Winning Space Race with Data Science

Daniela Knoll

Feb 2024

Outline

Executive Summary

Summary of methodologies

-Data collection

-Data Wrangling

-EDA with Data visualisation

-EDA with SQL

-Building an interactive map with Folium

-Building an interactive dashboard with plotly-dash

-Predictive analysis

Summary of all results

-Exploratory data analysis results

-Interactive analytics

-Predictive analysis results

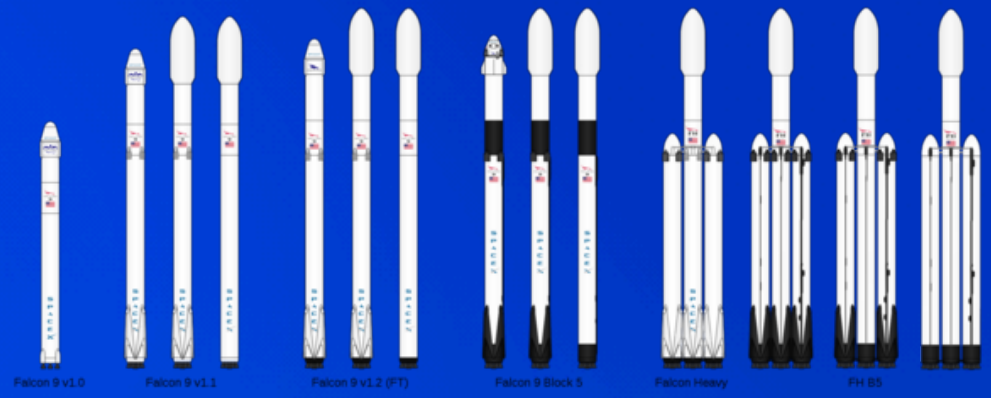
Project Background

Falcon 9 (Figure 2-1) is a two-stage launch vehicle powered by liquid oxygen (LOX) and rocket-grade kerosene (RP-1).

The vehicle is designed, built and operated by SpaceX.

Falcon 9 can be flown with a fairing or with a SpaceX Dragon spacecraft.

All first- and second-stage vehicle systems are the same in the two configurations; only the payload interface to the second stage changes between the fairing and Dragon configurations.



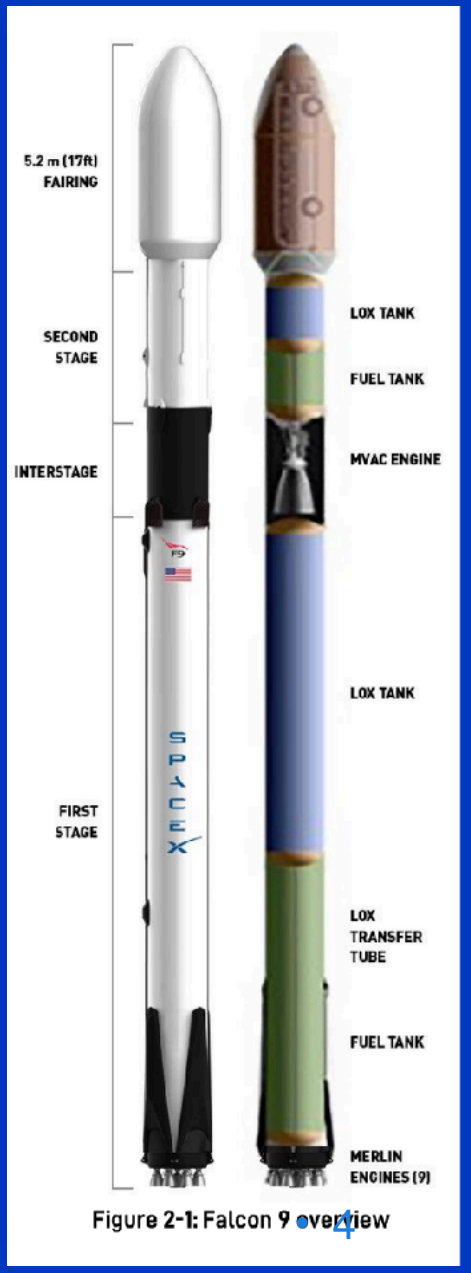
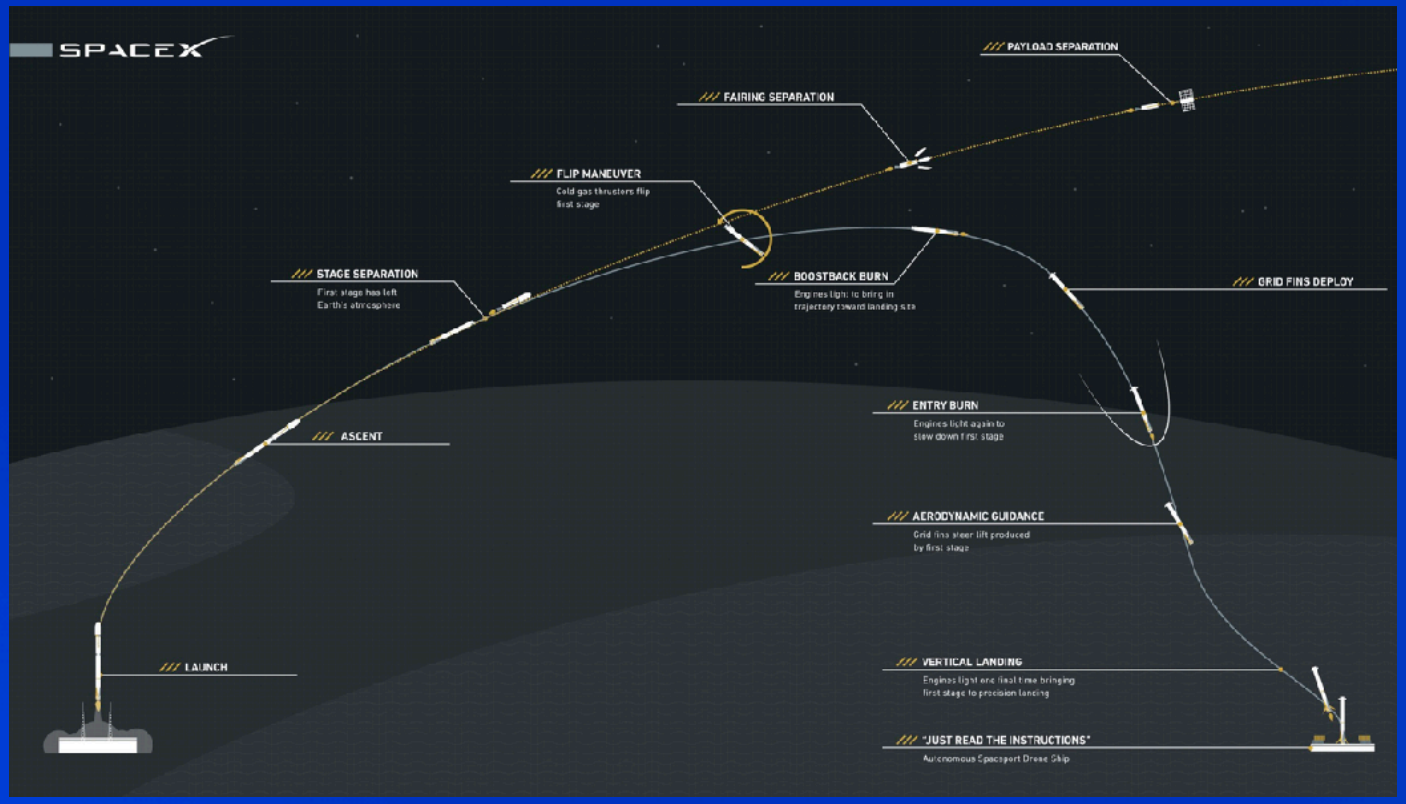


Figure 2-1: Falcon 9 overview

Introduction

SpaceX advertises Falcon 9 rocket launches with a comparatively low cost of 62 million dollars due to their capability to reuse the first-stage.

Other providers cost more than 165 million dollars per launch, and much of SpaceX savings are due to SpaceX's capability to reuse the first-stage



Falcon 9 sample mission profile (SpaceX)

This data science project will determine the cost of a launch, based on the predictions whether the first-stage has crashed, landed or sacrificed due to mission parameters such as launch site, orbit or payload.

Section 1

Methodology

Methodology

Executive Summary

Data collection methodology:

Collected data from SpaceX and Wikipedia using API and Web Scraping

https://en.wikipedia.org/wiki/SpaceX_reusable_launch_system_development_program#Economics_of_rocket_reuse

Perform data wrangling :

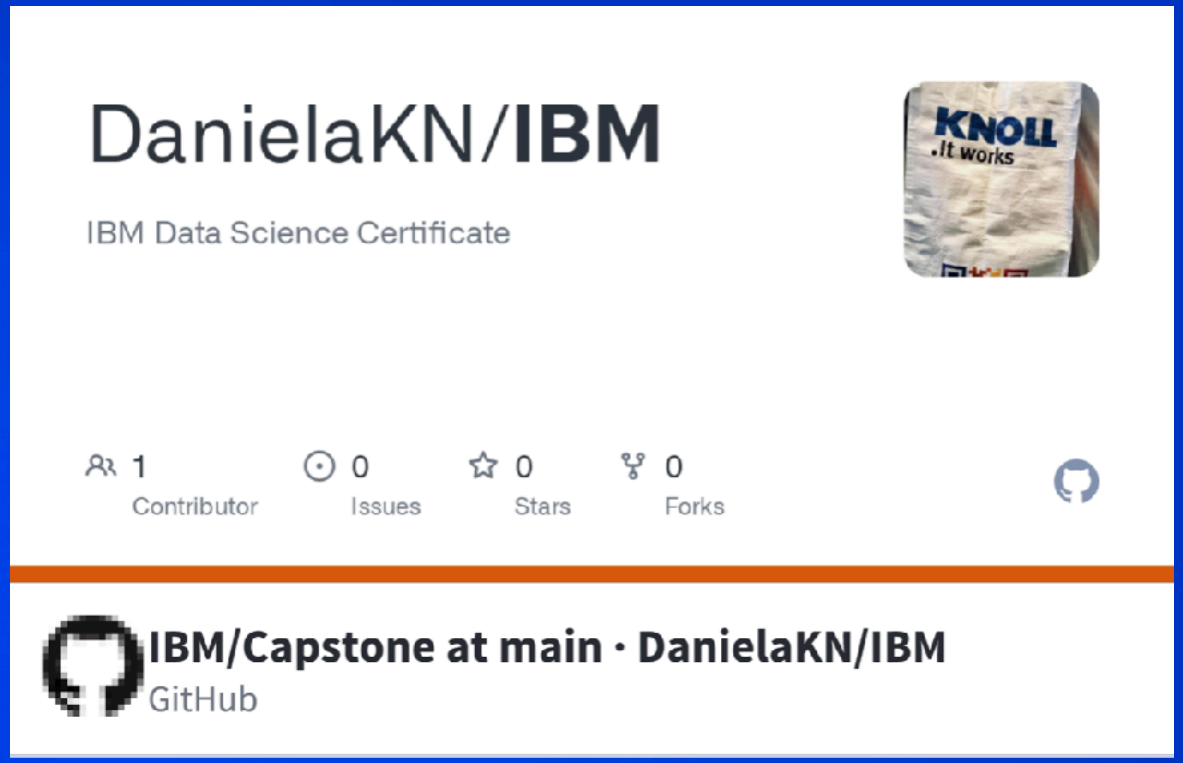
Improve data quality by dealing with missing values

Perform exploratory data analysis (EDA) using visualisation and SQL

Perform interactive visual analytics using Folium and Plotly Dash

My GitHub Link

All my Jupyter Lab Notebooks are available on my IBM Capstone Repo



Data Collection

Two data sources using the request.get() method

1. SpaceX open source API

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url)
```

2. Wikipedia

“List of Falcon 9 and Falcon Heavy Launches” (updated June, 9th 2021)Wikipedia

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

Data Collection - SpaceX API

Step 1

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

Step 2

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

Step 3

From 'rocket' we would like to learn the booster name
From 'payload' we would like to learn the mass of the payload and the orbit that it is going to
From 'launchpad' we would like to know the name of the launch site being used, the longitude, and the latitude.
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, whether the core is reused, whether 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.

The data from these requests will be stored in lists and will be used to create a new dataframe.

Step 4

```
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}
```

Step 5

```
data_falcon9 = data[data['BoosterVersion']!='Falcon 1']
```

Step 6

```
# Calculate the mean value of PayloadMass column
mean_payload_mass = data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, mean_payload_mass
```

Step 7

```
data = pd.DataFrame({key:pd.Series(value) for key, value in launch_dict.items()})
```

Data Collection - Scraping

Step 1

```
static_url =
"https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

Step 2

```
soup = BeautifulSoup(response.text, "lxml")
```

Step 3

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

Step 4

```
launch_dict= dict.fromkeys(column_names)
del launch_dict['Date and time ( )']

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'] = []

launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

Step 5

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

Data Wrangling

Step 1

```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv")
```

Step 2

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

Step 3

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

Step 4

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

Step 5

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

Step 6

0 means the first stage did not land successfully, 1 means it did land successfully
True ASDS(41) mission successfully landed to a drone ship
None None(19) failure to land
True RTLS(14) mission successfully landed on a ground pad
False ASDS(6) mission unsuccessfully landed to a drone ship
True Ocean(5) mission successfully landed in a specific region in the ocean
False Ocean(2) mission unsuccessfully landed in a specific region in the ocean
None ASDS(2) failure to land
False RTLS(1)mission unsuccessfully landed on a ground pad

Step 7

```
landing_class = []
for key,value in df["Outcome"].items():
    if value in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

Step 8

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

EDA with Data Visualisation

The following charts were plotted:

- Flight Number vs Payload Mass, overlaid by Launch Outcome; as the flight number increases, the first stage is more likely to land successfully. The payload mass is also important as the more massive the payload, the less likely the first stage to return.
- Flight Number vs Launch Site, overlaid by the Launch Outcome; as the flight number increases, the first stage is more likely to land successfully.
- Payload and Launch Site; in the VAFB-SLC launch site there are no rockets launched for heavy payload mass (more than 10000).
- Success rate by orbit type; ES-L1, GEO, HEO, SSO, and VLEO have high success rate.
- Flight Number and Orbit type; in the Leo orbit the success is related to the number of flights. There is no relationship between flight number and the GTO orbit.
- Payload and Orbit; Polar, LEO and ISS have higher success rates with heavier payloads.
- Launch success yearly trend; success rate keeps increasing since 2013.

https://github.com/DanielaKN/IBM/blob/main/Capstone/jupyter-labs-eda-dataviz.ipynb

EDA with SQL - page 1

- List the number of success/failure mission outcomes:

  SELECT mission_outcome, COUNT(mission_outcome) Count from SPACEXTBL GROUP BY mission_outcome;

- List the names of the booster_versions carrying the maximum payload mass:

  SELECT booster_version FROM spacextbl

  WHERE payload_mass__kg_ = (SELECT MAX(payload_mass__kg_) FROM spacextbl);

- List the records to display the month, failure landing outcomes in drone ship, booster versions, launch site for the months in 2015:

  SELECT substr(Date 4,2) as month, “Landing_Outcome”, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL

  WHERE “Landing_Outcome” = ‘Failure(drone ship)’ AND substr(Date, 7,4) = ‘2015’;

- Rank the count of successful landing_outcome between 04-06-2010 and 20-03-2017 in descending order:

  SELECT “Landing_Outcome”, COUNT(“Landing_Outcome”) AS COUNT_LAUNCHES, (substr(Date, 7,4)||’-‘|| substr(Date,4,2)||’_’|| (Date,1,2)) as my DATE FROM SPACEXTBL

  WHERE my DATE BETWEEN ‘2010-06-04’ AND ‘2017-03-20’ GROUP BY “Landing_Outcome” ORDER BY COUNT_LAUNCHES DESC;

EDA with SQL - page 2

- List total number of success/failure mission outcomes:

  SELECT mission_outcome, COUNT(mission_outcome) Count from SPACEXTBL GROUP BY mission_outcome;

- List booster_versions with maximum payload mass(subquery):

  SELECT booster_version FROM spacextbl WHERE payload_mass__kg_ = (SELECT MAX(payload_mass__kg_) FROM SPACEXTBL;

- List the records with month names, failure landing_outcomes in drone ship, booster versions, launch_site for 2015.

  SELECT substr(Date, 4,2) as month, “Landing_OUtcome”, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE “Landing_OUtcome”= ‘Failure(drone ship)’ ANS substr(Date,7,4) = 2015;

  Rank count of successful landing_outcomes between 04-06-2010 and 20-03-2017 in descending order;

  SELECT “Landing_Outcome”, COUNT(“Landing_Outcome”) AS COUNT_LAUNCHES, (substr(Date,7,4) ||’-‘|| substr(Date,4,2)||’-‘|| substr(Date,1,2)) as myDATE

  FROM SPACEXTBL WHERE my DATE BETWEEN ‘2010-06-04’ AND ‘2017-03-20’ GROUP BY “Landing_Outcome” ORDER BY COUNT_LAUNCHES DESC;

https://github.com/DanielaKN/IBM/blob/main/Capstone/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

Created map objects and added them to a Folium map:

- Created markers to indicate points like launch sites.
- Created circles to highlight areas around coordinates such as the NASA Johnson Space Center.
- Used lines to indicate distances between two coordinates.
- Created marker clusters to indicate group of events in each coordinate such as launches in a specific launch site.

https://github.com/DanielaKN/IBM/blob/main/Capstone/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly-Dash

- This dashboard application contains input components such as dropdown list, a range slider that creates an interactive pie chart and a scatter point chart.

  Launch Site Drop-down Input Component for launch site selection:

  It shows which one has the largest success count. By selecting a specific site, it shows a detailed success rate (class 0 = failure vs. Class 1= success).

- Callback function:

  It allows rendering of success-pie-chart based on the selected site dropdown; it gets a selected launch site from the site-dropdown and renders a pie chart visualising launch success counts.

- Range Slider to Select Payload:

  Allows easy selection of different payload range in order to see if we can identify visual patterns (whether payload is correlated with mission outcome).

- Callback function for rendering success-payload-scatter-chart plot:

  It shows how payload is correlated with mission outcomes for selected sites. Color label the Booster version on each scatter point to observe mission outcomes with different boosters.

https://github.com/DanielaKN/IBM/blob/main/Capstone/Dashboard.ipynb

Predictive Analysis (Classification)

- Imported the relevant libraries, created a function to plot the confusion matrix and loaded the data frame.
- Created a NumPy array (from column Class in the data) and assigned it to variable Y.
- Standardised the data in X and reassigned it to the variable X.
- Used the function train_test_split to split the data X and Y into training and testing (test_size = 0.2 and random_state = 2).
- Created a logistic regression object.
- Created a GridSearchCV object logreg_cv with cv = 10.
- Fitted the object to find the best parameters from the parameter dictionary.
- Outputted GridSearchCV object for logistic regression, displayed the best parameters using:

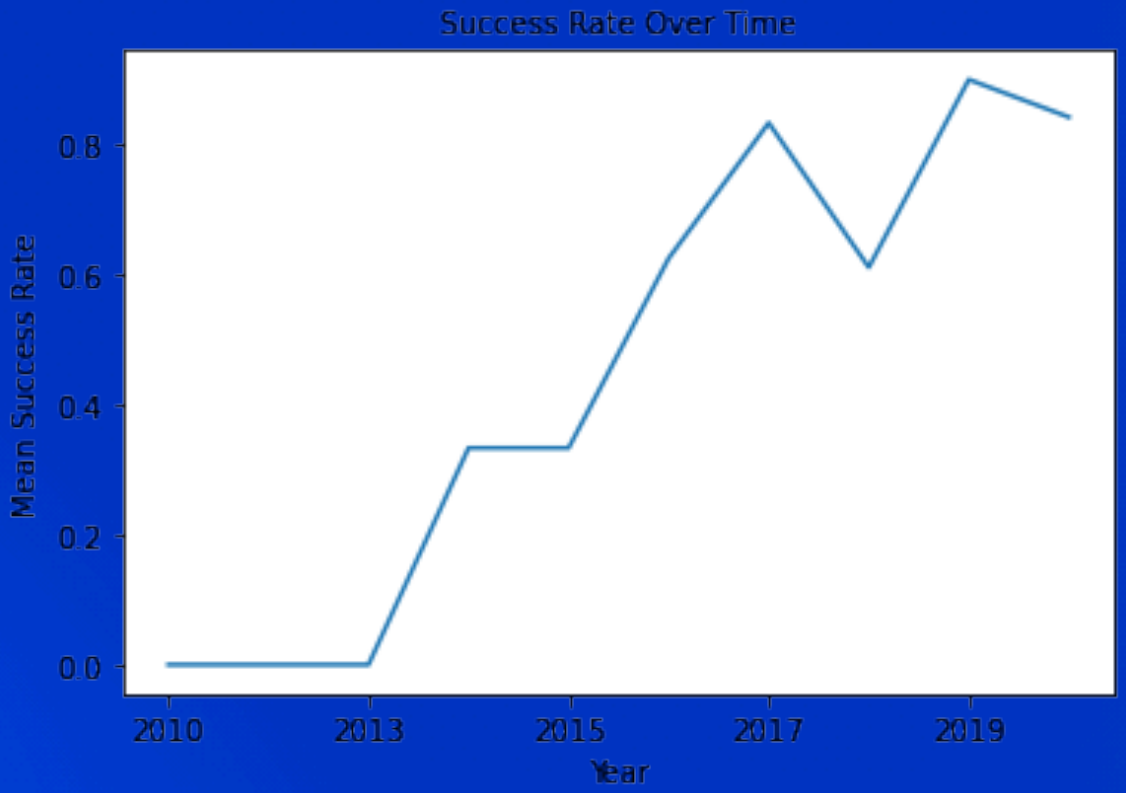
  best_params and looking at the accuracy on the attribute best_score
- Calculated the accuracy on the test data using the method score.
- Repeated the process for SVM, Decision Tree Classifier and KNN.

https://github.com/DanielaKN/IBM/blob/main/Capstone/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

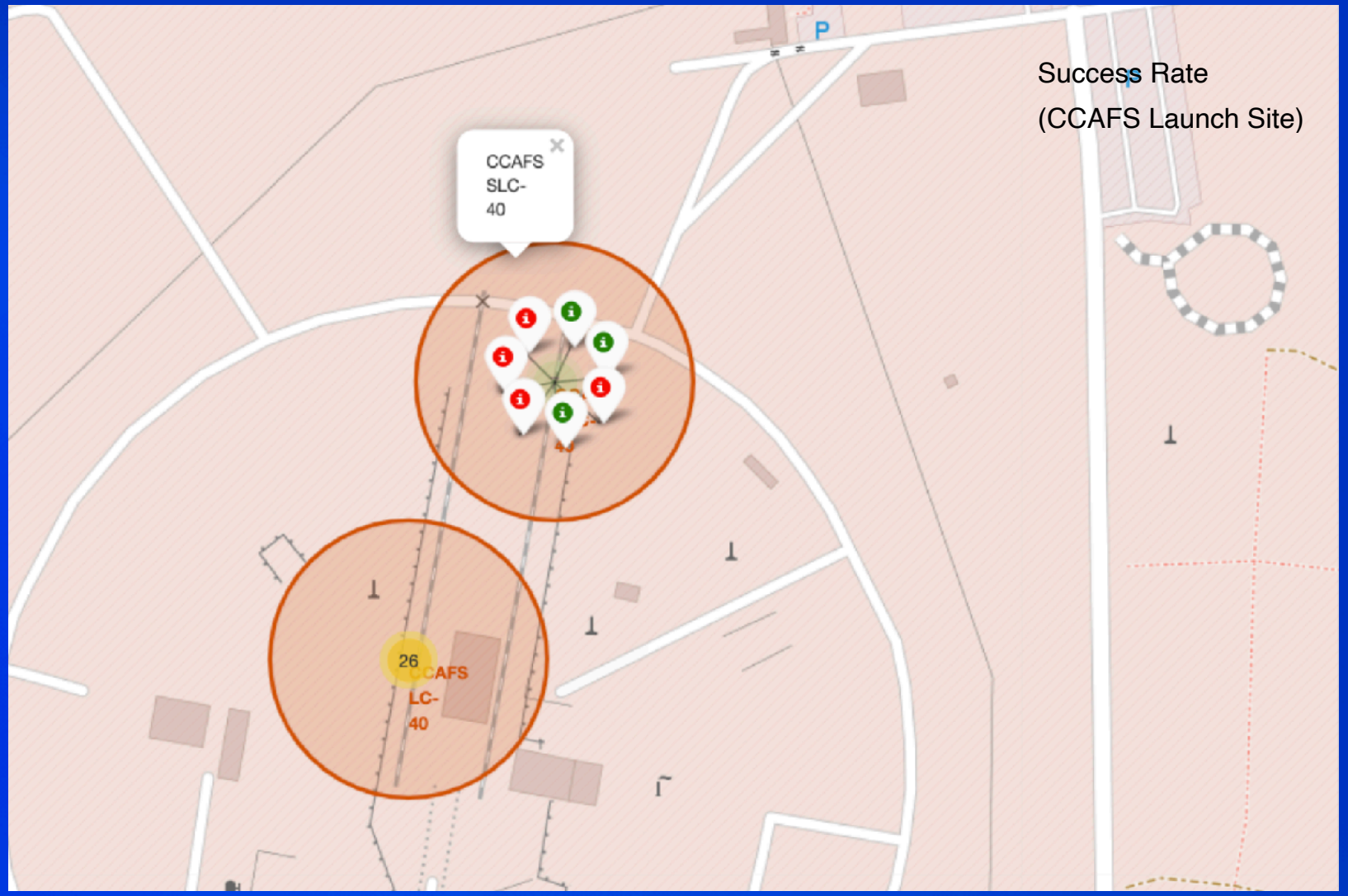
Results from Exploratory Data Analysis

Count of successful landing_outcomes between 04-06-2010 and 20-03-2017(des)

Landing Outcome	Launch Count	Date
No attempt	10	2017-03-16
Failure(drone-ship)	5	2016-06-15
Success (drone-ship)	5	2017-01-14
Controlled(ocean)	3	2015-02-11
Success (grounded)	3	2017-02-19
Failure(parachute)	2	2010-12-08
Uncontrolled(ocean)	2	2014-09-21
Precluded (drone-ship)	1	2015-06-28



Results of Interactive Demo



Results of Predictive Analysis

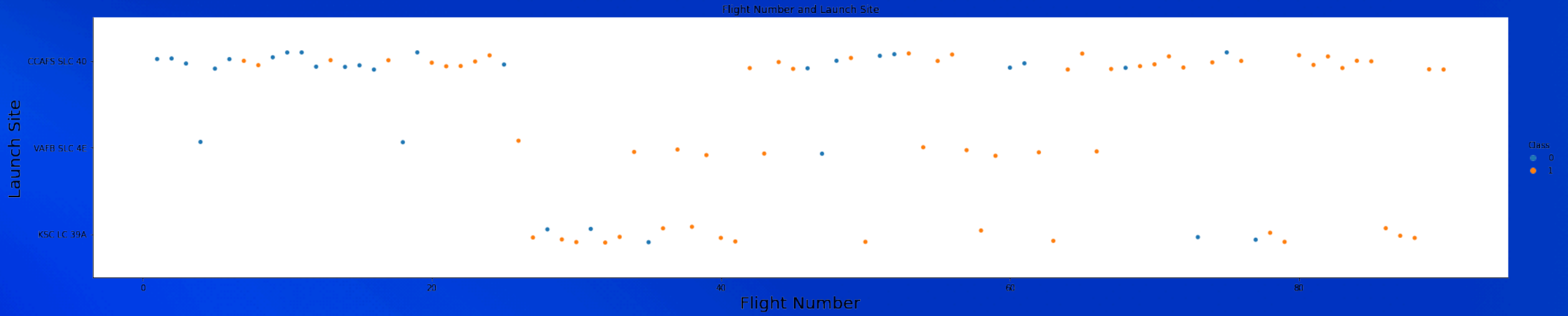
Model	Test data Accuracy	Best Parameters
Log Reg	0.88889	{'C': 0.1, 'penalty': 'l2', 'solver': 'lbfgs'}
SVM	0.66667	{'C': 0.001, 'gamma': 0.001, 'kernel': 'rbf'}
Tree	1.0	{'criterion': 'entropy', 'max_depth': 8, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2, 'splitter': 'best'}
KNN	0.61111	{'algorithm': 'auto', 'n_neighbors': 3, 'p': 1}

As you can see from the table the best model was the Tree model with 0.8339 test data accuracy.

Section 2

Insights drawn from EDA

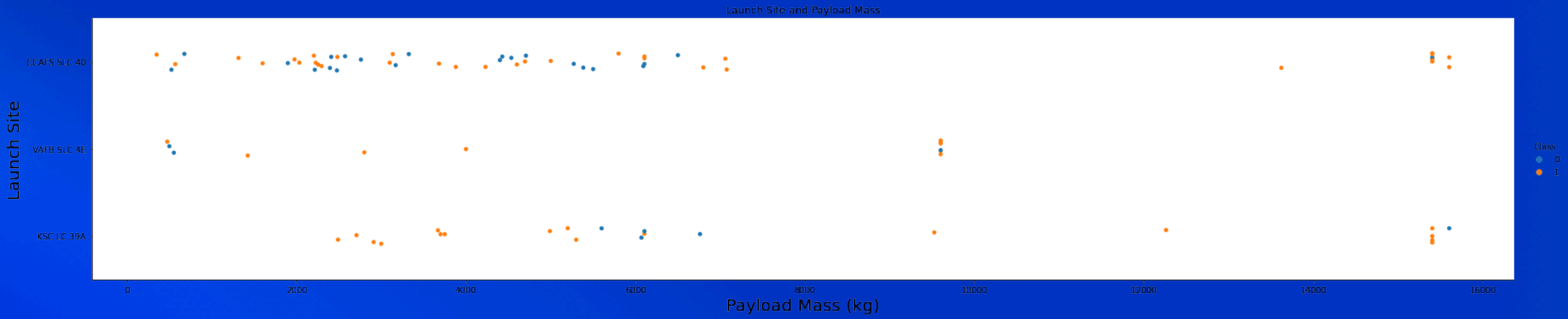
Flight Number vs. Launch Site



Explanation:

As the number of of flights increase on each launch site, the first stage is more likely to land successfully.

Payload vs. Launch Site

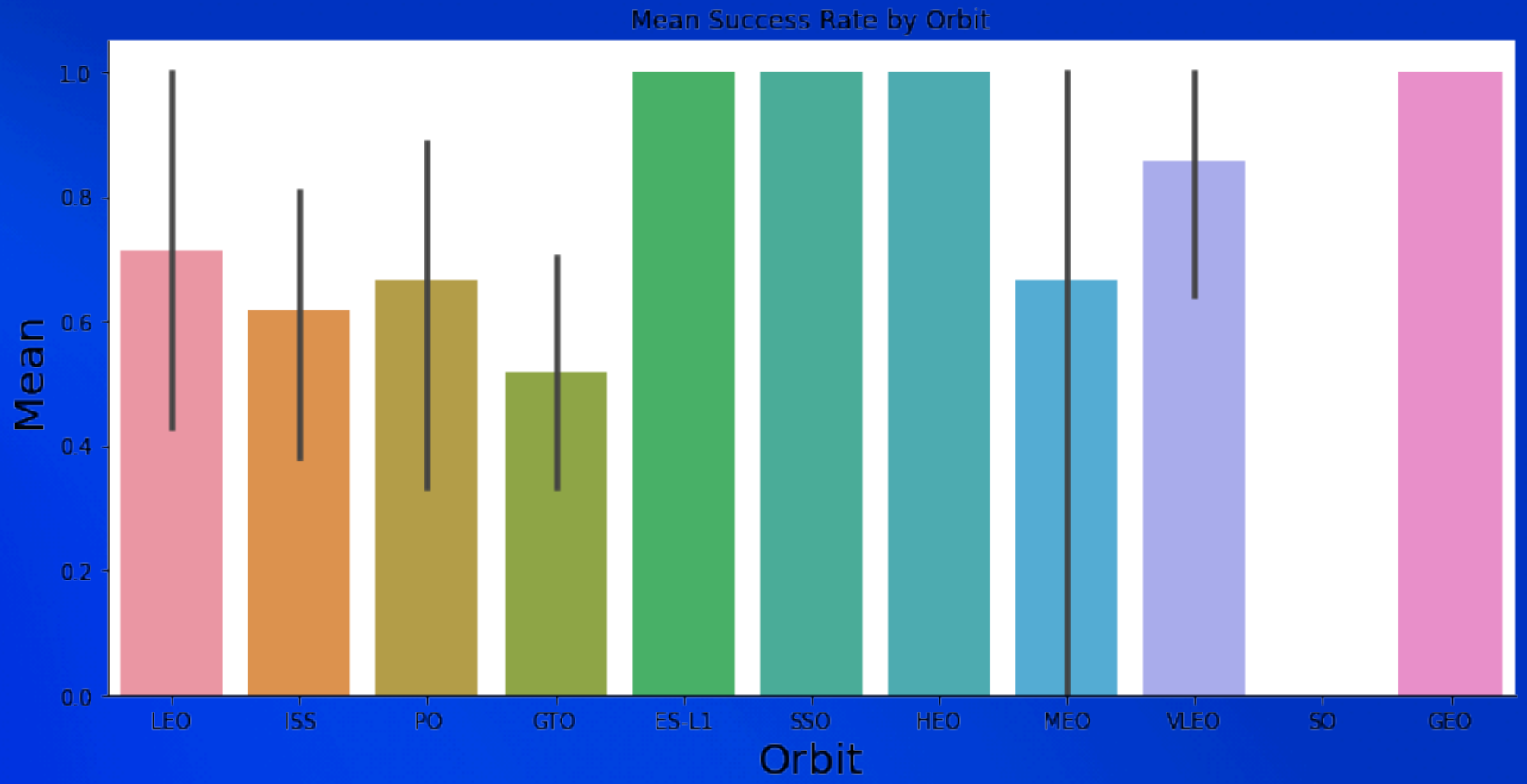


Explanation:

The greater the payload for site CCAF SLC-40 the higher the rocket success rate.

From this figure there are no other apparent patterns to be determined.

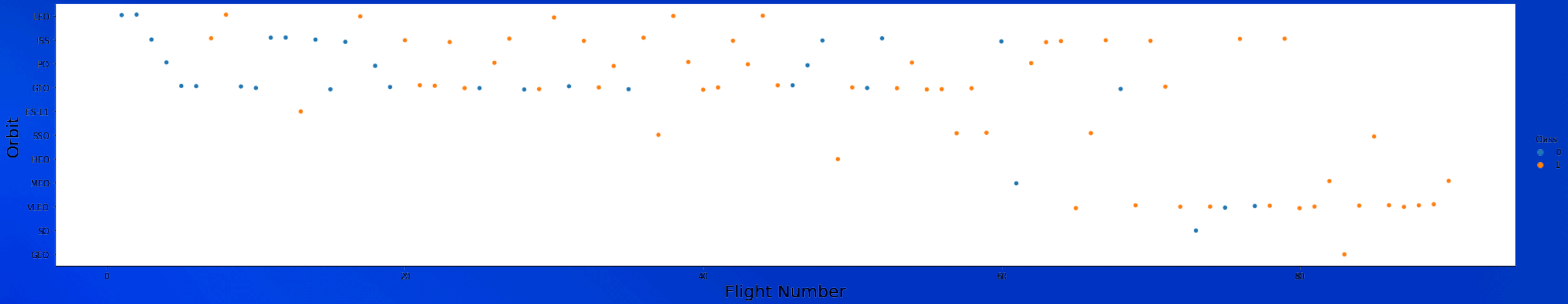
Success Rate vs. Orbit Type



Explanation:

Orbit HEO,SSO, ES-L1, GEO and LEO have the highest success rates.

Flight Number vs. Orbit Type

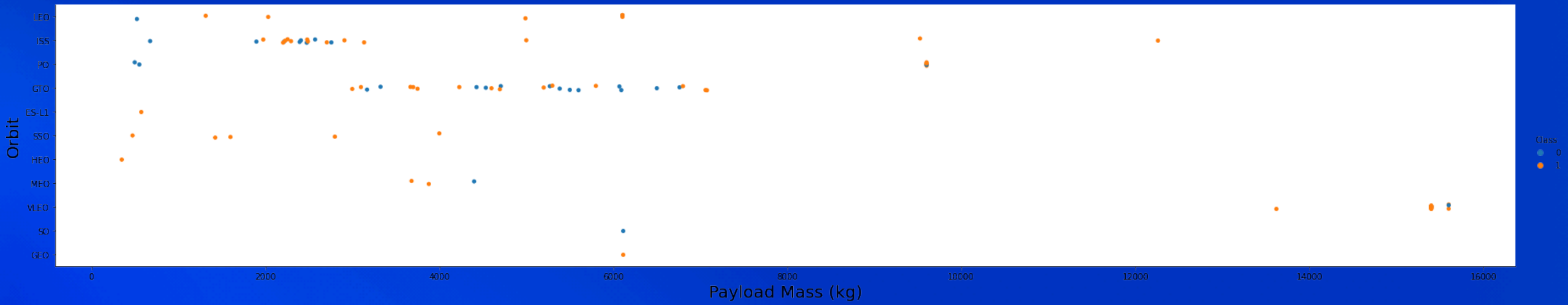


Explanation:

In the LEO orbit, the higher the flight number the higher success rate.

In the other orbits there seem to be no apparent patterns.

Payload vs. Orbit Type



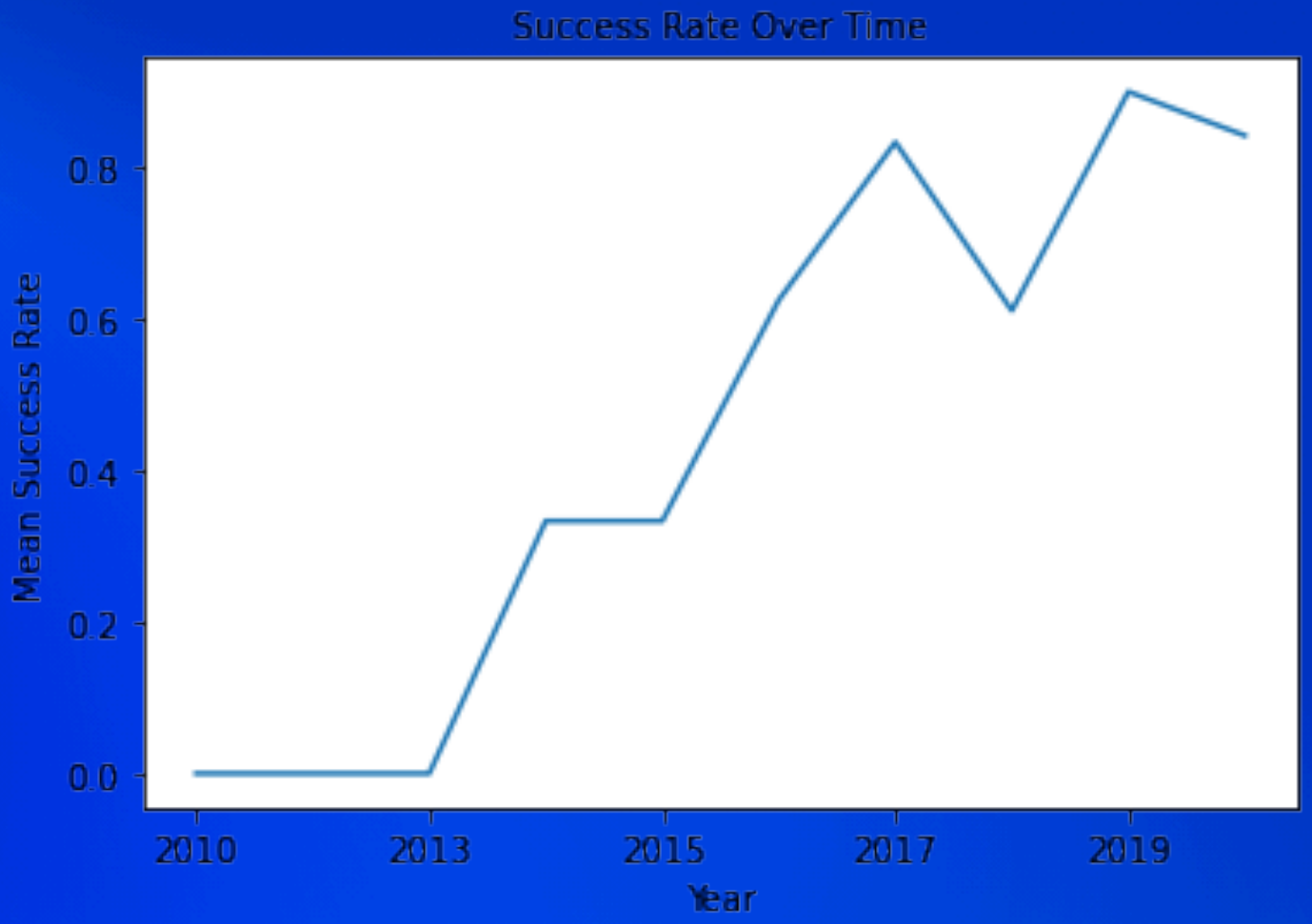
Explanation:

In the LEO orbit the heavier the payload the higher the success rate.

In the GTO orbit the heavier the payload the lower the success rate.

In the ISS the results are more mixed.

Launch Success Yearly Trend



Explanation:

The yearly launch success rate increasingly went up between the years 2013-2020.

All Launch Site Names

```
%%sql
SELECT DISTINCT LAUNCH_SITE
FROM SPACEXTBL;
```

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

Explanation:

Using the term DISTINCT will show only unique Launch_Site entries .

Launch Site Names Begin with 'CCA'

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Explanation:

The sign % is the ‘wild card’, the first 5 items starting with ‘CCA’ are displayed.

Total Payload Mass

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

SUM(PAYLOAD_MASS__KG_)
45596

Explanation:

The SUM function sums up the payload_mass(kg) for the customer ‘NASA(CRS)’.

Average Payload Mass by F9 v1.1

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) \
    FROM SPACEXTBL \
    WHERE BOOSTER_VERSION = 'F9 v1.1';
```

AVG(PAYLOAD_MASS__KG_)
2928.4

Explanation:

Using the AVG function to average the payload_mass(kg) for booster_version= ‘F9 v1.1 .

First Successful Ground Landing Date

```
%sql SELECT MIN(Date) \
FROM SPACEXTBL \
WHERE (Landing_Outcome)= 'Success (ground pad)';
```

MIN(Date)
2015-12-22

Explanation:

The MIN function filters out the first date for successful landing on a ground pad.

Successful Drone Ship Landing with Payload between 4000 and 6000

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

Payload
JCSAT-14
JCSAT-16
SES-10
SES-11 / EchoStar 105

Explanation:

Here we are combining our search using the term AND to find the list of boosters that landed successfully on drone ship with payload between 4000-6000.

Total Number of Success and Failure Mission Outcomes

```
%sql SELECT MISSION_OUTCOME, COUNT(*) as total_number \
FROM SPACEXTBL \
GROUP BY MISSION_OUTCOME;
```

Mission_Outcome	total_number
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Explanation:

Here we use the COUNT function to figure out the total success/failure mission outcomes.

Boosters Carried Maximum Payload

```
%sql SELECT BOOSTER_VERSION \
FROM SPACEXTBL \
WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

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

Explanation:

We used the MAX function to filter out the boosters that carried the maximum payload_mass(kg).

2015 Launch Records

```
%sql SELECT substr(Date,6,2) as month, DATE,BOOSTER_VERSION, LAUNCH_SITE, [Landing_Outcome] \
FROM SPACEXTBL \
where [Landing_Outcome] = 'Failure (drone ship)' and substr(Date,0,5)='2015';
```

month	Date	Booster_Version	Launch_Site	Landing_Outcome
01	2015-01-10	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
04	2015-04-14	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

Explanation:

We are filtering drone ship failed landings and their booster version in the year 2015.

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

```
%%sql
SELECT "Landing _Outcome", COUNT("Landing _Outcome") AS COUNT_LAUNCHES, myDATE
FROM (
    SELECT "Landing _Outcome",
        (substr(Date, 7, 4) || '-' || substr(Date, 4, 2) || '-' || substr(Date, 1, 2)) as myDATE
    FROM SPACEXTBL
) AS subquery
WHERE myDATE BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY "Landing _Outcome", myDATE
ORDER BY COUNT_LAUNCHES DESC;
```

Landing _Outcome	COUNT_LAUNCHES	myDATE
No attempt	10	2017-03-16
Failure (drone ship)	5	2016-06-15
Success (drone ship)	5	2017-01-14
Controlled (ocean)	3	2015-02-11
Success (ground pad)	3	2017-02-19
Failure (parachute)	2	2010-12-08
Uncontrolled (ocean)	2	2014-09-21
Precluded (drone ship)	1	2015-06-28

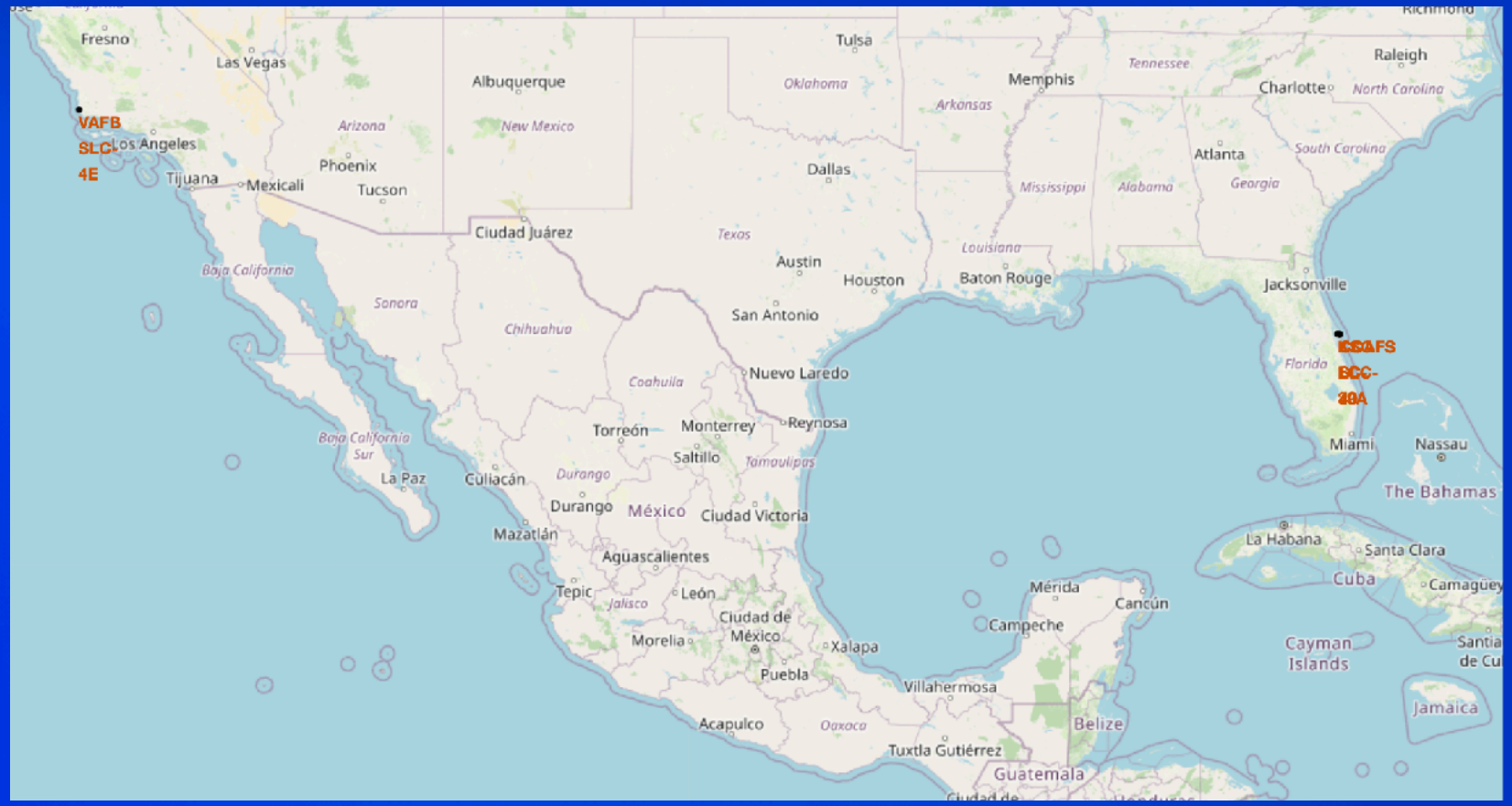
Explanation:

We rank the count for landing outcomes between 2010-06-04 and 2017-03-20 in descending order.

Section 3

Launch Sites Proximities Analysis

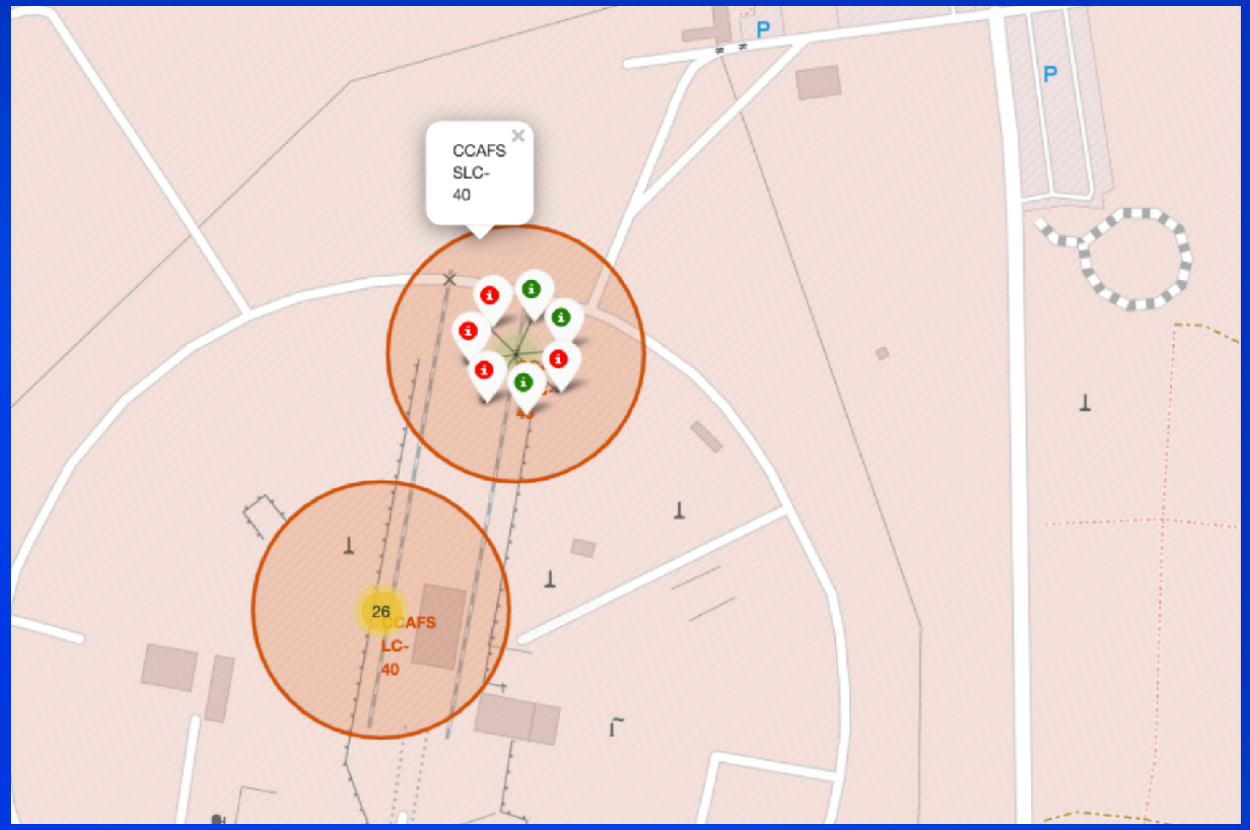
Folium Map Part 1
All Launch Site Location on a Global Map



Explanation:

We can see all launch sites on a global map.

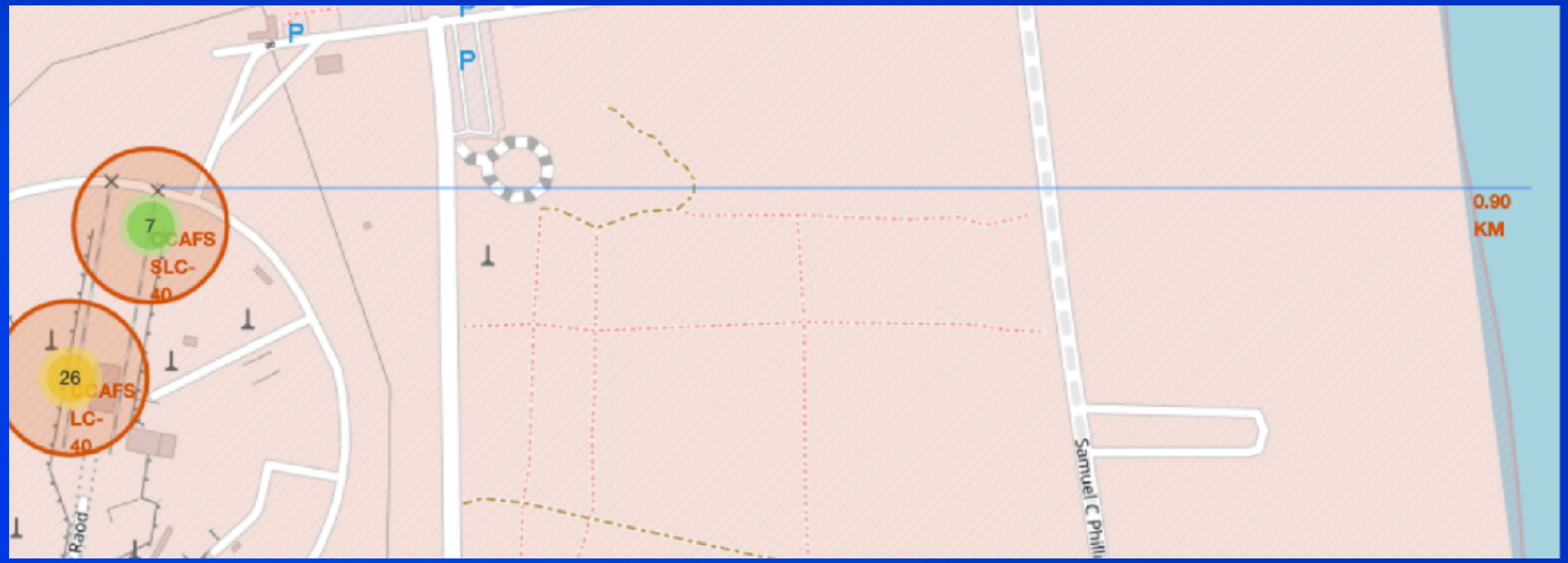
Folium Map Part 2
CCAFS LC-40 Launch Site Outcomes



Explanation:

We can see the color labeled map for the CCAFS LC-40 launch site outcomes.

Folium Map Part 3
Polyline Between A Launch Site and the Ocean



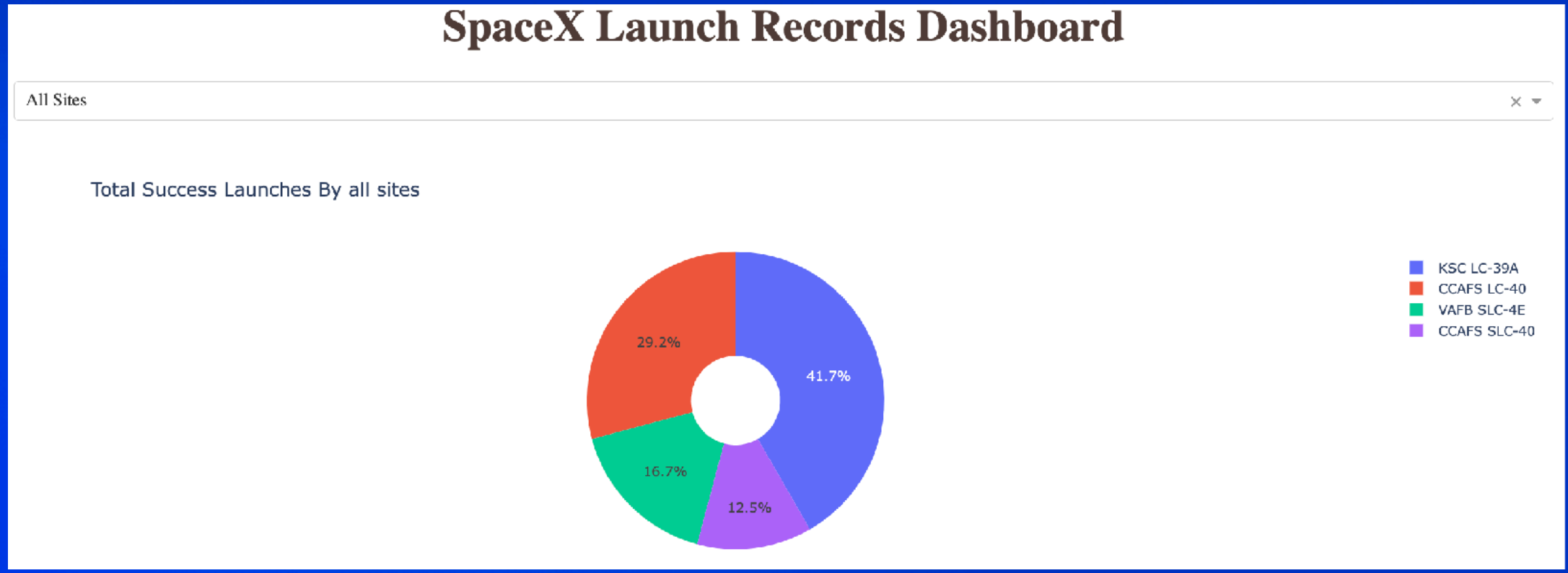
Explanation:

We can draw a line between the launch site CCAFS LC-40 and the ocean, measuring 0,9km.

Section 4

Build a Dashboard with Plotly Dash

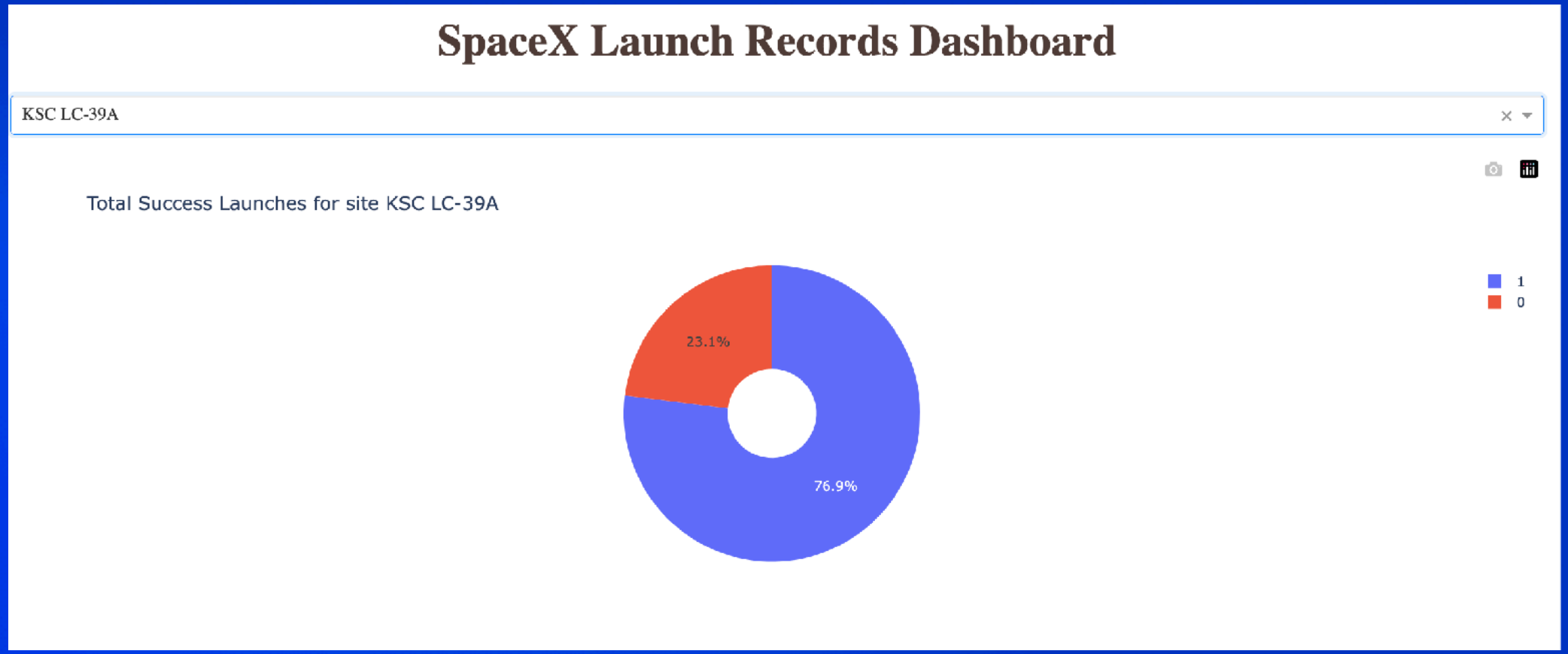
Dashboard Part 1
Launch Records for all Launch Sites



Explanation:

In the pie chart above we can see the total launch percentage success rate by site.

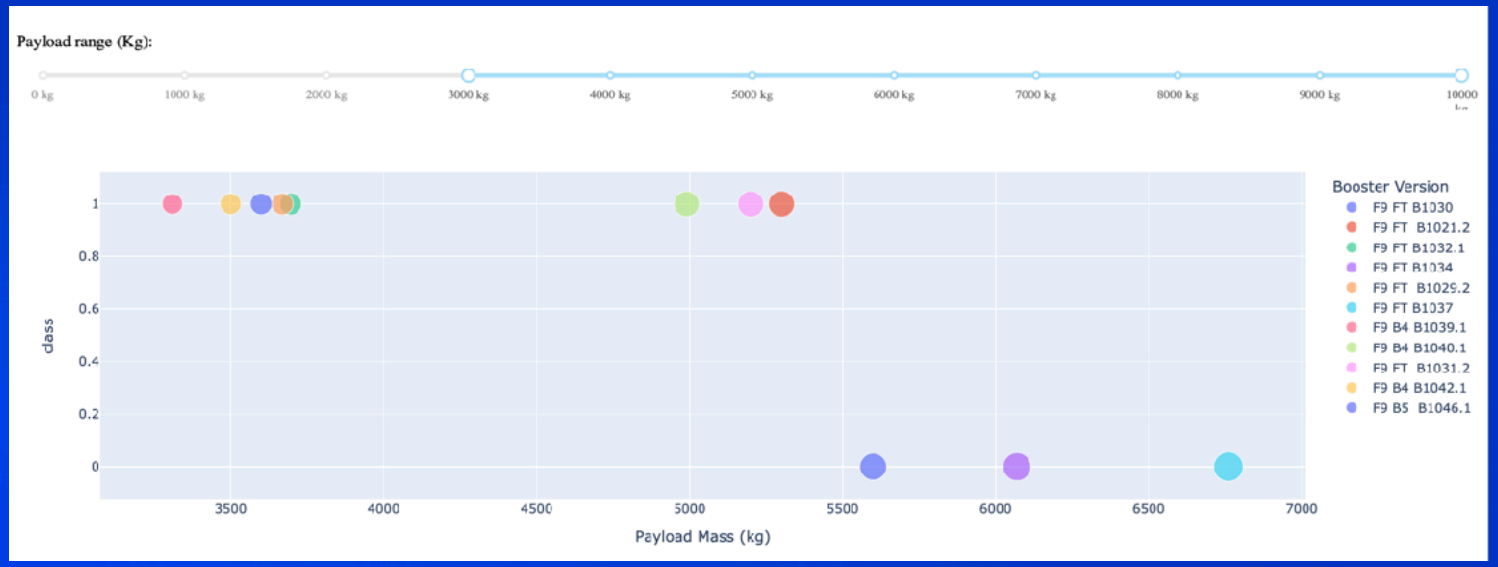
Dashboard Part 2
Launch Site with the Highest Launch Success Ratio

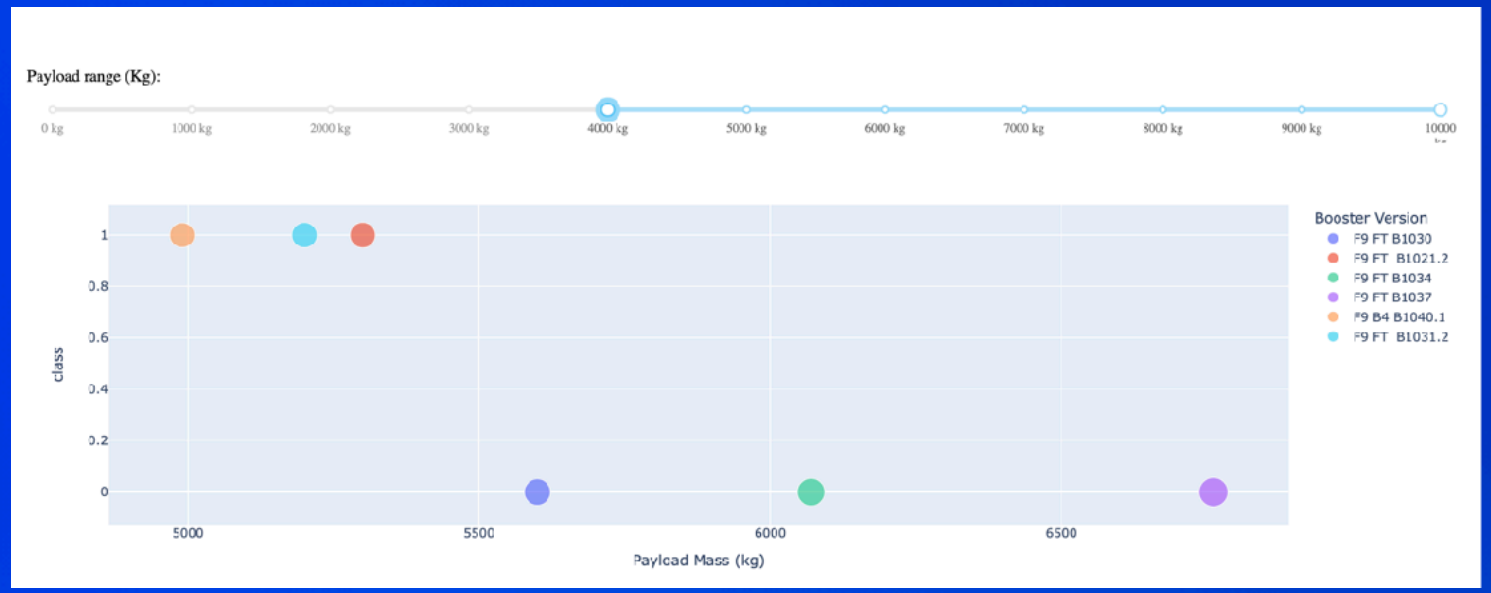


Explanation:

In the pie chart above we can see the percentage launch success rate for the most successful launch site (KSC LC-39A).

Dashboard Part 3
Payload vs. Launch Outcome Using an Interactive Range Slider





Explanation:

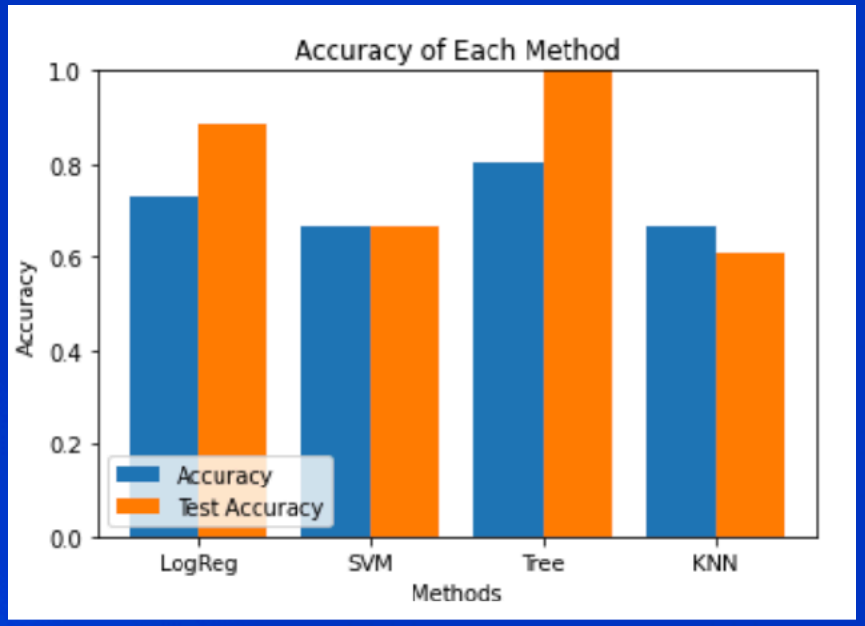
Using the interactive range slider we can observe that as the payload increases from 3000 to 4000 the number of booster versions succeeding is going down from 8 booster versions for 3000kg to only 3 booster versions succeeding with 4000kg payload.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

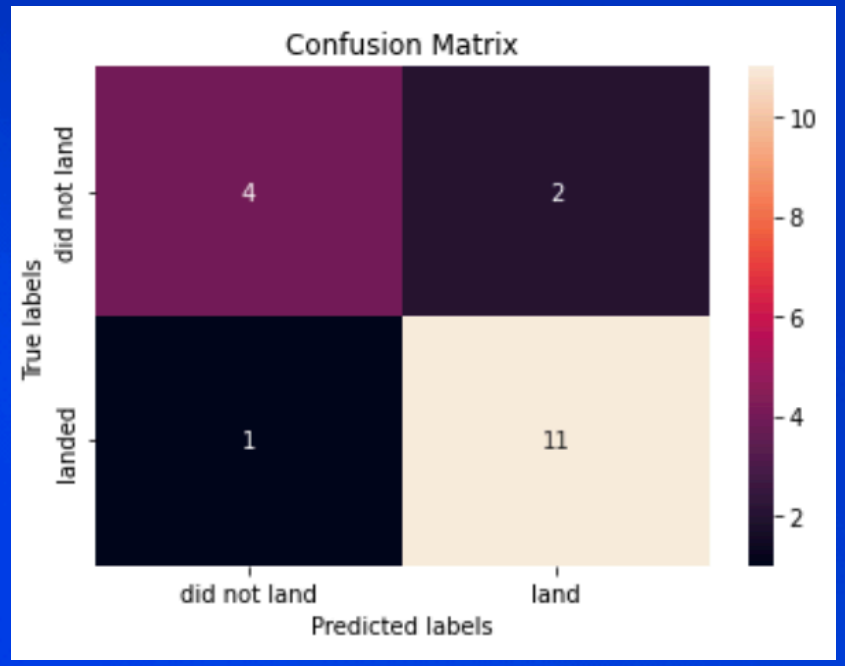
Model	Test data Accuracy	Best Parameters
Log Reg	0.88889	{'C': 0.1, 'penalty': 'l2', 'solver': 'lbfgs'}
SVM	0.66667	{'C': 0.001, 'gamma': 0.001, 'kernel': 'rbf'}
Tree	1.0	{'criterion': 'entropy', 'max_depth': 8, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2, 'splitter': 'best'}
KNN	0.61111	{'algorithm': 'auto', 'n_neighbors': 3, 'p': 1}



Explanation:

The Tree model has the highest Test data accuracy score as compared to the three other models.

Confusion Matrix - Tree model



Explanation:

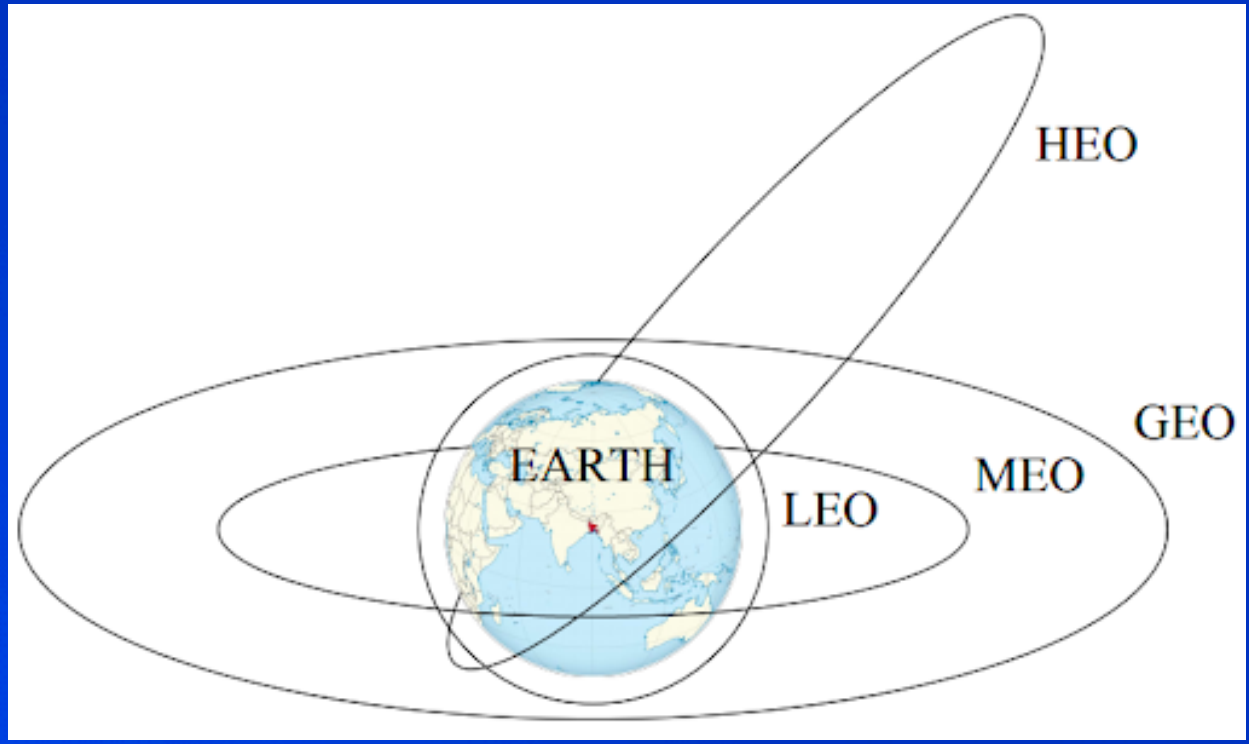
Examining the confusion matrix for the Tree model, we can see that the model can distinguish between the classes but the major problem is false positive.

Conclusions

- Launch success yearly trend; success rate keeps increasing since 2013.
- As the more massive the payload, the less likely the first stage to return.
- As the flight number increases, the first stage is more likely to land successfully.
- Success rate by orbit type; ES-L1, GEO, HEO and SSO have higher success rate.
- The success rate of the LEO orbit is related to the number of flights. Polar, LEO and ISS have higher success rates with heavier payloads.
- The Tree classifier is the best fit for the purpose of this analysis and this data set.

Appendix
Orbit and Landing Option Clarification

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



S. M. Rezaul Karim, Shadman Sakib, Md. Turiqul Islam, F. A. Sabbir Ahamed, A Review of Communications Satellite by Focusing on 'Bangabandhu Satellite-1', the First GEO Communications Satellite of Bangladesh, *International Journal of Networks and Communications*, Vol. 8 No. 5, 2018, pp. 123-128. doi: 10.5923 j.ijnc.20180805.01.

Thank you!