Winning Space Race

by Rohit Adhikari

Outline

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

Introduction

SpaceX's Falcon 9 rocket launches are estimated to cost around $62 million per launch, significantly lower than the traditional $165 million. This cost reduction is largely due to SpaceX's ability to reuse the rocket's first stage, which carries the payload. Accurately predicting the success of this stage can help determine the overall cost of a launch.
The objective of this project is to develop a machine learning pipeline to predict whether the Falcon 9's first stage will successfully land, providing valuable insights for estimating launch costs.

- **Problems you want to find answers**
  - Factors that contribute to the success of a rocket landing
  - Effects of each attribute towards the outcome

Executive Summary

- **Summary of methodologies**
  - Data Collection using APIs, Web Scraping
  - Data Wrangling
  - EDA using SQL, Pandas, Matplotlib
  - Interactive Geospatial maps using Folium
  - Predictive Analysis for individual classification models
- **Summary of all results**

  - Best models for our Predictions
  - Data analysis with interactive visuals

Section 1

Methodology

Methodology

Executive Summary

- **Data collection methodology:**
  - Web Scraping on Wikipedia
  - SpaceX REST Api
- **Perform data wrangling**
  - One Hot Encoding on 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

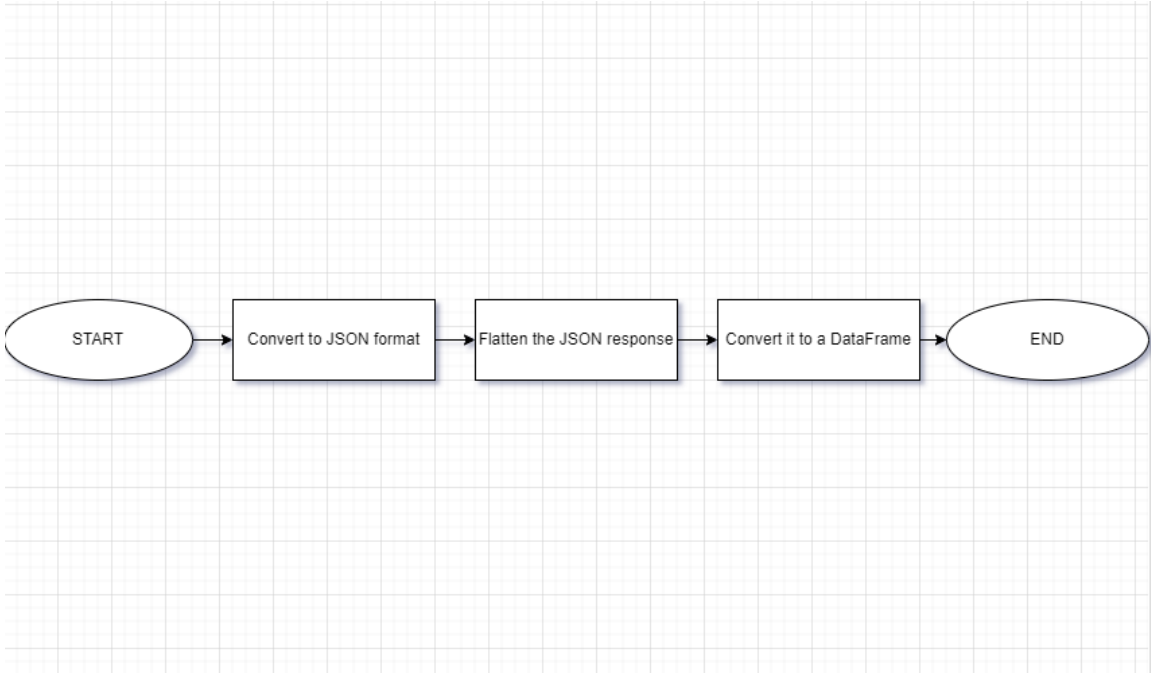
Data Collection

Describe how data sets were collected

- **SpaceX API**
  - Using a GET request
  - Decoded the GET response to a JSON format using .json() method and flattened it using the .json_normalize() function
- **Wikipedia**
  - Web Scraped Using BeautifulSoup to gather required Falcon 9 data
  - Extract all the necessary tables and convert it to a pandas dataframe
- Check and fill null values either using the mean and/or dropping the data record

Data Collection – SpaceX API

- Used the GET request on the SpaceX Api to gather information. Then cleaned the data by inserting extra features and dropping unnecessary features. Finally converting it to a workable dataframe.
- Github Link: https://github.com/rohitadhikarii/SpaceX-Falcon9-/blob/main/jupyter-labs-spacex-data-collection-api.ipynb



Data Collection – SpaceX API

```python
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```python
response = requests.get(spacex_url)
```

```python
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

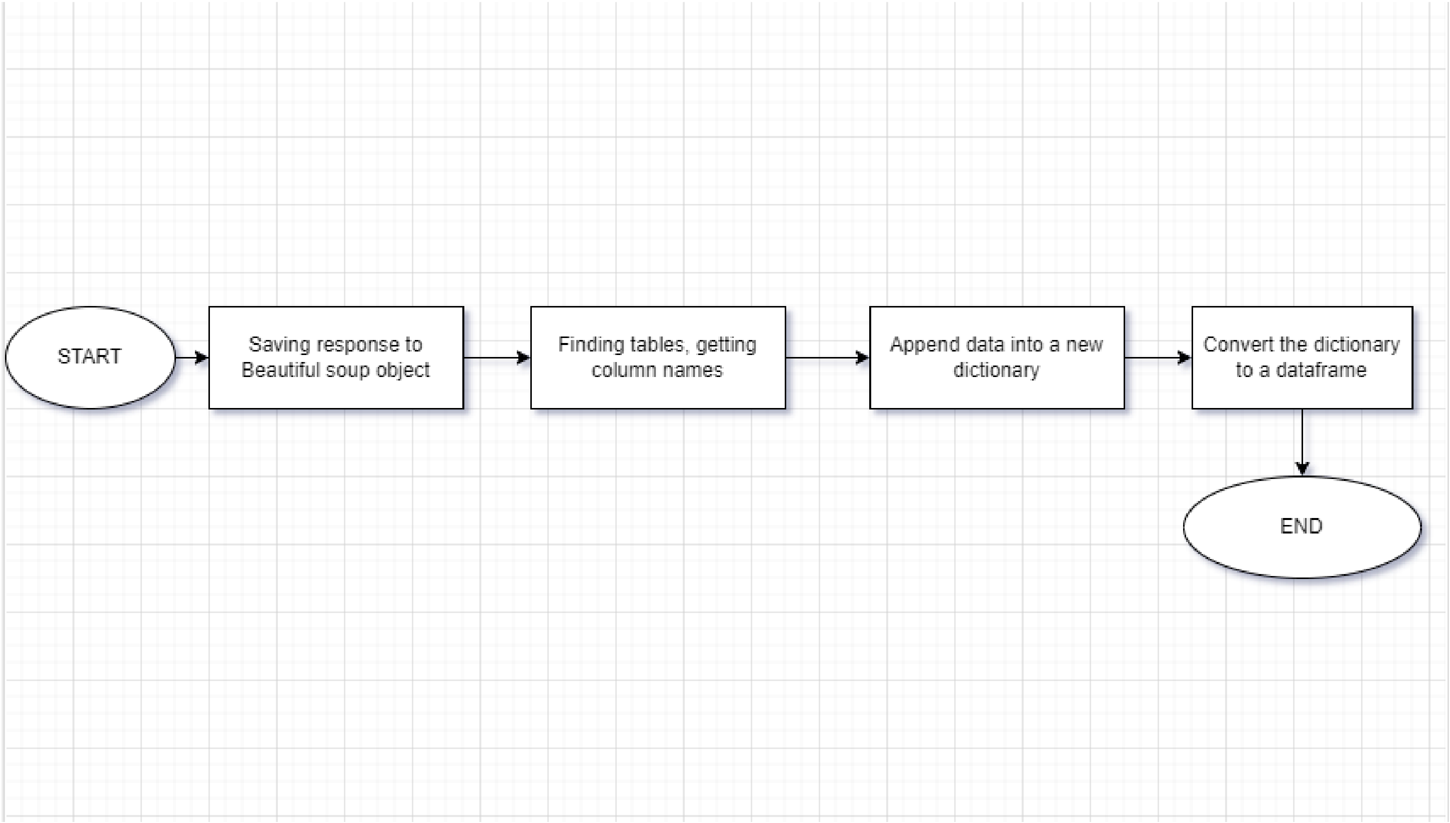
```python
# Use json_normalize meethod to convert the json result into a dataframe
json_resp = response.json()
data= pd.json_normalize(json_resp)
```

```python
# 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 rows that have multiple payloads in a single rocket.
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 replace the feature.
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)]
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
0	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin1A	167.743129	9.047721
1	2	2007-03-21	Falcon 1	NaN	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2A	167.743129	9.047721
2	4	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2C	167.743129	9.047721
3	5	2009-07-13	Falcon 1	200.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin3C	167.743129	9.047721
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003	-80.577366	28.561857
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
89	102	2020-09-03	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	2	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	12	B1060	-80.603956	28.608058
90	103	2020-10-06	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	3	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	13	B1058	-80.603956	28.608058
91	104	2020-10-18	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	6	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	12	B1051	-80.603956	28.608058
92	105	2020-10-24	Falcon 9	15600.0	VLEO	CCSFS SLC 40	True ASDS	3	True	True	True	5e9e3033383ecbb9e534e7cc	5.0	12	B1060	-80.577366	28.561857
93	106	2020-11-05	Falcon 9	3681.0	MEO	CCSFS SLC 40	True ASDS	1	True	False	True	5e9e3032383ecb6bb234e7ca	5.0	8	B1062	-80.577366	28.561857

Data Collection - Scraping

- Using requests we parse the page, which we can then parse to a beautiful soup object.
- We then obtained required data, column names and converted it to a dataframe.
- Github Link: https://github.com/rohitadhikarii/SpaceX-Falcon9-/blob/main/jupyter-labs-webscraping.ipynb



Data Collection – Scraping

```
# use requests.get() method with the provided sta
# assign the response to a object
response = requests.get(static_url)
response = response.text
```

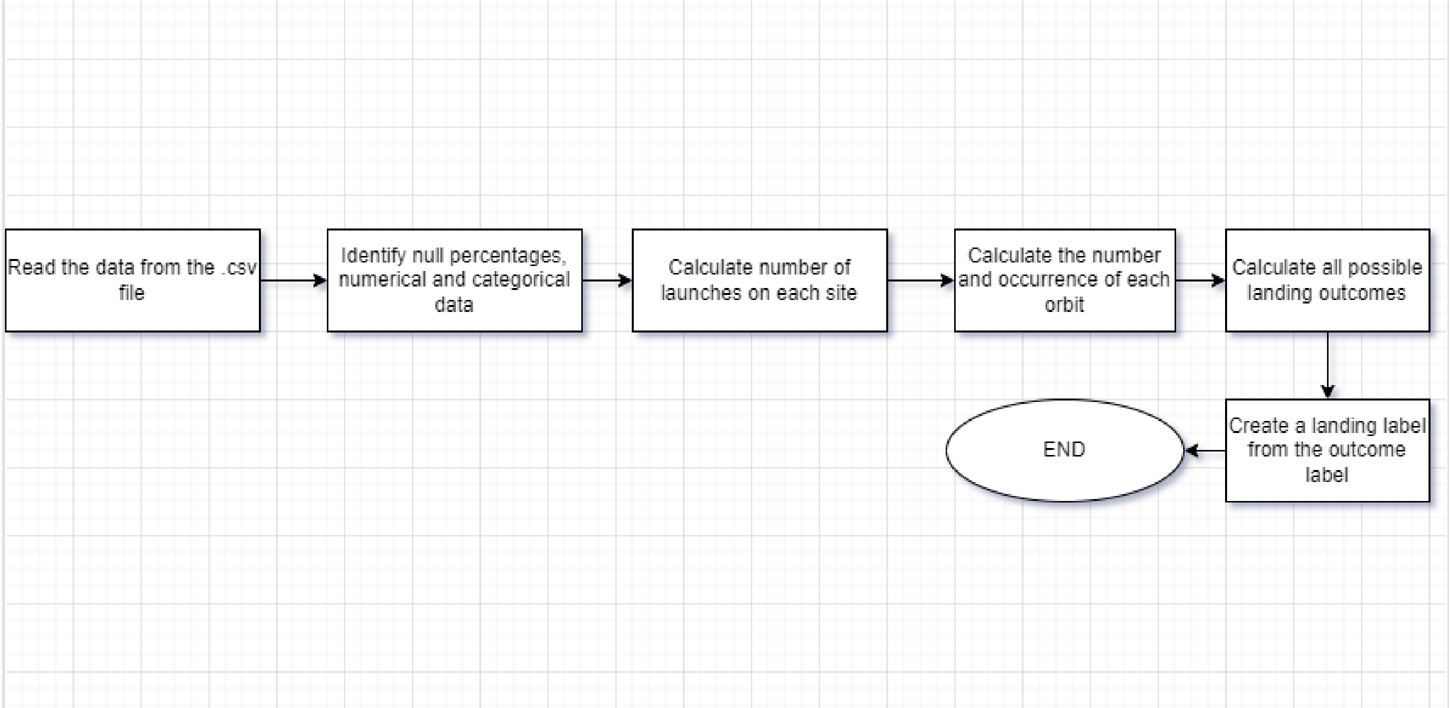
```
# Use BeautifulSoup() to create a BeautifulSoup o
soup = BeautifulSoup(response,'html.parser')
```

```
# Use the find_all function in the BeautifulSoup
# Assign the result to a list called `html_tables
html_tables = soup.find_all('table')
```

```
column_names = []
th_headers = (first_launch_table.find_all('th'))
for th in th_headers:
    each_th = extract_column_from_header(th)
    if th is not None and len(th)> 0:
        column_names.append(each_th)
```

Data Wrangling

- We converted our final outcomes into training labels with constraints: 1 being a success booster land and 0 being a failure.
- Github Link: https://github.com/rohitadhikarii/SpaceX-Falcon9-/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb



```
df.head(5)
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857	0
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857	0
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857	0
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632093	0
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857	0

We can use the following line of code to determine the success rate:

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

```
np.float64(0.6666666666666666)
```

Data Wrangling

```
landing_outcomes = df['Outcome'].value_counts()
```

```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

```
0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 False Ocean
6 None ASDS
7 False RTLS
```

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

```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

```
df['Orbit'].value_counts()
```

```
Orbit
GTO       27
ISS       21
VLEO      14
PO         9
LEO        7
SSO        5
MEO        3
HEO        1
ES-L1      1
SO         1
GEO        1
Name: count, dtype: int64
```

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

```
FlightNumber       0.000000
Date               0.000000
BoosterVersion     0.000000
PayloadMass        0.000000
Orbit              0.000000
LaunchSite         0.000000
Outcome            0.000000
Flights            0.000000
GridFins           0.000000
Reused             0.000000
Legs               0.000000
LandingPad        28.888889
Block              0.000000
ReusedCount        0.000000
Serial             0.000000
Longitude          0.000000
Latitude           0.000000
```

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

```
df['Class']=landing_class
```

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

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

EDA with Data Visualization

Scatter Plots

- Payload and Flight Number
- Flight Number and Launch Site
- Payload and Launch Site
- Flight Number and Orbit Type
- Payload and Orbit Type

Line Graph

- Launch Success Yearly Trend

Bar Graph

- Success Rate vs Orbit Type

Github Link:
https://github.com/rohitadhikarii/SpaceX-Falcon9-/blob/main/edadataviz%20(1).ipynb

EDA with SQL

- We used SQL queries to gather:
  - names of the unique launch sites in the space mission
  - 5 records where launch sites begin with the string 'CCA'
  - the total payload mass carried by boosters launched by NASA (CRS)
  - average payload mass carried by booster version F9 v1.1
  - the date when the first successful landing outcome in ground pad was achieved
  - the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
  - the total number of successful and failure mission outcomes
  - the names of the booster_versions which have carried the maximum payload mass
  - 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/rohitadhikarii/SpaceX-Falcon9-/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- GitHub Link: https://github.com/rohitadhikarii/SpaceX-Falcon9-/blob/main/lab_jupyter_launch_site_location.ipynb

Map Objects	Reason
Map Marker	Creation of marks on the map
Icon Marker	Creation of Icons on the map
Circle Marker	Creation of Circle on the map
PolyLine	Creation of line between points
Marker Cluster Object	Simplify many markers that share similar coordinates

Build a Dashboard with Plotly Dash

GitHub Link: https://github.com/rohitadhikari/SpaceX-Falcon9-/blob/main/spacex_dash_app.py

Functionalities	Reason
Dropdown	A interactive dropdown for launch sites
Rangeslider	Created a slider for the Payload Masses
Scatter Plot	A plot for displaying the correlation
Pie Chart	A chart of Success percentages

Predictive Analysis (Classification)

- Model Build

- Loaded feature engineered data
- Convert into np arrays
- Standardize and transform the data
- Split to Training and Testing data sets
- Gather listed machine learning models
- Set the parameters
- Train the models and fit to GridSearchCv

Model Evaluation

- Gather accuracy insights of each model
- Plot Confusion Matrix
- Find the best hyperparameters for each model

Find the optimal performing classification models amongst:

- K-Nearest Neighbors
- Support Vector Machines
- Logistic Regression
- Decision Trees

GitHub Link: https://github.com/rohitadhikarii/SpaceX-Falcon9-/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

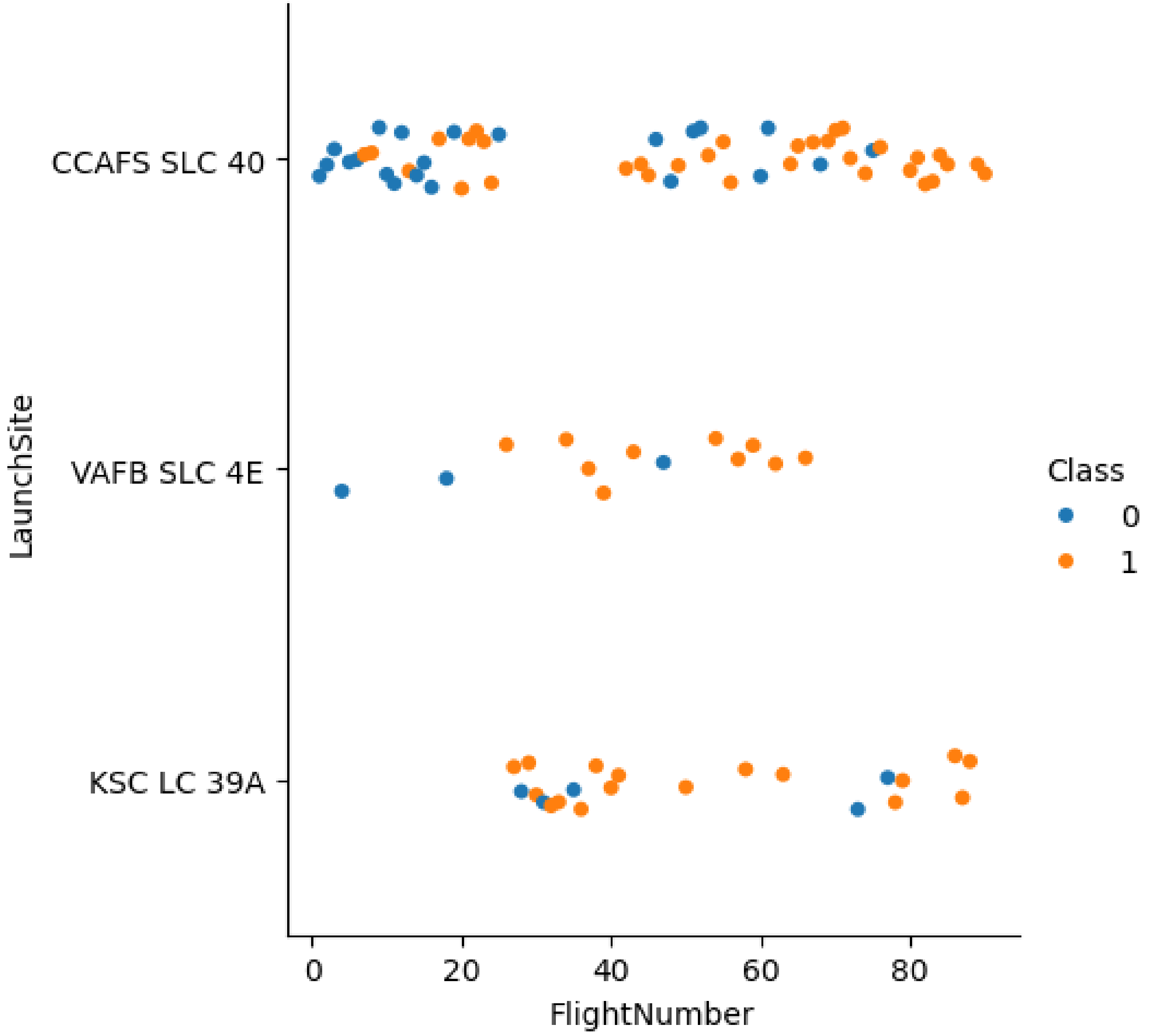
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

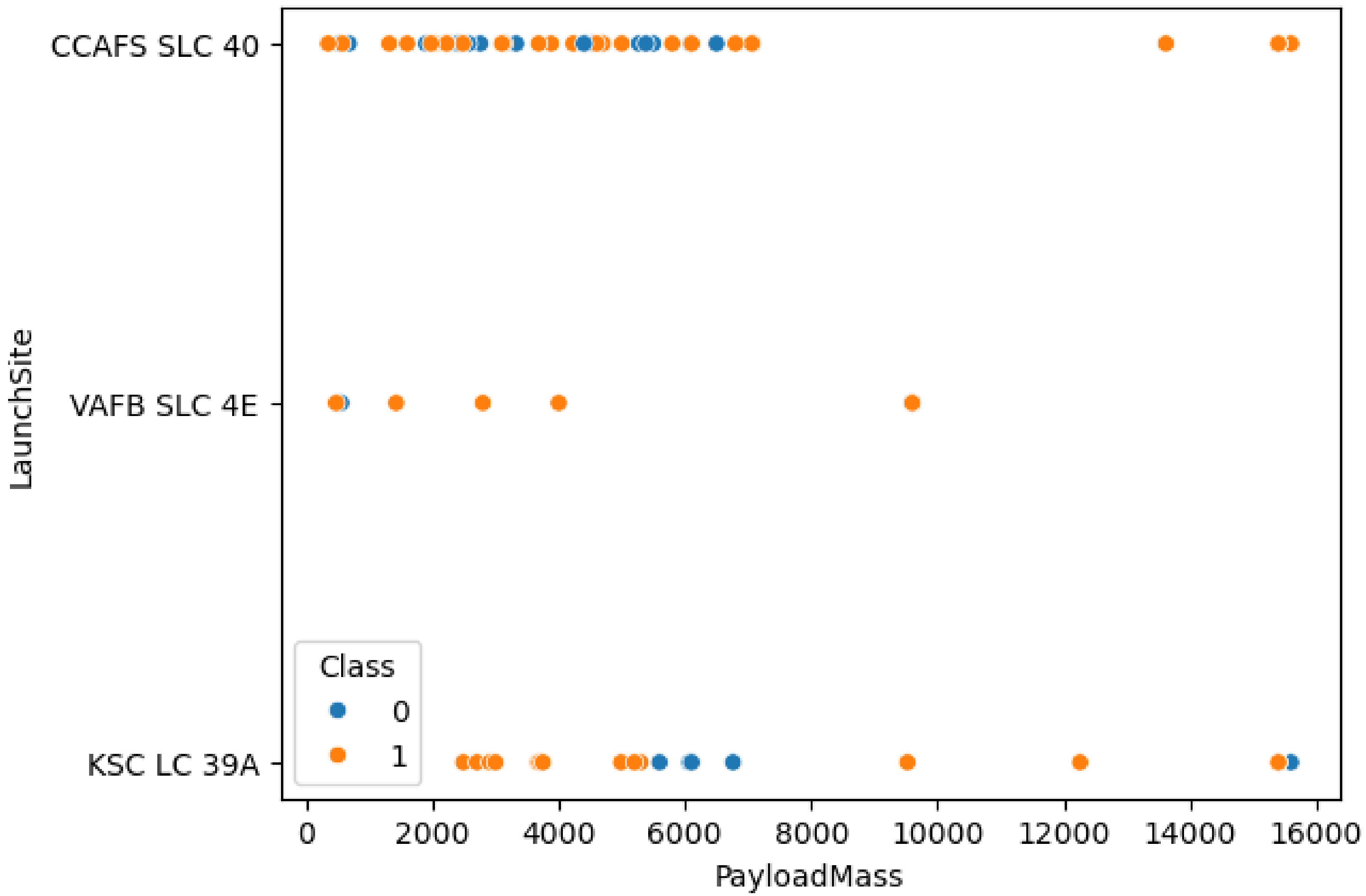
Insights drawn from EDA

Flight Number vs. Launch Site

- We see an increase in class 1 (success landings) as we increase in flight numbers at each LaunchSite. Approx. Flight numbers greater than 30



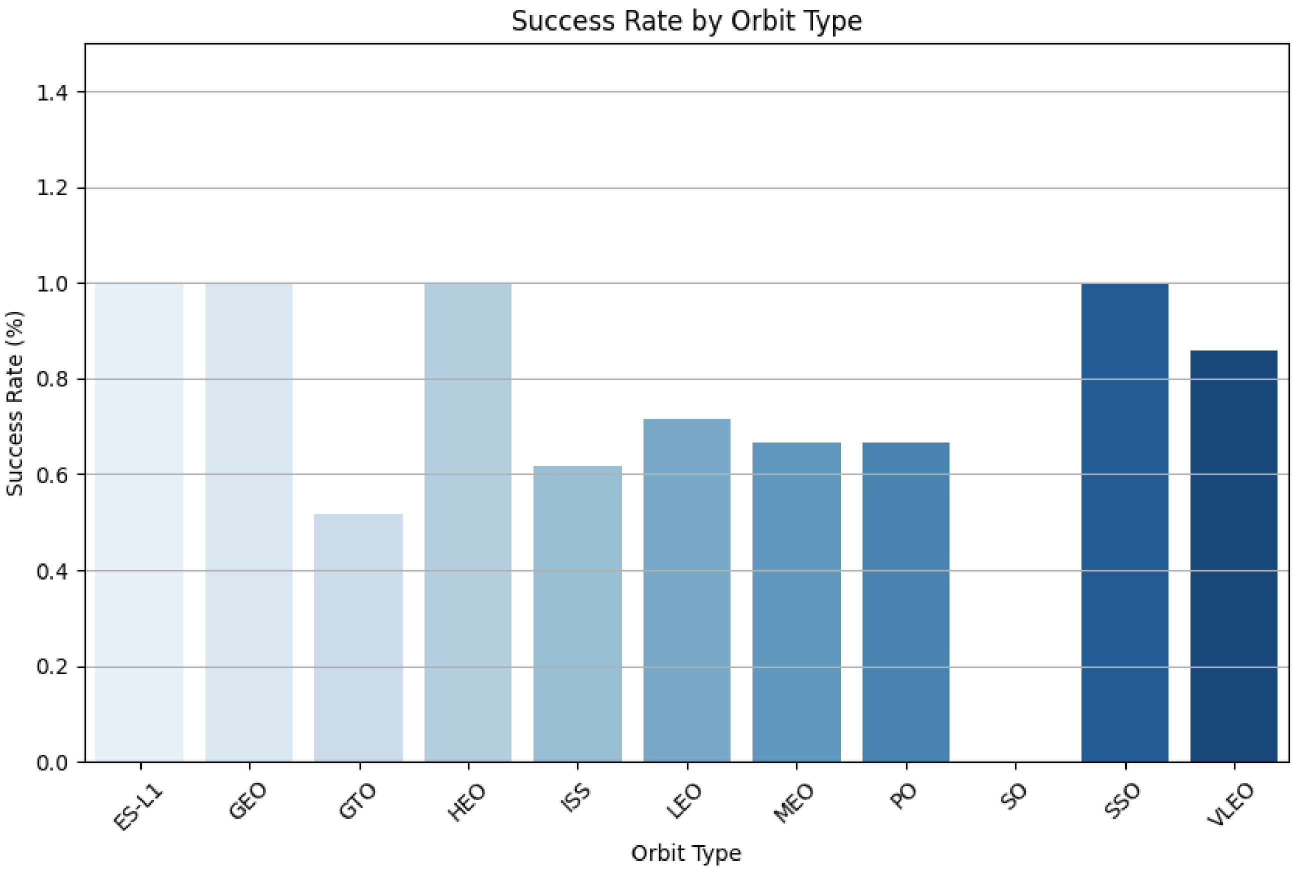
Payload vs. Launch Site



We see that most of the successful findings were when PayloadMass was below 6000 kg

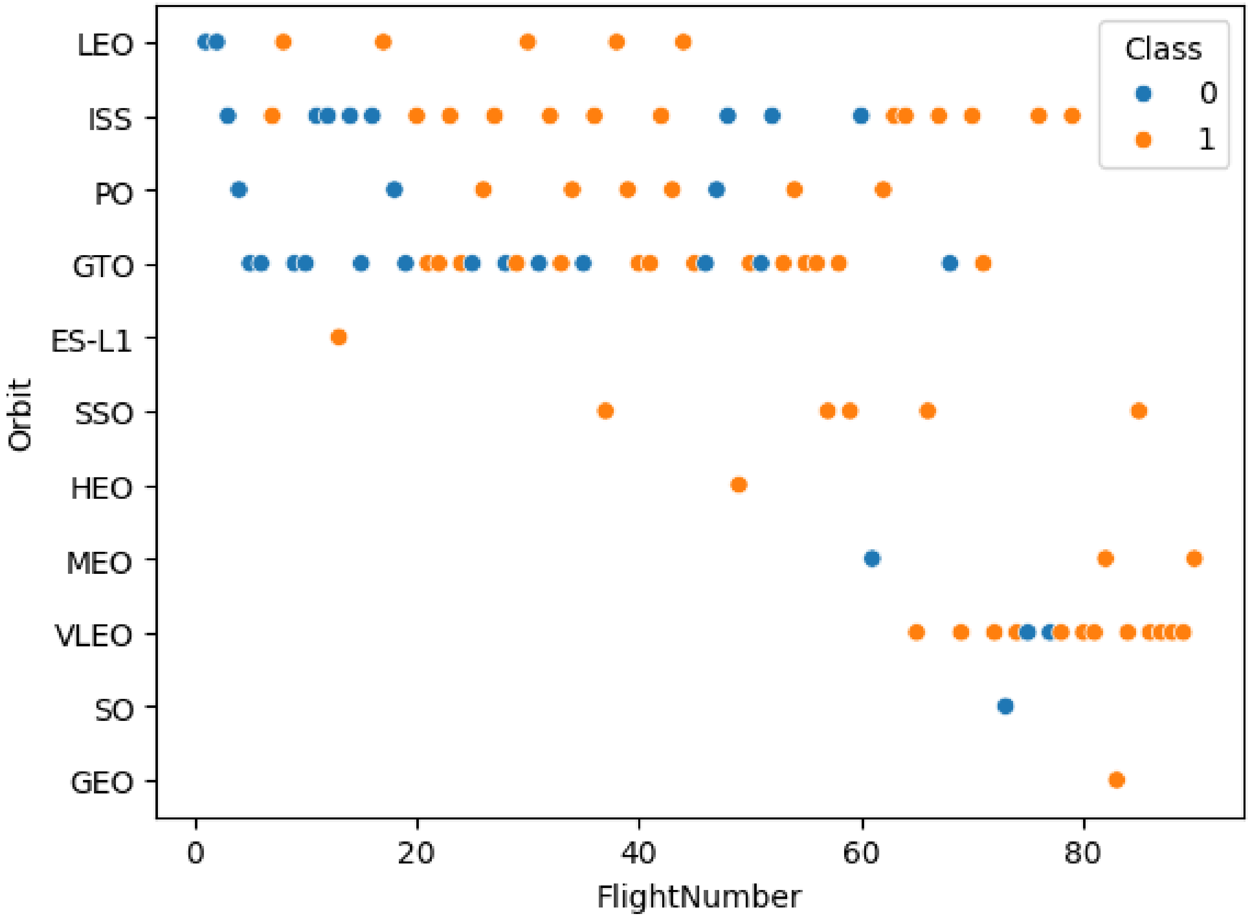
Success Rate vs. Orbit Type

- We see a very high success rate for orbits; ES-L1, GEO, HEO and SSO. Where as SO and GTO none/low success.



Flight Number vs. Orbit Type

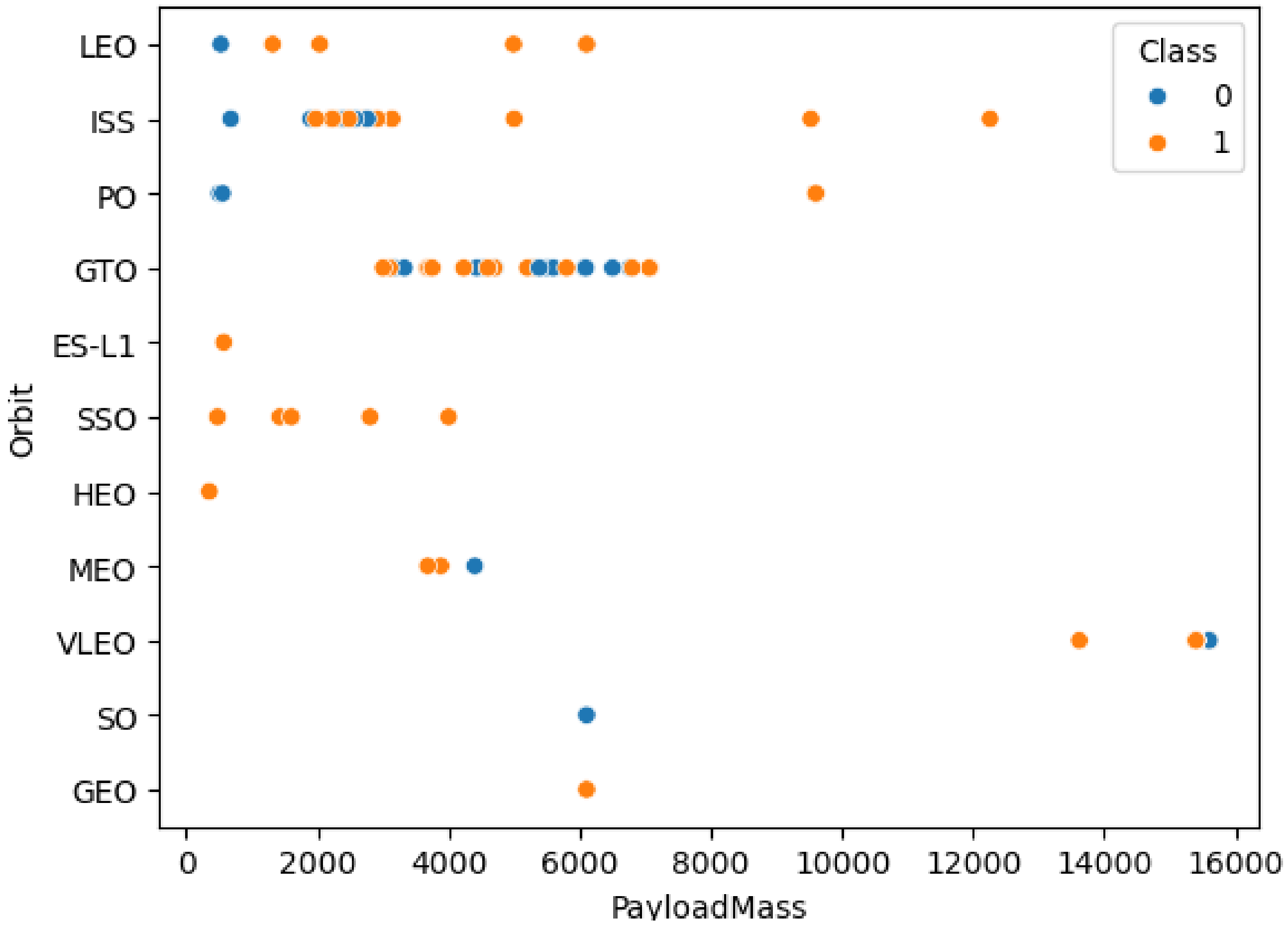
- We see a majority success for flight numbers between 20-50 for orbits LEO, ISS, PO.
- Where as for MEO, VLEO, ISS there is high success for flight number 65 and up.
- Lastly, GEO, ES-L1, SSO and HEO have a 100% success.



Payload vs. Orbit Type

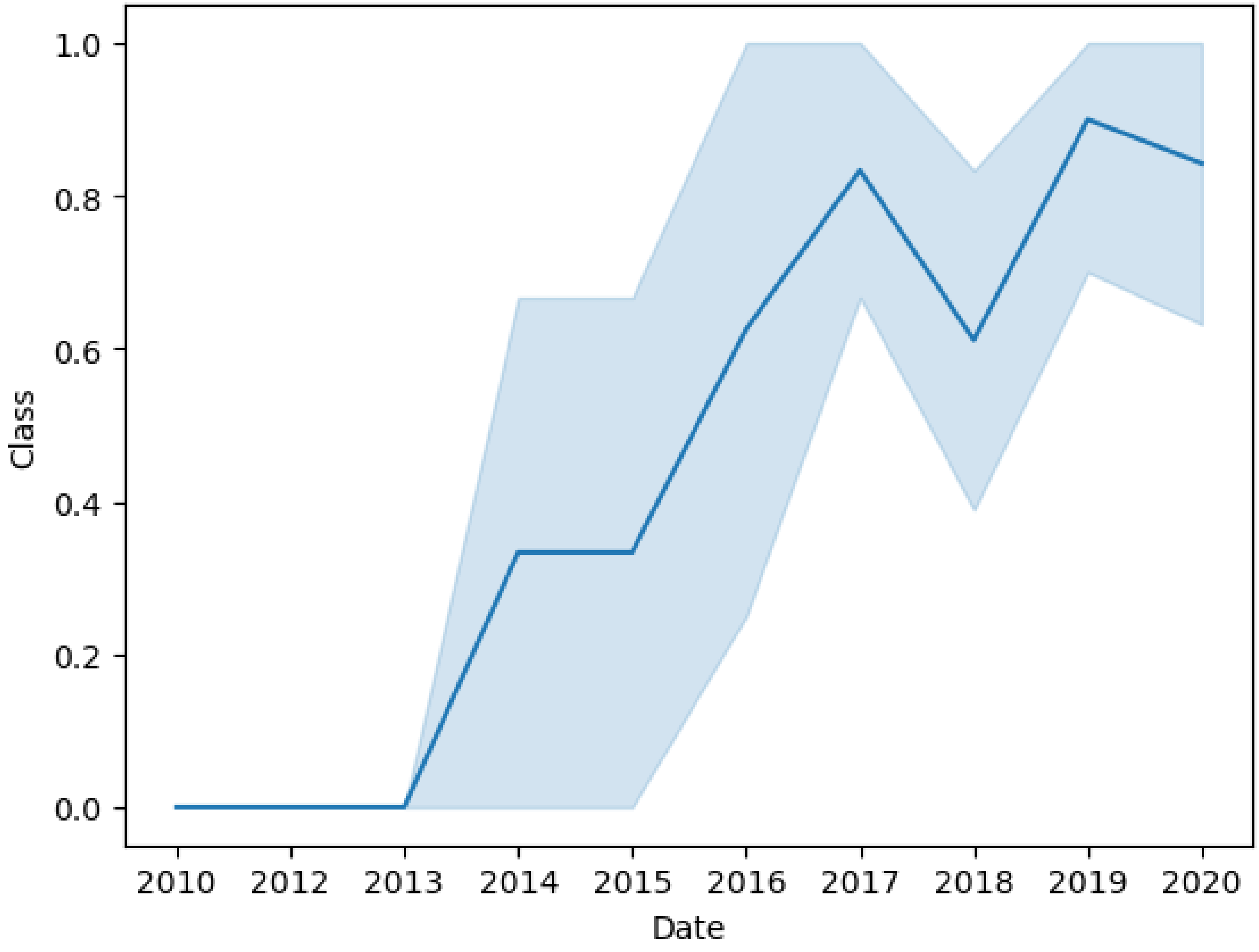
- We see for orbits ISS, PO, VLEO that higher the payloadmass, greater the success rate.

- For ES-L1, SSO,HEO, MEO the lower the payloadmass, greater the success rate.



Launch Success Yearly Trend

We see a positive trend for Dates and Class. The further the years, the higher the success rate.



"All Launch Site Names"

We used DISTINCT keyword to get all the unique values from the SPACEXTABLE

```
%sql select DISTINCT(Launch_Site) from SPACEXTABLE
```

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

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

" Launch Site Names Begin with 'CCA' "

Using the LIKE we can query the names that contain the characters around the '%'

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

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

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"

Using the WHERE clause and SUM we can specify the customer needed

```
%%sql
select Customer, sum(PAYLOAD_MASS__KG_) as total_payload
from SPACEXTABLE
where Customer = 'NASA (CRS)'
```

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

Customer	total_payload
NASA (CRS)	45596

"Average Payload Mass by F9 v1.1"

We use the aggregate AVG function and the WHERE clause to specify our booster version

```
%%sql
select Booster_Version, AVG(PAYLOAD_MASS__KG_) as average_mass_KG
from SPACEXTABLE
where Booster_Version = 'F9 v1.1'
```

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

Booster_Version	average_mass_KG
F9 v1.1	2928.4

"First Successful Ground Landing Date"

Using a subquery and WHERE clause we can retrieve the first successful ground landing

```
%%sql
select *
from SPACEXTABLE
where Date = (select min(Date)
                from SPACEXTABLE
                where Landing_Outcome like '%ground%')
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2015-12-22	1:29:00	F9 FT B1019	CCAFS LC-40	OG2 Mission 2 11 Orbcomm-OG2 satellites	2034	LEO	Orbcomm	Success	Success (ground pad)

"Successful Drone Ship Landing with Payload between 4000 and 6000"

- WHERE clause and BETWEEN operator can fetch results where a successful drone ship landing with payload between the constraints.

```
%%sql
select Booster_Version
from SPACEXTABLE
where Landing_Outcome = 'Success (drone ship)'
    and PAYLOAD_MASS__KG_ between 4001 and 5999
```

```
 * 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"

Using CASE to distinguish LIKE words to group only between Fail and Success and also GROUP BY

```
%%sql
select
    case
        when mission_outcome like '%Success%' then 'Success'
        when mission_outcome like '%Failure%' then 'Failure'
        else 'Other'
    end as Outcome,
    count(*) as total
from spacextable
group by Outcome
```

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

Outcome	total
Failure	1
Success	100

"Boosters Carried Maximum Payload"

Using a Subquery and a WHERE clause we can get the booster with maximum payloads

```
%%sql
select booster_version, PAYLOAD_MASS__KG_ as max_payload
from spacextable
where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_)
                           from spacextable)
```

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

Booster_Version	max_payload
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

"2015 Launch Records"

- Again, Using CASE we can connect names with month numbers. Also the substr function to extract the month/year

```
%%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,
    landing_outcome, booster_version, launch_site
from spacextable
where substr(Date,0,5)= '2015' and landing_outcome like '%Failure%'
```

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

Month	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"

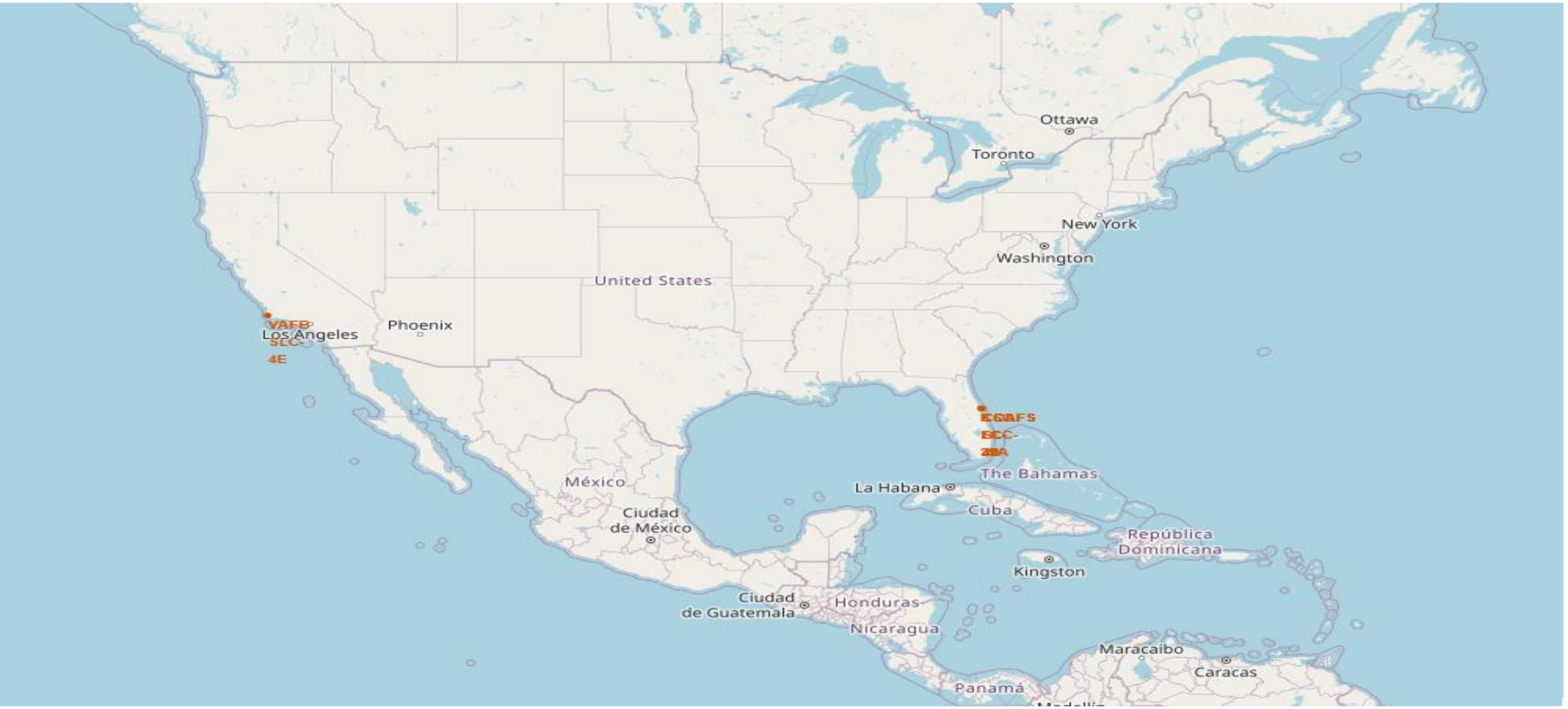
- Using WHERE clause, AND/OR operators, and between to diminish the landing outcomes between June 04 2010 and Match 20 2017

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

Section 3

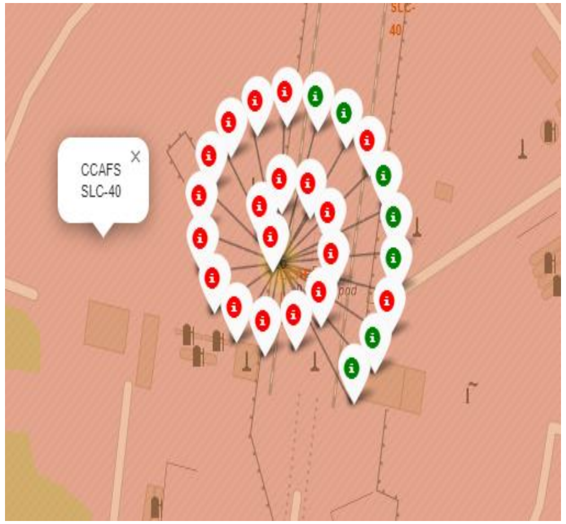
Launch Sites Proximities Analysis

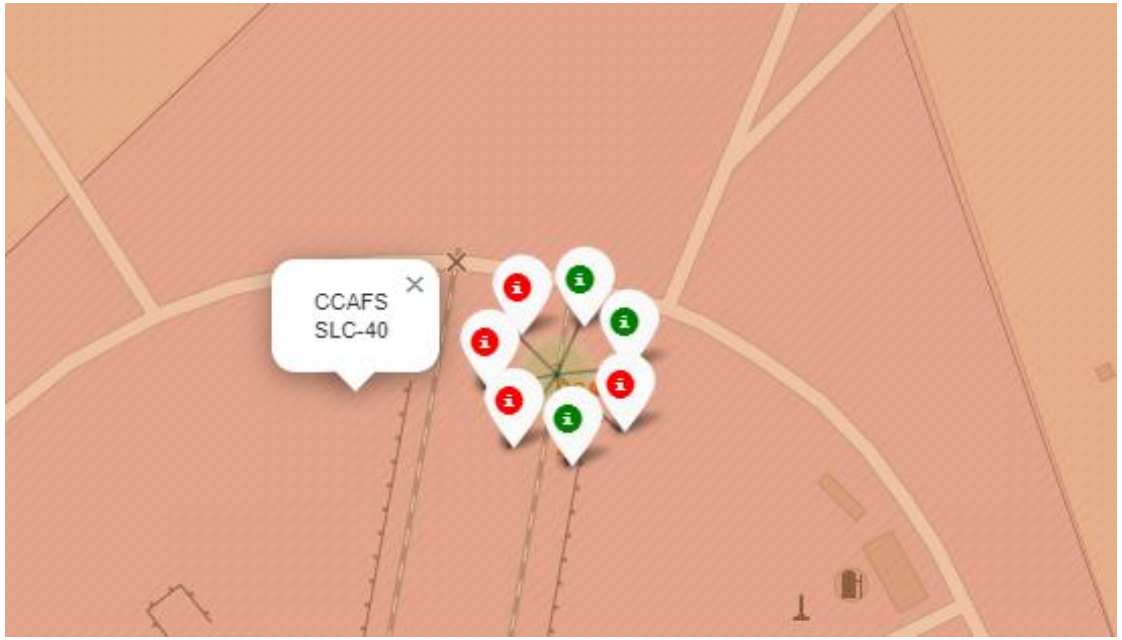
Every Launch Site



Florida Launch Sites

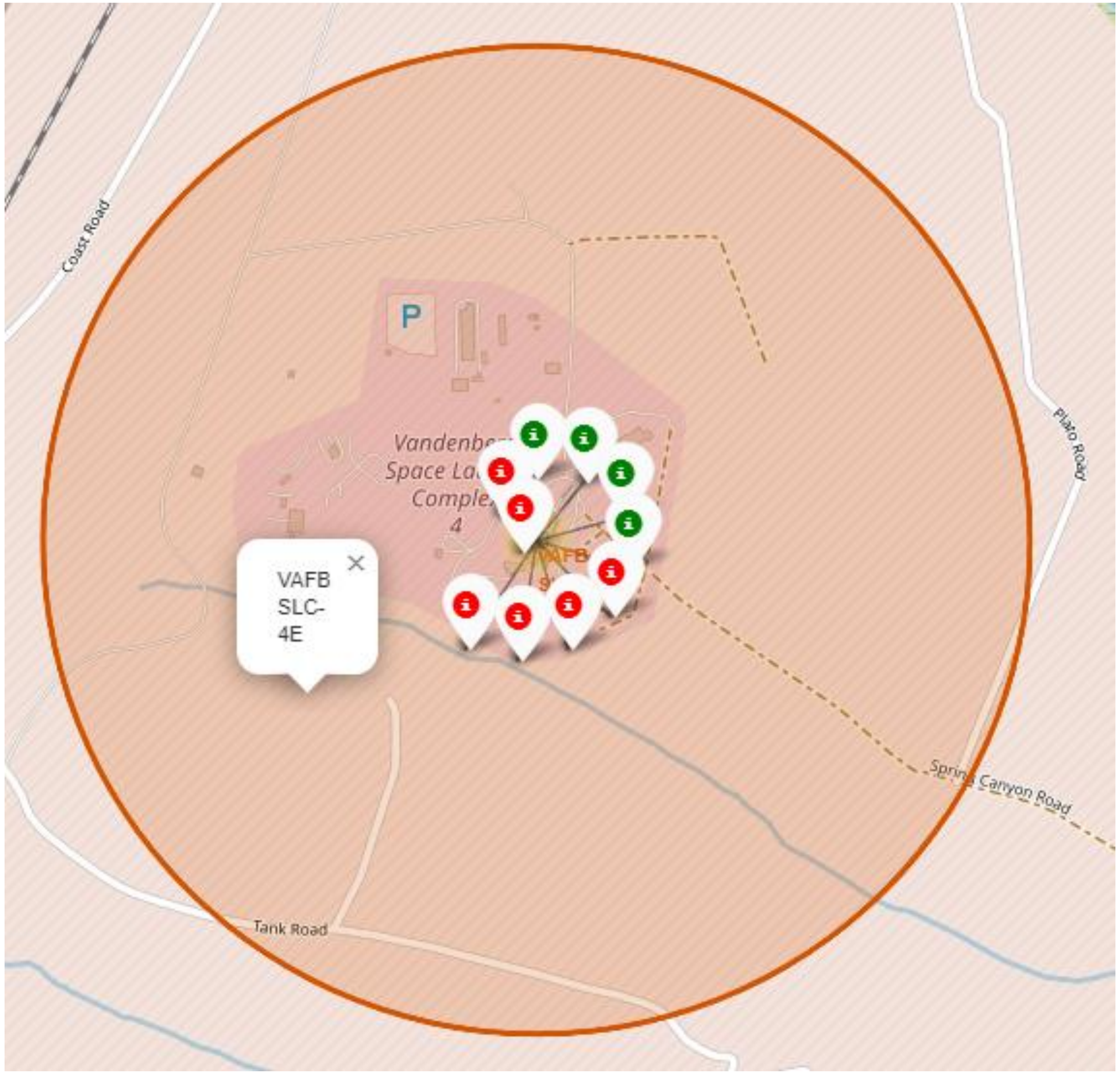
- Green is for successful landings
- Red is for failed landings





California Launch Sites

- Green is for successful landings
- Red is for failed landings



Important Takeaways

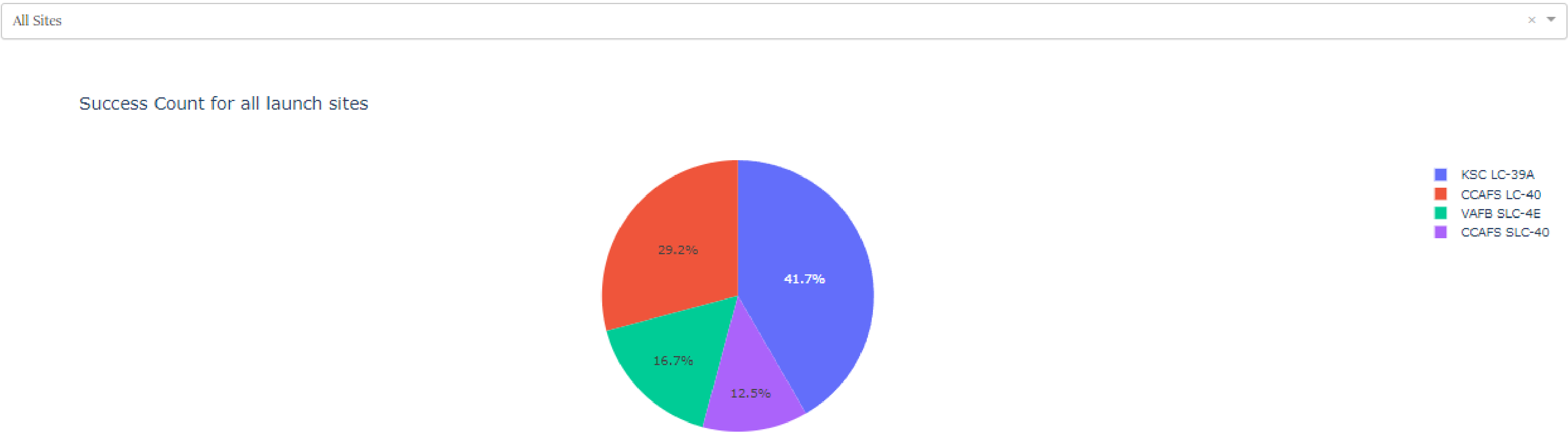
- Are all launch sites in proximity to the Equator Line? **No**
- Are all launch sites in close proximity to railways? **Yes**
- Are all launch sites in close proximity to highways? **No**
- Are all launch sites in close proximity to coastlines? **Yes**
- Do Launch sites have a certain distance away from the cities? **Yes**

Section 4

Build a Dashboard with Plotly Dash

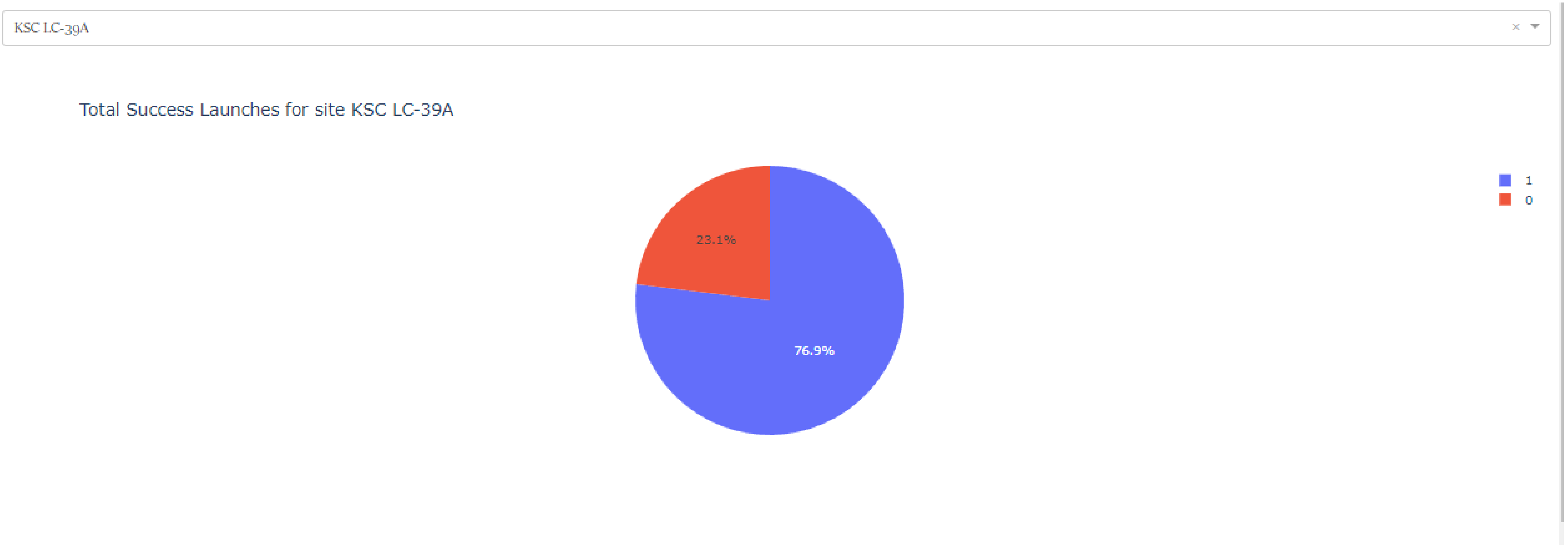
Launch Success for all sites

We see that KSC LC-39A has the highest success count whereas CCAFS SLC-40 has the lowest.



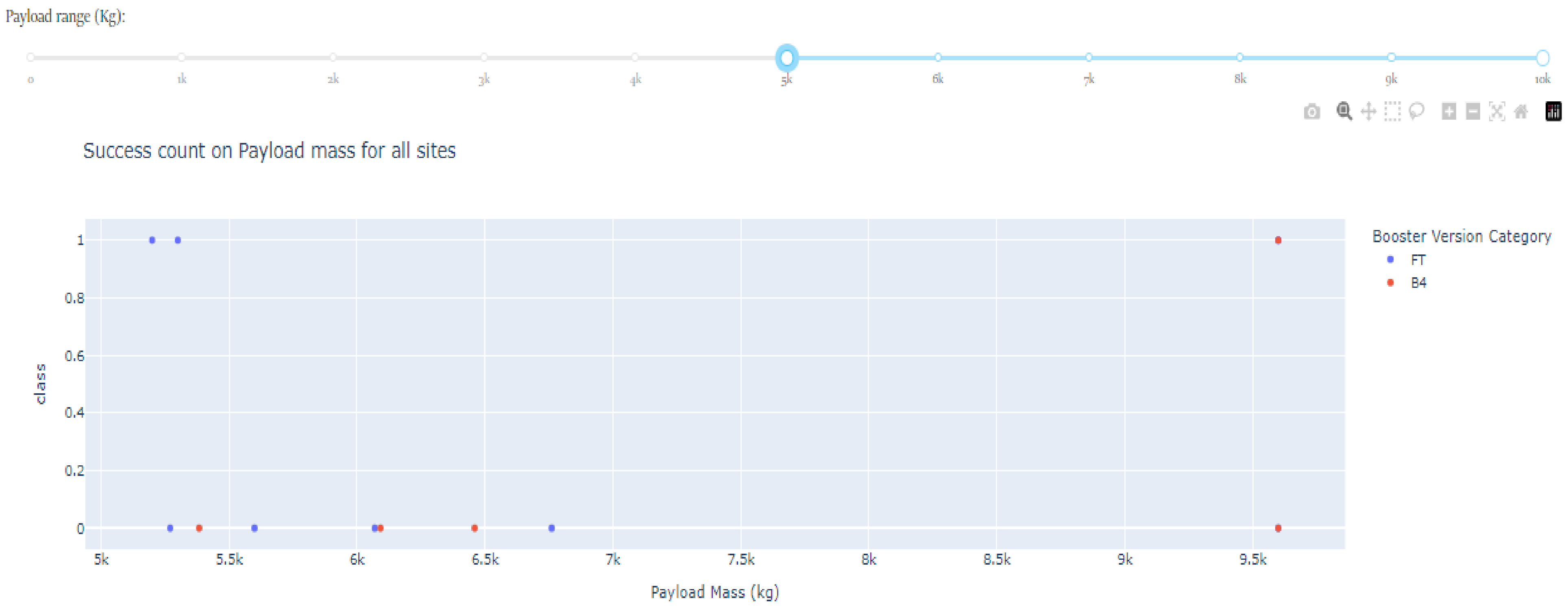
Launch Site with the Highest Success Ratio

KSC LC 39-A has the highest success ratio with a 76.9% success rate and a 23.1% failure rate



Payload vs Launch Outcomes for 0-5000 kg

Payload vs Launch Outcomes for 5000-10000 kg



Takeaways

- There is a clear pattern looking at the two scatter plots. Payload Mass between 0-5k(kg) has higher success rates than Payload Masses between 5k-10k(kg)

Section 5

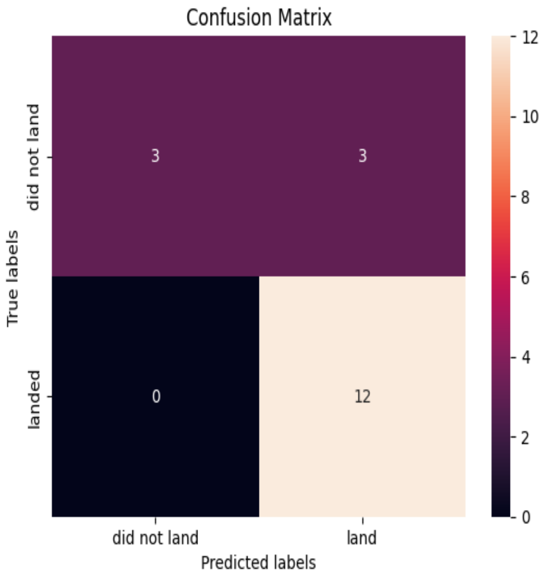
Predictive Analysis (Classification)

Classification Accuracy

Algorithm	Accuracy(best_score_)	Accuracy on the test data	Hyperparameters
Logistic Regression	0.8464285714285713	0.8333333333333334	{'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
Decision Tree	0.8714285714285713	0.6666666666666666	{'criterion': 'gini', 'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 10, 'splitter': 'random'}
Support Vector Machines	0.8482142857142856	0.8333333333333334	{'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
K Nearest Neighbors	0.8482142857142858	0.8333333333333334	{'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'], 'p': [1,2]}

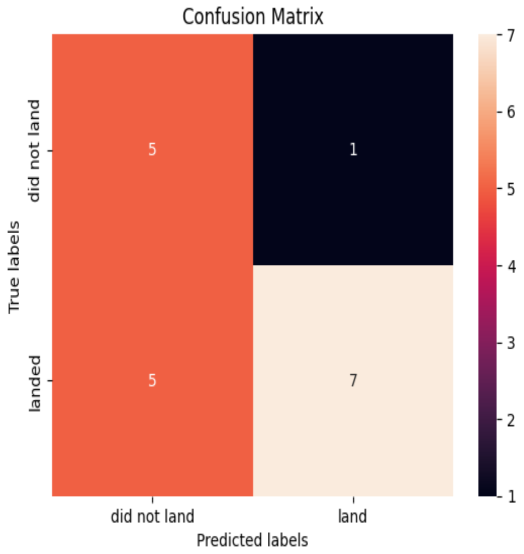
Confusion Matrix

```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



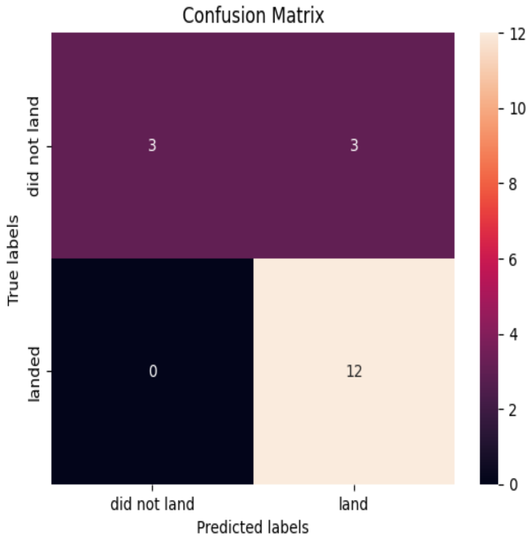
Logistic Regression

```
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



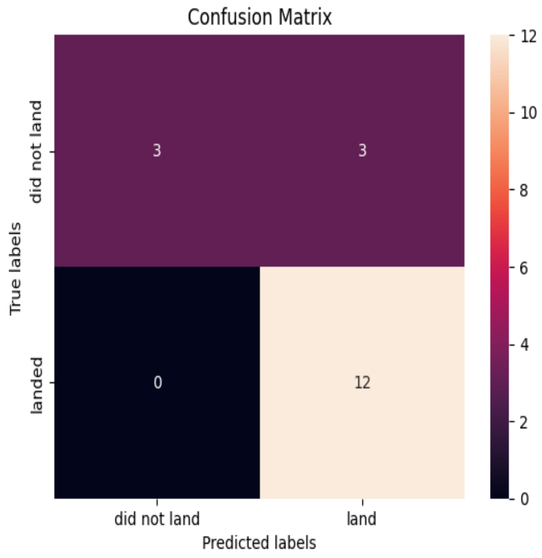
Decision Tree

```
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



K Nearest Neighbors

```
yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Support Vector Machines

Conclusions

- Decision Tree Algorithm had the highest Accuracy on the data test
- Orbits SSO, HEO, GEO and ES-L1 had the highest success rates
- KSC LC 39-A had seen a successful trend but gradually decreased as the payload mass started to increase
- SpaceX's success rates have been increasing as we progress more and more into the years making it susceptible to breaking more records