IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Pol Monné Parera
January 31, 2026

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection
  - Data wrangling
  - EDA with data visualization
  - EDA with SQL
  - Interactive visual analytics using Folium
  - interactive dashboard for visual analytics using Plotly Dash
  - Predictive analysis with classification models
- Summary of all results
  - Results of the Exploratory Data Analysis (EDA)
  - Results of the interactive visual analytics
  - Results of the predictive analysis

Introduction

Background and Context

This project marks the end of a journey of learning through the different courses offered by IBM as part of their Data Science Professional certification on Coursera.

Our company, SpaceY is aiming to launch rockets into space in the same way SpaceX has been doing during the past years. This report aims to showcase the findings encountered while evaluating data from the SpaceX program to gain an understanding on how we should direct our journey to obtain better results than the competition.

Problems to Solve

One of the main reasons why SpaceX has been in the spotlight for the past years has been how they have been able to transform the aero spatial industry by lowering the costs of their launches. To put it in perspective NASA's Space Launch System (SLS) costs around $4.1 billion per launch, compared to the $67 million that SpaceX's launches cost.

The main reason for this cost difference comes from the fact that SpaceX has found a way to reuse parts of their rockets by landing them safely instead of discarding them. We aim to uncover how to predict if the rocket parts will make their way back and be able to be reused (saving a ton of money).

Section 1

Methodology

Methodology - Executive Summary

- Data collection methodology:

  Data was collected from two main sources:

  - SpaceX open-source Rest API
  - Web scraping data from Wikipedia (List of Falcon 9 and Falcon Heavy launches)

- Perform data wrangling

  Empty or unnecessary data was removed from the dataset. Furthermore, categorical data was transformed using One-Hot Encoding.

- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models

  The classification models implemented for this task consisted of Logistic Regression, K-Nearest Neighbors, Support Vector Machines and Decision Trees.

Data Collection

SpaceX Rest API

- Retrieved data from the SpaceX Rest API with a GET request.
- Limited the data to only include Falcon 9 launches.
- Missing payload mass data was replaced with the mean value for that column.

Wikipedia Web Scraping

- Accessed the Wikipedia page containing Falcon9 and Falcon Heavy launch data.
- Used BeautifulSoup to find and parse the table with the necessary data in the HTML document.
- Transformed the extracted data into a Pandas dataframe.

Data Collection – SpaceX API - P1

To make the requested JSON results more consistent, we will use the following static response object for this project:

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

We should see that the request was successfull with the 200 status response code

```
response=requests.get(static_json_url)
```

```
response.status_code
```

```
200
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

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

Using the dataframe `data` print the first 5 rows

```
# Get the head of the dataframe
data.head()
```

static_fire_date_utc	static_fire_date_unix	tbd	net	window	rocket	success	details	crew	ships	capsules

Obtained and parsed the data from the Rest API via a GET request

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'Booster File display 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}
```

Then, we need to create a Pandas data frame from the dictionary launch_dict.

```
# Create a data from launch_dict
df = pd.DataFrame(launch_dict)
```

Show the summary of the dataframe

```
# Show the head of the dataframe
df.head()
```

FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCo

Constructed the dataset using the previously obtained data

My GitHub repository: [Lab01-jupyter-labs-spacex-data-collection-api.ipynb]

Data Collection – SpaceX API – P2

```
# Hint data['BoosterVersion']!='Falcon 1'
mask = df["BoosterVersion"] == "Falcon 9"
data_falcon9 = df[mask]
data_falcon9.head()
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCo
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	
5	8	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	
6	10	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	
7	11	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None	1.0	
8	12	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	

Now that we have removed some values we should reset the FlgihtNumber column

```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

Filtered the Dataframe to only include Falcon 9 launches

```
# Calculate the mean value of PayloadMass column
payload_mean = data_falcon9['PayloadMass'].mean()

# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, payload_mean)
data_falcon9.isnull().sum()
```

Replaced missing values in Payload Mass for their column's mean

My GitHub repository: [Lab01-jupyter-labs-spacex-data-collection-api.ipynb]

Data Collection - Scraping

```
response = requests.get(static_url, headers=headers)
```

Create a `BeautifulSoup` object from the HTML `response`

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

Print the page title to verify if the `BeautifulSoup` object was created properly

```
# Use soup.title attribute
print(soup.title.string)
```

```
List of Falcon 9 and Falcon Heavy launches - Wikipedia
```

Requested the Falcon 9 Launch Wiki page from its URL

File display

```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
for th in first_launch_table.find_all('th'):
    name = extract_column_from_header(th)

    if name is not None and len(name) > 0:
        column_names.append(name)
```

Check the extracted column names

```
print(column_names)
```

```
['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome']
```

Extracted the column names from the HTML table headers

My GitHub repository: [Lab02-jupyter-labs-webscraping.ipynb]

Data Collection - Scraping

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

Created an empty dictionary with keys from the extracted column names

```
extracted_row = 0

# 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 if first table heading is a flight number
        if rows.th:
            if rows.th.string:
                flight_number = rows.th.string.strip()
                flag = flight_number.isdigit()
        else:
            flag = False
        # Get table data cells
        row = rows.find_all('td')
        # If it is a valid flight number, save data
        if flag:
            extracted_row += 1

            # Flight Number
            launch_dict['Flight No.'].append(flight_number)

            # Date and Time
            datatimelist = date_time(row[0])
            date = datatimelist[0].strip(',')
            time = datatimelist[1]
            launch_dict['Date'].append(date)
            launch_dict['Time'].append(time)

            # Booster Version
            bv = booster_version(row[1])
            if not bv:
                bv = row[1].a.string
            launch_dict['Version Booster'].append(bv)

            # Launch Site
            launch_site = row[2].a.string
            launch_dict['Launch site'].append(launch_site)

            # Payload
            payload = row[3].a.string
            launch_dict['Payload'].append(payload)

            # Payload Mass
            payload_mass = get_mass(row[4])
            launch_dict['Payload mass'].append(payload_mass)

            # Orbit
            orbit = row[5].a.string
            launch_dict['Orbit'].append(orbit)

            # Customer
            customer = row[6].get_text(strip=True)
            launch_dict['Customer'].append(customer)

            # Launch Outcome
            launch_outcome = list(row[7].strings)[0]
            launch_dict['Launch outcome'].append(launch_outcome)

            # Booster Landing
            booster_landing = landing_status(row[8])
            launch_dict['Booster landing'].append(booster_landing)
```

Filled the dictionary with the scraped data, then converted it to a Dataframe

My GitHub repository: [Lab02-jupyter-labs-webscraping.ipynb]

Data Wrangling

Use the method `value_counts()` on the column `LaunchSite` to determine the number of launches on each site:

```
# 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
```

Calculated the number of launches per site

Use the method `.value_counts()` on the column `Outcome` to determine the number of `landing_outcomes` .Then assign it to a variable landing_outcomes.

```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
landing_outcomes
```

File display

```
Outcome
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
Name: count, dtype: int64
```

Calculated the number and occurrence of mission outcome of the orbits

Use the method `.value_counts()` to determine the number and occurrence of each orbit in the column `Orbit`

Note: Do not count GTO, as it is a transfer orbit and not itself geostationary.

```
# Apply value_counts on Orbit column
df["Orbit"].value_counts()
```

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

Calculated the number and occurrence of each orbit

Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome`; otherwise, it's one. Then assign it to the variable `landing_class`:

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = [0 if outcome in bad_outcomes else 1 for outcome in df['Outcome']]
```

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

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

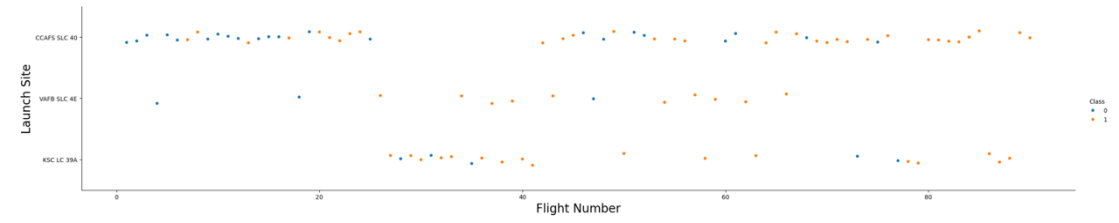
	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

Created a landing outcome label from the Outcome column

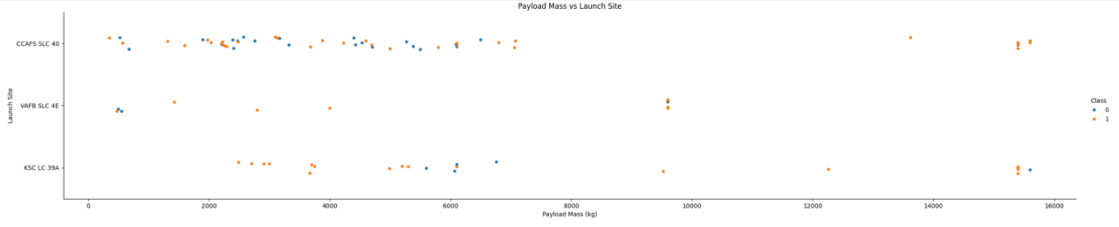
My GitHub repository: [Lab03-labs-jupyter-spacex-Data%20wrangling.ipynb]

EDA with Data Visualization

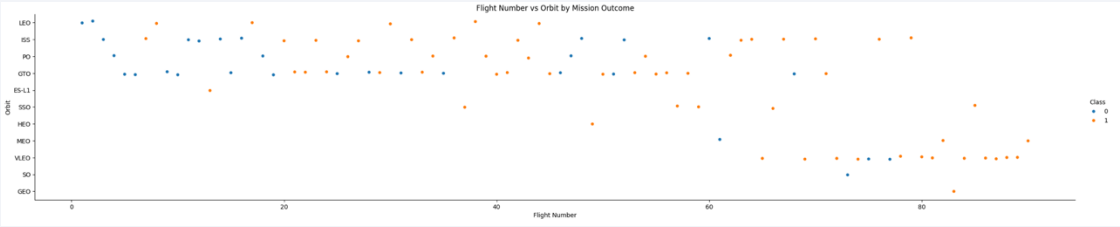
The scatter plots below show the relationship between various variables labeled in the titles



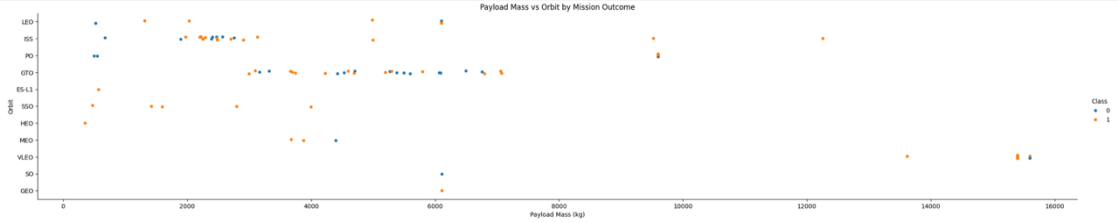
Launch Site
Vs
Flight Number



Launch Site
Vs
Payload Mass (kg)

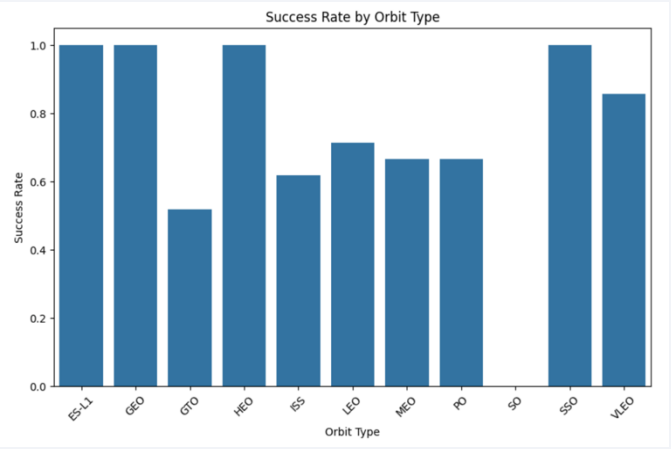


Orbit
Vs
Flight Number

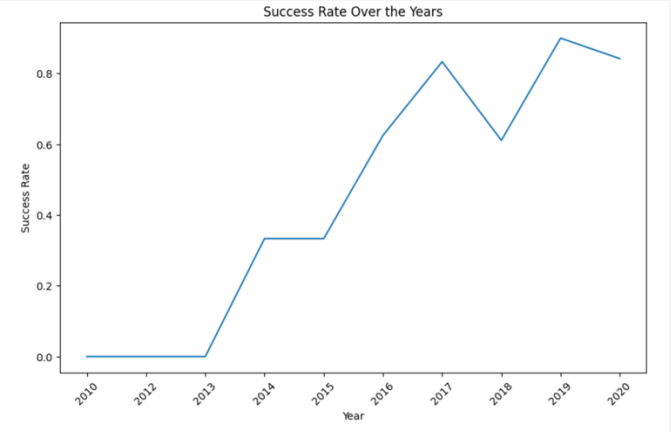


Orbit
Vs
Payload Mass (kg)

Relationship between success rate and orbit type



Launch success over a period of time



My GitHub repository: [Lab05-edadataviz.ipynb]

EDA with SQL

Below you can find a summary of the SQL queries performed for the EDA:

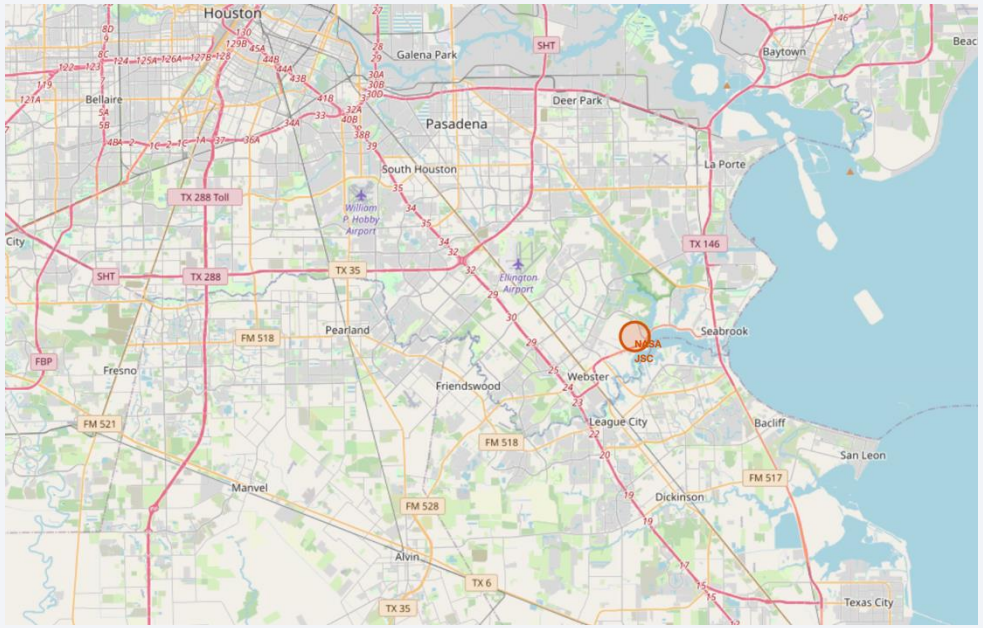
- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the 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 all the booster_versions that have carried the maximum payload mass, using a subquery with a suitable aggregate function.
- 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.

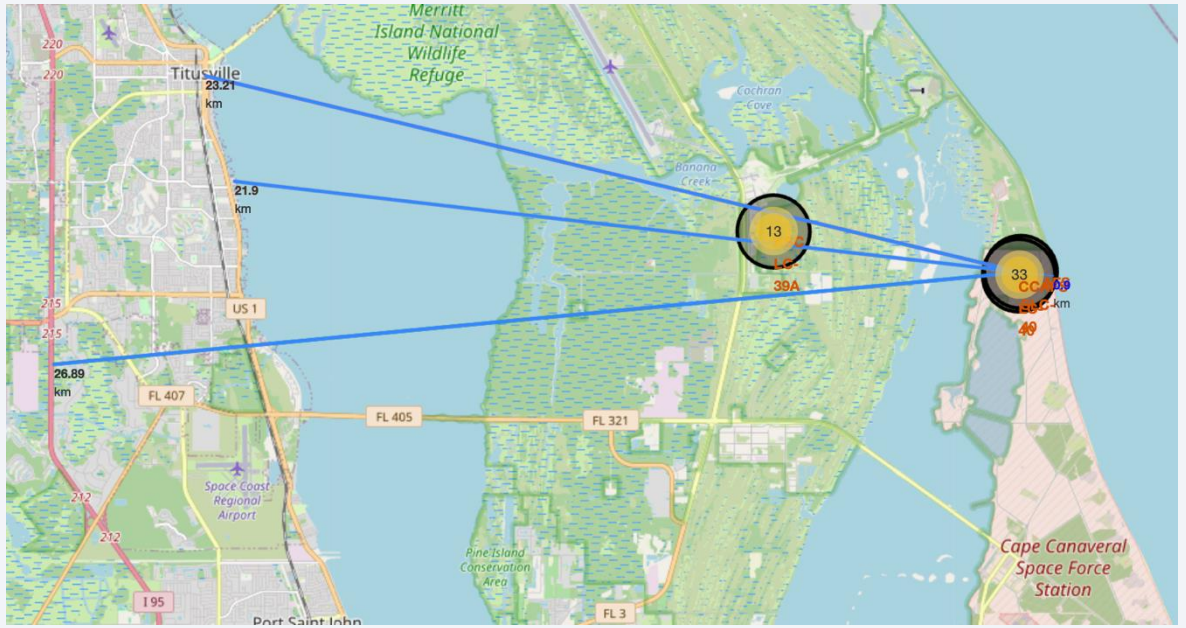
My GitHub repository: Lab04-jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

The following are map objects that have been created to construct an interactive map with Folium

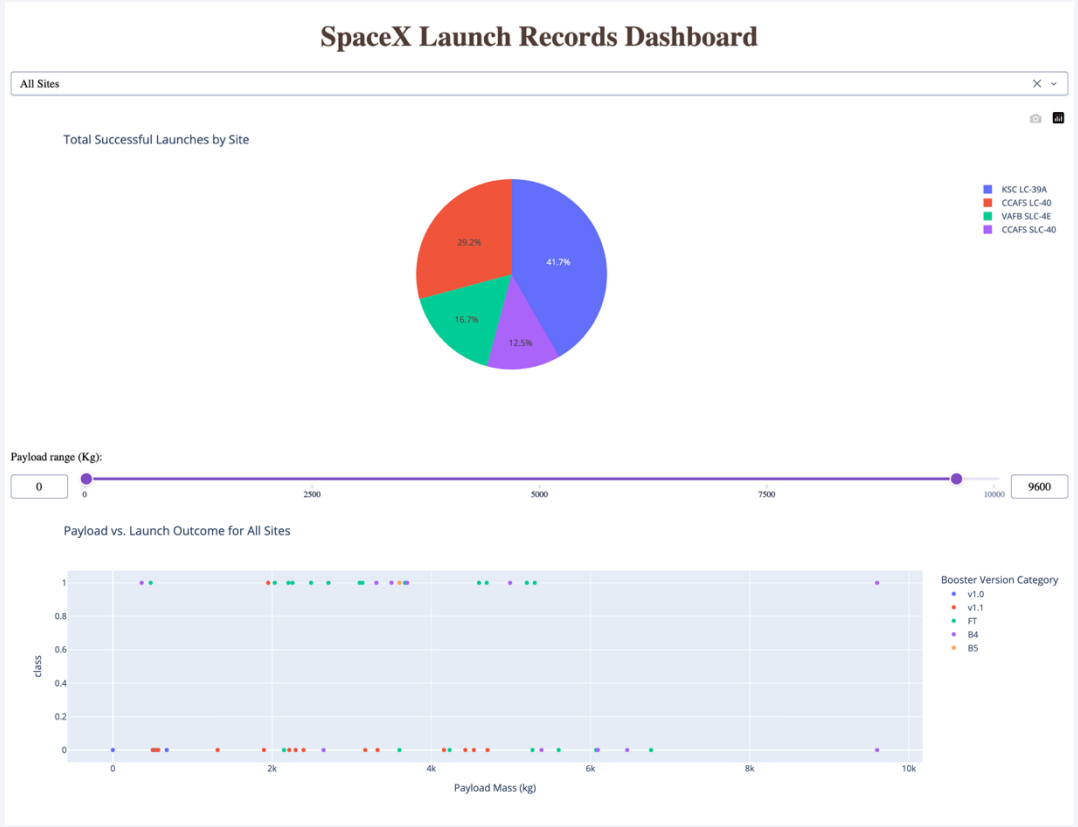
- Marker: To display launch site name in the map
- Circle: To mark the position of the launch site in the map
- Line: To show different metrics in the map (e.g. distance to the ocean, nearest city…)





My GitHub repository: [Lab06-lab_jupyter_launch_site_location.ipynb]

Build a Dashboard with Plotly Dash



The dashboard consists of an interactive pie and scatter plot.

The first one displays information on percentages of successful launches by site and can be updated to show information on a concrete site via the dropdown menu on top.

On the other hand, the scatter plot displays the outcome (success) of a mission against its payload mass in kgs. This second graph can be interacted to show information on concrete or all sites via the same dropdown as the pie chart, but it can also be interacted with by selecting the range of masses to be displayed with the slider on top.

My GitHub repository: [Lab07-spacex-dash-app.py]

Predictive Analysis (Classification)

I have trained four different classification models trying to achieve the best possible accuracy, those being: a KNN, Logistic Regression, SVM and Decision Tree models. All the models where trained in the same previously defined training set and with the use of a GridSearchCV object to determine the optimal parameters.

Finally, the models where tested on their accuracy with the testing set previously defined and, based on the results, Logistic Regression, SVM, and KNN were the best models with all of them achieved similar test accuracies of 83.33%, while the Decision Tree lagged at 61.11%.

Definition of training and testing sets

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

Results of the GridSearchCV

```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2, 'splitter': 'random'}
accuracy : 0.875
```

Accuracy of the models

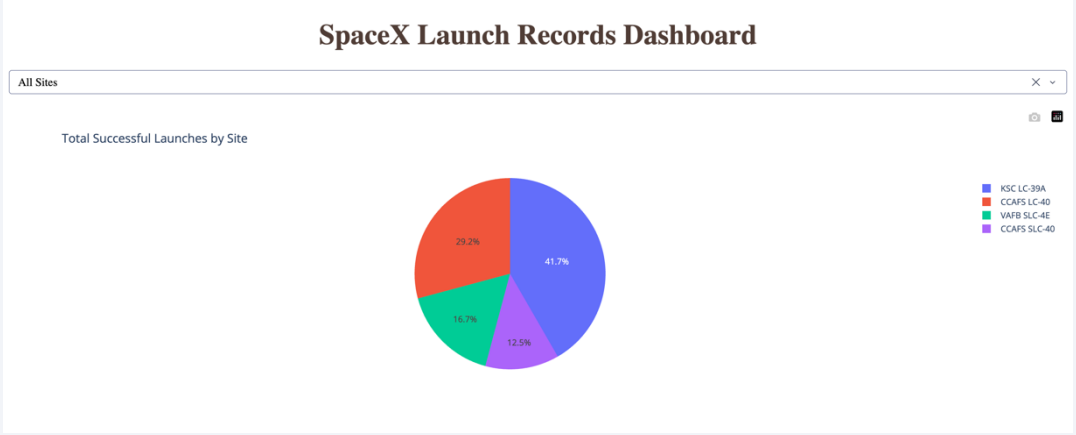
```
print('LR Accuracy:', '{:.2%}'.format(test_accuracy))
print( 'SVM Accuracy:', '{:.2%}'.format(test_accuracy_svm))
print('Decision Tree Accuracy:', '{:.2%}'.format(test_accuracy_tree))
print('KNN Accuracy:', '{:.2%}'.format(test_accuracy_knn))
```

```
LR Accuracy: 83.33%
SVM Accuracy: 83.33%
Decision Tree Accuracy: 61.11%
KNN Accuracy: 83.33%
```


My GitHub repository: [Lab08-SpaceX_Machine Learning Prediction_Part_5.ipynb]

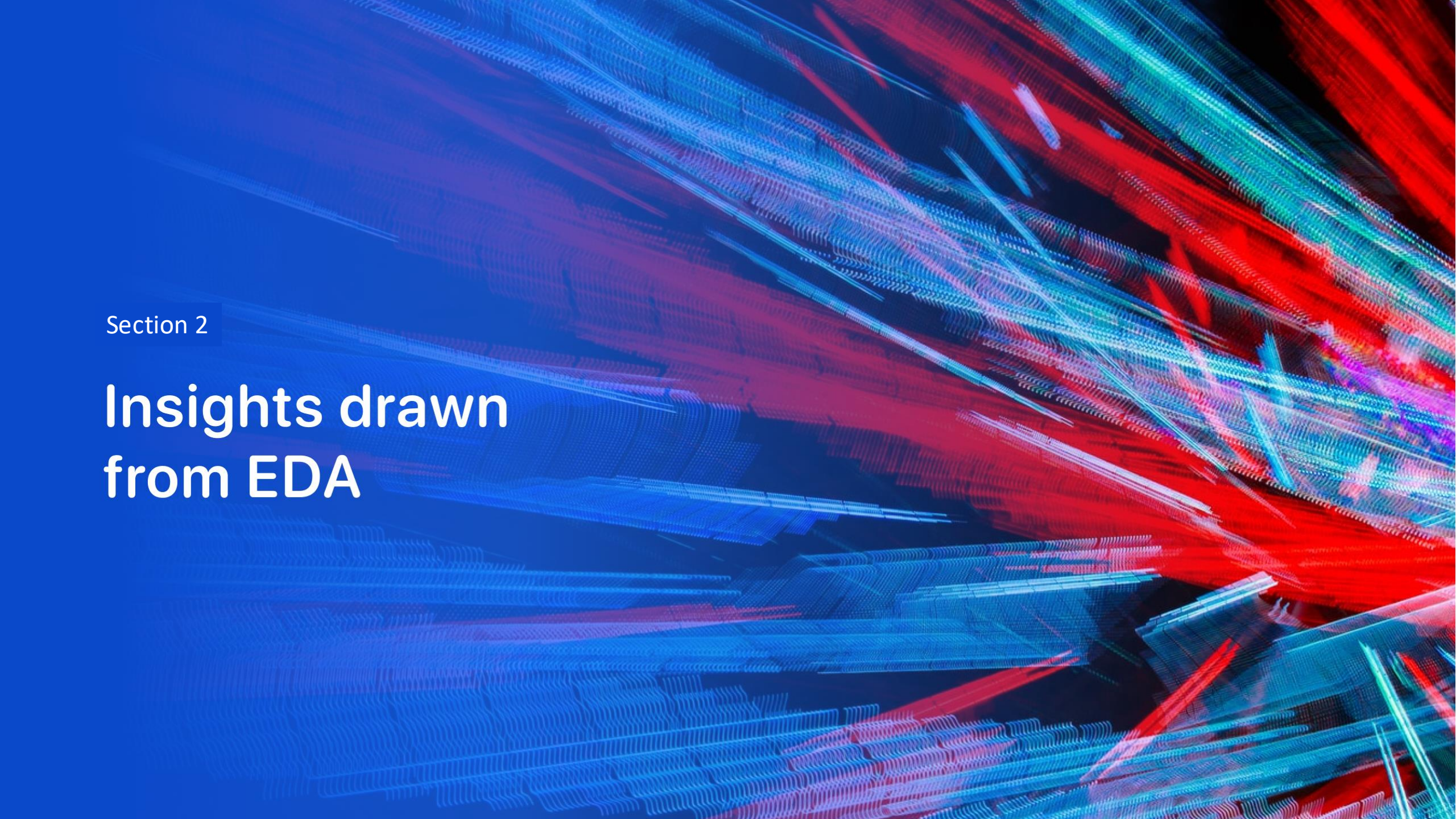
Results

- The GTO orbit is the one with the lowest success rate by difference.
- Success rate has a general increasing rate over the years, showing improvements and perfection of the methods used.
- The launch site with the highest count of success missions is KSC LC-39A (41.7%), followed by CCAFS LC-40 (29.2%).
- The best performing models, by accuracy, are the Logistic Regression, KNN and SVM models. All sharing a 83.3% accuracy on the test set.
- Launchsites seem to be close to all possible methods of transportation, like railways, highways and costlines.



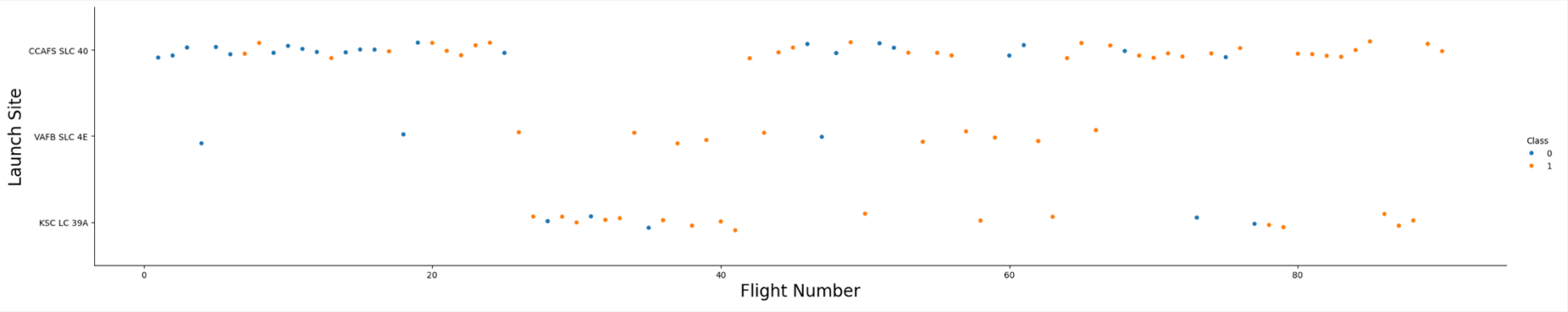
Section 2

Insights drawn from EDA



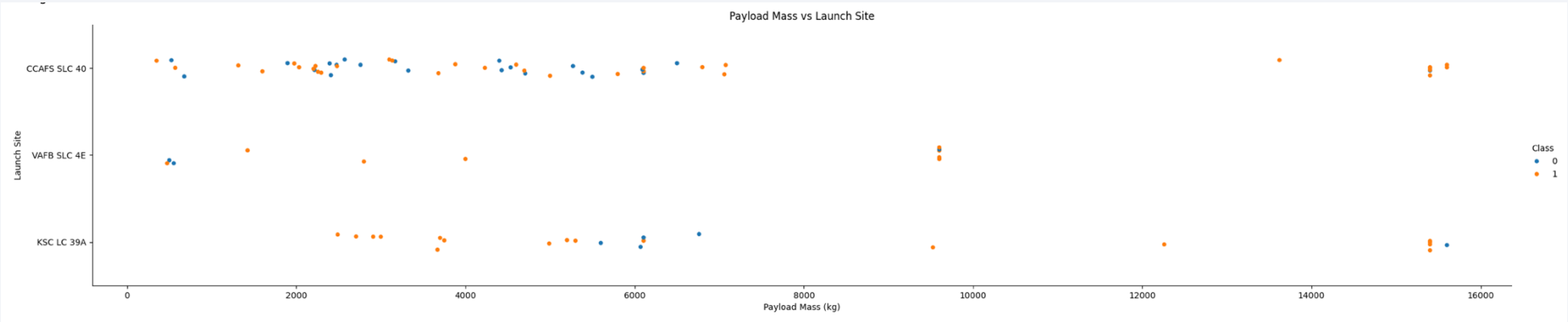
Flight Number vs. Launch Site

Launch site CCAFS SLC 40 is the most active one, having most launches out of the three launch sites studdied. Nevertheless, the rate of success (blue colored points being a failure and orange ones success) does not seem to be high, with a lot of failed instances at lower flight numbers.



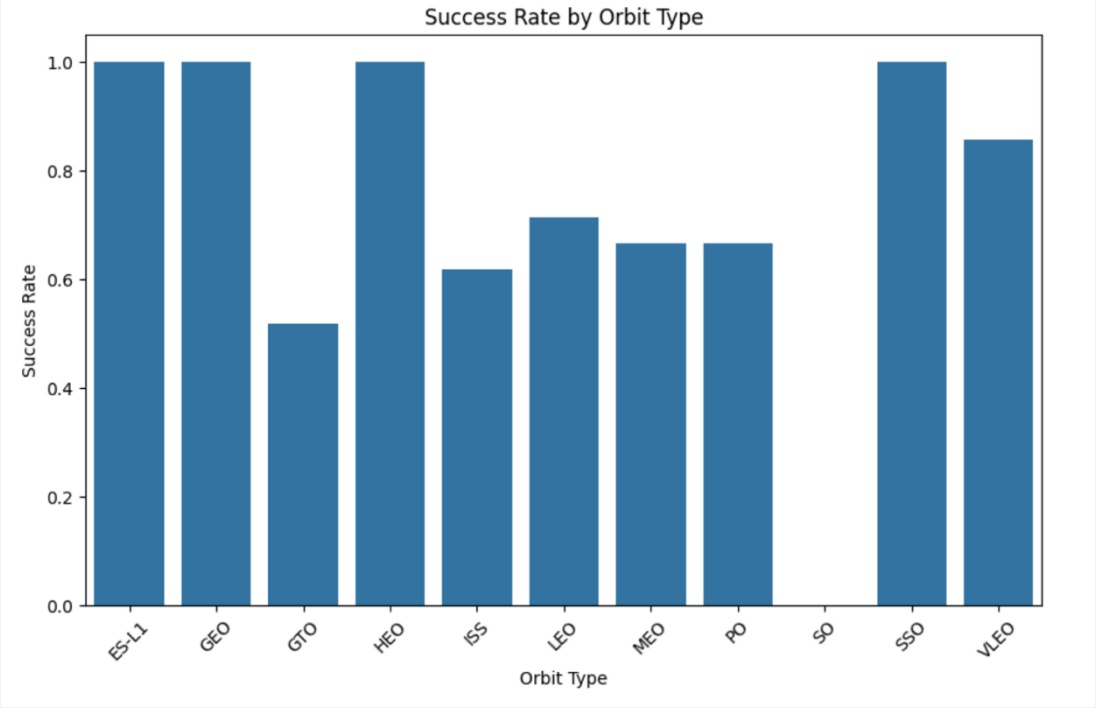
Payload vs. Launch Site

The first half of the X axis in the scatter plot has a much higher density than the second half. Launches tend to have lower payloads, with few instances bringing more than 7000kg.



Success Rate vs. Orbit Type

The orbits with the highest success rate are ES-L1, GEO, HEO and SSO, with a value close to a 100%. On the other hand, the orbit with the lowest success rate is GTO, with a mission success of around 50%.

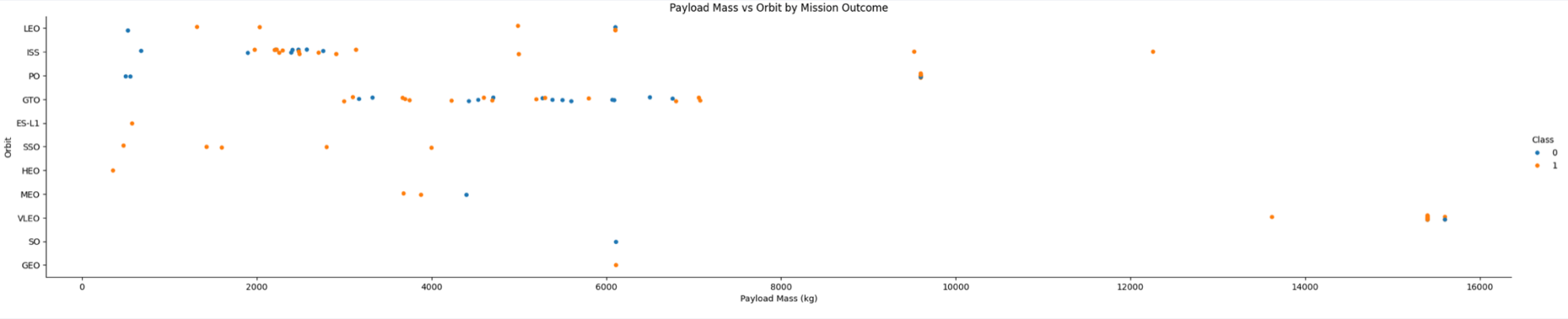


Flight Number vs. Orbit Type

GTO and ISS are the orbits with the biggest amount of launches during the first 60 to 70 launches, accompanied by LEO and PO on a lower scale. On the latest launches VLEO orbits are the most dominant.

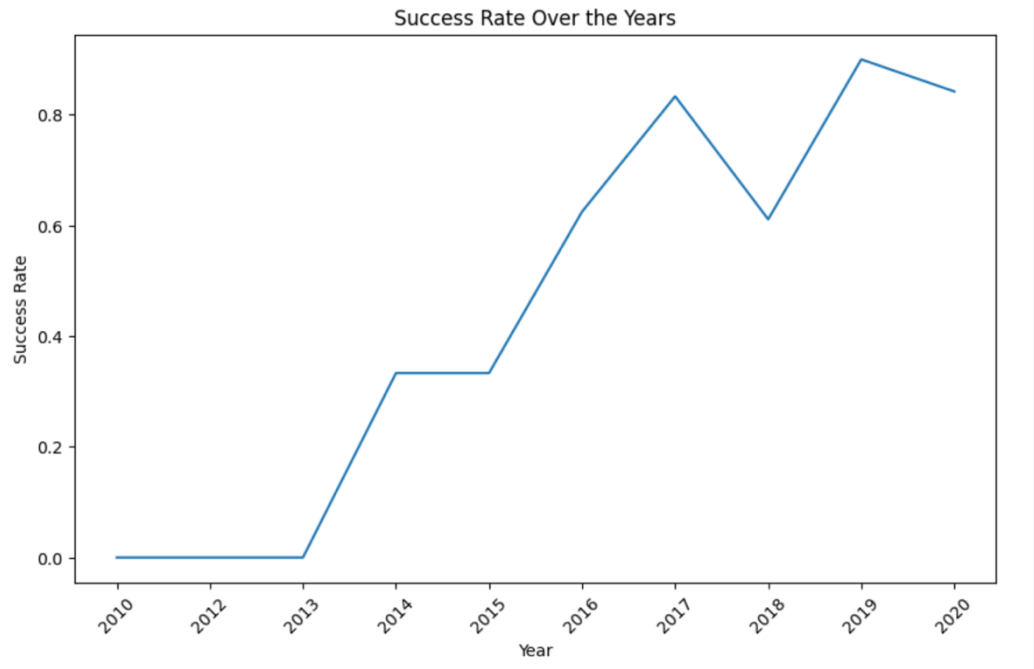
Payload vs. Orbit Type

Most missions cluster at low to mid payload masses (roughly below 7,000 kg), especially in LEO, ISS, and GTO orbits. Very high payload masses (above ~10,000 kg) are rare and appear almost exclusively in VLEO, with both successful and unsuccessful outcomes present, suggesting payload mass alone doesn't determine mission success.



Launch Success Yearly Trend

From 2013 to 2020 the success rate has kept an almost never stopping increasing rate, showing improvements and perfection of the methods used. On 2018, the success rate dipped from around 0.8 to aproximately 0.6, but the next year it fastly recovered to a value above 0.8.



All Launch Site Names

The table below was obtained by executing an SQL query designed to select all distinct launchsites from the SpaceX table in a Jupyter Notebook environment. It allowed to identify that only four distinct launch sites exist in the dataset.

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

Launch Site Names Begin with 'CCA'

The table below was obtained by executing an SQL query designed to select 5 records where launch sites begin with the string 'CCA' from the SpaceX table in a Jupyter Notebook environment.

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

The table below was obtained by executing an SQL query designed to obtain the total payload mass carried by boosters launched by NASA (CRS) from the SpaceX table in a Jupyter Notebook environment. It allowed to observe that over all the missions carried out for the selected customer more than 45 tons of material where transported.

Total Payload Mass
45596

Average Payload Mass by F9 v1.1

The table below was obtained by executing an SQL query designed to obtain the average payload mass carried by booster version F9 v1.1 from the SpaceX table in a Jupyter Notebook environment.

AVG(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

The table below was obtained by executing an SQL query designed to obtain the date when the first succesful landing outcome in ground pad was acheived from the SpaceX table in a Jupyter Notebook environment.

First Successful Landing in Ground Pad
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

The table below was obtained by executing an SQL query designed to obtain the names of the boosters which have success in drone ship and have payload mass between 4000 and 6000 from the SpaceX table in a Jupyter Notebook environment.

Booster_Version	PAYLOAD_MASS__KG_
F9 FT B1022	4696
F9 FT B1026	4600
F9 FT B1021.2	5300
F9 FT B1031.2	5200

Total Number of Successful and Failure Mission Outcomes

The table below was obtained by executing an SQL query designed to obtain the total number of successful and failure mission outcomes from the SpaceX table in a Jupyter Notebook environment. With the information obtained we could calculate that the general success rate was a 99%.

Outcome	Total
Failure	1
Success	100

Boosters Carried Maximum Payload

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

The table to the left was obtained by executing an SQL query designed to obtain all the booster_versions that have carried the maximum payload mass from the SpaceX table in a Jupyter Notebook environment.

2015 Launch Records

The table below was obtained by executing an SQL query designed to obtain the records which will display the month names, failed landing outcomes in drone ship, their booster versions, and launch sites for the months in year 2015 from the SpaceX table in a Jupyter Notebook environment.

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

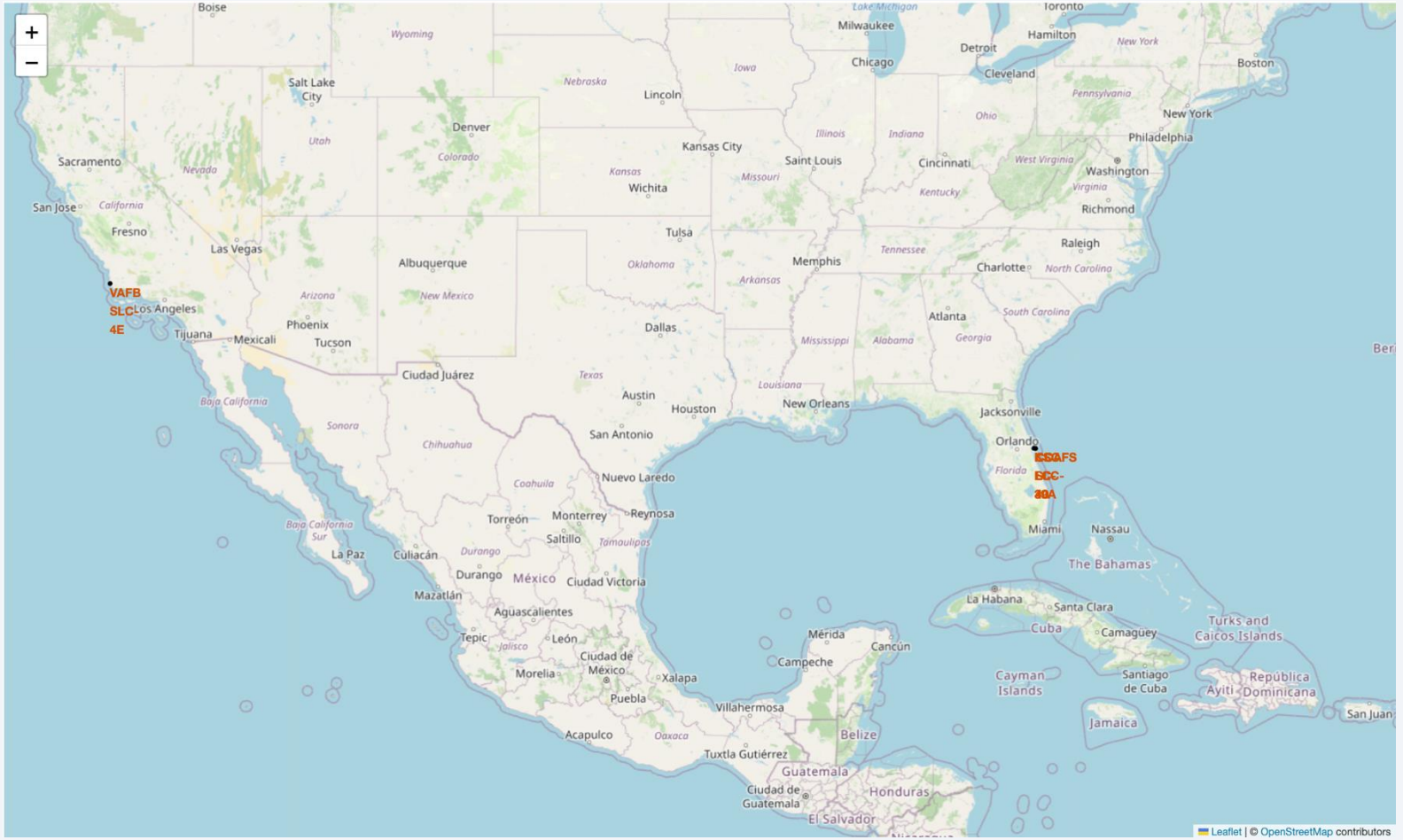
Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

The table below was obtained by executing an SQL query designed to obtain the count of landing outcomes between the date 2010-06-04 and 2017-03-20, in descending order from the SpaceX table in a Jupyter Notebook environment.

Landing_Outcome	Outcome_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

All Launch Sites on a Global Map



On the map we can observe the launch sites with their name labeled in red in the map.

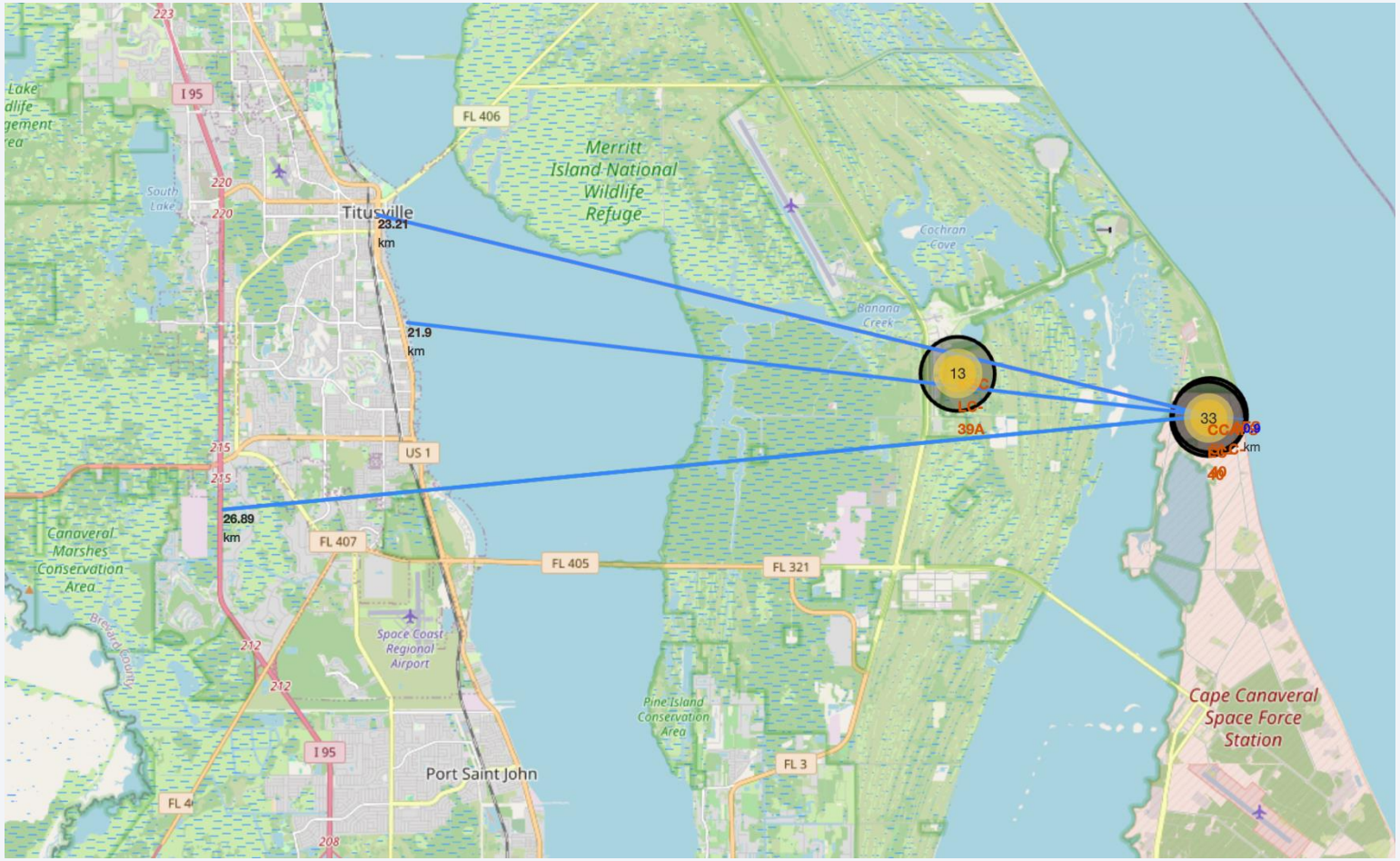
It is interesting to note that all launch sites are next to the coast line, either in the west or east of the United States.

Launch Outcomes per Site

On the map we can observe the launch sites with their launches clustered in the map, when pressing one cluster the detailed information of the launch site is displayed. Green markers represent successful launches and red markers unsuccessful ones.

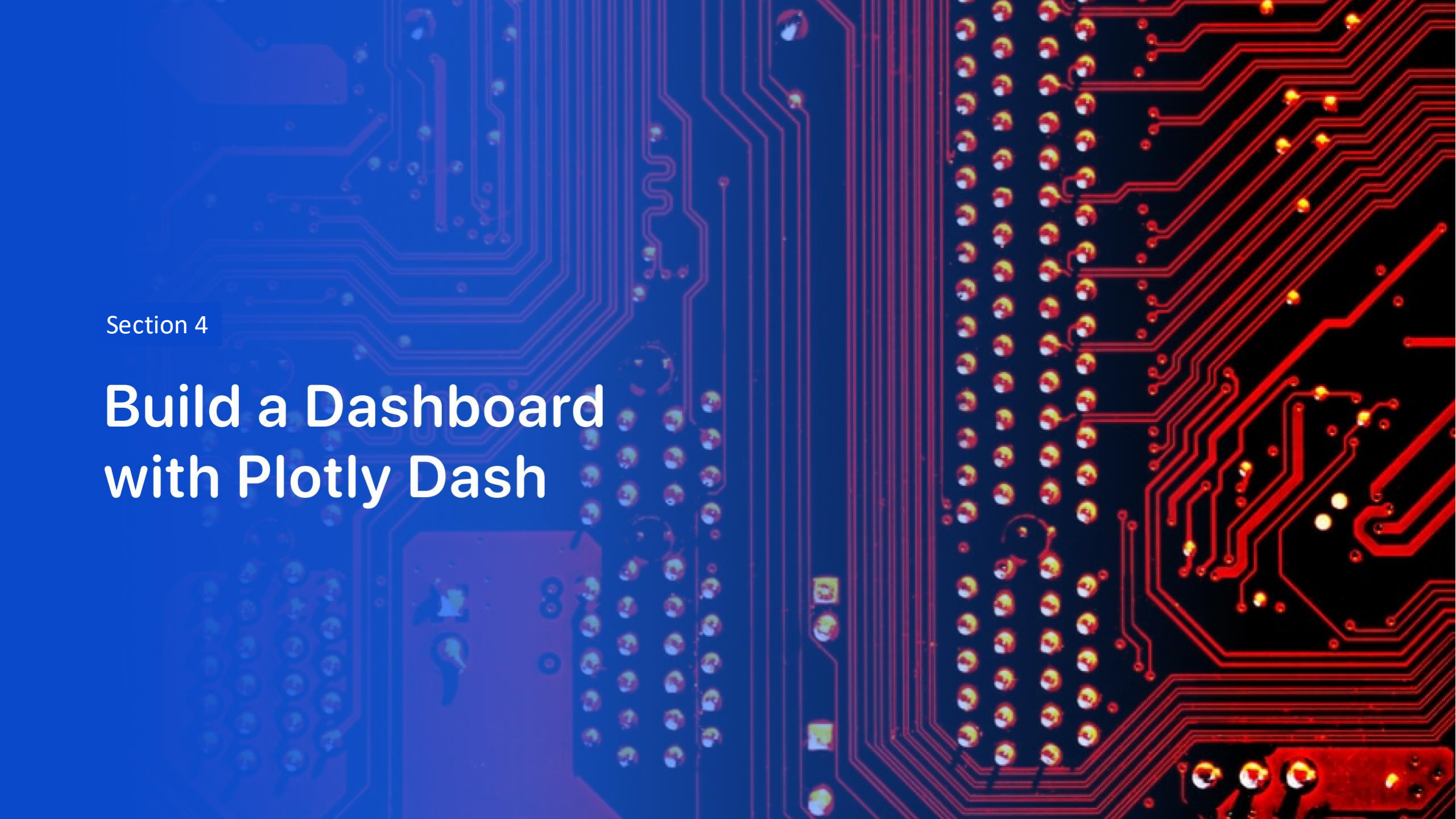
With this method it can be easily identified if the success rate of a launch site is high or low.

Distance from Launch Site to its Proximities



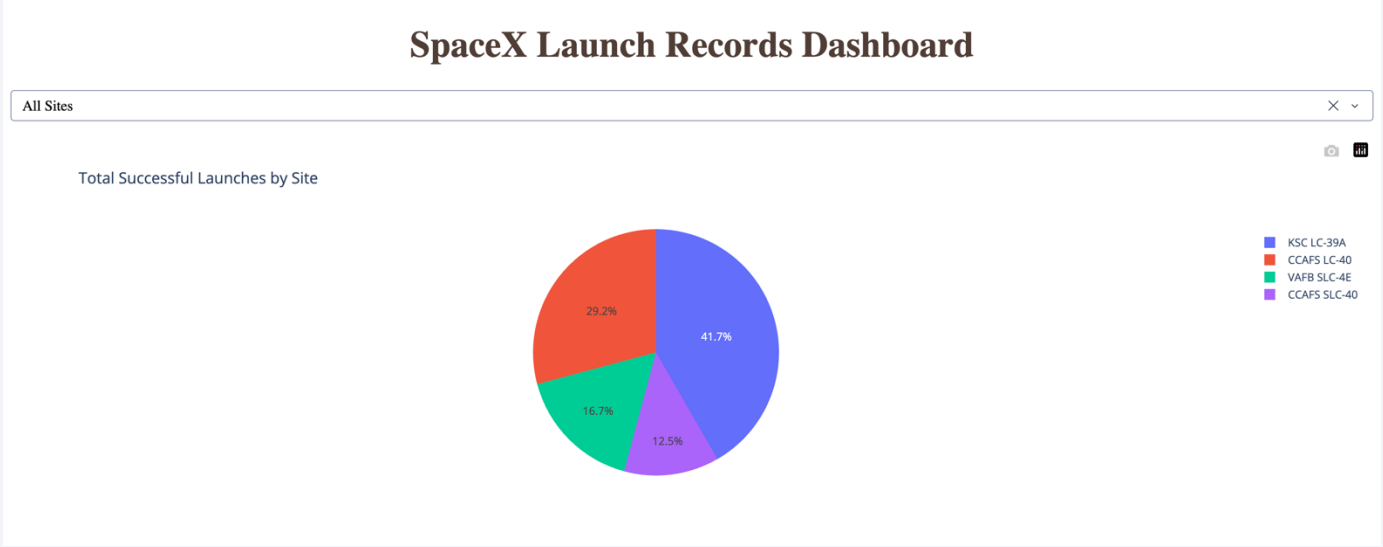
On the map we can observe the direct marker lines between the CCAFS SLC-40 launch site and its proximities, with the end of each line having a text marker displaying the distance in the map.

It can be easily observed how this launch site is relatively close to a city, the railroad, the highway and the coastline.


Section 4
Build a Dashboard with Plotly Dash

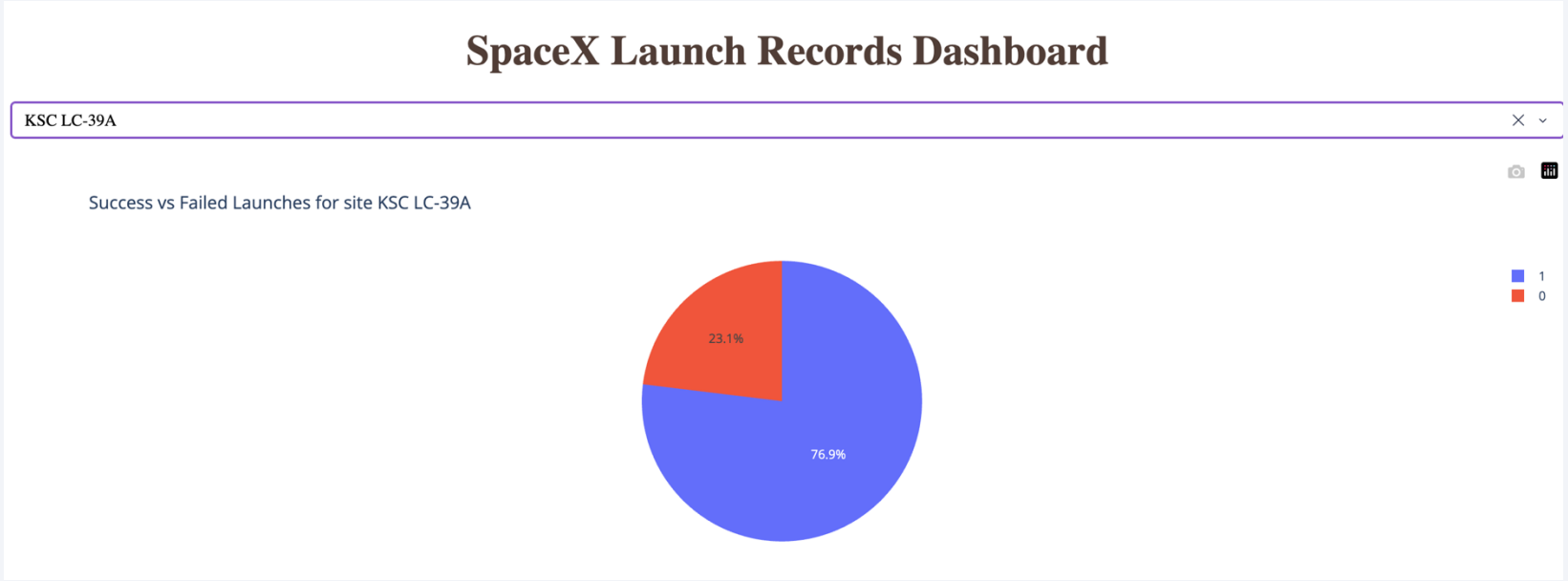
Total Success Launches by Site

In the pie chart below that displays the percentage of the total number of successful launch sites attributed to each site, we can observe how KSC LC-39A has the highest number of successful launches with a 41.7% of the total. On the other hand, we can also observe how CCAFS SLC-40 has the lowest number with just 12.5%.



Success vs Failed Launches for Site KSC LC-39A

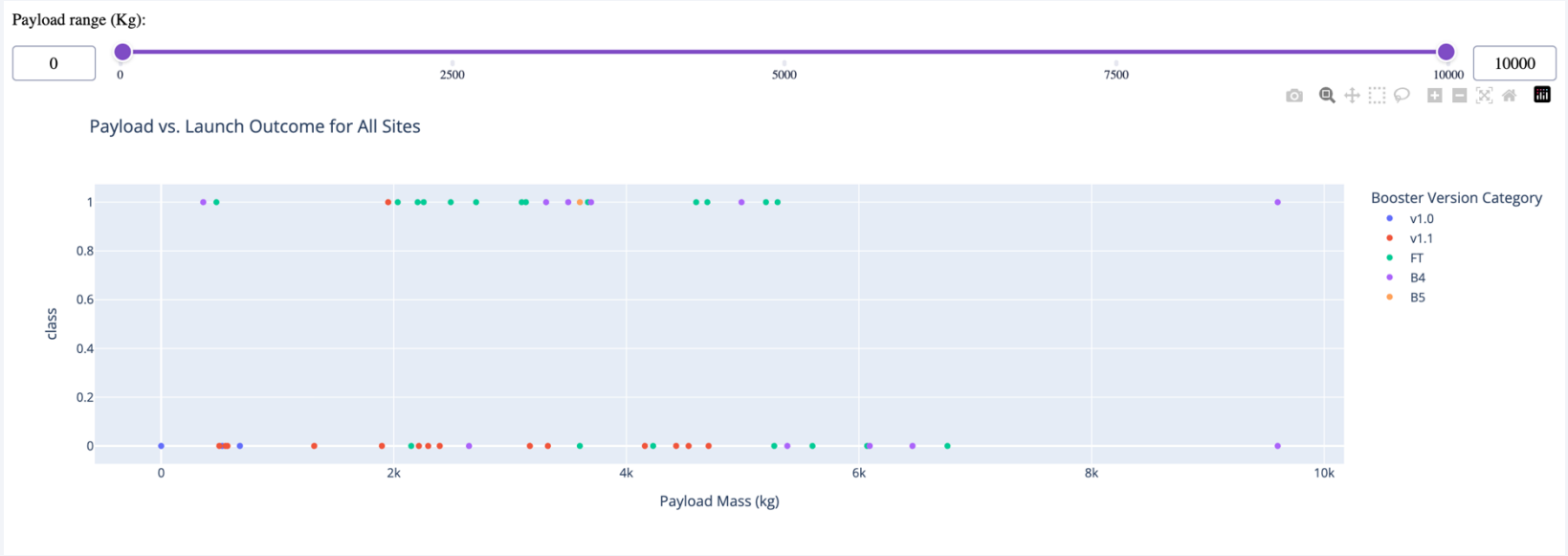
In the pie chart below, which displays the launch site with the highest number of successful launches, we can observe how it has a success rate of 76.9% for all launches in that site. Meaning that around 23 of every 100 launches from KSC LC-39A end in a failure.



Payload vs Launch Outcome for all Sites

In the scatter plot below, the payload range that seems to have the highest success rate is in the range of 2000 to 4000 kilograms. Payloads from 6000 to 10000 kilograms seem to have the lowest success rate.

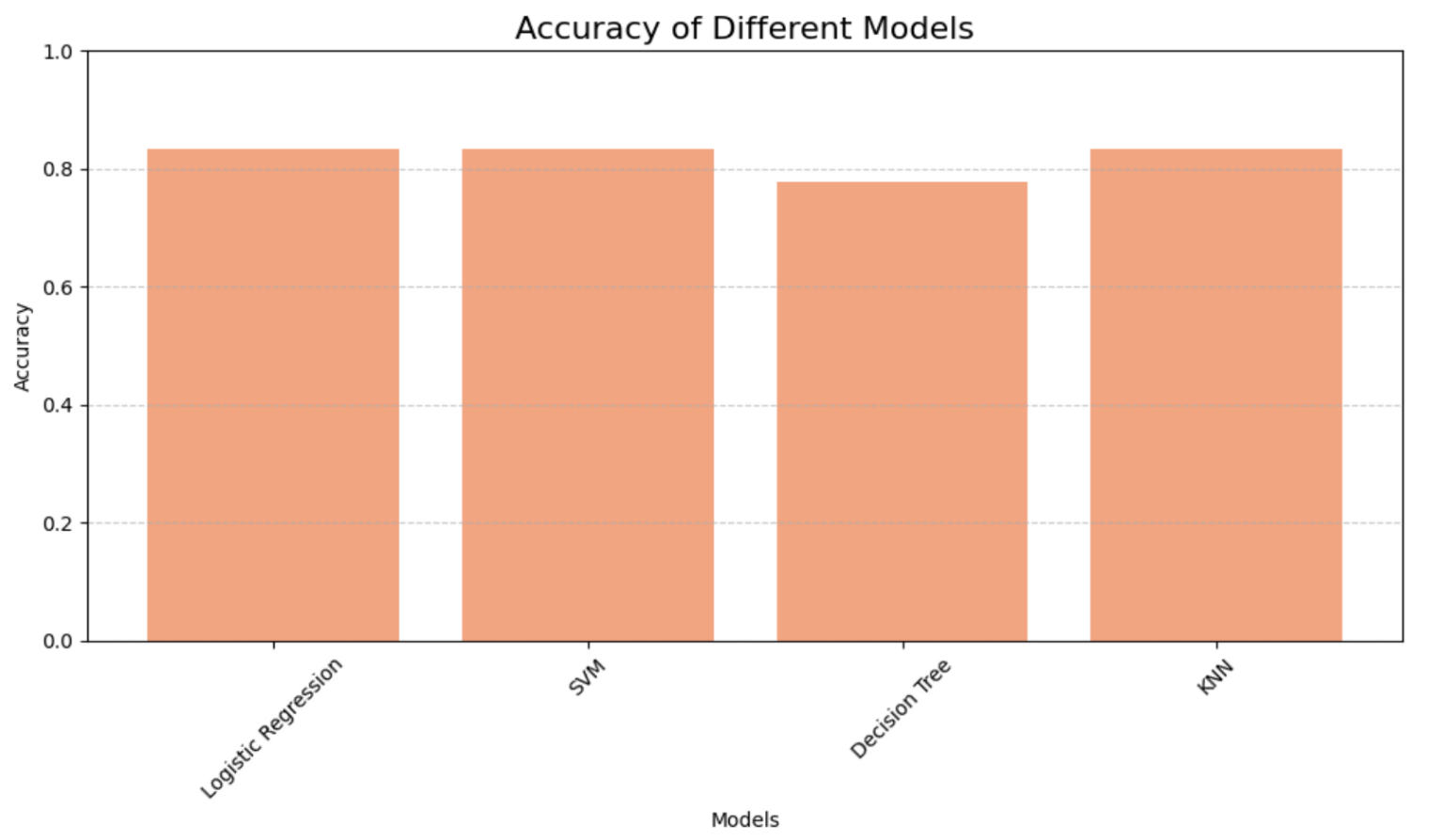
Furthermore, the booster version FT appears to have the highest success rate out of all the versions. On the other hand, booster versions v1.0 and v1.1 are the ones that have the highest failure rate.



Section 5

Predictive Analysis (Classification)

Classification Accuracy



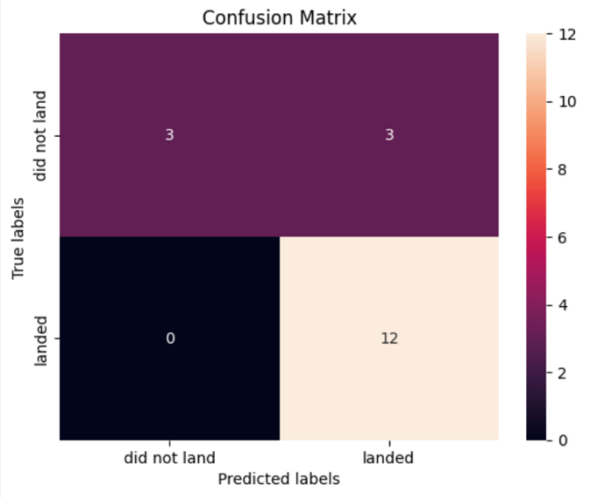

```
LR Accuracy: 83.33%
SVM Accuracy: 83.33%
Decision Tree Accuracy: 77.78%
KNN Accuracy: 83.33%
```

The Logistic Regression, SVM and KNN models all shared the same accuracy value (83.33%). This three models all outperformed equally the decision tree model.

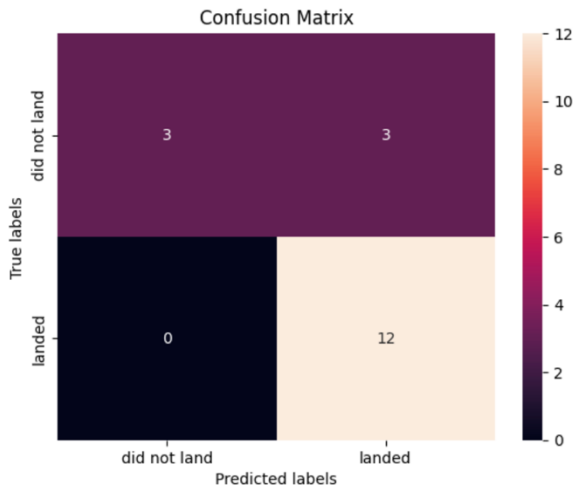
Confusion Matrix

The Linear Regression, SVM and KNN models all performed equally on terms of accuracy on the test set. Therefore, all of them are to be considered as the “best” model given the data at hands.

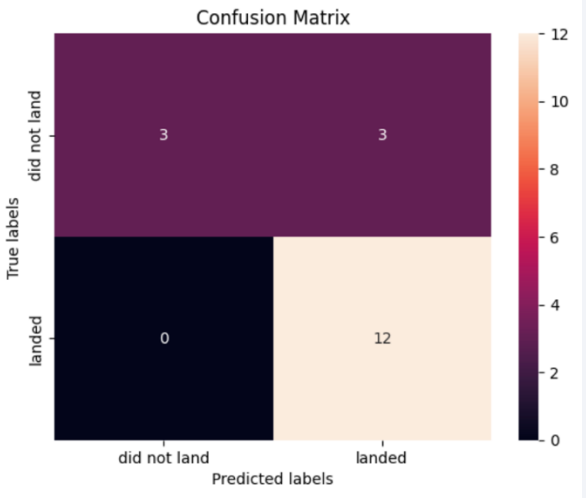
Looking at their confusion matrices the first clear thing is that they did not only performed equally in terms of accuracy, but they also did in terms of how they distributed the instances in their confusion matrices. These models show a failure by predicting three false positives, while they do not mistake any prediction on the negative class.



LR



SVM



KNN

Conclusions

- SpaceX has been constantly improving their launches success rate for the past decade.
- Lower payload masses have a higher success rate than heavier launches.
- The orbits with the highest success rate are ES-L1, GEO, HEO and SSO
- Launch sites are close to the ocean and other strategic sites.
- KSC LC-39A has the highest number of successful launches.
- Logistic Regression, SVM and KNN models work good at predicting landing outcomes.
- A decision tree model is not the best for predicting landing outcomes.

Thank you!