IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Roberto Delgado Ferrezuelo
23/08/2022

Outline

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

Executive Summary

Methodologies

The aim of this work is to determine the performance of SpaceX in the **Falcon 9 first stage land**

To achieve this, data has been collected from an open-source REST API and from Wikipedia

Results

With the processed data, different statistical models have been trained and evaluated in order to obtain the best possible prediction. The models are the following:

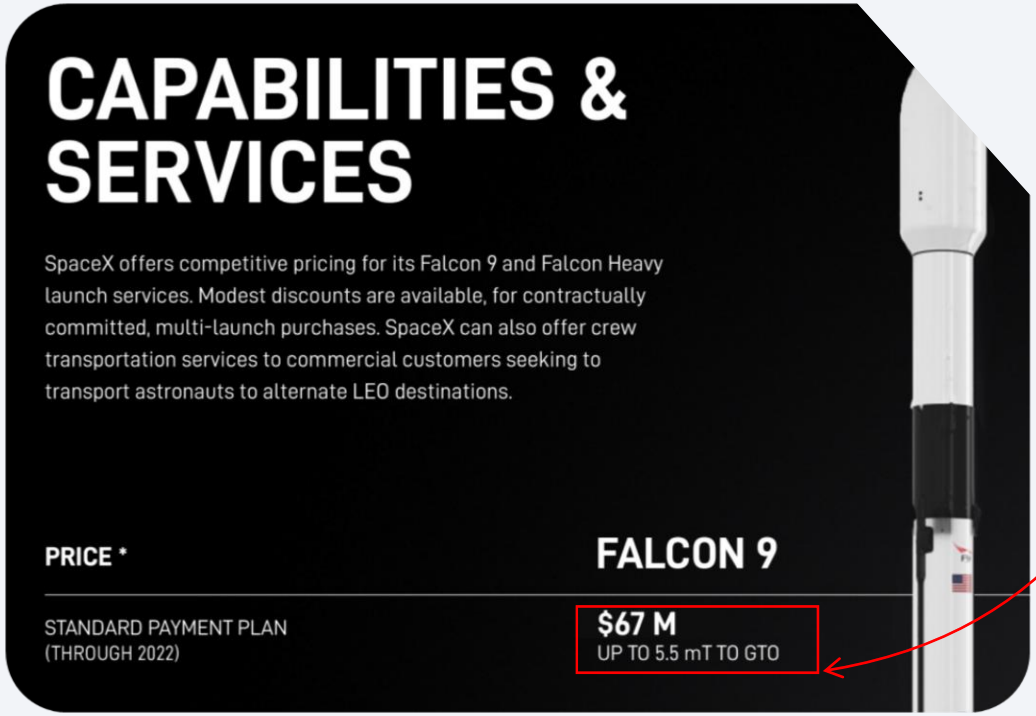
Decision Tree

Support Vector Machine

K-Nearest Neighbor

Logistic Regression

Introduction



Project background

SPACEX is the biggest competitor in the space industry

The launch cost of the **Falcon 9** has been published on their website

Other providers costs can be up to **$165 million per launch**

SpaceX reuses the first stage

Problem to find answers

How much is going to cost to SpaceX the launch? → **Probability of first stage lands successfully**

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data has been collected from both Wikipedia and a REST API (endpoint: https://api.spacexdata.com/v4/)
- Perform data wrangling
  - Data have been analyzed and processed in order to get a better insight. One of the categorical variables has been replaced with a numerical one. Some null values have been replaced
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform feature engineering
- Perform predictive analysis using classification models
  - 4 different models have been trained and tested with the processed data

Data Collection

REST API

- Use a get request to the URL from the endpoint
- Covert the json object to a dataframe
- Define functions to process the raw data from the dataframe
- Process a subset of the dataframe using the functions
- Filter the dataframe to only include the desire rows

Web scraping

- Use a get request to the Wikipedia URL
- Create a BeautifulSoup object from the response
- Extract the column headers and use them as keys of a dictionary
- Use a pre-defined functions in a for loop in each row of each table
- Store the values into the dictionary and convert it to a dataframe

Data Collection – SpaceX API

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

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

Get request to the API endpoint

```
# Use json_normalize meethod to convert the
data=pd.json_normalize(response.json())
```

JSON object to dataframe

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

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

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

Functions that use the API to get launches information from the IDs

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
```

Columns combination in a dataframe

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9=data_falcon9[data_falcon9["BoosterVersion"]=="Falcon 9"]
```

Filter to only include Flacon 9 launches

GitHub URL of the completed SpaceX API calls notebook:

https://github.com/rdf5/IBM-Skills-Network-Applied-Data-Science-Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Data Collection - Scraping

```
response=requests.get(static_url)
soup=BeautifulSoup(response.content)
soup
```

Get request and BeautifulSoup object creation

```
column_names = []
# Apply find_all() function with `th` elemer
elements= first_launch_table.find_all("th")
# Iterate each th element and apply the prov
for i in elements:
    column=extract_column_from_header(i)
    if(column!=None and len(column)>0):
        column_names.append(column)
```

Columns headers extraction

```
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
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            #print(flight_number)
            launch_dict['Flight No.'].append(flight_number)
            datatimelist=date_time(row[0])
```

For loop to store values in dictionary lists

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

Dataframe creation from dictionary

GitHub URL of the completed web scraping notebook:

https://github.com/rdf5/IBM-Skills-Network-Applied-Data-Science-Capstone/blob/main/jupyter-labs-webscraping.ipynb

Data Wrangling

Firs step in Data Wrangling has been dealing with null values

```
In [105…  data_falcon9.isnull().sum()

Out[105…  FlightNumber       0
          Date               0
          BoosterVersion     0
          PayloadMass        5
          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
```

5 values from the PayloadMass column seems to be null so they have been replaced by the column mean

Null values from LandingPad column are not replaced since they mean no landing pad were used

GitHub URL of the completed data wrangling notebook:

https://github.com/rdf5/IBM-Skills-Network-Applied-Data-Science-Capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

Data Wrangling

Data Analysis

Next step is the data analysis

```
In [7]: # Apply value_counts() on column LaunchSite
        df["LaunchSite"].value_counts()

Out[7]: CCAFS SLC 40    55
        KSC LC 39A      22
        VAFB SLC 4E     13
        Name: LaunchSite, dty
```

Calculate number of launches in each site

```
In [8]: # Apply value_counts on Orbit column
        df["Orbit"].value_counts()

Out[8]: 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: i
```

Calculate number of launches to each orbit

```
In [9]: df["Outcome"].value_counts()

Out[9]: 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: int6
```

Calculate occurrence of each mission outcome

```
In [79]: landing_class=np.ones(len(df["Outcome"]))
         landing_class[df["Outcome"].isin (bad_outcomes)]=0
         landing_class=landing_class.tolist()
```

Create a numerical variable for landing outcome

EDA with Data Visualization

Exploratory data analysis has been carried out to discover patterns in the data. It has been done with the help of some helpful libraries for data visualization and with SQL

plotly Folium matplotlib

Interesting findings have been made. After that **features engineering** has been performed

- Variables of interest have been selected for training the model
- One hot encoding has been applied to categorical variables
- The resulting dataframe has been casted to float type

EDA with Data Visualization

Charts used

Scatter plots

- Payload mass vs Flight number
- Flight number vs Launch site
- Payload mass vs Launch site
- Flight Number vs Orbit type
- Payload mass vs Orbit type

Bar graphs

- Success rate at each orbit type

Line plots

- Launch success yearly trend

GitHub URL of the completed EDA notebook:

https://github.com/rdf5/IBM-Skills-Network-Applied-Data-Science-Capstone/blob/main/jupyter-labs-eda-dataviz.ipynb

EDA with SQL

The following subsets of the data have been displayed using SQL:

1. Names of the unique launch sites
2. 5 records where the launch sites begin with ‘CCA’
3. Payload mass carried by boosters launched by NASA
4. Average payload mass carried by booster version F9 v1.1
5. Date when the first successful landing outcome in ground pad was achieved
6. Names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
7. Total number of successful and failure mission outcomes
8. Names of the booster versions which have carried the maximum payload mass
9. Failed landing outcomes in drone ship, their booster versions, and launch site names for in year 2015
10. Ranking of the count of landing outcomes between the date 2010-06-04 and 2017-03-20

GitHub URL of the completed EDA-SQL notebook:

http://localhost:8888/notebooks/jupyter-labs-eda-sql-coursera.ipynb

Build an Interactive Map with Folium

An interactive map has been built to examine the proximity of each site to nearby cities, railways, coastlines…

For this purpose, the following objects have been added to the map:

- Circles surrounding the launch sites with a popup text indicating their names
- Markers in the coordinates of each site displaying their name
- Markers with different colors representing each of the launches and a popup text with the flight number
  - Green markers: Success in the first stage land
  - Red markers: Failure in the first stage land
- MarkerClusters to group markers with the same coordinates
- MousePosition to get the coordinates of the points of interest
- Markers indicating the distance from the launch site to the point of interest and a polyline between them

GitHub URL of the completed Folium Map notebook:

https://github.com/rdf5/IBM-Skills-Network-Applied-Data-Science-Capstone/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

Interactive dashboard app has been deployed in a web server using Dash

The app shows two graphs with information of the launches in the selected sites in a specified range of payload

It include the following elements:

- Title
- DropDown list to select sites
- Pie chart indicating the success rate in the selected site
- RangeSlider to select the payload mass range
- Scatter plot of the payload mass vs outcome, colored by booster version category for the launches in the specified site and payload mass range

GitHub URL of the .py file with the Dash APP:

https://github.com/rdf5/IBM-Skills-Network-Applied-Data-Science-Capstone/blob/main/spacex_dash.py

Predictive Analysis (Classification)

The best model has been found following the next flowchart:

Data preparation
- Data conversion to numpy array
- Data standarization
- Train-test data split

Model definition
- Create the objects for each model
- Define the parameters to be tested

Hyperparameter selection
- Parameters are selected using GridSearchCV in the train data
- Model is defined with the best parameters

Model evaluation
- Accuracy score in the test set is calculated
- Confusion matrix is printed

GitHub URL of the completed Machine Learning notebook:

https://github.com/rdf5/IBM-Skills-Network-Applied-Data-Science-Capstone/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

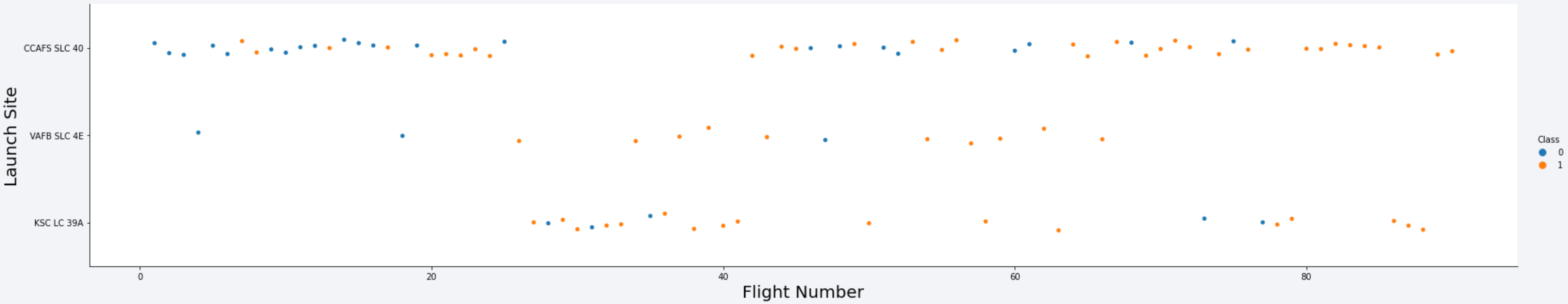
Results

- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

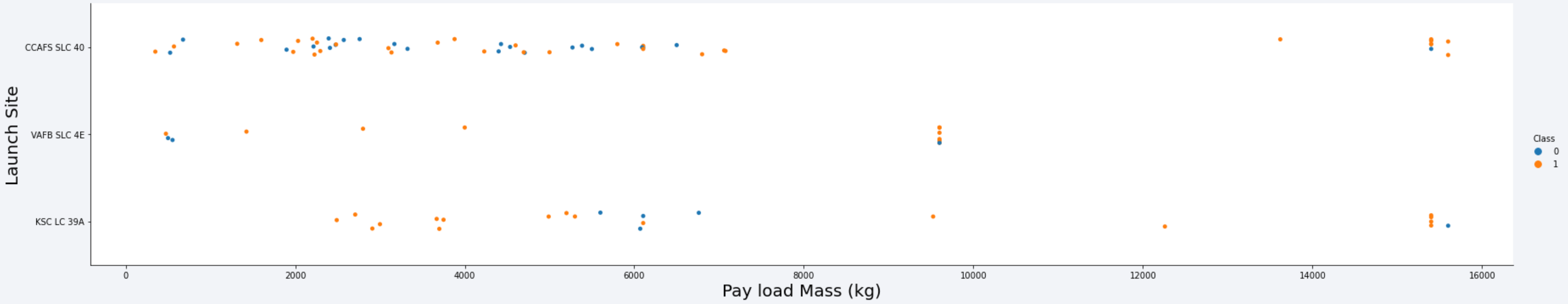


0: failure

1: success

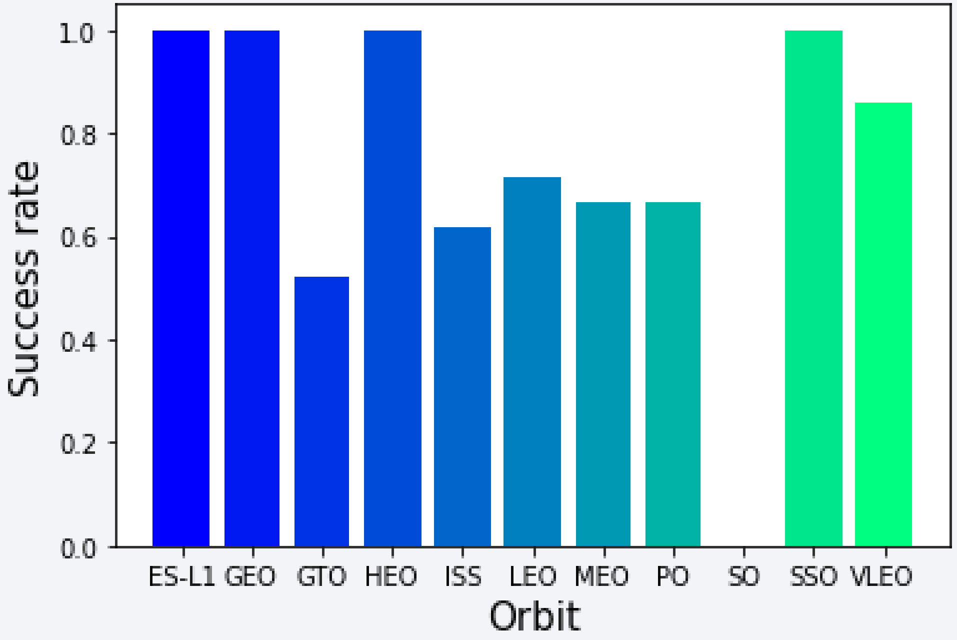
- Success rate at VAFB SLC 4E is the highest
- As the flight number increase, the probability of success clearly increase in two of the sites
- Most of the launches were carried out at CCAFS SLC 40

Payload vs. Launch Site



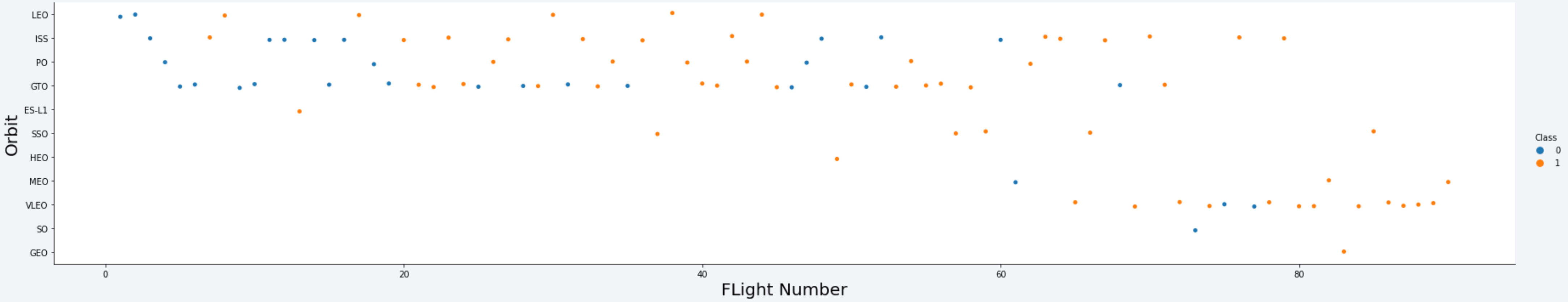
- There are no heavy payload launches at VAFB SLC 4E
- Success rate at CCAFS SLC 40 seems not to be correlated with the payload mass
- At KSC LC 39A most successful missions are the ones with the lowest payload mass

Success Rate vs. Orbit Type



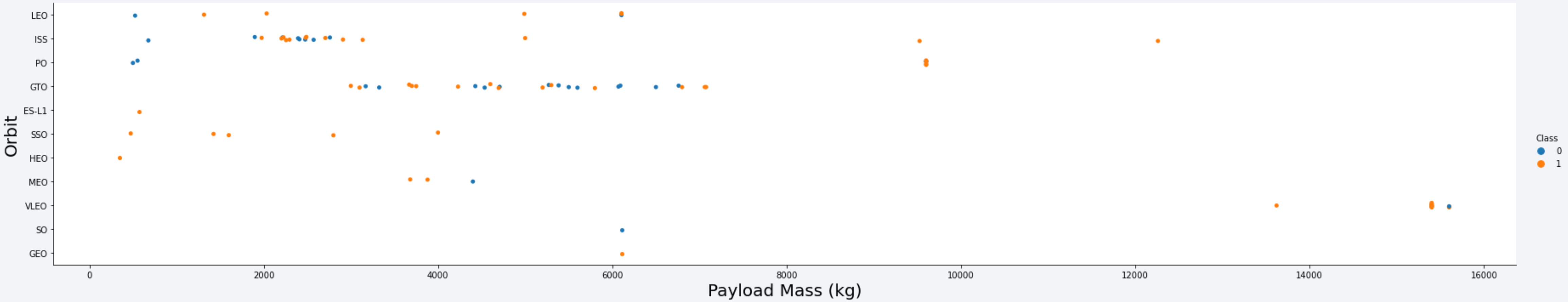
- There are no successful lands at SO orbit group
- The least successful lands are the ones corresponding to missions sent to the GTO orbit
- ES-L1, GEO, HEO, SSO and VLEO have a high success rate

Flight Number vs. Orbit Type



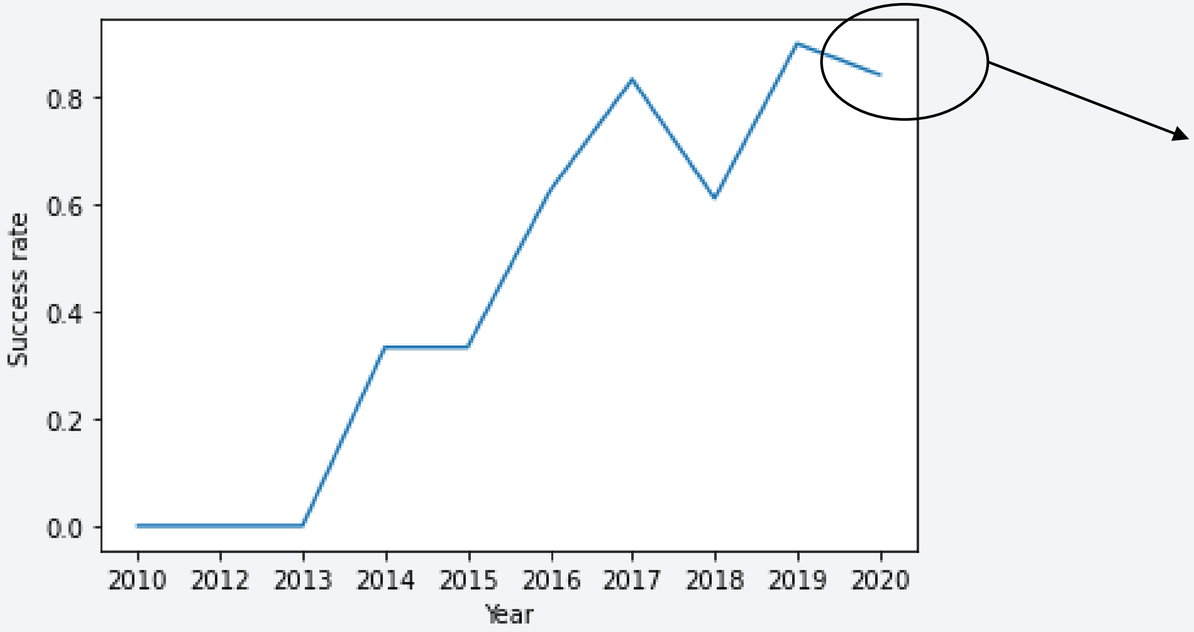
- In most of the orbits the number of flights seems to have a strong influence except in GTO orbit
- Only one mission has been sent to each of the SO and GEO orbits

Payload vs. Orbit Type



- With heavy payloads the probability of success for the ISS and PO is higher
- Missions sent to the GTO orbit does not present a strong dependence on the payload mass

Launch Success Yearly Trend



Mission success rate has been increasing since 2013 till 2020

All Launch Site Names

```
%sql select distinct launch_site FROM SPACEX
```

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

Distinct is used in the select statement to display the unique values

*There are four distinct locations, yet CCAFS LC-40 seems to be the previous name of the actual CCAFS SLC-40 (that is why they can be seen mixed in the previous plots)

Launch Site Names Begin with 'CCA'

```
In [7]: %sql select * from spacex where \
        launch_site like 'CCA%';
```

```
 * ibm_db_sa://hrz02103:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30699/bludb
Done.
```

Out[7]:

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
2013-12-03	22:41:00	F9 v1.1	CCAFS LC-40	SES-8	3170	GTO	SES	Success	No attempt

The where clause allows to only display the rows that match the specified criteria, in this case, those rows whose columns begin with ‘CCA’

Total Payload Mass

```
In [22]: %sql select sum(payload_mass__kg_) as TOTAL_MASS from spacex where customer='NASA (CRS)'

          * ibm_db_sa://hrz02103:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databa
         Done.
```

Out[22]:

total_mass
45596

SUM() is a built-in function that calculates the sum of the values in the specified column. The where statement restricts the number of rows to be included in the column sum

Average Payload Mass by F9 v1.1

```
%sql select avg(payload_mass__kg_) as average_payload_mass from spacex \
where booster_version like'%F9 v1.1%'
```

```
 * ibm_db_sa://hrz02103:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd
Done.
```

average_payload_mass
2534

AVG() is other of the built-in functions, it can be used to calculate the mean of the specified columns, considering only the rows that matches the where condition.

First Successful Ground Landing Date

```
%sql select min(date) from spacex where landing__outcome='Success (ground pad)'
```

```
 * ibm_db_sa://hrz02103:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lq
Done.
```

1
2015-12-22

MIN() calculates the minimum value in the specified column. In this case, date has been selected, also a where clause has been added

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql select distinct booster_version from spacex where payload_mass__kg_ >4000 and payload_mass__kg_ <6000 and \
landing__outcome='Success (drone ship)'
```

```
 * ibm_db_sa://hrz02103:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:3(
Done.
```

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

Distinct statement has been used and in the where statement a logical operator has been added to indicate that all the three conditions need to be met in order to include the row. Instead of the and operator for selecting the payload mass range it could be used the between operator

Total Number of Successful and Failure Mission Outcomes

```
%sql select mission_outcome, count(mission_outcome) as total_number from spacex group by mission_outcome
```

```
 * ibm_db_sa://hrz02103:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomai
Done.
```

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

COUNT() and group by have been used to calculate the total number of values in each of the groups formed in the mission_outcome column.

*Note that mission outcome doesn´t mean successful first stage land which is the main objective to determine in this report

Boosters Carried Maximum Payload

```
%sql select distinct booster_version from spacex where payload_mass__kg_\
=(select max(payload_mass__kg_) from spacex)
```

In this statement a subquery has been added in order to apply to the where clause one of the built-in functions. A distinct statement is used to only display different types of boosters

booster_version
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

2015 Launch Records

```
%sql select date, landing__outcome, booster_version, launch_site from spacex where \
landing__outcome='Failure (drone ship)' and DATE like '2015%'
```

```
 * ibm_db_sa://hrz02103:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.c
Done.
```

DATE	landing__outcome	booster_version	launch_site
2015-01-10	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
2015-04-14	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

4 of the dataframe columns are displayed, two conditions within the where statement needs to be met in order to display the rows, this is done by using an and operator

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

```
%sql select landing__outcome, count(landing__outcome) as total from spacex where \
date between '2010-06-04' and '2017-03-20' group by landing__outcome \
order by total desc
```

```
 * ibm_db_sa://hrz02103:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde
Done.
```

landing__outcome	total
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

Landing outcome column is display together with the count of the values that fall in each of the groups within the specified dates, this is done using the COUNT() function followed by a where and group by statements. Order by command is applied to sort the values in the total column in descending order

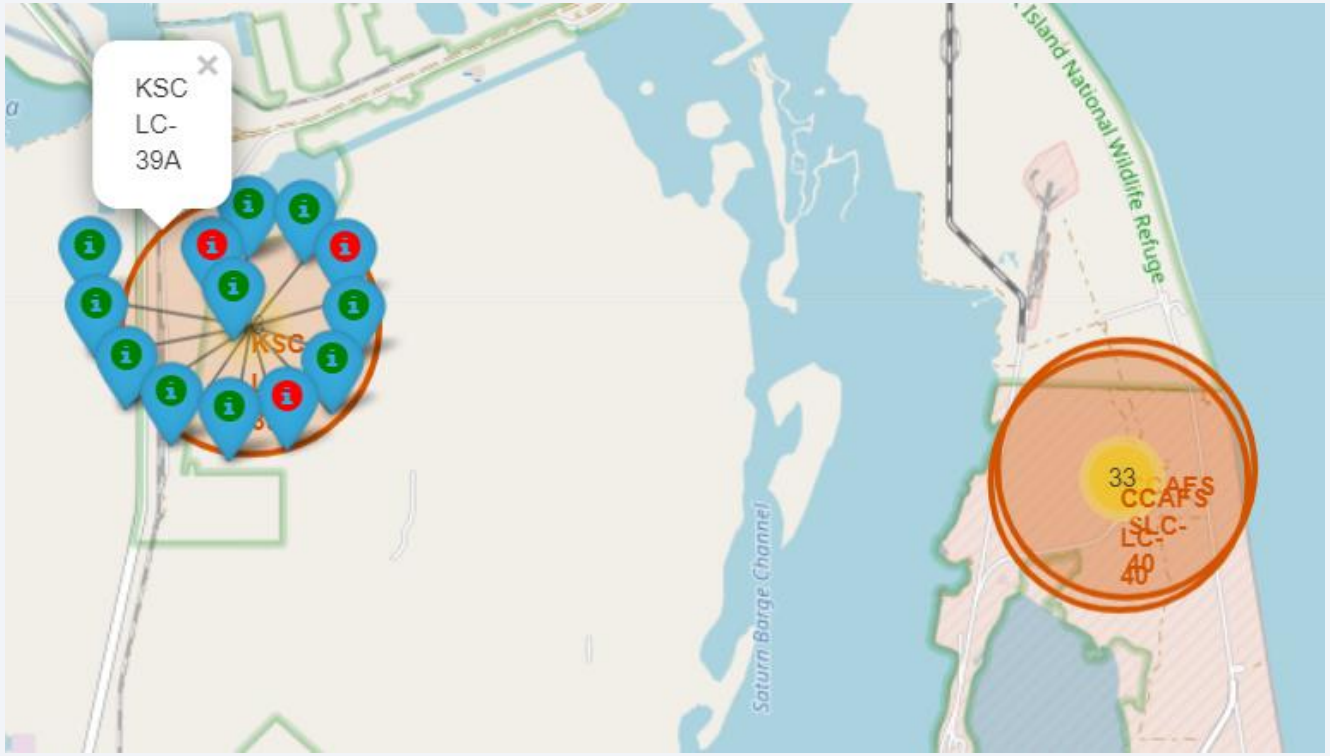
Section 3

Launch Sites Proximities Analysis

Folium Map Launch sites

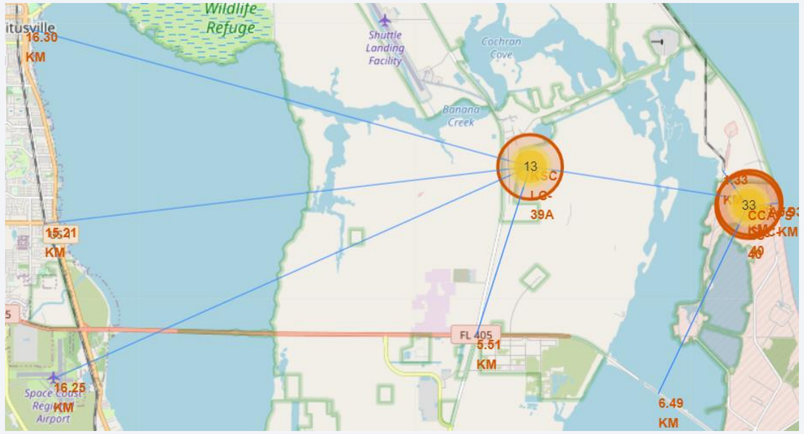
The four stations have been represented in the map using orange circles and orange markers with their names. Clusters have been used to group the launch markers in each site

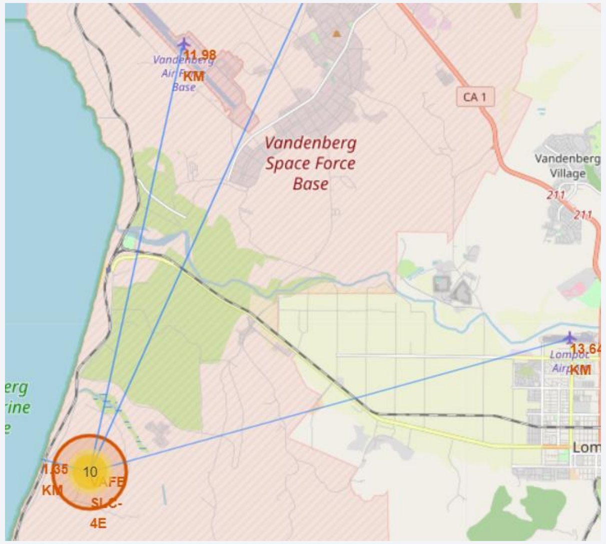
Folium Map Markers



Different color have been set in each marker depending on the first stage land outcome. Also, a popup text from one of the circles can be seen in the image

Folium Map Distance to closest points





Distance to the closest points to each site has been calculated using the Harvesine formula and represented with markers and polylines in the folium map

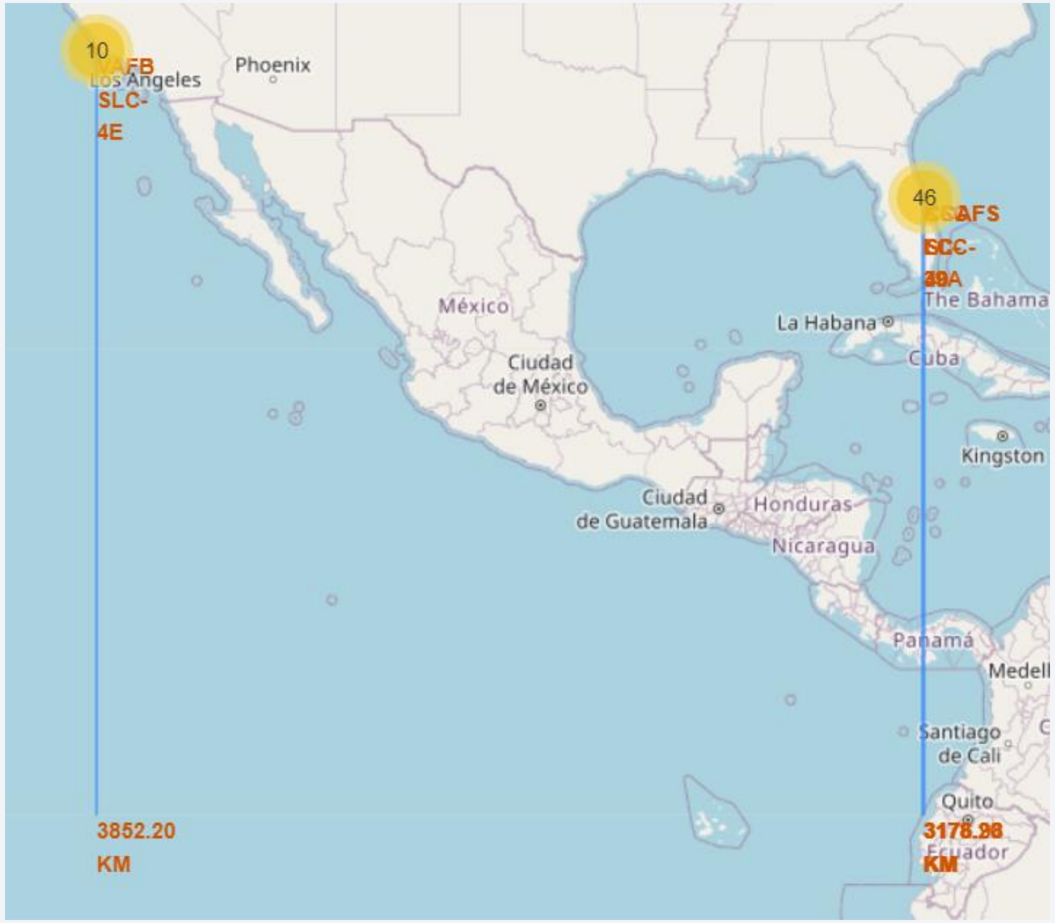
From the analysis we obtain the following conclusions:

- CCAFS SLC-40 and VAFB SLC-4E are close to a railway and to the coastline
- All sites are more than 10 km away from cities
- KSC LC-39A is 5 km from Nasa Parkway

The proximity of the launch sites to the railways can be explained by the need to supply of some of the elements such as rocket boosters

The proximity to the coastline can be an advantage in case of failure, water is a safe place to drop the rocket and in case of fire it is convenient to have a water supply near the incident

Folium Map Distance to the equator line



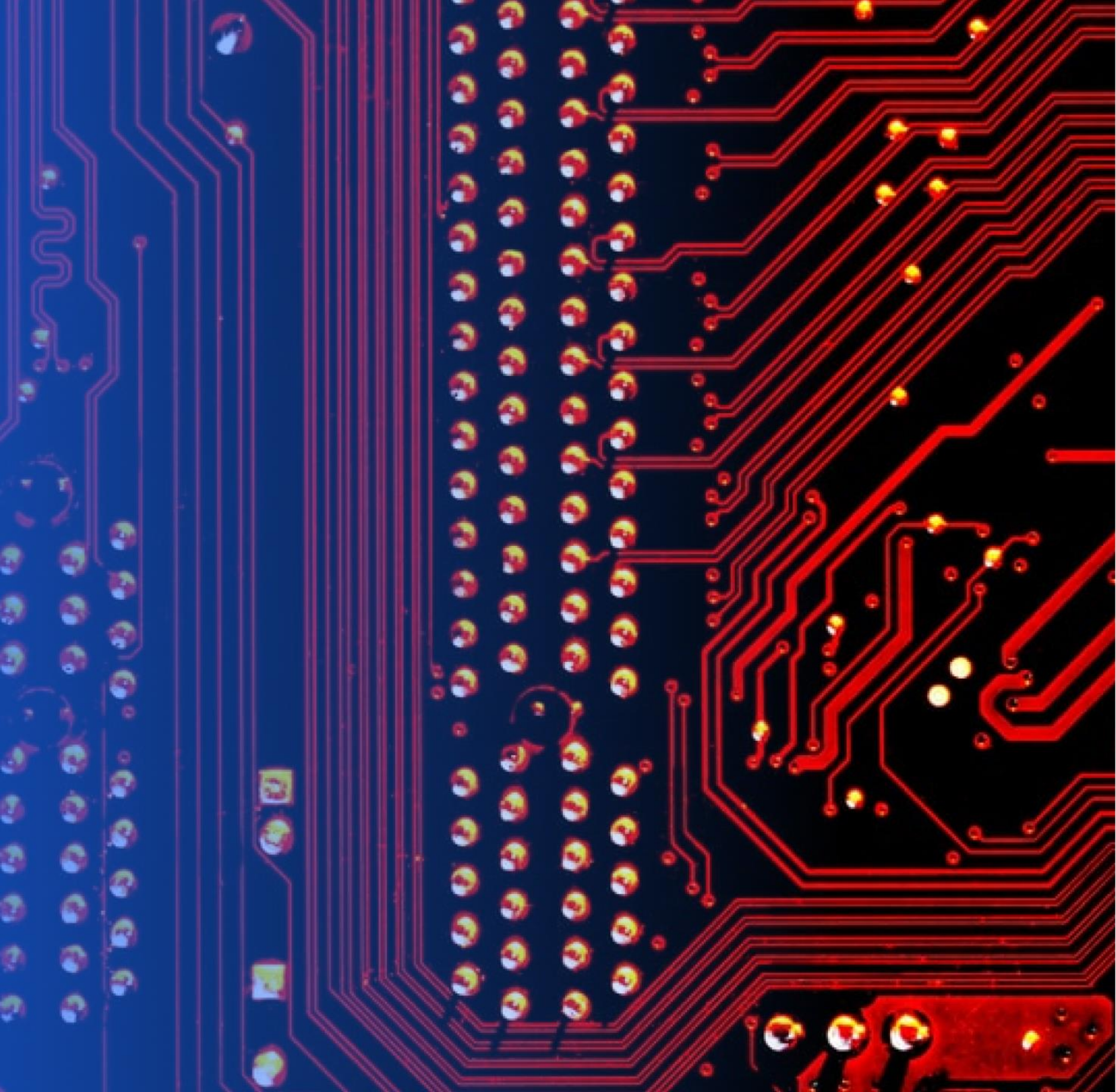
Due to the earth´s rotation, most efficient launches are the ones made near to the equator line. However, launches near to the earth poles offer advantages to launches in some orbit types.

As can be seen in the image, the four stations are in close latitudes, about 3.000 km from the equator line.

It can be concluded that **the four stations are in places away from cities, near to railways and coastlines and in the closest points within the US borders to the equator line**

Section 4

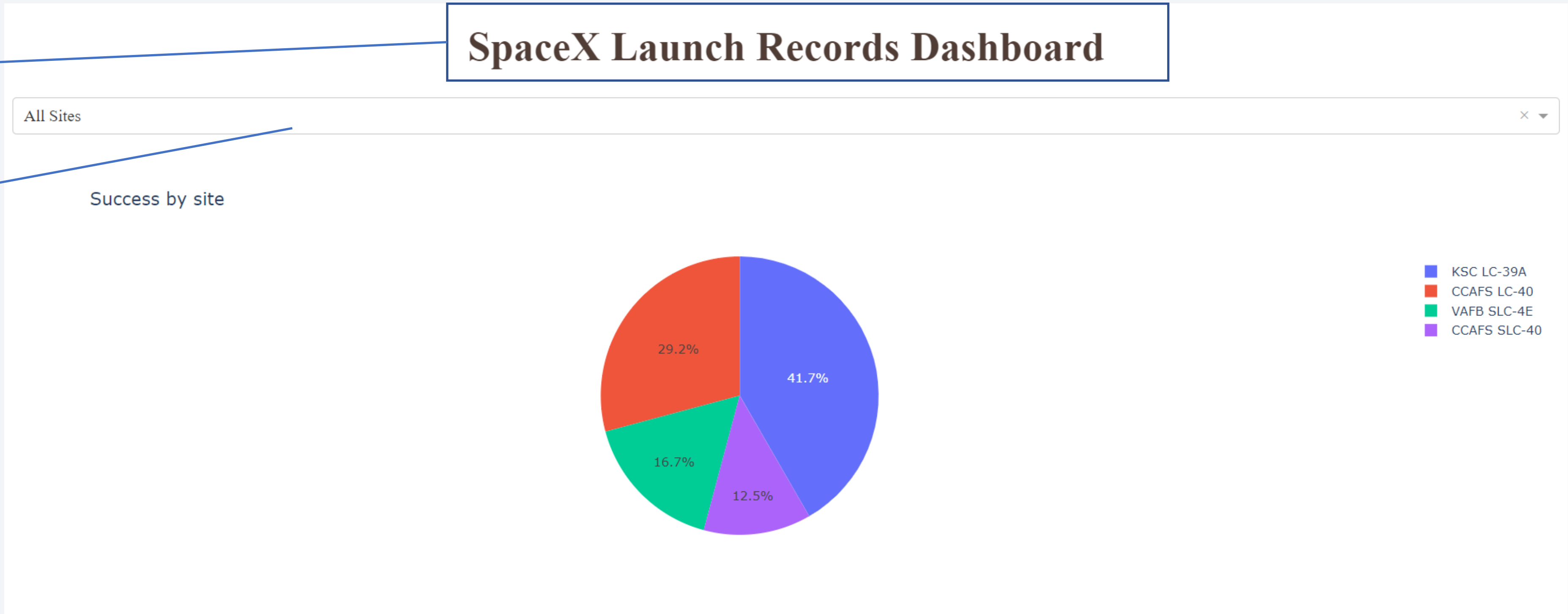
Build a Dashboard with Plotly Dash



Dashboard - land success by site

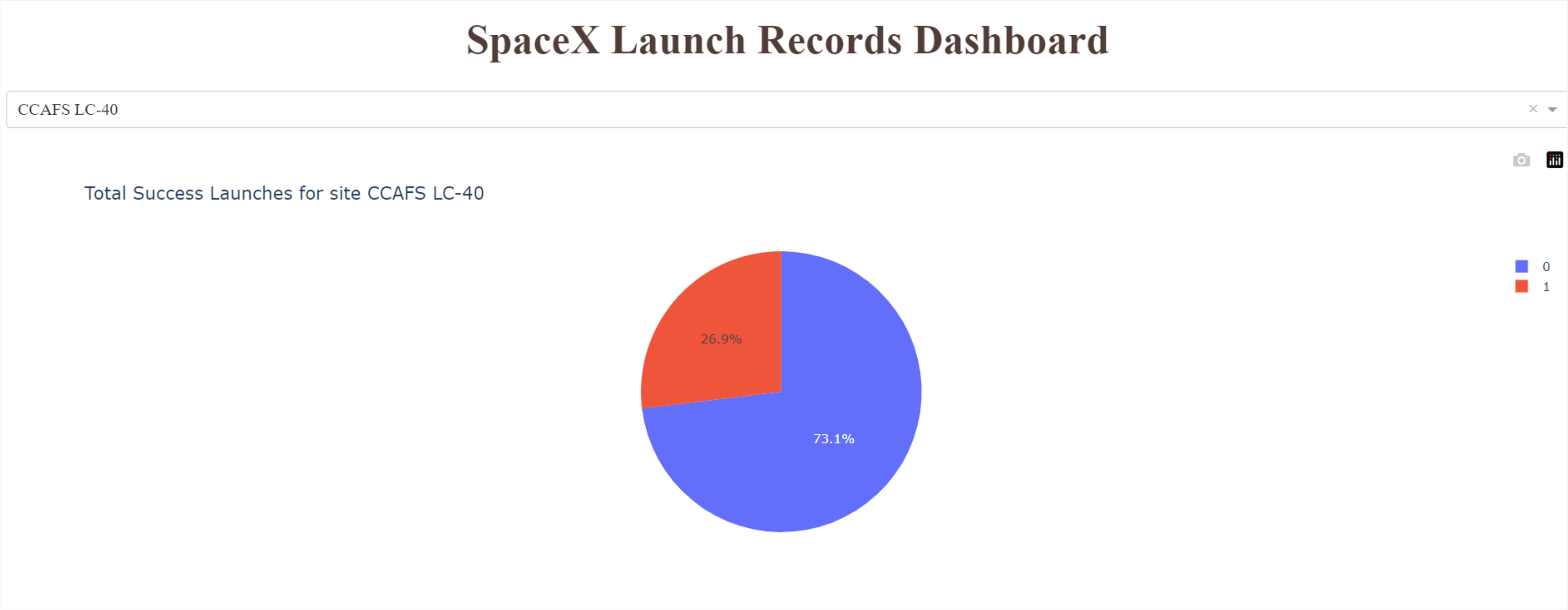
Title

DropDown list



In the pie chart success rate for each of the sites can be seen, KSC LC-39A has the highest

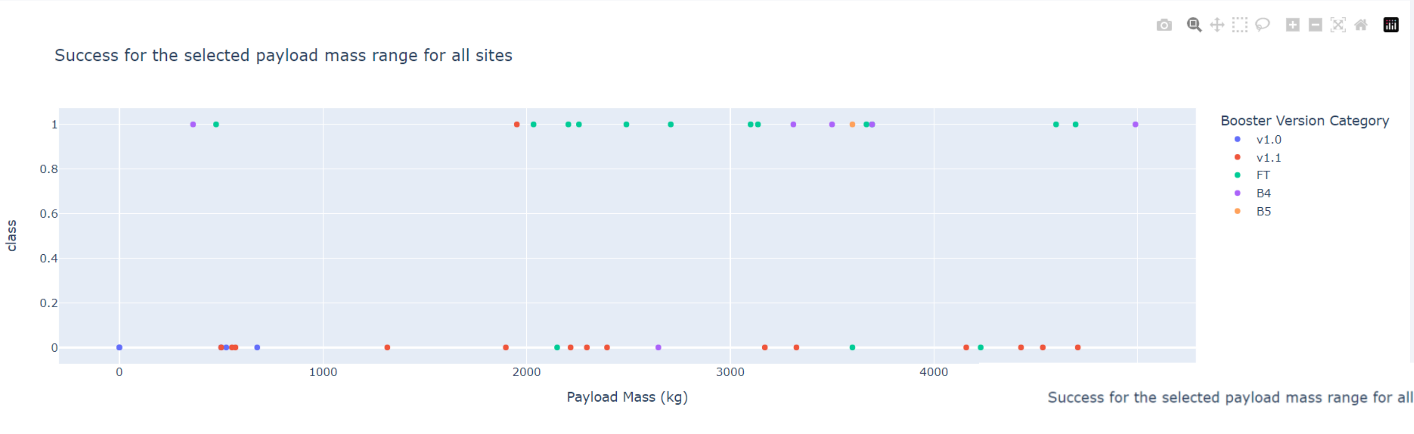
Dashboard - success rate at a specific site



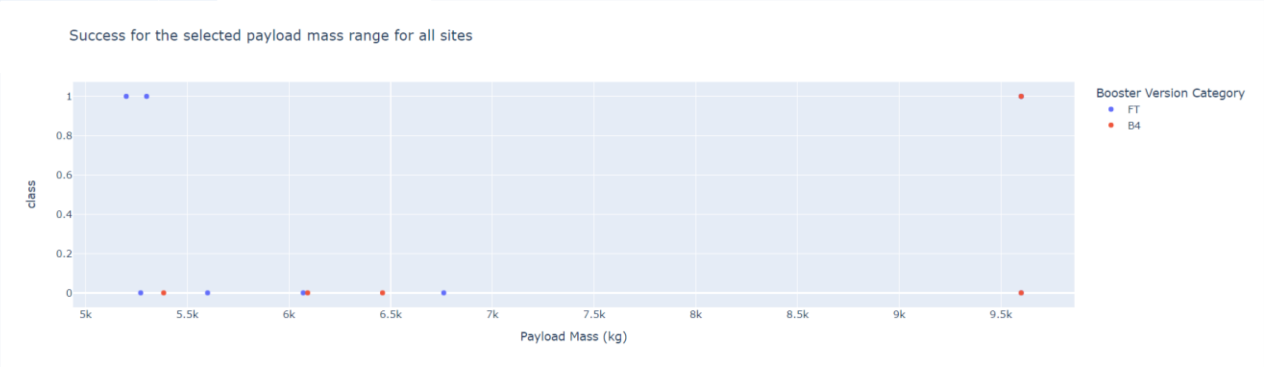
Success rate for CCAFS LC-40 is displayed, red color correspond to the successful lands and blue to the failures

Dashboard - Scatter plots

Payload mass < 5.000 kg



5.000 kg < Payload mass < 10.000 kg

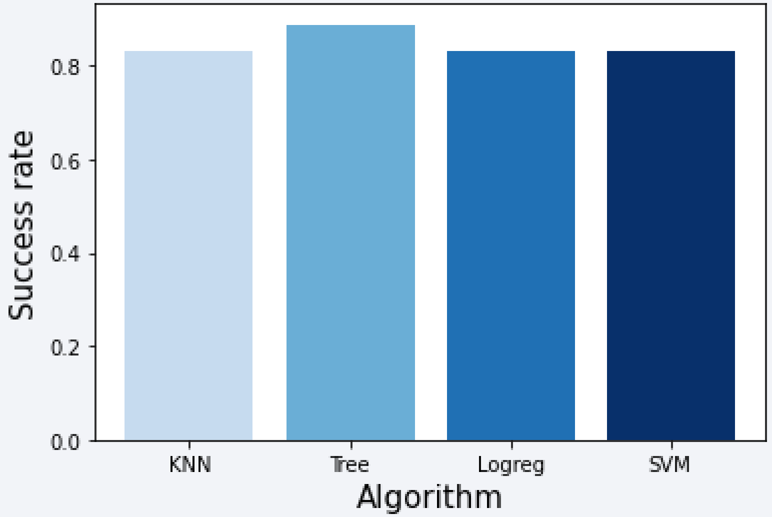


Launches are displayed for two different payload mass ranges. Class 1 represent successful lands while 0 failure. Each point has been colored depending on the Booster Version Category used. It can be clearly seen that most of the class 1 points are in the first graph (lighter payloads)

Section 5

Predictive Analysis (Classification)

Classification Accuracy



	Accuracy scores
Algorithm	
KNN	0.833333
Tree	0.888889
Logreg	0.833333
SVM	0.833333

The three models perform similarly, yet the accuracy of the decision tree is a bit higher than the other three
Other metrics could be considered to better differentiate between them

Classification Accuracy

Algorithm	Jaccard	F1-score	LogLoss
KNN	0.700000	0.814815	NA
Decision Tree	0.802198	0.888889	NA
SVM	0.700000	0.814815	NA
LogisticRegression	0.700000	0.814815	0.478667

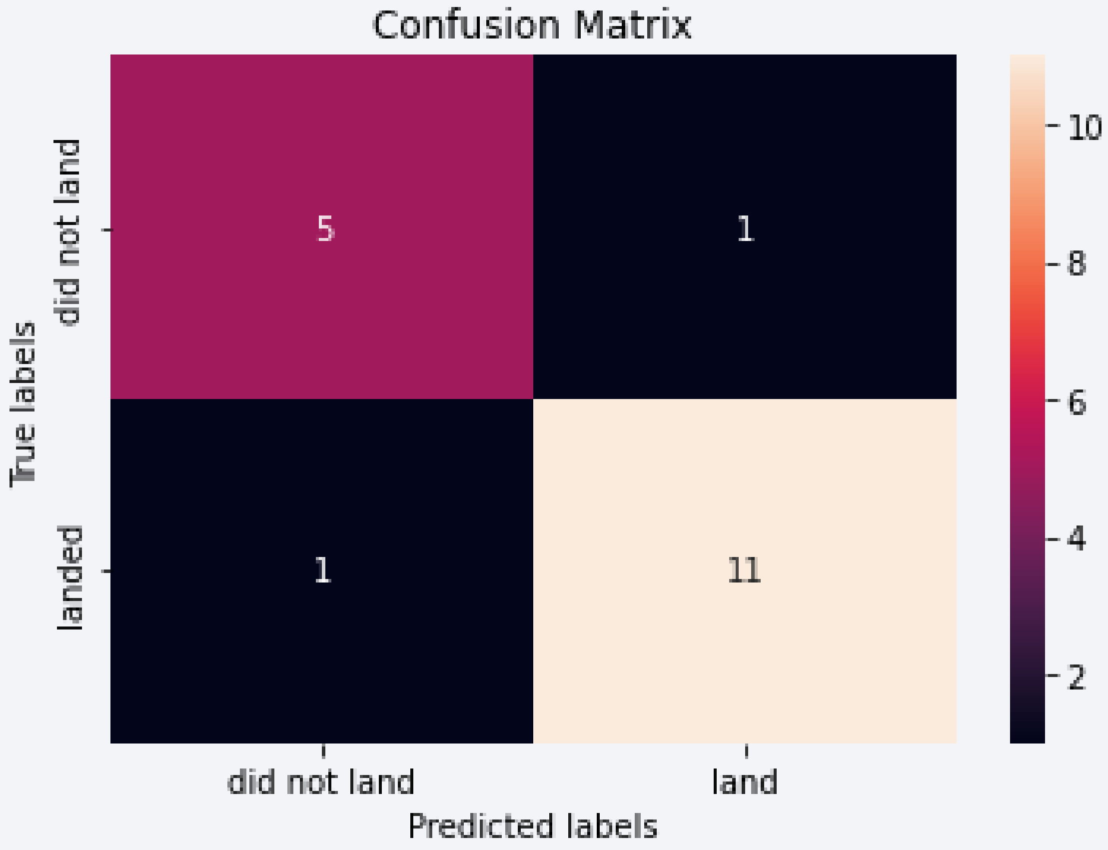
Jaccard index for each label has been calculated and the weighted mean has been represented in the table together with the F1 score and the logarithmic loss

Now it is clearer that the best model is the decision tree

Decision tree´s parameters:

- criterion: entropy
- max_depth: 12
- max_features: auto
- min_samples_leaf: 2
- min_samples_split: 10
- splitter: random

Confusion Matrix



The model only fails predicting two labels out of 18

It is well balanced since it doesn´t tend to classify all the variables in the same group (there are same false positives as false negatives)

Conclusions

- SpaceX first stage successful land depends upon many factors. In the exploratory analysis explained in this report some of them have been analyzed
- When bidding against SpaceX all these factors need to be considered. The combination of payload mass and orbit type, the number of flights attempted in that launch site, the trend to continuously increase their success rate and so on
- Their launch sites have similar characteristics which can be good for some orbits or range of payload mass, depending on their type of mission the successful first stage land probability can vary
- Other variables such as the use of grid fins, if the first stage is reused, if legs were used etc. needs to be added since they seem to have a high influence in the model accuracy
- All 4 models have a similar accuracy, however further data collection would be necessary to create a larger database to test the models with more data to better differentiate between them and ensure their accuracy with increased confidence

Appendix

GitHub URL to the open-source REST API:

https://github.com/r-spacex/SpaceX-API

GitHub URL to all the notebooks used for this project:

https://github.com/rdf5/IBM-Skills-Network-Applied-Data-Science-Capstone

Wikipedia page from which the data has been collected:

https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches

Thank you!