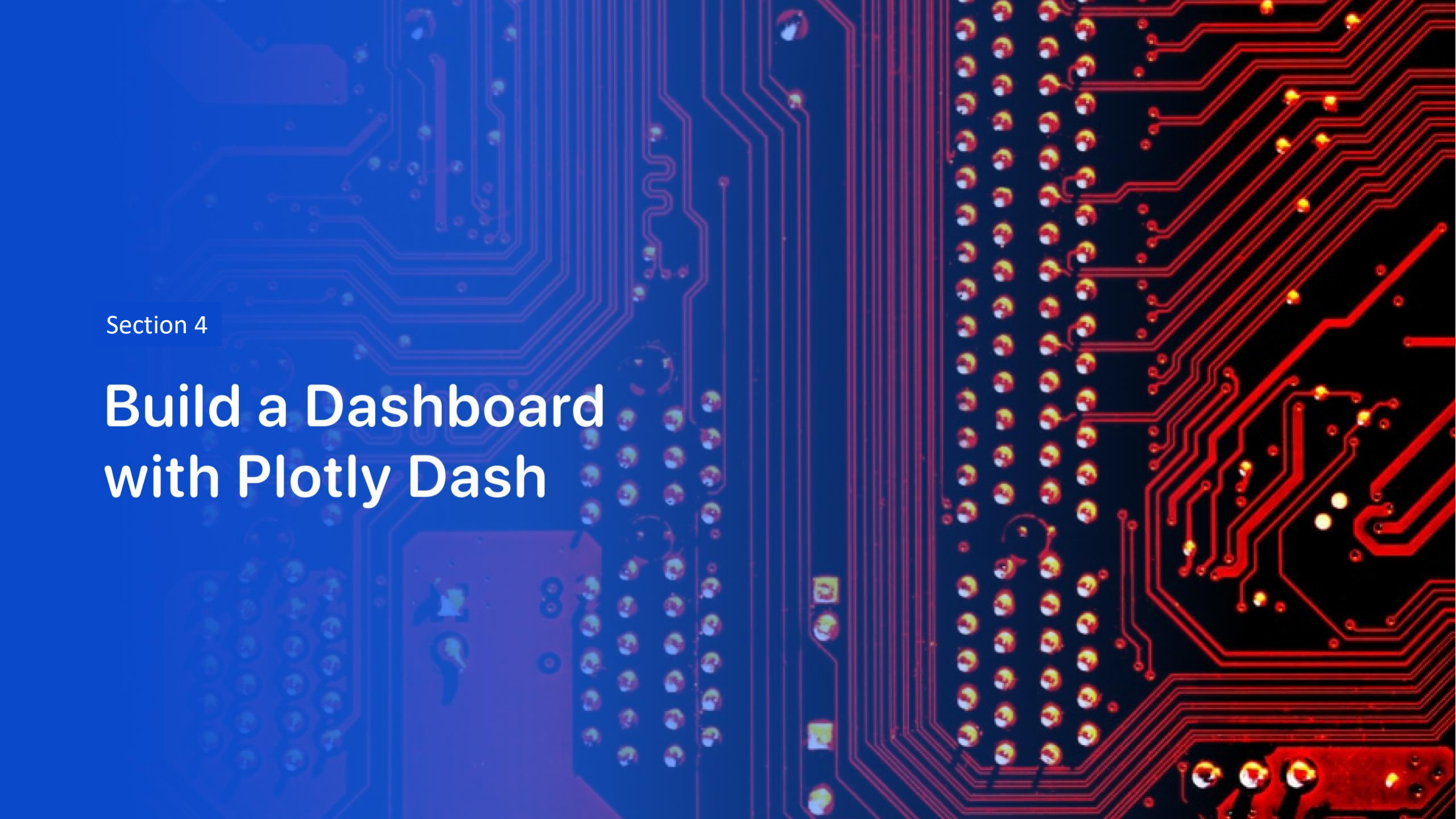
<Folium Map Screenshot 3>

Are launch sites in close proximity to railways? ***Yes***
Are launch sites in close proximity to highways? ***Yes***
Are launch sites in close proximity to coastline? ***Yes***

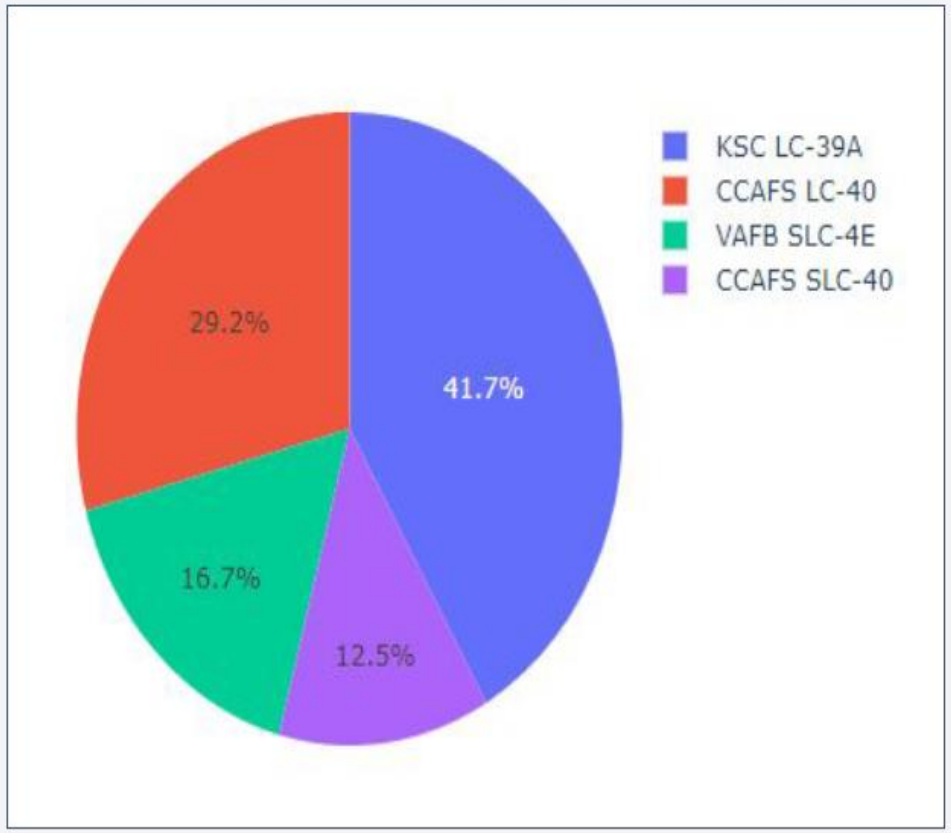


Do launch sites keep certain distance away from cities? ***Yes***

- It can be found that the launch site is **close to railways and highways** for transportation of equipment or personnel, and is also **close to coastline** and relatively **far from the cities** so that launch failure does not pose a threat.

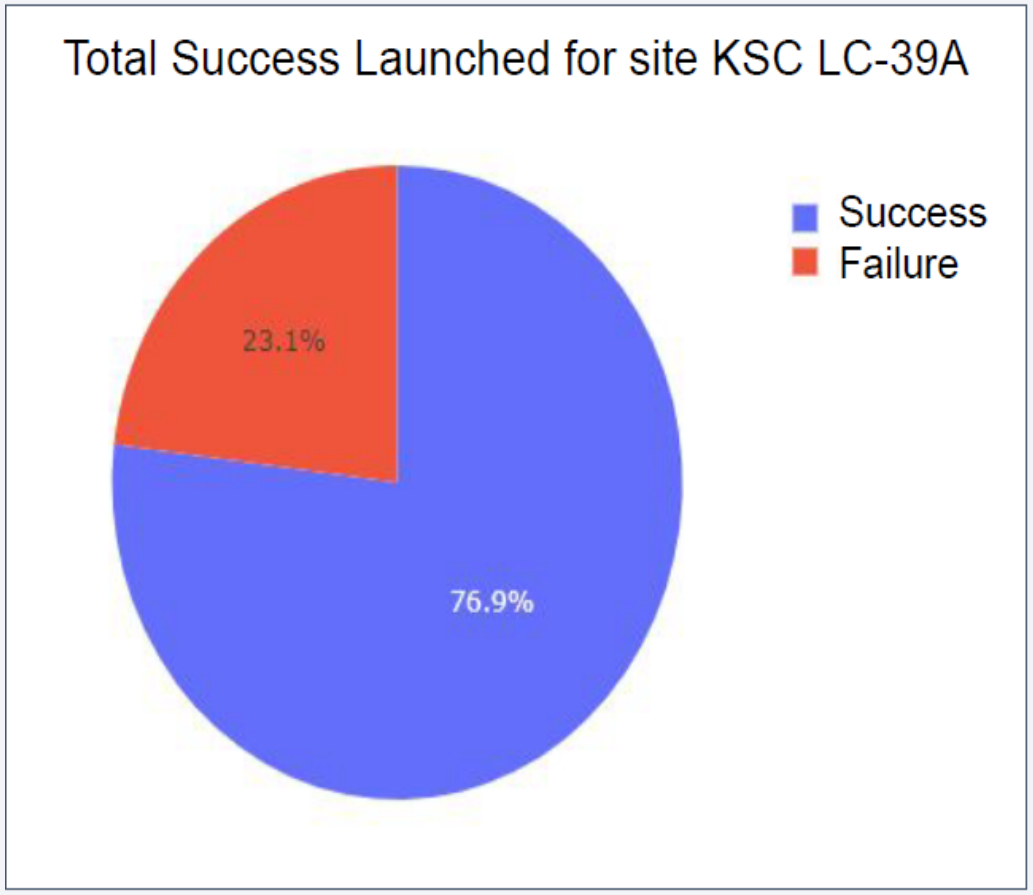

Section 4
Build a Dashboard with Plotly Dash

<Dashboard Screenshot 1>



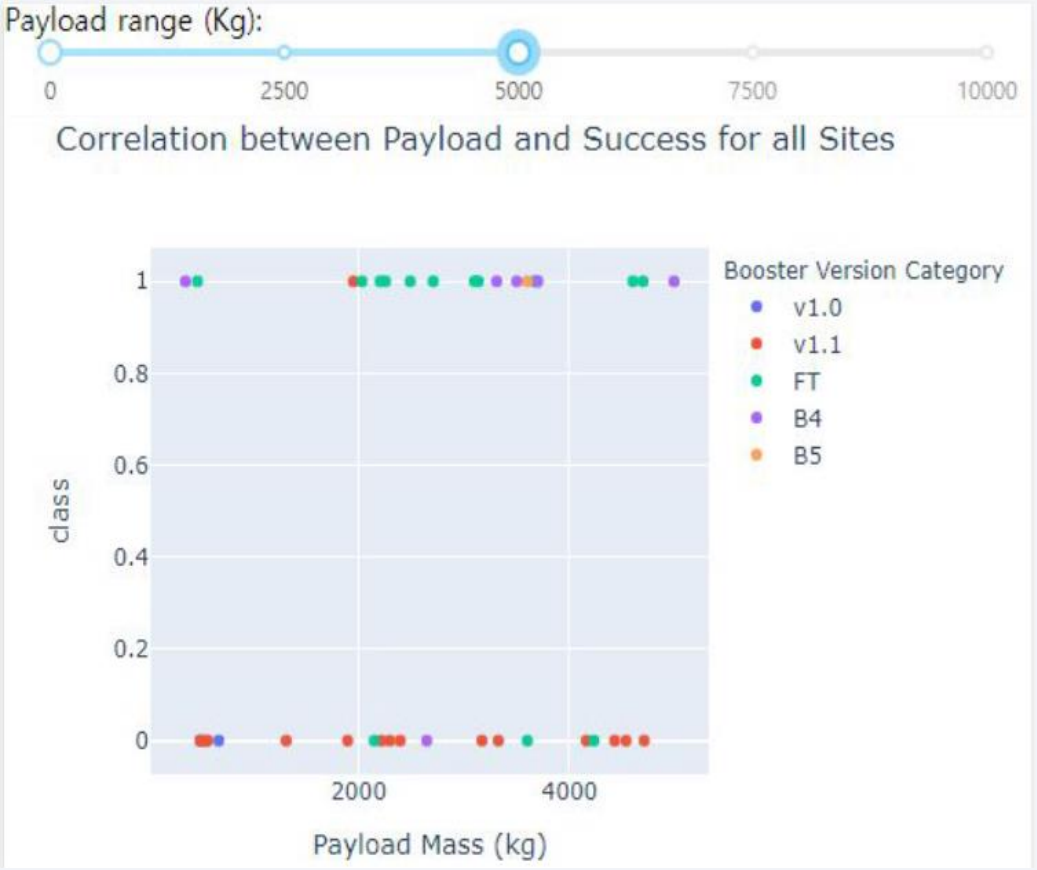
- KSLC-39A records the most launch success among all sites.
- The VAFB SLC-4E has the fewest launch success, possibly because
  - the data sample is small, or
  - because it is the only site located in California, so the launch difficulty on the west coast may be higher than on the east coast.

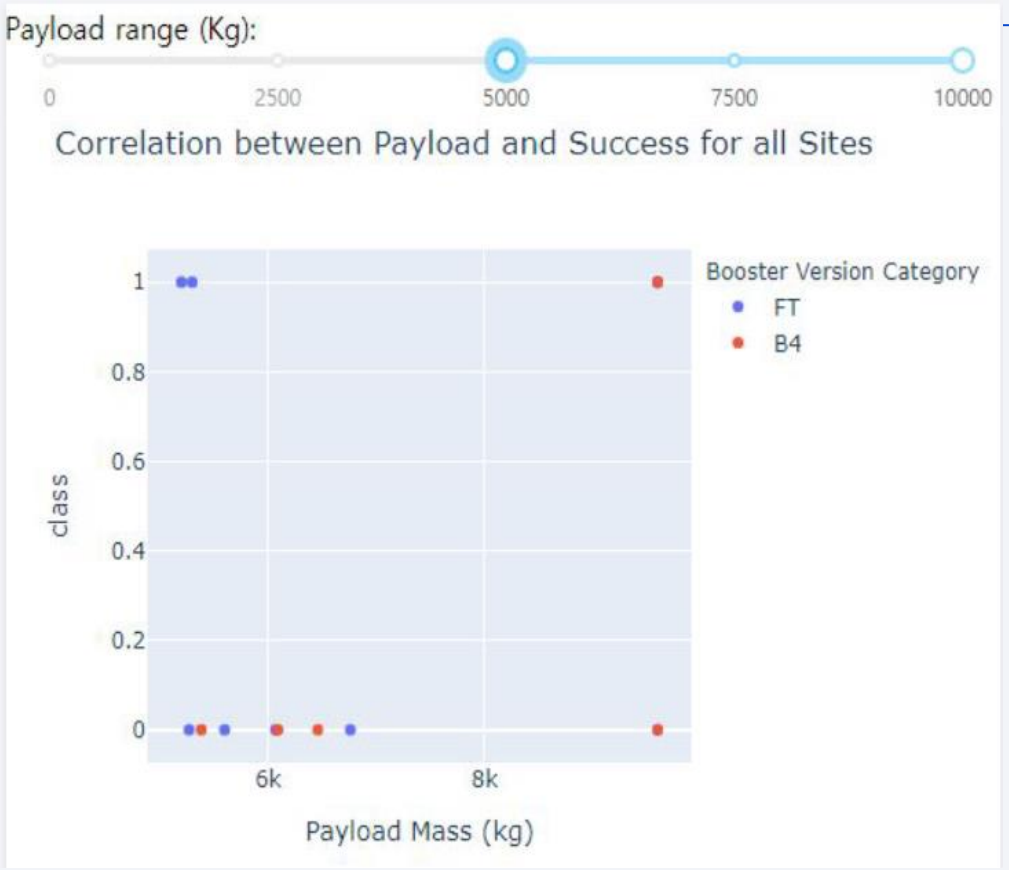
<Dashboard Screenshot 2>



- KSLC-39A has the highest success rate with 10 landing successes *(76.9%)* and 3 landing failures *(23.1%)*.

<Dashboard Screenshot 3>



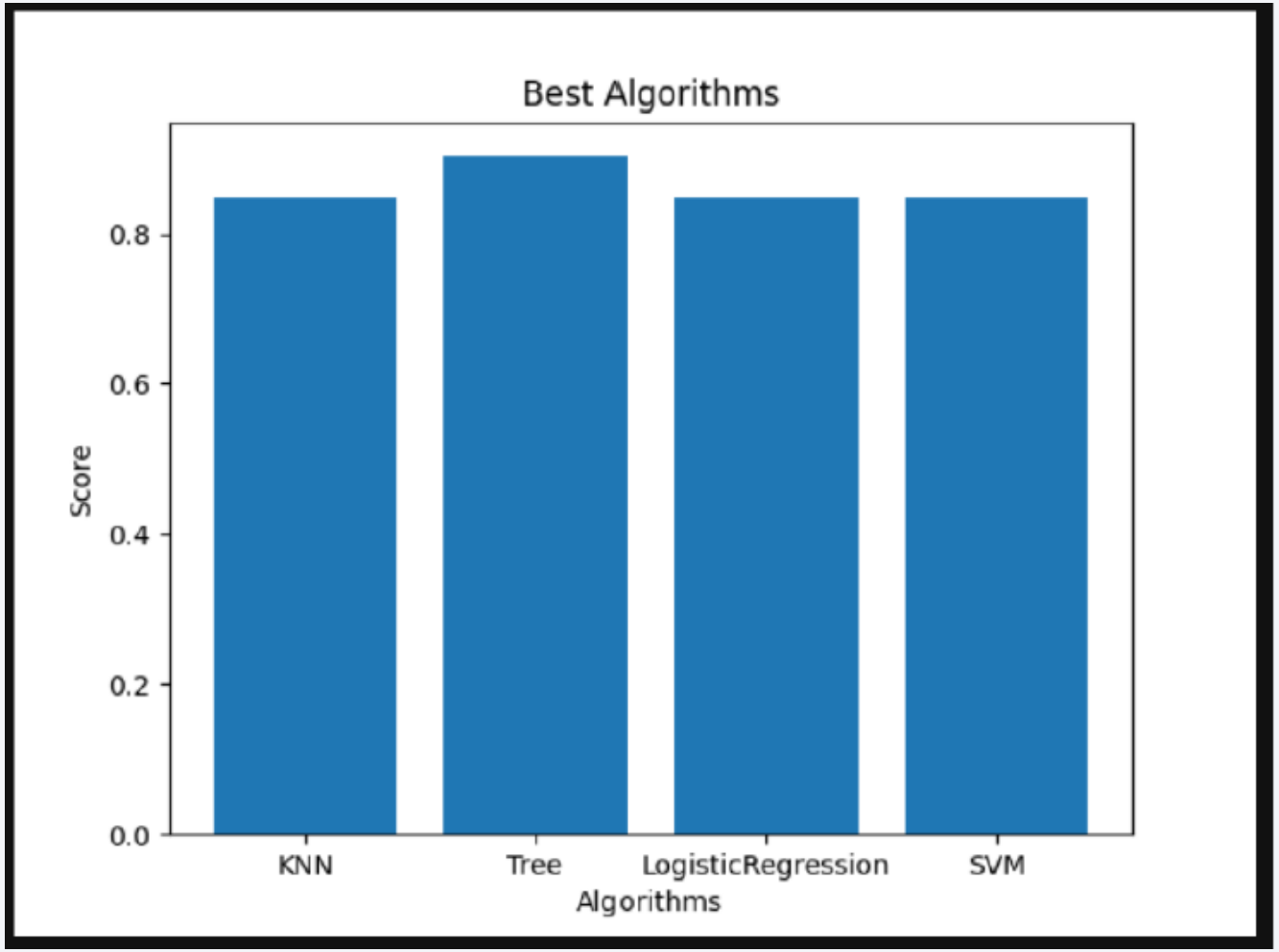


- These figures show that **the launch success rate** *(class 1)* **for low weighted payloads** ***(0-5000 kg)*** **is higher** than that of heavy weighted payloads *(5000-10000 kg)*.

Section 5

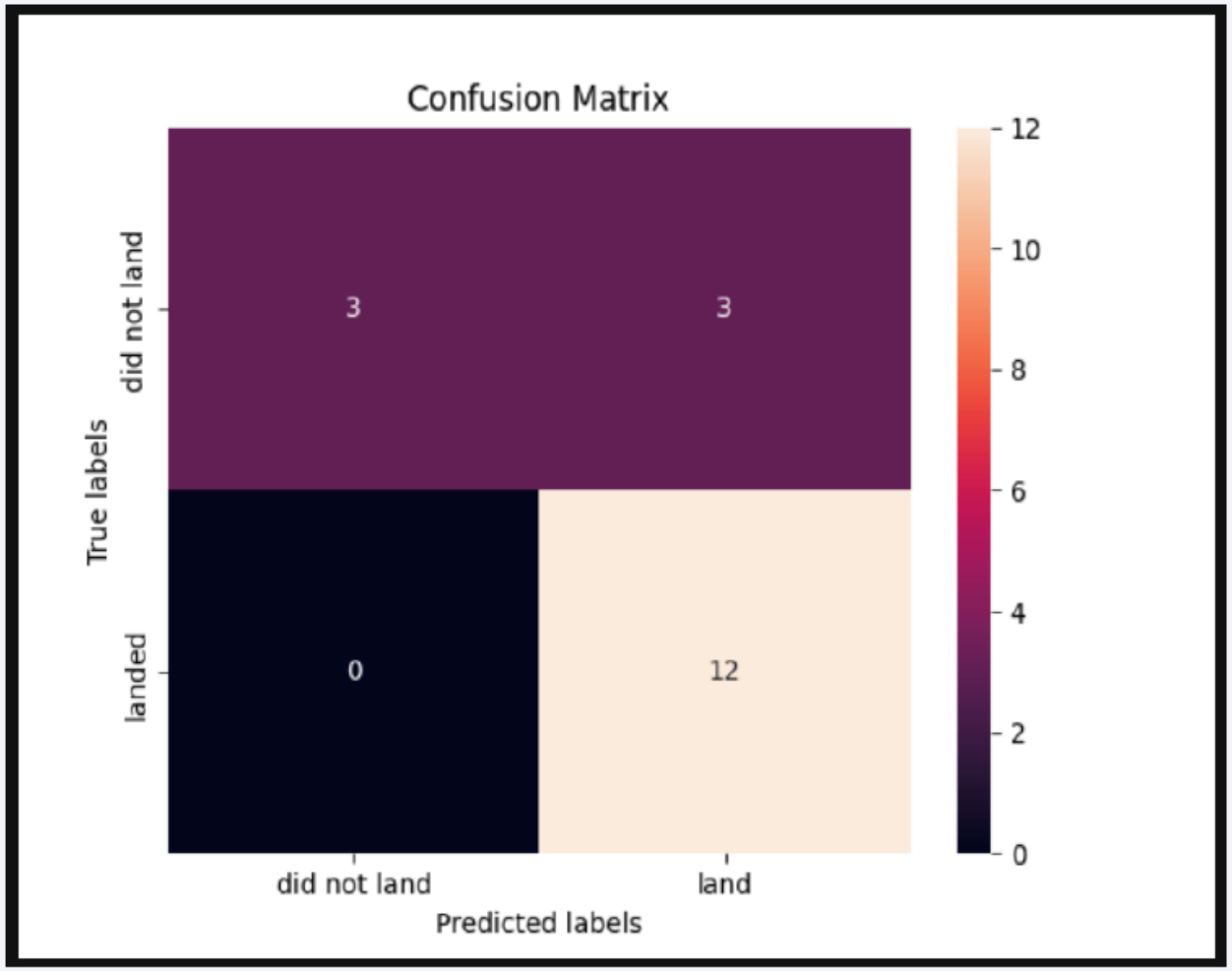
Predictive Analysis (Classification)

Classification Accuracy



✓ Decision Tree model has the highest classification accuracy

Confusion Matrix



- ✓ Decision tree model confusion matrix
- ✓ Predicted success results (TP) had an assertiveness of 80% and False positive (FP) 20%.
- ✓ With advancing technology and new calculation algorithms, the results for true negatives and false positives tend to become more accurate.

Conclusions

- Point 1 Launch site with the most successful result was the KSC LC 39A with a payload of up to 5500kg for larger loads do not use Booster version FT, for the other Boosters there is not enough data.
- Point 2 Booster version V1.0 and V1.1 with load up to 4000kg has the lowest success rate.
- Point 3 Payloads between 4000 and 8000 kg had the lowest success rate for all sites.
- Point 4 Booster version FT in the payload range between 2000-4000kg has the highest success rate.
- Point 5 The success rate from 2013 had a great growth until 2017, with the growth being interrupted in 2018 and the rate increasing again in 2019. It is possible that the experience had a great impact on success.
- Point 6 For the LEO, ISS and PO orbits after a payload of 4000kg the success rate was 100%.
- Point 7 SSO orbit had 100% success rate with payloads up to 4000kg.
- Point 8 GTO orbit has unstable success rate regardless of payload.
- Point 9 Due to the greater proximity to the earth operating at the observation points, the VLEO orbit has been more used in recent launches.
- Point 10 LEO's orbit has increased the rate of launches with more experience.

Appendix

- Acknowledgment to:

https://www.coursera.org/

https://cloud.ibm.com/

https://github.com/

Thank you!