IBM Developer
SKILLS NETWORK

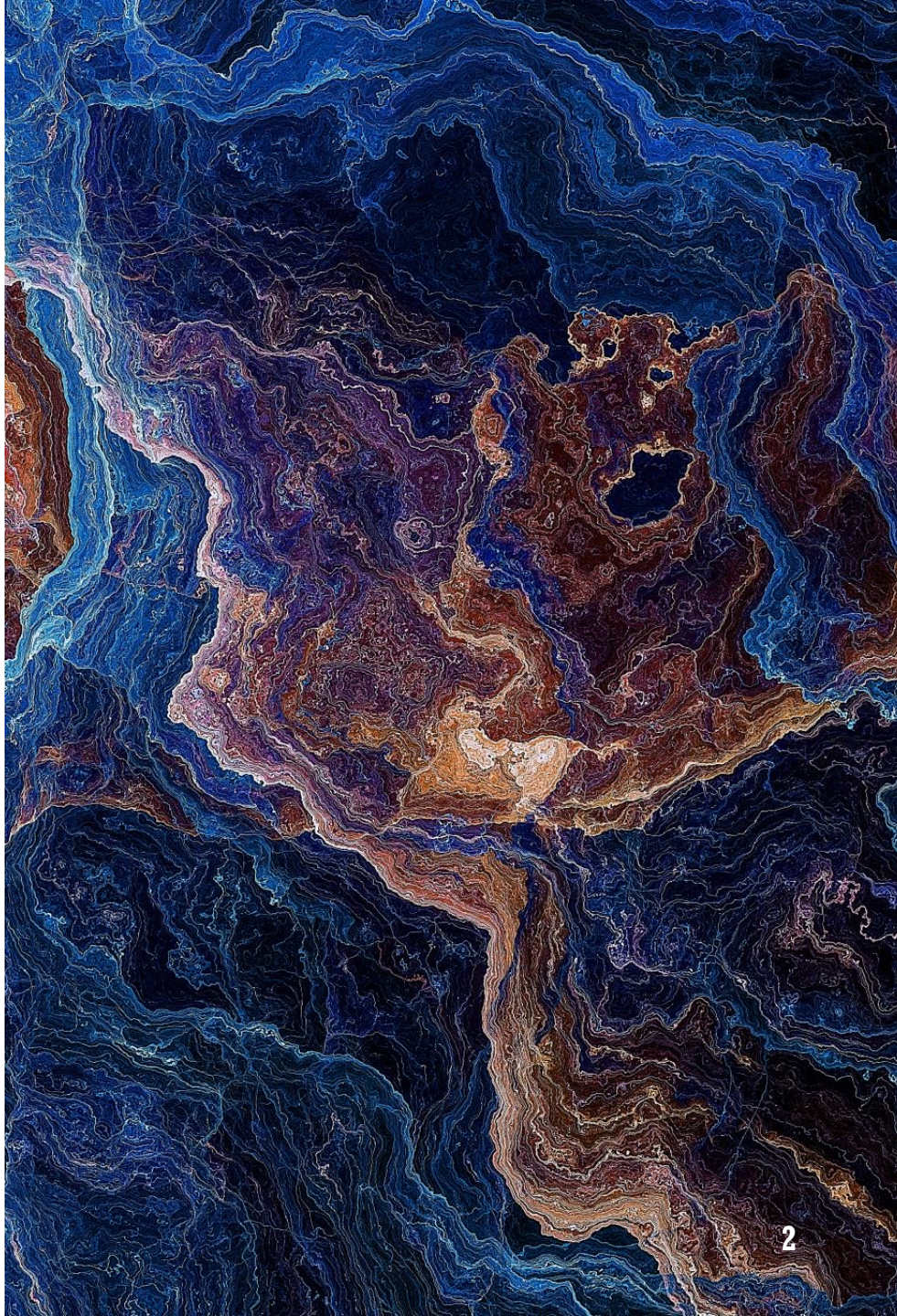
Winning Space Race with Data Science

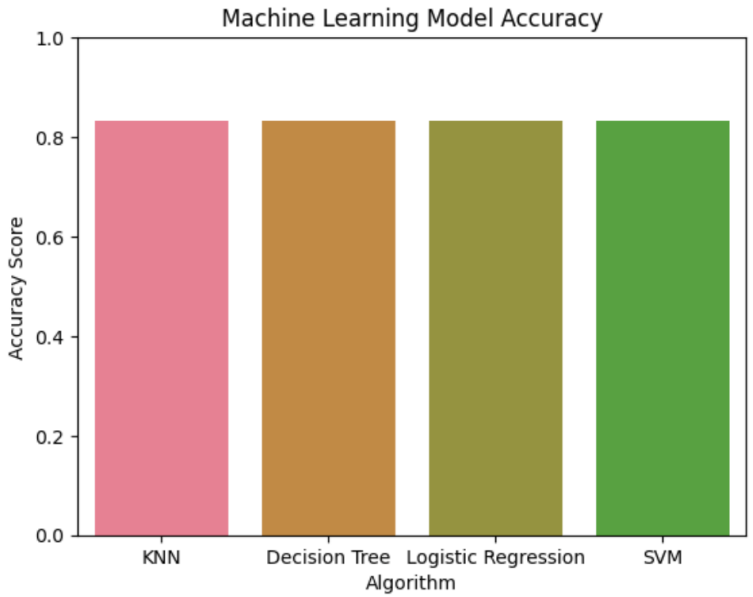
Affan Naushad

Thursday, July 4, 2024

OUTLINE

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





	Algorithm	Accuracy Score	Jaccard Score	F1 Score
0	KNN	0.833333	0.8	0.888889
1	Decision Tree	0.833333	0.8	0.888889
2	Logistic Regression	0.833333	0.8	0.888889
3	SVM	0.833333	0.8	0.888889

EXECUTIVE SUMMARY

- **Summary Of Methodologies –**
  - Data Acquisition:
    - API Requests to SpaceX API
    - Web Scraping with Beautiful Soup
  - Data Wrangling/Analysis:
    - Cleaning and transforming data using Python
    - SQL queries with SQLite3
  - Data Visualization:
    - Seaborn and Matplotlib for trend analysis, Folium for global mapping, Plotly and Dash for interactive dashboards
  - Predictive Modeling:
    - Machine Learning models for prediction: Achieved 96% accuracy with Decision Tree
- **Summary of all results:** "Successfully analyzed SpaceX Falcon 9 rocket landing outcomes, achieving 83% accuracy in predictive modeling using different Machine Learning algorithms."

INTRODUCTION

- **Project background and context:**

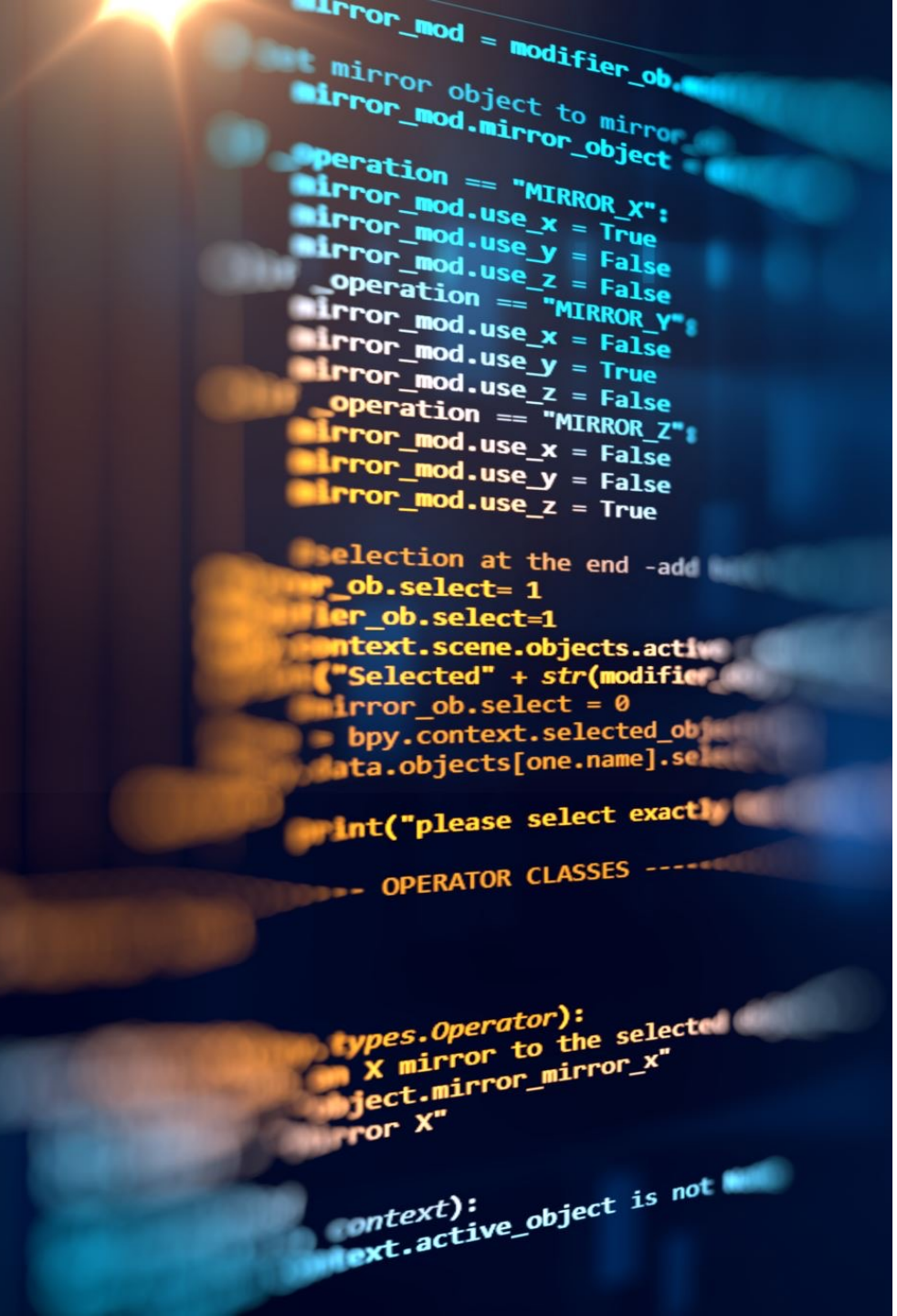
SpaceX, a leader in aerospace manufacturing and space transportation, advertises its Falcon 9 rocket launches at a cost of 62 million dollars per launch. This is significantly lower than the prices offered by other providers, which can go upwards of 165 million dollars per launch. The substantial cost savings achieved by SpaceX are largely attributed to their ability to reuse the first stage of the Falcon 9 rockets. Reusability of the first stage plays a crucial role in reducing overall launch costs. If we can accurately predict whether the first stage will successfully land, we can estimate the cost-effectiveness of a launch.

Problems you want to find answers:

- What factors influence the success of the first stage landing (example: Launch site, Payload Mass, Flight Number, etc.)
- How can we predict the best possible outcome of a Falcon 9 first stage landing.

Section 1

Methodology



METHODOLOGY

Executive Summary

- Data collection methodology:
  - API requests Via SpaceX Rest API
  - Web Scraping from Wikipedia using Beautiful Soup
- Perform data wrangling:
  - Filtering and cleaning the data. Using one hot encoding on categorical variables to get the data ready for Machine Learning.
- Perform exploratory data analysis (EDA) using visualization and SQL:
  - SQL queries for analysis and visualizations to see trends between different factors
- Perform interactive visual analytics using Folium and Plotly Dash:
  - Create Web-Based Interactive visualizations using Plotly and Dash
  - Used Folium to map the Launch sites on a Global Map and see the successes of launches vs failures
- Perform predictive analysis using classification models:
  - Logistic Regression, KNN, SVM, Decision Trees: Applied these models to predict rocket launch success. Achieved 94% accuracy with Decision Trees.

Data Collection - API

Data Collection Process: API-GitHubLink

1. API Requests to SpaceX REST API:

✓ Utilized API requests to gather comprehensive data on SpaceX rocket launches, including detailed launch records, rocket specifications, and mission outcomes.

Final Dataframe after API Calls:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
4	1	2010-06-04	Falcon 9	6123.547647	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003	-80.577366	28.561857
5	2	2012-05-22	Falcon 9	525.000000	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0005	-80.577366	28.561857
6	3	2013-03-01	Falcon 9	677.000000	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0007	-80.577366	28.561857
7	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None	1.0	0	B1003	-120.610829	34.632093
8	5	2013-12-03	Falcon 9	3170.000000	GTO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B1004	-80.577366	28.561857

Data Collection – SpaceX API Process

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

Step 1: This code fetches JSON data from a specified URL using an HTTP GET request (requests.get). It then decodes the JSON into a Python dictionary (json.loads(get.text)), converts it into a pandas DataFrame (pd.DataFrame(results)), and normalizes the JSON data into a tabular format (pd.json_normalize(results)). This process prepares the data for easy analysis and manipulation in 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
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
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)]
```

Step 2: This code organizes data about rocket launches by selecting essential details like which rocket was used, its payload, launch site, and core information. It cleans up the data by removing entries with extra boosters or multiple payloads, converts dates to a standard format, and limits the analysis to launches before November 13, 2020. This ensures we focus on the most relevant launch information for our analysis.

```
#Global variables
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

```
# Call getLaunchSite
getLaunchSite(data)
```

Step 3: List global variables and use get requests to store data into variables

```
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 4: Create a dictionary with all the data retrieved which we use to convert to a Pandas data frame ready for analysis

Data Collection - Scraping

Data Collection Process: Scraping-GitHub Link

1. Web Scraping from Wikipedia using Beautiful Soup:

✓ Utilized web scraping with Beautiful Soup to extract detailed records of space launches from Wikipedia, including rocket specifications, payload information, launch dates, and mission outcomes.

Final Dataframe after Scraping:

	Flight No.	Date	Time	Version Booster	Launch Site	Payload	Payload mass	Orbit	Customer	Launch outcome	Booster landing
0	1	4 June 2010	18:45	F9 v1.0B0003.1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	Failure
1	2	8 December 2010	15:43	F9 v1.0B0004.1	CCAFS	Dragon	0	LEO	NASA	Success	Failure
2	3	22 May 2012	07:44	F9 v1.0B0005.1	CCAFS	Dragon	525 kg	LEO	NASA	Success	No attempt\n
3	4	8 October 2012	00:35	F9 v1.0B0006.1	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success\n	No attempt
4	5	1 March 2013	15:10	F9 v1.0B0007.1	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success\n	No attempt\n

Data Collection – Scraping Process

Initiating the HTTP GET request to fetch data from a specified URL

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
data = requests.get(static_url).text
```

Creating a BeautifulSoup Object

```
soup = BeautifulSoup(data, 'html.parser')
```

Finding tables in the data

```
html_tables = soup.find_all('table')
html_tables
```

Creating Dictionary and appending data

```
launch_dict= dict.fromkeys(column_names)
```

1. It iterates through each row (<tr> tag) within the table.
2. Checks if the first cell (<th> tag) contains a numeric flight number.
3. If a numeric flight number is found (flag is True), it extracts data from each cell (<td> tag) in that row and saves it into a dictionary (launch_dict) under the key 'Flight No.'.
4. It prints each extracted flight number (flight_number) as it processes.
5. Then we go onto append all columns in the launch_dict with the data

```
# Extract each table
for table_number, table in enumerate(soup.find_all('table', "wikitable plainrowheaders collapsible")):
    # Get table rows
    for rows in table.find_all("tr"):
        # Check to see if first table heading is a number corresponding to the launch number
        if rows.th:
            if rows.th.string:
                flight_number = rows.th.string.strip()
                flag = flight_number.isdigit()
            else:
                flag = False
        else:
            flag = False

        # Get table elements
        row = rows.find_all('td')

        # If it is a number, save cells in a dictionary
        if flag:
            extracted_row += 1
            # Flight Number value
            # Append the flight_number into launch_dict with key `Flight No.`
            launch_dict['Flight No.'].append(flight_number)
            print(flight_number)  # Print flight number
```

Data Wrangling

Data Wrangling Process: Wranging-GitHub Link

1. Data Cleaning/Transforming:

- ✓ "We began by assessing SpaceX launch data for missing values and categorized columns into numerical and categorical types. We then analyzed launch site frequencies and categorized missions by orbit type. Mission outcomes were classified as successes or failures, influencing the creation of a success classification variable (df['Class']) based on first-stage landing outcomes. We concluded with a 66.67% success rate calculation from the df['Class'] column and saved the processed data to dataset_part_2.csv."

Final DataFrame after Wrangling:

	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

Data Wrangling Process (Some Steps)

1.

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

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

2.

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

```
GTO        27
ISS        21
VLEO       14
PO          9
LEO         7
SSO         5
MEO         3
ES-L1       1
HEO         1
SO          1
GEO         1
```

3.

```
data = df['Outcome']
landing_class = []

#Creating a loop to change row value to 0 if outcome was unsucessful and change it to 1 if it was sucessful
for i in data:
    if i in bad_outcomes:
        i = 0
    else:
        i = 1
    landing_class.append(i)  # Fixed appending to list
```

4.

```
df['Class']=landing_class
df[['Class']].head(8)
```

	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

EDA WITH DATA VISUALIZATION

Visualization-GitHub Link

Scatter Plots:

These graphs are valuable for visualizing relationships between two variables, offering insights into how one factor in the dataset influences another. This visualization helps inform the development of our machine learning algorithms.

- ❑ Payload Mass Vs Flight Number
- ❑ Launch Site Vs Flight Number
- ❑ Launch Site Vs Payload Mass
- ❑ Orbit Vs Flight Number
- ❑ Orbit Vs Payload Mass

➡ **Scatter Plots Drawn**

Bar Graphs:

Bar graphs are useful for comparing categorical data or displaying counts or frequencies of different categories.

- ❑ Class (Success Rate Vs Orbit)

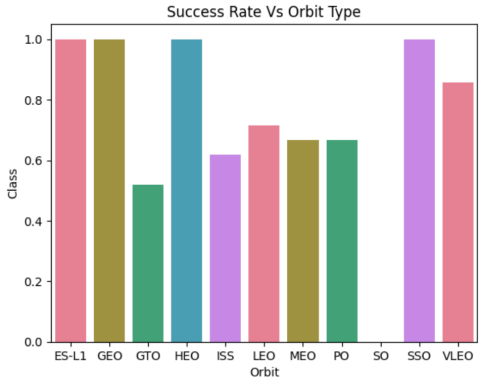
➡ **Bar Graphs Drawn**

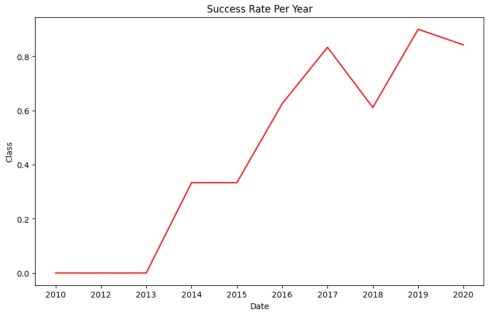
Line Graphs:

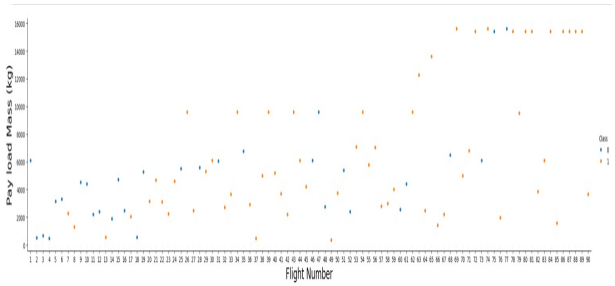
are useful for illustrating trends and changes over time or continuous data.

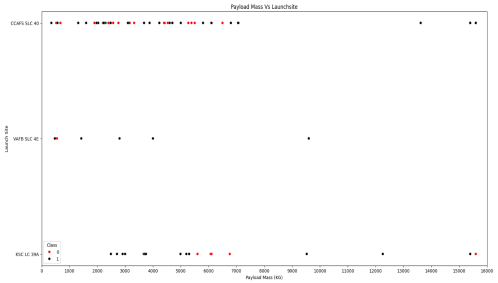
- ❑ Class (Success Rate) Vs Year

➡ **Line Graphs Drawn**









EDA WITH SQL

SQL-GitHub Link

The SQL queries performed to gain insight were:

- 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 successful 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 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.

Build an Interactive Map with Folium

Folium-GitHub Link

Map Objects	Code	Description
Map Marker	folium.marker()	Adds a map marker
Icon Marker	folium.Icon()	Customizes marker icons
Circle Marker	folium.circle()	Draw a circle on the map where marker is located
PolyLine	folium.polyline()	Creates a line
Marker Cluster	MarkerCluster()	Simplifies multiple markers having the same coordinate into one big cluster

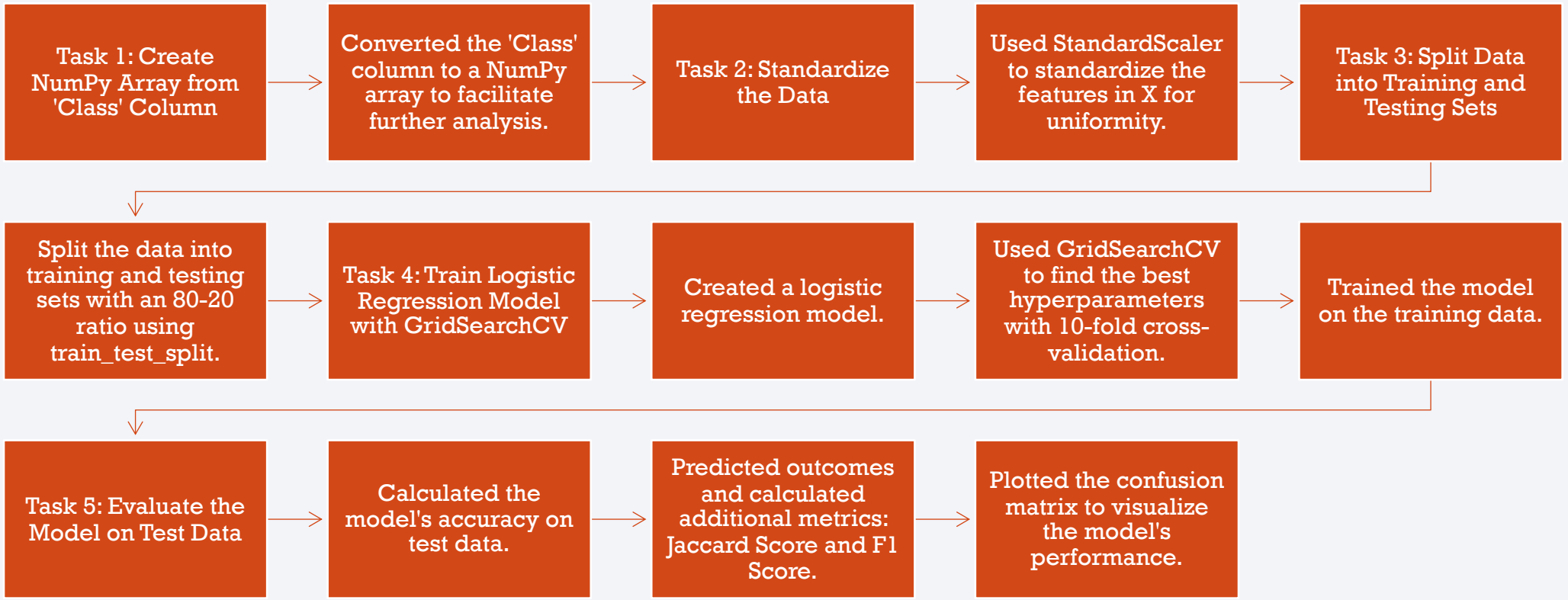
BUILD A DASHBOARD WITH PLOTLY DASH

Dash-GitHub Link

- We used two types of graphs in our dashboard:
  - **Pie Chart:** This chart showed the success rate for all sites combined or filtered to a specific launch cite
  - **Scatter Plot:** This chart was used to show the correlation between Success Rate and Payload Mass while also specifying the Booster Versions.

Command Names	Code	Description
Dropdown	dcc.Dropdown()	Creates a dropdown menu
Rangeslider	dcc.RangeSlider()	Adds a range slider
Callback	@app.callback	Links user interaction to updates
Pie Chart/ Scatter Plot	px.pie(), px.scatter()	Creates a pie graph or scatter plot

Predictive Analysis (Classification)



MachineLearning-GitHub Link

In the next slide we will display the code needed to perform these tasks.

Predictive Analysis (Classification)

```
y = data['Class'].to_numpy()
```

```
transform = preprocessing.StandardScaler()
X = transform.fit_transform(X)
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X,y, test_size=0.2, random_state=2)
```

```
#Performing a gridsearch to find the best parameters for our Logisitc Regression Model
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge

lr=LogisticRegression()

logreg_cv = GridSearchCV(estimator=lr, param_grid=parameters, cv=10)

logreg_cv.fit(X_train,Y_train)

#Printing the best parameters and the accuracy on the validation data
print("Best parameters are: ", logreg_cv.best_params_)
print("accuracy:", logreg_cv.best_score_)
```

```
Best parameters are:  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy: 0.8464285714285713
```

```
logistic_accuracy = logreg_cv.score(X_test,Y_test)
print("The Accuracy on the Test Data is:" , logistic_accuracy)

#Additional Metrics
predictions = logreg_cv.predict(X_test)
LR_JaccardIndex = jaccard_score(Y_test, predictions)
print("The Jaccard Score is:", LR_JaccardIndex)
LR_F1_Score = f1_score(Y_test, predictions)
print("The F1 Score is:", LR_F1_Score)
```

```
The Accuracy on the Test Data is: 0.8333333333333334
The Jaccard Score is: 0.8
The F1 Score is: 0.888888888888889
```

Lets look at the confusion matrix:

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

This example showed a machine learning model with Logistic Regression. In this project we follow the same code to create different models. We will also create a Decision Tree Model, SVM Model, and KNN Model. Upon completion of all we compare the metrics and can conclude the Decision Tree Model is the most accurate.

EXPLORATORY DATA ANALYSIS RESULTS

INTERACTIVE ANALYTICS DEMO IN SCREENSHOTS

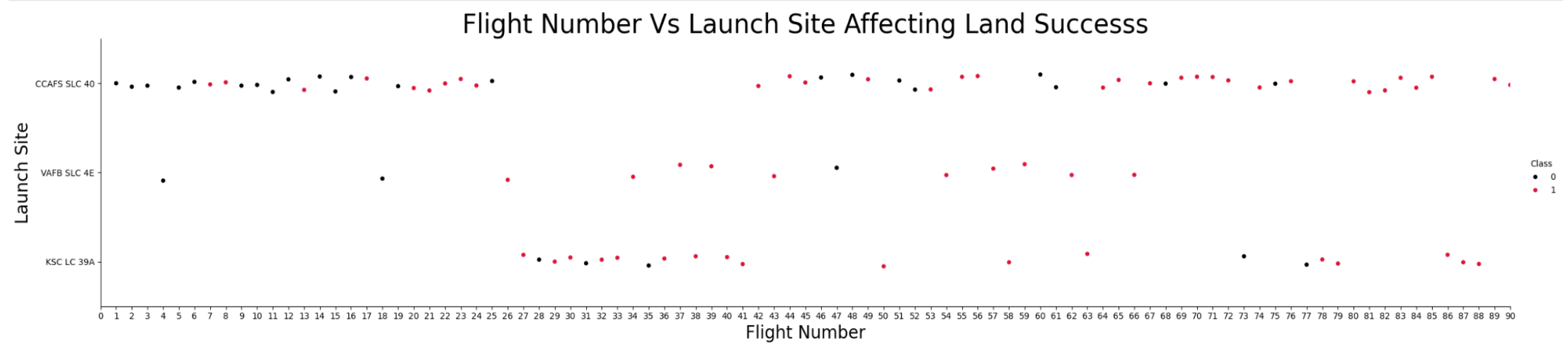
PREDICTIVE ANALYSIS RESULTS

RESULTS

Section 2

Insights drawn from EDA

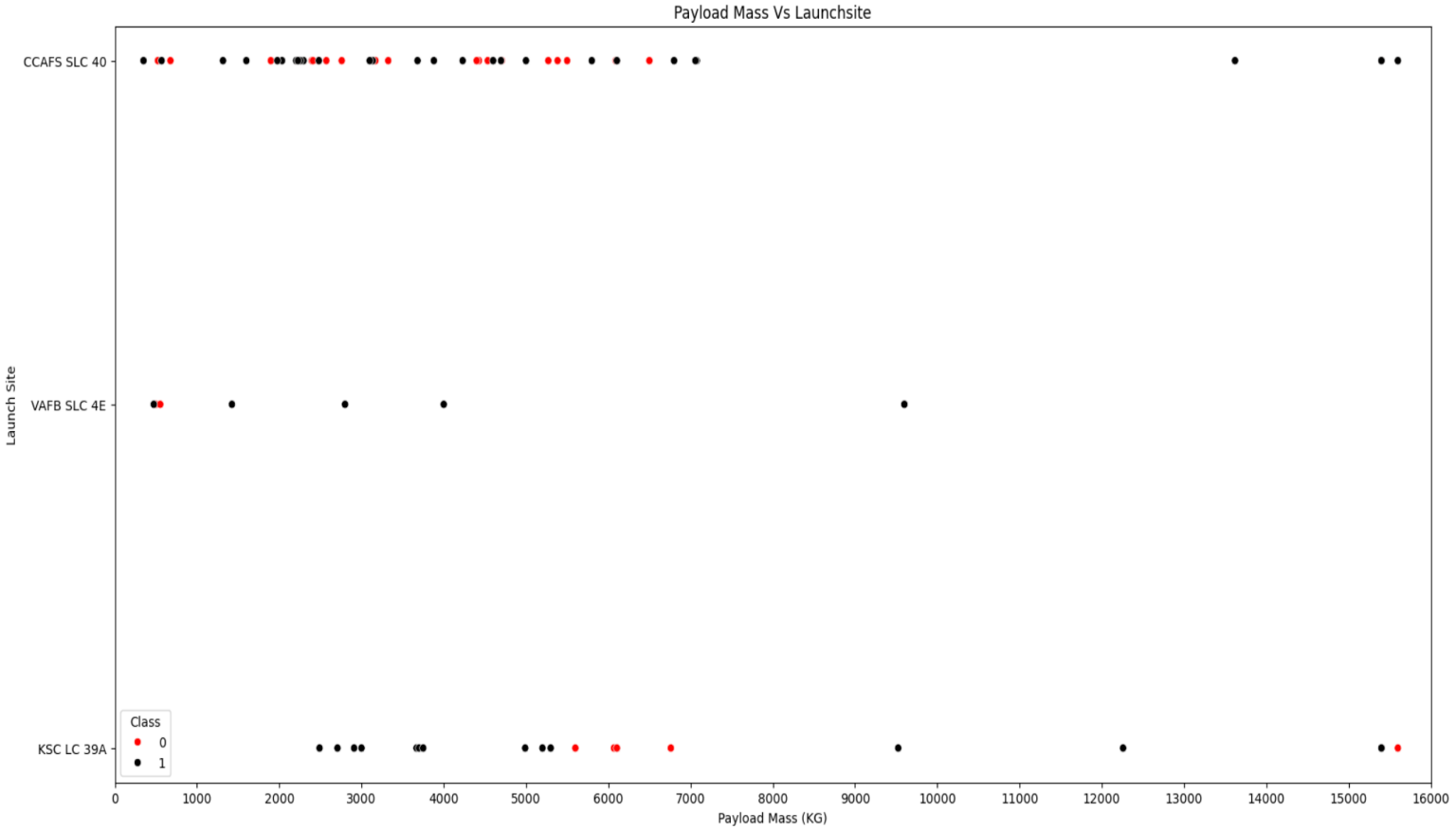
Flight Number vs. Launch Site



This scatter plot shows the correlation between Flight Number and Launch Site, and how they affect a successful landing.

- We see that with flight numbers higher than 30, the success rate increases for each launch site.

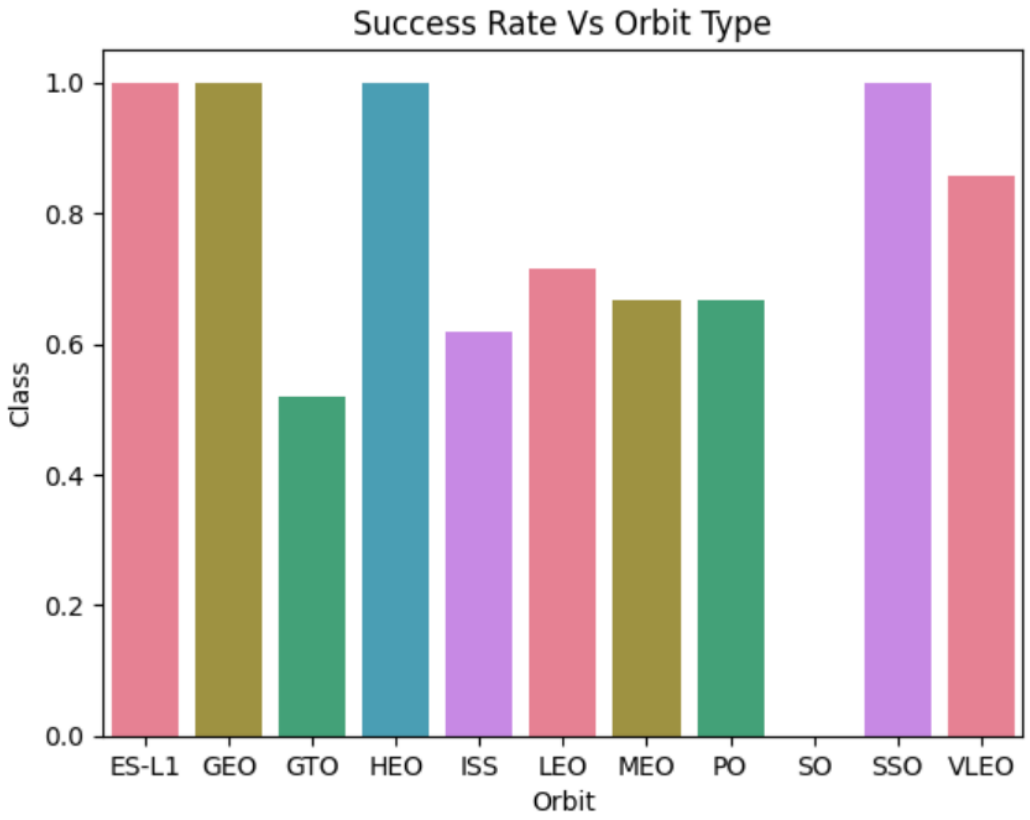
PAYLOAD VS. LAUNCH SITE



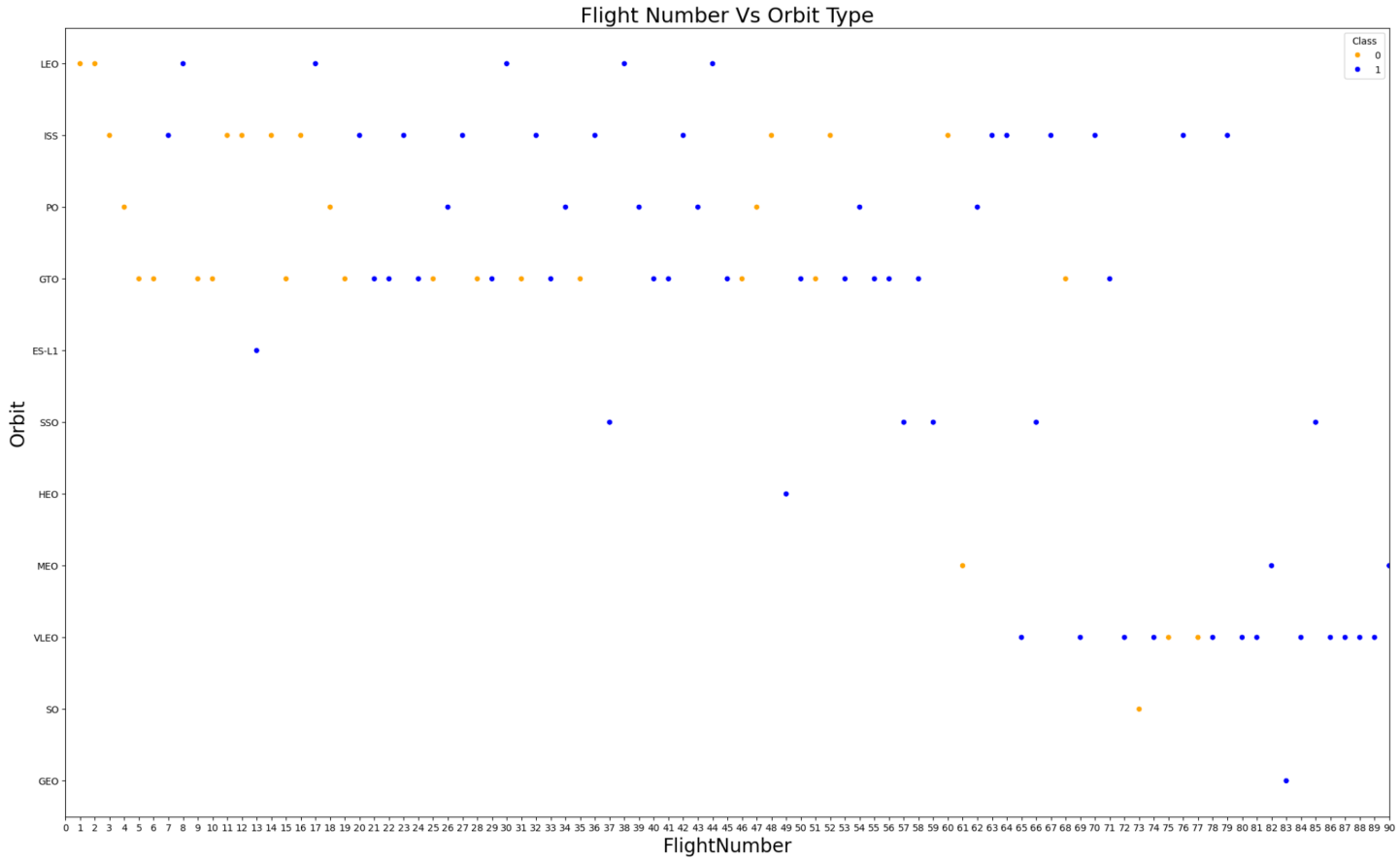
This scatter plot shows the correlation between Payload Mass and Launch Site, and how they affect a successful landing.

We see that with payload mass over 7000 kgs there is barely a failure in landing on our plotted data.

SUCCESS RATE VS. ORBIT TYPE



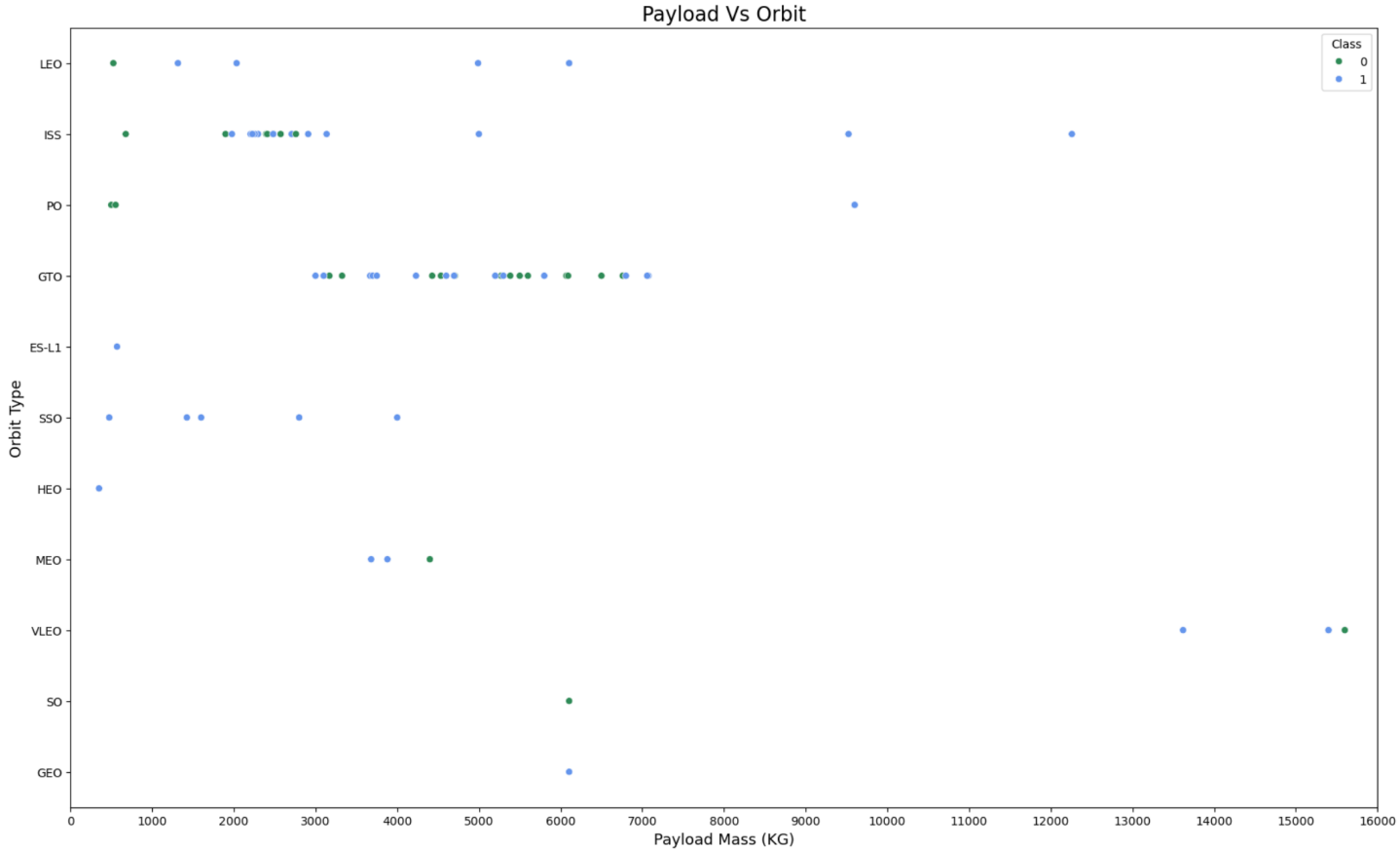
- This bar graph shows a hierarchal representation of success rates based off orbit types
  - We see the SO orbit type has a 0% success rate.
  - We see the ES-L1, GEO, HEO, SSO have a perfect success rate



FLIGHT NUMBER VS. ORBIT TYPE

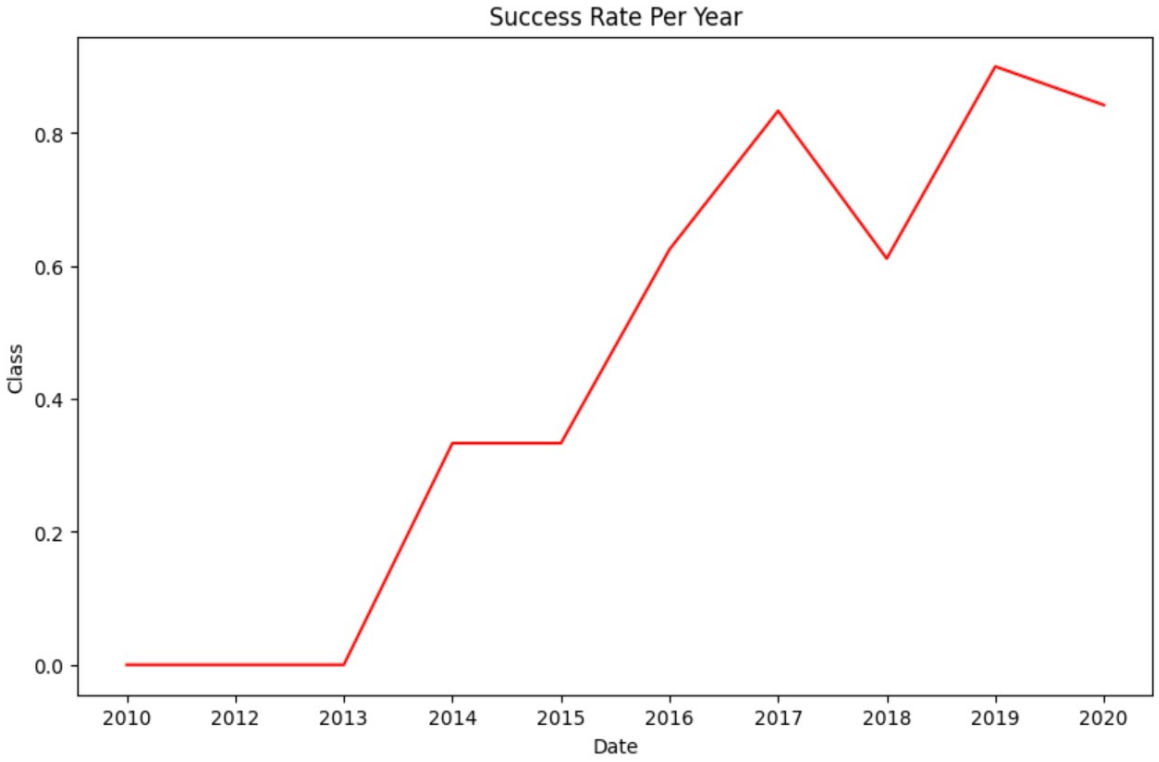
- This scatter plot shows the correlation between Flight Number and Orbit Type, and how they affect a successful landing.
- We see that in the LEO orbit, success appears related to the number of flights. On the other hand, compared to orbits like GTO, we see no correlation between orbit type and flight number.

PAYLOAD VS. ORBIT TYPE



- This scatter plot shows the correlation between Payload Mass and Orbit Type, and how they affect a successful landing.
- For heavy payloads the successful landing rate is more catered towards the Polar, LEO, and ISS orbit types
- However, for an orbit like GTO we can not make this conclusion as there are both successful and failed landings as the payload mass increases.

LAUNCH SUCCESS YEARLY TREND



THIS LINE PLOT SHOWS THE DRASTIC INCREASE OF SUCCESS RATE OVER THE YEARS FROM 2010 TO 2020

ALL LAUNCH SITE NAMES

```
%sql SELECT DISTINCT "Launch_Site" FROM SPACEXTBL;
```

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

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

Description:

- In this query we ask to output all unique launch sites
- The Query outputs all the unique launch sites which are:
  - CCAFSLC-40
  - VAFB SCL-4E
  - KSC LC-39A
  - CCAFS SLC-40

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

```
%sql SELECT * FROM SPACEXTBL 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

LAUNCH SITE NAMES BEGIN WITH 'CCA'

Description:

- In this query we ask to output 5 records where the launch site names begin with CCA.

Output:

- The Query outputs the 5 records seen above. All launch sites we see begin with the three letter CCA

Total Payload Mass

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

```
%sql SELECT SUM("PAYLOAD_MASS__KG_") AS "TOTAL PAYLOAD MASS" FROM SPACEXTBL WHERE Customer = "NASA (CRS)";
 * sqlite:///my_data1.db
Done.
```

TOTAL PAYLOAD MASS
45596

- Description:
- In this query we ask to output the total sum of payload mass carried by the boosters that are launched by NASA (CRS)
- Output:
- We see that the total sum of payload mass carried by boosters launched by NASA (CRS) is 45,596 KG.

AVERAGE PAYLOAD MASS BY F9 V1.1

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

```
%sql SELECT AVG("PAYLOAD_MASS__KG_") FROM SPACEXTBL WHERE "Booster_Version" = "F9 v1.1";
```

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

AVG("PAYLOAD_MASS__KG_")
2928.4

Description:

- In this query we ask to output the average payload mass in KG carried by the booster version F9 v1.1.

Output:

- The average payload mass in KG carried by booster version F9 v1.1 is 2928.4 KG

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

```
%sql SELECT MIN("Date") FROM SPACEXTBL WHERE "Landing_Outcome" LIKE "%Success (ground pad)%"
```

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

MIN("Date")
2015-12-22

FIRST SUCCESSFUL GROUND LANDING DATE

Description:

- We ask to output the date that the first successful landing outcome on the ground pad was achieved

Output:

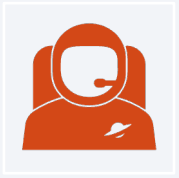
- December 22, 2015 was the date the first successful landing outcome on the ground pad was achieved.

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql SELECT "Booster_Version" FROM SPACEXTBL WHERE "Landing_Outcome" = "Success (drone ship)" AND "PAYLOAD_MASS__KG_" BETWEEN 4000 AND 6000;
 * sqlite:///my_data1.db
Done.
```

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

Description:



We ask to output the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

Output:

The names of the 4 boosters are listed above.

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

```
%sql SELECT CASE WHEN "Mission_Outcome" LIKE '%Success%' THEN 'Success' ELSE 'Failure' END AS "Mission_Outcome_Category", COUNT(*) AS "Total_Count" FROM SPACEXTBL GROUP BY "Mission_Outcome_Category";
```

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

Mission_Outcome_Category	Total_Count
Failure	1
Success	100

DESCRIPTION:

WE ASK TO OUTPUT THE TOTAL NUMBER OF SUCCESSFUL MISSION OUTCOMES IN THE DATA COMPARED TO THE TOTAL NUMBER OF FAILURES IN THE DATA.

OUTPUT:

WE HAVE 1 FAILURE AND 100 SUCCESSFUL MISSION OUTCOMES IN OUR DATA.

Boosters Carried Maximum Payload

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

```
%sql SELECT "Booster_Version" AS "Booster Version", "PAYLOAD_MASS__KG_" AS "Payload Mass in KG" FROM SPACEXTBL WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTBL);
```

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

Booster Version	Payload Mass in KG
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

Description:

We ask to output the the names of the booster version which have carried the maximum payload mass. We use a sub-query to complete this task

Output:

The output is a list of booster versions which have a payload mass of 15,600 which is the maximum payload mass.

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

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

```
%sql SELECT substr(Date, 6, 2) AS Month, Booster_Version, Launch_Site, Landing_Outcome FROM SPACEXTBL WHERE substr(Date, 0, 5) = '2015' AND Landing_Outcome = 'Failure (drone ship)';
```

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

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

2015 LAUNCH RECORDS

Description:

- We ask to output the month names, failed landing outcomes in drone ship, their booster versions, and launch site names for the months in year 2015

Output:

- Two rows are outputted as we see above. One failure was in January, while the other was in April. Both were from Launch Site CCAFS LC-40. However, the Booster Version is different.

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

Description:

We ask to output the number 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

We have to use the count command to count how many landing outcome occurred. We filter the date between the two dates specified and group by landing outcome values so that we can cluster the similar landing outcome values together and we specify desc at the end to output it in descending order

Output:

We see the output below

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.

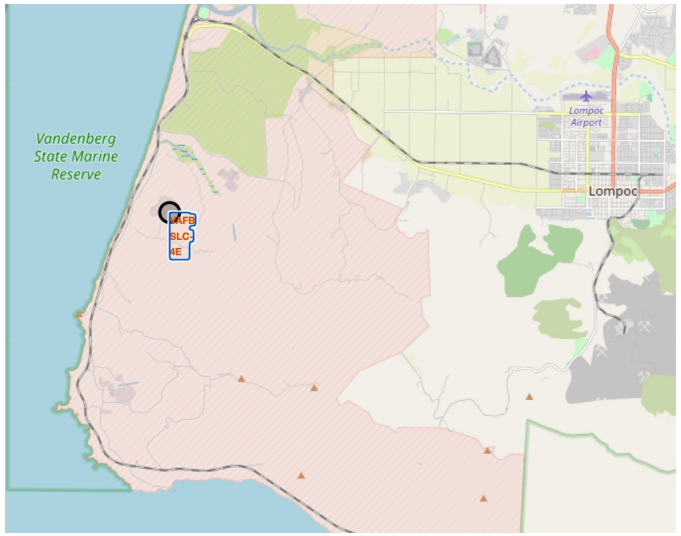
```
SELECT "Landing_Outcome" AS "Landing Outcome", COUNT("Landing_Outcome") AS "Total Count" FROM SPACEXTBL WHERE "Date" BETWEEN "2010-06-04" AND "2017-03-20" GROUP BY "Landing_Outcome" ORDER BY COUNT("Landing_Outcome") DES
```

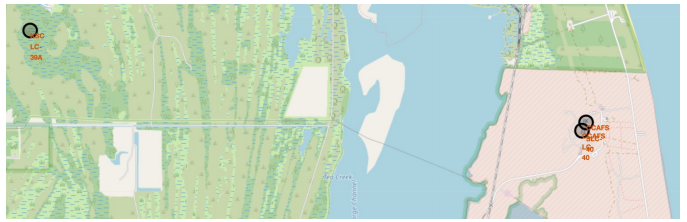
```
 * sqlite:///my_data1.db
Done.
```

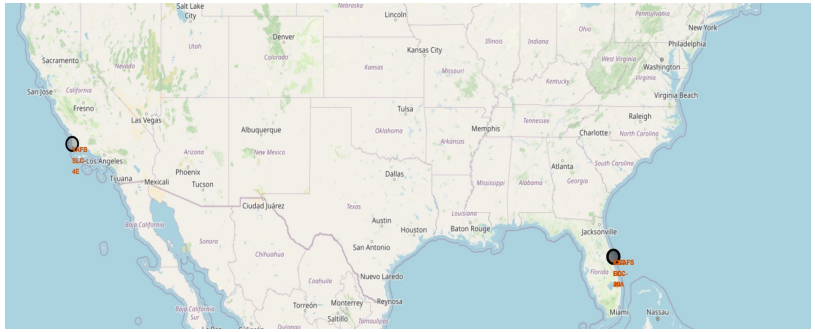
Landing Outcome	Total Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis





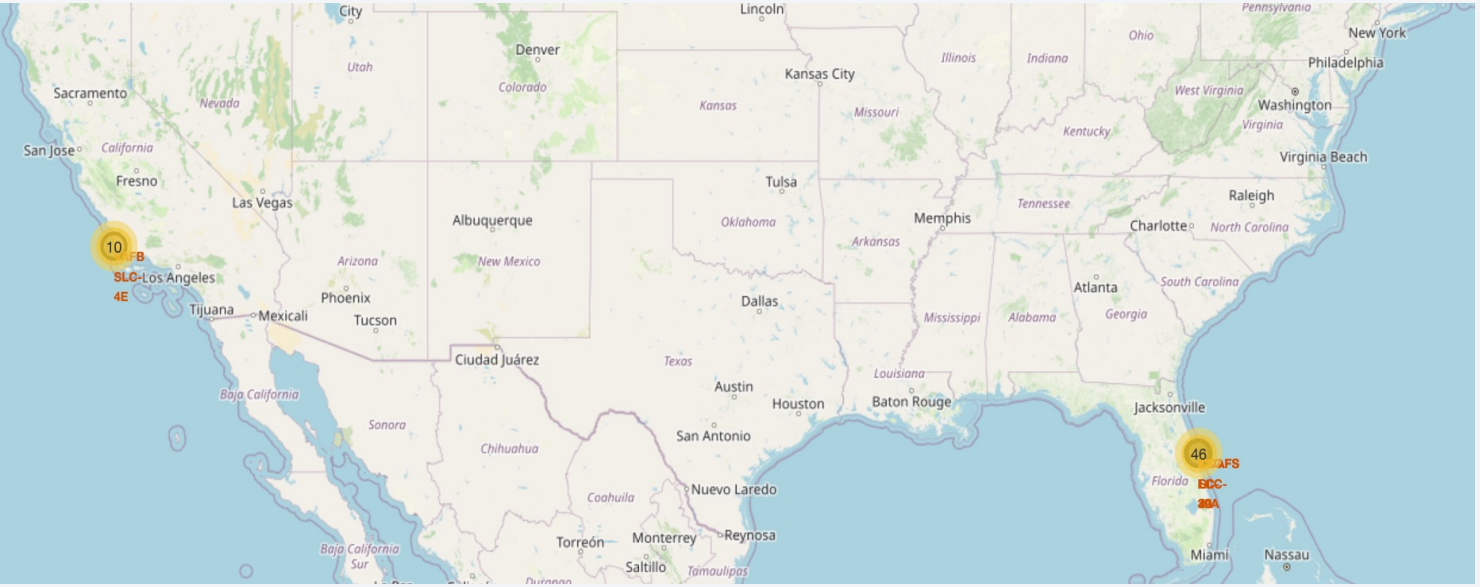


FOLIUM – ALL LAUNCH SITES DISPLAYED ON WORLD MAP

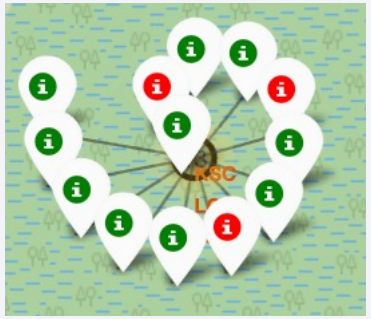
- We see that all of Space X Launch Sites are located in the United States near the Florida and California regions.

Folium - SpaceX Launches Clustered on World Map

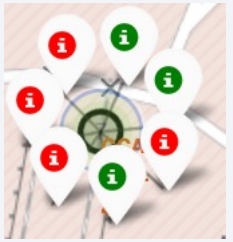
We see that 10 launches are made from the VAFB SLC-4E launch site. 26 are made from CCAFS LC-40, 7 are made from CCAFS SLC-40. Lastly, 13 were made from KSC LC-39A. In the pictures below the green markers represent successful lands while red represent failures.



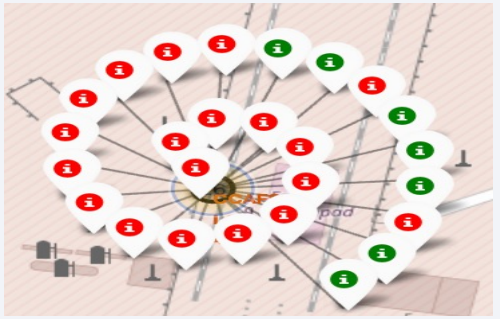
KSC LC-39A

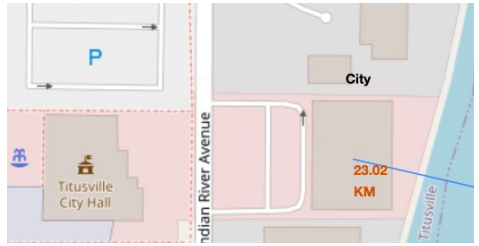


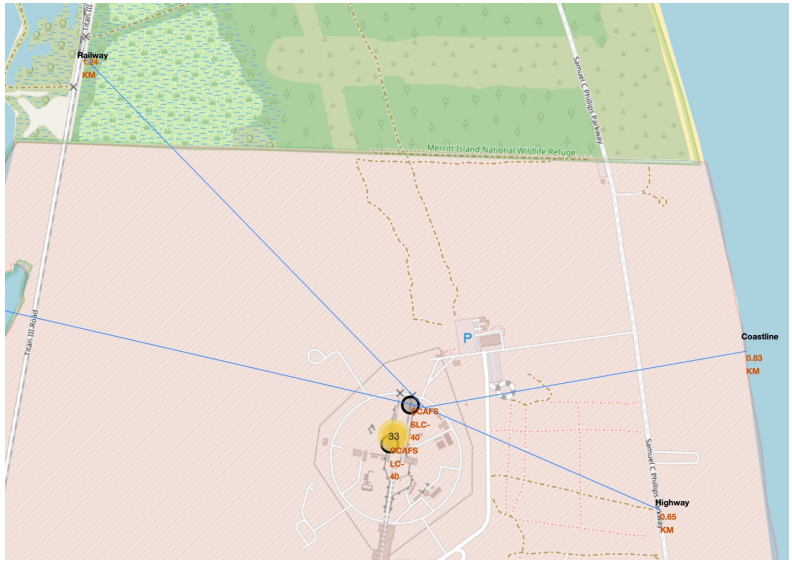
CCAFS SLC-40



CCAFS LC-40

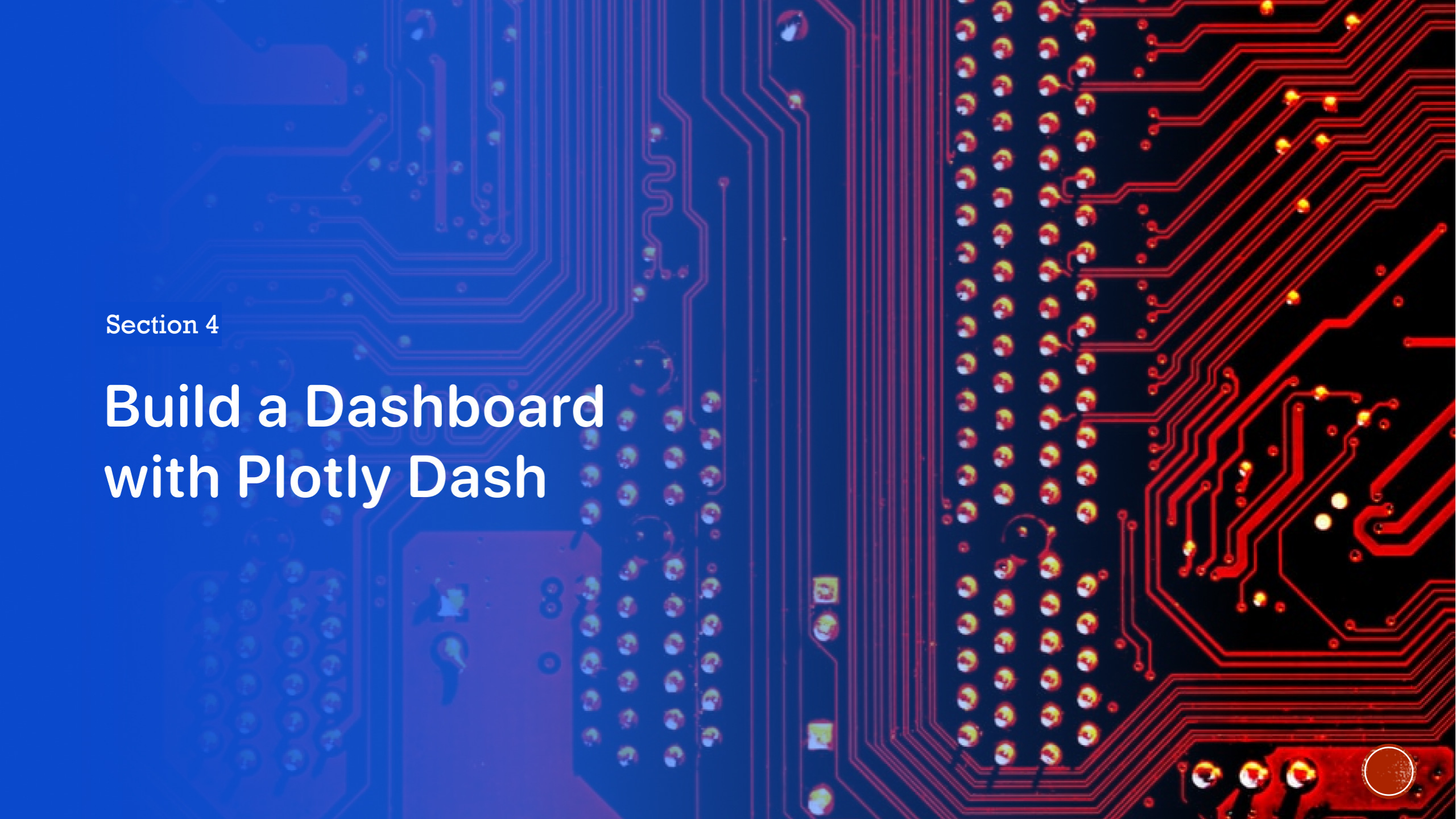


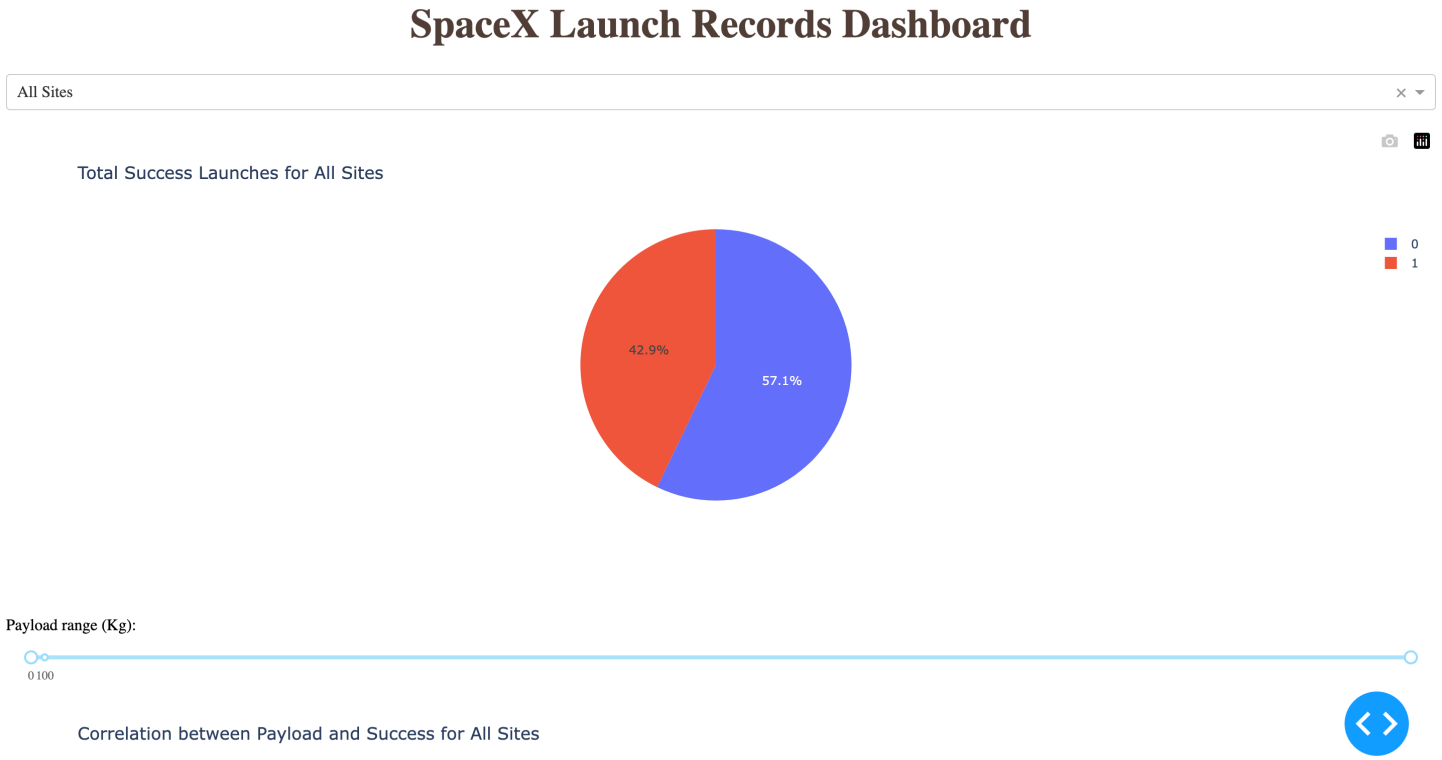




FOLIUM – CCAFS SLC-40 DISTANCE FROM RAILWAY, HIGHWAY, AND CITY

- This map will show the distance of the CCASF SLC-40 Launch Site to a railway, highway, and the city of Titusville in Florida.
- We use the polyline function to create a line from the launch site to the destination and use an equation to measure the distance.
- The distance from the coastline is 0.83 km. Distance from Railway is 1.24 km. Distance from Highway is 0.65 km. Lastly the distance to the city is 23.02 km.


Section 4
Build a Dashboard with Plotly Dash

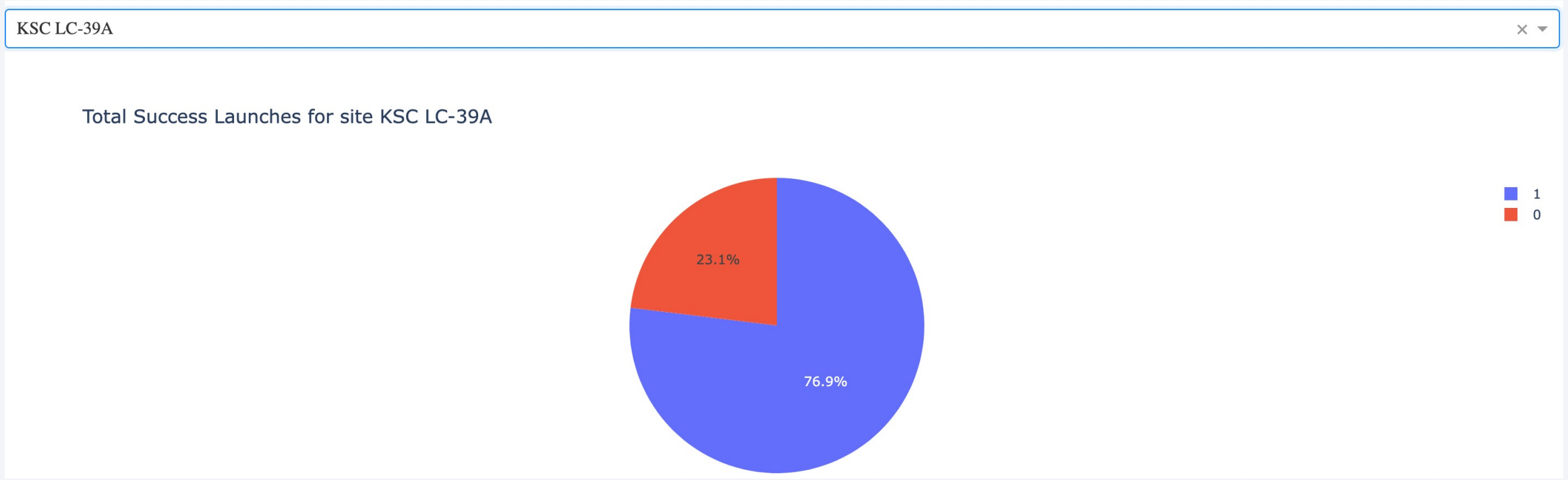


DASHBOARD – LAUNCH SUCCESS OF ALL SITES PIECHART

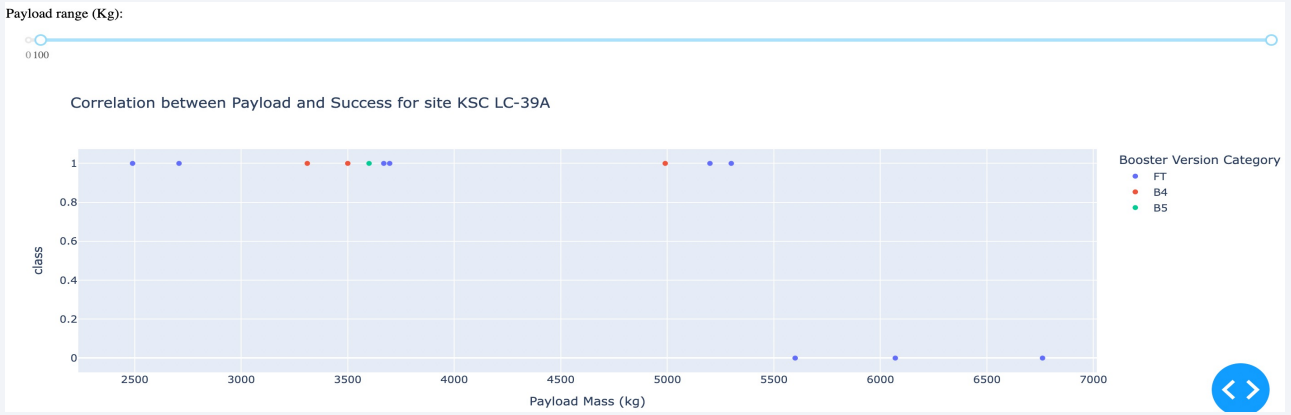
- This Web-Interactive Dashboard has the total successful launches rate for all the sites combined.
  - In total all the launch sites had a 42.9% successful landing
  - While 57.1% of launches were failures

DASHBOARD – LAUNCH SUCCESS OF KSC LC-39A (HIGHEST LAUNCH SUCCESS RATIO)

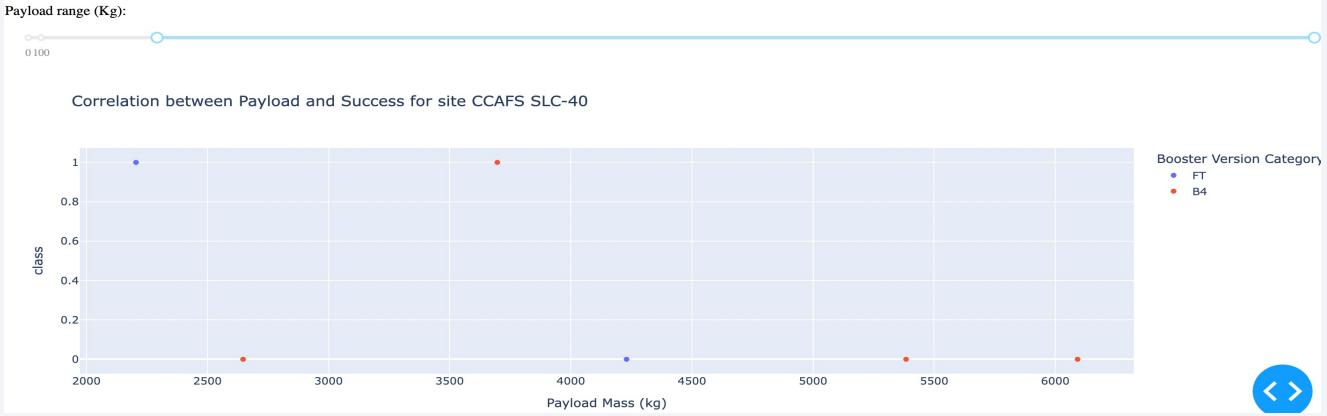
The KSC LC-39A has the highest launch success ratio out of all other launch sites. It boasts a 76.9% launch success and a 23.1% launch failure



DASHBOARD – PAYLOAD MASS (KG) VS LAUNCH SUCCESS SCATTER PLOT FOR LAUNCH SITES



We see that at the same payload mass slider range both launch sites CCAFS LC-40 and KSC LC-39A have different success rates. KSC LC-39A has much more failures and FT booster version leads to the most successes as compared to CCAFS LC- 40 which also has the FT booster version leading to the most successes.

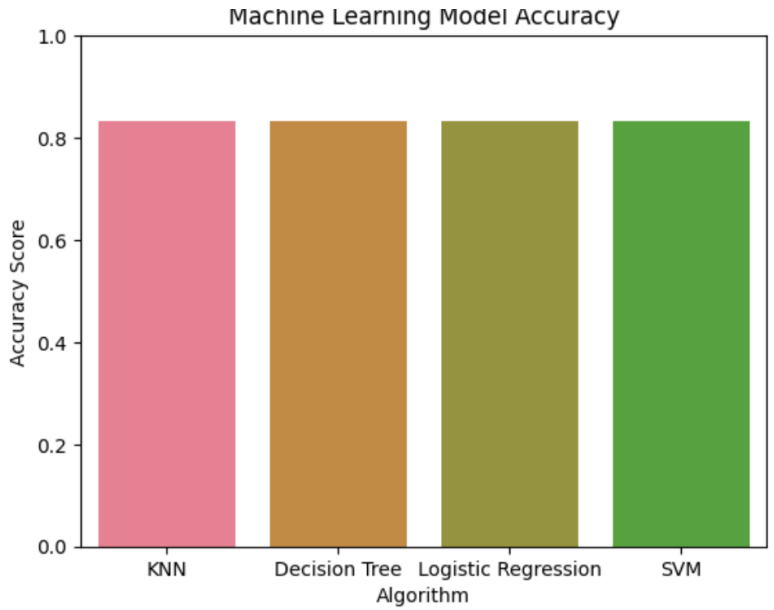


We see that at the same payload mass slider range both launch sites VAFB SLC-4E and CCAFS SLC-40 have different success rates. VAFB SLC-4E has one successful launch which was at a higher payload mass. However, it also had failures at higher payload mass. CCAFS SLC-40 has no success with a high payload mass.

Section 5

Predictive Analysis (Classification)

	Algorithm	Accuracy Score	Jaccard Score	F1 Score
0	KNN	0.833333	0.8	0.888889
1	Decision Tree	0.833333	0.8	0.888889
2	Logistic Regression	0.833333	0.8	0.888889
3	SVM	0.833333	0.8	0.888889



CLASSIFICATION ACCURACY

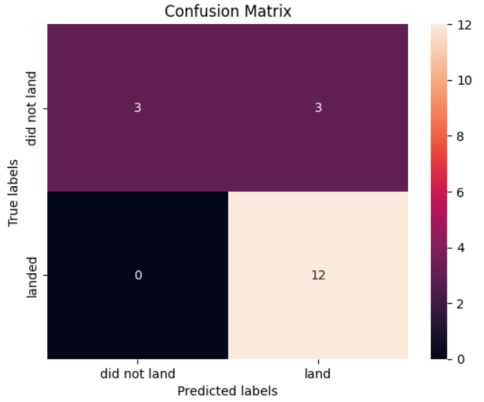
- All the models boast the same accuracy score which exactly is around 83.33% in predicting whether a launch would be successful or unsuccessful. Making them all a good model for prediction.

CONFUSION MATRIX

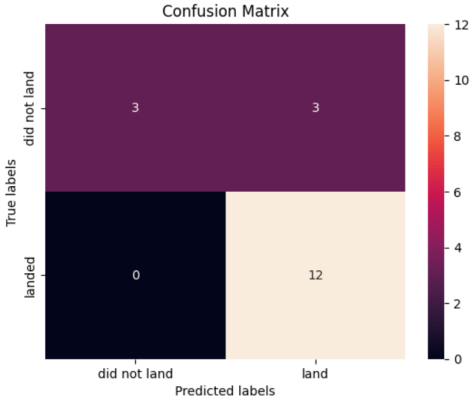
The confusion matrix for all the models were exactly the same:

- All the models predicted 15 successful lands whole only 12 successfully landed
- All the models predicted 3 not to successfully land and all 3 did not successfully land

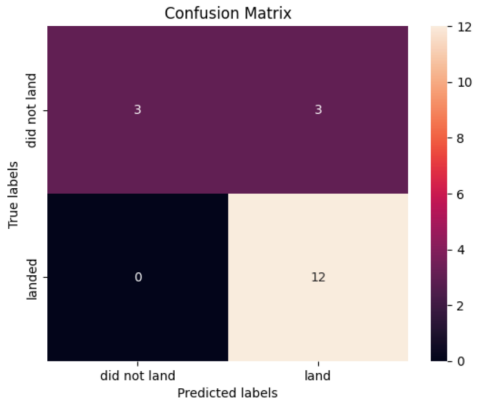
KNN



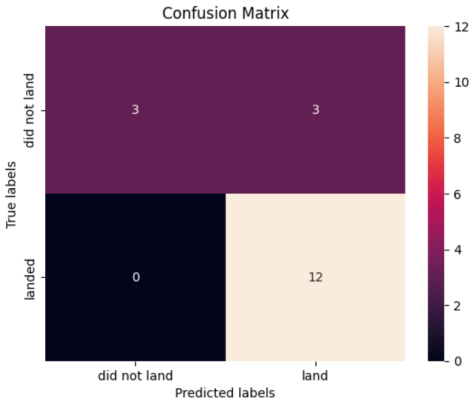
Decision Tree



Logistic Regression

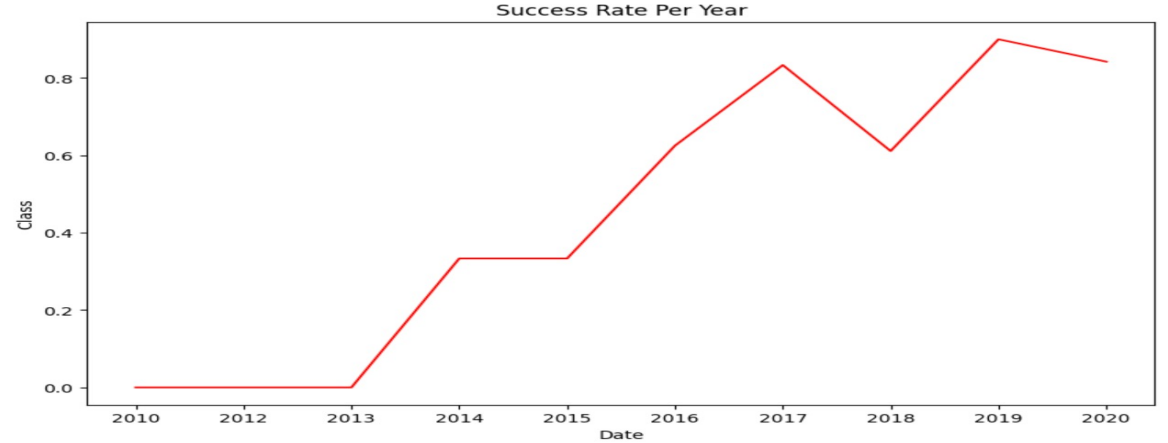


SVM

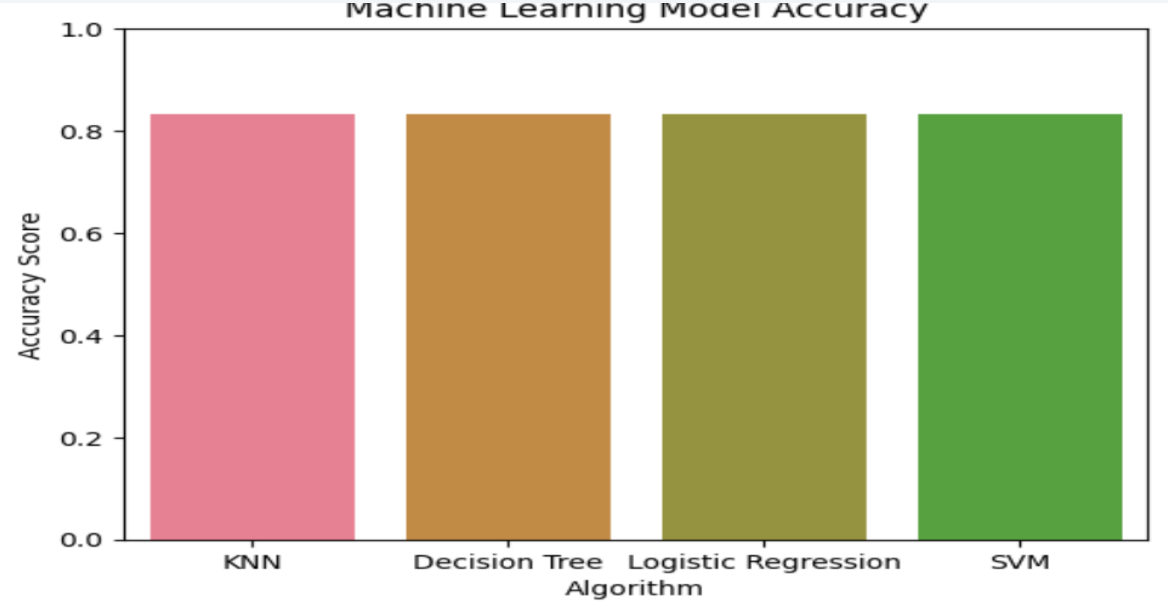


Conclusions

1. SpaceX is looking good for a future investment as it is increasing its success rate over the years



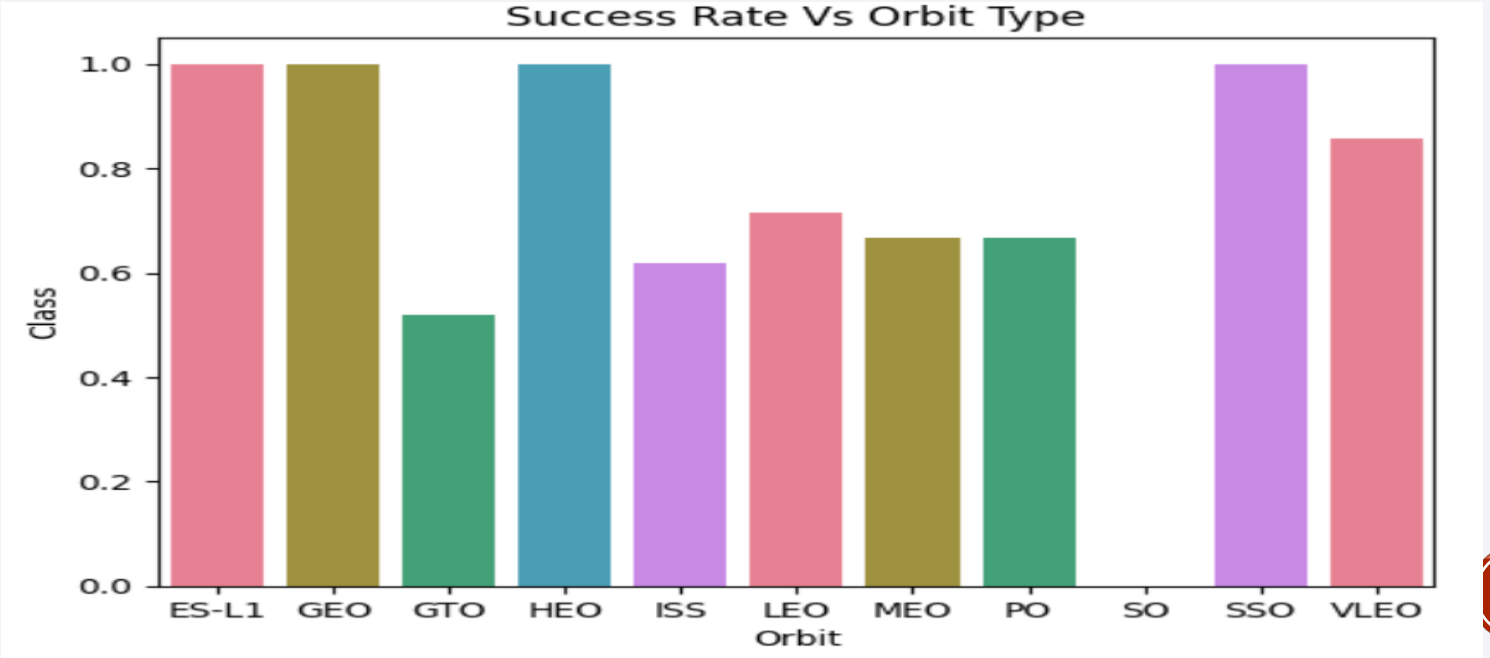
2. All our machine learning models boast a 83% prediction accuracy

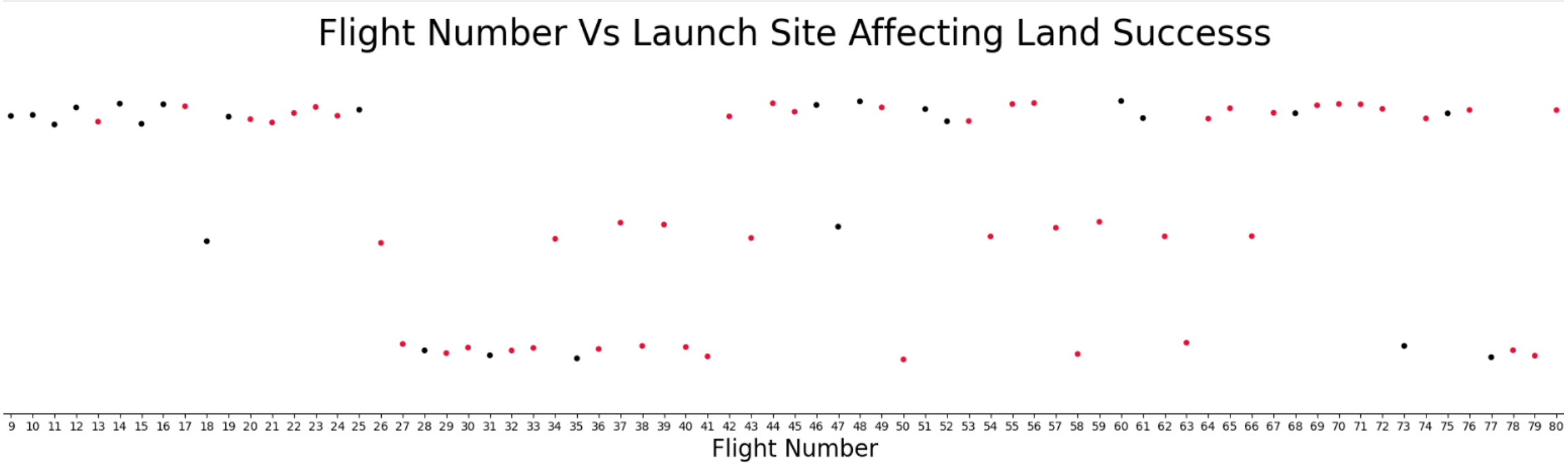


Conclusions (Continued)

3. The KSC LC-39A has the highest launch success ratio at 76.9%

4. The ES-L1, GEO, HEO, SSO lead to the highest success rate

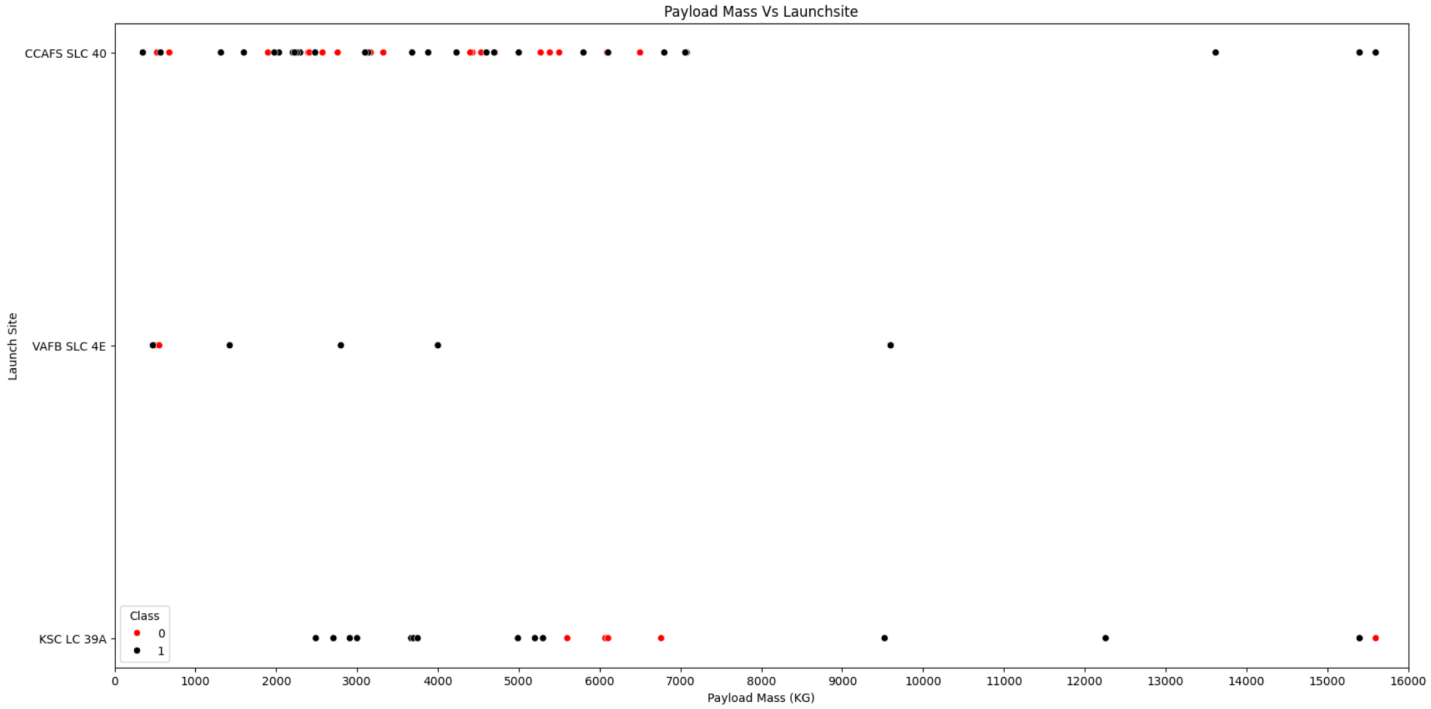




CONCLUSIONS (CONTINUED)

We see that flight numbers higher then 30, success rate increases for all Launch Sites.

APPENDIX



- We see that Payload Mass over 7000 KG leads to an almost perfect success rate in all the following Launch Sites.

Thank you!