IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Stefanos Vasileiadis
07-01-2024

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection via API, Web Scraping
  - Exploratory Data Analysis (EDA) with Data Visualization
  - EDA using SQL via sqlite3 in Jupyter Notebook
  - Plotting interactive maps with Folium
  - Creating interactive dashboards with Plotly Dash
  - Predictive Analysis by using 4 different ML models
- Summary of all results

Introduction

- **Project background and context**
  - The project's objective is to use metrics such as the **payload mass**, **flight number**, and **orbit** to forecast the successful landing of the Falcon 9 first stage. According to SpaceX's website, a Falcon 9 rocket launch costs 62 million dollars, significantly less than other providers charging over 165 million dollars per launch. The cost difference arises from SpaceX's ability to reuse the first stage. Predicting the landing success helps estimate launch costs, valuable information for companies competing with SpaceX in rocket launches.
- **Problems you want to find answers**
  - Do the aforementioned metrics affect the outcome of the landing stage and if so, how much?
  - Can we trust Machine Learning models for successfully predicting the outcome of a landing?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data collected by using SpaceX REST API and web-scraping information from Wikipedia.
- Perform data wrangling
  - Filling missing values, dropping unnecessary columns, and using one-hot-encoding to make variables useful for classification models.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

Datasets were collected with two main methods

- Obtain information by making a **get request** to the REST API from https://api.spacexdata.com/v4/rockets/. The main procedure is depicted on the flowchart below:

Call SpaceX REST API → API returns JSON file → Transforming JSON file to a pd Data Frame → Cleaning data and exporting the final CSV

- Obtain info from the URL https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches by performing **web scraping**. The procedure is depicted on the flowchart below:

Get HTML response from Wikipedia → Create a BeautifulSoup object and extract data. → Create a Data Frame by parsing the HTML tables → Export the data to a CSV file

Data Collection – SpaceX API

Get response from API

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

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

Convert to JSON file

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

Transform data with the appropriate functions

```
getBoosterVersion(data)
```

the list has now been update

```
BoosterVersion[0:5]
```

```
['Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 9']
```

we can apply the rest of the functions here:

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

```
# Call getPayloadData
getPayloadData(data)
```

```
# Call getCoreData
getCoreData(data)
```

Pass the data into a dictionary

```
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}
```

Use the dictionary to create a Data Frame

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

Filter data

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

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))
```

[Notebook URL]

Replace missing values

```
# 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'].replace(np.nan, payload_mean, inplace=True)
```

Export to CSV

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

Data Collection - Scraping

Notebook URL

Get response from HTML, then create a BeautifulSoup object

```
# use requests.get() method with the provided static_url
response = requests.get(static_url)
# assign the response to a object
soup = BeautifulSoup(response.text, 'html.parser')
```

Find all table in the object

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

Collect columns names

```
column_names = []

th_elements = first_launch_table.find_all('th')

# Extract column names using the provided function and filter out empty names
column_names = [extract_column_from_header(th) for th in th_elements
                if extract_column_from_header(th) is not None and len(extract_column_from_header(th)) > 0]
```

Create a dictionary

```
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']=[]
```

Assigning values to keys

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
```

Create a Data Frame from the dictionary

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

Export to CSV

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Data Wrangling

- The dataset contains a column that specifies if the landing of the booster was successful or not, with outcomes such as True/False Ocean, True/False RTLS, True/False ASDS. Our job is to convert these categorical values to numerical values (0 for failure and 1 for success).
- This procedure is depicted below:

Counting the number of launches for different launch sites

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

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

Counting different types of orbits

```
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
Name: Orbit, dtype: int64
```

Counting the number of different types of landing outcomes

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

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

Assigning numerical values to the categorical labels of landing outcomes

```
landing_class = []

for outcome in df['Outcome']:

    if outcome in bad_outcomes:

        landing_class.append(0)

    else:

        landing_class.append(1)
```

Exporting Data Frame to CSV

```
df.to_csv("dataset_part_2.csv", index=False)
```

[Notebook URL]

EDA with Data Visualization

- **In this section, we used the following types of graphs:**

- **<u>Bar Plots</u>**
  - ➢ Orbit vs Success Rate

- **<u>Scatter Plots</u>**
  - ➢ Flight Number vs Payload Mass
  - ➢ Flight Number vs Launch Site
  - ➢ Payload Mass vs Launch Site
  - ➢ Flight Number vs Orbit
  - ➢ Payload Mass vs Orbit

- **<u>Line Plots</u>**
  - ➢ Year vs Success Rate

[Notebook URL]

EDA with SQL

- **In this section, we used the following SQL queries on Jupyter Notebook to derive several results, such as:**

➢ 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.

➢ 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.

Notebook URL

Build an Interactive Map with Folium

- **We used Folium to plot maps which show the locations of the launch sites and their proximities to different types of locations. More specifically, we used Folium to:**

  - Mark all launch sites on a map.
  - Mark the successful/failed launches for each site on the map.
  - Calculate the distances between a launch site and its proximities.

Notebook URL

Build a Dashboard with Plotly Dash

- **We used Plotly Dash to create an interactive dashboard containing:**

  - A dropdown which gives the option to select a specific launch site or all launch sites.
  - A pie chart which depicts the portions of success and failure for the corresponding launch sites chosen via the dropdown element.
  - A rangeslider which gives the option to select a specific range for the payload mass.
  - A scatter chart which depicts the relation between success and payload mass in the chosen range for the payload mass.

Notebook URL

Predictive Analysis (Classification)

- **In this section, we used Machine Learning models to correctly classify the launch outcomes based on several factors. The process to achieve that was the following:**

  - Load and normalize the data, and split it into train and test sets.
  - Select different ML models and use them as estimators in GridSearchCV.
  - Get the best set of hyperparameters for each model and compute the accuracy of each model on the train and test sets respectively.
  - Finally, compare the accuracy of the model and choose the best one.

Notebook URL

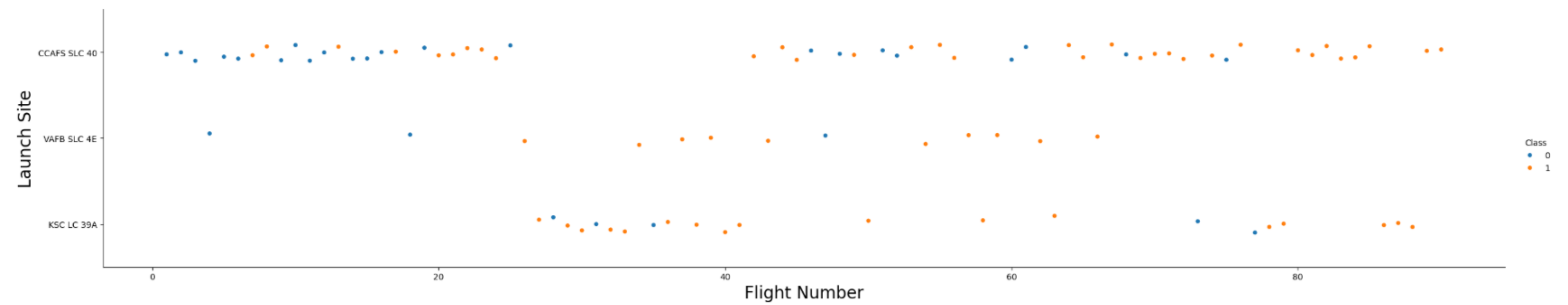
Results

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

Section 2

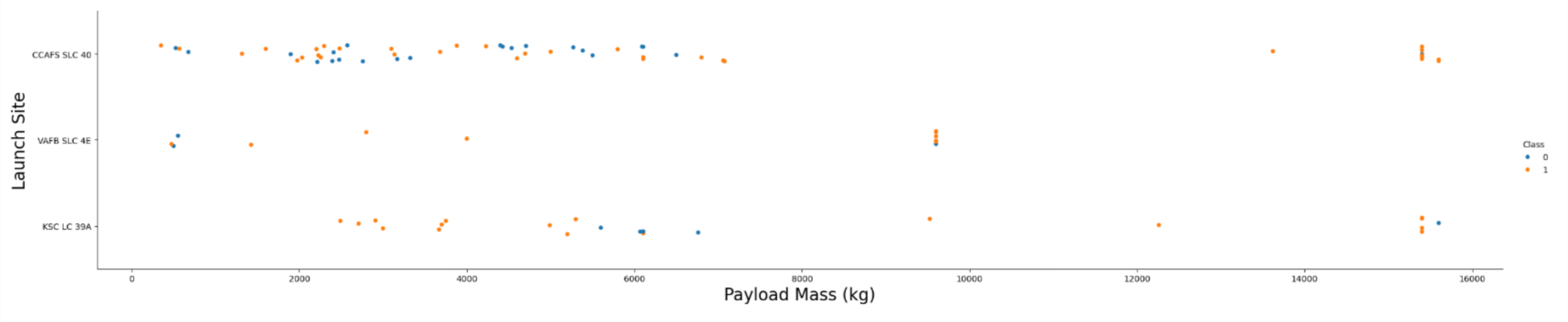
Insights drawn from EDA

Flight Number vs. Launch Site



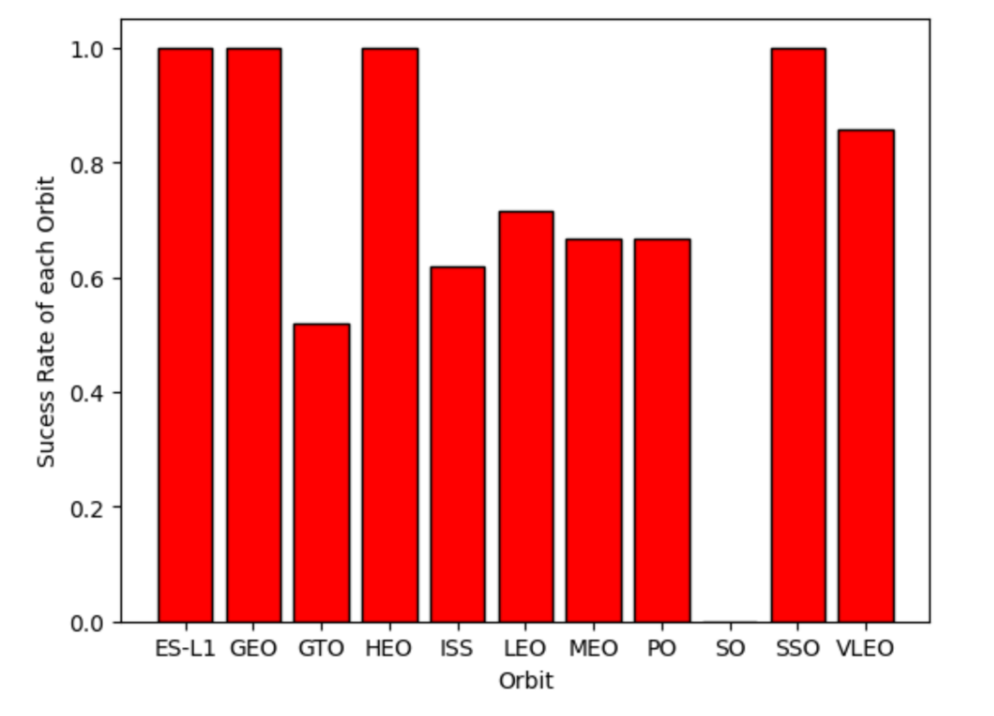
- While there is an increase in the success rate of CCAFS SLC-40 as the Flight Number increases, there is not such an obvious behaviour for the two other Launch Sites

Payload vs. Launch Site



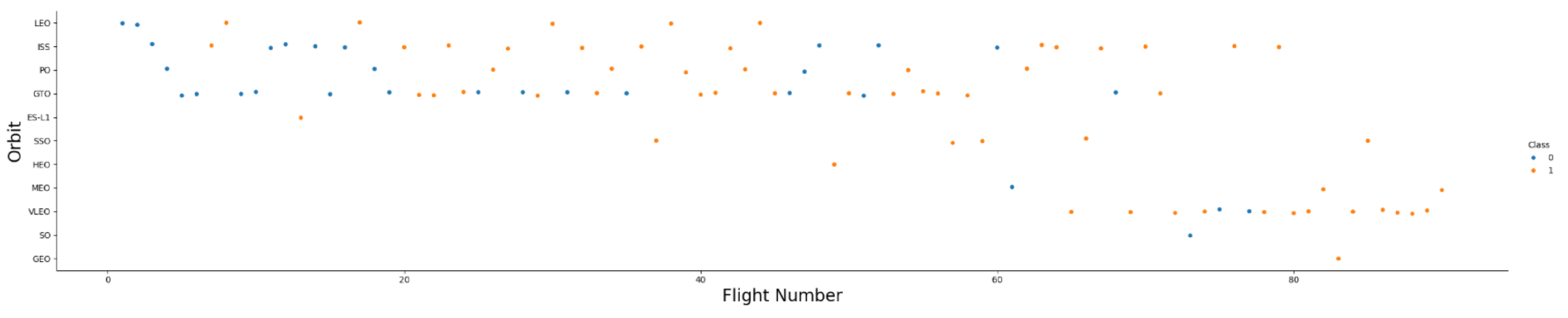
- In some cases, lower payload corresponds to more successful landings, while in others it is the opposite. Besides that, the effect is not always obvious, as there are fluctuations in the success rate as the payload changes, like in CCAFS SLC-40.

Success Rate vs. Orbit Type



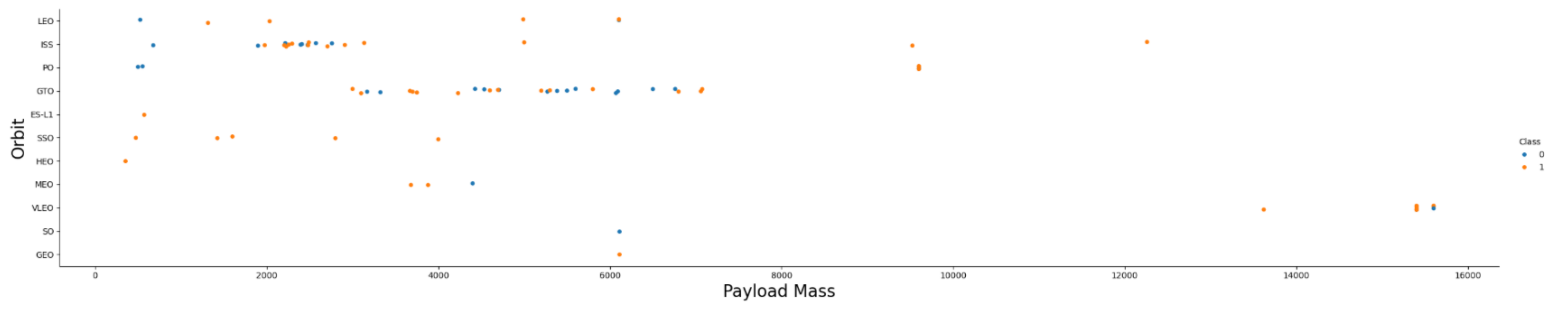
- It can be seen that Orbit Types such as ES-L1, GEO, HEO, SSO have a 100% success rate, while others have a considerably lower success rate.

Flight Number vs. Orbit Type



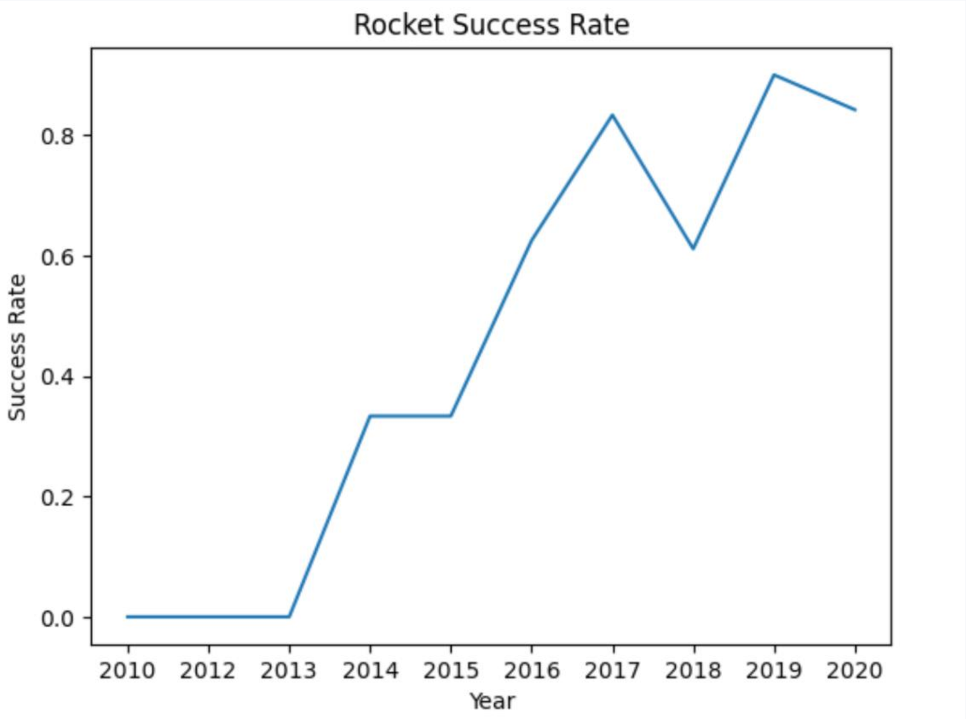
- In most cases, the higher the flight number is, the more likely is the orbit being successful. However, this is not the pattern always, given that orbits like ISS have a more uniform distribution of successes and failures.

Payload vs. Orbit Type



- In some orbits, like HEO, lower payload corresponds to a successful orbit, while in others, like LEO, there are more successful orbits for higher payloads. Overall, this pattern is not clear, especially for orbits like GTO.

Launch Success Yearly Trend



- Besides a sharp decline between 2017 and 2018, we conclude that the SpaceX Rocket success rate keeps increasing on average.

All Launch Site Names

- SQL query:

```
%%sql

SELECT DISTINCT(Launch_Site)
FROM SPACEXTABLE;
```

output

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

- Explanation:

The DISTINCT command gives the unique Launch Sites that appear on the table.

Launch Site Names Begin with 'CCA'

- SQL query:

```
%%sql

SELECT * FROM SPACEXTABLE
WHERE Launch_Site LIKE "%CCA%"
LIMIT 5;
```

output

Date	Time (UTC)	Booster_Version	Launch_Site	Payload
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2

- Explanation:

The LIKE command inside WHERE filters the outputs based on Launch Site names that start with "CCA".

Total Payload Mass

- SQL query:

```
%%sql

SELECT SUM(PAYLOAD_MASS__KG_) AS Total_Payload_Mass
FROM SPACEXTABLE
WHERE Customer = "NASA (CRS)";
```

output

Total_Payload_Mass
45596

- Explanation:

The SUM function gives the total Payload Mass and the WHERE command filters the output based on the customer only being NASA (CRS).

Average Payload Mass by F9 v1.1

- SQL query:

```
%%sql

SELECT AVG(PAYLOAD_MASS__KG_) AS Avg_Payload_Mass
FROM SPACEXTABLE
WHERE Booster_Version LIKE "F9 v1.1";
```

output

Avg_Payload_Mass
2928.4

- Explanation:

The AVG function gives the average Payload Mass and the WHERE command filters the output based on the Booster Version only being F9 V1.1.

First Successful Ground Landing Date

- SQL query:

```
%%sql

SELECT MIN(Date) AS First_Date
FROM SPACEXTABLE
WHERE Landing_Outcome = "Success (ground pad)";
```

output

First_Date
2015-12-22

- Explanation:

The MIN function gives the minimum (or first) Date and the WHERE command filters the output based on the Landing Outcome only being “Success (ground pad)”.

Successful Drone Ship Landing with Payload between 4000 and 6000

- SQL query:

```
%%sql

SELECT Booster_Version
FROM SPACEXTABLE
WHERE (Landing_Outcome = "Success (drone ship)") AND (PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000);
```

output

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

- Explanation:

The WHERE command filters the output based on the Landing Outcome only being “Success (drone ship)” and the Payload Mass being between 4000 and 6000 kg, by using the BETWEEN command.

Total Number of Successful and Failure Mission Outcomes

- SQL query:

```
%%sql

SELECT (SELECT COUNT(Mission_Outcome) FROM SPACEXTBL WHERE Mission_Outcome LIKE '%Success%') AS Success,
(SELECT COUNT(Mission_Outcome) FROM SPACEXTBL WHERE Mission_Outcome LIKE '%Failure%') AS Failure
```

output

Success	Failure
100	1

- Explanation:

We use two subqueries combined with the COUNT function to collect the total number of successful and failure mission outcomes.

Boosters Carried Maximum Payload

- SQL query:

```
%%sql

SELECT Booster_Version
FROM SPACEXTABLE
WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_)
                           FROM SPACEXTABLE);
```

output

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 use a subquery combined with the WHERE command and the MAX function to collect the Booster Versions with Payload Mass equal to the maximum payload mass.

2015 Launch Records

- SQL query:

```
%%sql

SELECT SUBSTR(Date, 6, 2) AS Month, Landing_Outcome, Booster_Version, Launch_Site
FROM SPACEXTABLE
WHERE SUBSTR(Date, 0, 5) = "2015" AND Landing_Outcome = "Failure (drone ship)";
```

output

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

- Explanation:

We use the SUBSTR function to pick the Year and the Month, combined with the WHERE command, which filters the outputs based on the Year being 2015 and the Landing Outcome "Failure (drone ship)".

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

- SQL query:

```
%%sql

SELECT Landing_Outcome, COUNT(Landing_Outcome) AS Count_of_Landing_Outcomes
FROM (SELECT * FROM SPACEXTABLE WHERE Landing_Outcome = "Failure (drone ship)" OR Landing_Outcome = "Success (ground pad)")
GROUP BY Landing_Outcome
HAVING "DATE" BETWEEN "2010-06-04" AND "2017-03-20"
ORDER BY Count_of_Landing_Outcomes DESC;
```

output

Landing_Outcome	Count_of_Landing_Outcomes
Success (ground pad)	9
Failure (drone ship)	5

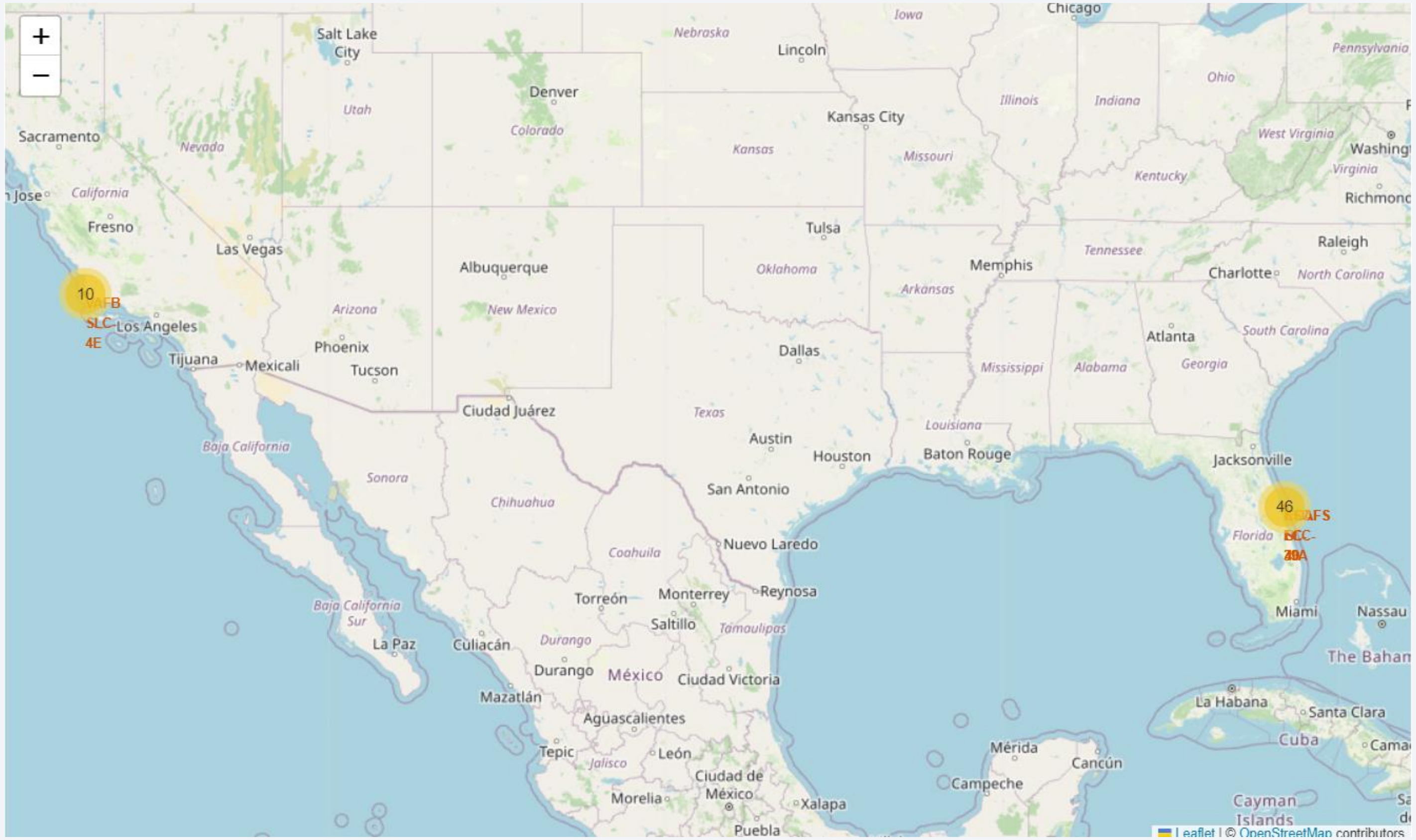
- Explanation:

We first use the COUNT function to collect the number of Landing Outcomes. Then, we use a subquery to filter the table based on the Landing Outcome being “Failure (drone ship)” or “Success (ground pad)”. Then, we group the results based on the Landing Outcome and we use the HAVING command to filter the results between “2010-06-04” and “2017-03-20”. Finally, we use the ORDER BY command and DESC to order the results by the number of Landing Outcomes in a descending order.

Section 3

Launch Sites Proximities Analysis

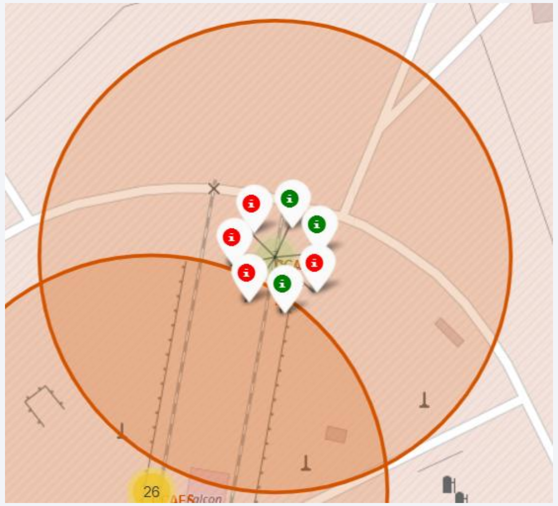
Main locations of Launch Sites



- All Launch Sites are located near coastlines of the United States.

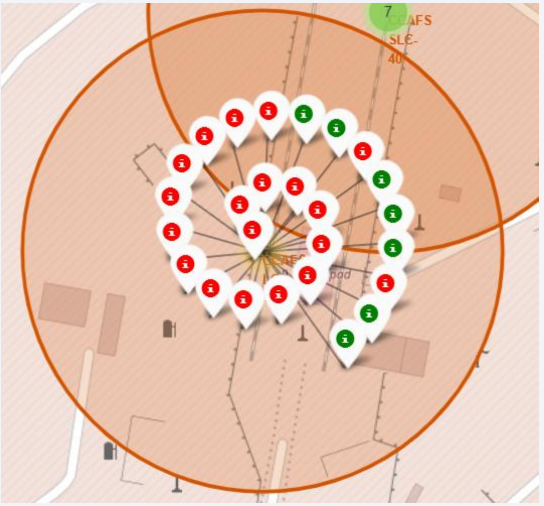
Launch Site locations with Color Labeled Markers

CCAFS SLC-40

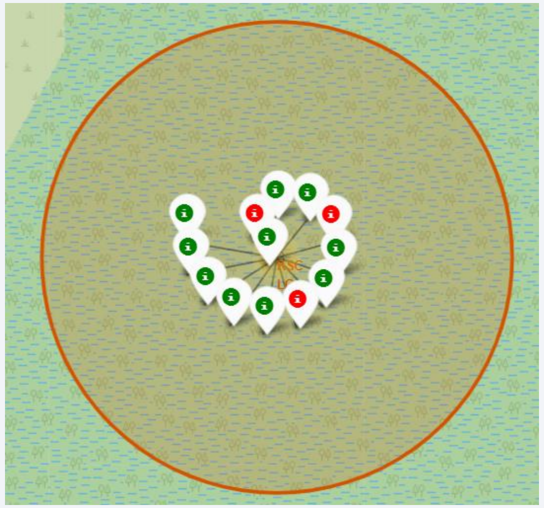


VAFB SLC-4E

CCAFS LC-40

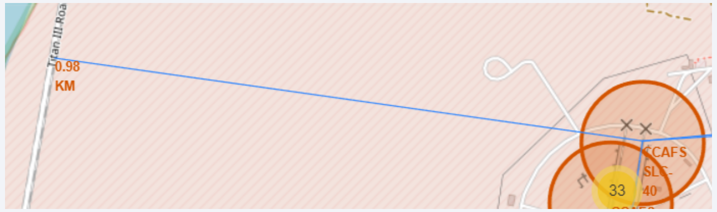


KSC LC-39A



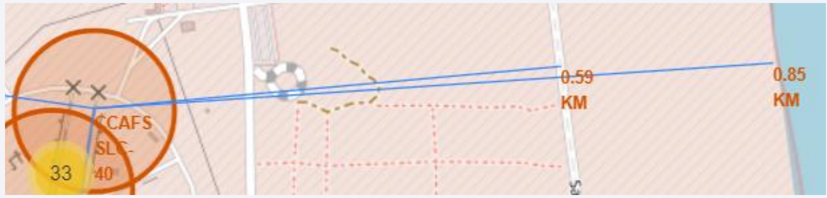
- Given that red markers represent failure and green markers represent successful landings, we can see that **KSC LC-39A** has a much higher success rate compared to the other launch sites.

Distances between CCAFS SLC-40 and its proximities



Distance from nearest railway:
~0.98 km

Distance from nearest city:
~18.18 km



Distance from nearest highway:
~0.59 km
Distance from nearest coastline:
~0.85 km

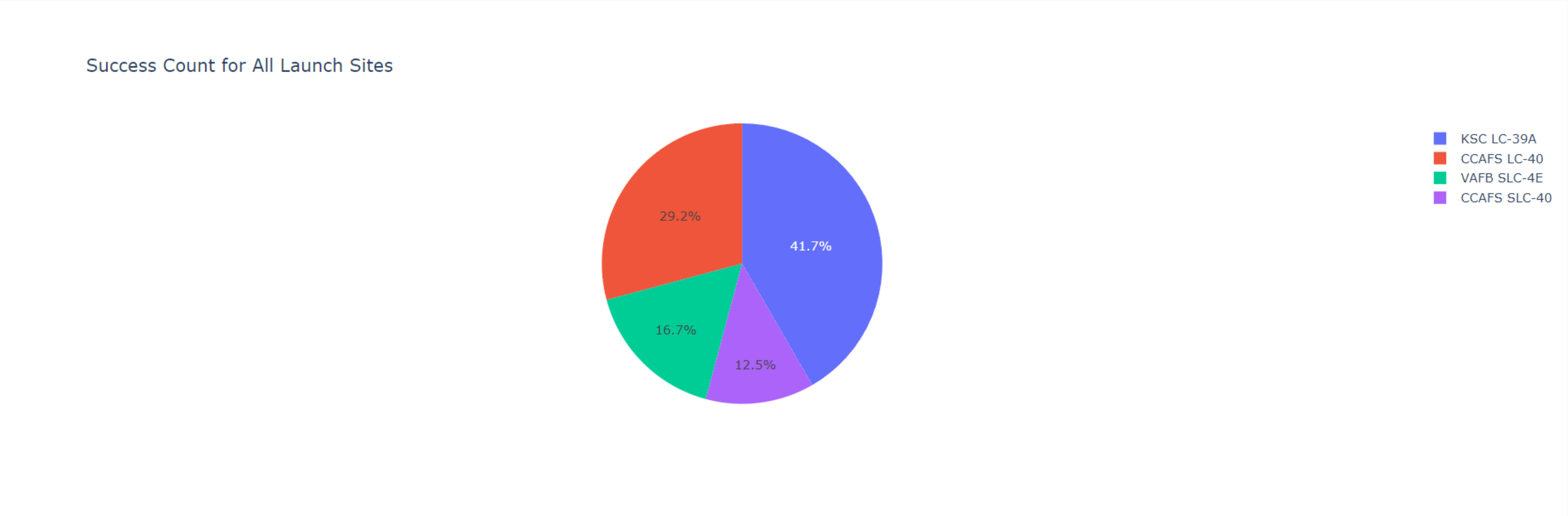
As far as ***CCAFS SLC-40*** is concerned:

- Are launch sites in close proximity to railways? **Yes**
- Are launch sites in close proximity to highways? **Yes**
- Are launch sites in close proximity to coastline? **Yes**
- Do launch sites keep certain distance away from cities? **Yes**

Section 4

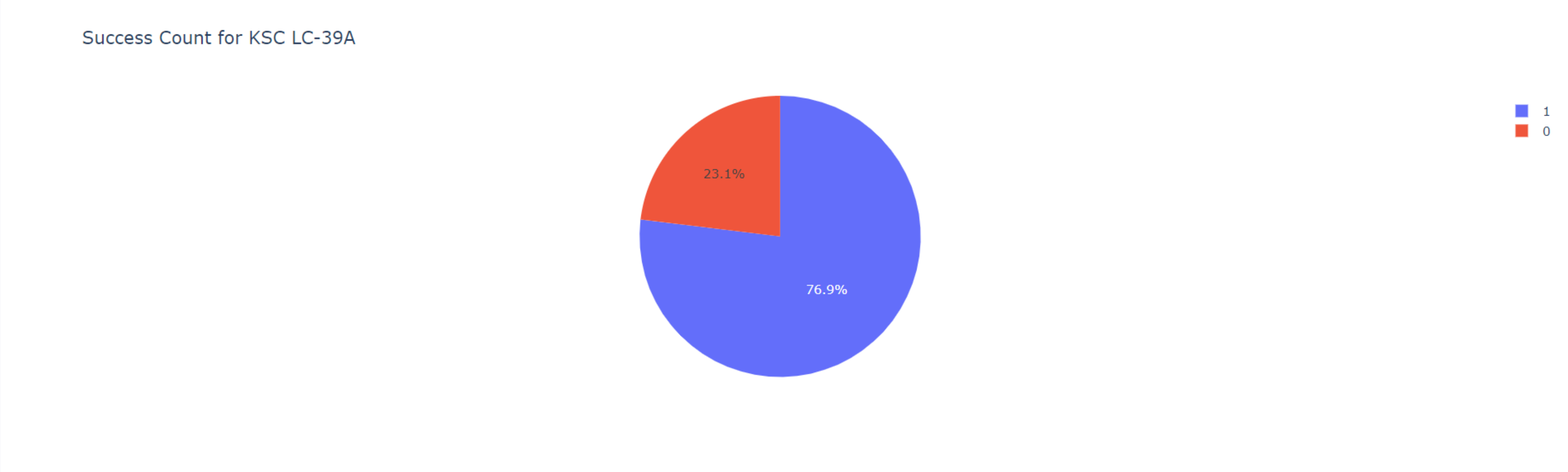
Build a Dashboard with Plotly Dash

Total success rate for all launch sites via Plotly Dash



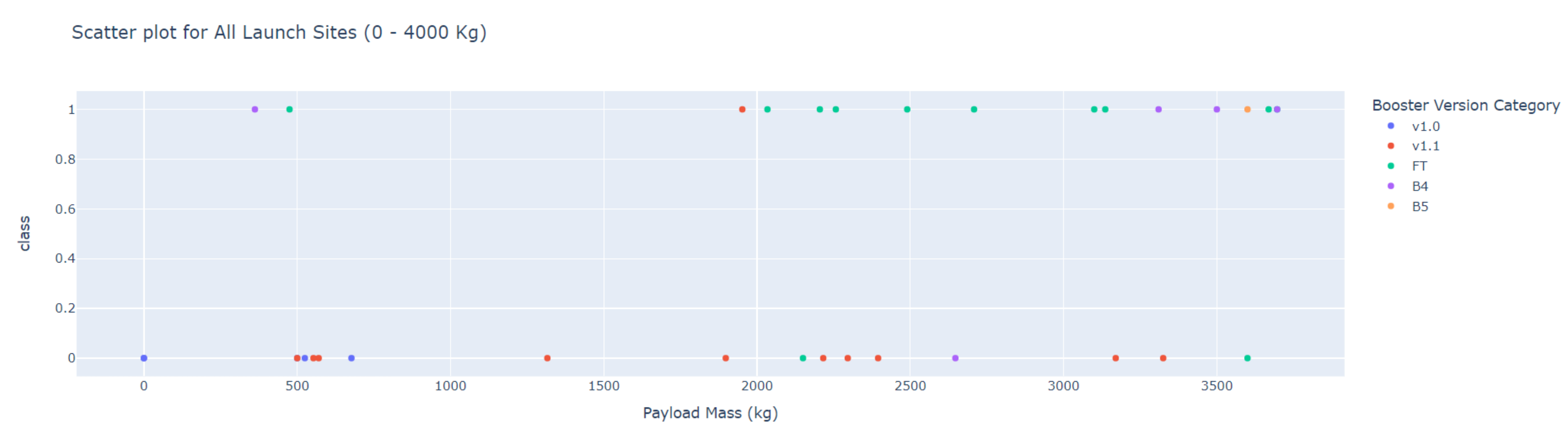
In agreement with the results given by the colored markers on the Folium map, the pie chart shows that KSC LC-39A has the largest number of successful rocket launches.

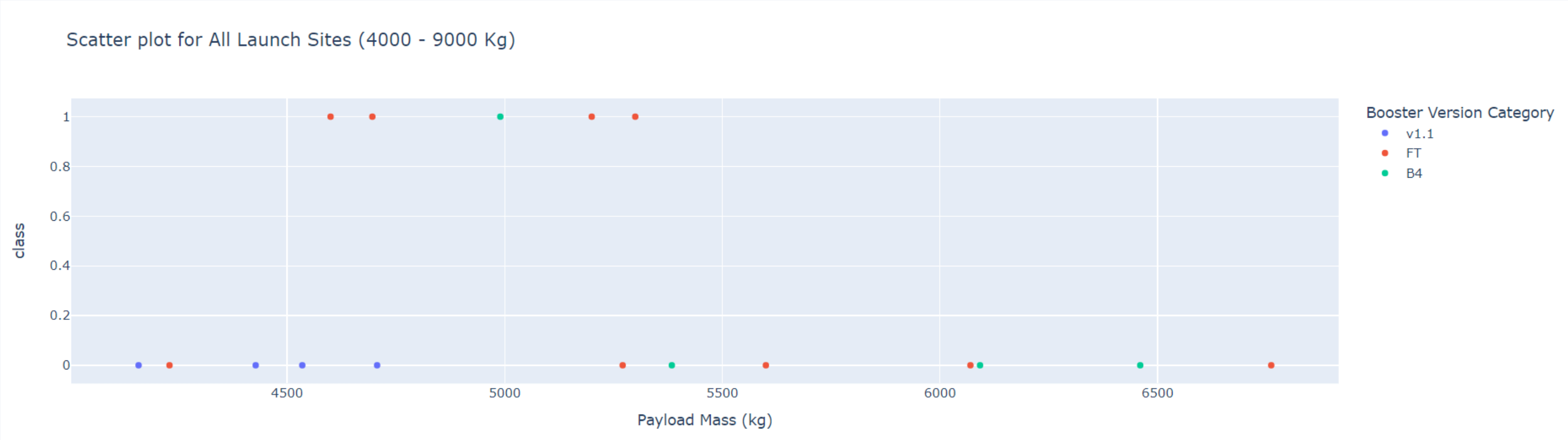
Success/Failure rate for KSC LC-39A via Plotly Dash



Again, in agreement with the results given by the colored markers on the Folium map, the pie chart shows that KSC LC-39A has the largest success rate among all launch sites.

Launch outcomes for different payloads via Plotly Dash



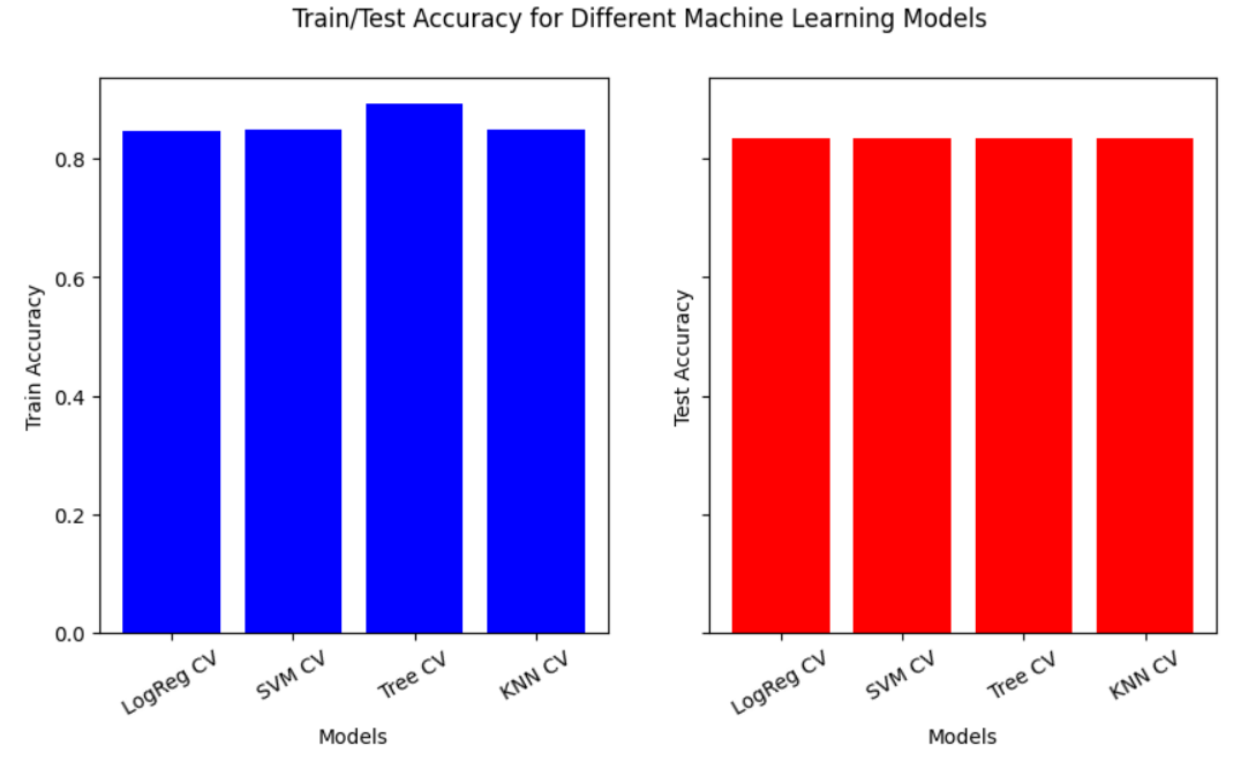


Overall, we see that lower payloads (<4000 kg) tend to have a larger success rate compared to higher payloads.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

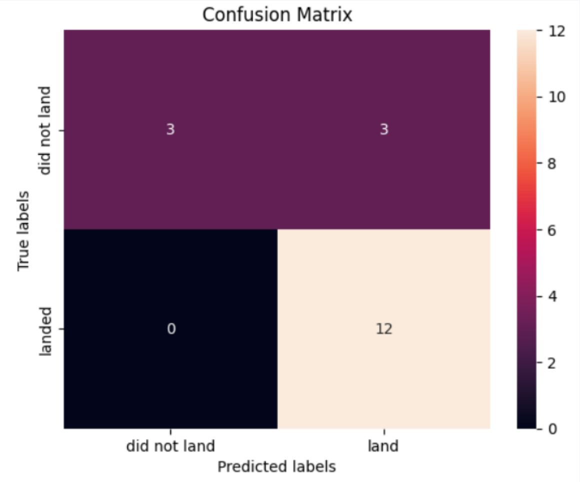


	Train Accuracy	Test Accuracy
Tree CV	0.891071	0.833333
K Nearest Neighbors CV	0.848214	0.833333
Support Vector Machine CV	0.848214	0.833333
Logistic Regression CV	0.846429	0.833333

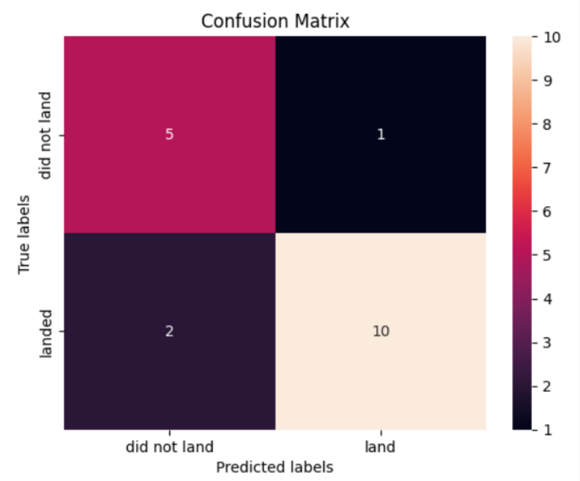
As far as Test Accuracy is concerned, all models performed almost the same. However, Tree CV gives the best Train Accuracy among all models. Therefore, we would probably choose this model for future predictions on new data.

Confusion Matrix

Logistic Regression



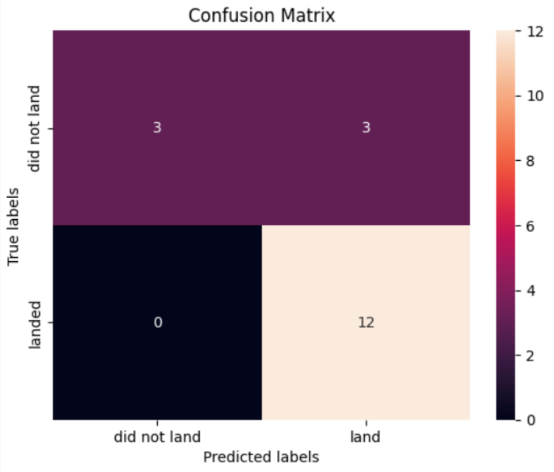
Tree



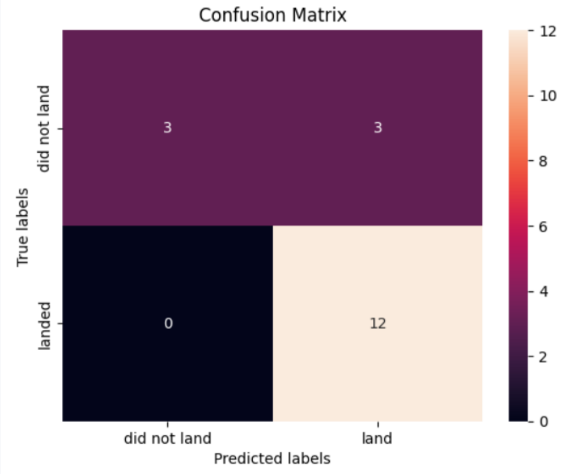
All confusion matrices are identical except for the confusion matrix of the Tree model. Again, the sum of **True Positives** and **True Negatives** is the same with the other matrices, but there is a rearrangement in the **False Positives** and **False Negatives**, which are now more equally distributed. All other models had a larger percentage of False Positives.

It turns out that the Tree model has higher **Precision** while the other models have higher **Recall**.

Support Vector Machine



K Nearest Neighbors



Conclusions

- While metrics such as the Payload Mass, the Orbit type, and Flight Number form patterns that can indicate if a landing would be successful or not, this is not always the case.
- Rockets with lower payload are more likely to land successfully.
- Launch sites are typically in close proximity to railways, highways, and coastlines but not to cities.
- Although all chosen Machine Learning models performed well, no one produced an accuracy score higher than 90%.
- Since rocket launches are a really expensive task, we must be able to predict with much higher accuracy if a landing would be successful or not. We will need a larger dataset to train the models in order to obtain such a result.

Thank you!