Propelling Space Y to a new orbit with Data Science

David Rosado Belza

December 7, 2024

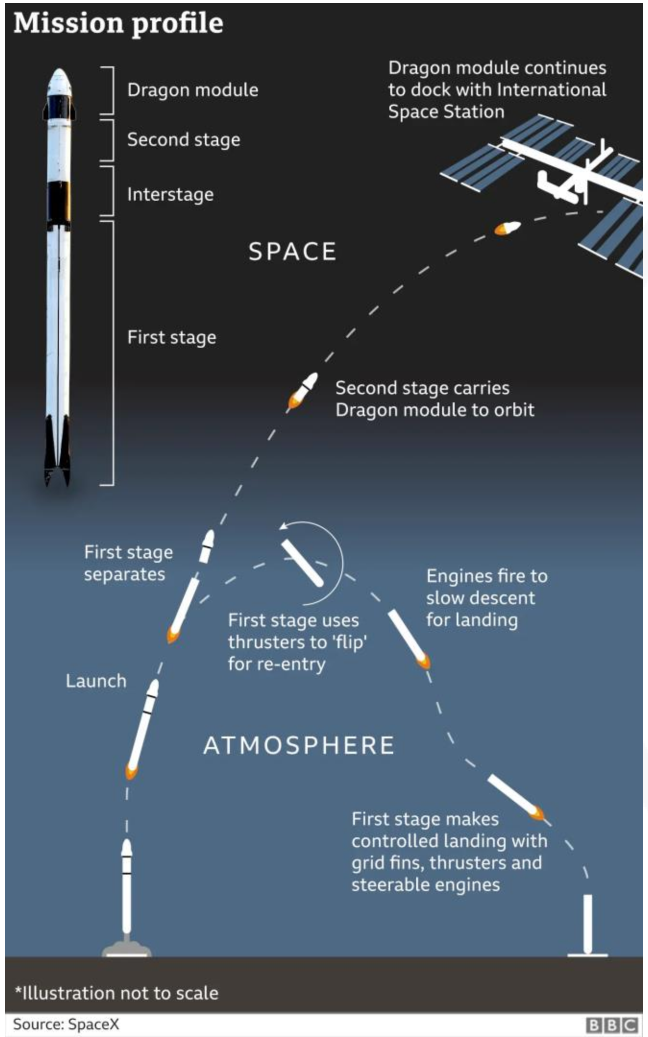
OUTLINE

- Executive Summary
- Introduction
- Methodology
- Results
- Discussion
- Conclusion

EXECUTIVE SUMMARY

- Methodology
  - Data Collection
  - Data Wrangling
  - Exploratory Analysis with Pandas and Matplotlib
  - Exploratory Analysis with SQL
  - Interactive Visual Analytics
  - Predictive Analysis using Classification Algorithms
- Results
  - Exploratory Data Analysis Results
  - Visual Analytics Results
  - Predictive Analysis Results

INTRODUCTION



- **General overview**

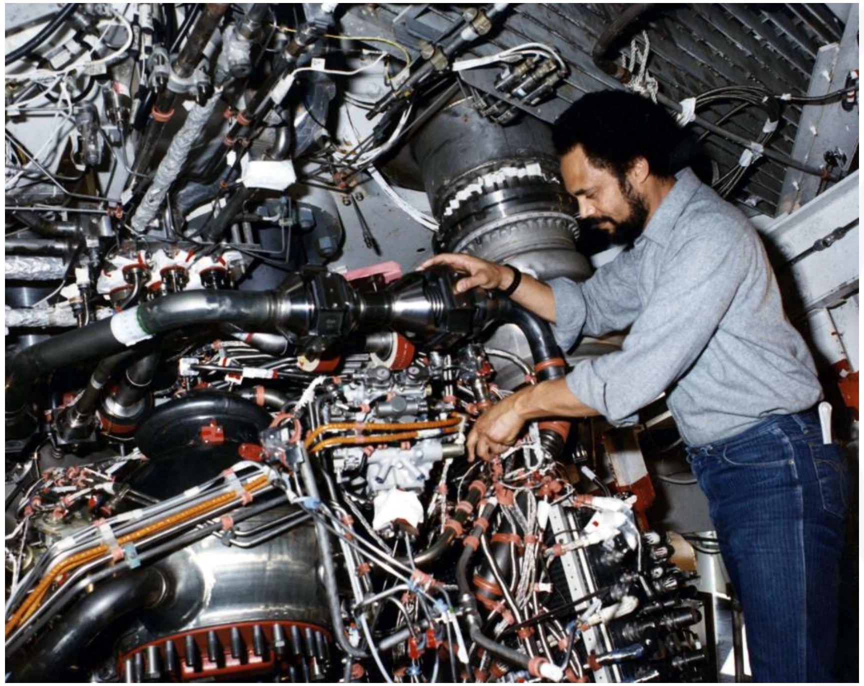
We are Space Y, a young company aiming to compete in the space launch market. Our goal: to dethrone our main competitor, Space X. To achieve this, we have chosen the path of learning from the mistakes of other companies. We will accomplish this by making data-driven decisions through thorough analyses, which will help us identify the critical areas of the launch process that require greater attention.

- **Problem to be solved**

Space X is the big player in the launch market, largely thanks to a partially reusable rocket model that saves millions of dollars in costs. However, this model comes with risks, and on some occasions, the launch of these rockets ends in disaster, causing significant losses for both clients and the company.

We aim to do better by ensuring that our launches are safer and more efficient. We want to identify the factors that may have contributed to the failures in our competitor's launches (such as excessive payload, unsuitable launch sites, or overly distant orbits, etc...) so that we can address potential issues in our own operations.

METHODOLOGY



- Data Collection
  - Collecting data from Space X REST API
  - Web Scrapping data from Wikipedia
- Data Wrangling
- Exploratory Data Analysis
  - EDA with SQL
  - EDA with visualisations using Pandas and Matplotlib
  - Interactive visual analytics with Dash and Folium
- Predictive Analysis for Classification
  - Logistic Regression
  - Suppot Vector Machine
  - Decission Trees
  - K-Nearest Neighbors

METHODOLOGY

- **<u>Data Collection</u>**: In this part we collect the data sets required for our analysis using the following two methods methods.

   1) **<u>Making a request to the Space X REST API:</u>**

   We retrieve the data from past Space X launches by using a GET request, select the fields we are interested in (Booster version, Payload mass, Orbit, etc...), and create a dataframe with them. Then we filter the data to get only the data for the Booster version Falcon 9. Finally, we fill in the missing values for payload mass with the mean value.

   <u>2) Web Scrapping</u>:

   We extract launch records for Falcon 9 boosters from Wikipedia using BeautifulSoup, getting all the information from the columns we are interested in. In the last step we parse the retrieved data to create a dataframe by using a dictionary.

METHODOLOGY

- **Data Collection**: **Making a request to the Space X REST API:**

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/da
```

We should see that the request was successfull with the 200 status response code

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

Using the dataframe `data` print the first 5 rows

```
# Get the head of the dataframe
data.head()
```

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	s
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 33, 'altitude': None, 'reason': 'merlin engine failure'}]	Engine failure at 33 seconds and loss of vehicle	[]	
							[{'time': 301, 'altitude':	Successful first stage burn and transition to second stage, maximum		

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'
data_falcon9 = launch_df[launch_df['BoosterVersion']!='Falcon 1']
data_falcon9.head()
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None
5	8	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False	None
6	10	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False	None
7	11	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None
8	12	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False	None

Now that we have removed some values we should reset the FlgihtNumber column

```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

```
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/pandas/core/indexing.py:1773: SettingWithCopyWarnin
g:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy
  self._setitem_single_column(ilocs[0], value, pi)
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs
4	1	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False
5	2	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False

Task 3: Dealing with Missing Values

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace `np.nan` values in the data with the mean you calculated.

```
# Calculate the mean value of PayloadMass column
meanpay = data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].fillna(meanpay)
data_falcon9.isnull().sum()
```

```
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy
  after removing the cwd from sys.path.
```

```
FlightNumber       0
Date               0
BoosterVersion     0
PayloadMass        0
Orbit              0
LaunchSite         0
Outcome            0
Flights            0
GridFins           0
Reused             0
Legs               0
LandingPad        26
Block              0
ReusedCount        0
Serial             0
Longitude          0
Latitude           0
dtype: int64
```

You should see the number of missing values of the `PayLoadMass` change to zero.

Now we should have no missing values in our dataset except for in `LandingPad`.

We can now export it to a **CSV** for the next section,but to make the answers consistent, in the next lab we will provide data in a pre-selected date range.

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

Complete jupyter notebook at https://github.com/Darobel/Applied_Data_Science_Capstl-/blob/main/Lab1_spacex-data-collection-api.ipynb

METHODOLOGY

- **Data Collection**: **Web Scrapping**:

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

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

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(data,'lxml')
```

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`, please check the external re
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)
```

```
<table class="wikitable plainrowheaders collapsible" style="width: 100%;">
<tbody><tr>
<th scope="col">Flight No.
</th>
<th scope="col">Date and<br/>time (<a href="/wiki/Coordinated_Universal_Time" title="Coordinated Universal Time">UTC</a>)
</th>
```

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 ( )']

# 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']=[]
```

Next, we just need to fill up the `launch_dict` with launch records extracted from table rows.

Usually, HTML tables in Wiki pages are likely to contain unexpected annotations and other types of noises, such as reference links `B0004.1[8]`, missing values `N/A [e]`, inconsistent formatting, etc.

To simplify the parsing process, we have provided an incomplete code snippet below to help you to fill up the `launch_dict`. Please complete the following code snippet with TODOs or you can choose to write your own logic to parse all launch tables:

```
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:
```

Complete jupyter notebook at
https://github.com/Darobel/Applied_Data_Science_Capstl-/blob/main/Lab2_webscraping.ipynb

METHODOLOGY

- **Data Wrangling**: In this part of the analysis we determine the most suited variable to use as a label when training our models. We analyse this depending on the success in the landing maneuver when the first stage of a Falcon 9 comes back from a mission. We estimate the number of launches on each of the available sites, the occurrence of the different orbits, and the frequency of the different outcomes for each mission.

TASK 1: Calculate the number of launches on each site

The data contains several Space X launch facilities: Cape Canaveral Space Launch Complex 40 **VAFB SLC 4E** , Vandenberg Air Force Base Space Launch Complex 4E **(SLC-4E)**, Kennedy Space Center Launch Complex 39A **KSC LC 39A** .The location of each Launch Is placed in the column `LaunchSite`

Next, let's see the number of launches for each site.

Use the method `value_counts()` on the column `LaunchSite` to determine the number of launches on each site:

```
# 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
```

Each launch aims to an dedicated orbit, and here are some common orbit types:

- **LEO**: Low Earth orbit (LEO)is an Earth-centred orbit with an altitude of 2,000 km (1,200 mi) or less (approximately one-third of the radius of Earth),[1] or with at least 11.25 periods per day (an orbital period of 128 minutes or less) and an eccentricity less than 0.25.[2] Most of the manmade objects in outer space are in LEO [1].
- **VLEO**: Very Low Earth Orbits (VLEO) can be defined as the orbits with a mean altitude below 450 km. Operating in these orbits can provide a number of benefits to Earth observation spacecraft as the spacecraft operates closer to the observation[2].
- **GTO** A geosynchronous orbit is a high Earth orbit that allows satellites to match Earth's rotation. Located at 22,236 miles (35,786 kilometers) above Earth's equator, this position is a valuable spot for monitoring weather, communications and surveillance. Because the satellite orbits at the same speed that the Earth is turning, the satellite seems to stay in place over a single longitude, though it may drift north to south," NASA wrote on its Earth Observatory website [3] .
- **SSO (or SO)**: It is a Sun-synchronous orbit also called a heliosynchronous orbit is a nearly polar orbit around a planet, in which the satellite passes over any given point of the planet's surface at the same local mean solar time [4] .
- **ES-L1** :At the Lagrange points the gravitational forces of the two large bodies cancel out in such a way that a small object placed in orbit there is in equilibrium relative to the center of mass of the large bodies. L1 is one such point between the sun and the earth [5] .
- **HEO** A highly elliptical orbit, is an elliptic orbit with high eccentricity, usually referring to one around Earth [6].

TASK 2: Calculate the number and occurrence of each orbit

Use the method `.value_counts()` to determine the number and occurrence of each orbit in the column `Orbit`

```
# 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
```

TASK 3: Calculate the number and occurence of mission outcome of the orbits

Use the method `.value_counts()` on the column `Outcome` to determine the number of `landing_outcomes` .Then assign it to a variable landing_outcomes.

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

`True Ocean` means the mission outcome was successfully landed to a specific region of the ocean while `False Ocean` means the mission outcome was unsuccessfully landed to a specific region of the ocean. `True RTLS` means the mission outcome was successfully landed to a ground pad `False RTLS` means the mission outcome was unsuccessfully landed to a ground pad. `True ASDS` means the mission outcome was successfully landed to a drone ship `False ASDS` means the mission outcome was unsuccessfully landed to a drone ship. `None ASDS` and `None None` these represent a failure to land.

```
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
```

TASK 4: Create a landing outcome label from Outcome column

Using the `Outcome` , create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome` ; otherwise, it's one. Then assign it to the variable `landing_class` :

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

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

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

	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

Complete jupyter notebook at https://github.com/Darobel/Applied_Data_Science_Capstl-/blob/main/Lab3-Data_wrangling.ipynb

METHODOLOGY

- **Exploratory Data Analysis**: In this part of the work we explore the dataset and perform a preliminary analysis that will allow us to find thoughtful insights regarding the collected data set that will be very useful for the predictive analysis. This exploratory analysis consisted of three stages:

    1) **EDA with SQL.**
    2) **EDA with Data Visualisation.**
    3) **Creating an interactive map using Folium.**
    4) **Creating an interactive dashboard with Ploty Dash.**
    5) **Performing an analysis using predictive algorithms.**

METHODOLOGY

- **Exploratory Data Analysis**: **EDA with SQL:** In this section we use a series of SQL queries with sqlite for analysing the dataset. These queries consisted of the following tasks:
  1) Display the names of the unique launch sites in the space mission.
  2) Display up to five records where the launch sites begin with the string 'CCA'.
  3) Display the total payload mass carried by boosters launched by NASA (CRS).
  4) Display the average payload mass carried by booster version F9v1.1.
  5) List the date when the first successful landing outcome in ground pad was achieved.
  6) List the names of the boosters which have success in drone ship and have a payload mass greater than 4000 but less than 6000.
  7) List the total number of successful and failure mission outcomes.
  8) List the names of the booster versions which have carried the maximum payload mass.
  9) 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.
  10) Rank the count of landing outcomes (such as Failure or Success depending on the kind drone ship/ground pad) between the dates 2010-06-04 and 2017-03-20, in descending order.

  You can check the complete jupyter notebook at https://github.com/Darobel/Applied_Data_Science_Capstl-/blob/main/Lab4_eda-sql-coursera_sqllite.ipynb

METHODOLOGY

- **Exploratory Data Analysis**: **EDA with Data Visualisation:**

We analyse the data by visualising them with a combination of Pandas, Matplotlib and Seaborn prompts. We explore different trends between the selected fields of the dataset using scatter plots (to see the relationship between variables), line plots (for plotting the success rate), and bar charts (to get the average launch success rate). Here we show a sample of them:

Complete jupyter notebook at https://github.com/Darobel/Applied_Data_Science_Capstl-/blob/main/Lab5_edadataviz.ipynb

METHODOLOGY

- **Exploratory Data Analysis**: **Creating an interactive map with Folium:** We also want to locate the launching sites and analyse them to understand how these locations are related to their environment. We perform tasks such as:
- Marking the launching sites and adding marker clusters to show launch success and failure in each site.
- Estimating the distances to different infrastructures such as highways or railroads and also the coastline and the closest city.
- Adding markers and lines to indicate these distances.

Complete jupyter notebook at https://github.com/Darobel/Applied_Data_Science_Capstl-/blob/main/Lab6_launch_site_location.ipynb

METHODOLOGY

- **Exploratory Data Analysis**: **Creating an interactive dashboard with Ploty.** We build an interactive dashboard for a better understanding of the data. In it we include:
- A dropdown list for selecting the launch site.
- A pie chart for showing the total successful launches for all sites or the one selected from the dropdown list.
- A slider for selecting the payload mass range.
- A scatter chart showing the payload mass and the outcome of each mission.

Complete jupyter notebook at https://github.com/Darobel/Applied_Data_Science_Capstl-/blob/main/lab7_spacex_dash_app.py

METHODOLOGY

- **Exploratory Data Analysis: Predictive analysis.** In this part we analyse the data using different predictive algorithms. With this purpose in mind we perform the following tasks:
- Prepare the data and adding an additional column for the 'Class' variable.
- Standardise the data.
- Split the data into train and test data in a proportion of 20% and 80%, respectively.
- Define different models and parameters, train and grid search for best parameters for different algorithms: Logistic regression, SVM, Decision Tree, and KNN.
- Evaluate each model in terms of its accuracy and confusion matrix.

Complete jupyter notebook at https://github.com/Darobel/Applied_Data_Science_Capstl-/blob/main/Lab8_SpaceX_ML_Prediction.ipynb

RESULTS

- Exploratory Data Analysis
  - EDA with SQL
  - EDA with visualisations using Pandas and Matplotlib
  - Interactive visual analytics with Dash and Folium
- Predictive Analysis for Classification
  - Logistic Regression
  - Suppot Vector Machine
  - Decission Trees
  - K-Nearest Neighbors

RESULTS

- **Exploratory Data Analysis: EDA with SQL**
  1) Display the names of the unique launch sites in the space mission.

```
%sql select distinct Launch_Site from SPACEXTBL
```

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

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

  2) Display up to five records where the launch sites begin with the string 'CCA'.

```
%sql select * from SPACEXTBL 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_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

RESULTS

- **Exploratory Data Analysis: EDA with SQL**

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

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

```
%sql select sum(PAYLOAD_MASS__KG_) from SPACEXTBL where Customer='NASA (CRS)'
```

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

sum(PAYLOAD_MASS__KG_)
45596

4) Display the average payload mass carried by booster version F9v1.1.

```
%sql select avg(PAYLOAD_MASS__KG_) from SPACEXTBL where Booster_Version='F9 v1.1'
```

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

avg(PAYLOAD_MASS__KG_)
2928.4

5) List the date when the first successful landing outcome in ground pad was achieved.

```
%sql select min(Date) from SPACEXTBL where Landing_Outcome='Success (ground pad)'
```

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

min(Date)
2015-12-22

RESULTS

- **Exploratory Data Analysis**: **EDA with SQL:**

6) List the names of the boosters which have success in drone ship and have a payload mass greater than 4000 but less than 6000.

```
%sql select Booster_Version from SPACEXTBL where Landing_Outcome='Success (drone ship)' and PAYLOAD_MASS__KG_>4000 and PAYLOAD_MASS__KG_<6000
```

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

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

7) List the total number of successful and failure mission outcomes.

```
%sql select distinct Mission_Outcome, count(*) from SPACEXTBL 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

RESULTS

- **Exploratory Data Analysis**: **EDA with SQL:**

8) List the names of the booster versions which have carried the maximum payload mass.

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

```
 * 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

9) 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.

```
%sql select substr(Date, 6,2) as Month, Landing_Outcome, Booster_Version, Launch_Site from SPACEXTBL where substr(Date,0,5)='2015' and Landing_Outcome='Failure (drone ship)'
```

```
 * 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

RESULTS

- **Exploratory Data Analysis**: **EDA with SQL:**

  10) Rank the count of landing outcomes (such as Failure or Success depending on the kind drone ship/ground pad) between the dates 2010-06-04 and 2017-03-20, in descending order.

```
%sql select Landing_Outcome, count(*) as Number from SPACEXTBL where Date between '2010-06-04' and '2017-03-20' group by Landing_Outcome order by Number desc
```

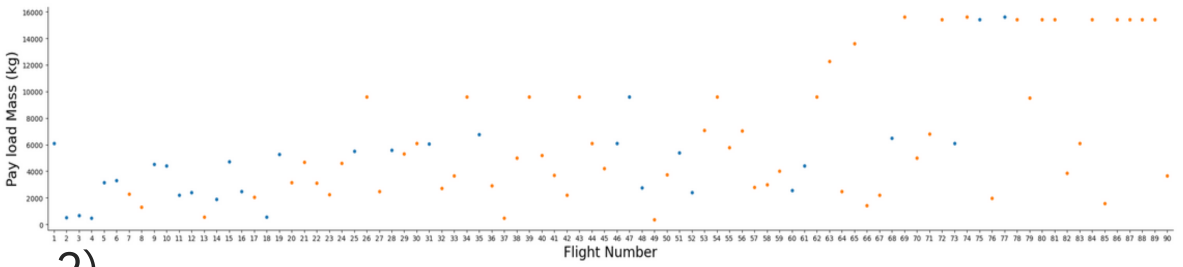
```
 * sqlite:///my_data1.db
Done.
```

Landing_Outcome	Number
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

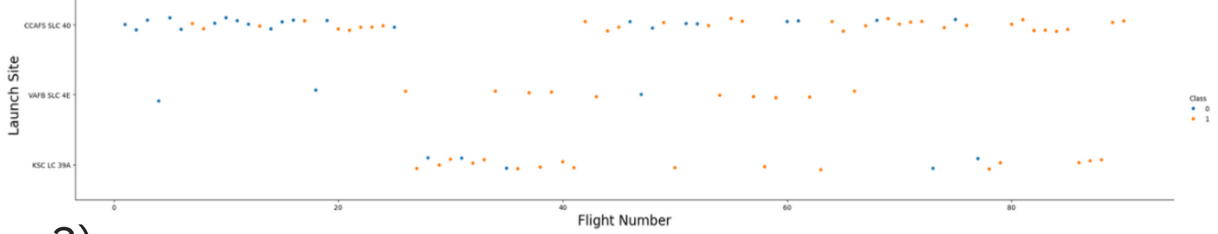
RESULTS

- **Exploratory Data Analysis**: **EDA with Data Visualisation:**

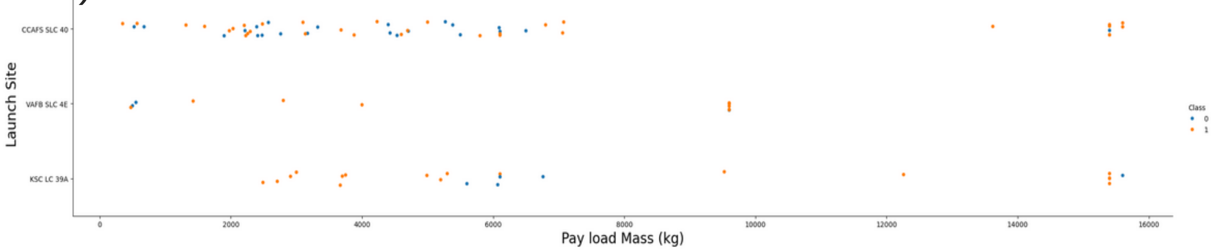
1)



2)

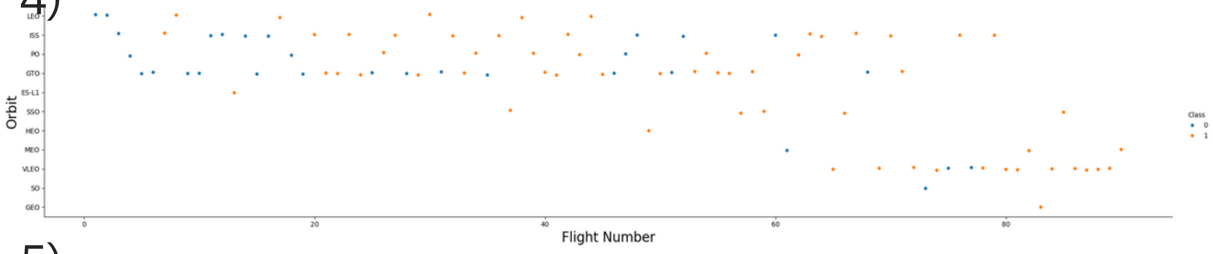


3)

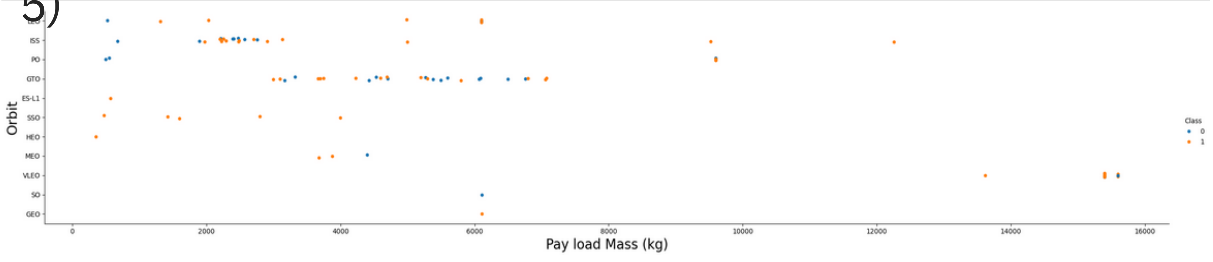


Scatter plots depicting the relationship between 1) Flight number vs Payload Mass, 2) Flight number vs Launch Site, 3) Payload Mass vs Launch Site, 4) Flight number vs Orbit Type, 5) Payload Mass vs Orbit Type.

4)



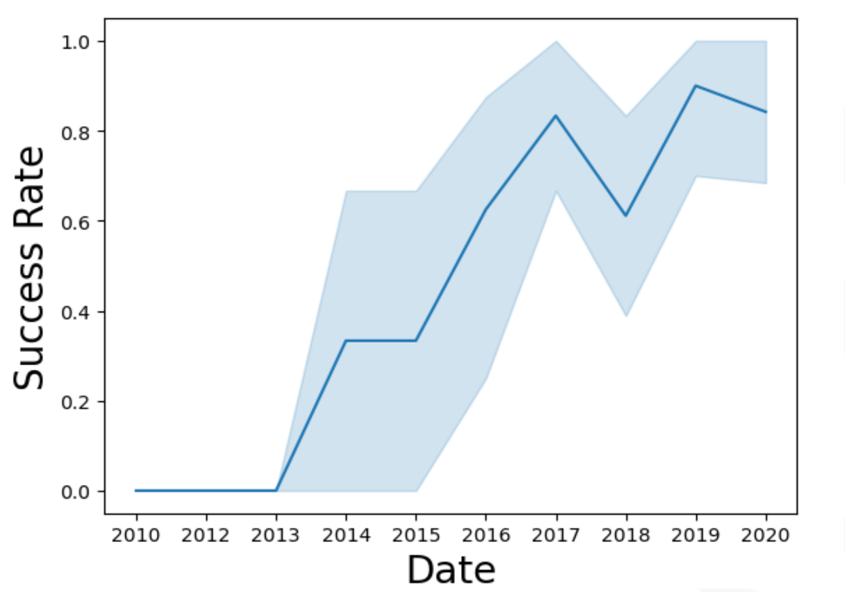
5)



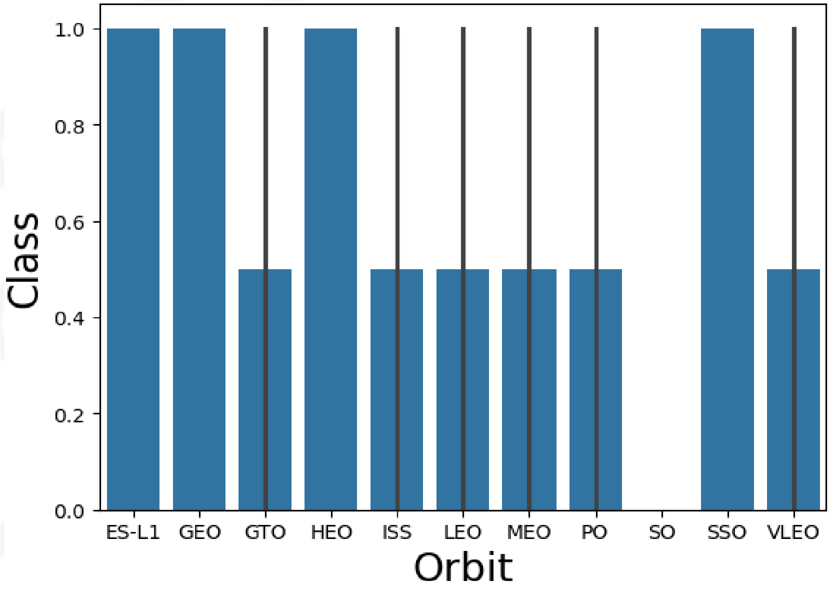
RESULTS

- **<u>Exploratory Data Analysis</u>: <u>EDA with Data Visualisation:</u>**

Line plot showing the relationship between the Success Rate vs the Date to visualise the yearly success of the launches.



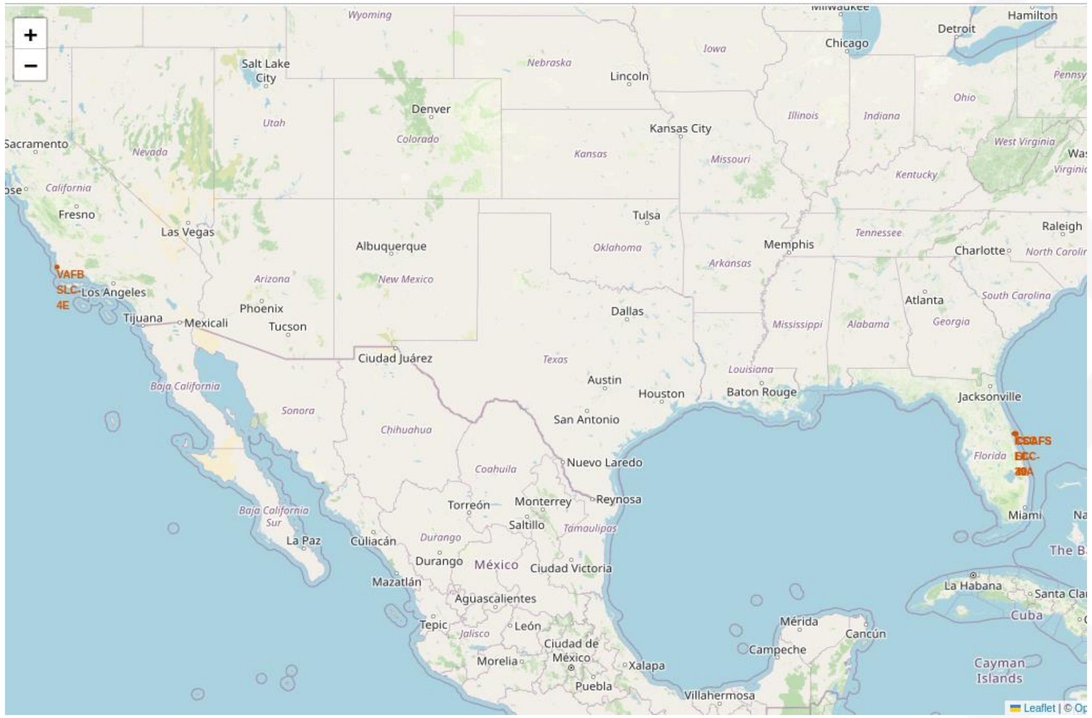
Bar chart for visualising the distribution of successful launches for each orbit type.



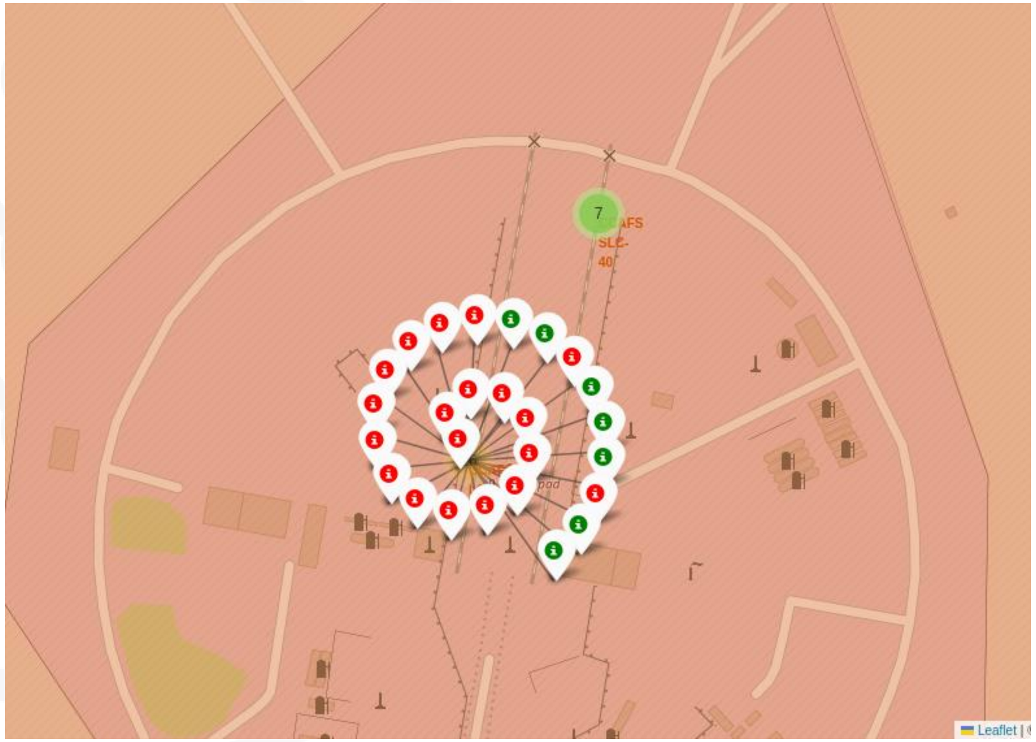
RESULTS

- **<u>Exploratory Data Analysis</u>**: **<u>Creating an interactive map with Folium:</u>**

- Marking the launching sites.



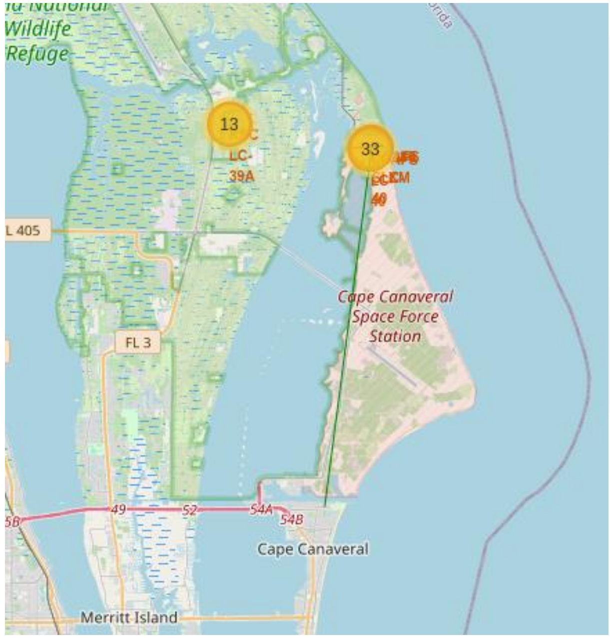
- Adding marker clusters to show launch success and failure in each site.

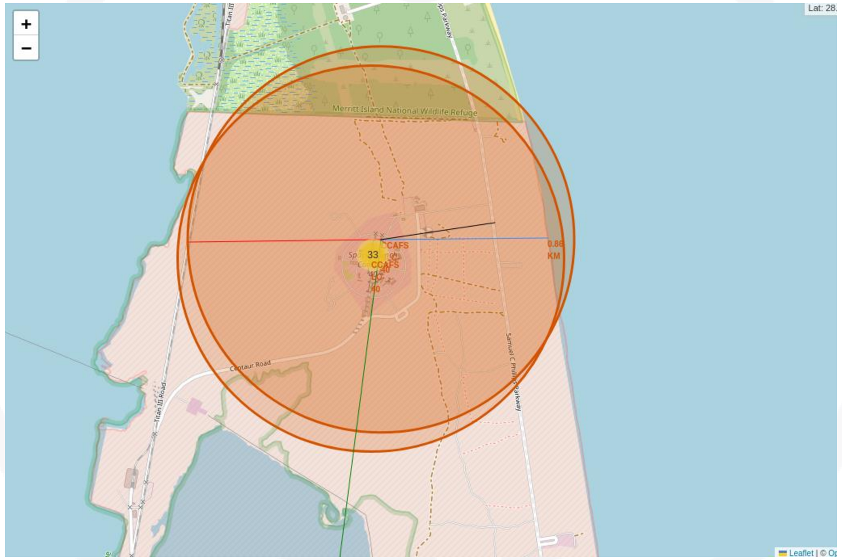


RESULTS

- **<u>Exploratory Data Analysis</u>: <u>Creating an interactive map with Folium:</u>**

- Estimating the distances to different infrastructures such as highways or railroads and also the coastline and the closest city.



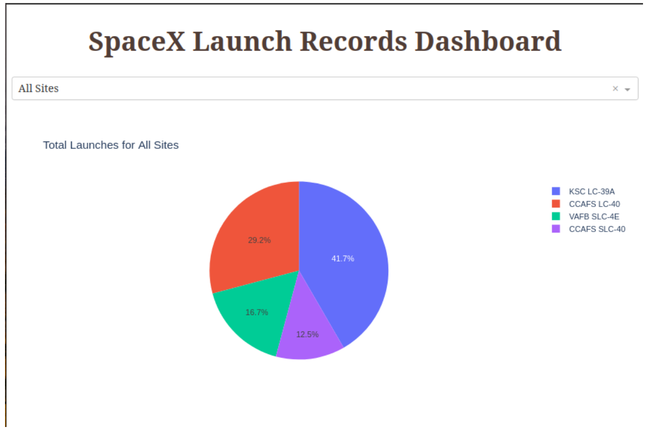


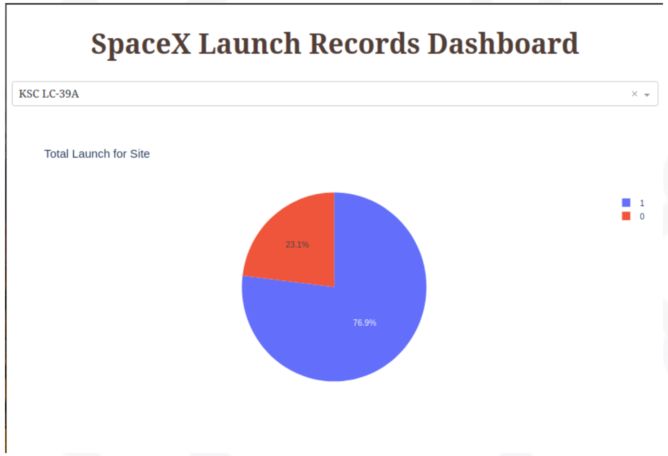
0.86 km to the coastline
17.54 km to the city
0.60 km to the road
0.99 km to the railroad

RESULTS

- **Exploratory Data Analysis: Creating an interactive dashboard with Ploty.**

We create a pie chart for the total launches with a dropdown menu that allows to select a specific location and show the successful launches.



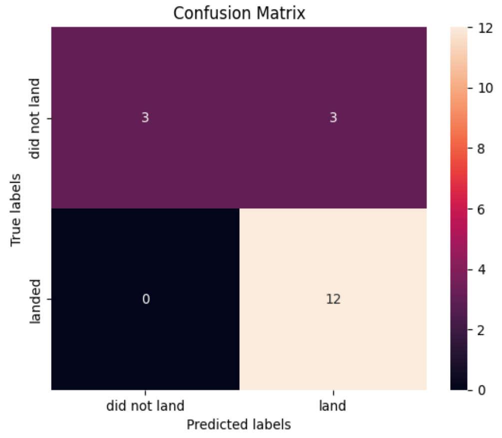


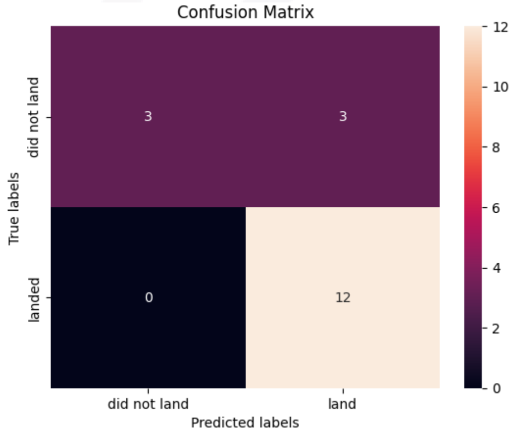
We create a scatter plot showing the payload mass vs the class variable. We add a slider to select different payload mass ranges.

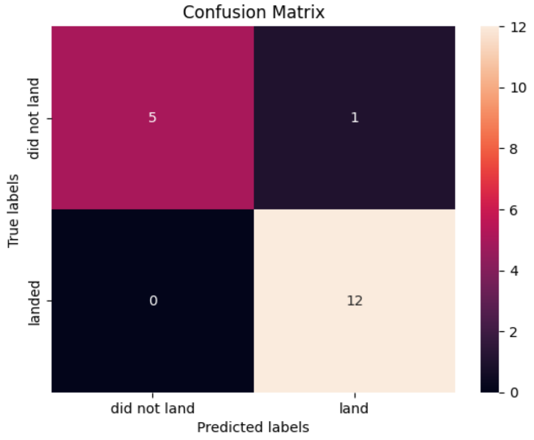
RESULTS

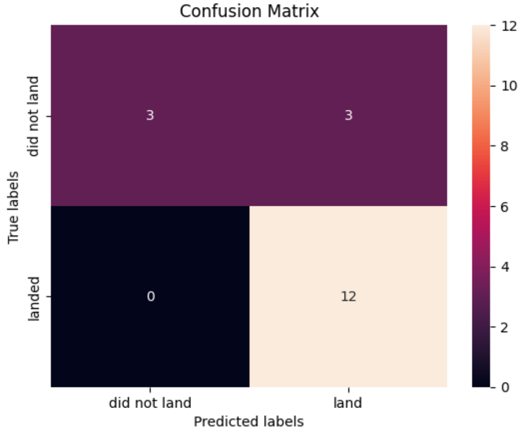
- **Exploratory Data Analysis: Predictive analysis.**

We obtain the confusion matrix for the four algorithms considered









Logistic regression
Accuracy: 0.83333...

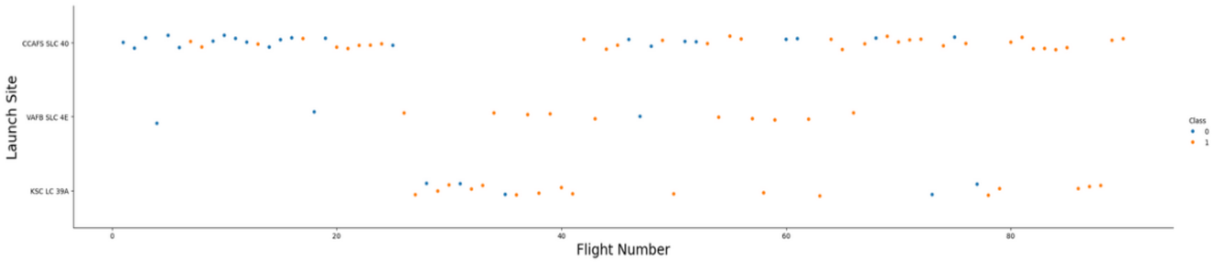
SVM
Accuracy: 0.83333...

Decision Tree
Accuracy: 0.88888...

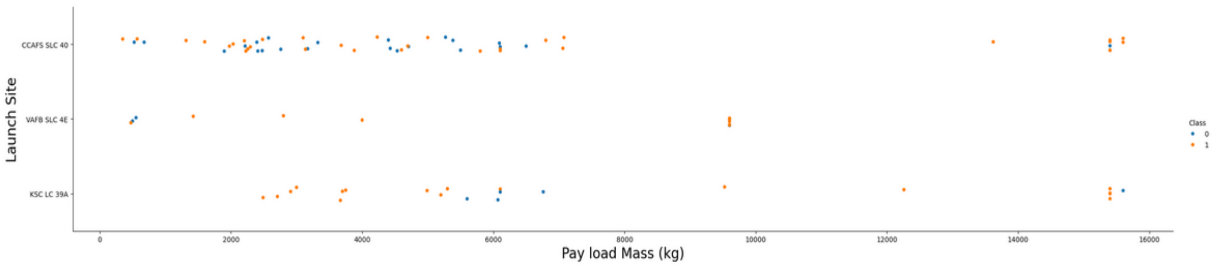
KNNK
Accuracy: 0.83333...

DISCUSION

- **Exploratory Data Analysis**: **EDA with Data Visualisation:**



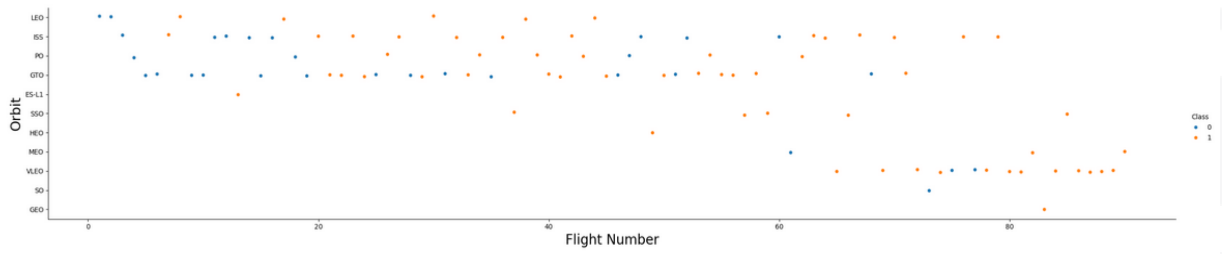
From the relation flight number vs launch site we see that the success rate increases as the number of flights increases. At the same time we observe that the CCAF5 SLC 40 launching site is the one with more successful launches since it is the most frequently used.



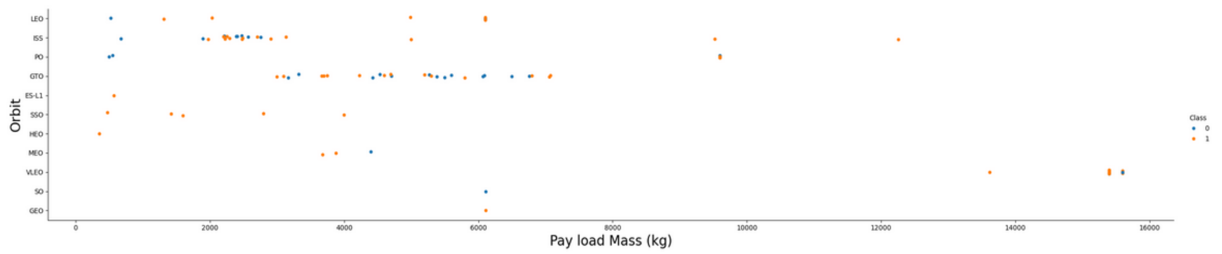
From the relation payload mass vs launch site we determine that there is no clear correlation between the payload mass and the launching site. However, we see that there are no launches in VAFB SLC 4E with payload masses over 10000 kg. We also note that launches with low payload masses are more frequent.

DISCUSION

- **Exploratory Data Analysis: EDA with Data Visualisation:**



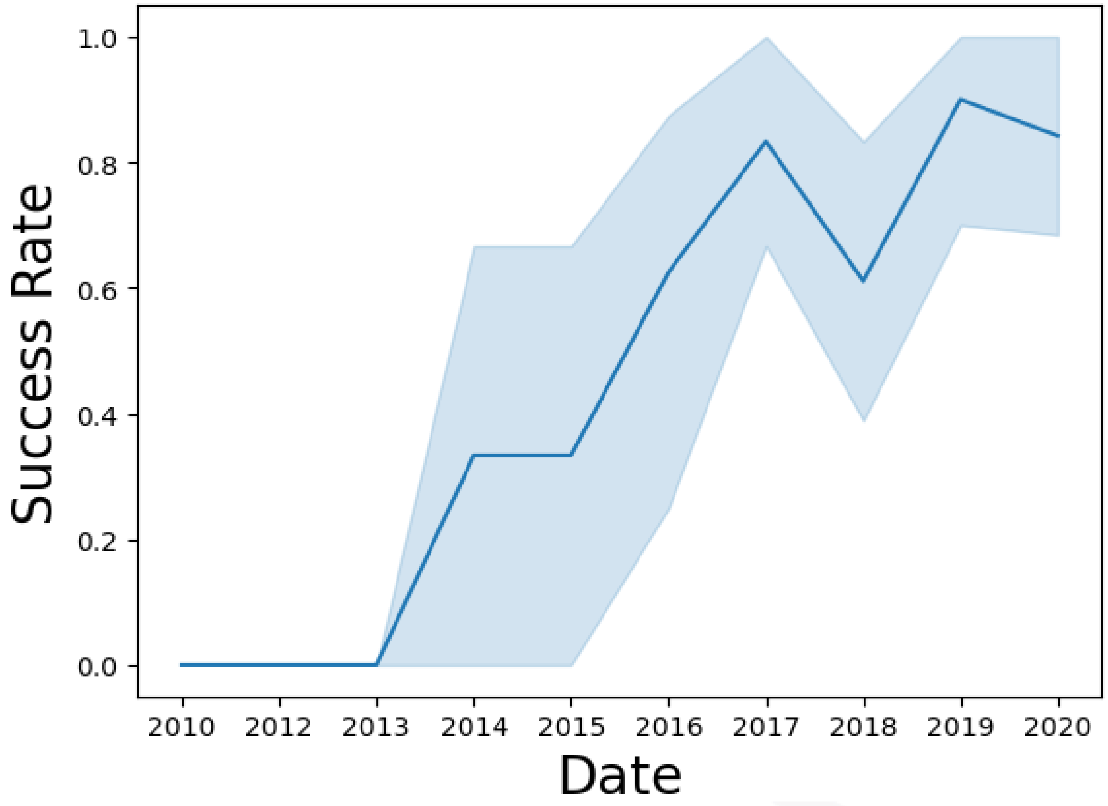
From the relation flight number vs orbit type we see that the orbits GTO, PO, LEO and ISS had more launches in the early years. As time passes, VLEO becomes the most frequent orbit. As stated in the previous relationships, as the flight number increases the number of successful landings also increases for all orbits.



From the relation payload mass vs orbit type we observe a trend of higher masses having higher success rates for ISS, LEO, SSO, and PO orbits. There is no clear pattern for the GTO orbits.

DISCUSION

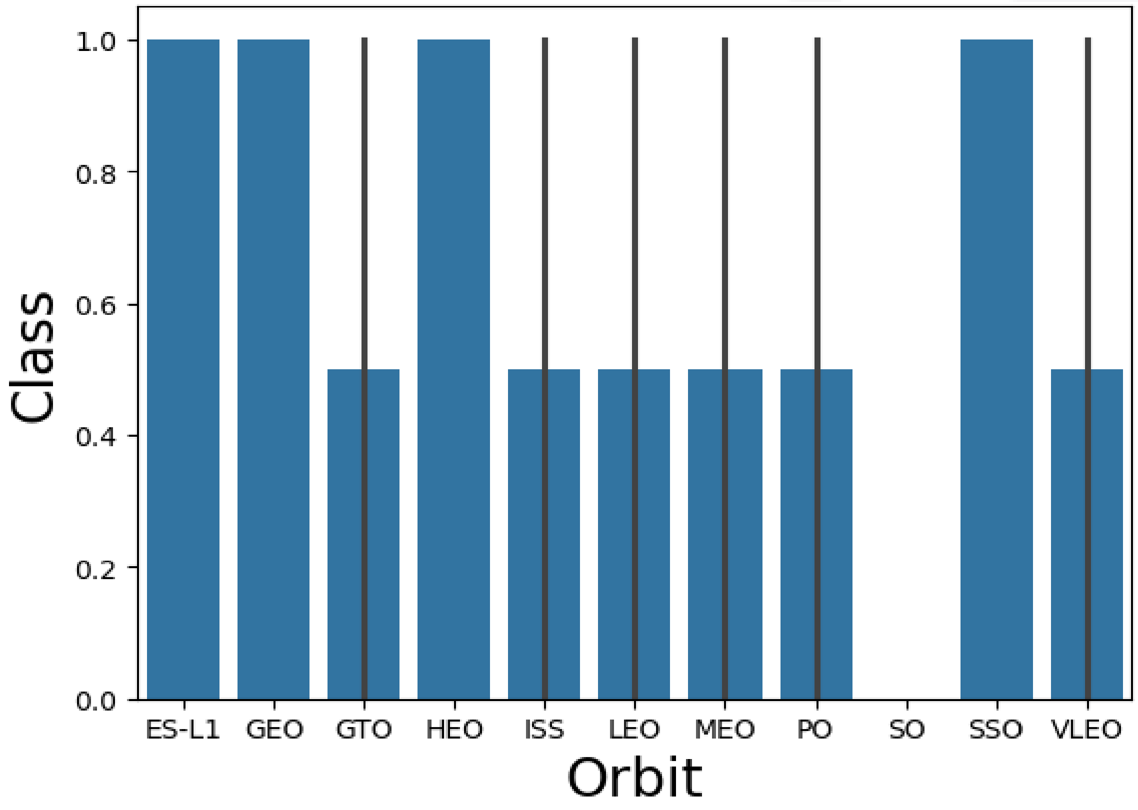
- **Exploratory Data Analysis**: **EDA with Data Visualisation:**



From the yearly success rate we see that the successful landings have been consistently increasing up to over 80% since 2017. We also observe brief periods of time (2017-2018 and 2019-2020) where the success rate slightly decreases.

DISCUSION

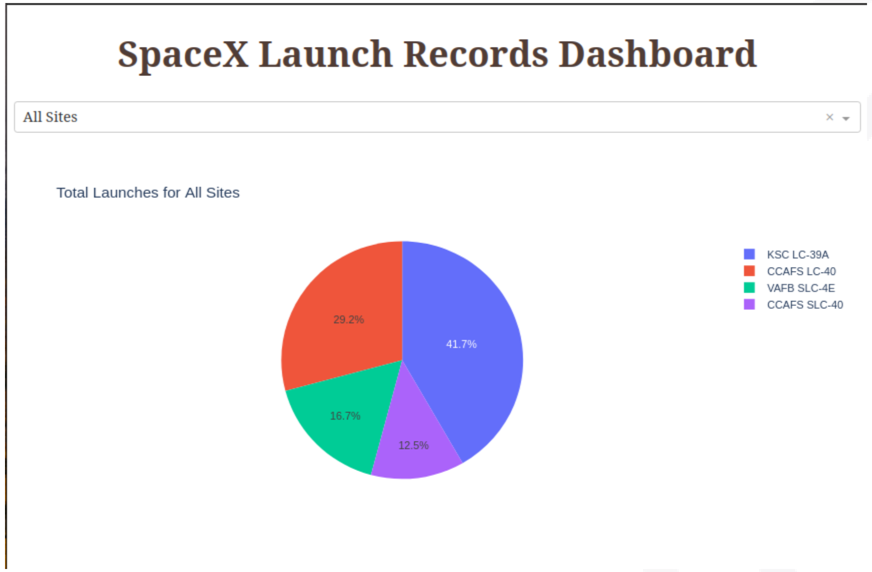
- **Exploratory Data Analysis**: **EDA with Data Visualisation:**



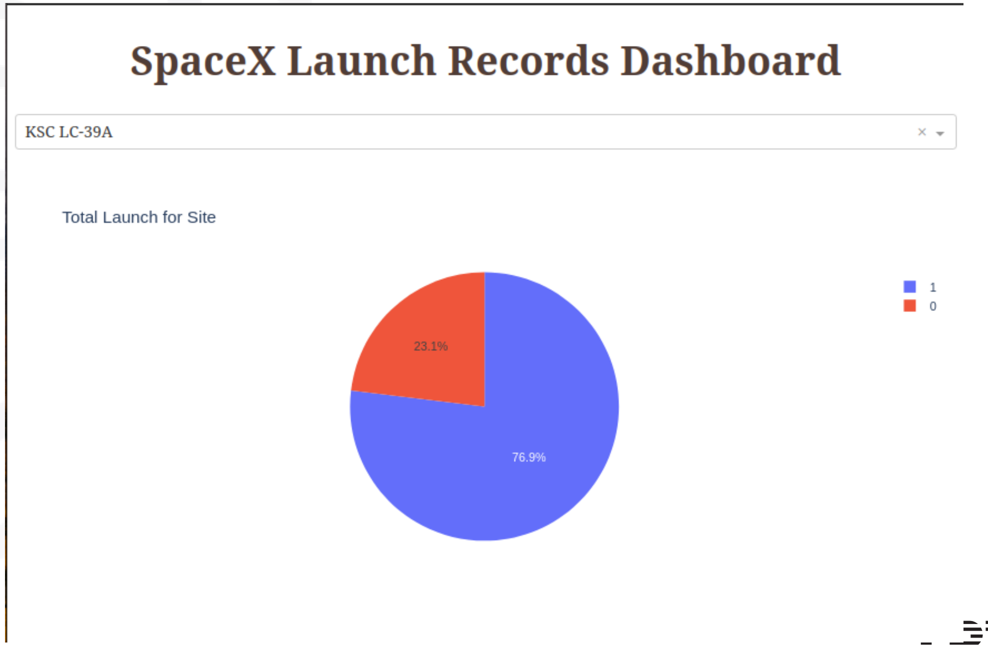
From the bar chart sucess rate vs orbit type we see that the higher rates happen for the orbits GEO, HEO, ES-L1, and SSO.

DISCUSION

- **<u>Exploratory Data Analysis</u>: <u>Creating an interactive dashboard with Ploty.</u>**



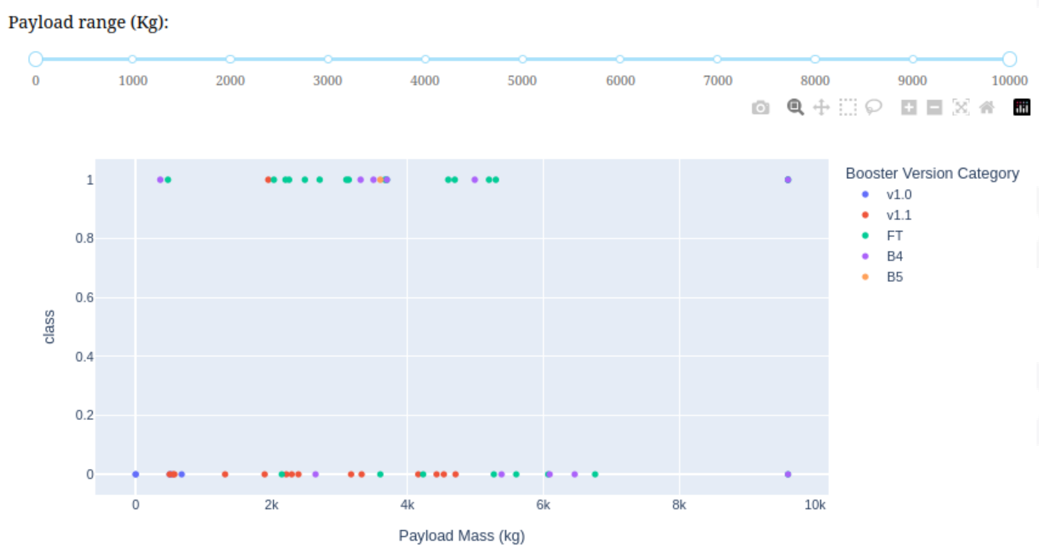
From the pie chart for all the launching sites we can see that KSC LC-39A has the higher amount of sucessful launches from the total number, with a 41.7%.



Looking into detail, KSC LC-39A has a rate of success of 76.9%.

DISCUSION

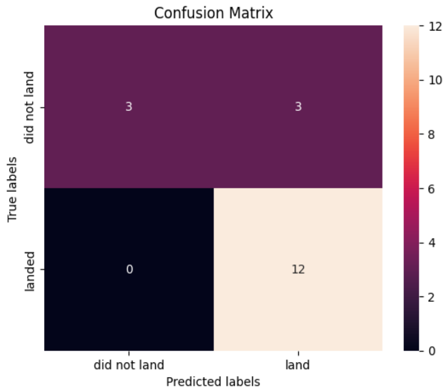
- **<u>Exploratory Data Analysis</u>**: **<u>Creating an interactive dashboard with Ploty.</u>**

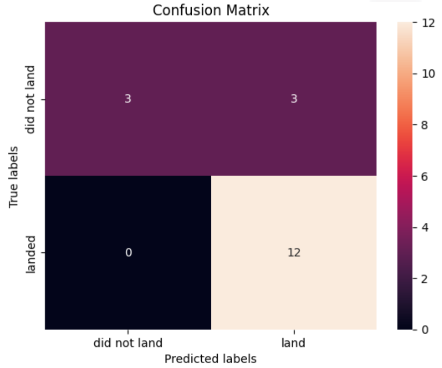


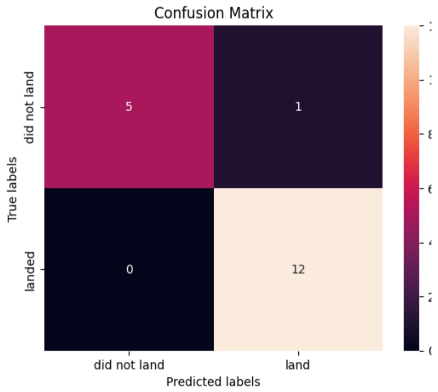
Attending to the payload mass, the higher number of successful landings happen in the ranges 2000 to 4000 kg and 4000 to 6000 kg. Concerning the booster version, FT is the one more effective.

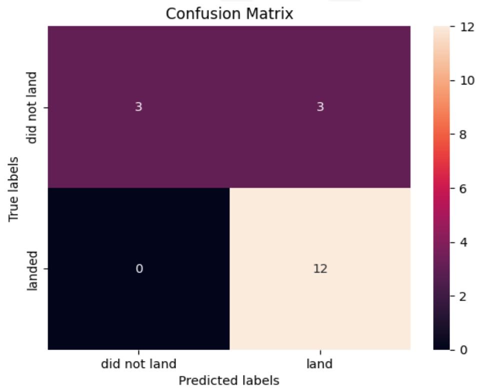
DISCUSION

- **Exploratory Data Analysis: Predictive analysis.**









From the confusion matrices we can state that all the models seem to work fairly well, with no false negative predictions. The decision tree seems to work specially well, exhibiting the same amount of true positive predictions as the rest of models and a lower amount of false positive. It also has a higher accuracy, 88.8%, in comparison with the other models, 83.3%.

CONCLUSIONS

- The launch site KSC LC-39A is the one with the highest number of successful landings over the years.
- The booster version FT has the highest rate of success in launches.
- GEO, HEO, SSO, and ES-L1 are the most successful orbits.
- As the number of flights increase, it is more likely the ship will safely land again.
- Missions with low payload mass have a better performance than the heavier ones.
- From all the considered models for a predictive analysis the most accurate is the decision tree (88% of accuracy).