IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Antonio Domínguez
10th March 2025

Outline

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

Executive Summary

Summary of methodologies

- Data collection
- Data wrangling
- EDA with Data Visualization
- EDA with SQL
- Visual Analysis with Folium
- Visual Analysis with Plotly Dash
- Machine Learning Prediction

Introduction

- Project background, context and problems to find answers

According the information provided by SpaceX on its website, the Falcon 9 rocket launches costs are around one third cheaper than its competitor´s launches.

The aim of the project is to determine if the first stage of the Falcon 9 rockets will land successfully. Because the cost of launching a rocket is closely related to it, this knowledge becomes very important for providers.

Section 1

Methodology

Methodology

- Data collection methodology: The data were acquired from two sources:
  - SpaceX Open Source Rest API : https://api.spacexdata.com/v4/rockets/
  - From Wikipedia through a web scraping process. https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches
- Perform data wrangling
  - Where the data was cleaned from unnecessary information.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Where the analysis was carry out through the Logistic Regression, Support Vector Machines, K-NN and Decision Tree methods in order to determine the best accuracy model.

Data Collection

The data were acquired from two different sources:

- SpaceX Open Source Rest API
- From an information table of Wikipedia through a web scraping process.

More specifically, the launch records are stored in a HTML table shown below:

2020 [edit]

In late 2019, Gwynne Shotwell stated that SpaceX hoped for as many as 24 launches for Starlink satellites in 2020,[490] in addition to 14 or 15 non-Starlink launches. At 26 launches, 13 of which for Starlink satellites, Falcon 9 had its most prolific year, and Falcon rockets were second most prolific rocket family of 2020, only behind China's Long March rocket family.[491]

[hide] Flight No.	Date and time (UTC)	Version, Booster[b]	Launch site	Payload[c]	Payload mass	Orbit	Customer	Launch outcome	Booster landing
78	7 January 2020, 02:19:21[492]	F9 B5 ♺ B1049.4	CCAFS, SLC-40	Starlink 2 v1.0 (60 satellites)	15,600 kg (34,400 lb)[5]	LEO	SpaceX	Success	Success (drone ship)
	Third large batch and second operational flight of Starlink constellation. One of the 60 satellites included a test coating to make the satellite less reflective, and thus less likely to interfere with ground-based astronomical observations.[493]								
79	19 January 2020, 15:30[494]	F9 B5 ♺ B1046.4	KSC, LC-39A	Crew Dragon in-flight abort test[495] (Dragon C205.1)	12,050 kg (26,570 lb)	Sub-orbital[496]	NASA (CTS)[497]	Success	No attempt
	An atmospheric test of the Dragon 2 abort system after Max Q. The capsule fired its SuperDraco engines, reached an apogee of 40 km (25 mi), deployed parachutes after reentry, and splashed down in the ocean 31 km (19 mi) downrange from the launch site. The test was previously slated to be accomplished with the Crew Dragon Demo-1 capsule;[498] but that test article exploded during a ground test of SuperDraco engines on 20 April 2019.[419] The abort test used the capsule originally intended for the first crewed flight.[499] As expected, the booster was destroyed by aerodynamic forces after the capsule aborted.[500] First flight of a Falcon 9 with only one functional stage — the second stage had a mass simulator in place of its engine.								
80	29 January 2020, 14:07[501]	F9 B5 ♺ B1051.3	CCAFS, SLC-40	Starlink 3 v1.0 (60 satellites)	15,600 kg (34,400 lb)[5]	LEO	SpaceX	Success	Success (drone ship)
	Third operational and fourth large batch of Starlink satellites, deployed in a circular 290 km (180 mi) orbit. One of the fairing halves was caught, while the other was fished out of the ocean.[502]								
81	17 February 2020, 15:05[503]	F9 B5 ♺ B1056.4	CCAFS, SLC-40	Starlink 4 v1.0 (60 satellites)	15,600 kg (34,400 lb)[5]	LEO	SpaceX	Success	Failure (drone ship)
	Fourth operational and fifth large batch of Starlink satellites. Used a new flight profile which deployed into a 212 km × 386 km (132 mi × 240 mi) elliptical orbit instead of launching into a circular orbit and firing the second stage engine twice. The first stage booster failed to land on the drone ship[504] due to incorrect wind data.[505] This was the first time a flight proven booster failed to land.								
82	7 March 2020, 04:50[506]	F9 B5 ♺ B1059.2	CCAFS, SLC-40	SpaceX CRS-20 (Dragon C112.3 ♺)	1,977 kg (4,359 lb)[507]	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
	Last launch of phase 1 of the CRS contract. Carries *Bartolomeo*, an ESA platform for hosting external payloads onto ISS.[508] Originally scheduled to launch on 2 March 2020, the launch date was pushed back due to a second stage engine failure. SpaceX decided to swap out the second stage instead of replacing the faulty part.[509] It was SpaceX's 50th successful landing of a first stage booster, the third flight of the Dragon C112 and the last launch of the cargo Dragon spacecraft.								
83	18 March 2020, 12:16[510]	F9 B5 ♺ B1048.5	KSC, LC-39A	Starlink 5 v1.0 (60 satellites)	15,600 kg (34,400 lb)[5]	LEO	SpaceX	Success	Failure (drone ship)
	Fifth operational launch of Starlink satellites. It was the first time a first stage booster flew for a fifth time and the second time the fairings were reused (Starlink flight in May 2019).[511] Towards the end of the first stage burn, the booster suffered premature shut down of an engine, the first of a Merlin 1D variant and first since the CRS-1 mission in October 2012. However, the payload still reached the targeted orbit.[512] This was the second Starlink launch booster landing failure in a row, later revealed to be caused by residual cleaning fluid trapped inside a sensor.[513]								
84	22 April 2020, 19:30[514]	F9 B5 ♺ B1051.4	KSC, LC-39A	Starlink 6 v1.0 (60 satellites)	15,600 kg (34,400 lb)[5]	LEO	SpaceX	Success	Success (drone ship)

Data Collection – SpaceX API

The process to obtain Data Collection was as follows:

- Request and parse data
- Filter the data related to Falcon 9
- Clean the data

GitHub URL of the SpaceX API calls notebook:

https://github.com/Aedm12/Applied-Data-Science-Capstone/blob/main/Module%201%20a-%20jupyter-labs-spacex-data-collection-api.ipynb

Task 1: Request and parse the SpaceX launch data using the GET request

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_
```

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
import pandas as pd
import json
from pandas.io.json import json_normalize

static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_
response = requests.get(static_json_url)

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

print(data)
```

Finally lets construct our dataset using the data we have obtained. We we combine the columns 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}
```

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)
print(df)
```

Task 2: Filter the dataframe to only include `Falcon 9` launches

Finally we will remove the Falcon 1 launches keeping only the Falcon 9 launches. Filter the data dataframe using the `BoosterVersion` column to only keep the Falcon 9 launches. Save the filtered data to a new dataframe called `data_falcon9`.

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

Data Collection - Scraping

The scraping process was as follows:

- The data was obtained from Wikipedia
- Colum names were extracted from html table header
- A data frame was created by parsing process

GitHub URL of the web scraping notebook:

https://github.com/Aedm12/Applied-Data-Science-Capstone/blob/main/Module%201%20b-%20jupyter-labs-webscraping.ipynb

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

```
# use requests.get() method with the provided static_url
# assign the response to a object
r = requests.get(static_url)
data = r.text
```

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

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

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

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

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, pl reference link towards the end of this lab

```
# 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")
```

Starting from the third table is our target table contains the actual launch records.

```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

TASK 3: Create a data frame by parsing the launch HTML tables

We will create an empty dictionary with keys from the extracted column names in the previous task. Later, this dictionary will be converted into a Pandas dataframe

```
launch_dict= dict.fromkeys(column_names)

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

# 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']=[]
```

Data Wrangling

The Data wrangling process was as follows:

- Identify and calculate the percentage of the missing values in each attribute and identify which columns are numerical and categorical.
- The number of launches on each site were calculated as well the number and occurrence of each orbit and the number and the occurrence of mission outcome of the orbits.
- Finally a landing outcome label from Outcome column was created.

GitHub URL of the web scraping notebook:

https://github.com/Aedm12/Applied-Data-Science-Capstone/blob/main/Module%201%20c-%20labs-jupyter-spacex-Data%20wrangling.ipynb

```
# 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
```

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

```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

```
0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 False Ocean
6 None ASDS
7 False RTLS
```

We create a set of outcomes where the second stage did not land successfully:

```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

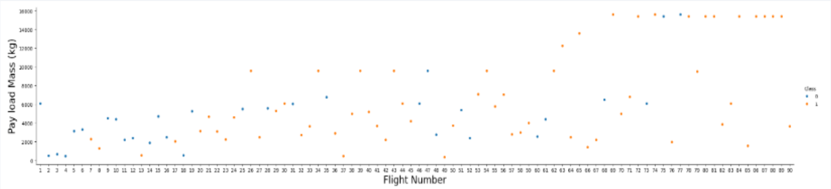
```
# Apply value_counts on Orbit column
df["Orbit"].value_counts()
```

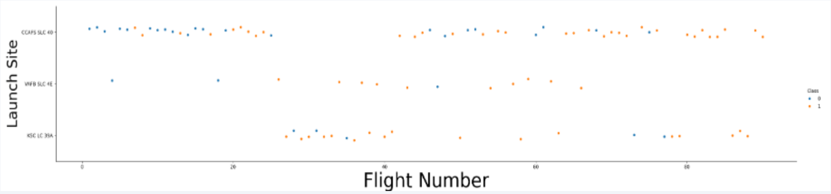
```
Orbit
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
HEO       1
ES-L1     1
SO        1
GEO       1
Name: count, dtype: int64
```

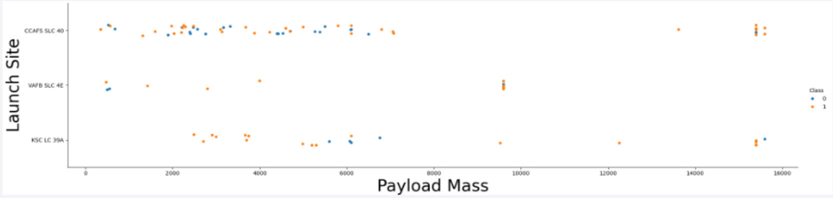
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = [0 if x in bad_outcomes else 1 for x in df['Outcome']]
```

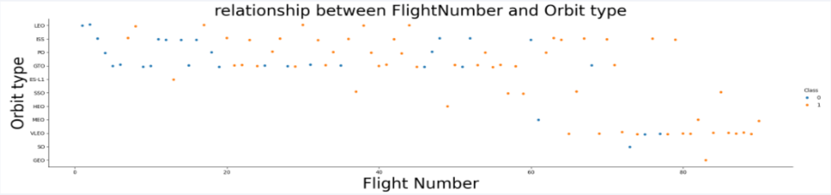
```
df['Class']=landing_class
df[['Class']].head(8)
```

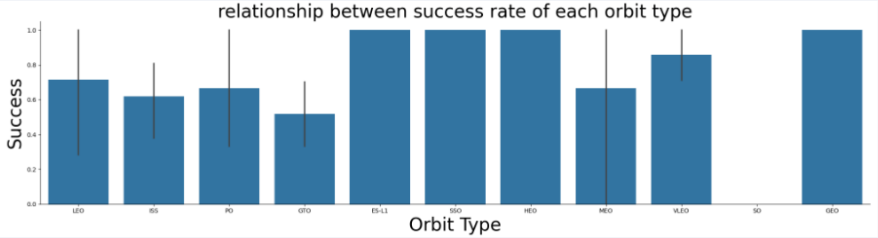
EDA with Data Visualization

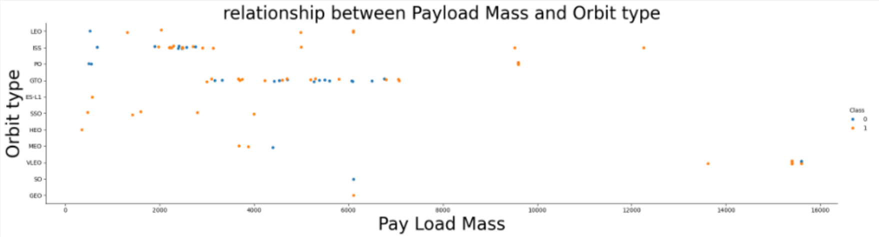


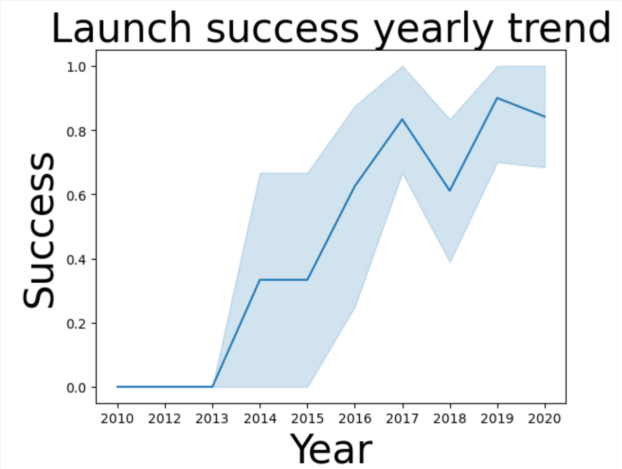












- Scater point charts are used to visualize the difference between two variables (such as Orbit type and Payload Mass or Flight Number) while bar chart can show us the relationship between success rate of each orbit tipe.
- Launch success over the years can be visualized in the line chart.

GitHub URL of the EDA with Data Visualization notebook:

https://github.com/Aedm12/Applied-Data-Science-Capstone/blob/main/Module%202%20b-%20EDA%20Exploring%20Preparing%20Visualization%20Data.ipynb

EDA with SQL

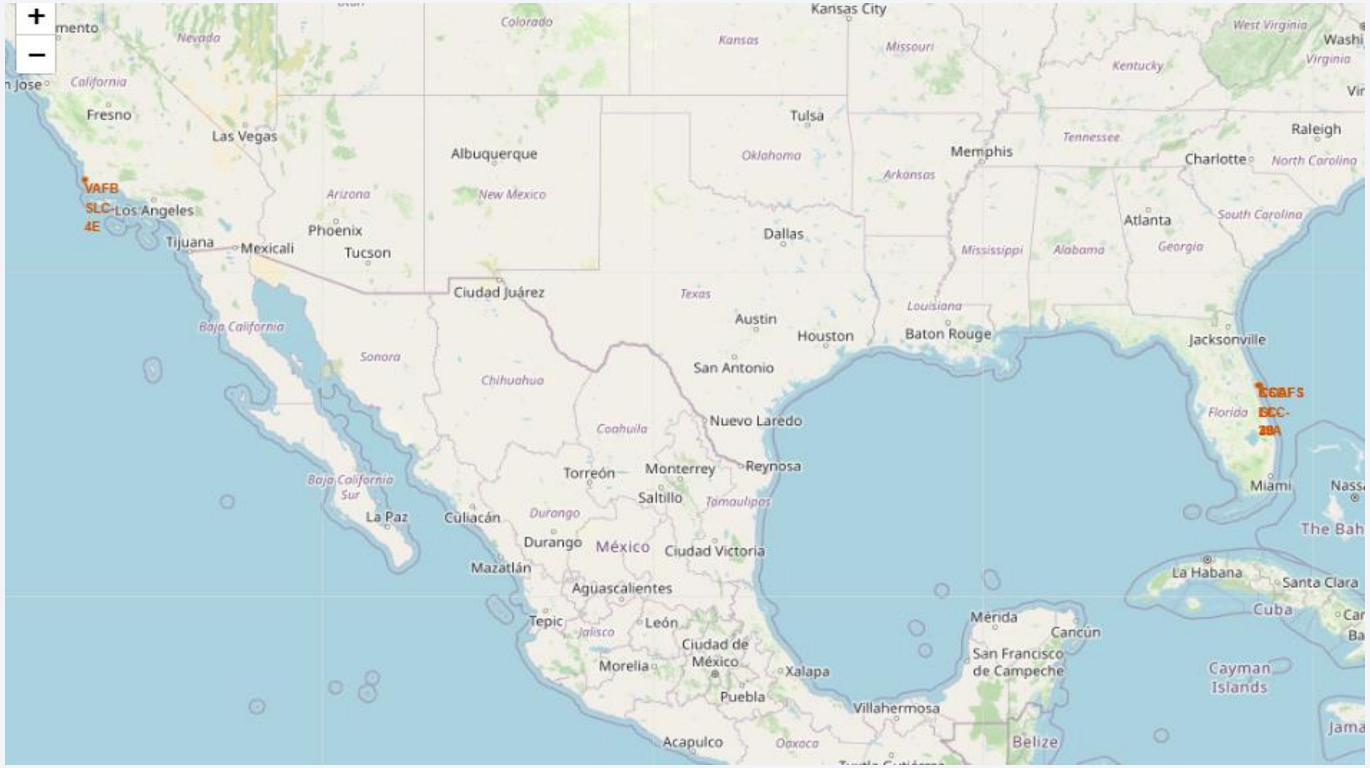
- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first succesful landing outcome in ground pad was acheived.
- 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.

GitHub URL of the EDA with SQL notebook:

https://github.com/Aedm12/Applied-Data-Science-Capstone/blob/main/Module%202%20a-%20jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

The map shows the different locations of the launch sites with a highlighted circle area and with a text label on the specific coordinates.



GitHub URL of the web scraping notebook:

https://github.com/Aedm12/Applied-Data-Science-Capstone/blob/main/Module%203%20-%20Folium%20-%20lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- Summarize what plots/graphs and interactions you have added to a dashboard
- Explain why you added those plots and interactions
- Add the GitHub URL of your completed Plotly Dash lab, as an external reference and peer-review purpose

GitHub URL of the web scraping notebook:

Predictive Analysis (Classification)

- Load the Dataframe
- Perform exploratory Data Analysis and determine Training Labels
- Create a column for the class
- Standardize the data
- Split into training data and test data
- Find best Hyperparameter for SVM, Classification Trees and Logistic Regression
- Find the method performs best using test data

GitHub URL of the Predictive Analysis notebook:

https://github.com/Aedm12/Applied-Data-Science-Capstone/blob/main/Module%204%20-%20SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

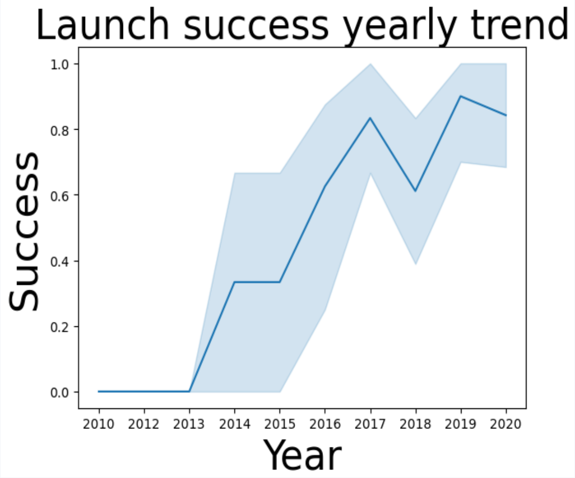
Results

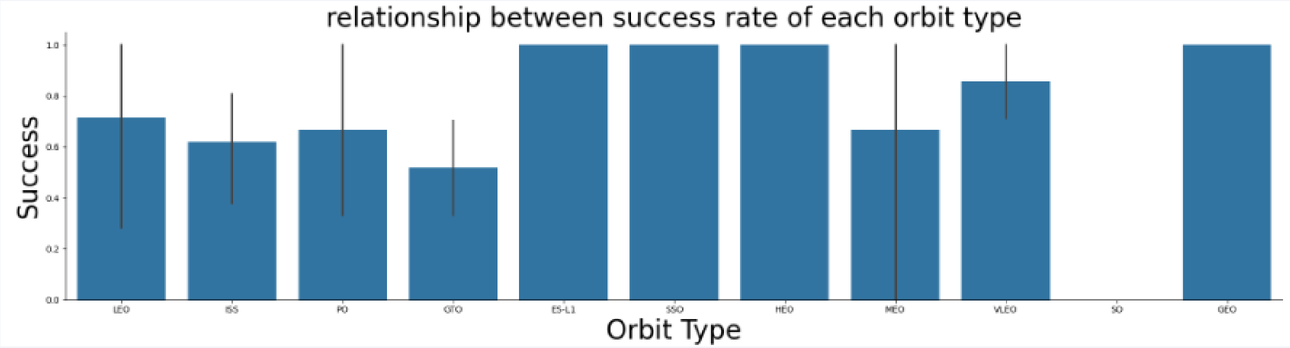
We can found results on 5 different ways

- Exploratory data analysis with SQL
- Exploratory data analysis through Matplotlib and Seaborn visualizations
- Visual Analysis with Folium
- Visual Analysis with Plotly Dash
- Predictive analysis

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

Landing_Outcome	RESULTADOS SOLICITADOS
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

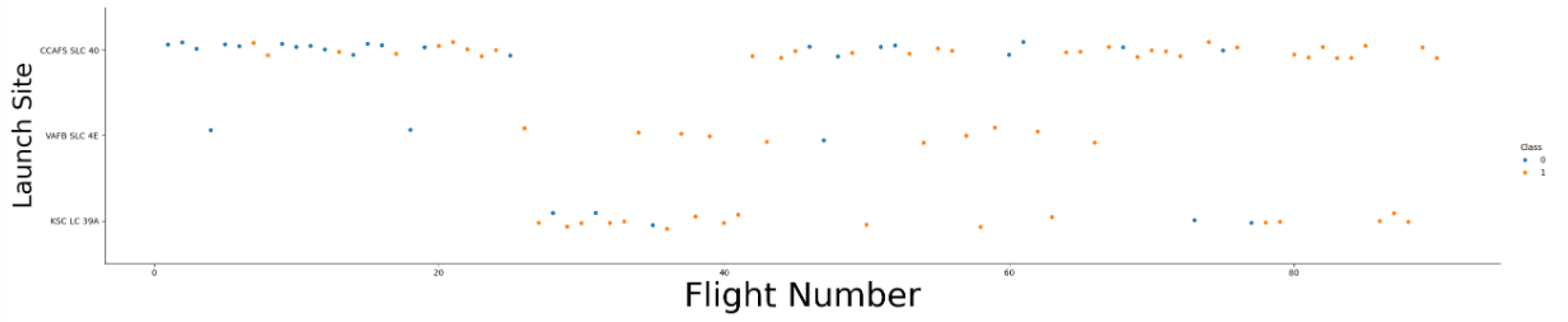




Section 2

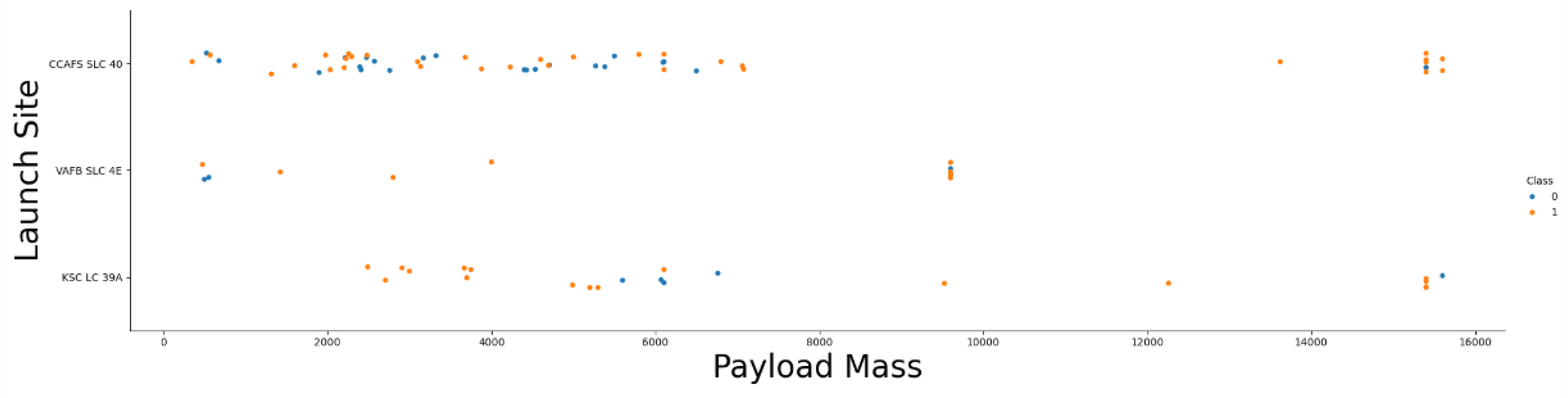
Insights drawn from EDA

Flight Number vs. Launch Site



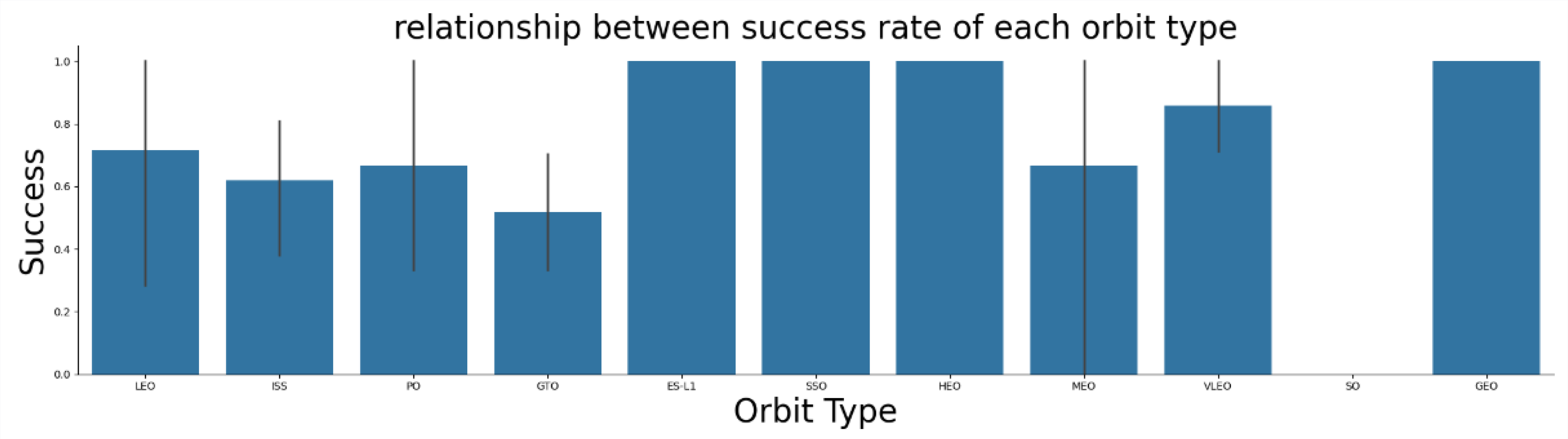
The first Launch Site (CCAFS SLC 40) was the most frequently used from the beginning and still remains the most used. It´s activity only decreased when KSC LC 39A began to work but, we do not why, CCAFS SLC 40 recovered soon it´s activity while KSC LC 39A lost it.

Payload vs. Launch Site



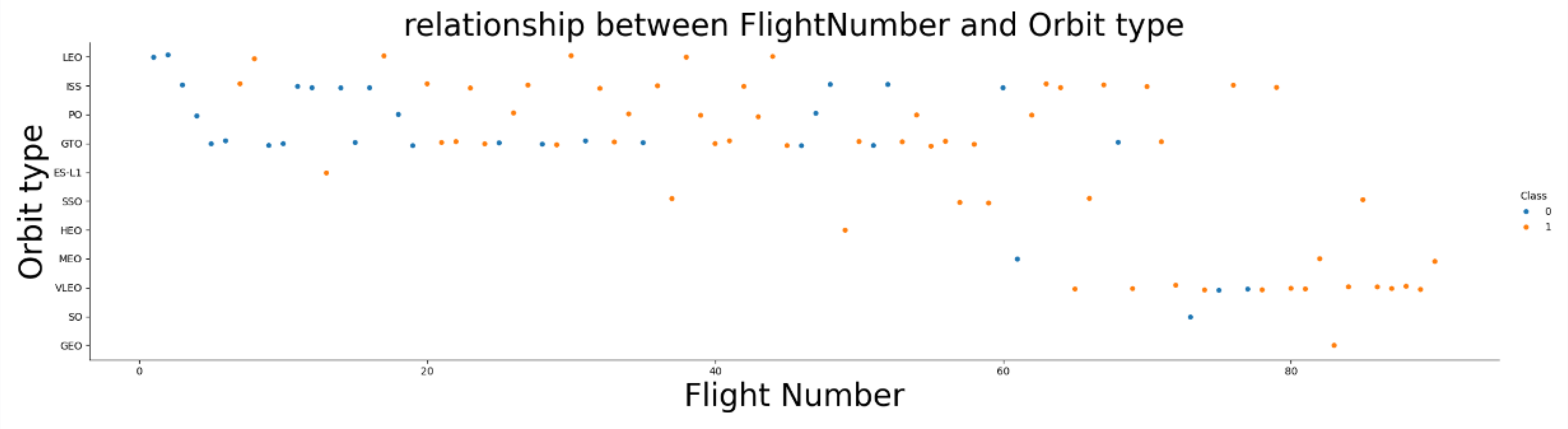
In general low Payload Mass launches are more frequents. The heavy ones were launched more often from CCAFS SLC 40.

Success Rate vs. Orbit Type



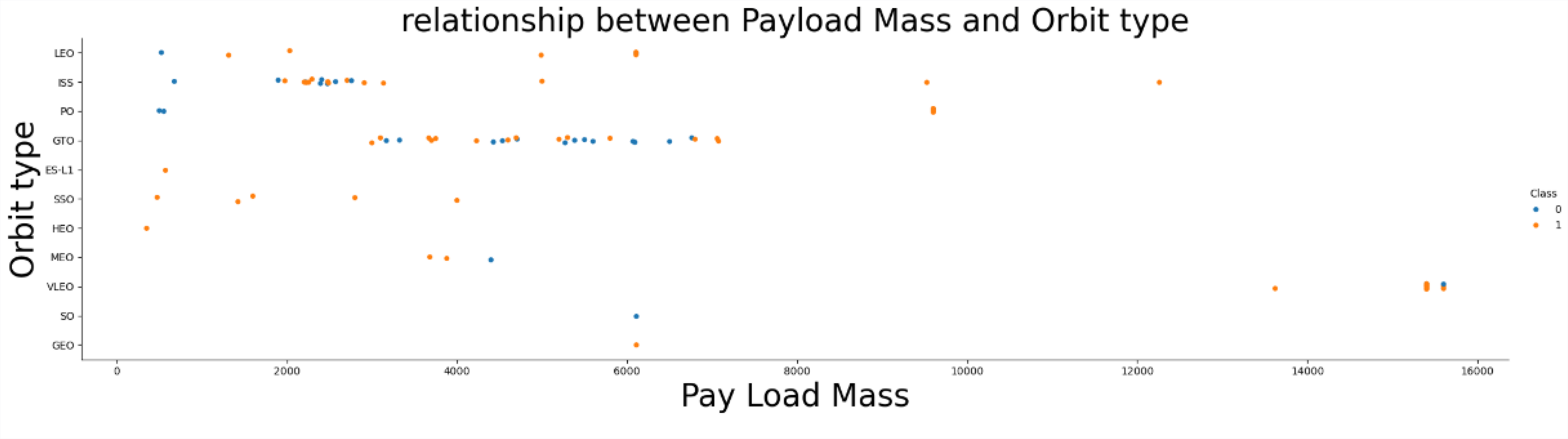
The most successful Orbit Types were ES-L1, SSO, HEO and GEO

Flight Number vs. Orbit Type



The first flights has a low success that was increasing maybe due the experience while the success seems to increase faster in the last years when different orbits were adopted.

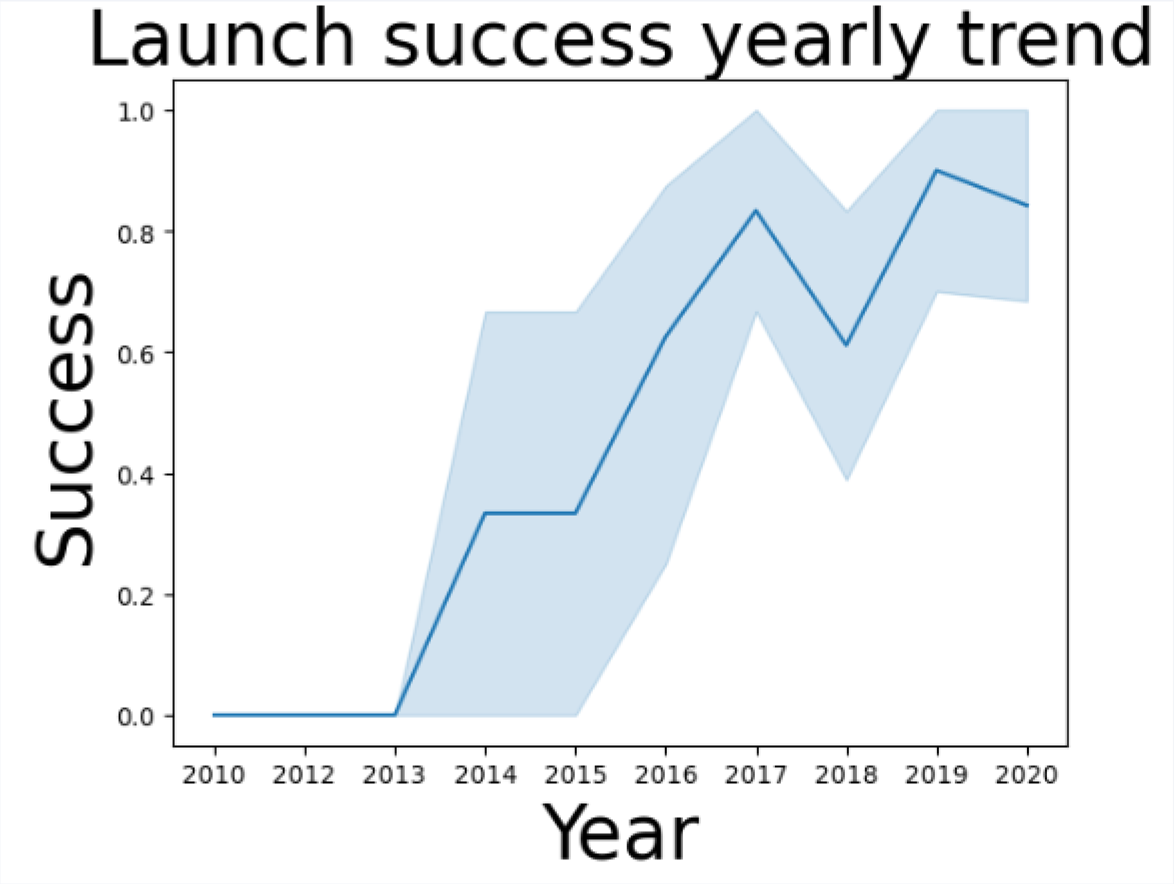
Payload vs. Orbit Type



- ISS, PO and LEO orbit types has a better results with heavy Payload mass but, in all other Orbit types, the Payload Mass does not seem to have a clear relationship.

Launch Success Yearly Trend

Since 2013 the success rate of launches have been increasing year after year due to the experience acquired (despite two short periods of decreasing).



All Launch Site Names

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

- The names of the unique launch sites were extracted from SPACEXTABLE using the following SQL query:

```
%sql select distinct Launch_Site from SPACEXTABLE
```

- The SQL "Select Distinct" statement is used to find only the different values.

Launch Site Names Begin with 'CCA'

The next 5 records contains "CCA" on the name of the launch sites. This information was easily found by a SQL query with "WHERE" statement was used. The "LIMIT" statement sets the maximum number of returns and the sign "%" indicates that we do not care about the rest of the launch sites names after "CCA".

```
%sql select * from SPACEXTABLE where Launch_Site like "CCA%" limit 5;
```

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

- We selected the column to sum and we restricted the operation to where the Customer´s name was equal to "NASA (CRS)"

Task 3

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

```
%sql select SUM (PAYLOAD_MASS__KG_) from SPACEXTABLE where Customer like "NASA (CRS)";
```

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

SUM (PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

- We selected the column to figure out the average and we restricted the operation to where the name of the Booster version was equal to "F9 v1.1"

Task 4

Display average payload mass carried by booster version 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

First Successful Ground Landing Date

- The following query allows to find the dates of the first successful landing outcome on ground pad by using the "min(Date)" statement.

```
%sql select min(Date) from SPACEXTABLE where Landing_Outcome like "Success (ground pad)";
```

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

min(Date)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- The following query return us the list of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000, by using "AND" statement to allow the two conditions to be met.

```
%sql select Booster_Version from SPACEXTABLE where Landing_Outcome = "Success (drone ship)" and PAYLOAD_MASS__KG_ between "4
```

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

%sql select Booster_Version from SPACEXTABLE where Landing_Outcome = "Success (drone ship)" and PAYLOAD_MASS__KG_ between "4000" and "6000";

Total Number of Successful and Failure Mission Outcomes

- The following query list the total number of successful and failure mission outcomes.

```
%sql select Mission_Outcome, count(*) as RESULTADOS from SPACEXTABLE group by Mission_Outcome ;
```

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

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

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 following query allow to list the names of the booster which have carried the maximum payload mass using a subquery.

```
%sql select Booster_Version from SPACEXTABLE where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTABLE)
```

%sql select Booster_Version from SPACEXTABLE where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTABLE);

2015 Launch Records

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

- Present your query result with a short explanation here

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

Landing_Outcome	RESULTADOS SOLICITADOS
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

The landing outcomes between the date 2010-06-04 and 2017-03-20, in descending order were obtained by using "between" statement to determinate the dates period and grouped thanks to "group by" statement.

Finally "Order by" and "DESC" statements allowed us to rank the results as we want.

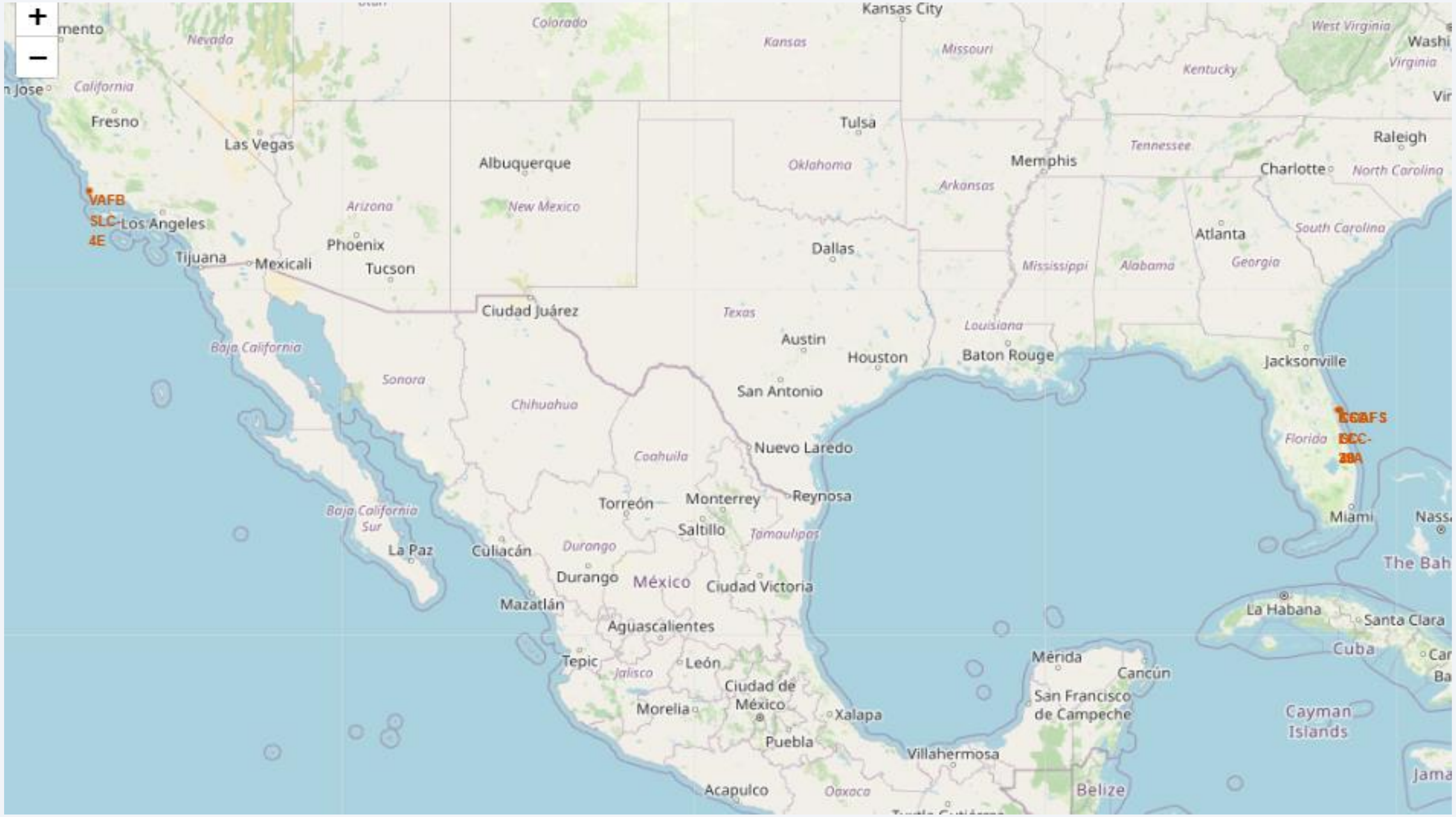
%sql select Landing_Outcome, count(*) as "RESULTADOS SOLICITADOS" from SPACEXTABLE where date between "2010-06-04" and "2017-03-20" group by Landing_Outcome order by "RESULTADOS SOLICITADOS" DESC;

```
%sql select Landing_Outcome, count(*) as "RESULTADOS SOLICITADOS" from SPACEXTABLE where date between "2010-06-04" and "2017
```

Section 3

Launch Sites Proximities Analysis

Map with the location of the Launch Sites



The map shows the different locations of the launch sites with a highlighted circle area and with a text label on the specific coordinates.

<Folium Map Screenshot 2>

- Replace <Folium map screenshot 2> title with an appropriate title

- Explore the folium map and make a proper screenshot to show the color-labeled launch outcomes on the map

- Explain the important elements and findings on the screenshot

<Folium Map Screenshot 3>

- Replace <Folium map screenshot 3> title with an appropriate title

- Explore the generated folium map and show the screenshot of a selected launch site to its proximities such as railway, highway, coastline, with distance calculated and displayed

- Explain the important elements and findings on the screenshot

Section 4

Build a Dashboard with Plotly Dash

<Dashboard Screenshot 1>

- Replace <Dashboard screenshot 1> title with an appropriate title
- Show the screenshot of launch success count for all sites, in a piechart
- Explain the important elements and findings on the screenshot

<Dashboard Screenshot 2>

- Replace <Dashboard screenshot 2> title with an appropriate title

- Show the screenshot of the piechart for the launch site with highest launch success ratio

- Explain the important elements and findings on the screenshot

<Dashboard Screenshot 3>

- Replace <Dashboard screenshot 3> title with an appropriate title

- Show screenshots of Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider

- Explain the important elements and findings on the screenshot, such as which payload range or booster version have the largest success rate, etc.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

Find the method performs best:

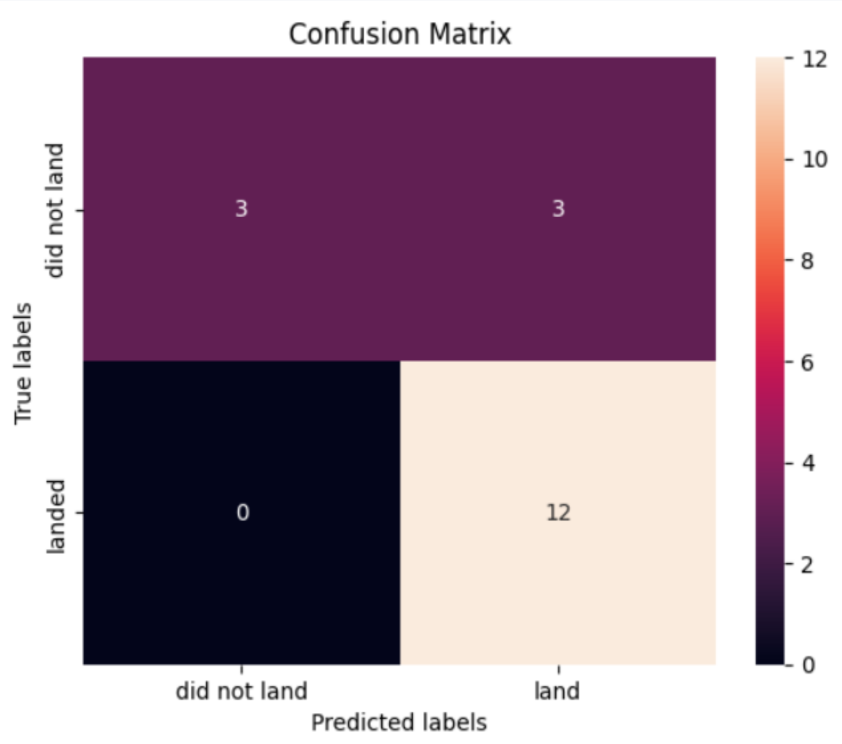
```
print(f"\nLogistic Regression Test accuracy: {test_score:.2%}")
print(f"\nSVM Test accuracy: {test_score2:.2%}")
print(f"\nDecision Tree Test accuracy: {test_score3:.2%}")
print(f"\nK Nearest Neighbors Test accuracy: {test_score4:.2%}")
```

```
Logistic Regression Test accuracy: 83.33%

SVM Test accuracy: 83.33%

Decision Tree Test accuracy: 66.67%

K Nearest Neighbors Test accuracy: 77.78%
```

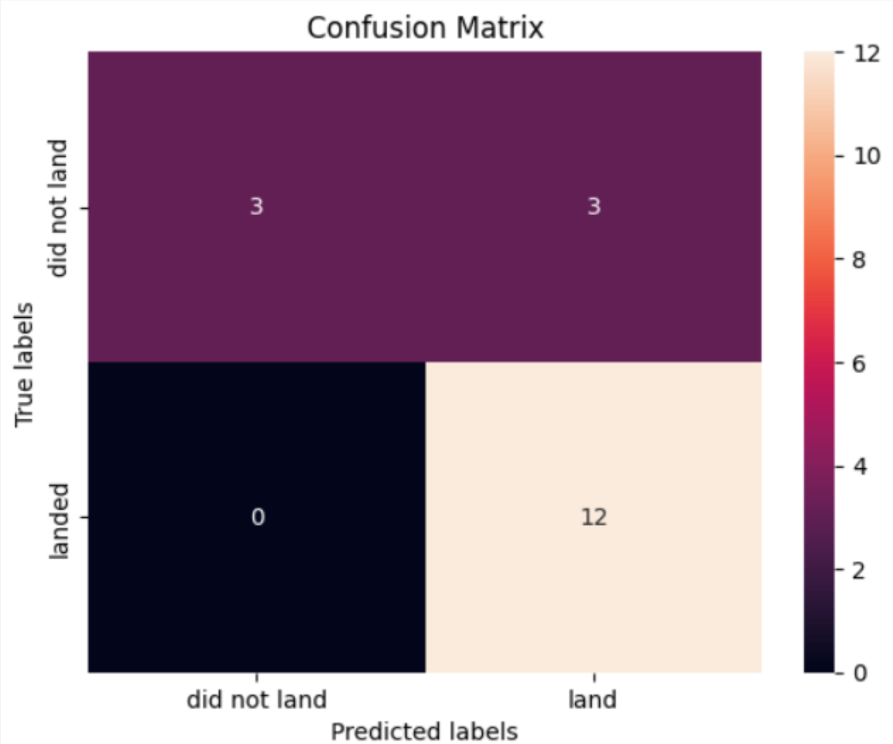
Based on the results of the accuracy tests, both Logistic Regression and SVM models achieve around to 83%.

The worst accuracy result was obtained by the Decision tree model.

Confusion Matrix



Both Logistic Regression and SVM models achieve around to 83% of accuracy and their confusion matrix shows the same results related to false/true positive and negative.



Conclusions

- The online information available about launches is sufficient to allow for an in-depth analysis.
- The data allows for evaluating the evolution of the success of launches both over time and at different Launch Sites.
- The question of whether the first stage of Falcon 9 rockets will land successfully can be answered and supported by consistent data analysis.
- The most accuracy model predicts a success rate of 83,33%

Appendix

SOURCES OF INFORMATION

- SpaceX Open Source Rest API : https://api.spacexdata.com/v4/rockets/
- From Wikipedia through a web scraping process. https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches

Thank you!