IBM Developer
SKILLS NETWORK

IBM Data Science Professional Certificate

Applied Data Science Capstone Project: SpaceX Falcon-9 Rocket Performance Analysis

by Sofia Guerin

Image from: https://www.discovermagazine.com/

Outline

01 Executive Summary

The “SpaceX Falcon-9 Rocket Performance Analysis” project analyzed real-world data from the company SpaceX in about rocket launching and landing, with the objective of creating a comprehensive study in which can help future space companies to optimize their rocket performance. In this project, we did **data collection**, **wrangling** and **formatting** using data from SpaceX API. This data was then analyzed via basic **EDA** and further **interactive data visualization** in order to obtain more detailed information. Statistical analysis was finally performed via **Predictive Analysis**.

Our **results** show that, with the data provided, it is possible to statistically predict with high certainty and accuracy **launching** as well as **landing probabilities**, of which the **success rate** will mainly depend on the **first stage** used and their **payloads**.

02 Introduction

In this capstone, we were able to predict the **landing probability of the Falcon 9 first stage**. SpaceX advertises the launching of Falcon 9 rocket at a cost of 62 million dollars. In contrast to other institutions and companies that spend over $160m in the same process, SpaceX is able to land and further re-use the first stage of the rocket, an advantage that allows the company to reduce the costs by more than 50%.

Being able of determining the landing probabilities of a first stage would help new companies to assess the cost of their rockets and in turn compete or bit against SpaceX for a rocket launch.

03 Methodology

Summary

1/ Data collection

- **.get()** requests to SpaceX REST API
- Web scraping

2/ Data Wrangling

- **.fillna()** for removing non-numeric values
- **.value_counts()** was used to determine the number of:
  - Launches on each site
  - Occurrence of each orbit per rocket type
  - Occurrence of mission outcome per orbit type
- Creation of a landing outcome label with Boolean values using **for** and **if** clauses.

3/ Exploratory data analysis (EDA)

- **SQL queries** were used to retrieve data sets from SpaceX databases and Wikipedia as well as for obtaining basic information about the datasets, such as the name of rocket launch Sites.
- Data was visualized using **Pandas** and **Matplotlib** to determine relationship between variables and trends over time.

4/ Interactive Visual Analytics

- **Folium** was used to create a map in which to display launching sites with markers and pop-up comments.
- **Dash** was used to create an interactive dashboard for representing launching success and payload vs. launching success.

5/ Data modelling & evaluation

- Scikit-Learn has been used to preprocess ans standardize the data as well as for splitting the data into **training** and **testing sets**.
- **GridSearchCV** was used to fir the model to the data and to find the best parameters for each model.
- **Confusion matrix** was used to visualize each model`s **classification accuracy**.
- **F1 score** as well as accuracy were used to select the best fitting model.

03 Methodology

Data collection – SpaceX API

We used **SpaceX API** to retrieve their data about rocket launching and landing, with detailed information about the rocket model used, the payload carried, the launching and landing coordinates as well as the landing success.

1/ Request and parse SpaceX launch data using a **.get()** request and then convert the response **.json file** into **Pandas DataFrame**.

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

```
response = requests.get(static_json_url)
data = pd.json_normalize(response.json())
```

2/ Format the IDs given in the original dataset from SpaceX into our working data set using **GET requests**. We first define **empty lists** that we fill up as a dictionary with our data of interest (**Booster name**, **Payload Mass**, **Launchpad coordinates** and **cores** - landing success and coordinates).

```
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

```
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}
```

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

3/ Next, we transform this library into a **dataframe** using **Pandas** and we filter it to include only information about **Falcon 9 rocket**:

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

```
data_falcon9 = df[df['BoosterVersion'] == 'Falcon 9']
data_falcon9.head()
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	Reus
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	
5	8	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	
6	10	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	

4/ Finally, we remove all non-numeric values (**NaN**) and replace them with the mean (**.mean()**) in that category using **.replace()** and **numpy.nan**:

```
data_falcon9["PayloadMass"] = data_falcon9['PayloadMass'].replace(np.nan, mean)
```

```
data_falcon9.isnull().sum()
```

```
FlightNumber      0
Date              0
BoosterVersion    0
PayloadMass       5
Orbit             0
LaunchSite        0
Outcome           0
```

```
FlightNumber      0
Date              0
BoosterVersion    0
PayloadMass       0
Orbit             0
LaunchSite        0
Outcome           0
```

GitHub Link

03 Methodology

Data collection - Web Scraping

Here, we performed Web scraping to collect Falcon 9 historical launch records form the Wikipedia page titled "List of Falcon 9 and Falcon Heavy launches" (https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922)

1/ Request the Falcon 9 Launch Wikipedia page from its **URL** and create a **BeautifulSoup object** from the **HTML response** (in text format).

```
html_data = requests.get(static_url).text
```

```
soup = BeautifulSoup(html_data, 'html.parser')
```

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

```
column_names = []
for i in first_launch_table.find_all('th'):
    col_name = extract_column_from_header(i)
    if col_name != None and len(col_name)>0:
        column_names= column_names.append(col_name)
```

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

3/ Create a dataframe by parsing the launch HTML tables and creating the launch **dictionary** 'launch_dict'

```
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']=[]
```

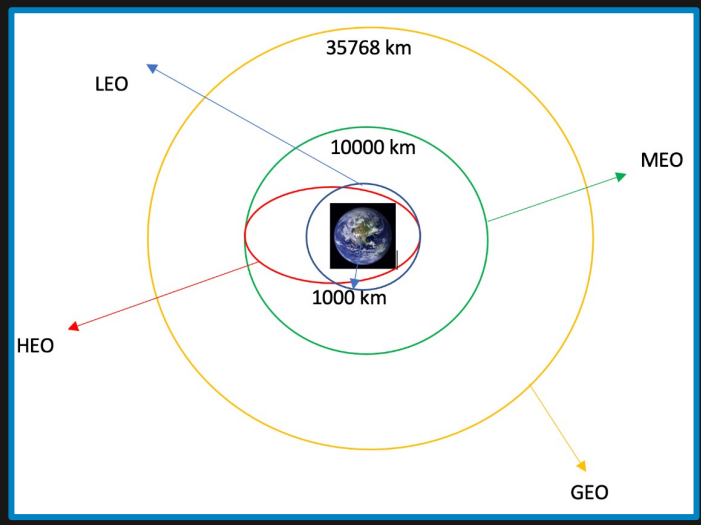
```
df=pd.DataFrame(launch_dict)
```

GitHub Link

03 Methodology

Exploratory Data Analysis - Data Wrangling Part 01

The main objective of wrangling the data was to produce a series of labels indicating the landing success of the rocket booster: 0 for unsuccessful landings and 1 for successful ones.



1/ Import the dataframe from URL address using **Pandas .read_csv()** fucntion

```
df=pd.read_csv(dataset_part_1_csv)
df.head(10)
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857

2/ Calculate the numberof launches on each launching site and the number of occurrence of each orbit (how many times each orbit has been used for a mission)using **.value_counts()**

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

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

```
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 the number of occurrence of mission outcome per orbit type (meaning, the number of successful and unsuccessful landings of boosters coming from the different orbits).

```
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
```

GitHub Link

03 Methodology

Exploratory Data Analysis - Data Wrangling Part 02

The main objective of wrangling the data was to produce a series of labels indicating the landing success of the rocket booster: 0 for unsuccessful landings and 1 for successful ones.

4/ The we identify the different labels that SpaceX assigned to the landing of each mission and separate a list with only unsuccessful ones:

```
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
```

True = successful landing
False = unsuccessful landing

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

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

5/ Next, we wrote a function to iterate through the 'Outcome' column of our dataframe: if the outcome of the mission was found in the 'bad_outcomes' list, then we appended a '0' to out new list 'landing_class', else, a '1':

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

```
['0', '0', '0', '0', '0', '0', '1', '1', '0', '0', '0', '0', '1', '0', '0', '0', '1', '0', '0', '1', '1', '1', '1', '1', '1', '0', '1', '1', '0', '1', '1', '0', '1', '1', '1',
'1', '1', '1', '1', '1', '1', '1', '1', '0', '0', '0', '1', '1', '0', '0', '1', '1', '1', '1', '1', '1', '1', '1', '0', '0', '1', '1', '1', '1', '1', '1', '1', '0', '1', '1', '1',
'1', '0', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1']
```

6/ Finaly, we transformed the 'landing_class' list into a new column named 'Class' in our dataframe:

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

```
df.head(5)
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857	0
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857	0
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857	0
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632093	0
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857	0

GitHub Link

03 Methodology

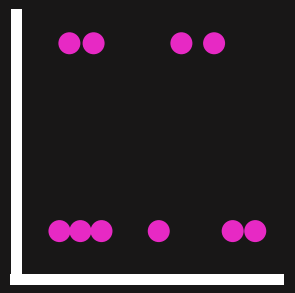
Exploratory Data Analysis - Visualization

SCATTER PLOTS

Scatter plots are useful tools to observe or represent relationships or correlations between two numeric variables.

Scatter plots were used to visualize the relationship between:

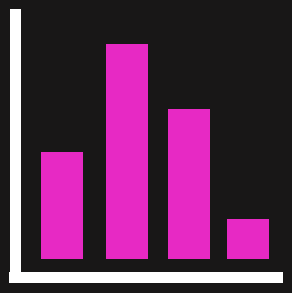
- Flight number and Launch Site
- Payload carried and Launch Site
- Payload and Orbit
- Orbit Type and Flight Number



BAR CHARTS

Bar chart are used to represent categorical variables that correlate to numeric values. Depending on the nature of the data, these charts can be displayed horizontally or vertically.

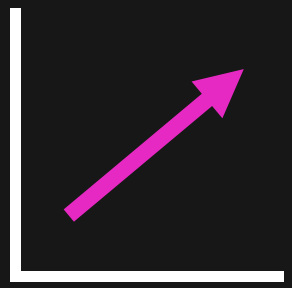
Here, we used bar charts to visualize the relationship between the landing success rate of the booster versus the orbit chosen for the mission.



LINE CHARTS

Line charts are used to represent two numerical variables against each other. They are normally used to visualize the change of a dependent variable over time or over an independent variable.

In this project we used line charts to visualize they early increase in Landing success rate of Boosters.



GitHub Link

03 Methodology

Exploratory Data Analysis - SQL

SQL queries were used to...

- Display the names of unique launch sites in the space mission
- Display five entries where the launch site name begins with the string 'CCA'
- Display the total Payload Mass carried by boosters launched by NASA (CRS)
- Display the average payload carried by boosters of the 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 4k but less than 6k
- List the total number of successful and failure mission outcomes
- Using a subquery, list the names of the booster versions which have carried the maximum payload mass
- List the records containing month names, failure landing outcomes in drone ship ,booster versions and launch site for the months in the 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 Link

03 Methodology

Interactive Visual Analytics with Folium

Launch site locations were analyzes using Folium as follows:

01/ Mark all launch sites on the map:

- **folium.Map()** function was used to initialize a map element
- **folium.Circle()** and **folium.Marker()** were used to point the launch sites locations on the map
- Circle and marker elements were added to the map using the **.add_child()** function

02/ Mark the success/failed launches for each site on the map:

- Given that many launches have the same coordinates, we first created a **MarkerCluster() object**
- Creation of a ‚**marker_color**‘ list which stored (usinf a **for loop**) the color of each marker (green for successful and red for not-successful launchings)
- For each launch, we created a **folium.Marker** that was then added to the MarkerCluster object
- Finally, we created an **icon** for each launch as a text label, of which color (**icon_color**) corresopnded to the **marker_color** list

03/ Calculate the distance between a launch site and elements of ist proximity

- To explore and analyze the proximities of a launch site, we looked at the distance with elements of ist surroundings.
- To acquire te coordinates (**Lat** and **Long** values) of a point of interes on the map we used the function **MousePosition()** and added it to the map usin **add.child()** function
- To calculate the distance between the launch site and the point of interest we used **folium.PolyLine()** function and again add it to the map with **add.Child()** function

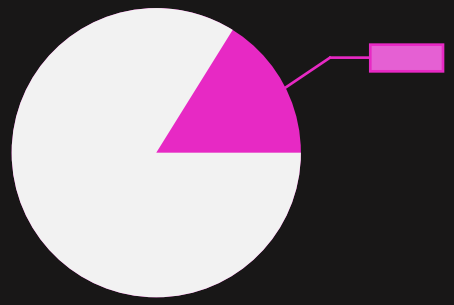
GitHub Link

03 Methodology

Dashboard Application with Plotly Dash

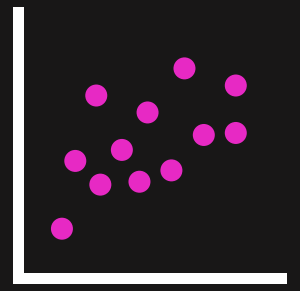
PIE CHART

- A Pie chart was used to display the number of successful vs unsuccessful launches per launching site, as well as the total number of successful vs unsuccessful launches among all launching sites.
- To select between launching sites we added a dropdown menue (**dcc.Dropdown()**)where the used could selec „All sites“ or single launching sites.
- **px.pie()** function was used to produce the pie chart.



SCATTER PLOT

- A scatter plot was used to show the correlation between the mission outcome (value = 1 marked successful launching while 0 marked unsuccessful launching).
- The scatter plot data could be filteres via the launch site (selected in the dropdown menue) and via payload mass (which could be selected thanks to the addition of an interactive range slider (**RangeSlider()**) object.
- **px.scatter()** function was used to produce the scatter plot.



GitHub Link

03 Methodology

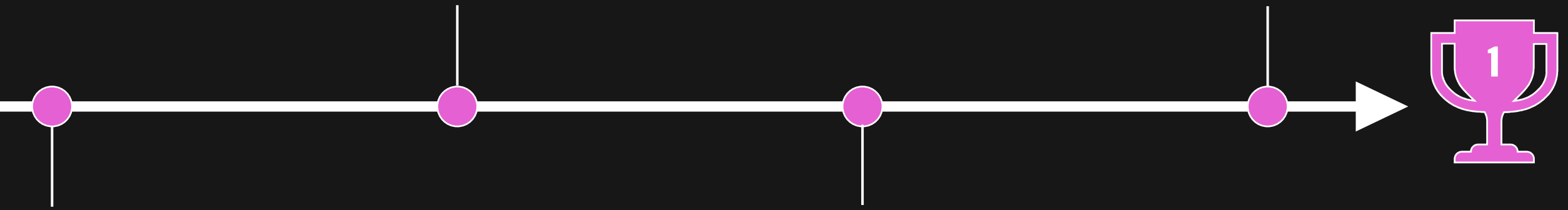
Predictive Analysis (Classification)

IMPROVEMENT

Model improvement was carried out by creating a **GridSearchCV** object to calculate the model **best parameters** (**best_params_**) and best score (**best_score_**) using the training data.

SELECTION

The best model was selected based on its **accuracy** and the **weighted f1 score**.



DEVELOPMENT

Before defining any predictive model:

1. Load the dataset followed by basic standardization and pre-processing.
2. split the data into training and test data using **train_test_split()** function.

We developed four different prediction models: **Logistic Regression**, **support vector machine** or **SVM**, **Decision Tree** and **K nearest neighbor** models.

EVALUATION

The **predictive accuracy** of the model was then assessed via the method **score** and visualized on a **confusion matrix** using the test data.

GitHub Link

GitHub Link

https://github.com/guerin2023/IBM-Applied-Data-Science-Capstone

Results

Exploratory Data Analysis – EDA

PART – 01.1

EDA with Visualization

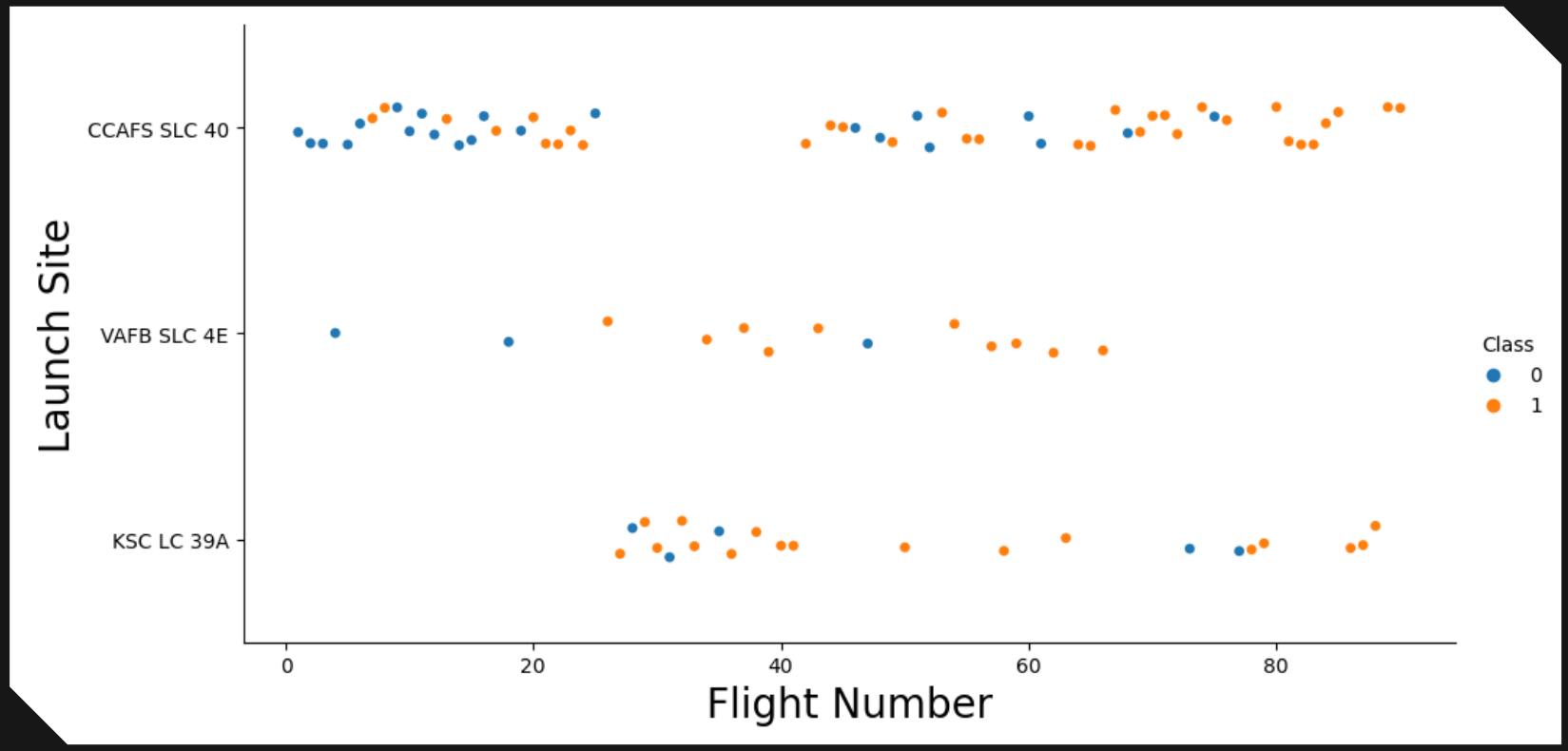
04 Results

EDA with visualization (Seaborn and MatPlotLib)

Flight number vs. Launch site

From the scatter plot representing Launch Site vs. Flight number, we can see that:

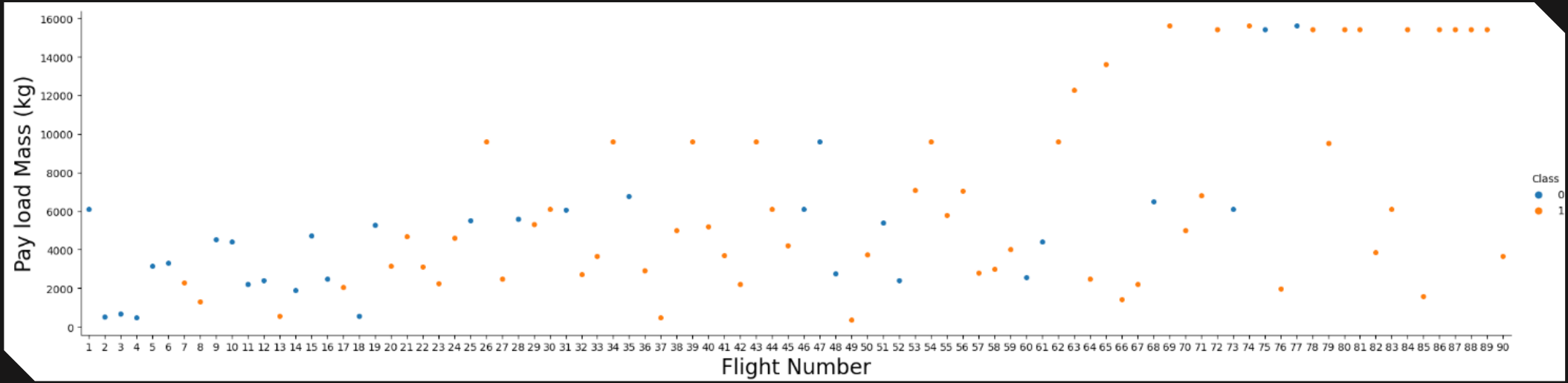
- Most of SpaceX launches have been performed at either **CCAFS SLC-40** or **KSC LC 39A** launching sites.
- CCAFS SLC-40 was the first launching site to be built/used by SpaceX with apparently the highest number of failed launch attempts.
- As the flight number increases, we can see that the launching success increases.



04 Results

EDA with visualization (Seaborn and MatPlotLib)

Flight number vs. Payload mass



From the scatter plot representing Flight number vs. Payload mass ,we can see that:

- From flight number 1 to around flight number 60, the payload carried by the rockets remained mostly below 8000kg.
- From flight number 60 on, some rockets started carrying higher payloads, up to 16000kg, which is double of the maximum they were carried before.

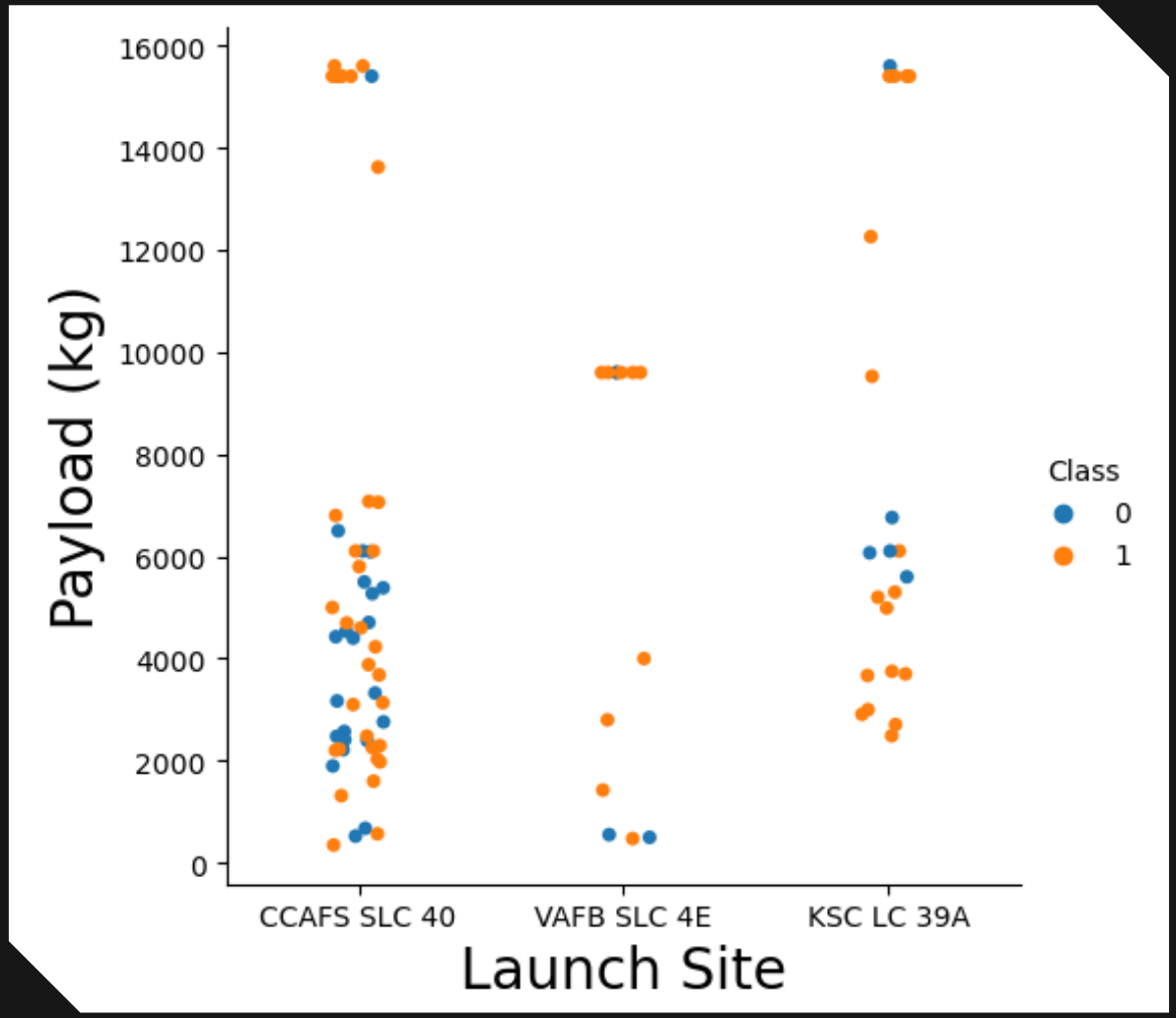
04 Results

EDA with visualization (Seaborn and MatPlotLib)

Payload mass vs. Launch site

From the scatter plot representing the Payload carried in kg vs. the launching site, we can see that:

- Again, most Falcon 9 rocket launches happened in the launching sites CCAFS SLC-40 and KSC LC-39A.
- Rockets launched from these two sites were also those carrying the highest payloads, of up to almost 16000kg.
- We can also see here (unlike in the previous plot) that rockets carrying payloads of 10000kg were almost exclusively launched from site VAFB 4E.

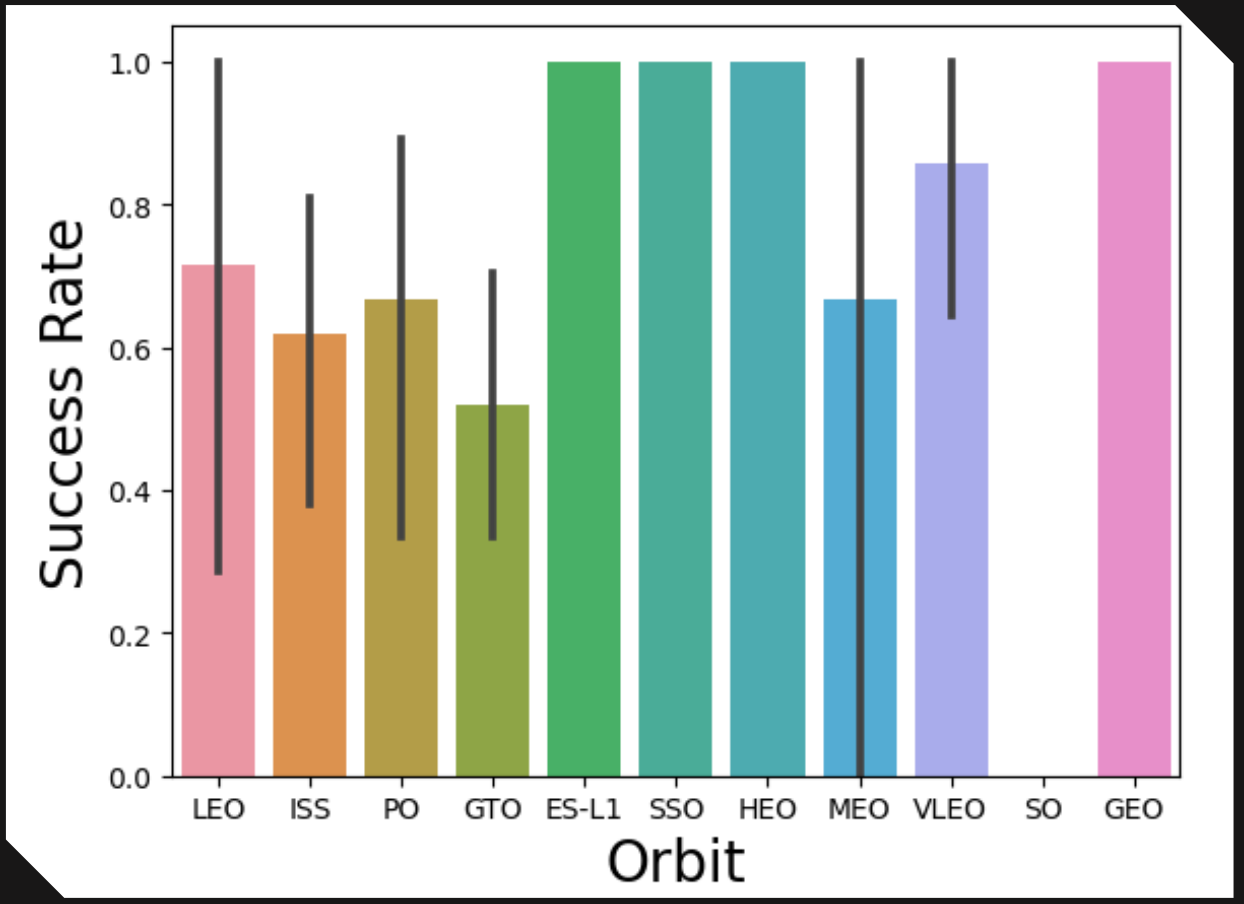


04 Results EDA with visualization (Seaborn and MatPlotLib)

Landing success rate vs. Mission's Orbit

From the scatter plot representing the rocket landing success and the orbit taken for that mission, we can see that:

- If we exclude orbits that have been used only once o far in missions (**GEO, SO, HEO** and **ES-L1**, see **Appendix 02**), the **orbit SSO** has the highest success rate of all orbits, with a total number of 5 flights (see **Appendix 02**) and a landing success rate of 100%.
- SSO orbit is followed by **orbit VLEO**, with a total number of 14 times orbited (see **Appendix 02**) and a landing success rate of over 80%.
- Orbit SSO, on the other hand, has a success
- On the other hand, **orbit GTO** has been orbited in 27 different missions (see **Appendix 02**) , and its landing success rate lays at around 50%.
- Finally, we see that there is no bar value for the orbit SO. This is due to the fact that **orbit SO** has been only orbited once (see **Appendix 02**) and this mission ended with an unsuccessful landing (this means, class = '0').

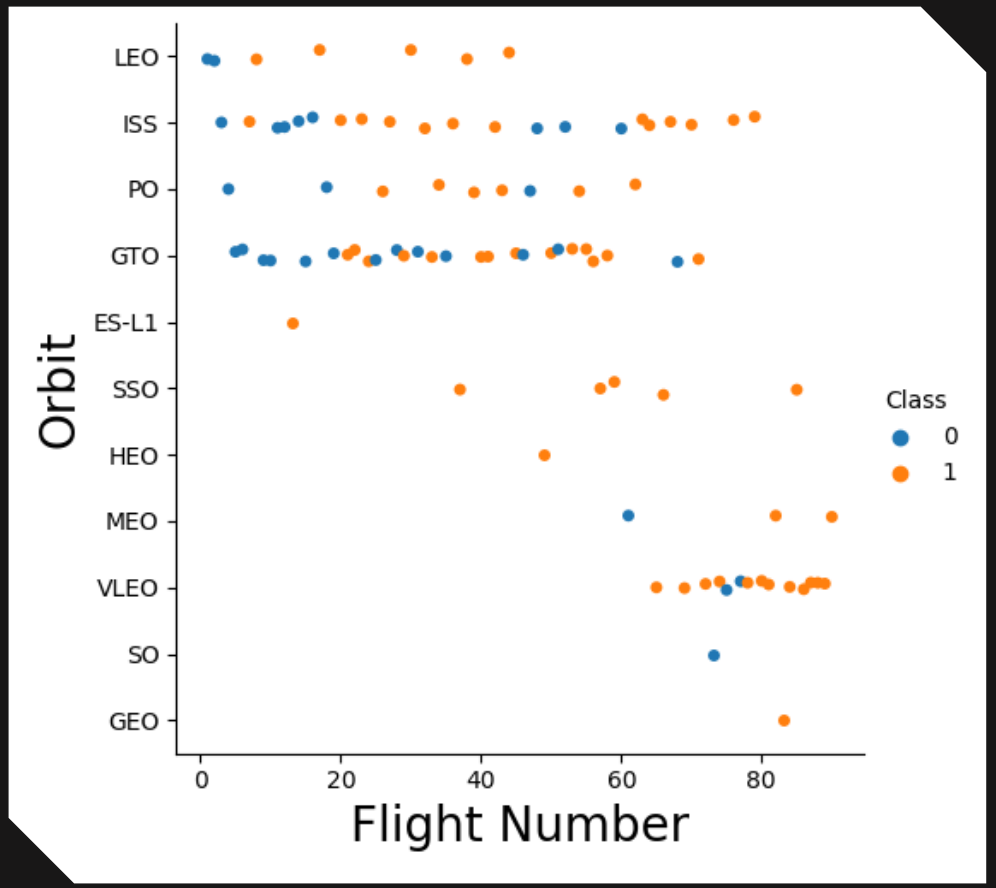


04 Results EDA with visualization (Seaborn and MatPlotLib)

Mission's Orbit vs. Flight number

From the scatter plot representing the Mission`s orbit (or Orbit type) vs. the flight number, we can see the following:

- Orbits **LEO, ISS, PO, GTO** and **VLEO** are the most used orbits for missions.
- Since the flight number increases with time (meaning that flights with higher numbers happened after flights with smaller numbers (see **Appendix 04**)) we can see that **orbits ISS** and **VLEO** are preferred in the most recent missions (years 2019 and 2020).

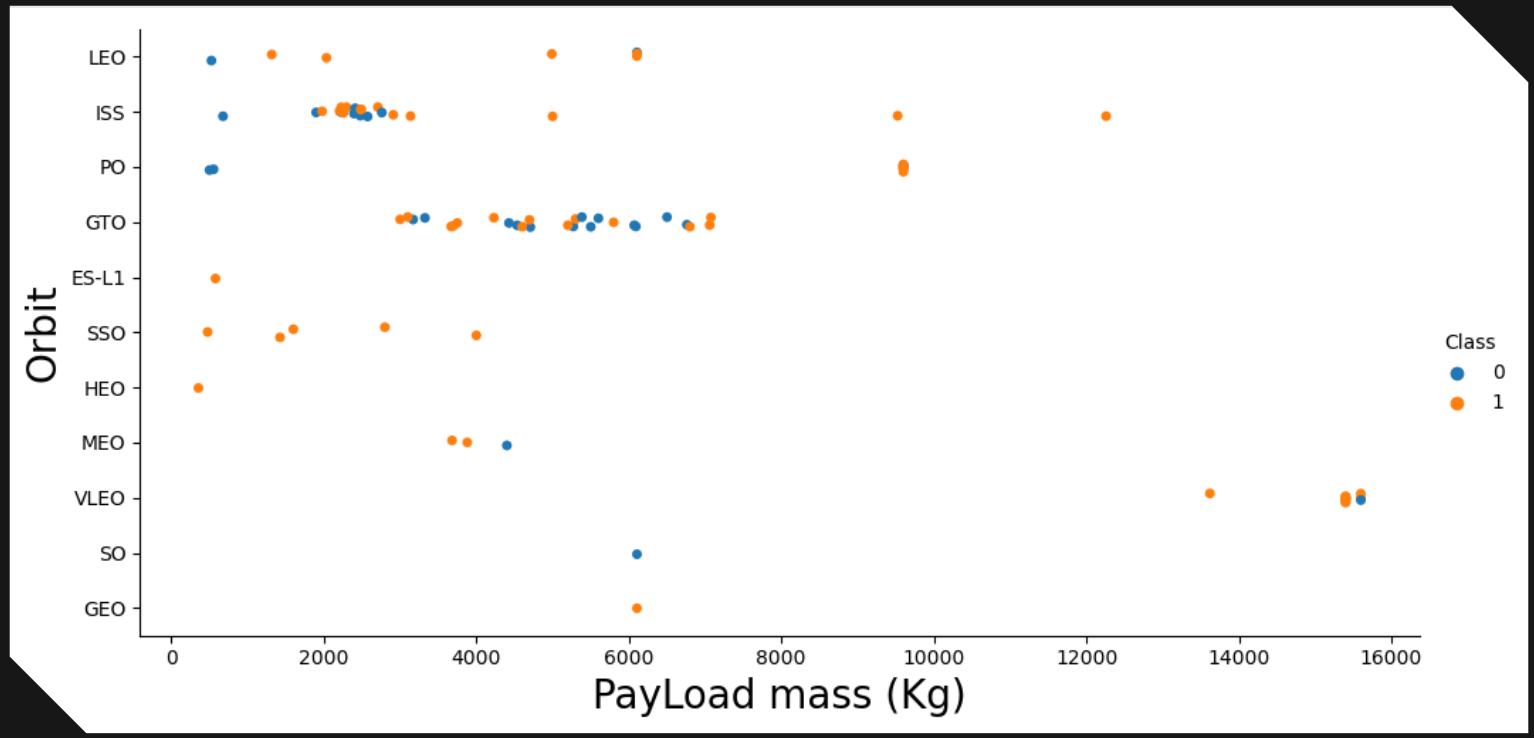


04 Results EDA with visualization (Seaborn and MatPlotLib)

Mission's Orbit vs. Payload mass

From the scatter plot representing the Mission`s orbit (or Orbit type) vs. payload mass, we can see that:

- There seems to be no direct correlation between the mass carried by the rocket and the landing success.
- We can see that for some orbits, different but very close weights have been used: for orbit ISS we can see that most of the missions carried a payload between 2000 and 3200kg, while missions in orbit GTO carried payloads mostly in the range between 3000 and 7000kg.
- In these two examples we can see that there seems to be no correlation between the payload carried and the success of the landing.



04 Results

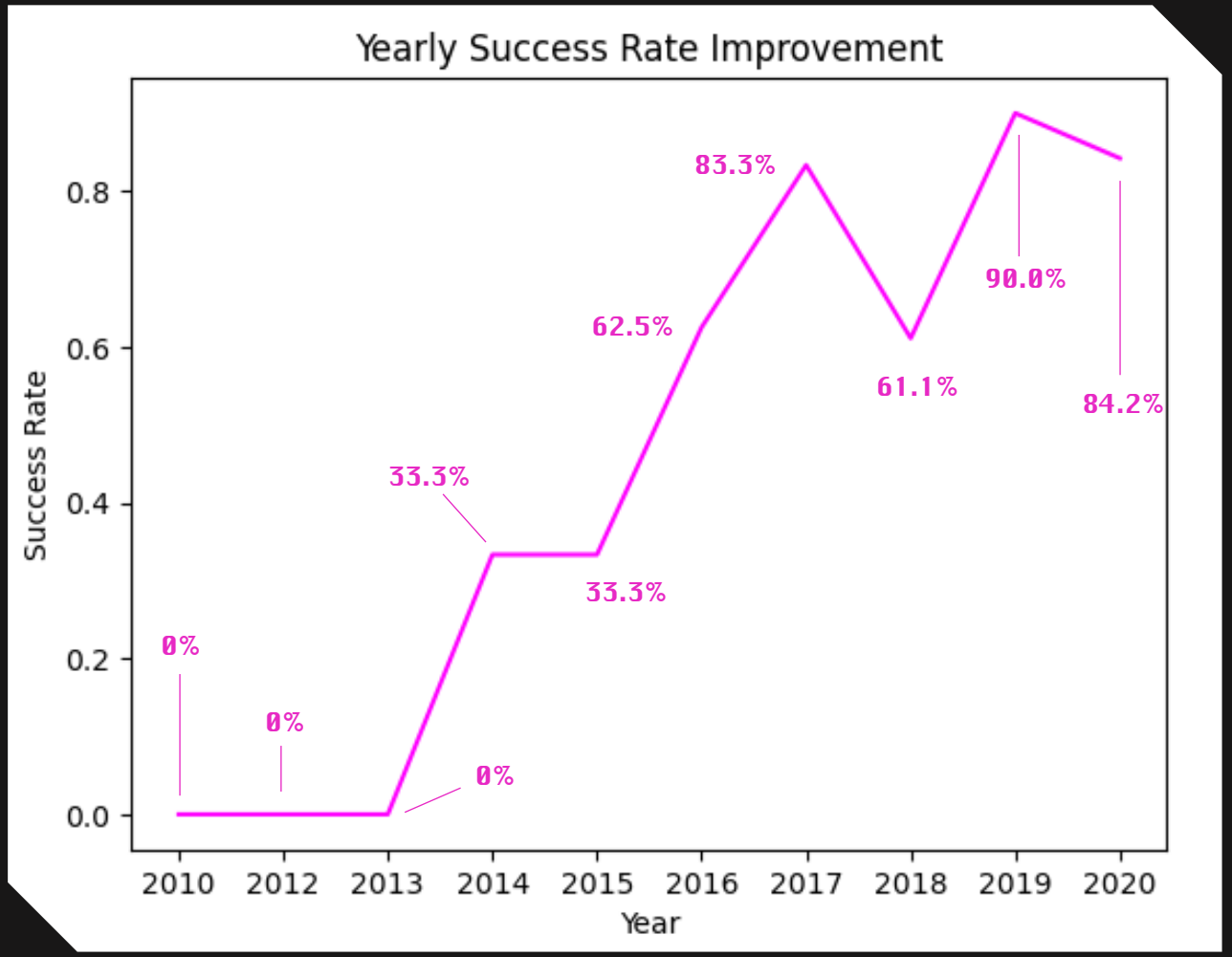
EDA with visualization (Seaborn and MatPlotLib)

Launch Success Yearly Trend

Here, we can see that the missions´ landing success has been increasing almost steadily since 2013, except between the years 2017 and 2018 (in which there was a decrease of around 20%), and between years 2019 and 2020 (where there was a decrease of almost 6%).

As we can see in **Appendix 04**, the the landing success increased to 33.3% in the years 2014 and 2015 and doubled in 2016. Between 2016 and 2017 there was a great increase of over 20%.

2019 was the most successful year in terms of landed missions, with 10 missions in total (see **Appendix 05**). However, in this year there was also a reduction in the number of missions: 2017 and 2018 had 18 missions each, and 2020 finalized with a total of 19 missions performed (see **Appendix 05**).



Results

Exploratory Data Analysis – EDA

PART – 01.2

EDA with SQL

04 Results EDA with SQL

Unique Launch Sites

In SQL language, we can find the unique components of a list/column by using the command **distinct()**.

Here, we used distinct() to find the names of the different launch sites.

```
%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

04 Results EDA with SQL

Five records where launch site begins with string ‘CCA’

```
%sql SELECT * from SPACEXTABLE WHERE Launch_Site LIKE 'CCA%';
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-08-12	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
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

In SQL language, we can use the function **WHERE ‘{col_name}’ LIKE ‘{str}’** to filter a data frame`s entries according to a specific string present in one of the columns. The string might be part of the name only.

Here, we used the function LIKE with the string ‘CCA’ to filter out all those entries in which the name of the launch site contained the letters/string CCA.

04 Results EDA with SQL

Total Payload mass carried by NASA Boosters

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) from SPACEXTABLE WHERE Customer == 'NASA (CRS)';
 * sqlite:///my_data1.db
Done.
```

SUM(PAYLOAD_MASS__KG_)
45596

In SQL language, we can use the function **SUM()** to compute the sum of all values of a column in a data frame, provided that the column values are of type float or int.

Additionally, the function **WHER {col_name} == '{str}'** to filter a data frame`s entries according to a specific string present in one of the columns. Here, the string provided to the function must coincide 100% with the string that is to be found in the column.

Here, we used these functions to compute the total payload that has been carried by NASA (CSR) rocket boosters..

04 Results EDA with SQL

Average Payload mass carried by Booster version F9 v1.1

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

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

AVG(PAYLOAD_MASS__KG_)
2928.4

In SQL language, we can use the function **AVG()** to compute the average of all values of a column in a data frame, provided that the column values are of type float or int.

Additionally, the function **WHER {col_name} == '{str}'** to filter a data frame`s entries according to a specific string present in one of the columns. Here, the string provided to the function must coincide 100% with the string that is to be found in the column.

Here, we used these functions to compute the average payload that has been carried by rocket boosters of the version F9 v1.1.

04 Results EDA with SQL

Date of first successful ground landing

```
%sql SELECT min(Date) from SPACEXTABLE WHERE Landing_Outcome == 'Success (ground pad)';
```

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

min(Date)
2015-12-22

```
%sql SELECT * from SPACEXTABLE WHERE Landing_Outcome == 'Success (ground pad)' ORDER BY Date limit 1;
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2015-12-22	01:29:00	F9 FT B1019	CCAFS LC-40	OG2 Mission 2 11 Orbcomm-OG2 satellites	2034	LEO	Orbcomm	Success	Success (ground pad)

In SQL language, we can use the function **MIN()** to compute the minimum of all values of a column in a data frame, provided that the column values are of type float, int or date.

Additionally, the function **WHER {col_name} == '{str}'** to filter a data frame`s entries according to a specific string present in one of the columns. Here, the string provided to the function must coincide 100% with the string that is to be found in the column.

Here, we used these functions to compute the earliest date in which there was a successful landing of a booster.

04 Results EDA with SQL

Successful Drone Ship Landing with Payloads between 4k and 6k

```
%sql SELECT DISTINCT(Booster_Version) from SPACEXTABLE WHERE Mission_Outcome like 'Success%' AND PAYLOAD_MASS__KG_ between 4000 and 6000;
```

Booster_Version
F9 v1.1
F9 v1.1 B1011
F9 v1.1 B1014
F9 v1.1 B1016
F9 FT B1020
F9 FT B1022
F9 FT B1026
F9 FT B1030
F9 FT B1021.2
F9 FT B1032.1
F9 B4 B1040.1
F9 FT B1031.2
F9 B4 B1043.1
F9 FT B1032.2
F9 B4 B1040.2
F9 B5 B1046.2
F9 B5 B1047.2
F9 B5 B1046.3
F9 B5B1054
F9 B5 B1048.3
F9 B5 B1051.2
F9 B5B1060.1
F9 B5 B1058.2
F9 B5B1062.1

In SQL language, we can find the unique components of a list/column by using the command **distinct()**.

Additionally, the function **WHER {col_name} like '{str%}'** to filter a data frame`s entries according to a specific string present in one of the columns. Here, the position of the % symbol indicates where the rest of the name coudl be located. In this case, the name of the entry in the 'Mission_Outcome' column must start with the string 'Success'.

Additionally, we can use the command **{'col_name'} BETWEEN {float/int} AND {float/int}** to filter all those entries in which the values of a specific column are between the values that we feed to the function (of type float or int).

Here, we used these functions to compute all booster versions that have successfully landed carrying a payload between 4000 and 6000 kg.

04 Results EDA with SQL

Total number of successful vs. failed mission outcomes

In SQL language, we can find the unique components of a list/column by using the command **distinct()**.

The function **COUNT()** helps us compute the total number of entries of a data frame.

Additionally, we can use the command **GROUP BY** to group our results according to the unique values of a specified column.

Here, we used these functions to compute the number of successful and unsuccessful landings.

```
%sql SELECT Mission_Outcome, COUNT(*) from SPACEXTABLE GROUP BY Mission_Outcome;
```

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

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

04 Results EDA with SQL

Booster carrying maximum payload (kg)

```
%sql SELECT DISTINCT(Booster_Version), PAYLOAD_MASS__KG_ from SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (select PAYLOAD_MASS__KG_ from SPACEXTABLE order by PAYLOAD_MASS__KG_ desc);
```

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

In SQL language, we can find the unique components of a list/column by using the command **distinct()**. The function **COUNT()** helps us compute the total number of entries of a data frame.

Additionally, we can use the command **ORDER BY** to group our results according to the values of a specified column. By default, the function will order the rows in ascending order, but we can change this by specifying '**desc**', for descending order.

Finally, in SQL language we can use **subqueries** to compute values that could not be computed in the main query due to function logic/grammar.

Here, we used these functions to select all booster versions that carried the maximum payload. The value of the maximum payload was calculated with a subquery.

04 Results EDA with SQL

2015 Launch Records

```
%sql SELECT * from SPACEXTABLE WHERE substr(Date, 0, 5) =='2015';
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2015-10-01	09:47:00	F9 v1.1 B1012	CCAFS LC-40	SpaceX CRS-5	2395	LEO (ISS)	NASA (CRS)	Success	Failure (drone ship)
2015-11-02	23:03:00	F9 v1.1 B1013	CCAFS LC-40	DSCOVR	570	HEO	U.S. Air Force NASA NOAA	Success	Controlled (ocean)
2015-02-03	03:50:00	F9 v1.1 B1014	CCAFS LC-40	ABS-3A Eutelsat 115 West B	4159	GTO	ABS Eutelsat	Success	No attempt
2015-04-14	20:10:00	F9 v1.1 B1015	CCAFS LC-40	SpaceX CRS-6	1898	LEO (ISS)	NASA (CRS)	Success	Failure (drone ship)
2015-04-27	23:03:00	F9 v1.1 B1016	CCAFS LC-40	Turkmen 52 / MonacoSAT	4707	GTO	Turkmenistan National Space Agency	Success	No attempt
2015-06-28	14:21:00	F9 v1.1 B1018	CCAFS LC-40	SpaceX CRS-7	1952	LEO (ISS)	NASA (CRS)	Failure (in flight)	Precluded (drone ship)
2015-12-22	01:29:00	F9 FT B1019	CCAFS LC-40	OG2 Mission 2 11 Orbcomm-OG2 satellites	2034	LEO	Orbcomm	Success	Success (ground pad)

In order to obtain all entries corresponding to the year 2015, we selected, with functions **WHERE** and **substr()**, those entries where the Date value had the string '2015' between the string position 0 and 4

04 Results EDA with SQL

Landing Outcomes between 2010-06-04 and 2017-03-20 In descending order

```
%%sql
SELECT LANDING_OUTCOME, count(LANDING_OUTCOME) as total_number from SPACEXTABLE
where CAST(substr(Date, 0, 5) as INTEGER) between 2010 and 2017
GROUP BY LANDING_OUTCOME ORDER BY Total_number desc;
```

Landing_Outcome	total_number
Success (drone ship)	12
No attempt	12
Success (ground pad)	8
Failure (drone ship)	5
Controlled (ocean)	4
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

In order to obtain the total number of each type of mission landing outcome between the years 2010 and 2017, we used the functions **COUNT()**, to count the total number of entries, **WHERE ... between {date1} and {date2}** to determine the condition that the entries` date should be between 2010 and 2017, the function **GROUP BY** to group the counts by the landing outcome description, and the function **ORDER BY**...desc to order the result entries by the total number in descending order.

Results

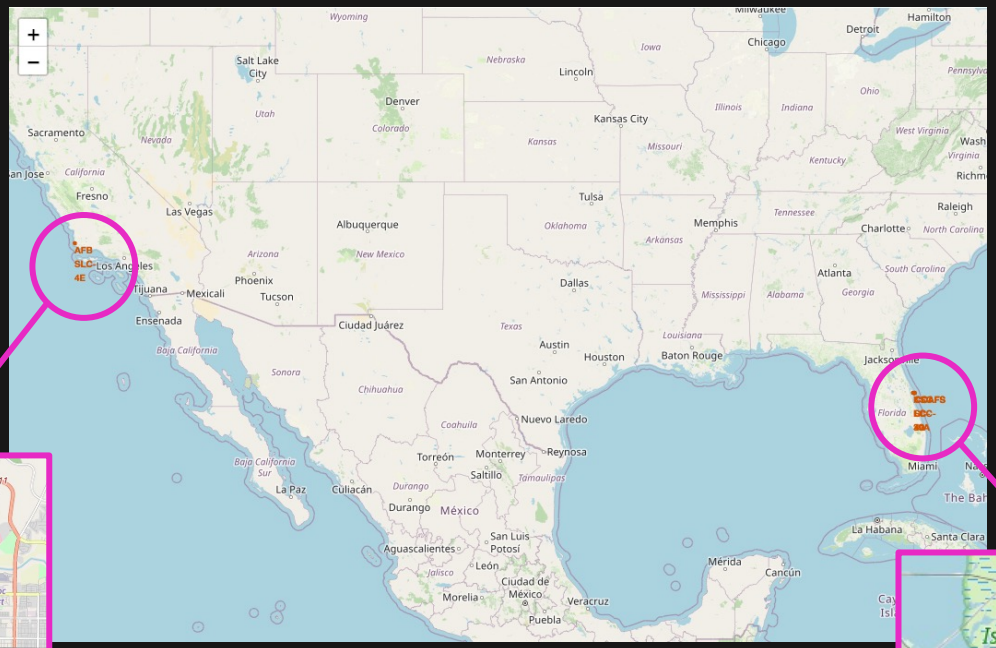
Interactive Visual Analytics

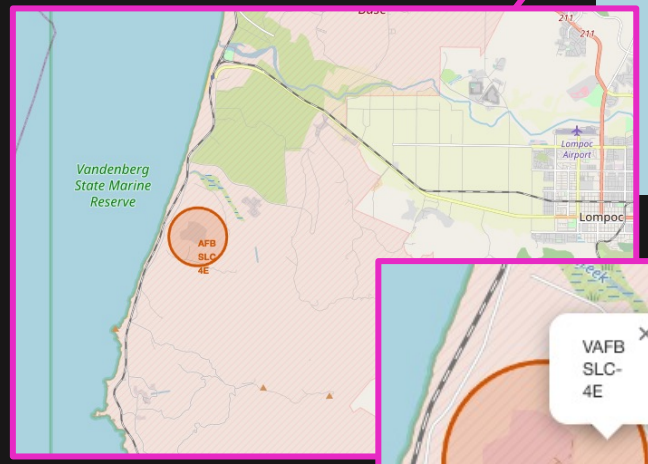
04 Results

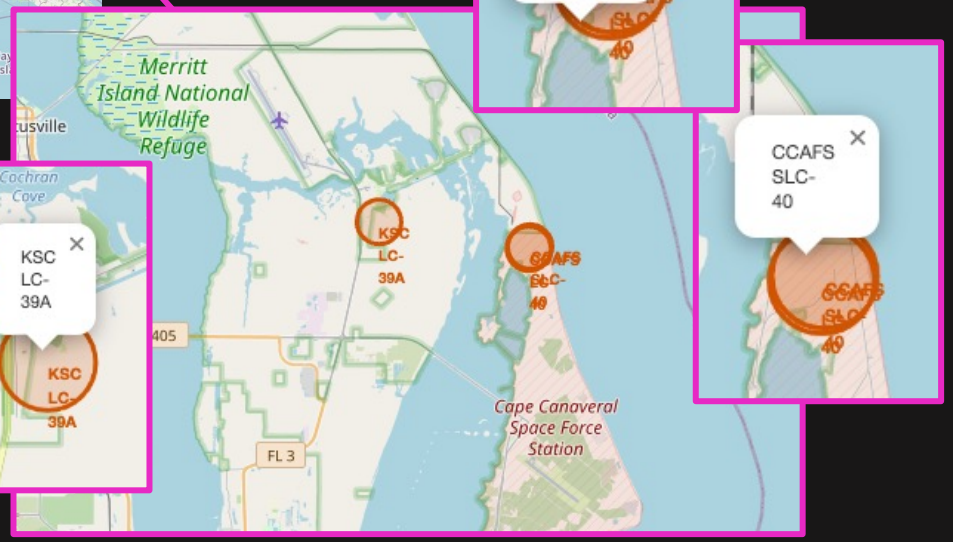
Launch Sites Proximity Analysis – Folium Interactive Map

Launching Sites on map

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





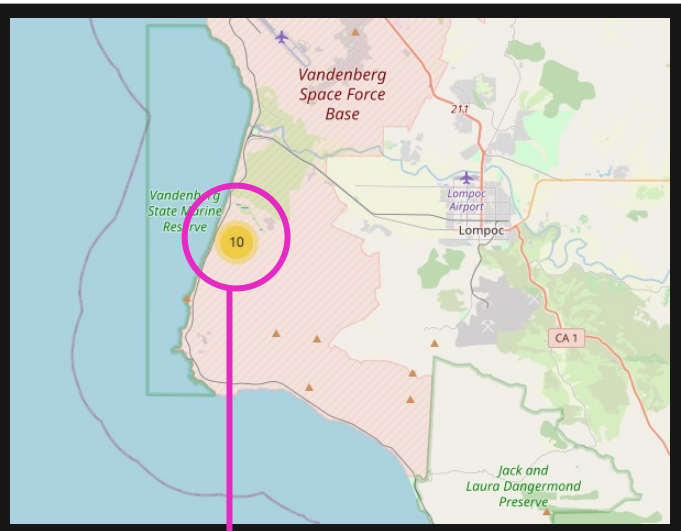


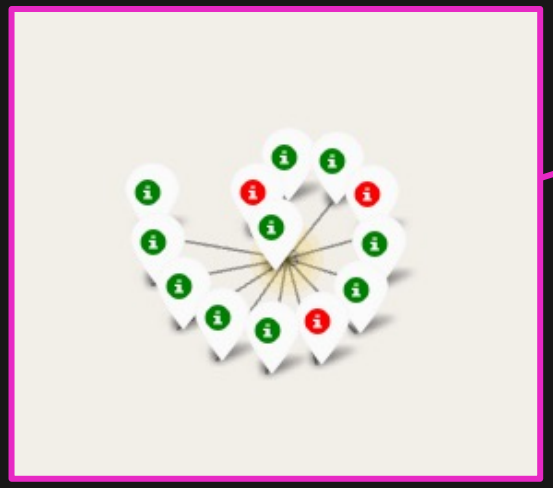
Using **Folium**, we were able to add a marking in the form of a circle and a label to each of the three different launching sites used by SpaceX.

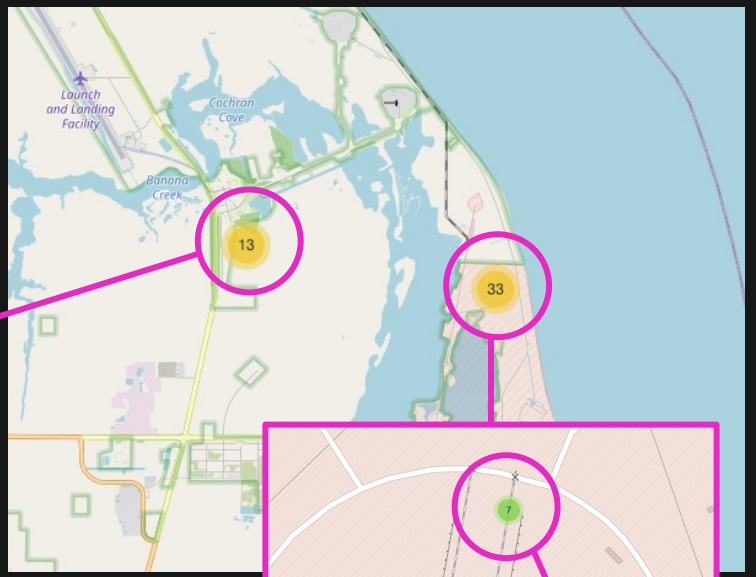
04 Results

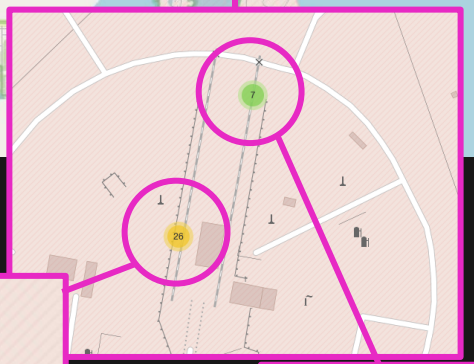
Launch Sites Proximity Analysis – Folium Interactive Map

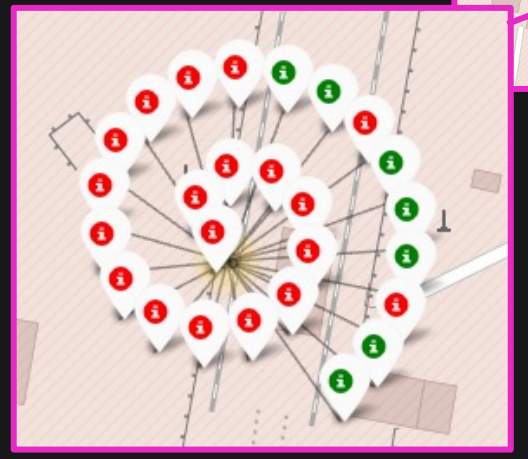
Marked successful/failed launches

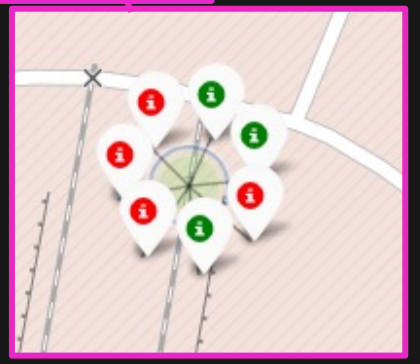












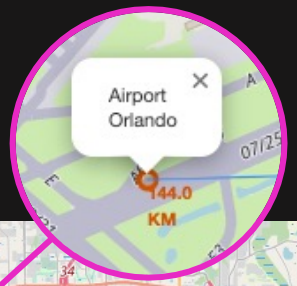
Using **Folium**, we were able to add a grouped popup mark with color indicating the outcome of each launch (green for successful launches and red for failed launches).

04 Results

Launch Sites Proximity Analysis – Folium Interactive Map

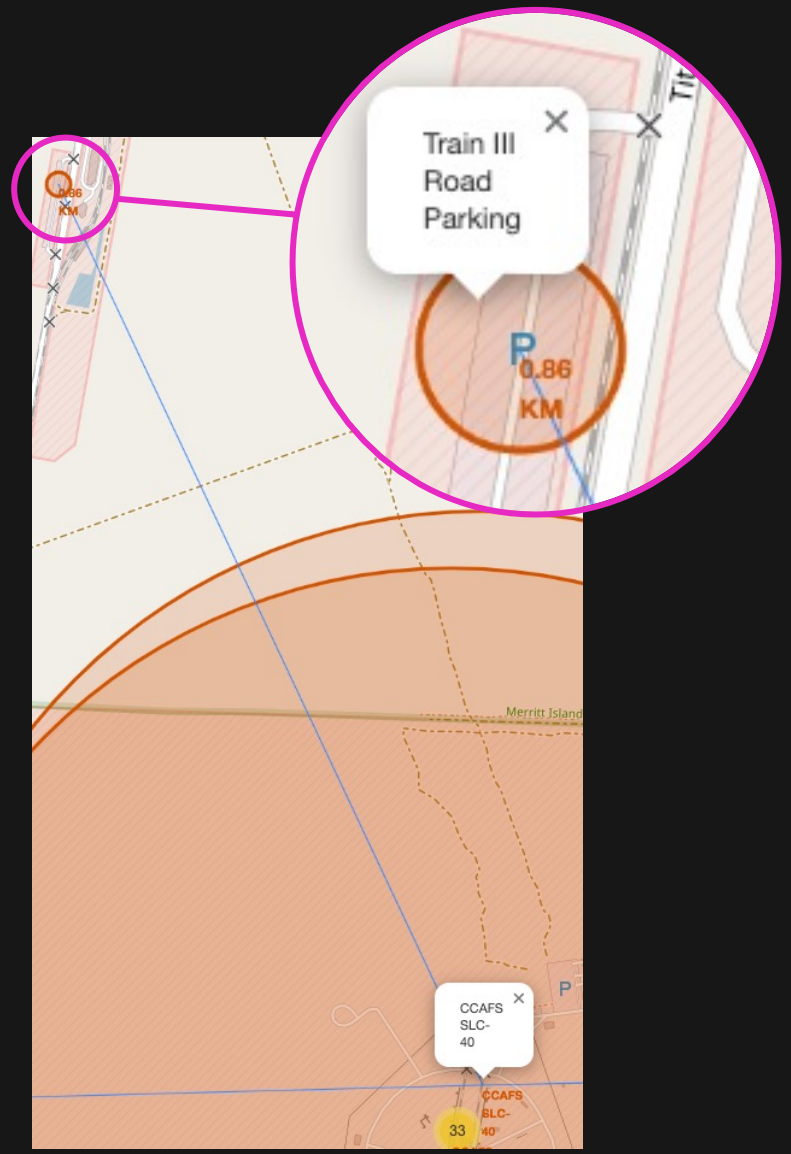
Proximity of Launching Site CCAFS SLC-40 to points of interest

Distance to coastline point **0.84km**



Distance to Orlando`s main Airport **144.0km**

Distance to Nearest Parking lot (TrainIII Road Parking) **0.86km**



Results

Dashboard with Plotly Dash

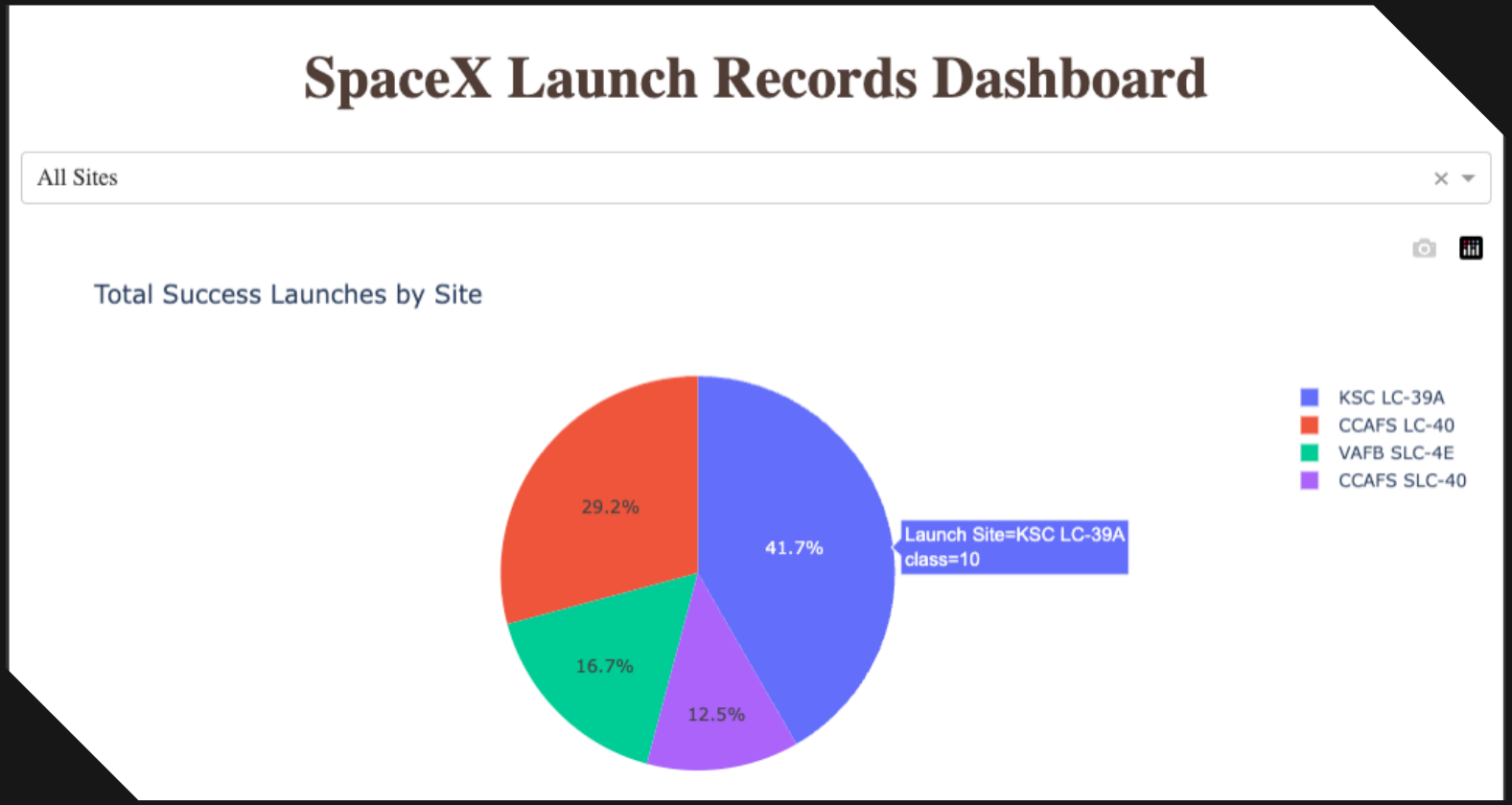
04 Results

Build a Dashboard with Plotly Dash

Launch success percentage among all launching sites

Using **Dash**, we were able to create an interactive Dashboard showing a pie chart showing the total number of success launches for different launch sites. The launch site (whether a specific one or all launching sites together) could be selected thanks to a dropdown menu (see **Appendix 06**).

When selecting 'All Sites', we can see that the Launching site with higher launching success rate is **KSC LC-39A**.

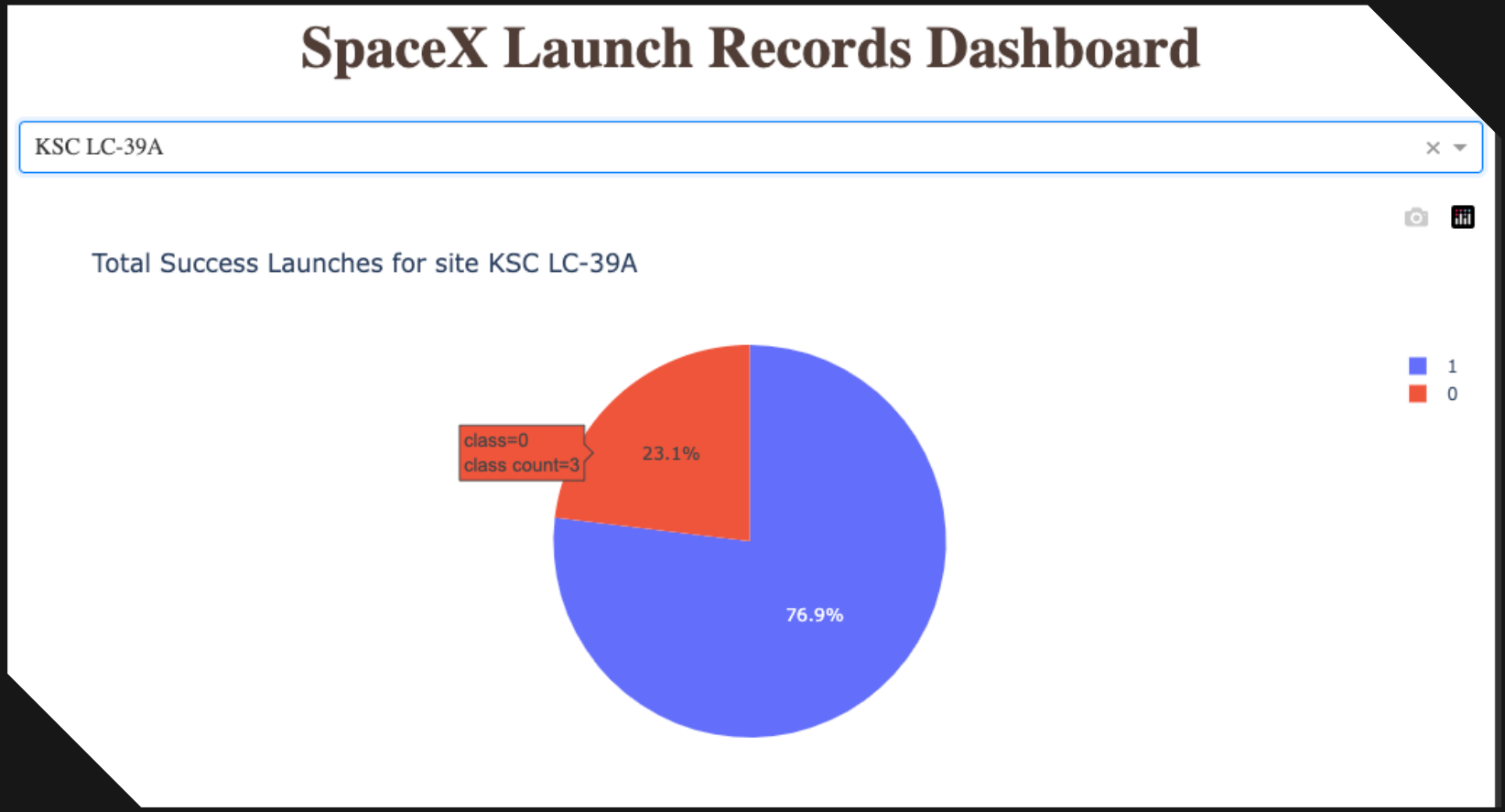


04 Results

Build a Dashboard with Plotly Dash

Launching success ratios of KSC LC-39A

We can look now specifically at the launch site data of KSC LC-39A: we can see that, out of **13 total launches**, 10 were successful (accounting to **76.9%** of all launches in this site) while 3 were failed launches (accounting for **23.1%**).

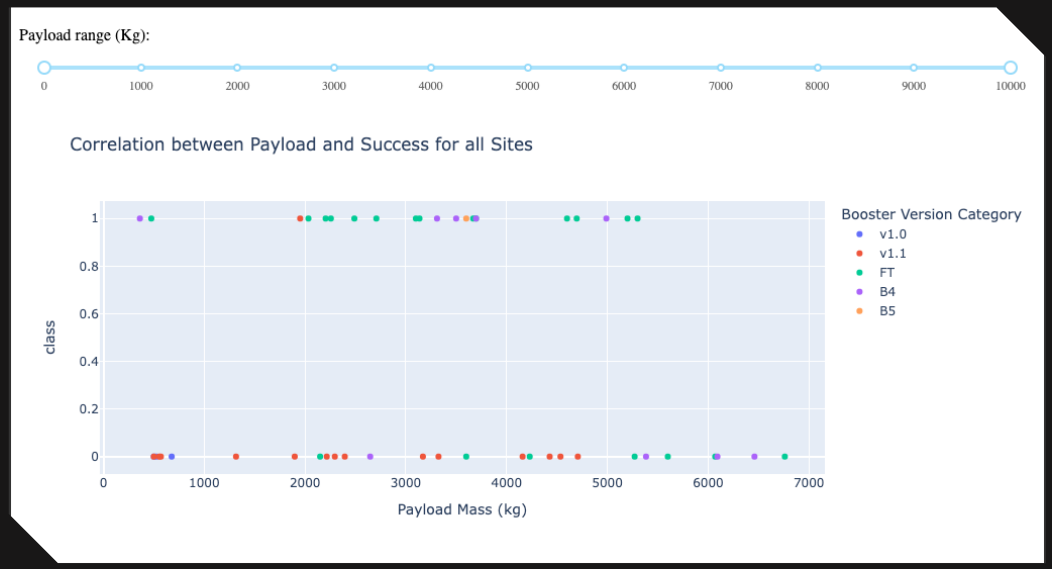


04 Results

Build a Dashboard with Plotly Dash

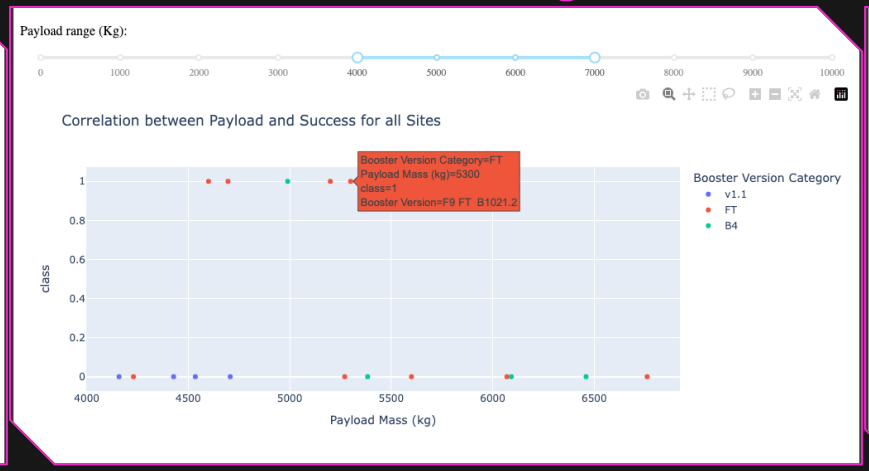
Influence of Payload on launching success for all Launching sites

Using **Dash**, we were able to create an interactive Dashboard with a scatter plot showing Payload mass vs. class of the launching (class = 0 for failed launches and class = 1 for successful launches).
The range of Payload values could be selected thanks to a slider, ranging from 0 to 10000kg.
Looking at the initial chart (top left figure), we divided the missions between those carrying a 'low' payload (between 0 and 4000kg), those carrying a 'middle' payload (between 4000 and 7000kg), and those carrying 'heavy' payloads (between 7000 and 10000kg).



Low payload range
0 - 4000kg

Middle payload range
4000 - 7000kg



Heavy payload range
7000 - 10000kg

Results

Predictive Analysis - Classification

04 Results

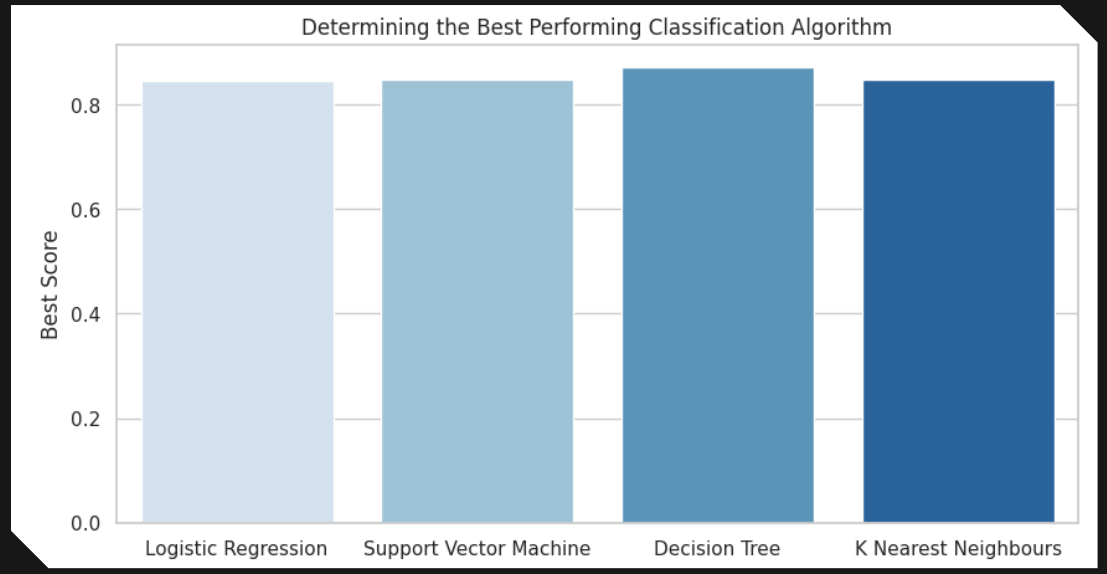
Predictive Analysis – Classification

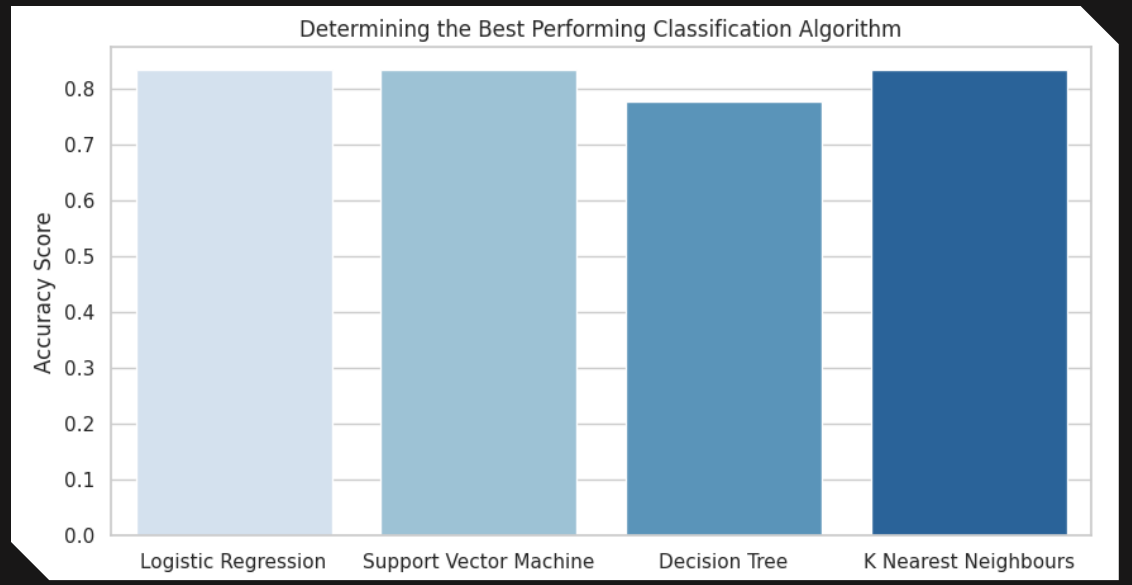
Four **Machine Learning Prediction algorithms** were used to model our landing data: **logistic regression** (**LR**), **support vector machine** (**SVM**), **decision tree** (**DT**) and **K nearest neighbor** (**KNN**).

In order to decide which model performs best in predicting rocket landing, we obtained the **best_score** and the **accuracy** (**f1-score**) of each model and compared between them.

Even though DT had the best score (0.87), its accuracy was the lowest (0.78). For this reason, we would not take this as the best model.

	Algorithm	Accuracy Score	Best Score
0	Logistic Regression	0.833333	0.846429
1	Support Vector Machine	0.833333	0.848214
2	Decision Tree	0.777778	0.871429
3	K Nearest Neighbours	0.833333	0.848214



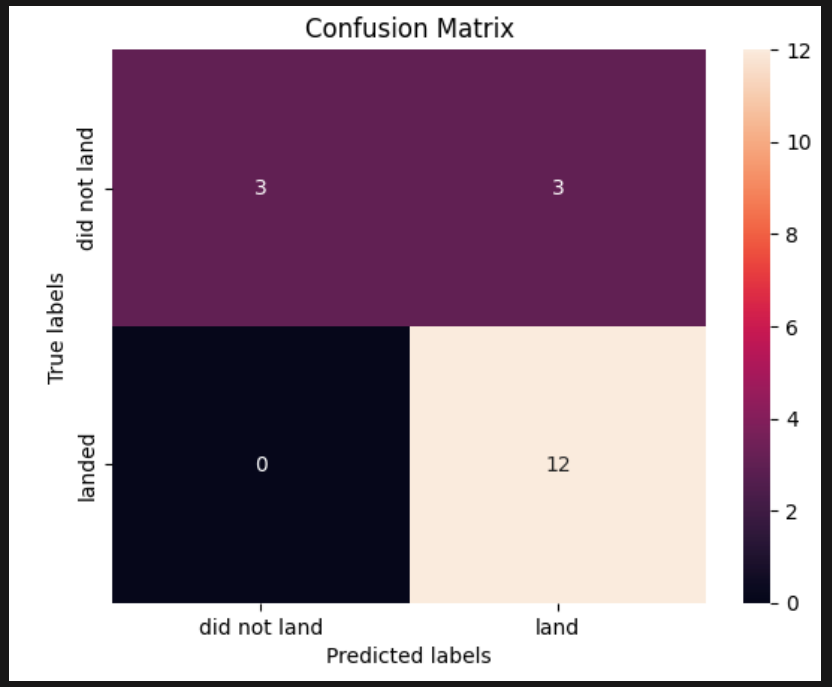


04 Results

Predictive Analysis – Classification

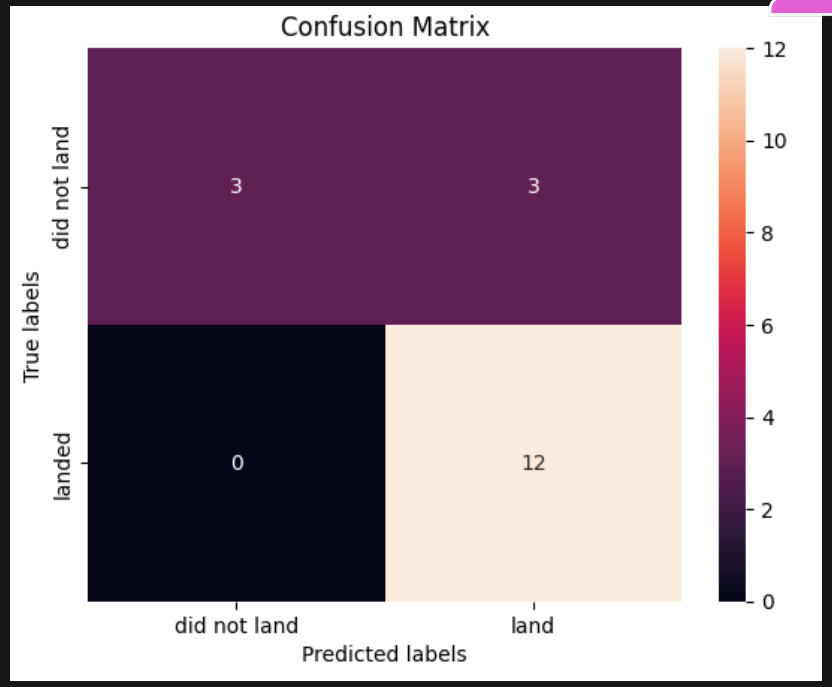
Looking at the two next best models, we see that SVM and KNN have the same **confusion matrix**. However, KNN has a slight better accuracy score when using the validation data. For this reason, we chose KNN as the best model.

Support Vector Machine Model




```
print("tuned hpyerp meters :(best parameters) ",svm_cv.best_params_)
print("accuracy :", cv.best_score_)
tuned hpy parameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy  0.8482142857142856
```

K-Nearest Neighbor Model




```
print("tuned hpyerp meters :(best parameters) ",KNN_cv.best_params_)
print("accuracy :", cv.best_score_)
tuned hpy parameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
accuracy  0.8482142857142858
```


Conclusions

05 Concluisons

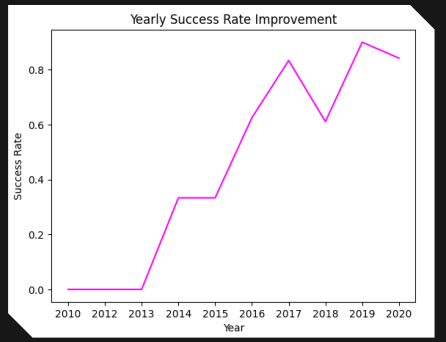
- Launching as well as landing of rockets has improved over the years. This increase in success rate is most likely due to an increase in technology and experience from SpaceX company.

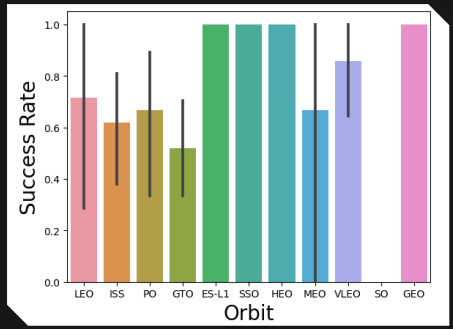
- The weight carried by the rockets seems to not determine launch or landing success, with the exception of very high payloads.

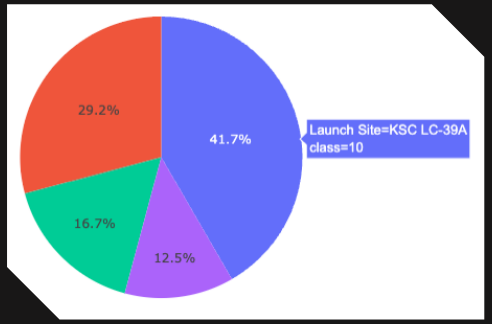
- Rockets that orbit **SSO** (Sun synchronous orbit) and **ULEO** (Very Low Earth Orbit) have a highest probability of landing compared to other orbits. This could be due to the proximity of the orbit to the Earth`s surface, at least in the case of VLEO (with a distance of <400km to the surface). However, this will be more specific in the future, when other orbits such as ES-L1 or HEO are further explored.

- Launch site **KSC LC-39A** is the most successful launch Site, with **41.7%** of total successful launches at a launch success rate of **76.9%**.

- According to our analysis, **KNN** is the best performing algorithm to predict landing of rockets.







Appendix

06 Appendix

Appendix #01 Name of Boosters launched form all three launching sites.

```
df_booster_version=df['BoosterVersion'].unique()
df_booster_version
```

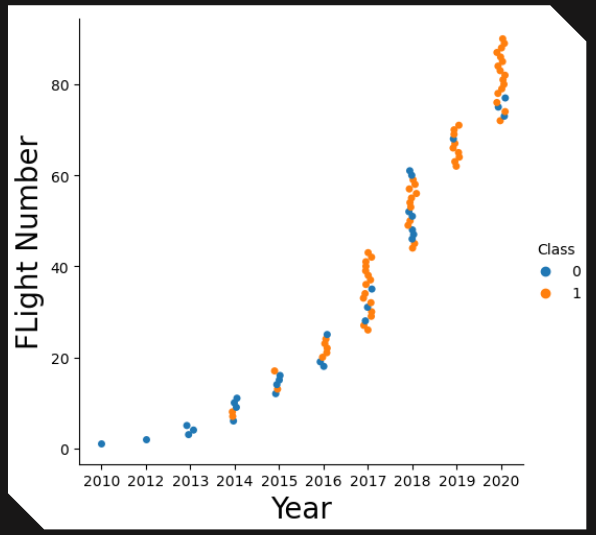
```
array(['Falcon 9'], dtype=object)
```

Appendix #02 Number of times that each orbit has been used for missions.

```
df_orbit = df.groupby("Orbit")[["Orbit"]].count()
df_orbit
```

Orbit	Orbit
ES-L1	1
GEO	1
GTO	27
HEO	1
ISS	21
LEO	7
MEO	3
PO	9
SO	1
SSO	5
VLEO	14

Appendix #03 Flight numbers organized by year.



Appendix #04 Landing success rate of SpaceX rocket Falcon 9 missions.

```
df2 = df.groupby("Date")[["Class"]].mean()
df2
```

Date	Class
2010	0.000000
2012	0.000000
2013	0.000000
2014	0.333333
2015	0.333333
2016	0.625000
2017	0.833333
2018	0.611111
2019	0.900000
2020	0.842105

GitHub Link

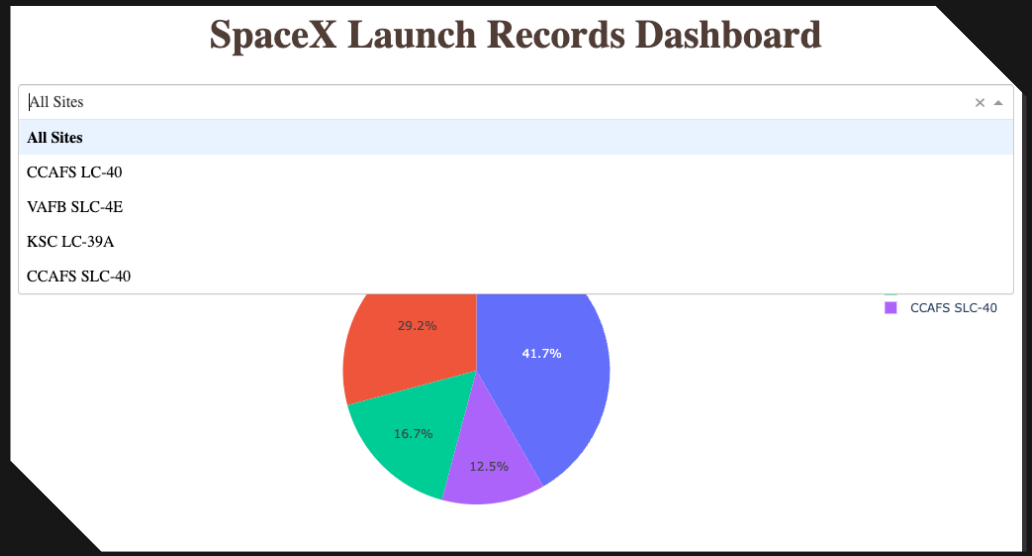
06 Appendix

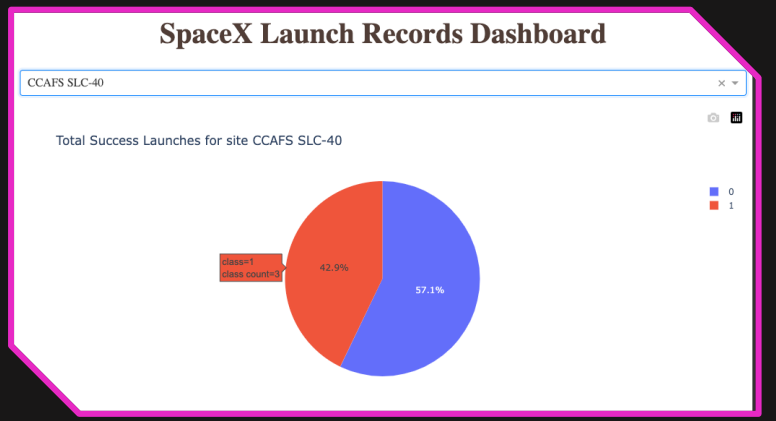
Appendix #05 Number of missions per year

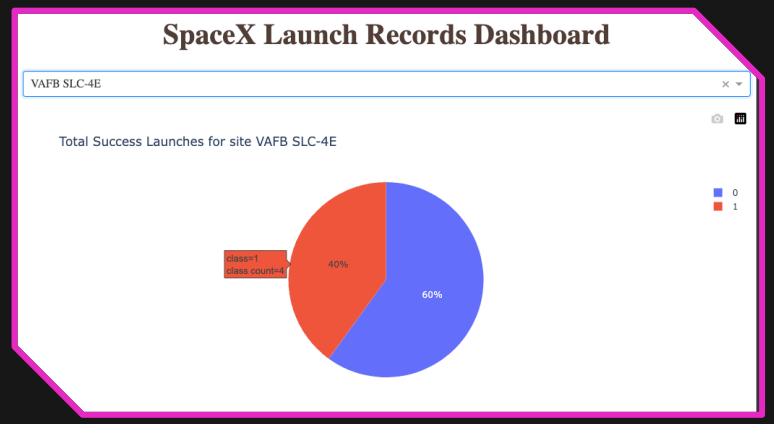
```
# How many missions per year?
df4 = df.groupby("Date")[["Date"]].count()
df4
```

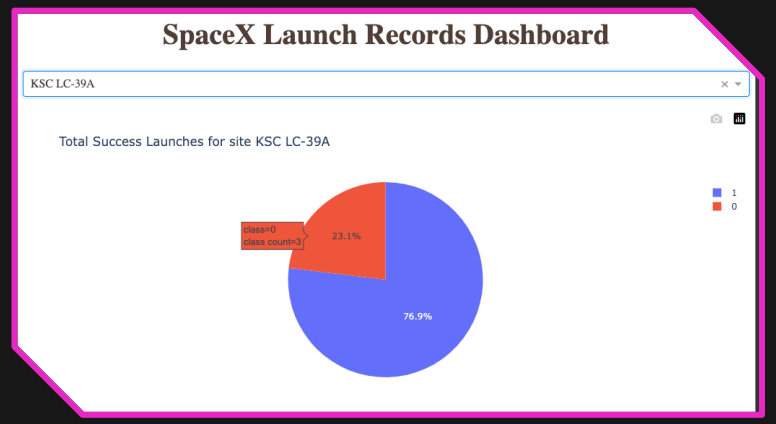
Date	Date
2010	1
2012	1
2013	3
2014	6
2015	6
2016	8
2017	18
2018	18
2019	10
2020	19

Appendix #06 Selection of different launching sites on the Interactive Dashboard through a dropdown menu.









GitHub Link


Thank you!