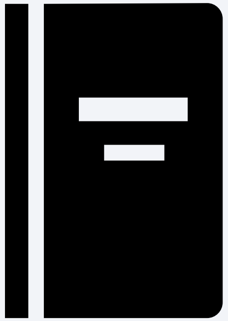
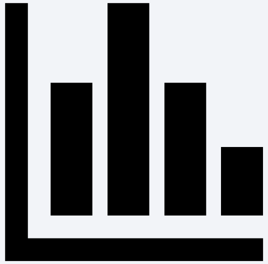
IBM Developer
SKILLS NETWORK

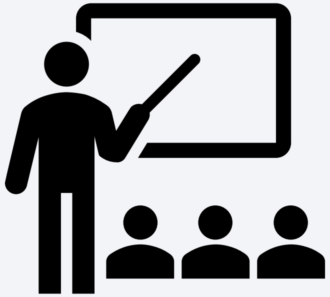
Winning Space Race with Data Science

Rounak Sen
11/06/2022

Outline







1. Executive Summary
2. Introduction
3. Methodology
4. Results
5. Conclusion
6. Appendix

Executive Summary

Summary of methodologies -

- Data Collection via API, Web Scraping
- Exploratory Data Analysis (EDA) with Data Visualization
- EDA with SQL
- Interactive Map with Folium
- Dashboards with Plotly Dash
- Predictive Analysis

Summary of all results -

- Exploratory Data Analysis results
- Interactive maps and dashboard
- Predictive results

Introduction

Project background and context

The aim of this project is to predict if the Falcon 9 first stage will successfully land. SpaceX says on its website that the Falcon 9 rocket launch cost 62 million dollars. Other providers cost upward of 165 million dollars each. The price difference is explained by the fact that SpaceX can reuse the first stage. By determining if the stage will land, we can determine the cost of a launch. This information is interesting for another company if it wants to compete with SpaceX for a rocket launch.

Problems you want to find answers

What are the main characteristics of a successful or failed landing ?

What are the effects of each relationship of the rocket variables on the success or failure of a landing ?

What are the conditions which will allow SpaceX to achieve the best landing success rate ?

Section 1

Methodology

Methodology

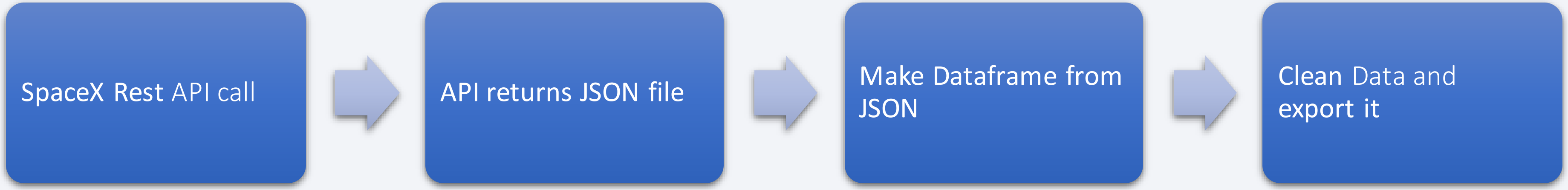
Executive Summary

- Data collection methodology:
  - Combined data from SpaceX public API and SpaceX Wikipedia page
- Perform data wrangling
  - Dropping unnecessary columns and one Hot Encoding for classification models
  - Classifying true landings as successful and unsuccessful otherwise
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Tuned models using GridSearchCV

Data Collection

Datasets are collected from Rest SpaceX API and webscrapping Wikipedia

- The information obtained by the API are rocket, launches, payload information.
- The Space X REST API URL is api.spacexdata.com/v4/



- The information obtained by the web scrapping of Wikipedia are launches, landing, payload information.
- URL is https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922

Data Collection – SpaceX API

1. Getting Response from API

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

2. Convert Response to JSON File

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

3. Transform data

```
getBoosterVersion(data)
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

4. Create dictionary with data

```
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}
```

5. Create dataframe

```
data_falcon9 = pd.DataFrame.from_dict(launch_dict)
```

6. Filter dataframe

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

7. Export to file

```
data_falcon9.to_csv('dataset_part\_1.csv', index=False)
```

[Link to Code]

Data Collection - Scraping

1. Getting Response from HTML

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

2. Create BeautifulSoup Object

```
soup = BeautifulSoup(response.text, 'html5lib')
```

3. Find all tables

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

4. Get 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)
```

5. Create 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']=[]
```

6. Add data to keys

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

7. Create dataframe from dictionary

```
df=pd.DataFrame(launch_dict)
```

8. Export to file

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

[Link to Code]

Data Wrangling

- In the dataset, there are several cases where the booster did not land successfully.
  - True Ocean, True RTLS, True ASDS means the mission has been successful.
  - False Ocean, False RTLS, False ASDS means the mission was a failure.
- We need to transform string variables into categorical variables where 1 means the mission has been successful and 0 means the mission was a failure.

1. Calculate launches number for each site

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

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

2. Calculate the number and occurrence of each orbit

```
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
```

3. Calculate number and occurrence of mission outcome per orbit type

```
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
```

4. Create landing outcome label from Outcome column

```
landing_class = []

for outcome in df['Outcome']:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
df['Class']=landing_class
```

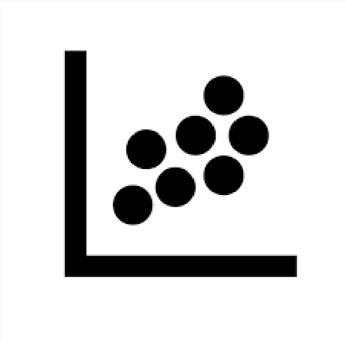
5. Export to file for each site

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

Link to Code

EDA with Data Visualization

- Scatter Graphs



- Flight Number vs. Payload Mass
- Flight Number vs. Launch Site
- Payload vs. Launch Site
- Orbit vs. Flight Number
- Payload vs. Orbit Type
- Orbit vs. Payload Mass

Scatter plots show relationship between variables. This relationship is called the correlation.

- Bar Graph
  - Success rate vs. Orbit

Bar graphs show the relationship between numeric and categoric variables

- Line Graph
  - Success rate vs. Year

Line graphs show data variables and their trends. Line graphs can help to show global behavior and make prediction for unseen data.

[Link to Code]

EDA with SQL

- We performed SQL queries to gather and understand data from dataset:
- Displaying 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_failure landing_outcomes in drone ship, booster versions, launch_site for the months in year 2015.
- Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

[Link to Code]()

Build an Interactive Map with Folium

- Folium map object is a map centered on NASA Johnson Space Center at Houson, Texas
  - Red circle at NASA Johnson Space Center's coordinate with label showing its name (folium.Circle, folium.map.Marker).
  - Red circles at each launch site coordinates with label showing launch site name (folium.Circle, folium.map.Marker, folium.features.DivIcon).
  - The grouping of points in a cluster to display multiple and different information for the same coordinates(folium.plugins.MarkerCluster).
  - Markers to show successful and unsuccessful landings. Green for successful landing and Red for unsuccessful landing.(folium.map.Marker, folium.Icon).
  - Markers to show distance between launch site to key locations (railway, highway, coastway, city) and plot a line between them.(folium.map.Marker, folium.PolyLine, folium.features.DivIcon)
- These objects are created in order to understand better the problem and the data. We can show easily all launch sites, their surroundings and the number of successful and unsuccessful landings.

Link to Code

Build a Dashboard with Plotly Dash

- Dashboard has dropdown, pie chart, range slider and scatter plot components

- Dropdown allows a user to choose the launch site or all launch sites (dash_core_components.Dropdown).
- Pie chart shows the total success and the total failure for the launch site chosen with the dropdown component (plotly.express.pie).
- Rangeslider allows a user to select a payload mass in a fixed range (dash_core_components.RangeSlider).
- Scatter chart shows the relationship between two variables, in particular Success vs Payload Mass (plotly.express.scatter).

Link to Code

Predictive Analysis (Classification)

- Data preparation
  - Load dataset
  - Normalize data
  - Split data into training and test sets.
- Model preparation
  - Selection of machine learning algorithms
  - Set parameters for each algorithm to GridSearchCV
  - Training GridSearchModel models with training dataset
- Model evaluation
  - Get best hyperparameters for each type of model
  - Compute accuracy for each model with test dataset
  - Plot Confusion Matrix
- Model comparison
  - Comparison of models according to their accuracy
  - The model with the best accuracy will be chosen (see Notebook for result)

[Link to Code]

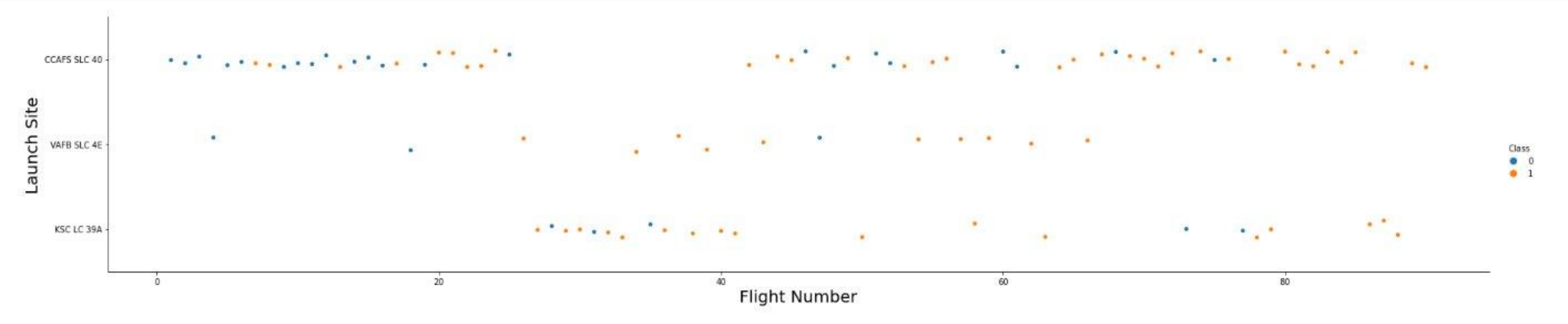
Results

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

Section 2

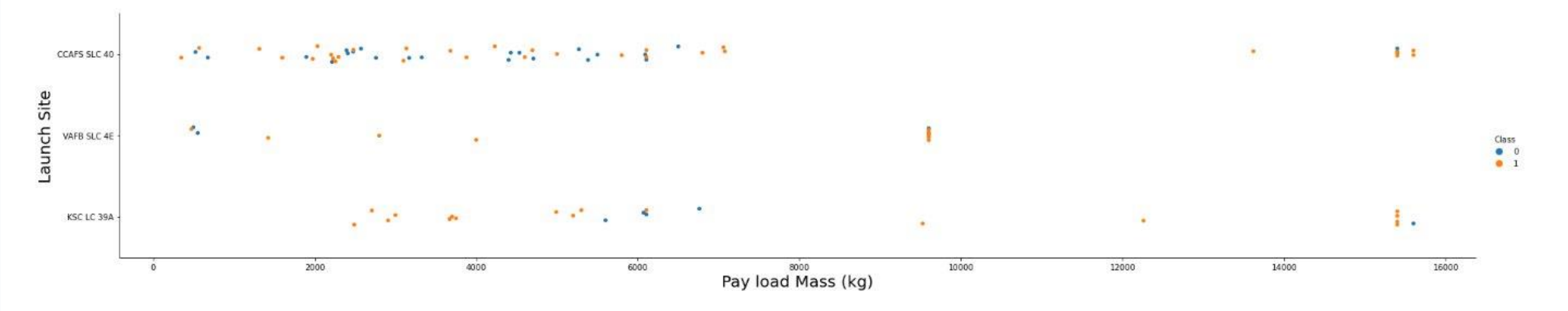
Insights drawn from EDA

Flight Number vs. Launch Site



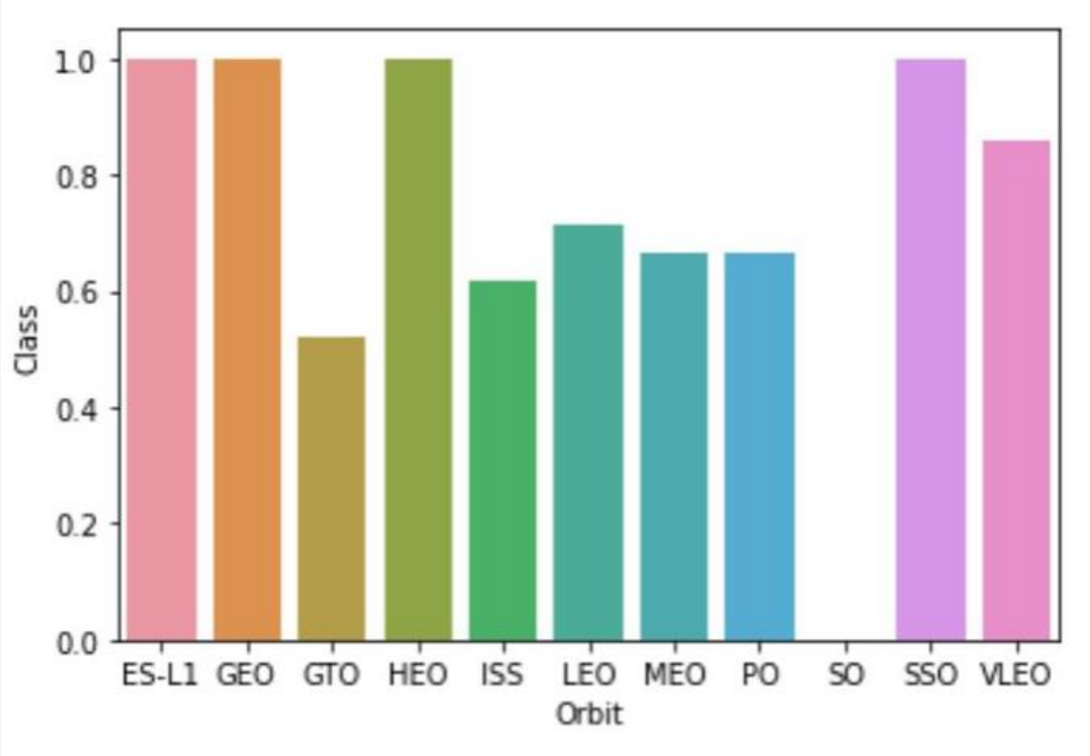
We observe that, for each site, the success rate is increasing.

Payload vs. Launch Site



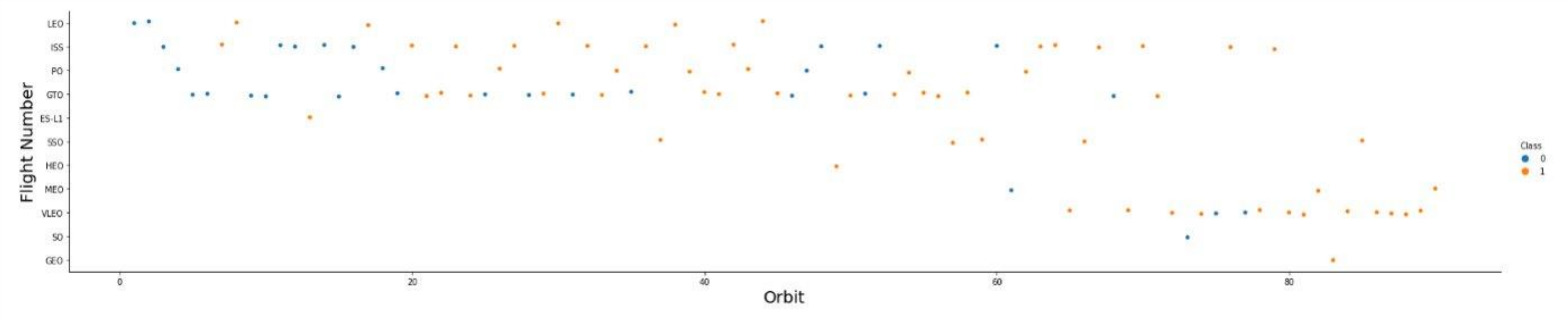
Depending on the launch site, a heavier payload may be a consideration for a successful landing. On the other hand, a too heavy payload can make a landing fail.

Success Rate vs. Orbit Type



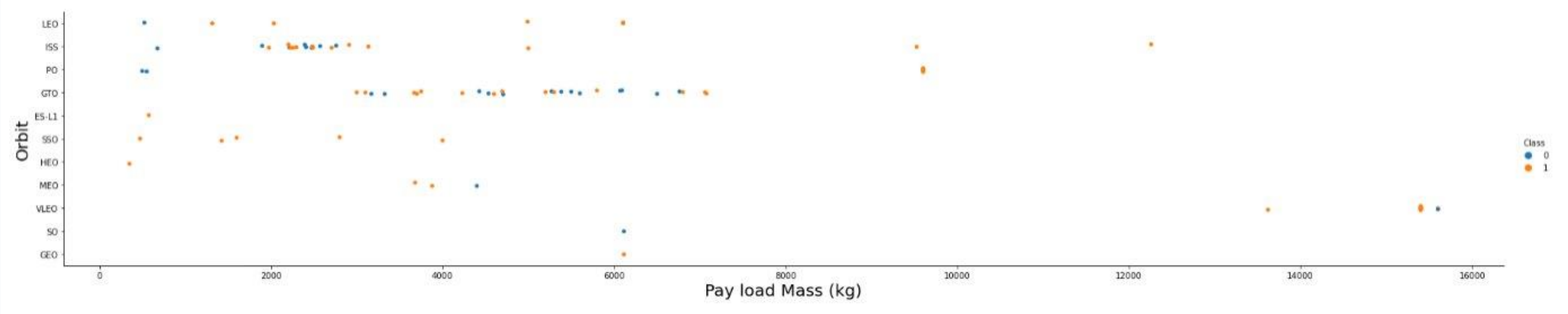
With this plot, we can see success rate for different orbit types. We note that ES-L1, GEO, HEO, SSO have the best success rate.

Flight Number vs. Orbit Type



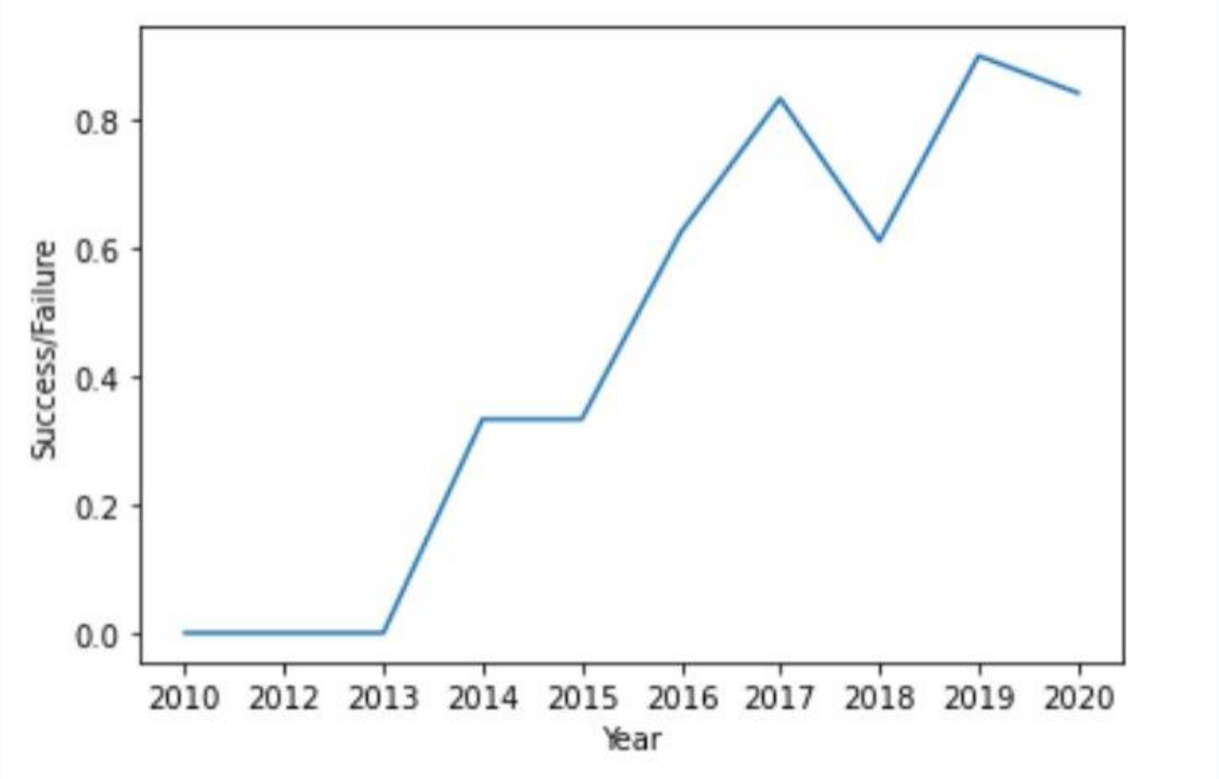
We notice that the success rate increases with the number of flights for the LEO orbit. For some orbits like GTO, there is no relation between the success rate and the number of flights. But we can suppose that the high success rate of some orbits like SSO or HEO is due to the knowledge learned during former launches for other orbits.

Payload vs. Orbit Type



The weight of the payloads can have a great influence on the success rate of the launches in certain orbits. For example, heavier payloads improve the success rate for the LEO orbit. Another finding is that decreasing the payload weight for a GTO orbit improves the success of a launch.

Launch Success Yearly Trend



Since 2013, we can see an increase in the Space X Rocket success rate.

All Launch Site Names

SQL Query

```
SELECT UNIQUE(LAUNCH_SITE) FROM SPACEXDATASET;
```

Explanation

The use of DISTINCT in the query allows to remove duplicate LAUNCH_SITE.

Results

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

Launch Site Names Begin with 'CCA'

SQL Query

```
SELECT * FROM SPACEXDATASET WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

Results

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	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Explanation

The WHERE clause followed by LIKE clause filters launch sites that contain the substring CCA. LIMIT 5 shows 5 records from filtering.

Total Payload Mass

SQL Query

```
SELECT SUM(PAYLOAD_MASS_KG_) AS TOTAL_MASS_PAYLOAD FROM SPACEXDATASET WHERE CUSTOMER = 'NASA (CRS)';
```

Results

total_mass_payload
45596

Explanation

This query returns the sum of all payload masses where the customer is NASA (CRS).

Average Payload Mass by F9 v1.1

SQL Query

```
SELECT AVG(PAYLOAD_MASS_KG_) AS AVERAGE_MASS_PAYLOAD FROM SPACEXDATASET WHERE BOOSTER_VERSION LIKE 'F9 v1.1';
```

Results

average_mass_payload
2928

Explanation

This query returns the average of all payload masses where the booster version contains the substring F9 v1.1.

First Successful Ground Landing Date

SQL Query

```
SELECT MIN(DATE) AS MIN_DATE FROM SPACEXDATASET WHERE LANDING_OUTCOME = 'Success (ground pad)';
```

Results

min_date
2015-12-22

Explanation

With this query, we select the oldest successful landing.
The WHERE clause filters dataset in order to keep only records where landing was successful.
With the MIN function, we select the record with the oldest date.

Successful Drone Ship Landing with Payload between 4000 and 6000

SQL Query

```
SELECT BOOSTER_VERSION FROM SPACEXDATASET WHERE LANDING_OUTCOME = 'Success (drone ship)'
AND PAYLOAD_MASS_KG_ > 4000 AND PAYLOAD_MASS_KG_ < 6000;
```

Results

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

Explanation

This query returns the booster version where landing was successful and payload mass is between 4000 and 6000 kg. The WHERE and AND clauses filter the dataset.

Total Number of Successful and Failure Mission Outcomes

SQL Query

```
SELECT COUNT(MISSION_OUTCOME) AS MISSION_OUTCOME_NUMBER, MISSION_OUTCOME FROM SPACEXDATASET GROUP BY MISSION_OUTCOME;
```

Results

mission_outcome_number	mission_outcome
1	Failure (in flight)
99	Success
1	Success (payload status unclear)

Explanation

With the first SELECT, we show the subqueries that return results. The first subquery counts the successful mission. The second subquery counts the unsuccessful mission. The WHERE clause followed by LIKE clause filters mission outcome. The COUNT function counts records filtered.

Boosters Carried Maximum Payload

SQL Query

```
SELECT BOOSTER_VERSION FROM SPACEXDATASET
WHERE PAYLOAD_MASS_KG_ = (SELECT MAX(PAYLOAD_MASS_KG_) FROM SPACEXDATASET);
```

Explanation

We used a subquery to filter data by returning only the heaviest payload mass with MAX function. The main query uses subquery results and returns unique booster version (SELECT DISTINCT) with the heaviest payload mass.

Results

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

2015 Launch Records

SQL Query

```
SELECT BOOSTER_VERSION, LAUNCH_SITE, DATE FROM SPACEXDATASET
WHERE LANDING_OUTCOME = 'Failure (drone ship)' AND YEAR(DATE) = 2015;
```

Results

booster_version	launch_site	DATE
F9 v1.1 B1012	CCAFS LC-40	2015-01-10
F9 v1.1 B1015	CCAFS LC-40	2015-04-14

Explanation

This query returns month, booster version, launch site where landing was unsuccessful and landing date took place in 2015. Substr function process date in order to take month or year. Substr(DATE, 4, 2) shows month. Substr(DATE,7, 4) shows year.

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

SQL Query

```
SELECT COUNT(LANDING_OUTCOME) AS RANK, LANDING_OUTCOME FROM SPACEXDATASET
WHERE DATE > '2010-06-04' AND DATE < '2017-03-20' GROUP BY LANDING_OUTCOME ORDER BY RANK DESC;
```

Explanation

This query returns landing outcomes and their count where mission was successful and date is between 04/06/2010 and 20/03/2017. The GROUP BY clause groups results by landing outcome and ORDER BY COUNT DESC shows results in decreasing order.

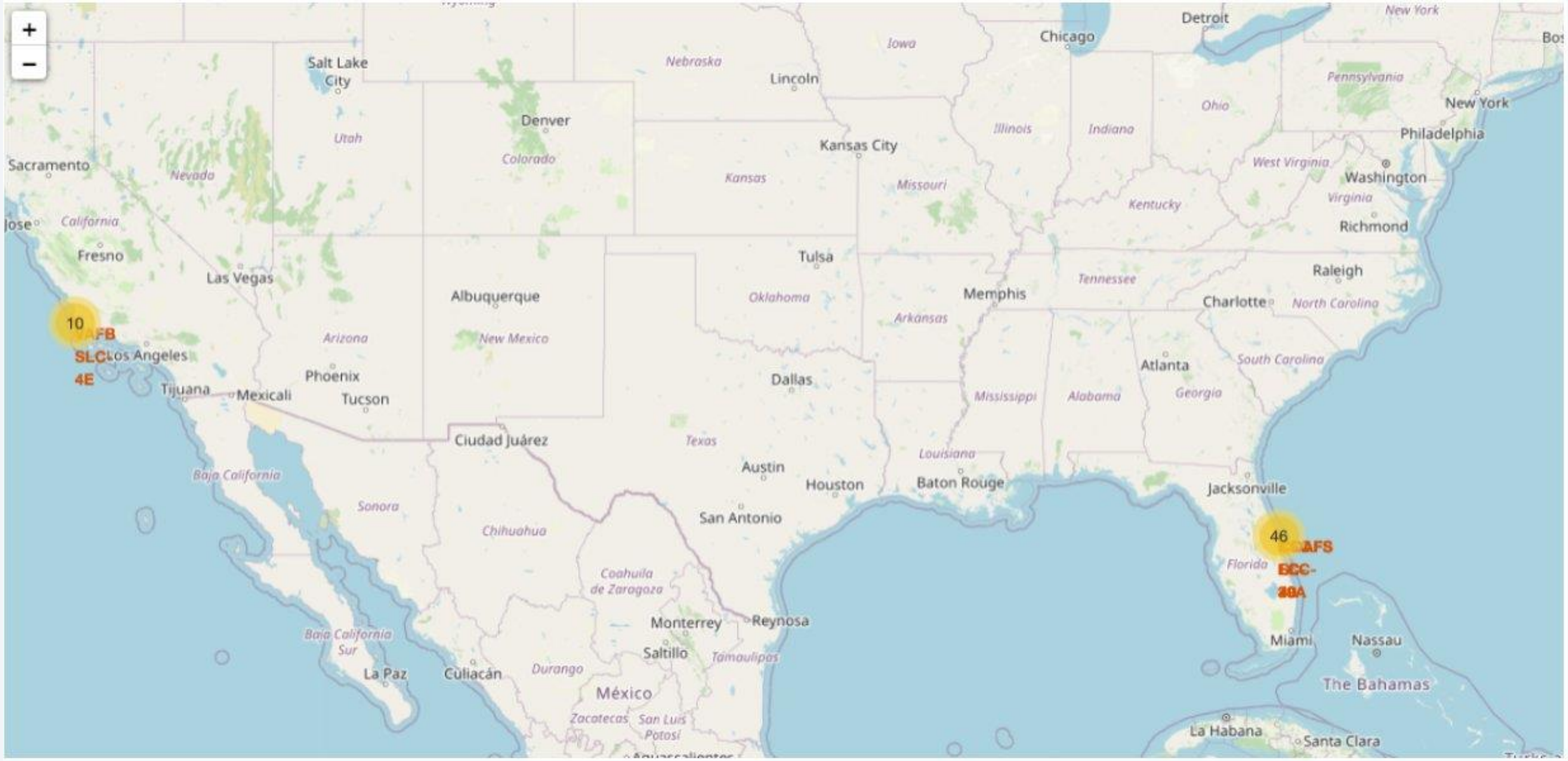
Results

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

Section 3

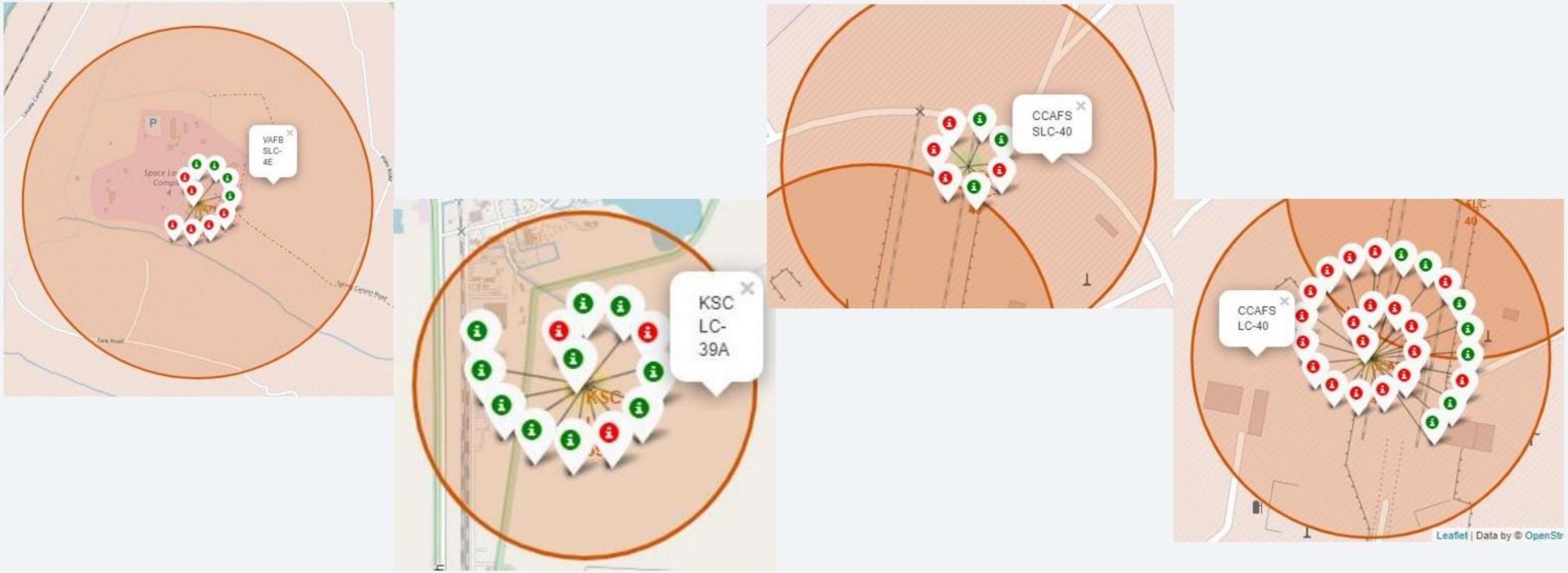
Launch Sites Proximities Analysis

Folium map – Ground stations



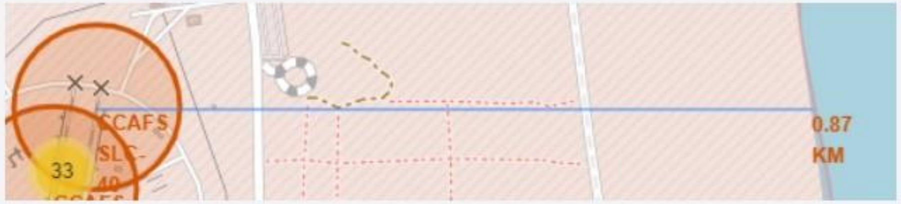
We see that Space X launch sites are located on the coast of the United States

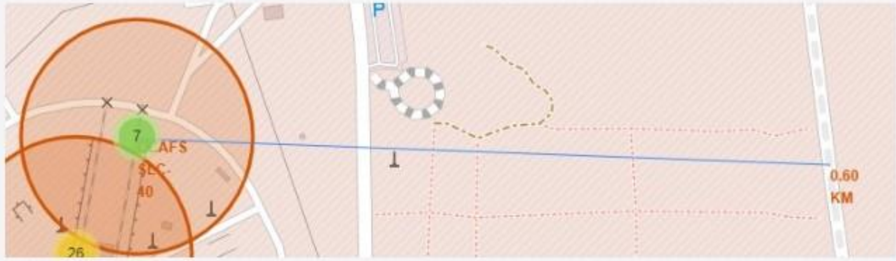
Folium map – Color Labeled Markers



Green marker represents successful launches. Red marker represents unsuccessful launches. We note that KSC LC-39A has a higher launch success rate.

Folium Map – Distances between CCAFS SLC-40 and its proximities



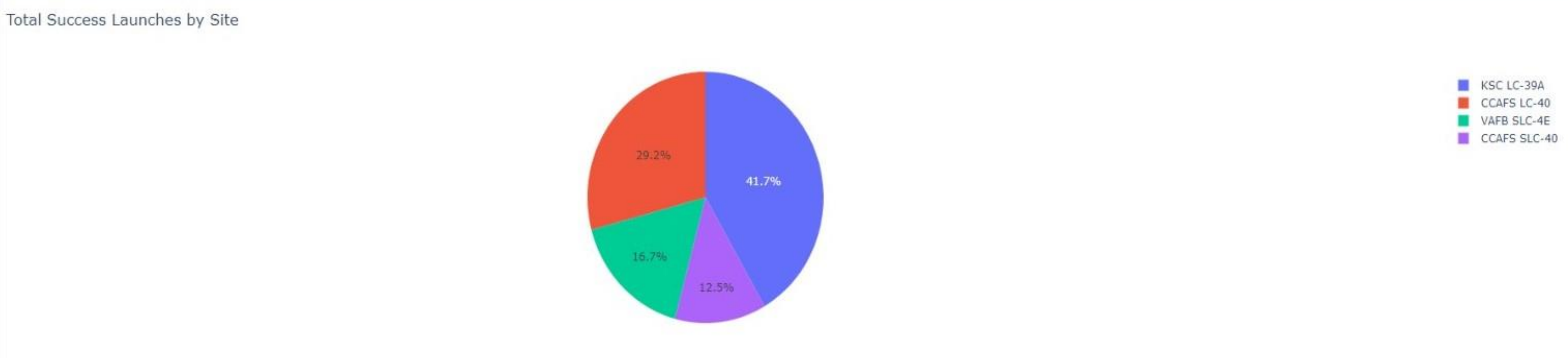


Is CCAFS SLC-40 in close proximity to railways ? Yes
Is CCAFS SLC-40 in close proximity to highways ? Yes
Is CCAFS SLC-40 in close proximity to coastline ? Yes
Do CCAFS SLC-40 keeps certain distance away from cities ? No

Section 4

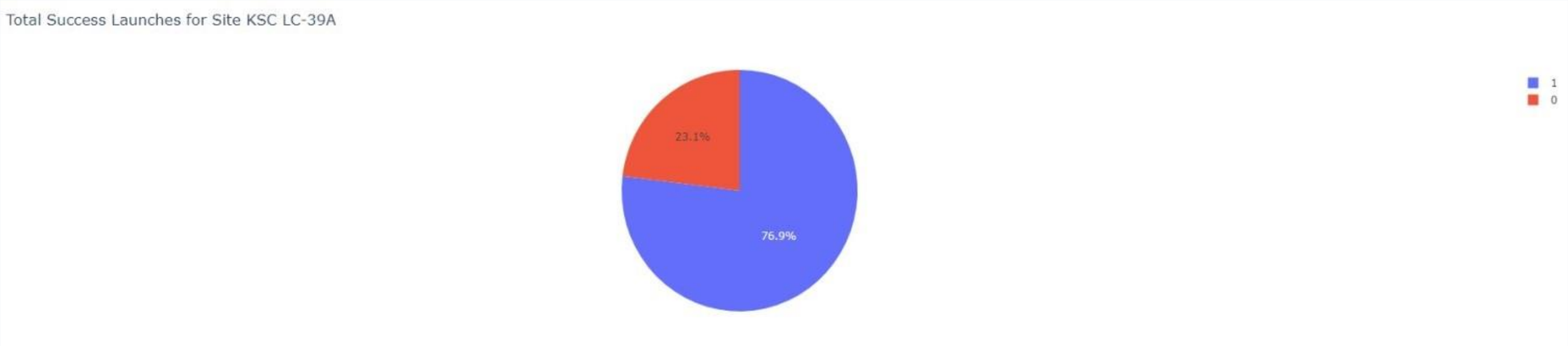
Build a Dashboard with Plotly Dash

Dashboard – Total success by Site



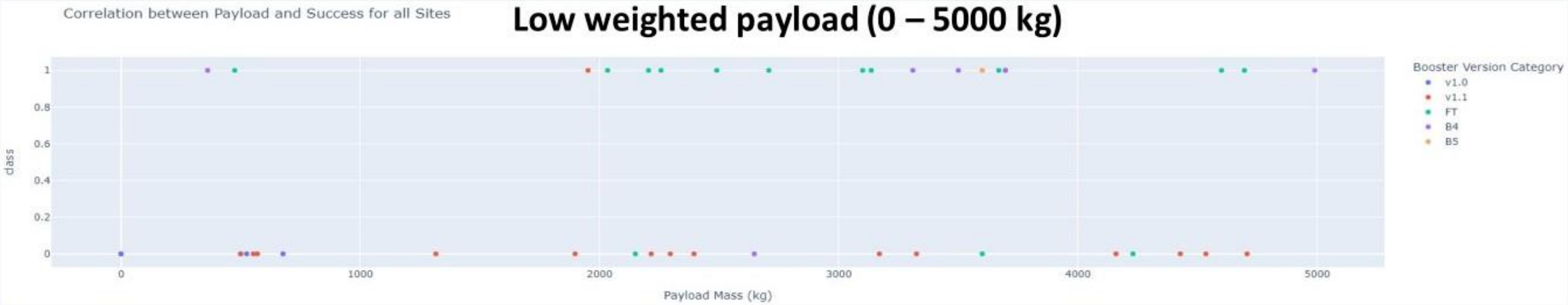
We see that KSC LC-39A has the best success rate of launches.

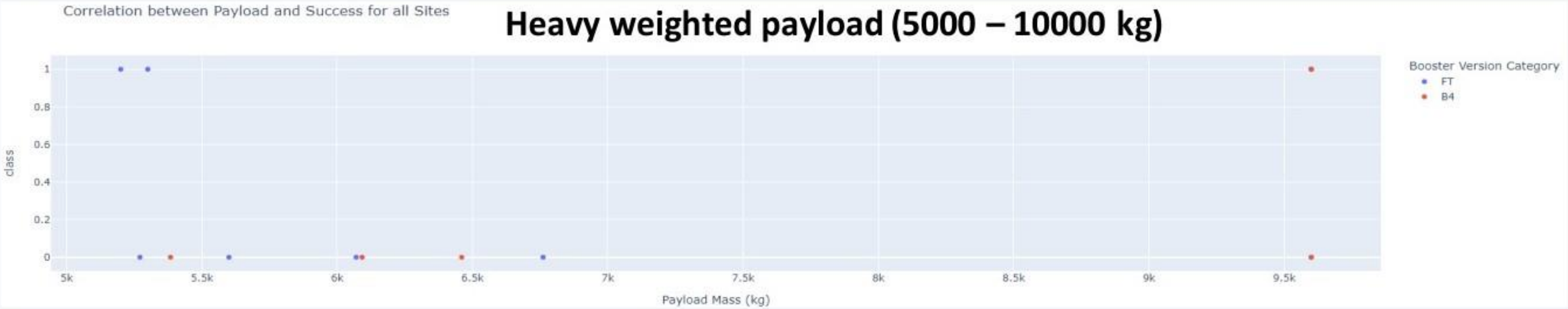
Dashboard – Total success launches for Site KSC LC-39A



We see that KSC LC-39A has achieved a 76.9% success rate while getting a 23.1% failure rate.

Dashboard – Payload mass vs Outcome for all sites with different payload mass selected





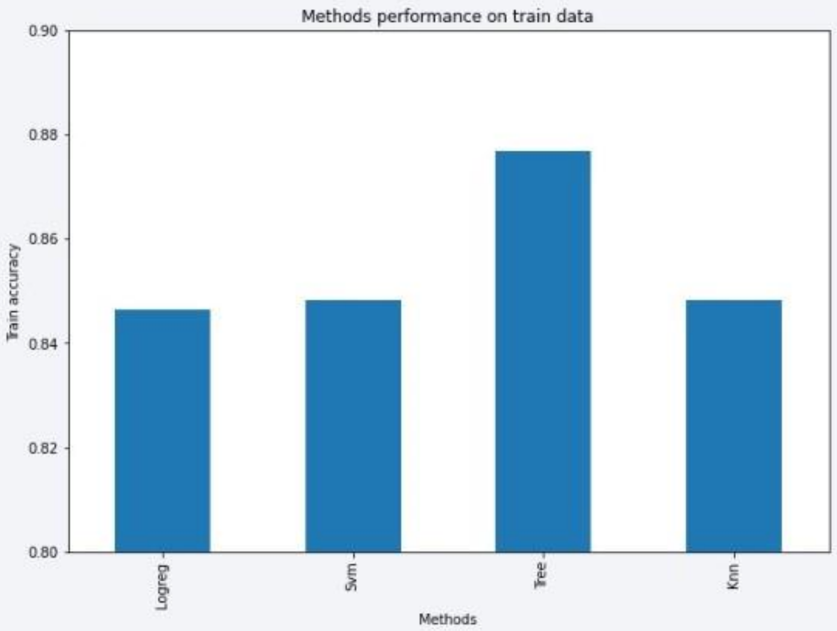
Low weighted payloads have a better success rate than the heavy weighted payloads.

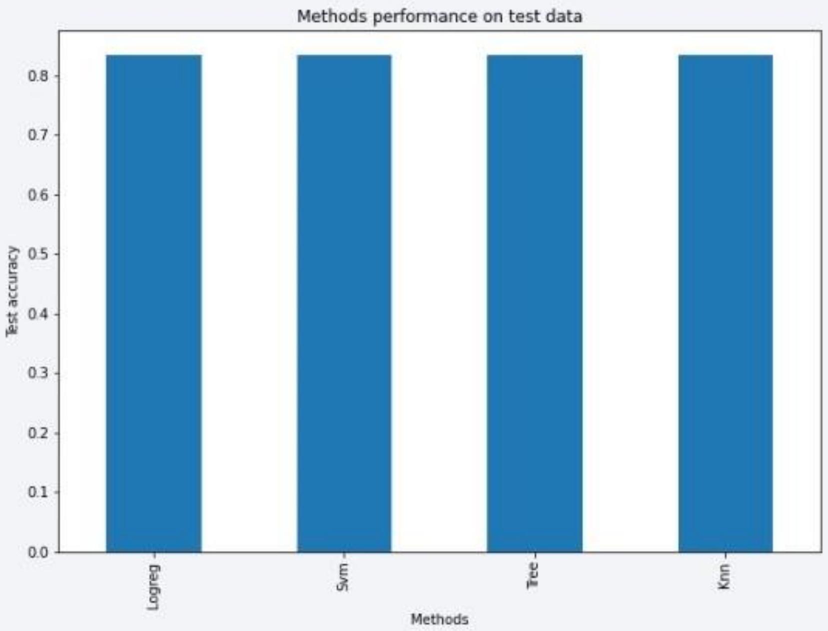
Section 5

Predictive Analysis (Classification)

Classification Accuracy

	Accuracy Train	Accuracy Test
Tree	0.876786	0.833333
Knn	0.848214	0.833333
Svm	0.848214	0.833333
Logreg	0.846429	0.833333





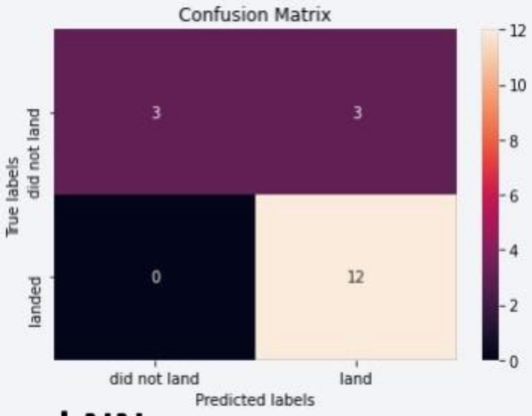
For accuracy test, all methods performed similar. **We could get more test data to decide between them. But** if we really need to choose one right now, we would take the decision tree.

Decision tree best parameters

```
tuned hyperparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 12, 'max_features': 'sqrt', 'min_samples_leaf':
4, 'min_samples_split': 2, 'splitter': 'random'}
```

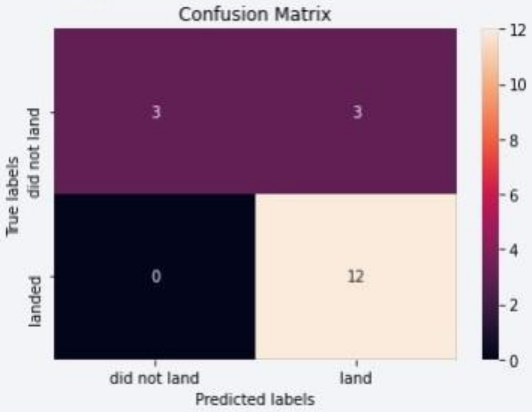
Confusion Matrix

Logistic regression

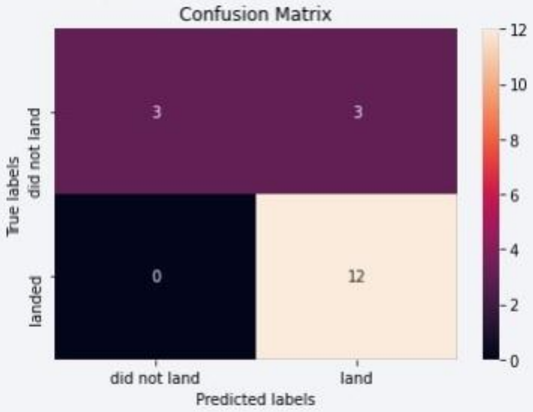


Decision Tree

kNN



SVM



As the test accuracy are all equal, the confusion matrices are also identical. The main problem of these models are false positives.

		Actual values	
		1	0
Predicted values	1	TP	FP
	0	FN	TN

Conclusions

- The success of a mission can be explained by several factors such as the launch site, the orbit and especially the number of previous launches. Indeed, we can assume that there has been a gain in knowledge between launches that allowed to go from a launch failure to a success.
- The orbits with the best success rates are GEO, HEO, SSO, ES-L1.
- Depending on the orbits, the payload mass can be a criterion to take into account for the success of a mission. Some orbits require a light or heavy payload mass. But generally low weighted payloads perform better than the heavy weighted payloads.
- With the current data, we cannot explain why some launch sites are better than others (KSC LC-39A is the best launch site). To get an answer to this problem, we could obtain atmospheric or other relevant data.
- For this dataset, we choose the Decision Tree Algorithm as the best model even if the test accuracy between all the models used is identical. We choose Decision Tree Algorithm because it has a better train accuracy.

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!