<Gabriel Romeo>

<06/09/2024>

Winning Space Race with Data Science

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 Visualizatio
  - 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

SpaceX is the most successful company of the commercial space age, making space travel affordable. The company advertises Falcon 9 rocket launches on its website, with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. Based on public information and machine learning models, we are going to predict if SpaceX will reuse the first stage.

Problems you want to find answers

- How do variables such as payload mass, launch site, number of flights, and orbits affect the success of the first stage landing?
- Does the rate of successful landings increase over the years?
- What is the best algorithm that can be used for binary classification in this case?

Section 1

Methodology

Methodology

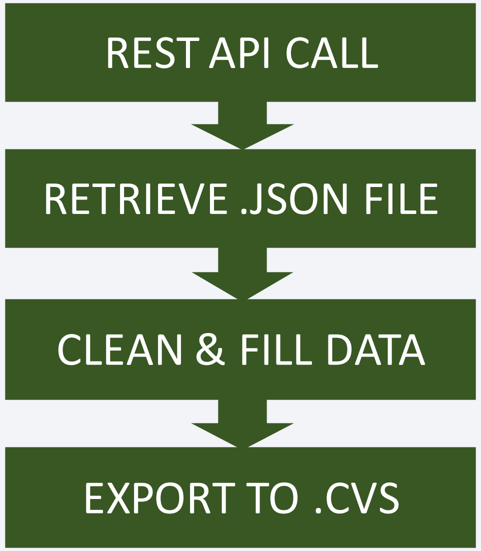
Executive Summary

- Data collection methodology:
  - Via SpaceX Rest API.
  - Via Web Scrapping from Wikipedia.
- Perform data wrangling
  - Exploratory data analysis using SQL & visualization.
  - Interactive visual analytics using Folium and Plotly Dash
  - Predictive analysis using classification models

Data Collection

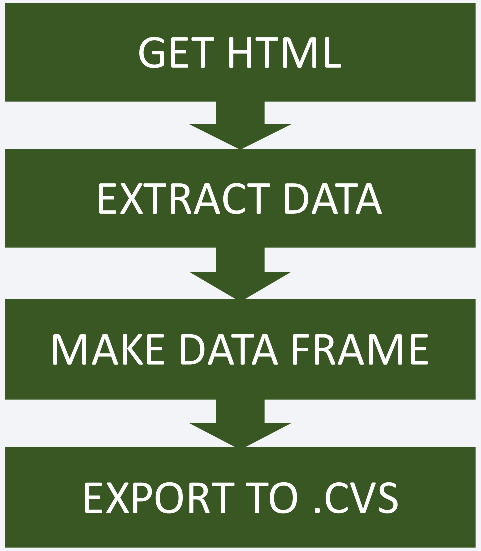
- Datasets are collected from SpaceX Rest API and web-scrapping Wikipedia

SpaceX Rest API



Information obtained: FlightNumber, Date, BoosterVersion, PayloadMass, Orbit, LaunchSite, Outcome, Flights, GridFins, Reused, Legs, LandingPad, Block, ReusedCount, Serial, Longitude, Latitude

Wikipedia



Information obtained: Flight No., Launch site, Payload, PayloadMass, Orbit, Customer, Launch outcome, Version Booster, Booster landing, Date, Time

Data Collection – SpaceX API

1. Getting response from the Rest API.
2. Converting response to a .json file.
3. Transforming the data.
4. Create a dictionary with de data.
5. Create a data frame with the dictionary.
6. Filter the DF to show only Falcon 9.
7. Export DF to a .CSV file.

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

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

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

```
# Call getBoosterVersion
getBoosterVersion(data)
```

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

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

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

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

```
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}
```

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = df.loc[df['BoosterVersion']!="Falcon 1"]
```

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

https://github.com/AtlasScientist/Jupyter-Notebook/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Data Collection - Scraping

1. Getting response from HTML.
2. Create a BeautifulSoup Object.
3. Find all tables.
4. Get column names.
5. Create dictionary.
6. Add data to keys.
7. Create Data Frame from the dictionary.
8. Export DF to .CSV file.

```python
# use requests.get() method with the provided static_url
# assign the response to a object
page = requests.get(static_url)
page.status_code
```

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

```python
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

```python
temp = soup.find_all('th')
for x in range(len(temp)):
    try:
     name = extract_column_from_header(temp[x])
     if (name is not None and len(name) > 0):
        column_names.append(name)
    except:
     pass
```

```python
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']=[]
```

```python
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
```

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

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

https://github.com/AtlasScientist/Jupyter-Notebook/blob/main/jupyter-labs-webscraping.ipynb

Data Wrangling

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. Caculate launches numbers for each launching site.
2. Calculate the number of each orbit type.
3. Calculate the number of missions outcome per orbit type.
4. Create the landing outcome label from the outcome column.
5. Export the DF to .CSV file.

```
# Apply value_counts() on column LaunchSite
df.LaunchSite.value_counts()
```

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

```
# Apply value_counts on Orbit column
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
```

```
# landing_outcomes = values on Outcome column
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
```

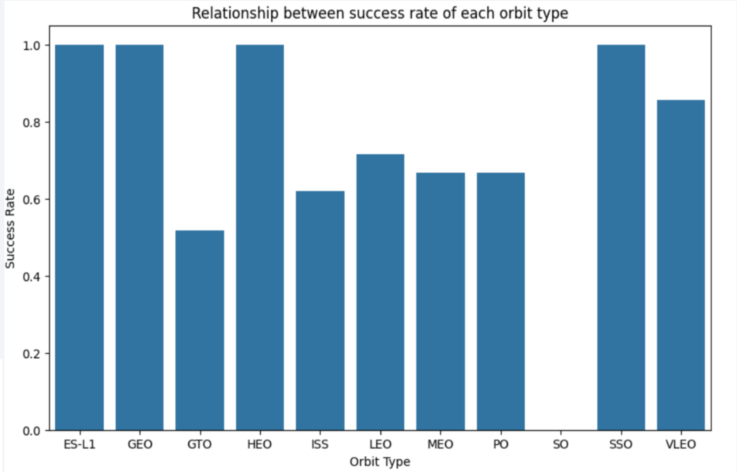
```
landing_class = []
for key,value in df["Outcome"].items():
    if value in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
landing_class
```

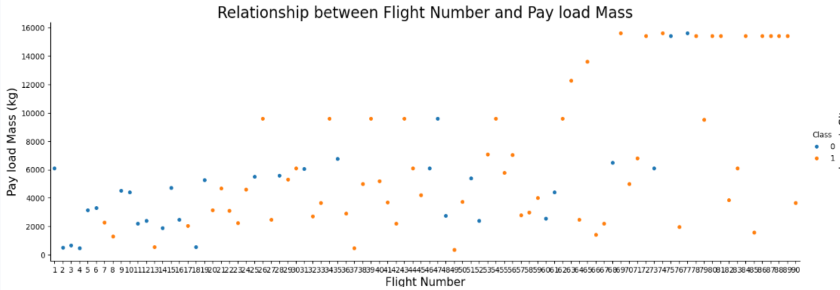
```
df.to_csv("dataset_part_2.csv", index=False)
```

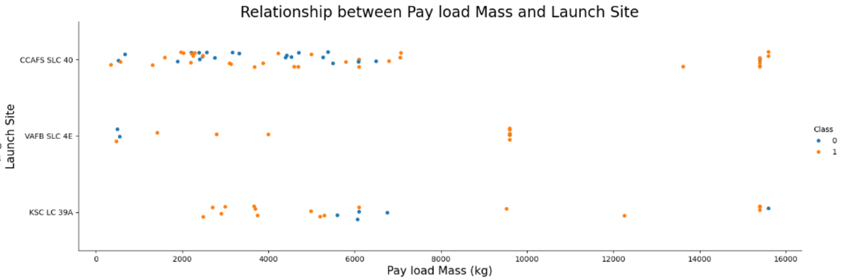
https://github.com/AtlasScientist/Jupyter-Notebook/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

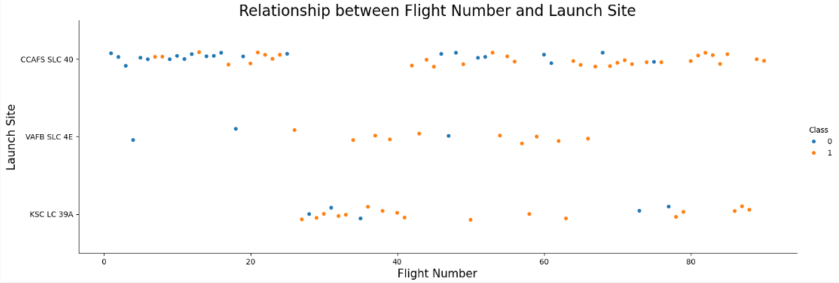
EDA with Data Visualization

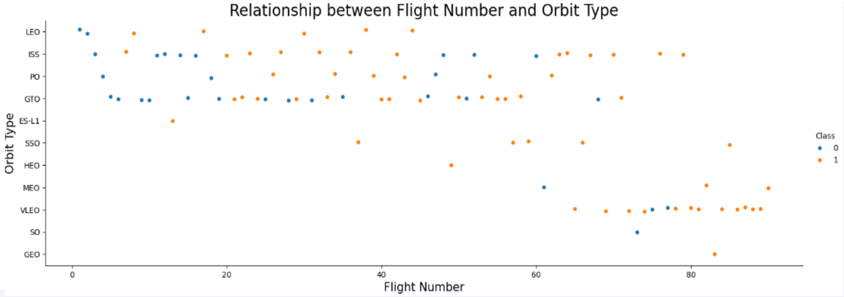
- Scatter plot point charts were used to visualize the relationship between the Flight Number and Pay Load Mass; between the Pay Load Mass and Launch Site; between the Flight Number and Launch Site; between the Flight Number and Orbit Type.
- Bar chart was used to visualize the relationship between Succes Rate of each Orbit Type.
- Line chart was used to visualize the Succes Rate of each Launch per Year.

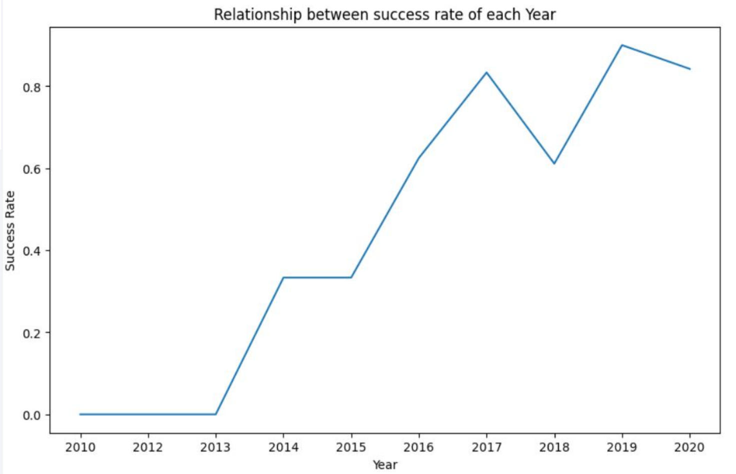












https://github.com/AtlasScientist/Jupyter-Notebook/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

SQL Queries performed included:

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

https://github.com/AtlasScientist/Jupyter-Notebook/blob/main/jupyter-labs-eda-sql-coursera_sqllite%20(1).ipynb

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.

https://github.com/AtlasScientist/Jupyter-Notebook/blob/main/lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

Dashboard has dropdown, pie chart, rangeslider and scatter plot components

- Dropdown allows a user to choose the launch site or all launch sites
- (dash_core_components.
- Pie chart shows the total success and the total failure for the launch site chosen with the
- dropdown component (plotly.express.
- 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)

https://github.com/AtlasScientist/Jupyter-Notebook/blob/main/spacex_dash_app.py

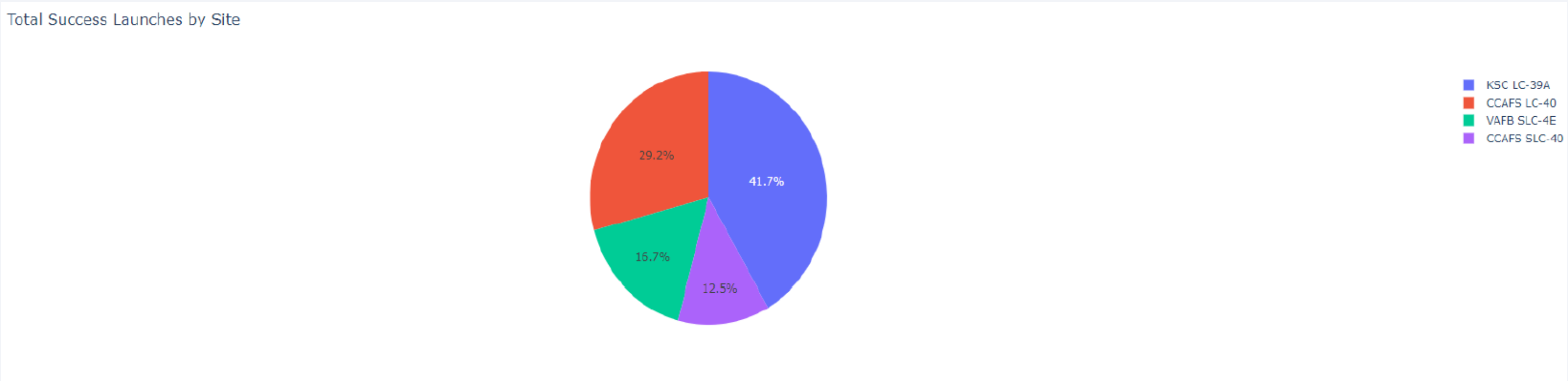
Predictive Analysis (Classification)

- 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)

https://github.com/AtlasScientist/Jupyter-Notebook/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

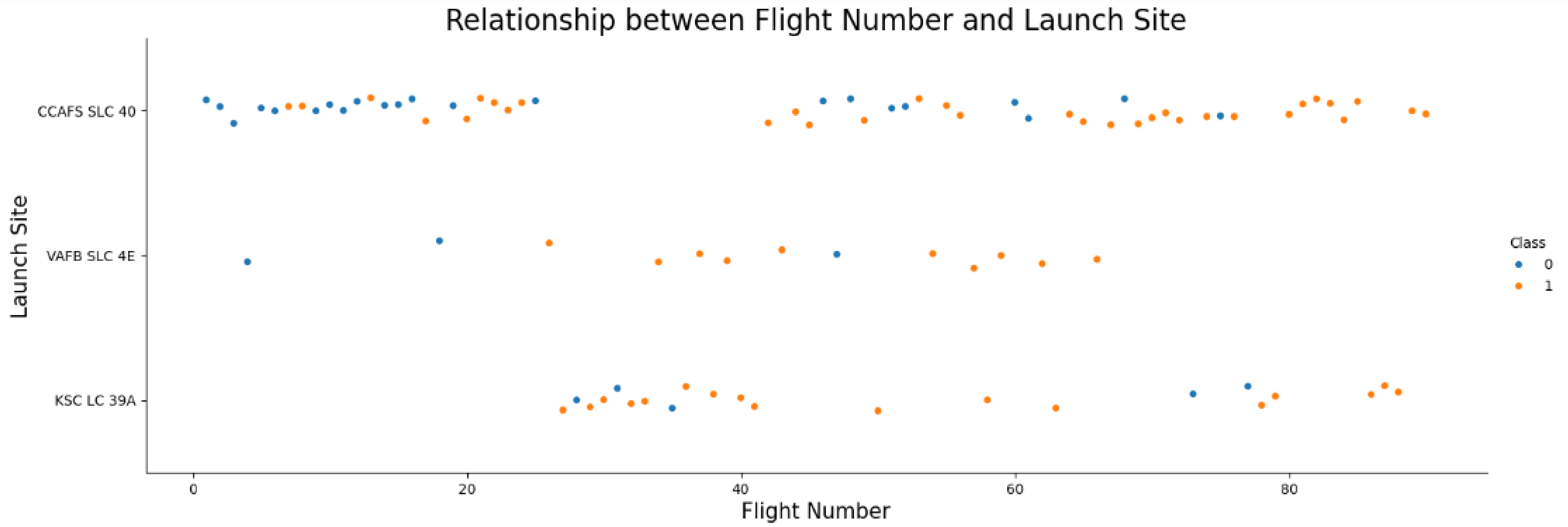
- LR, SVM, KNN are top-performing models for forecasting outcomes in this data. Lighter payloads have a higher performance compared to heavier ones.
- The likelihood of a SpaceX launch succeeding increases with the number of years of experience, suggesting a trend towards flawless launches over time.
- Launch Complex 39A at Kennedy Space Center has the highest number of successful launches compared to other launch sites.
- GEO,HEO,SSO,ES L1 orbit types exhibit the highest rates of successful launches.



Section 2

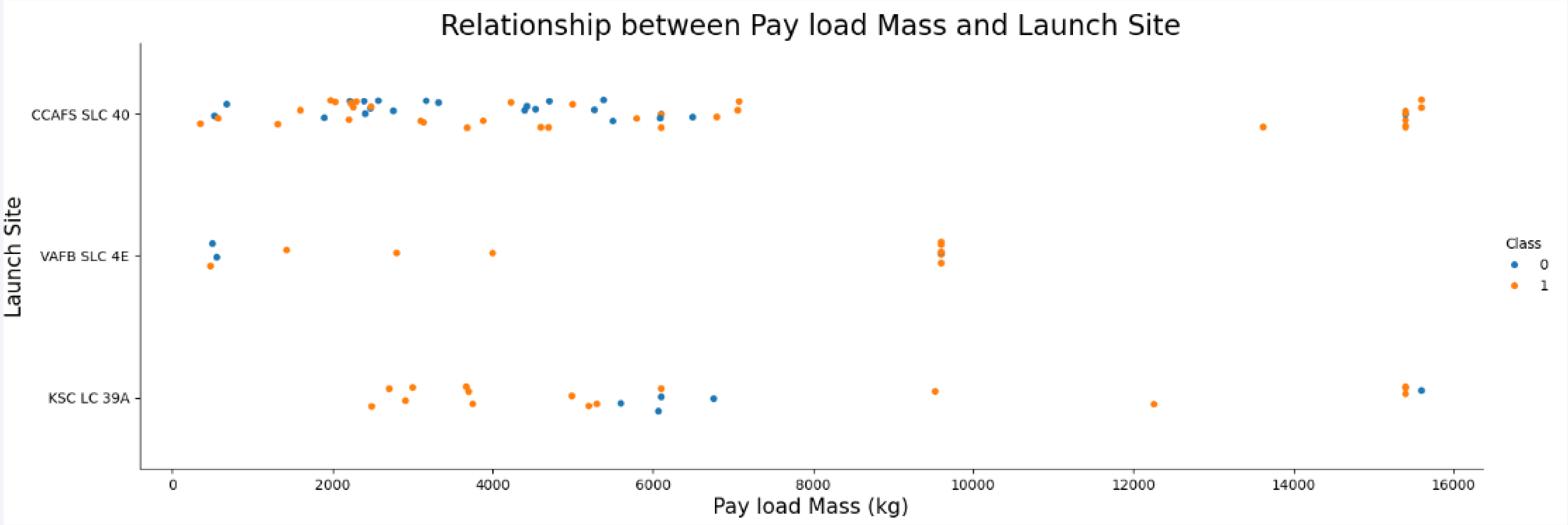
Insights drawn from EDA

Flight Number vs. Launch Site



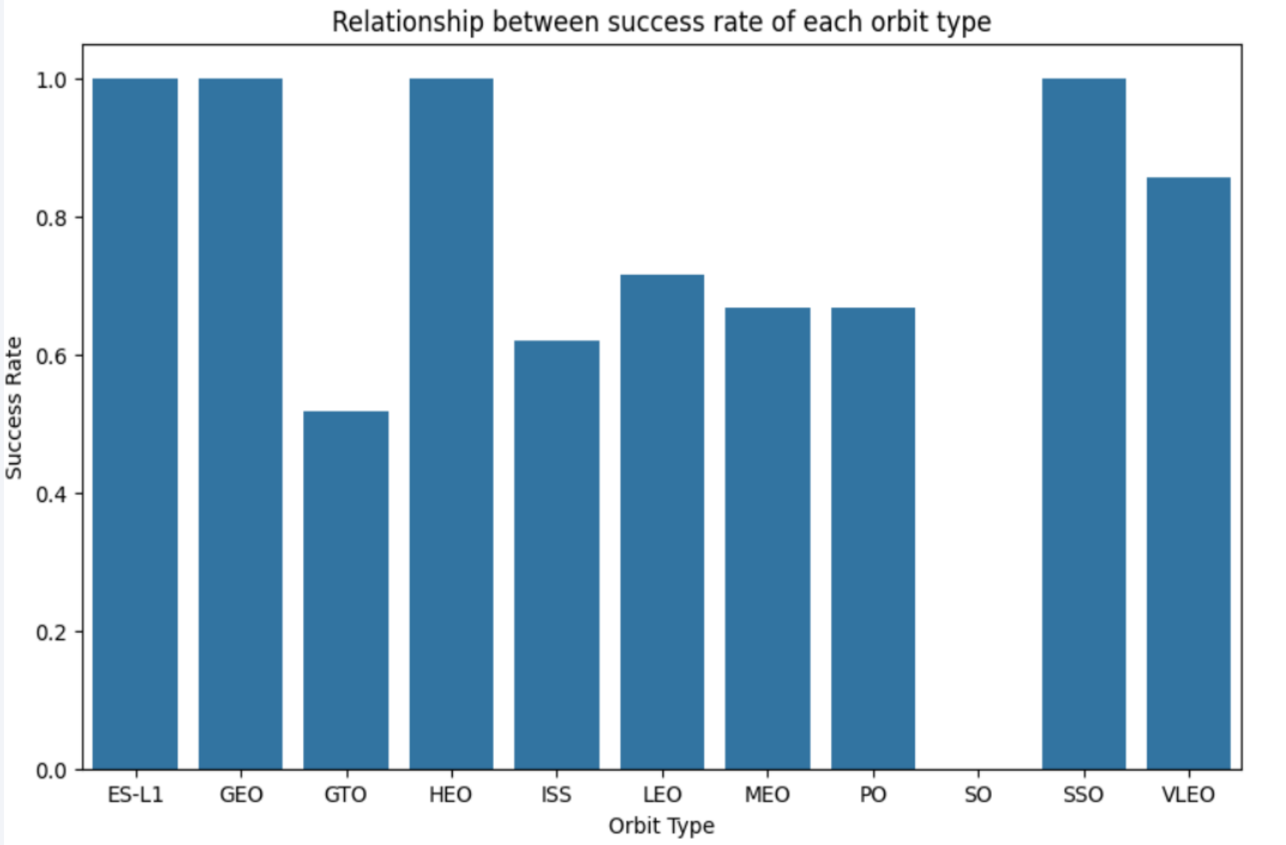
We observe that, for each site, the success rate is increasing.

Payload vs. Launch Site



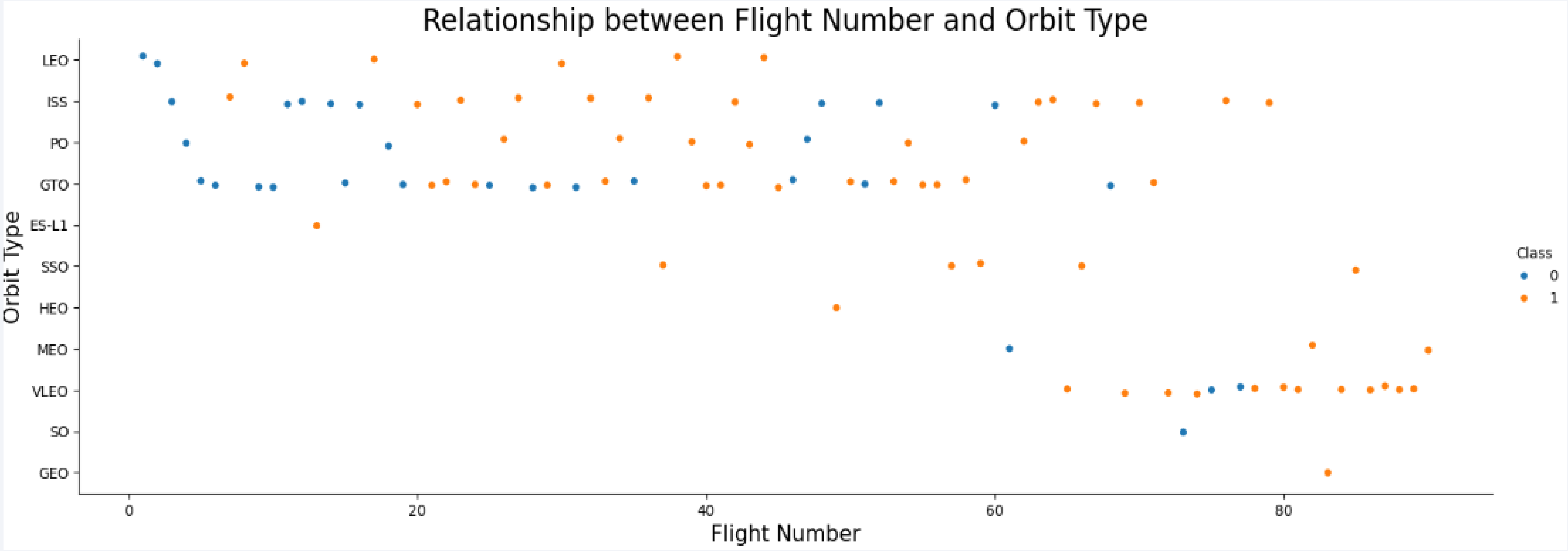
Depending on the launch site, a heavier payload may be a consideration for a successful landing. On the other hand, a too heavy payload can make a landing fail.

Success Rate vs. Orbit Type



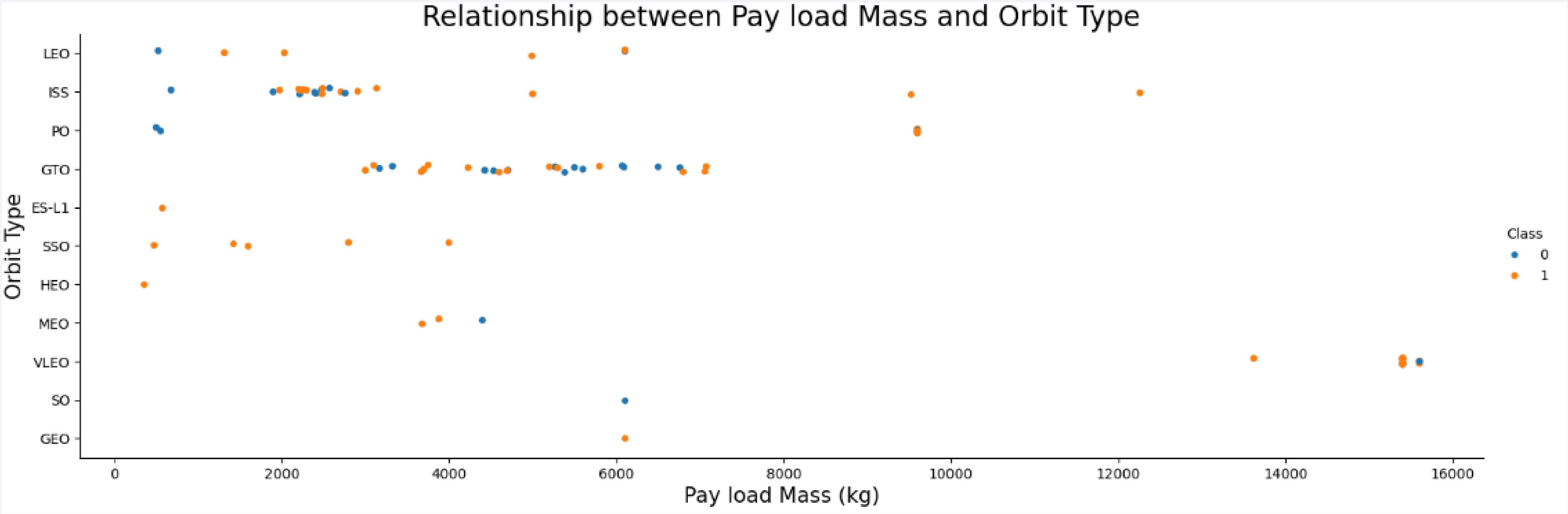
With this plot, we can see success rate for different orbit types. We note that ES L1, GEO, HEO, SSO have the best success rate.

Flight Number vs. Orbit Type



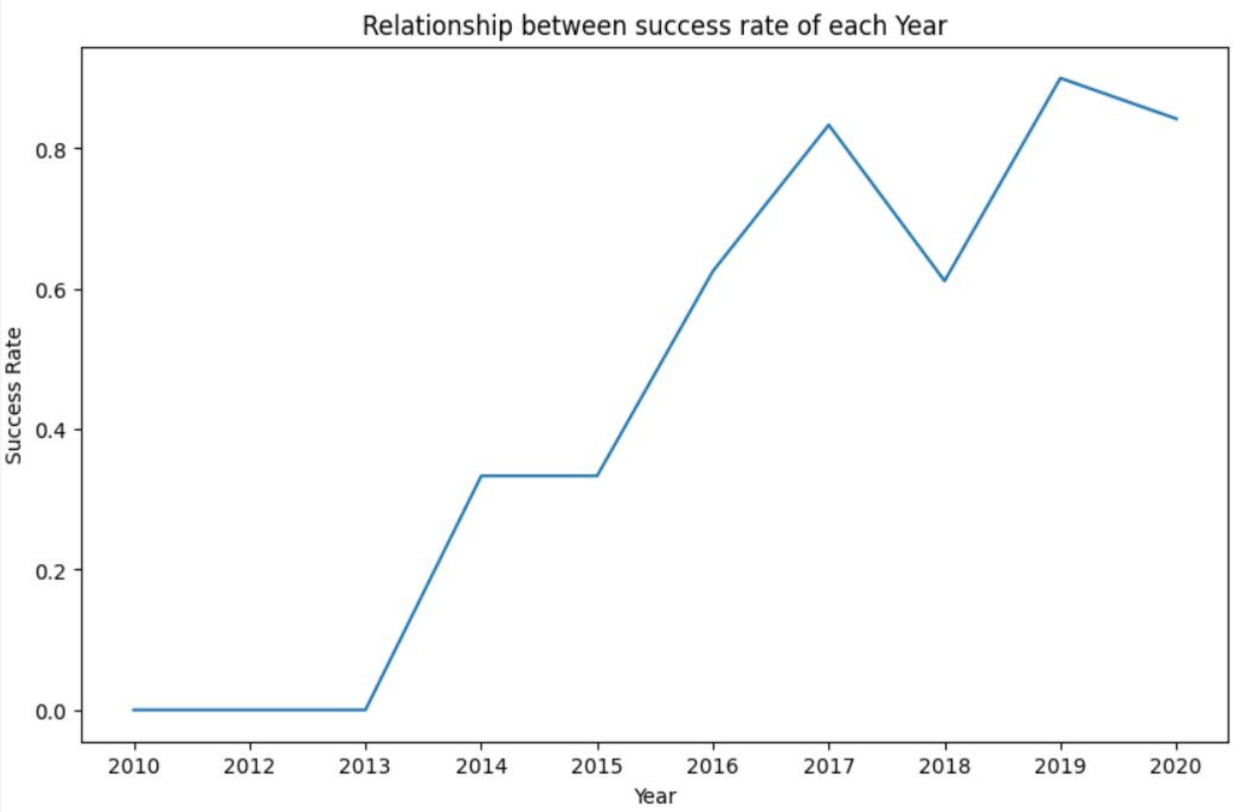
We observe that in the LEO orbit, succes is related to the number of flights whereas in the GEO orbit, there is no relationship between flight number and the orbit,

Payload vs. Orbit Type



We can observe that with heavy payloads, the succesful landing are more for PO, LEO and ISS orbits.

Launch Success Yearly Trend



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

All Launch Site Names

```
%sql SELECT DISTINCT(Launch_Site) FROM SPACEXTABLE
```

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

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

Performed an SQL query to obtain all launch sites names

Launch Site Names Begin with 'CCA'

```
%sql SELECT * FROM SPACEXTABLE 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_Outcon
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachut
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 (parachut
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attem
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attem
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attem

Performed an SQL query to obtain 5 launch sites names that begin with 'CCA'

Total Payload Mass

```
%sql SELECT PAYLOAD_MASS__KG_, Customer, Booster_Version FROM SPACEXTABLE WHERE Customer LIKE '%NASA%' GROUP BY Customer
```

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

PAYLOAD_MASS__KG_	Customer	Booster_Version
6460	Iridium Communications GFZ , NASA	F9 B4 B1043.2
12055	NASA (CCD)	F9 B5B1051.1
12530	NASA (CCDev)	F9 B5B1058.1
12500	NASA (CCP)	F9 B5B1061.1
525	NASA (COTS)	F9 v1.0 B0005
0	NASA (COTS) NRO	F9 v1.0 B0004
500	NASA (CRS)	F9 v1.0 B0006
2617	NASA (CRS), Kacific 1	F9 B5B1059.1
12050	NASA (CTS)	F9 B5 B1046.4
362	NASA (LSP)	F9 B4 B1045.1
553	NASA (LSP) NOAA CNES	F9 v1.1 B1017
1192	NASA / NOAA / ESA / EUMETSAT	F9 B5B1063.1
570	U.S. Air Force NASA NOAA	F9 v1.1 B1013

Performed an SQL query to obtain the total Payload mass carried by boosters launched by NASA (CRS).

Average Payload Mass by F9 v1.1

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Booster_Version LIKE 'F9 v1.1%'
```

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

AVG(PAYLOAD_MASS__KG_)
2534.6666666666665

Performed an SQL query to calculate the average Payload mass carried by booster version F9 V1.1

First Successful Ground Landing Date

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

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

MIN(Date)
2015-12-22

Performed an SQL query to finds the dates of the first successful landing outcome on ground pad

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql SELECT Booster_Version FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (drone ship)' and PAYLOAD_MASS__KG_ BETWEEN
```

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

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

Performed an SQL query to list names of boosters which have successful landed on drone ship and had payloads mass between 4000 and 6000 kilograms.

Total Number of Successful and Failure Mission Outcomes

```
%sql SELECT Mission_Outcome, COUNT(Mission_Outcome) AS Quantity FROM SPACEXTABLE GROUP BY Mission_Outcome
```

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

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

Performed an SQL query to calculate the total number of successful and failure mission outcomes.

Boosters Carried Maximum Payload

```
%sql SELECT Booster_Version FROM SPACEXTABLE where PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE)
```

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

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

Performed an SQL query to list the names of the boost which have carried the maximum payload mass.

2015 Launch Records

```
%sql SELECT substr(Date, 6,2) AS Month, LANDING_OUTCOME, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE LANDING_OUTCOM
```

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

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

Performed an SQL query to list the failed landing outcome in drone ships, their booster version and launch sites names for the year 2015,

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

```
%sql SELECT COUNT(Landing_Outcome), Landing_Outcome FROM SPACEXTBL WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' GROUP
```

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

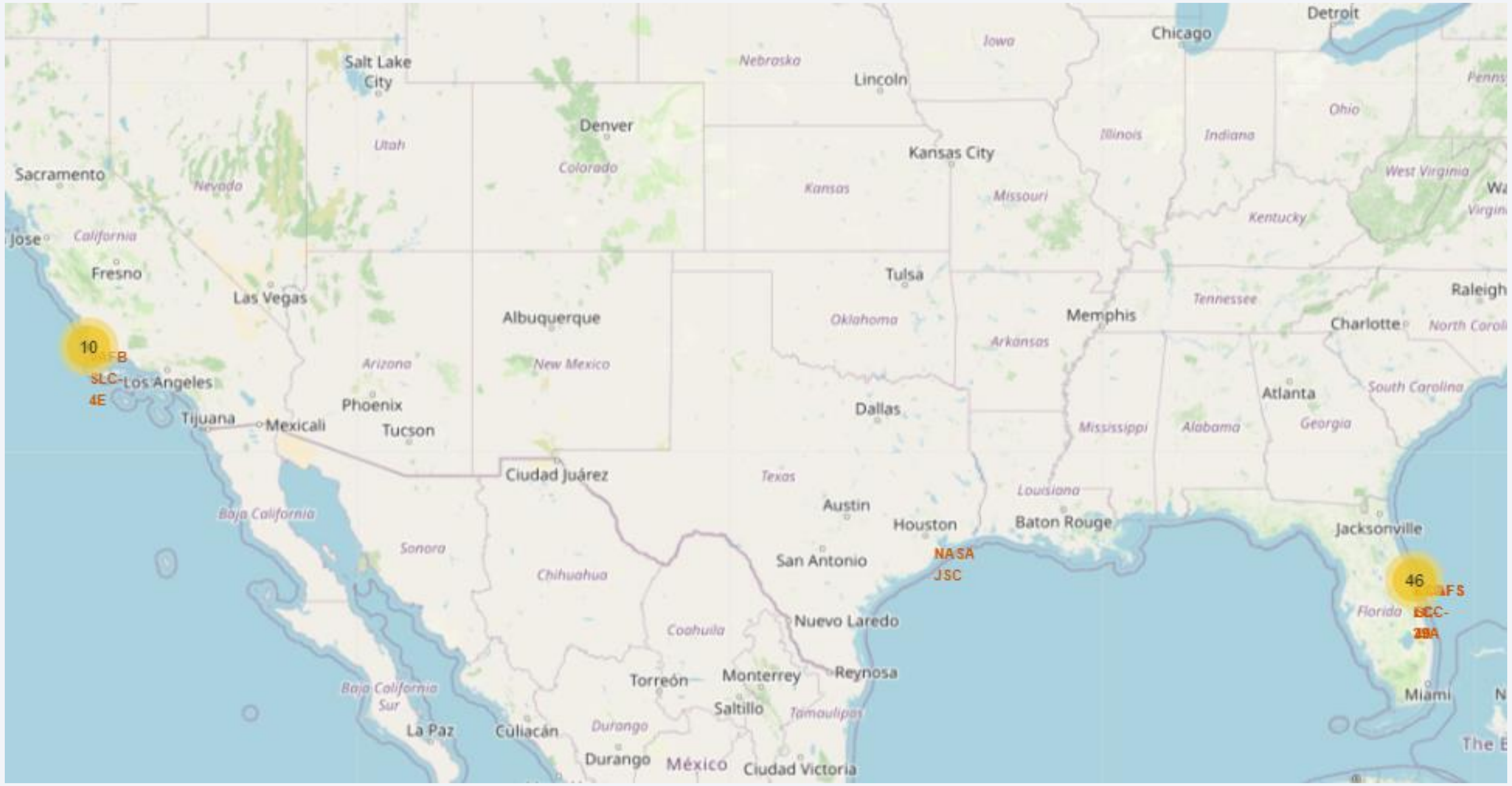
COUNT(Landing_Outcome)	Landing_Outcome
10	No attempt
5	Success (drone ship)
5	Failure (drone ship)
3	Success (ground pad)
3	Controlled (ocean)
2	Uncontrolled (ocean)
2	Failure (parachute)
1	Precluded (drone ship)

Performed an SQL query to rank the count of landing outcomes between the dates 2010-06-04 and 2017-03-20 on descending order.

Section 3

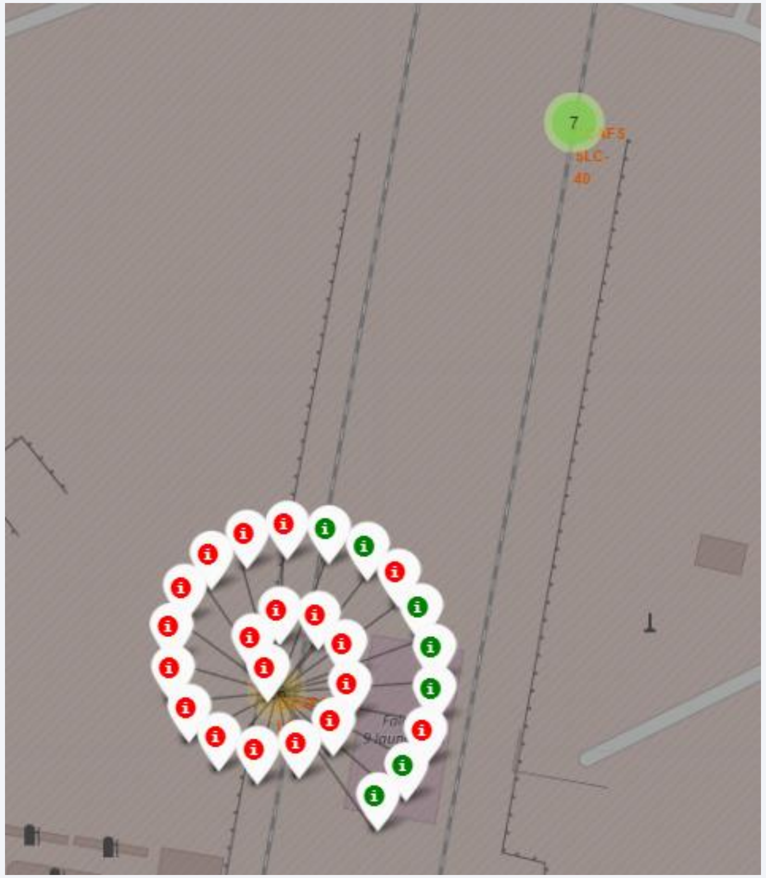
Launch Sites Proximities Analysis

Folium Map – Launch Sites



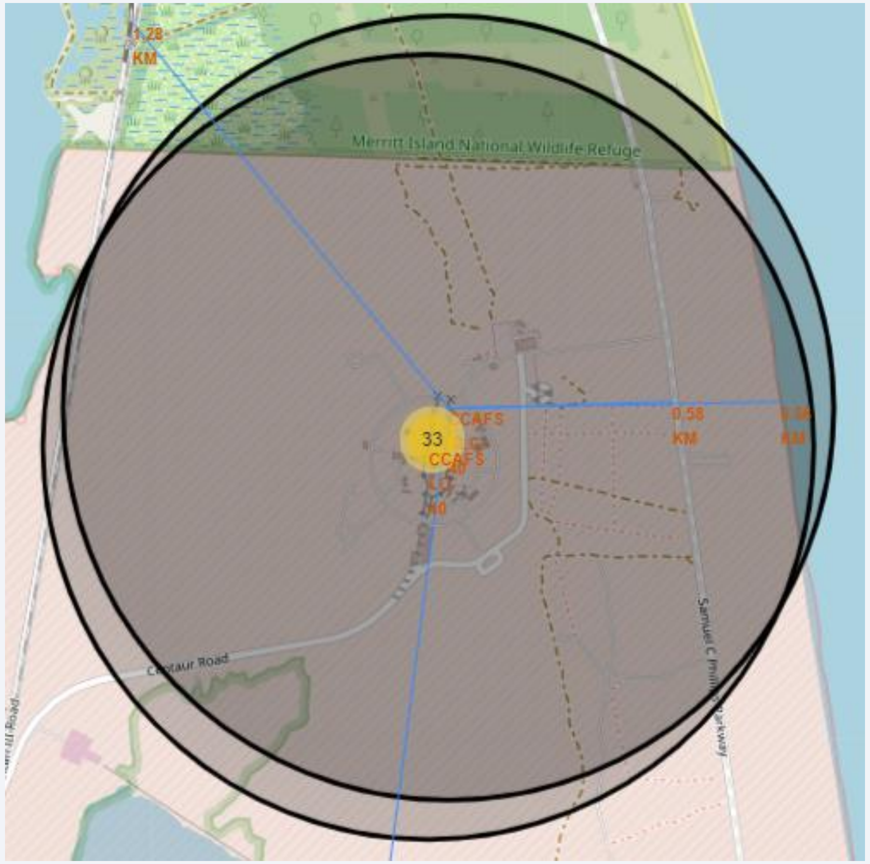
The launch sites are labelled by a marker with their names on the map

Folium Map - Color Labeled Markers



The launch records are grouped in clusters on the map, then labelled by green markers for successful lunches, and red markers for unsuccessful lunches.

Folium Map – Distances to its Proximities

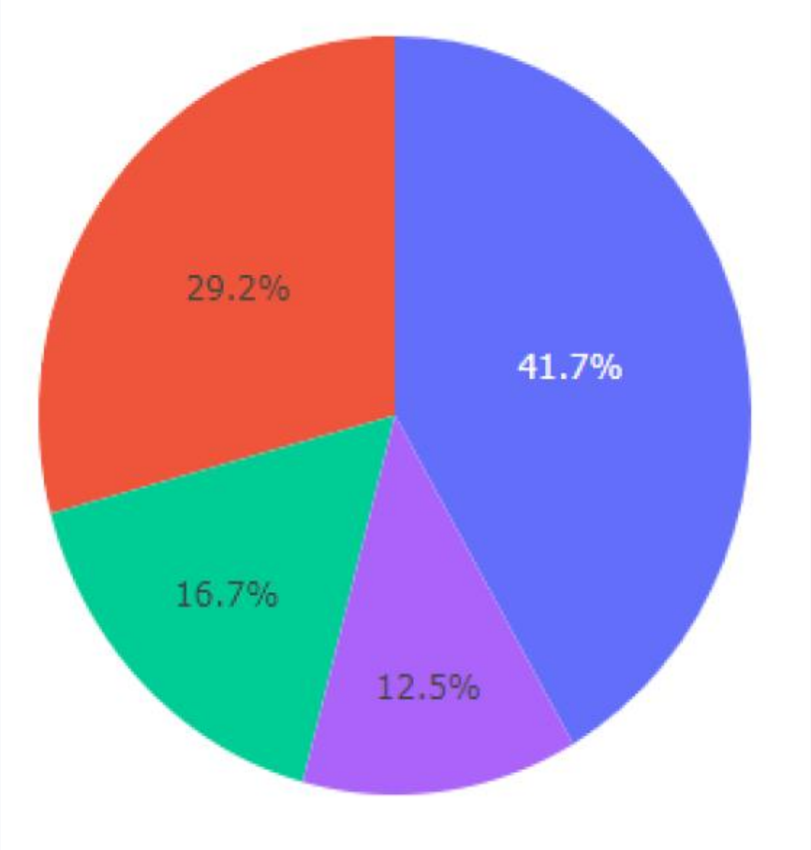


The closest coastline from NASA JSC is marked as a point using MousePosition and the distance between the coastline point and the launch site, which is approximately 0.86 km.

Section 4

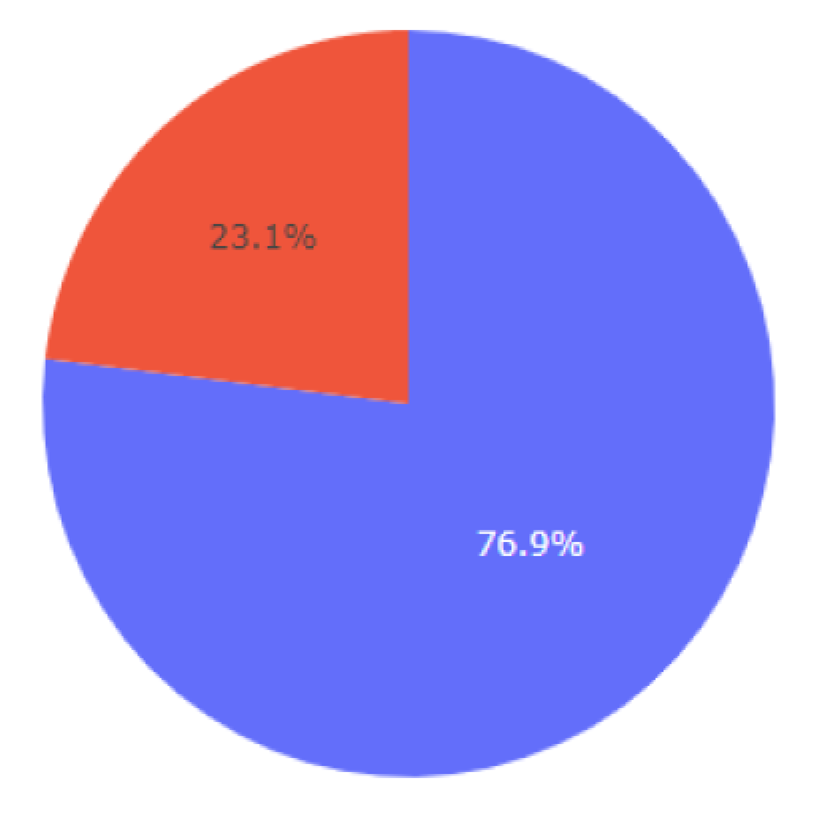
Build a Dashboard with Plotly Dash

Dashboard – Total success lunches for all sites



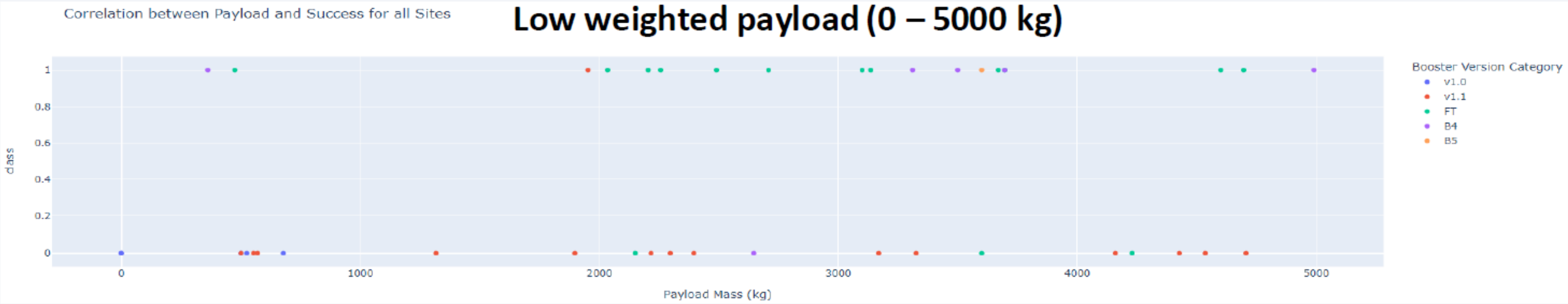
KSC LC-39A has the highest amount of success launches with 41.7% from the entire record, whereas CCAFS SLC-40 has the lowest amount of success launches with only 12.5%.

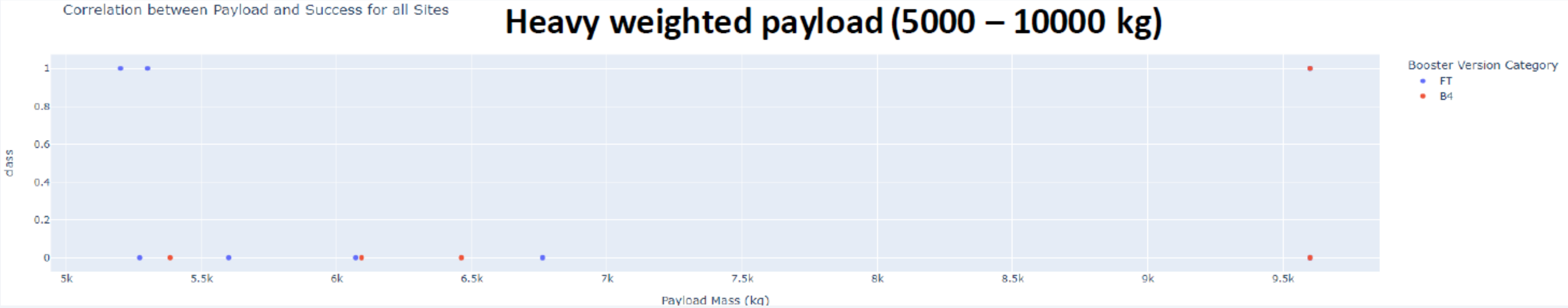
Dashboard - Total success launches for Site KSC LC 39A



KSC LC-39A which is the launch site with highest amount of success, has a 76.9% success rate for the launches from its site, and 23.1% failure rate.

Dashboard - Payload mass vs Launch Outcome





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

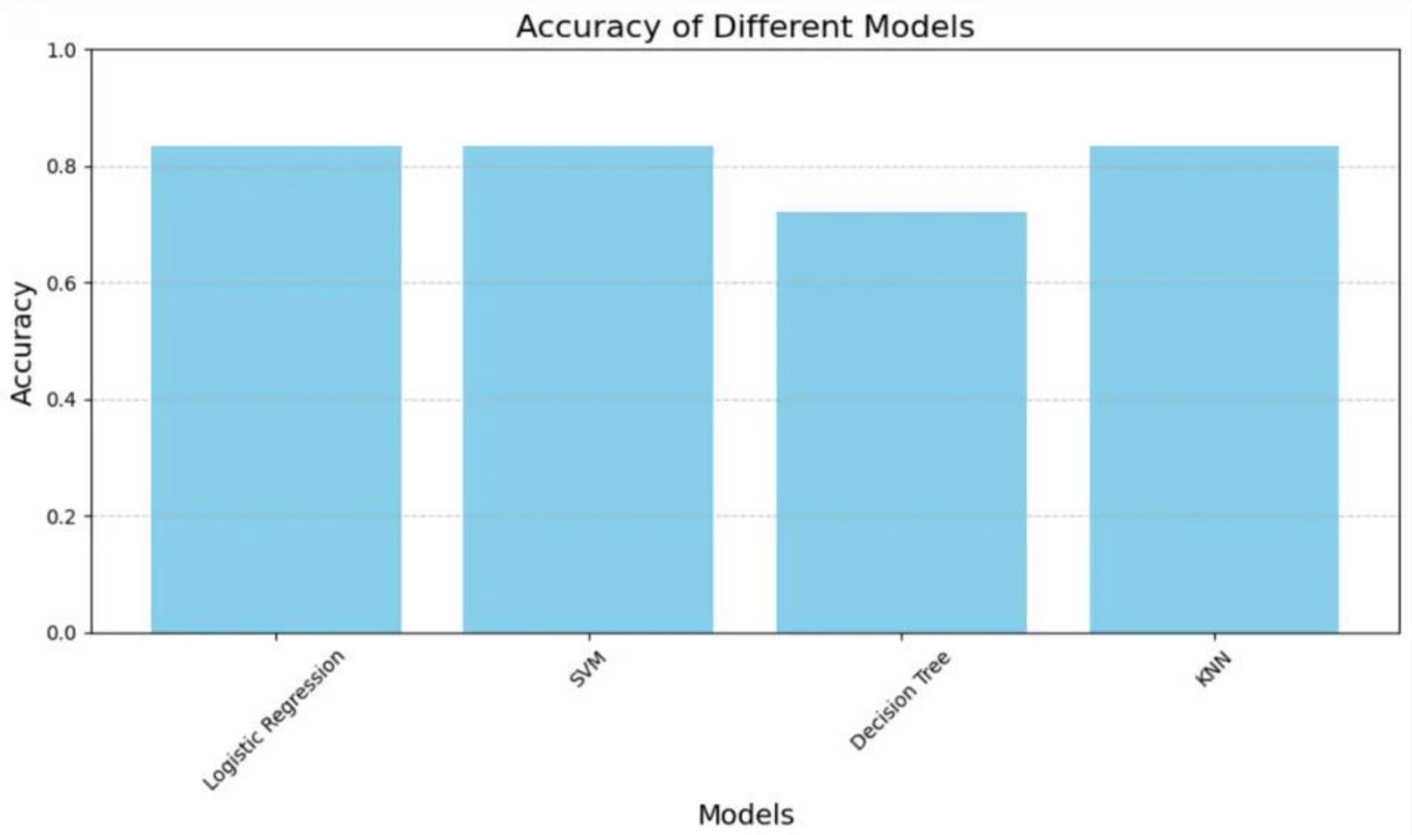
Section 5

Predictive Analysis (Classification)

Classification Accuracy

```
print('LR Accuracy:', '{:.2%}'.format(logreg_accuracy))
print( 'SVM Accuracy:', '{:.2%}'.format(svm_accuracy))
print('Decision Tree Accuracy:', '{:.2%}'.format(tree_accuracy))
print('KNN Accuracy:', '{:.2%}'.format(knn_accuracy))
```

```
LR Accuracy: 83.33%
SVM Accuracy: 83.33%
Decision Tree Accuracy: 72.22%
KNN Accuracy: 83.33%
```

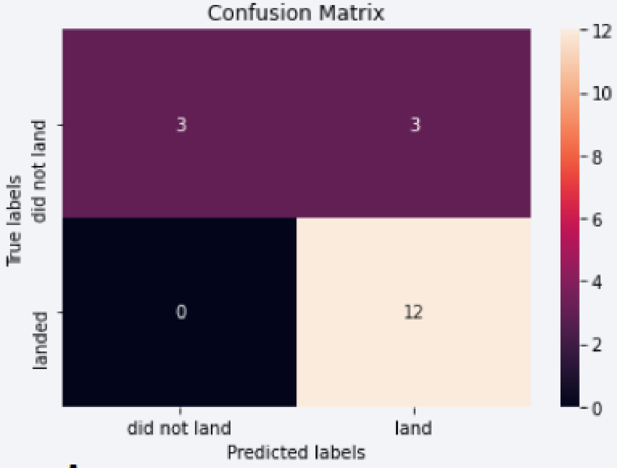


The model that performed best are LR, SVM, KNN where all 3 achieved the highest accuracy of 83.33%.

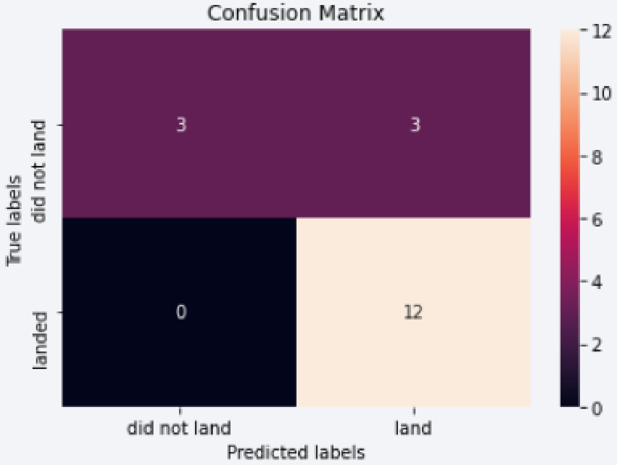
Confusion Matrix

- LR, SVM, KNN models are good as their confusion matrix show that they predicted all 12 successful landing correctly, with O error.
- However, the Decision Tree model only predicted 11 successful landing correctly, with one of them wrongly predicted as a failed / did not land.
- LR, SVM, KNN models have the same accuracy of 83.33% as displayed earlier, hence the same confusion matrix.

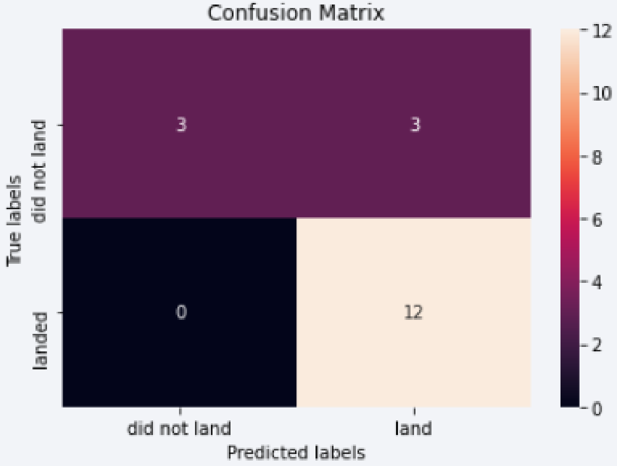
Logistic regression



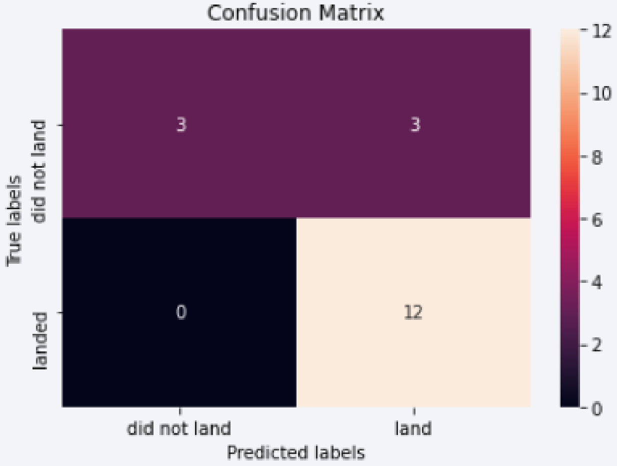
Decision Tree



kNN



SVM



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.

Thank you!