
IBM
HERNÁNDEZ GARCÍA LUIS GERARDO
3rd June 2022
SPACEX
Data Science Capstone Project

Outline

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

Source: HYGGER website [https://hygger.io/blog/what-is-project-proposal-in-project-management/]

Source: IBM website [https://www.ibm.com/us-en/]

Executive Summary

The following Project has the objective to predict Falcon 9 first stage landing

- **METHODOLOGIES**
  - Data collection using API
  - Data wrangling
  - Exploratory Data Analysis using SQL
  - Interactive Maps applying Folium
  - Interactive Dashboard with Ploty
  - Predictive Analysis through classification
- **FINDINGS**
  - Exploratory data analysis results
  - Data visualizations
  - Best predictive model

Source: Vista Projects website
[https://www.vistaprojects.com/blog/how-to-write-an-effective-executive-summary/]

Introduction

Background and context

It is well known that SpaceX is one of the most prestigious private companies that is trying to get realized different travels to the space. For this project, **we have predicted if the Falcon 9 on its first stage will land successfully.**

According to SpaceX website, the company announces Falcon 9 rocket launches, with a cost of 62 million dollars whereas other providers cost approximately 165 million.

One of the aspects that makes SpaceX being successful is that they can *reuse the first stage*.

Objectives

We want to determine how successful is the first stage and the landing probabilities. In order to conduct the analysis the next questions need to be address

- *What factors influence to make a landing successful?*
- *What variables influences the most for the success rate?*
- *In order to have the best results and ensure a landing rate, what conditions are needed?*
  What is the best rocket to use for a successful landing?

METHODOLOGY

Methodology

The methodology used for this project is divided in five different stages which involves from compiling data and cleaning process to visualization and classification.

1) **Data collection:**
   - SpaceX Rest API
   - Web Scrapping using information from Wikipedia

2) **Data Wrangling:**
   - One hot encoding data fields for Machine Learning and Dropping irrelevant columns

3) **Exploratory Data Analysis (EDA) visualization with SQL:**
   - The use of diverse plots such as bar, scatter plots, etc., to reveal the variable's relationship and provide patterns on the data

4) **Interactive visual analytics:**
   - The help of Folium and Ploty to create maps and Dashboards

5) **Classification models:**
   - Build, train, perform and evaluate models such as Logistic Regression, SVM, Decision Tree and KNN models

Methodology – Data collection

Dataset was collected from the Space X REST API which covers different information such as flight number, date, payload mass, landing pad, among others.

The main goal by using this dataset is to **predict whether Space X will attempt to land a rocket**

In order to start to work with the dataset. Some steps are needed:

1) Getting Data from API or Website
2) Transform the data into a data frame
3) Filter the data frame as a pre-requisite to start to work with the valuable information
4) Export the final dataset.

In order to complete the four steps we will be using SpaceX API and Web Scrapping

Source: Azul School website
[https://www.azulschool.net/presentacion/curso-web-scraping/

Methodology – Data collection – SpaceX API

Response from API

Some requirements were installed previously

```
In [6]: spacex_url="https://api.spacexdata.com/v4/launches/past"
In [7]: response = requests.get(spacex_url)
```

Converting response to a .jsonfile and normalize

[GitHub]

```
In [9]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
In [15]: # Use json_normalize meethod to convert the json result into a dataframe
response = requests.get(static_json_url)
data = pd.json_normalize(response.json())
```

Apply custom functions and cleaning process

we can apply the rest of the functions here:

```
In [23]: # Call getLaunchSite
getLaunchSite(data)
In [24]: # Call getPayloadData
getPayloadData(data)
In [25]: # Call getCoreData
getCoreData(data)
```

Assign list to a dictionary and create a new frame

Finally lets construct our dataset using the data we have obtained. We we combine the columns into a dictionary.

```
In [26]: 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.

```
In [27]: # Create a data from launch_dict
df = pd.DataFrame(launch_dict)
```

FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	Reus
4	1	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0
5	2	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0
6	3	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0
7	4	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None	1.0
8	5	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False	None	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
89	86	2020-09-03	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	2	True	True	True	5e9e3032383ecb6bb234e7ca	5.0
90	87	2020-10-06	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	3	True	True	True	5e9e3032383ecb6bb234e7ca	5.0
91	88	2020-10-18	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	6	True	True	True	5e9e3032383ecb6bb234e7ca	5.0
92	89	2020-10-24	Falcon 9	15600.0	VLEO	CCSFS SLC 40	True ASDS	3	True	True	True	5e9e3033383ecbb9e534e7cc	5.0
93	90	2020-11-05	Falcon 9	3681.0	MEO	CCSFS SLC 40	True ASDS	1	True	False	True	5e9e3032383ecb6bb234e7ca	5.0

Filter the data frame and export to flat file

```
In [29]: # Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = df
data_falcon9.drop(data_falcon9[data_falcon9['BoosterVersion']!='Falcon 9'].index, inplace = True)
```

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

```
In [30]: data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
In [40]: data_falcon9.to_csv('dataset_part_1.csv', index= False)
```

Methodology – Data collection – Web Scrapping

1) Response from API

```
In [4]: static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

```
In [5]: # use requests.get() method with the provided static_url
        # assign the response to a object
        data = requests.get(static_url).text
```

2) Creating BeautifulSoup Object

```
In [6]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
        soup = BeautifulSoup(data, 'html5lib')
```

3) Finding Tables

```
In [8]: # 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')
        html_tables
```

4) Column names

```
In [10]: column_names = []

         # Apply find_all() function with `th` element on first_launch_table
         # Iterate each th element and apply the provided extract_column_from_header() to get a column name
         # Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
         ths = first_launch_table.find_all('th')
         for th in ths:
             name = extract_column_from_header(th)
             if name is not None and len(name) > 0:
                 column_names.append(name)
```

5) Creation of dictionary and appending data

Refer to the notebook to see the whole code for appending

```
In [24]: 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']=[]
```

```
In [25]: extracted_row = 0
         #Extract each table
         for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
             # get table row
             for rows in table.find_all("tr"):
                 #check to see if first table heading is as number corresponding to launch a number
                 if rows.th:
                     if rows.th.string:
                         flight_number=rows.th.string.strip()
                         flag=flight_number.isdigit()
                 else:
                     flag=False
```

6) Converting dictionary to data frame

```
In [26]: df=pd.DataFrame(launch_dict)
```

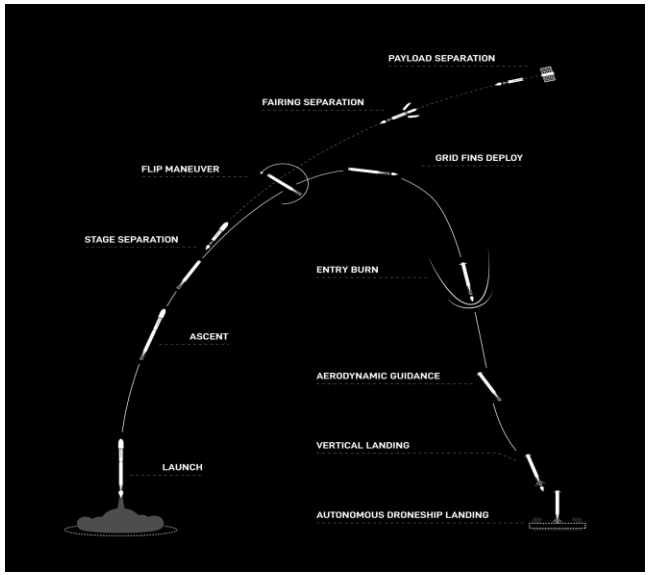
7) Saving data frame

```
In [29]: df.to_csv('spacex_web_scraped.csv', index=False)
```

GitHub

Methodology – Data Wrangling – What it is and its process?

Data wrangling **involves processing the data in various formats** like - *merging, grouping, concatenating* etc. for the purpose of **analyzing or getting them ready to be used** with another set of data. Python has built-in features to apply these wrangling methods to various data sets to achieve the analytical goal (…)
[Tutorials point, https://www.tutorialspoint.com/python_data_science/python_data_wrangling.htm]



Source: SpaceX website
https://www.spacex.com/mission/

Before start predictions, we need to split the training labels, therefore the following dummy for classes has to be applied

1 means the booster successfully landed

0 means the landing fail

PROCESS

Dataset → Transform data frame → Cleaning process → Change to Boolean → Export file → Start EDA anaylis

Methodology – Data Wrangling – needed steps

Before to start to Exploratory Data Analysis we will need to find some patterns in the data to label them for the training

1) Calculate launches

```
In [5]: # Apply value_counts() on column LaunchSite
        df['LaunchSite'].value_counts()

Out[5]: CCAFS SLC 40    55
        KSC LC 39A      22
        VAFB SLC 4E     13
        Name: LaunchSite, dtype: int64
```

2) Calculate number and occurrence on different orbits

```
In [6]: # Apply value_counts on Orbit column
        df['Orbit'].value_counts()

Out[6]: 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 occurrence of mission per orbit type

```
In [7]: # landing_outcomes = values on Outcome column
        landing_outcomes = df['Outcome'].value_counts()
        landing_outcomes

Out[7]: True ASDS      41
        None None      19
        True RTLS      14
        False ASDS      6
        True Ocean      5
        False Ocean     2
        None ASDS       2
        False RTLS      1
        Name: Outcome, dtype: int64
```

4) Landing outcome label from outcome column

```
In [10]: # 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)
         landing_class
```

5) Create and save the data frame

```
In [11]: df['Class']=landing_class
         df[['Class']].head(8)

Out[11]:
```

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

```
In [12]: df.head(5)

Out[12]:
```

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

```
In [14]: df.to_csv("dataset_part_1.csv", index=False)
```

[GitHub]

Methodology – Exploratory Data Analysis (EDA)

Part 1 – Data Visualization

We use the Exploratory Data Analysis to summarize the main characteristics of our data set and to find different patterns among data. In order to summarize this characteristics, it is necessary to use statistical tools such as graphs.

Source: Boostlabs
https://boostlabs.com/blog/10-types-of-data-visualization-tools/

Relationships between using Scatter plots:
- Flight number and Payload Mass
- Flight number and Launch Site
- Payload and Launch Site
- Orbit and Flight number
- Payload and Orbit
- Orbit and Payload Mass

Patterns between groups using Bar plots
- Orbits and the mean

Possibilities of success rate using Line plots
- Success rate by year

GitHub

Methodology – Exploratory Data Analysis (EDA)

Part 2 – Using SQL

For this second part we will use relational data bases in **SQL through the IBM´s Db2** service for the cloud in order to perform different questions that would help us to have better insights.

Source: db2 tutorial
https://www.db2tutorial.com/

Understanding our variables

- Display names of unique launch sites in the space mission
- Show five records on the table where the launch site start with CCA
- Total Payload Mass carried by boosters launches by NASA (CRS)
- Calculate and show the average Payload Mass carried by booster version F9 v1.1

Listing some successful landings and Payload mass

- Date when the first successful landing outcome in ground pad was achieved
- Boosters which have success in drone ship and have payload mass greater than 4,000 but less than 6,000
- Number of successful and failure mission outcomes
- Names of Booster version which have carried the maximum payload mass
- Failed landing outcomes in drone ship, their booster versions, and launch site names for only year 2015
- Ranking the count of successful landing outcomes between 2010-06-04 and 2017-03-20, by descending order

Methodology – Interactive visual analytics

Part 1 – Interactive Maps with Folium

The main objective is to have an interactive map that shows each launch site adding a circle marker and label for the launch site. By using folium we can visualize easily the successful and failure launch sites by colors.

Source: Youtube
https://www.youtube.com/watch?v=QpBmO35pmVE

Create objects for:

- Make a mark on the map by creating map marker
- Create icons on the map by creating icon marker
- Showing markers that have been selected by creating circle with circle marker
- Draw lines between points by using polyline
- Create clusters to group different objects that tend to be located close each other by using maker cluster object
- Draw different line points or arrows between line points by using ant path

This would help us to understand if launch sites are close each other, if there is a good access to the launch sites, and detect if there is cluster where the success launch rate is high and how many launches are located in that site.

GitHub

Methodology – Interactive visual analytics

Part 2 – Interactive Dashboard

The main objective is to have an interactive dashboard with two different plots, a pie chart and a scatter plot. The former will show the percentage of success in relation with the launch sites while the later the relationship between the success rate and the booster version category

Source: Train Test Split website
https://traintestsplit.com/interactive-data-visualization-in-python-a-plotly-and-dash-intro/

Create objects for:

- High level components for sliders, graphs, dropdowns, tables by creating dash and its components
- Fetch values from dataset in the CVS format and create a data frame
- Plot graphs interactively using plotly
- Launch sites by using dropdown
- Payload Mass range selection in a range slider form using Rangeslider
- Pie charts and Scatter plots

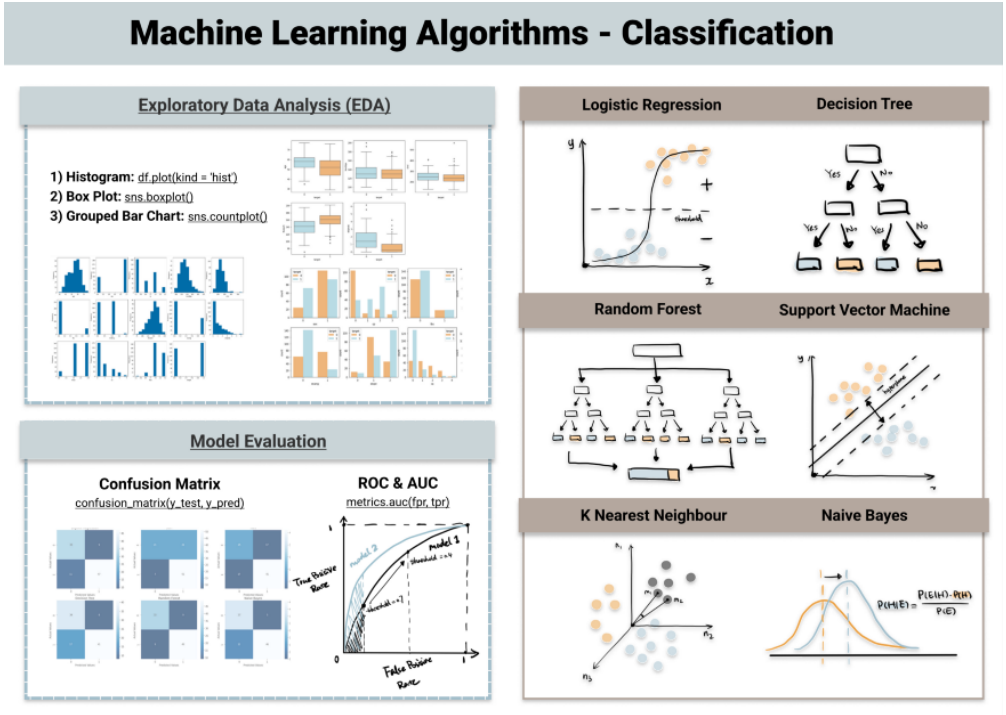
This would help us to understand and interact with the data by using visualization tools, and would lead us to start to guess which launch site has more success rate and its possible influence for the predictive models.

GitHub

Interactive web site

Methodology – Classification models:

We are aiming to build, evaluate, improve and predict the model that best performs the success rate for landing.



Source: Towards Data Science
https://towardsdatascience.com/top-machine-learning-algorithms-for-classification-2197870ff501

1) Building the model

- Load dataset into Numpy and Pandas
- Transform data
- Split data into training and test
- Check test samples
- Select the type of machine learning algorithm
- Set parameters and algorithms to the GridSearchCV
- Fit the data set into the GridSearchCV objects and train the data set

2) Evaluating Model

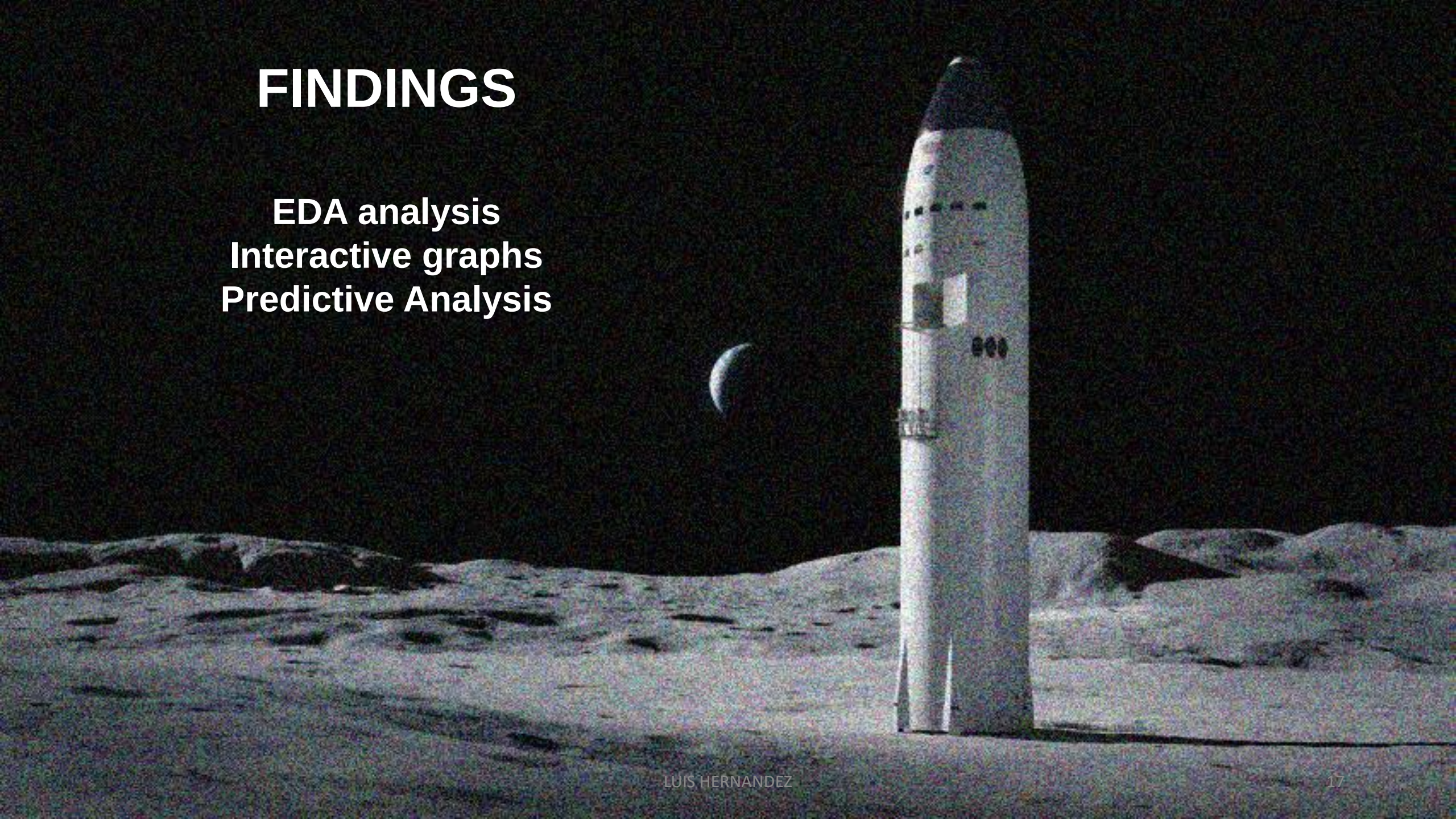
- Check accuracy for each model
- Get best hyperparameters for each type of algorithms
- Plot Confusion Matrix

3) Finding the best performing classification model

- Model with best accuracy score is the best
- Comparison of model performance accuracy

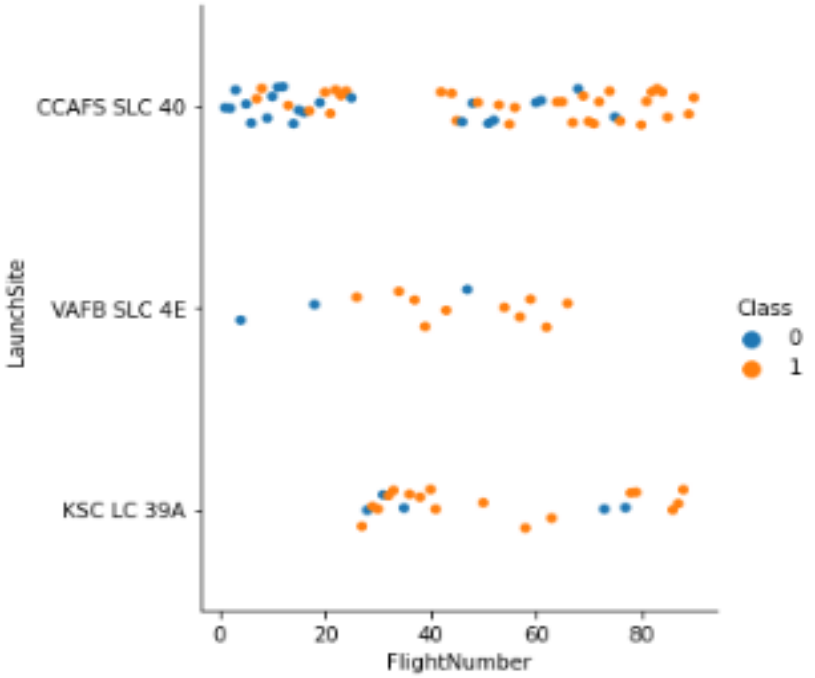
FINDINGS

EDA analysis
Interactive graphs
Predictive Analysis



Exploratory Data Analysis with Python

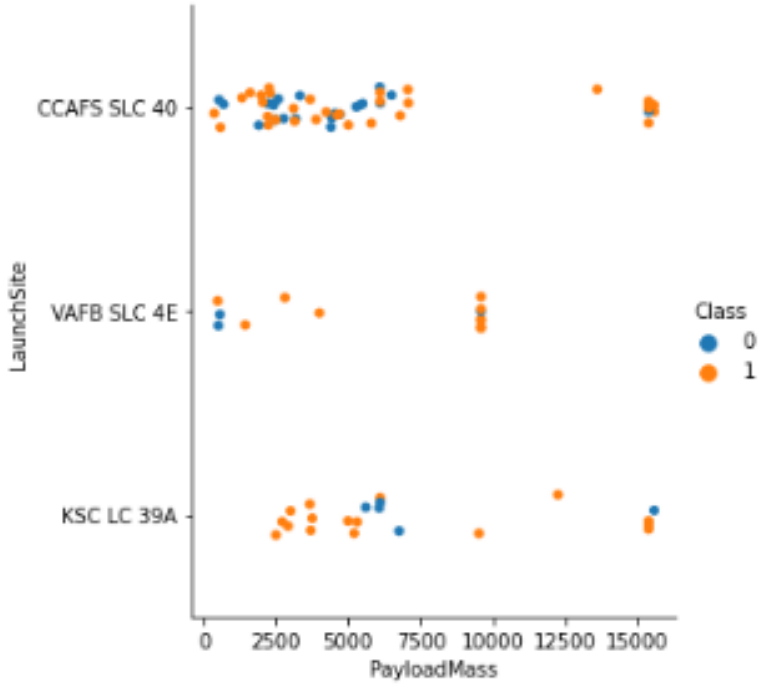
Results – Flight Number and Launch Site:



Source: Own elaboration

1) The more the flights occur, the greater the success rate increase
2) This flight numbers tend to be higher than 30
3) Even CCAFS SLC has more flight numbers, the less than 30 they are, its success rate become lower. On the other hand, VAFB SLC flights are less than the previous one, but it success rate use to happen more frequently.
4) KSC KC tend to have a greater successful rate related to the total number of flights

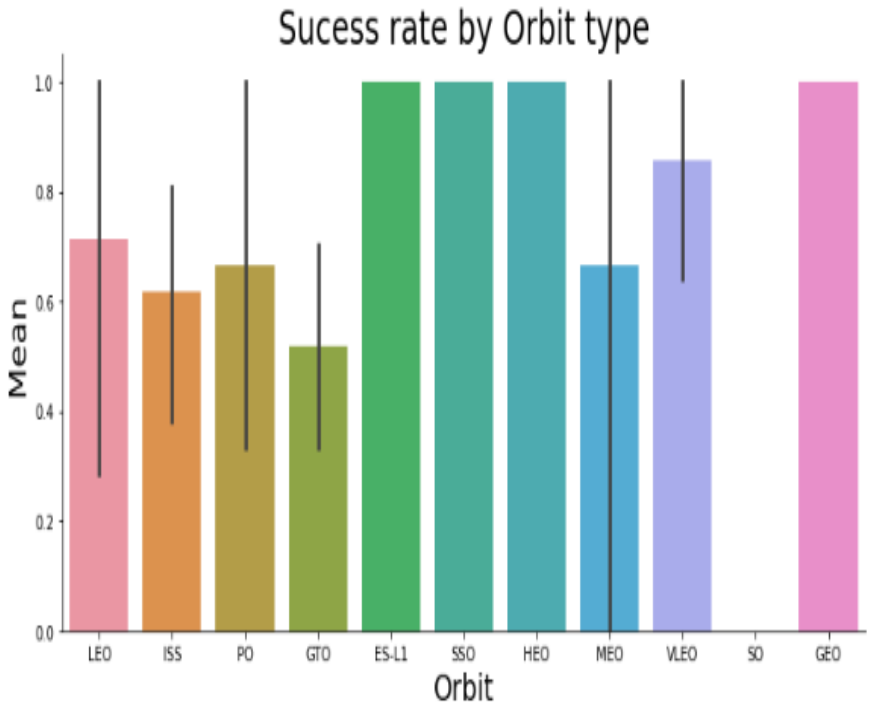
Results – Payload Mass and Launch Site:



Source: Own elaboration

1) There is not a clear evidence that lightly or heavily Pay Load Mass lead to a success/fail on the mission;
2) However, for the case of CCAFS SLC 40, almost all the heavier Payload Mass lead to a success (class 1), contrary to the case of KSC LC39A where lightly Payload Mass has a higher success rate.
3) Therefore, we can not drive a early conclusion by looking at this graph.

Results – Success rate and Orbit type:

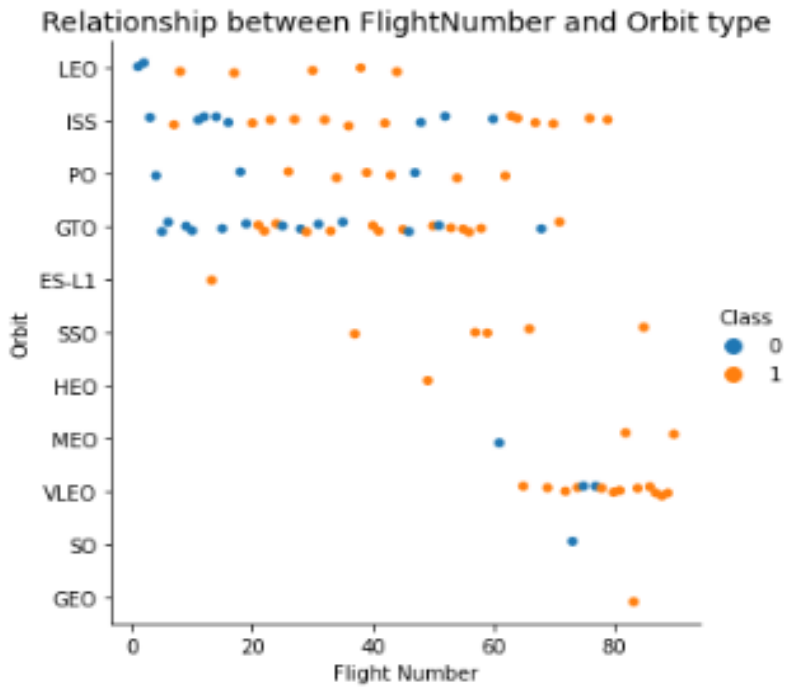


Source: Own elaboration

1) Orbits with the higher success rate are:
   - ES-L1
   - SSO
   - HEO
   - GEO

2) Orbits with the fewer success rate are:
   - SO (null)
   - GTO
   - ISS

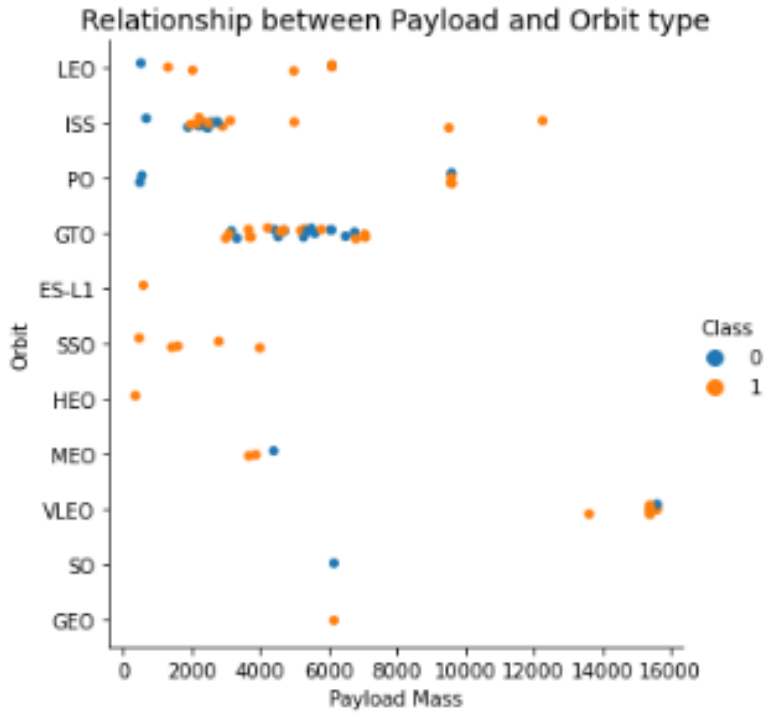
Results – Flight Number vs Orbit type:



Source: Own elaboration

1) The graph suggests that for the LEO orbit, the higher the flight number, the more success would happened
2) There is not enough evidence for ES-L1 and GEO orbit to drive a strong conclusion as observations are few
3) SSO has all success rate with middle term number of flights
4) GTO orbit does not present a pattern as we have mix results
5) VLEO is the orbit with more observations related with a higher flight number and most of them lead to a success which suggest could alongside with the LEO orbit, the ones to be used for the launching process.

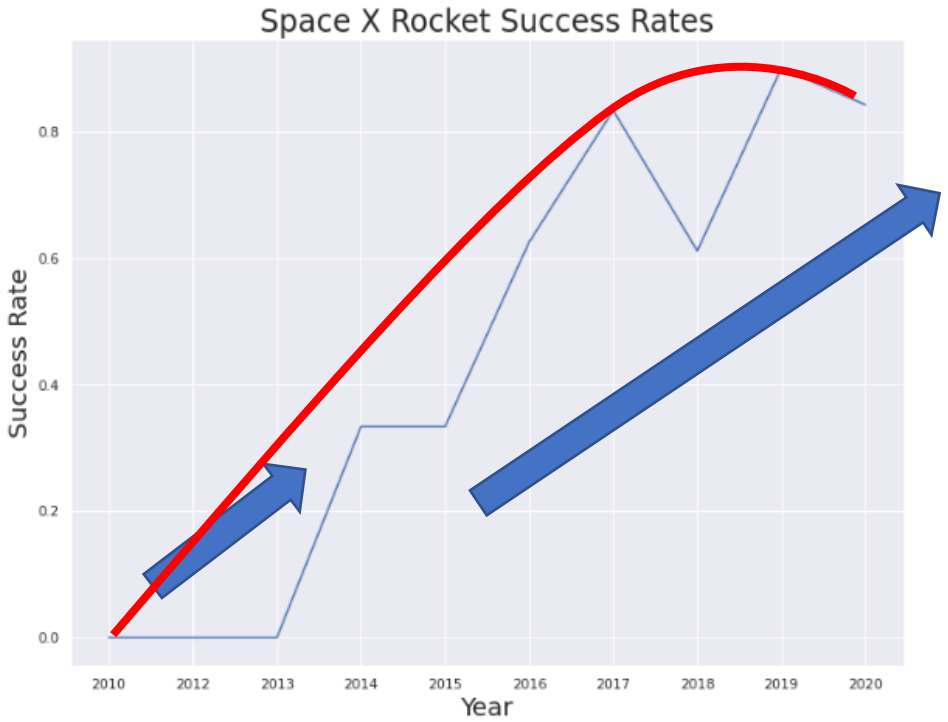
Results – Payload Mass vs Orbit type:



Source: Own elaboration

1) There are mixed results for GTO, however, on its observations, the heavier ones tend to a fail on the mission
2) There is a positive correlation on LEO, SSO and ISS orbits due to most of its observations tends to success and also, the heavier the Payload it is, the fail tendency tends to disappear
3) Few observations for GEO and SO orbits, which use this observations for predictions would lead us to a higher risk.
4) VLEO has the heavier payloads, whereas the top one lead to a fail the rest has a success rate.

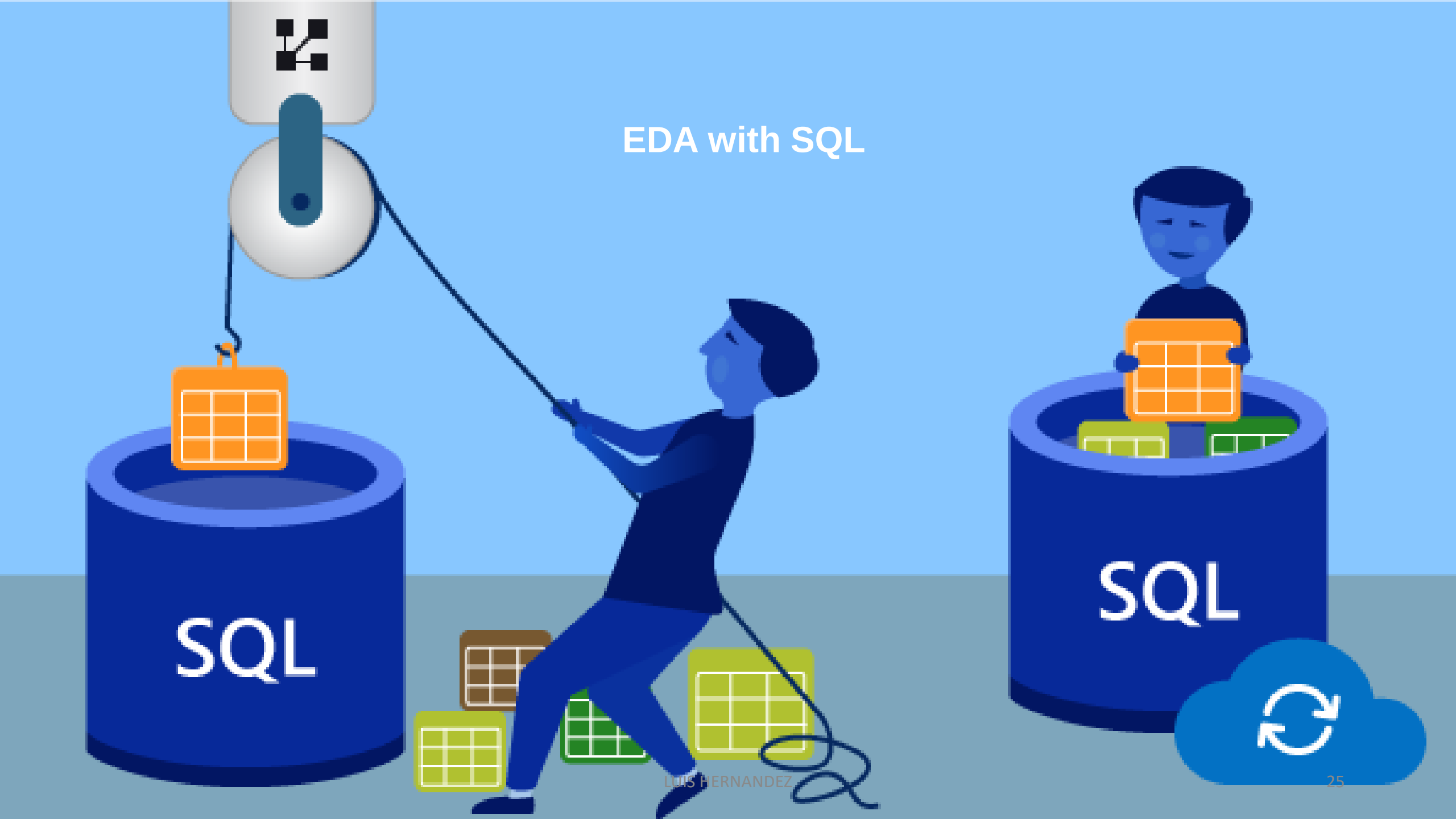
Results – Launch Success Yearly Trend:



Source: Own elaboration

1) **Observations can be split in three different periods:**
    1) A slowly increase on the success (training process) which covers from the year 2000 to 2015 and;
    2) A rapid increase on the success (better performance) that runs from 2015 to 2020
    3) A mature process that seems to run from 2019 to present, as the success rate maintains on the range of 80%.
2) **In general, we can observe that the success rate has been increasing since its launch process and now is entering in a flat process (mature stage)**

EDA with SQL



Results – Launch Site names:

USED COMMAND

```
In [6]: %sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXDATASET;
```

OUTPUT

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

OBJECTIVE

By using the command Distinct in the query, we will obtain all unique values for the Launch Site (used variable) from the SpaceXDataset table.

The results allows us to identify the unique values of the Launch Sites for easier identification in future queries, in this case there are 4 different outputs

Results – Launch Site names which starts with ‘CCA’:

USED COMMAND

```
In [8]: %sql SELECT * FROM SPACEXDATASET WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

OUTPUT

DATE	time_utc_	booster_version	launch_site	payload	payload_mass_kg_	orbit	customer	mission_outcome	landing_outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

OBJECTIVE

The objective of the query is to identify all launch site names from the table SpaceXdataset that must start with CCA by using the command LIKE and inside the ‘ ‘ include a % sign to say that may be some letters after the CCA. In addition the LIMIT 5 command makes that the output only provides 5 records.

Results – Total Payload Mass:

USED COMMAND

```
In [9]: %sql SELECT SUM(PAYLOAD_MASS__KG_) AS "TOTAL PAYLOAD MASS BY NASA (CRS)" FROM SPACEXDATASET WHERE CUSTOMER = 'NASA (CRS)';
```

OUTPUT

TOTAL PAYLOAD MASS BY NASA (CRS)
45596

OBJECTIVE

We want to identify the total mass of payload on the data set, therefore we need to use the command SUM and select the variable. As we want to know only the Customer NASA CRS, we should set this value in the WHERE clause.

Results – Average Payload Mass carried by booster F9 v1.1

USED COMMAND

```
In [11]: %sql SELECT AVG(PAYLOAD_MASS__KG_) AS "AVERAGE PAYLOAD MASS CARRIED BY BOOSTER VERSION F9 v1.1" FROM SPACEXDATASET \
         WHERE BOOSTER_VERSION = 'F9 v1.1'
```

OUTPUT

AVERAGE PAYLOAD MASS CARRIED BY BOOSTER VERSION F9 v1.1
2928

OBJECTIVE

We want to identify the average mass of payload on the data set, therefore we need to use the command AVG and select the variable. As we want to know only the average of Booster version V1.1, we should set this value in the WHERE clause.

Results – Date where the successful landing outcome in ground pad was achieved

USED COMMAND

```
In [14]: %sql SELECT MIN(DATE) AS "First Succesful Landing Outcome in Ground Pad" FROM SPACEXDATASET \
         WHERE LANDING__OUTCOME = 'Success (ground pad)';
```

OUTPUT

First Succesful Landing Outcome in Ground Pad
2015-12-22

OBJECTIVE

We want to identify the first date of successful landing in ground pad, therefore we need to use the command MIN and select the variable DATE. As we want to know only the date of Ground pad, we should set this value in the WHERE clause.

Results – Successful drone ship landing with payload mass between 4000 and 6000

USED COMMAND

```
In [16]: %sql SELECT BOOSTER_VERSION FROM SPACEXDATASET WHERE LANDING__OUTCOME = 'Success (drone ship)' \
         AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```

OUTPUT

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

OBJECTIVE

We want to identify the booster on the data set than have a payload mass between 4000 and 6000, therefore we need to use the command SELECT booster version from our table, and in the filtering, the WHERE clause, specify that payload mass is between 4000 and 6000.

Results – Total number of Successful and Failure Mission

USED COMMAND

```
In [19]: %sql SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS "SUCCESSFULL MISSION" FROM SPACEXDATASET GROUP BY MISSION_OUTCOME;
```

OUTPUT

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

OBJECTIVE

We want to identify the total failure and success mission outcomes. An easy way to get it, is using the MISSION OUTCOME and COUNT its values. In order to have all the outputs, we will use the GROUP BY clause to count the values. In total we have 101 values in 3 different outcomes.

Results – Boosters carried maximum payload

USED COMMAND

```
In [22]: %sql SELECT DISTINCT BOOSTER_VERSION AS "BOOSTER VERSIONS WHICH HAVE CARRIED THE MAXIMUM PAYLOAD MASS" \
         FROM SPACEXDATASET WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXDATASET);
```

OUTPUT

BOOSTER VERSIONS WHICH HAVE CARRIED THE MAXIMUM PAYLOAD MASS
F9 B5 B1048.4
F9 B5 B1048.5
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1049.7
F9 B5 B1051.3
F9 B5 B1051.4
F9 B5 B1051.6
F9 B5 B1056.4
F9 B5 B1058.3
F9 B5 B1060.2
F9 B5 B1060.3

OBJECTIVE

We want to identify the boosters that have carried the maximum payload mass, therefore we need to use a DISTINC clause to avoid duplicates and in the WHERE clause we will use a *subquery* to filter the payload mass in its maximum weight

Results – 2015 Failure Launch Records and its location names

USED COMMAND

```
In [27]: %sql SELECT LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXDATASET WHERE YEAR(DATE) = 2015 \
         AND LANDING__OUTCOME = 'Failure (drone ship)';
```

OUTPUT

landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

OBJECTIVE

We want to identify the launch records in 2015 and its location names that failed. In order to display the records we will need to SELECT the variables we want to show, and in the WHERE clause filter the 2015 year and failure drone ship.

Results – Rank Landing Outcomes between 2010-06-04 and 2017-03-20

USED COMMAND

```
In [28]: %sql SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) FROM (SELECT * FROM SPACEXDATASET WHERE DATE BETWEEN '2010-06-04' AND '201
         7-03-20') \
         GROUP BY LANDING__OUTCOME ORDER BY COUNT(LANDING__OUTCOME) DESC;
```

OUTPUT

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

OBJECTIVE

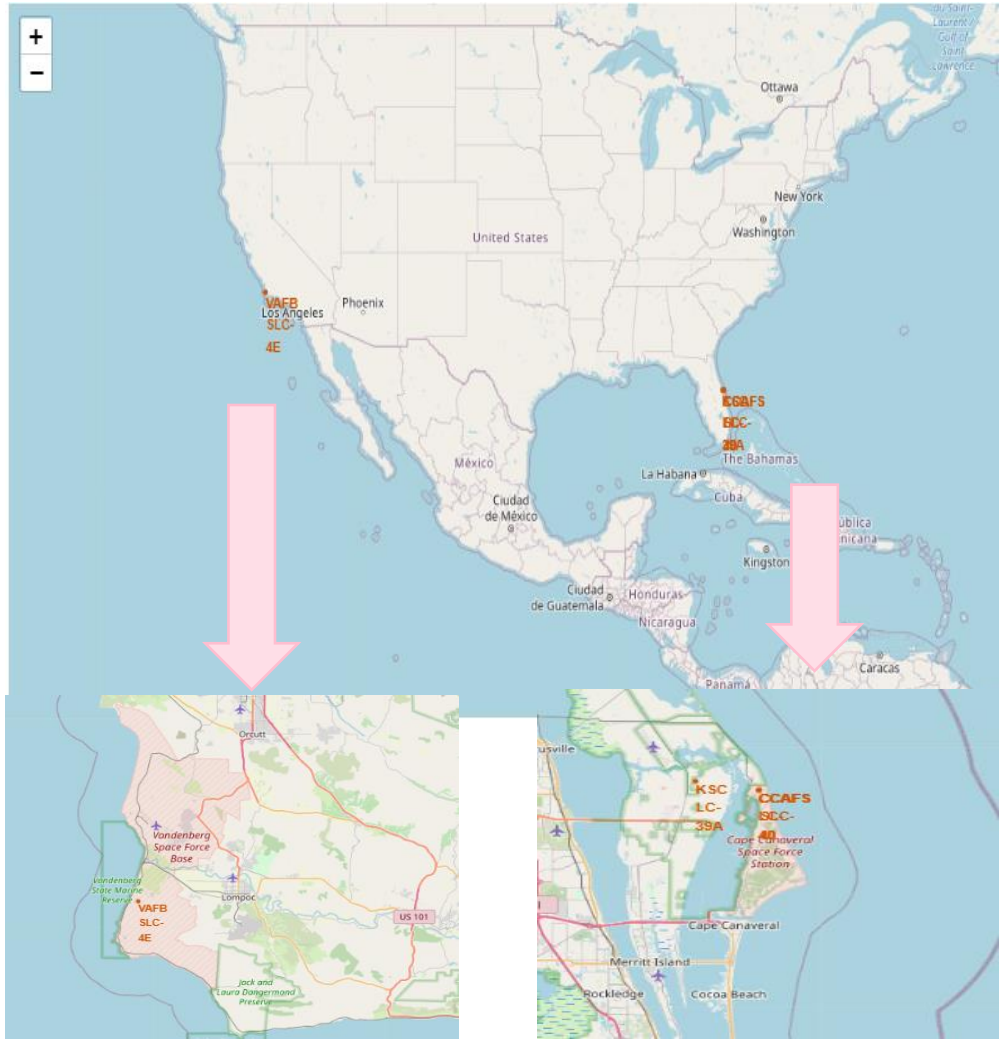
We need to SELECT the landing outcome and with the function COUNT we will count the records. We need a subquery in the clause FROM to select specific dates, in this case between 2010-06-04 and 2017-03-20. In addition, we need to GROUP BY the landing outcome and as we aim to RANK the counts, we need to add the DESC to order by descending order.

Interactive maps

with Folium

Results – All launches Sites on Folium Map:

We are aiming to have interactive maps for the launches sites for a better visualization analysis.



Source: Own elaboration

```
In [10]: # Initial the map
site_map = folium.Map(location=nasa_coordinate, zoom_start=5)
# For each launch site, add a Circle object based on its coordinate (Lat, Long) values. In addition, add Launch site name as a p
opup label
for index, site in launch_sites_df.iterrows():
    circle = folium.Circle([site['Lat'], site['Long']], color='#d35400', radius=50, fill=True).add_child(folium.Popup(site['Laun
ch Site']))
    marker = folium.Marker(
    [site['Lat'], site['Long']],
    # Create an icon as a text label
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % site['Launch Site'],
        )
    )
    site_map.add_child(circle)
    site_map.add_child(marker)

site map
```

Our task is to **mark all launch sites on the map** by making the reference of the latitude and longitude of the places.

As a result, we can see that **SpaceX launches sites are located on the United States' coast**, in Florida and California.

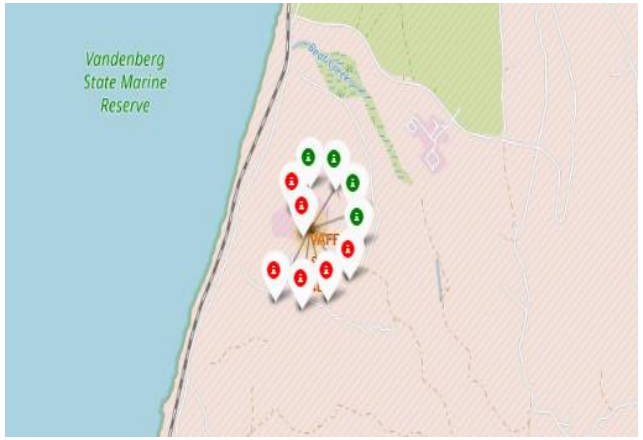
The code shown above reflects the final code to display the locations on the map

Results – Mark the success/failed launches for each site on the map and create colors for the markers:

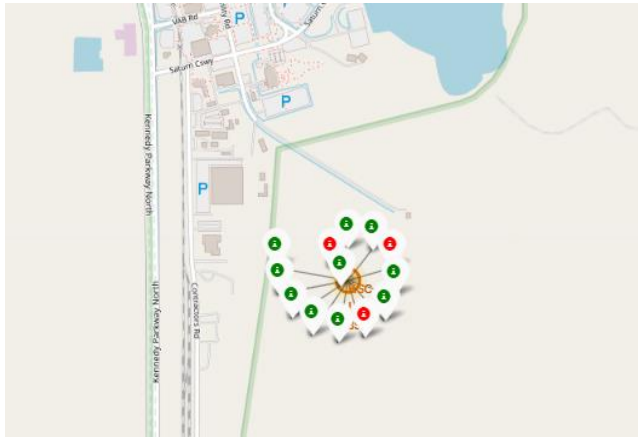
Source: Own elaboration

The **GREEN MARKER** shows the successful launches whereas the **RED MARKER** shows the failures ones.

The KSC location (the middle screen shot) provides valuable information, as we can see just 3 failures and the rest are success.



Source: Own elaboration



Source: Own elaboration

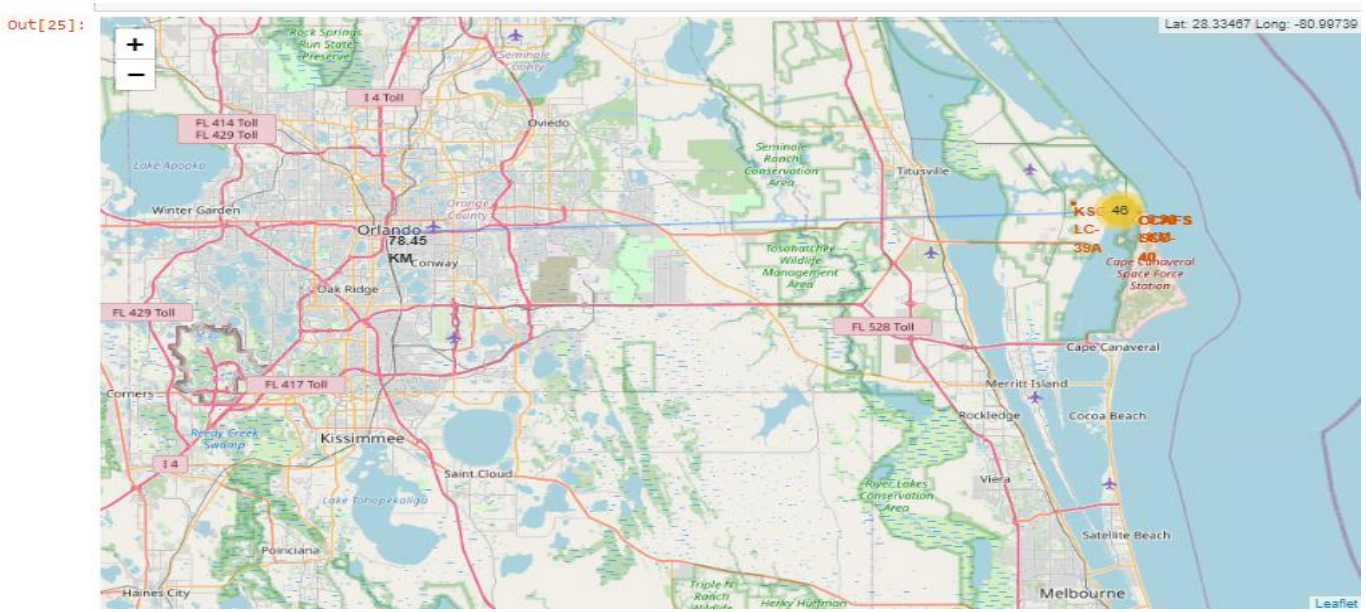
Source: Own elaboration

Results – Launch Site Distances on the locations:

Source: Own elaboration

The objective is to show the distance that exists between the cost and the location site, for references we use the CCAFS and shows that there exist a total of 0.90km between the coast and the site.

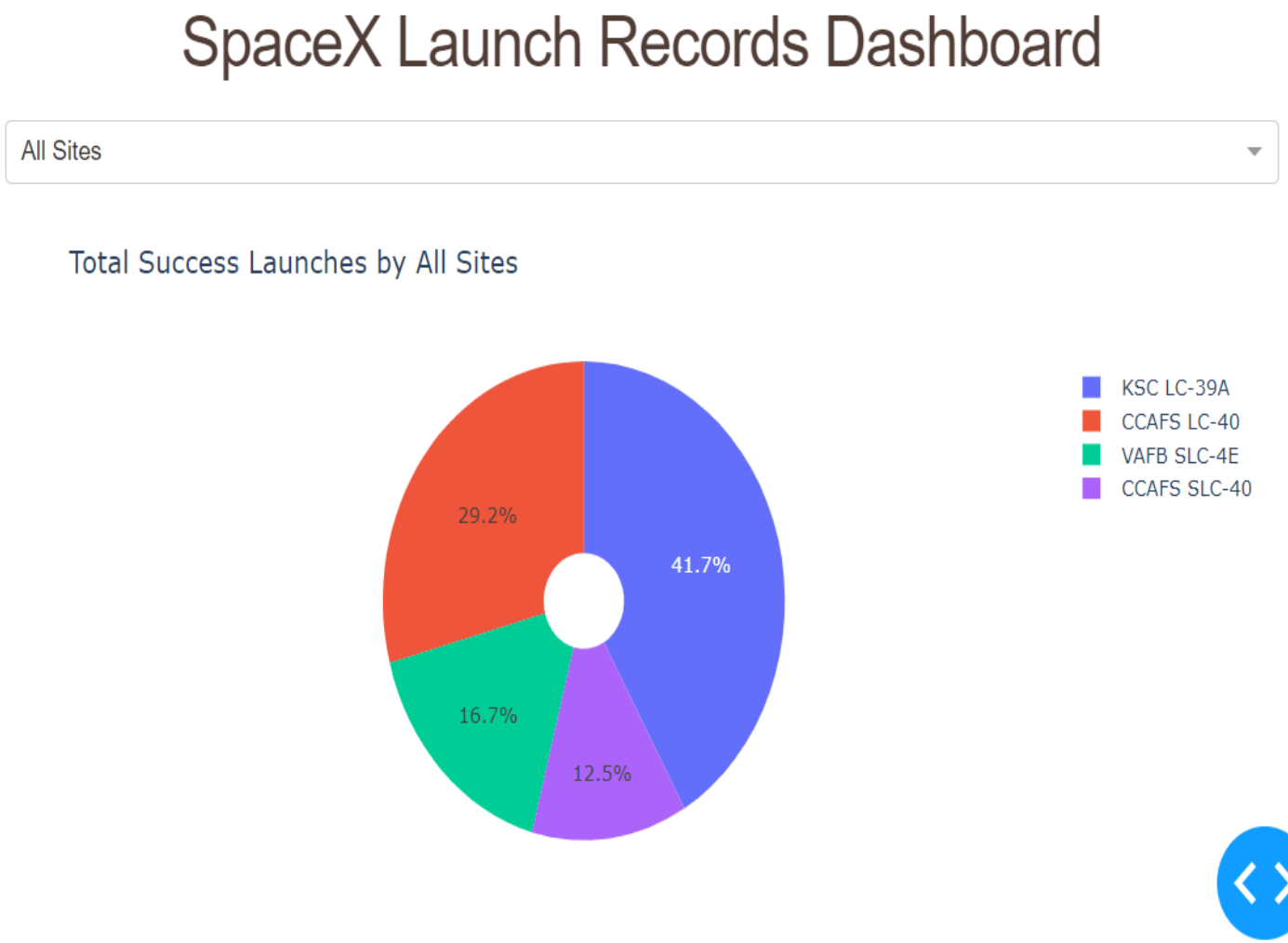
This helps to understand how accessible are the locations by the coast, road site, etc. The second picture shows the distance between the Orlando city.



Source: Own elaboration

BULDING A DASHBOARD WITH PLOTY DASH

Results – Launch Success Count for All Sites:

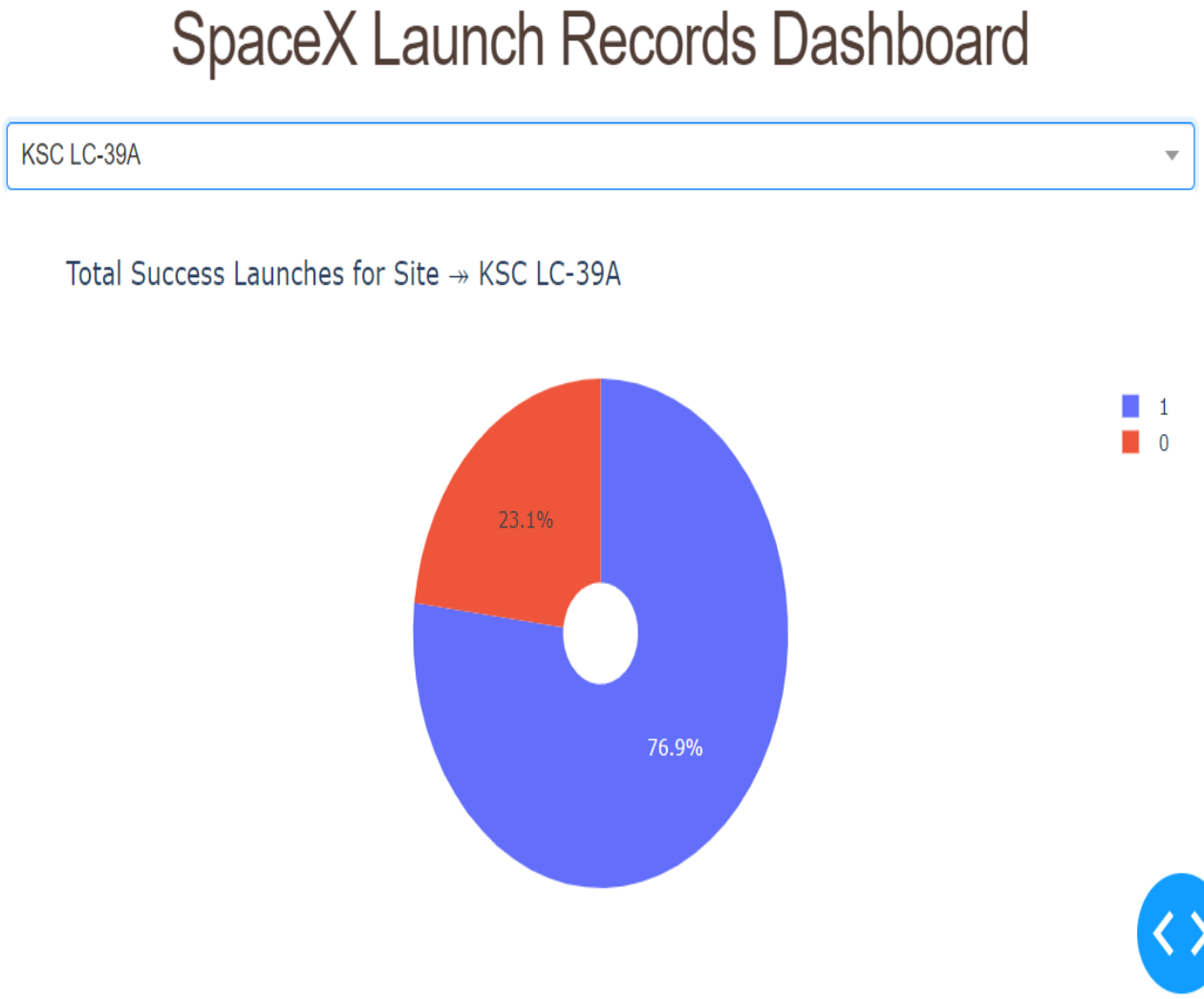


From all launches sites KSC LC-39A is the one with the most successful launches, counting for 41.7%.

It is followed by CCAFS LC-40 with 29.2%, VAFB SLC-4E and CCAFS SLC-40 with 16.7% and 12.5% respectively.

Interactive web site

Results – Pie Chart for the launch site with highest launch success ratio:



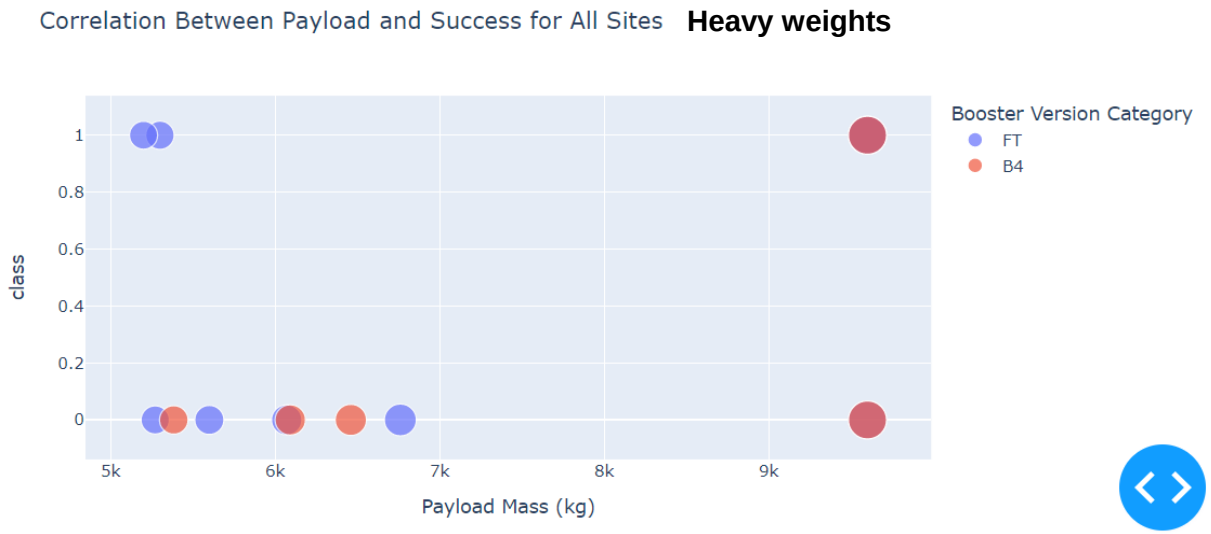
Inside the observations of KSC LC-39 A we concluded that:

1) Its success rate is three times greater than the failure, that is, it has a rate of 77% approximately.

2) That from the 41.7% of the total launches, the corresponding rate of total success is 32% (that is the 76.9% of the 41.7%)

Interactive web site

Results – Payload vs Launch Outcome All sites:



Our first insight is that, low payload mass have a greater success rate than the heavy ones.

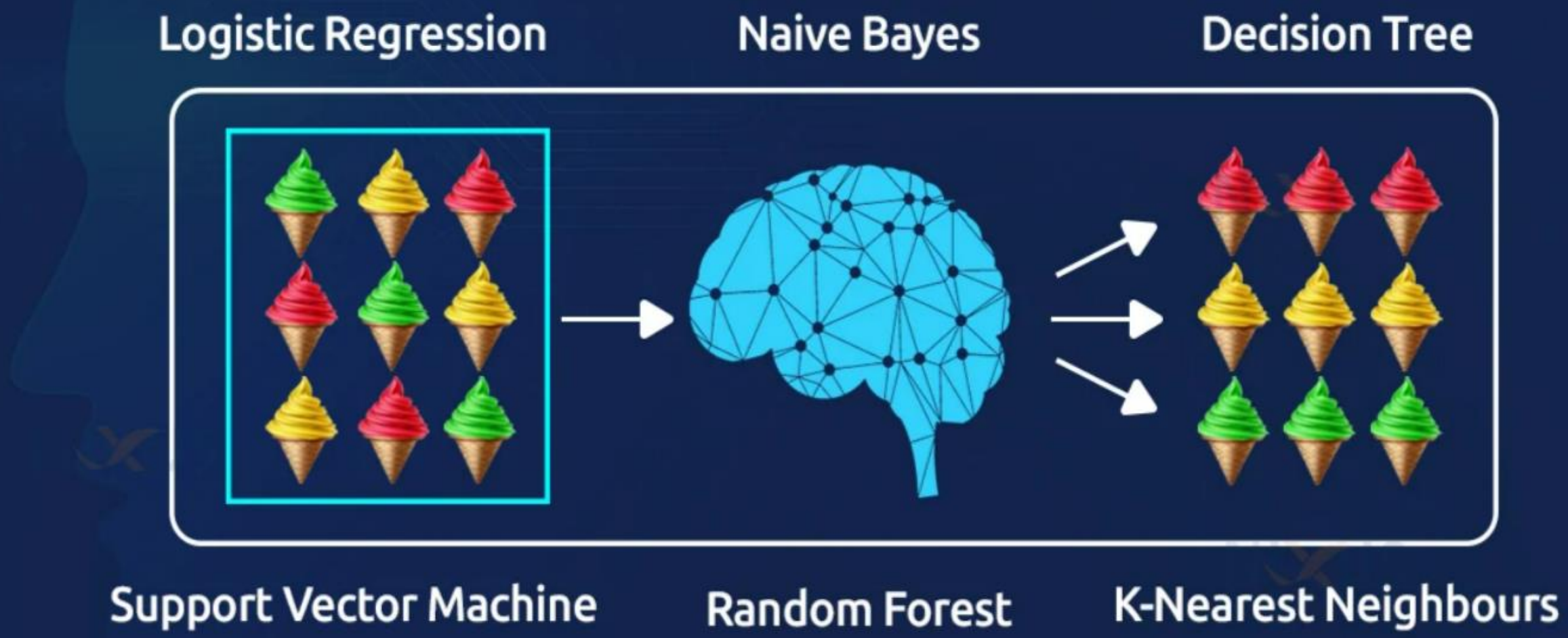
If we detect the booster version, the one with higher success rate is the FT whereas the booster version with the most failures is V1.1

In addition, B5 has just few observations, but it is successful. Nevertheless, using this booster version can lead to a failure in the mission in upcoming launches.

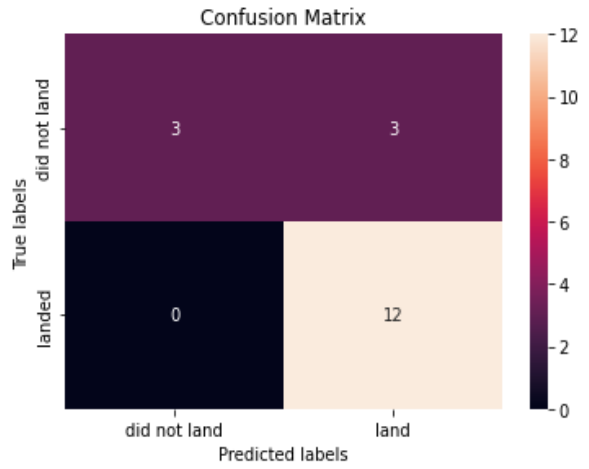
The highest success rate are located between the 2000 and 3000kg and it is almost for the FT version.

Interactive web site

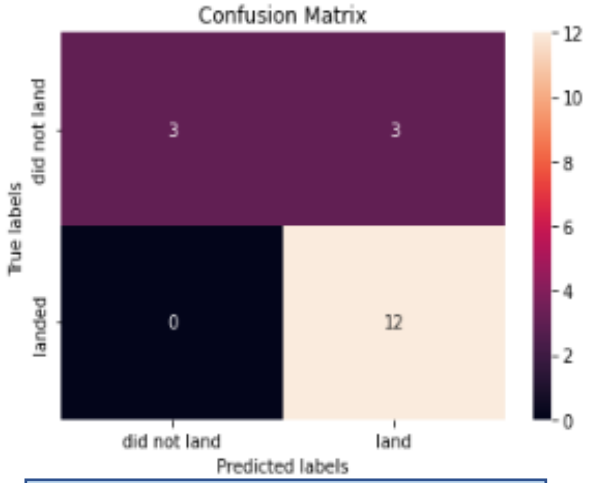
Predictive Analysis (Classifications)



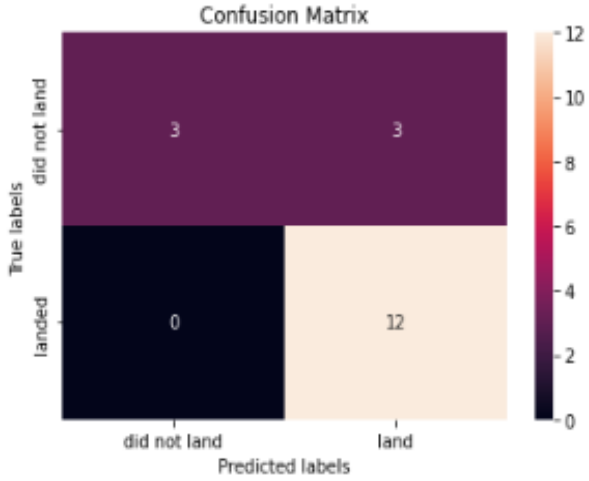
Results – Confusion Matrix



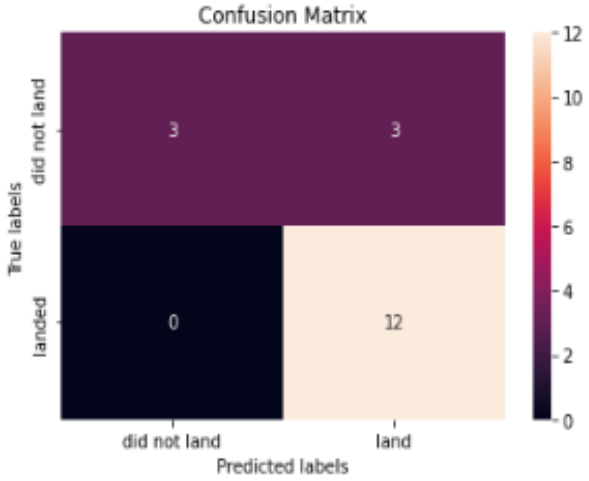
Logistic Regression



Decision Tree



SVM



KNN

Confusion Matrix allows us to compare the actual values against the predicted values as follows

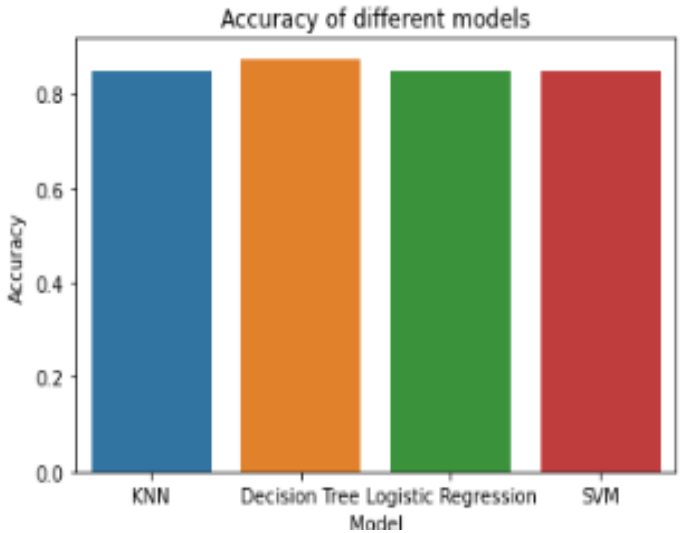
True Negatives	False Positives
False Negatives	True Positives

Doing some calculations we have that

Accuracy: 0.83333 → ((TP+TN)/Total) = (12+3)/18
Misclassification Rate: 0.1667 ((FP+FN)/Total) (3+0)/18
True Positive Rate: 1 → (TP/Actual Yes) = (12/12)
False Positive Rate: 0.5 → (FP/ Actual No) = (3/6)
True Negative Rate: 0.5 → (TN/ Actual No) = (3/6)
Precision: 0.6667 → (Actual yes / Total) = 12/18

Results – Classification Accuracy

	Accuracy
Logistic Regression	0.846429
SVM	0.848214
KNN	0.848214
Decision Tree	0.875000



As Accuracy table shows, the Decision Tree is the one with the largest prediction values with 87.50% of accuracy, while the lowest is the Logistic regression with 84.64%.

Even accuracy rate has only a 3% of difference, in prediction models this difference is big, because we can reduce the risk of failures. In this case, For the Space X launching, the closer to 1 the accuracy the coefficient, the better.

In addition, even the confusion matrix provides the same results for the four predictors, at the end the decision tree would let us to a more successful results.

CONCLUDING REMARKS

Data Science Capstone Project

Concluding Remarks

1) **The Tree Classifier Algorithm performs the best the prediction of success rate for the data set**
2) **In general terms, low weighted payloads have a greater accuracy for success than the heavier payloads**
3) **KSC LC-39A has the most successful launches from all the sites inside the study**
4) **Orbits such as GEO, HEO, SSO, ES-L1 have the best option for a greater success rate**
5) **There is a tendency for a greater success which is directly related with the time, and the launches stage is already in a mature phase**

Source: IT Pro Today
https://www.itprotoday.com/iaaspaas/10th-ibm-cloud-multi-zone-region-opens-more-works

References

Azul School website https://www.azulschool.net/presentacion/curso-web-scraping/

Boostlabs https://boostlabs.com/blog/10-types-of-data-visualization-tools/

db2 tutorial https://www.db2tutorial.com/

HYGGER website https://hygger.io/blog/what-is-project-proposal-in-project-management/

SpaceX website https://www.spacex.com/mission/

Train Test Split website https://traintestsplit.com/interactive-data-visualization-in-python-a-plotly-and-dash-intro/

Towards Data Science https://towardsdatascience.com/top-machine-learning-algorithms-for-classification-2197870ff501

Vista Projects website https://www.vistaprojects.com/blog/how-to-write-an-effective-executive-summary/

YouTube https://www.youtube.com/watch?v=QpBmO35pmVE