IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Mojtaba Mahmoodan
2022-07-27

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection (API & Web Scraping)
  - Data Wrangling
  - Exploratory Data Analysis (EDA) & Data Visualization
  - Interactive Visual Analytics with Folium
  - Predictive analysis
- Summary of all results
  - EDA Results
  - Interactive Analytics Results
  - Predictive Analytics Results

Introduction

Project background and context

- The market aims to making space travel affordable for everyone.
- SpaceX advertises the Falcon 9 rocket launches with a cost of 62 million US$
- Most of the savings comes from SpaceX reusing the first stage instruments.
- The goal of is to analyze and predict the outcome of successful first stage landings

Problems you want to find answers

- Determining factors for a successful first stage landing
- Predict the outcome of a first stage landing (successful / unsuccessful)

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Using SpaceX Rest API with Python and web scraping Wikipedia pages
- Perform data wrangling
  - Using Python pandas and numpy libraries
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Using Scikit-learn library

Data Collection

SpaceX API

Get data using SpaceX REST API (get request)

↓

Decode the response into Json format (using json() function)

↓

Transform the Json format data to a panda data frame (using json_normalize)

↓

Selecting the required data

Web scraping

Get data using web scraping (BeautifulSoup)

↓

Get the desired table in a data frame format using find_all(table) function

↓

Get the required part of table to data frame Using find_all('tr') and find_all('td')

↓

Selecting the required data

Data Collection – SpaceX API

- Get Request:

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

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

- Get result in json format and json_normalize to convert json into data frame:

```
resp_content = response.json()
```

```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(resp_content)
```

- Get desired parameters out of the data frame:

```
# Lets take a subset of our dataframe keeping only the features we want and the flight n
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the da
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

- GitHub URL of the completed notebook:

https://github.com/mojister/IBM_Applied-Data_Science_Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Data Collection – Scraping

- Get Request:

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

- Parse value using BeautifulSoup:

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

- Extract desired parameters out of the data frame

```
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
        #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.`
            launch_dict['Flight No.'].append(flight_number)
            #print(flight_number)
            datatimelist=date_time(row[0])

            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            date = datatimelist[0].strip(',')
            launch_dict['Date'].append(date)
            #print(date)
```

- GitHub URL of the completed notebook:

https://github.com/mojister/IBM_Applied-Data_Science_Capstone/blob/main/jupyter-labs-webscraping.ipynb

Data Wrangling

Transform categorical data into numeric data, by converting the outcomes into **Training Labels**, where 1 means successfully landing and 0 means it was unsuccessful

↓

Calculate the number of launches at each site

↓

Calculate the number and occurrence of each orbit

↓

Create a landing outcome label

↓

Calculate success rates

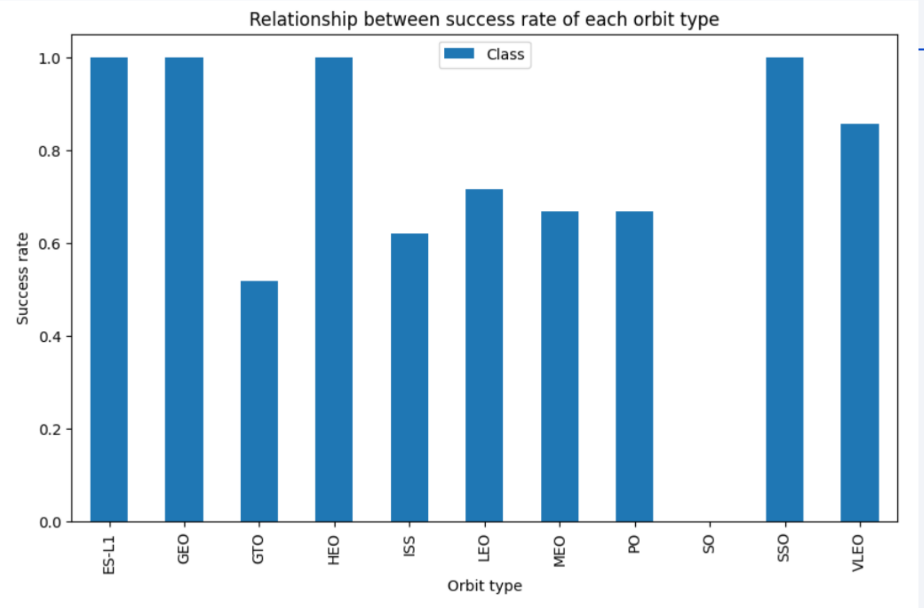
GitHub URL of the completed notebook:
https://github.com/mojister/IBM_Applied-Data_Science_Capstone/blob/main/jupyter-labs-spacex-Data_wrangling.ipynb

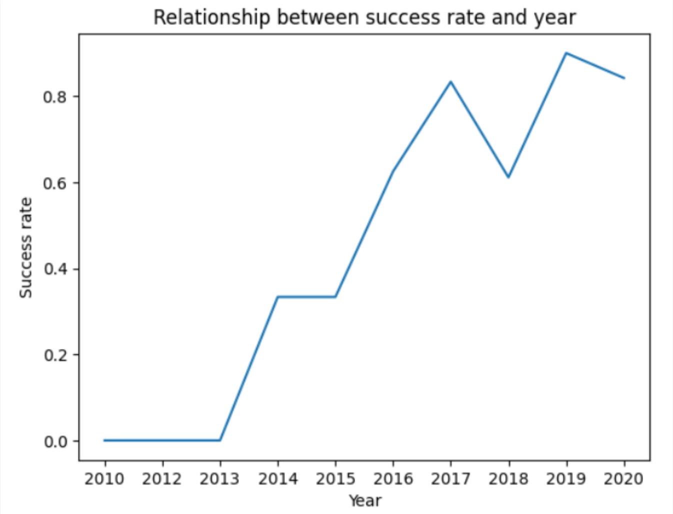
EDA with Data Visualization

Data visualization to show importance of variables (features) for the target parameter:

- Scatterplot:
  - Flight Number - Payload Mass
  - Flight Number - Launch Site
  - Launch Site - Payload Mass
  - Flight Number – Orbit
  - Payload Mass - Orbit
- Bar chart of the success rate of each orbit
- Line chart of success rate evolution in time

GitHub URL of the completed notebook:
https://github.com/mojister/IBM_Applied-Data_Science_Capstone/blob/main/jupyter-labs-eda-dataviz.ipynb





EDA with SQL

Summary of SQL queries:

```
%sql select "Launch_Site" from SPACEXTBL GROUP BY "Launch_Site";
```

```
%sql select "Launch_Site" from SPACEXTBL where UPPER("Launch_Site") like "CCA%"  limit 5;
```

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

```
%sql select AVG(PAYLOAD_MASS__KG_) from SPACEXTBL where UPPER("Booster_Version") like "%F9 V1.1";
```

```
%sql SELECT "Date" As "first_successful_landing", "Landing _Outcome" from SPACEXTBL where "Landing _Outcome" like "%S
```

```
%sql select "Booster_Version", PAYLOAD_MASS__KG_, "Landing _Outcome" from SPACEXTBL where (PAYLOAD_MASS__KG_ between
```

```
%sql select "Mission_Outcome", COUNT(*) As "Total_Number" from SPACEXTBL GROUP BY "Mission_Outcome";
```

```
%sql select "Booster_Version" , PAYLOAD_MASS__KG_ from SPACEXTBL where PAYLOAD_MASS__KG_ = (select Max(PAYLOAD_MASS__
```

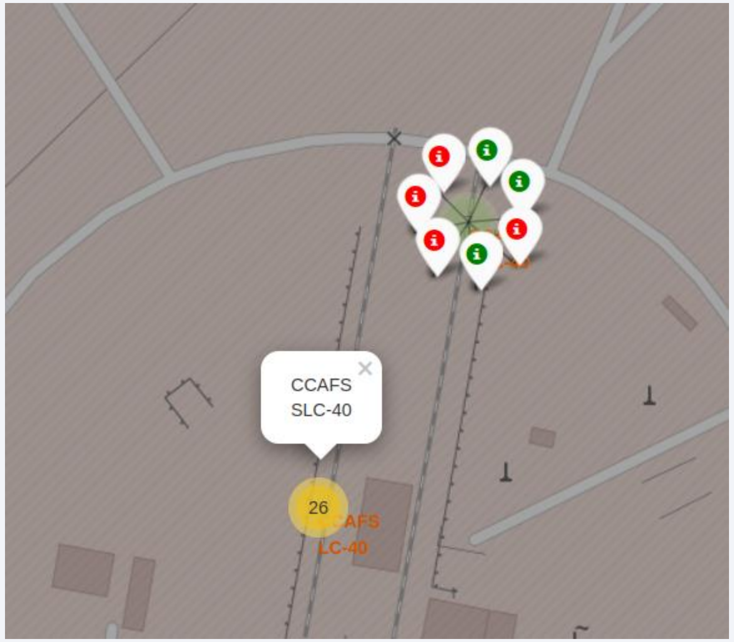
```
%sql select "Date", substr(Date, 4, 2) As "monthnames", "Landing _Outcome", "Booster_Version", "Launch_Site" from SPA
```

```
%sql SELECT "Date", "Landing _Outcome", COUNT("Landing _Outcome") AS Successful_Landing_Count from SPACEXTBL where ("
```

GitHub URL of the completed notebook:
https://github.com/mojister/IBM_Applied-Data_Science_Capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Each launch site is marked with a circle along with the name of site as a label
- Launch outcomes are split to unsuccessful and successful classes with green and red colors
- Distances between a launch site to landmarks are calculated
- Lines are drawn between launch sites and landmarks



GitHub URL of the completed notebook:
https://github.com/mojister/IBM_Applied-Data_Science_Capstone/blob/main/jupyter_labs_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

An interactive dashboard with plotly dash was build:

- Pie charts to show the total launches and success share of launches for each site
- Scatterplots showed the correlation between payload and launch success. Payload can be selected using a slider

GitHub URL of the completed notebook:
https://github.com/mojister/IBM_Applied-Data_Science_Capstone/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

Load the data and Create a data frame

Create a NumPy array from the column Class as target value

Standardize the data using StandardScaler

split the data X and Y into training and test data sets and use it for modeling and testing

Define parameters for the desired model and building the model

Find the best parameters of the model on train data set using GridSearch

Calculate the accuracy on the test data set and plot the confusion matrix on test set

GitHub URL of the completed notebook:
https://github.com/mojister/IBM_Applied-Data_Science_Capstone/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.ipynb

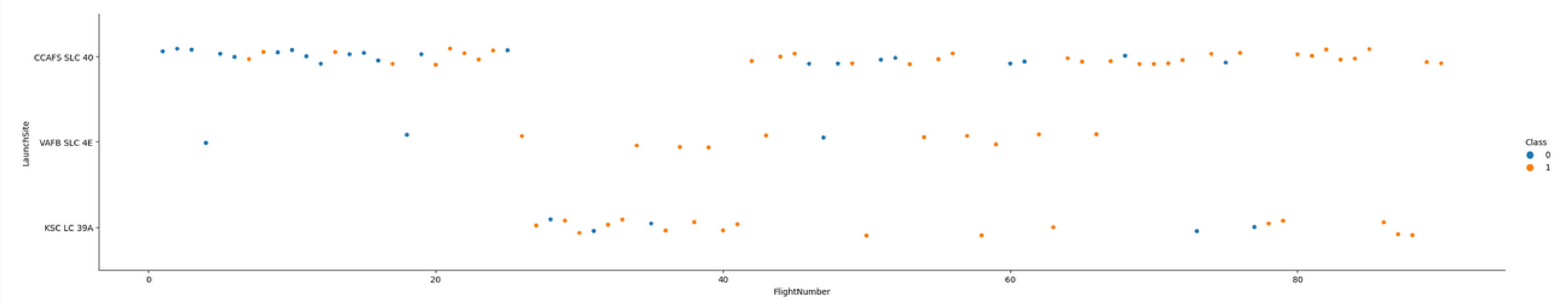
Results

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

Section 2

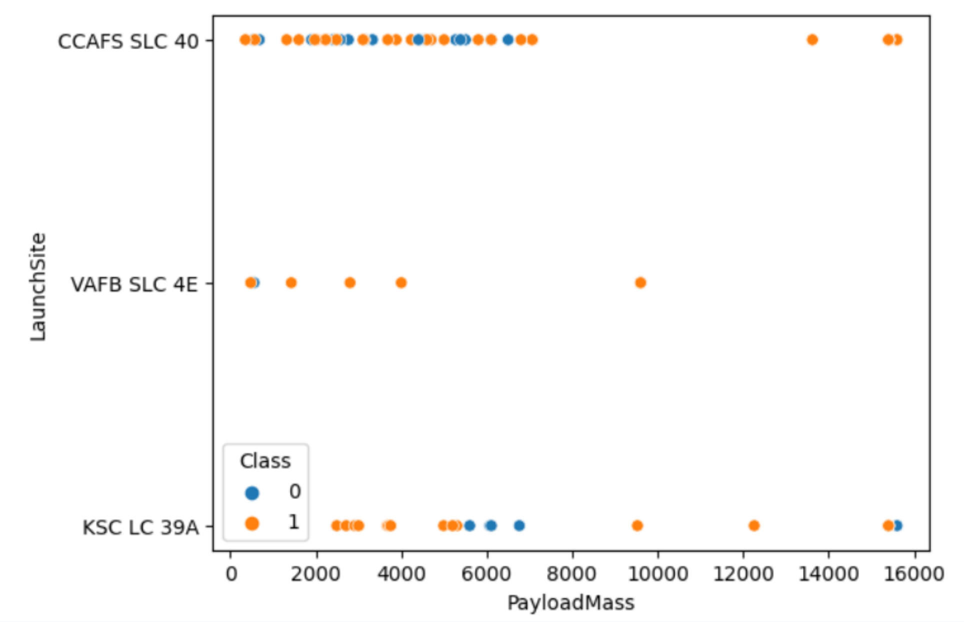
Insights drawn from EDA

Flight Number vs. Launch Site



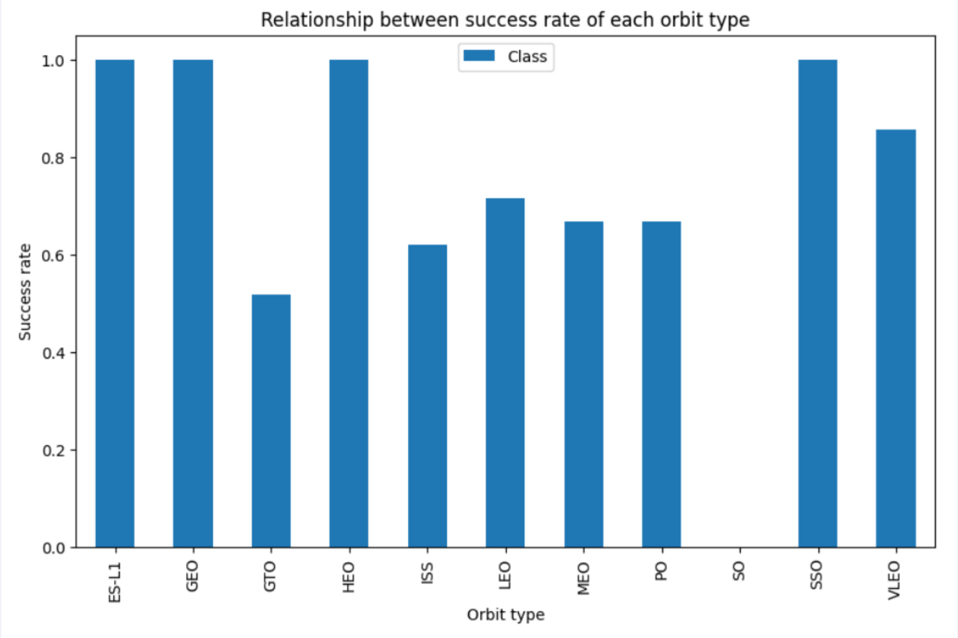
- Launch sites have different success rates
- Increasing flight number resulted more successful landings.

Payload vs. Launch Site



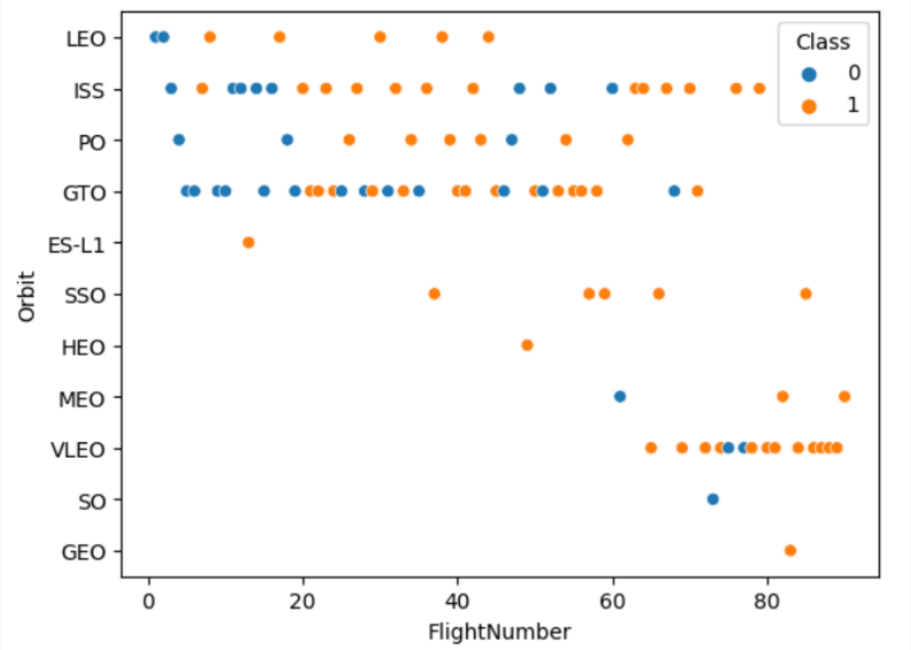
- Payload - launch site scatterplot shows that in the VAFB-SLC launch site there are no rockets launched for heavy payload mass(greater than 10000 kg)
- Light payloads (less than 5500 kg) have a good chance of successful landing on the KSC LC 39A launch site

Success Rate vs. Orbit Type



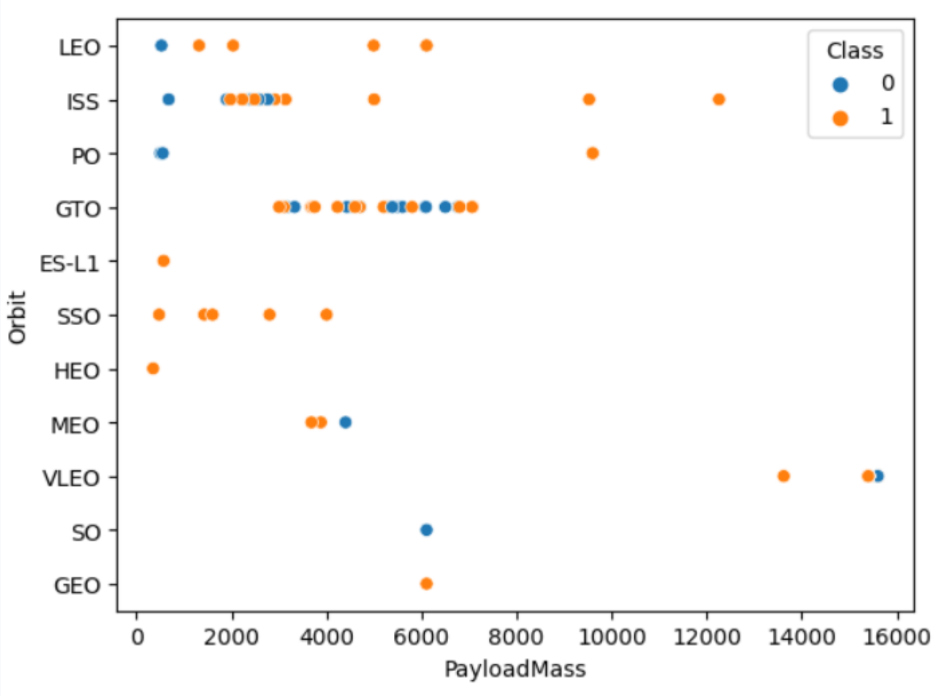
- 100% success rate for ES-L1, GEO, HEO and SSO orbit types
- 0% success rate for SO orbit type
- Good success rate (greater than 80%) for VLEO orbit type

Flight Number vs. Orbit Type



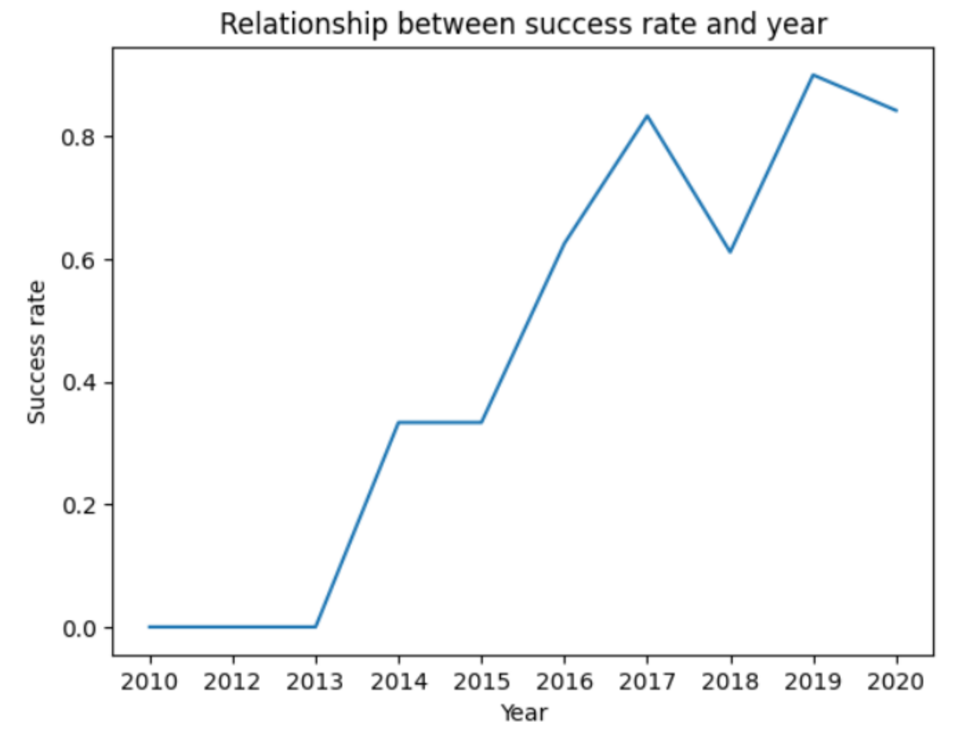
- In the LEO and VLEO orbit the success is related to the number of flights (more success rate by increasing flight number)
- No relation between flight number and success for the GTO and ISS orbits
- Nearly half of orbits have few flights

Payload vs. Orbit Type



- For heavy payloads, landing rate is more successful in Polar, LEO and ISS orbits
- Few flights above 8000kg

Launch Success Yearly Trend



- Overall trend of increase in sucess rate since 2013
- Reduction of success rate in 2018

All Launch Site Names

Group by function is used to show unique launch sites (Alternatively distinct could be used.)

```
%sql select "Launch_Site" from SPACEXTBL GROUP BY "Launch_Site";
```

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

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

Launch Site Names Begin with 'CCA'

```
%sql select * from SPACEXTBL where UPPER("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
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	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)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- LIKE is used to query for CCA sites (% wildcard must be included)
- LIMIT is used for limiting the number of records

Total Payload Mass

SUM function is used to calculate the total payload carried by boosters from NASA

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

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

SUM(PAYLOAD_MASS__KG_)
48213

One of the customers is NASA (CRS), Kacific 1. It is included in SUM.

Without it, the sum would be:

```
%sql select SUM(PAYLOAD_MASS__KG_)from SPACEXTBL where UPPER("Customer") = "NASA (CRS)";
```

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

SUM(PAYLOAD_MASS__KG_)
45596

PAYLOAD_MASS__KG_	Customer
2617	NASA (CRS), Kacific 1
1977	NASA (CRS)

Average Payload Mass by F9 v1.1

- AVG function is used to calculate the average payload for all flights for F9 v.1.1
- The result is filtered using Like function and % wildcard

```
%sql select AVG(PAYLOAD_MASS__KG_) from SPACEXTBL where UPPER("Booster_Version") like "%F9 V1.1";
```

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

AVG(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

- Although in the Hint, it is stated MIN should be used to get the first Date, it returns the last date in ipython (at least by me):

```
%sql SELECT MIN("Date") As "first_successful_landing", "Landing _Outcome" from SPACEXTBL where ("Landing _Outcome" li
```

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

first_successful_landing	Landing _Outcome
01-05-2017	Success (ground pad)

False result

- To get the first date of successful landing on a ground pad Substr and Like functions are used.

```
%sql SELECT "Date" As "first_successful_landing", "Landing _Outcome" from SPACEXTBL where "Landing _Outcome" like "%S
```

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

first_successful_landing	Landing _Outcome
22-12-2015	Success (ground pad)

Successful Drone Ship Landing with Payload between 4000 and 6000

- Like is used to query only successful landings
- (Between … and …) is used to query for payload mass between 4000 and 6000

```
%sql select "Booster_Version", PAYLOAD_MASS__KG_, "Landing _Outcome" from SPACEXTBL where (PAYLOAD_MASS__KG_ between
```

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

Booster_Version	PAYLOAD_MASS__KG_	Landing _Outcome
F9 FT B1022	4696	Success (drone ship)
F9 FT B1026	4600	Success (drone ship)
F9 FT B1021.2	5300	Success (drone ship)
F9 FT B1031.2	5200	Success (drone ship)

Total Number of Successful and Failure Mission Outcomes

- COUNT function is used to count number of missions
- Group by function is used to split the failed and successful cases

```
%sql select "Mission_Outcome", COUNT(*) As "Total_Number" from SPACEXTBL GROUP BY "Mission_Outcome";
```

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

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

Although Group by is used there are two rows with "Mission_Outcome" = "Success"

I have no explanation for this behavior

Boosters Carried Maximum Payload

- MAX function is used to query for maximum carried payload mass per booster version

```
%sql select "Booster_Version" , PAYLOAD_MASS__KG_ from SPACEXTBL where PAYLOAD_MASS__KG_ = (select Max(PAYLOAD_MASS__
```

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

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

2015 Launch Records

- List the failed landing outcomes in drone ship, their booster versions, and launch site names for in year 2015
- Substr function is used to get the month and year from the date

```
%sql select "Date", substr(Date, 4, 2) As "monthnames", "Landing _Outcome", "Booster_Version", "Launch_Site" from SPA
```

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

Date	monthnames	Landing _Outcome	Booster_Version	Launch_Site
10-01-2015	01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
14-04-2015	04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order
- count function and where clause , group by clause, order by clause and DESC (for descending order) are used. For conditioning between and like functions are used.

```
%sql SELECT "Date", "Landing _Outcome", COUNT("Landing _Outcome") AS Successful_Landing_Count from SPACEXTBL where ("
```

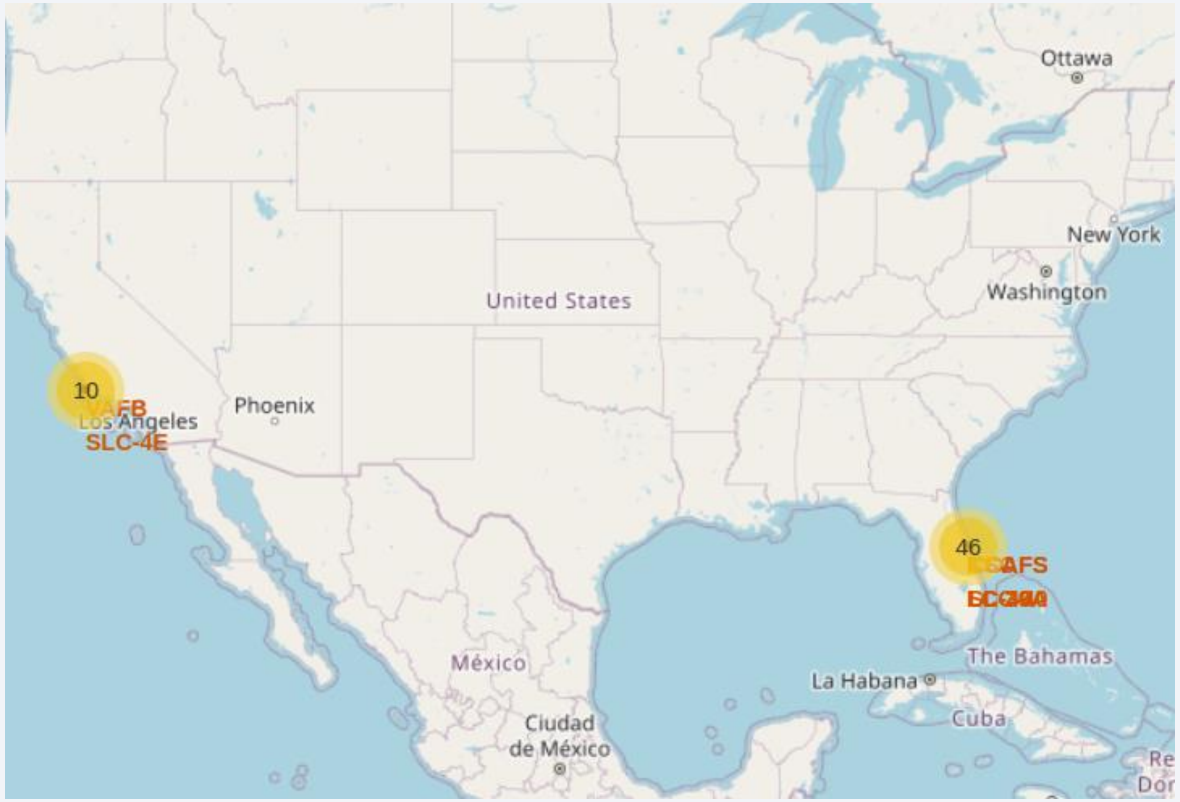
```
 * sqlite:///my_data1.db
Done.
```

Date	Landing _Outcome	Successful_Landing_Count
08-04-2016	Success (drone ship)	8
18-07-2016	Success (ground pad)	6

Section 3

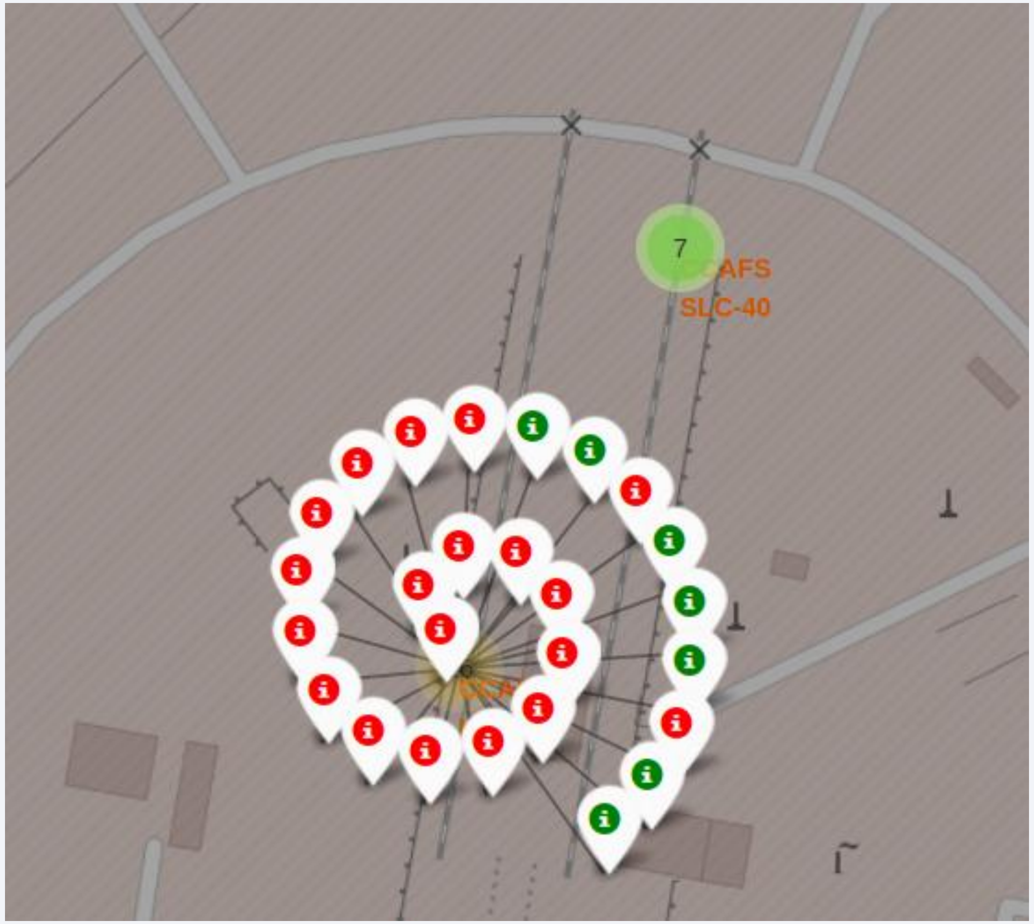
Launch Sites Proximities Analysis

Launch Sites on Map



- SpaceX launch sites are all on coast areas
- Most of the launching sites are on the eastern coast

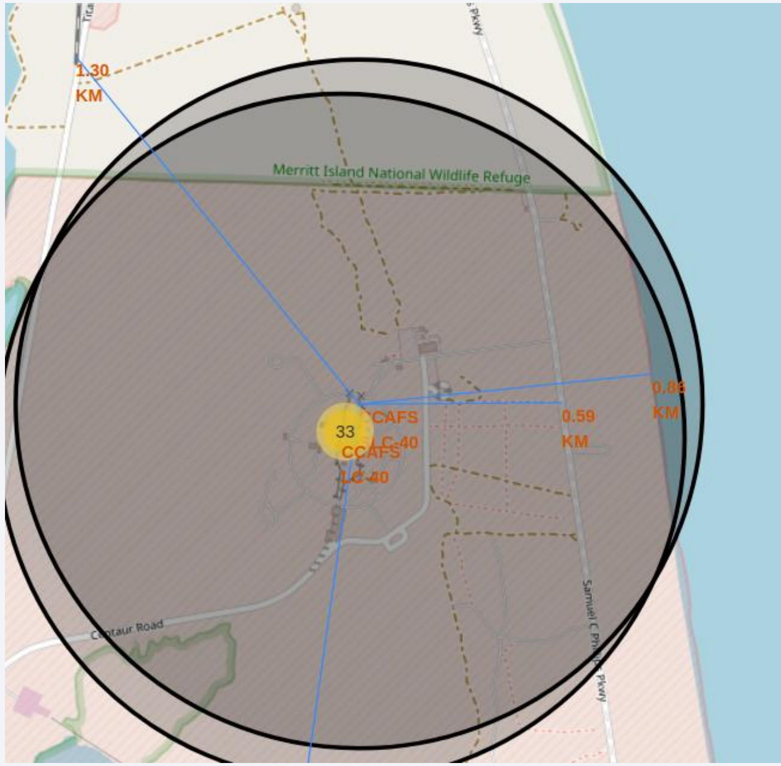
Launch Sites Outcomes

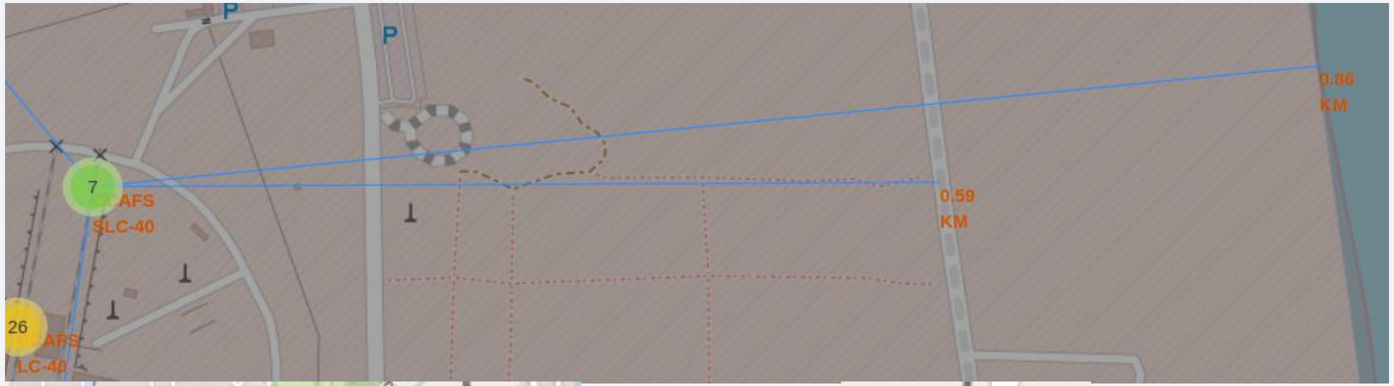


Green: successful launches

Red: failed launches

Distance of Location Sites to Proximities





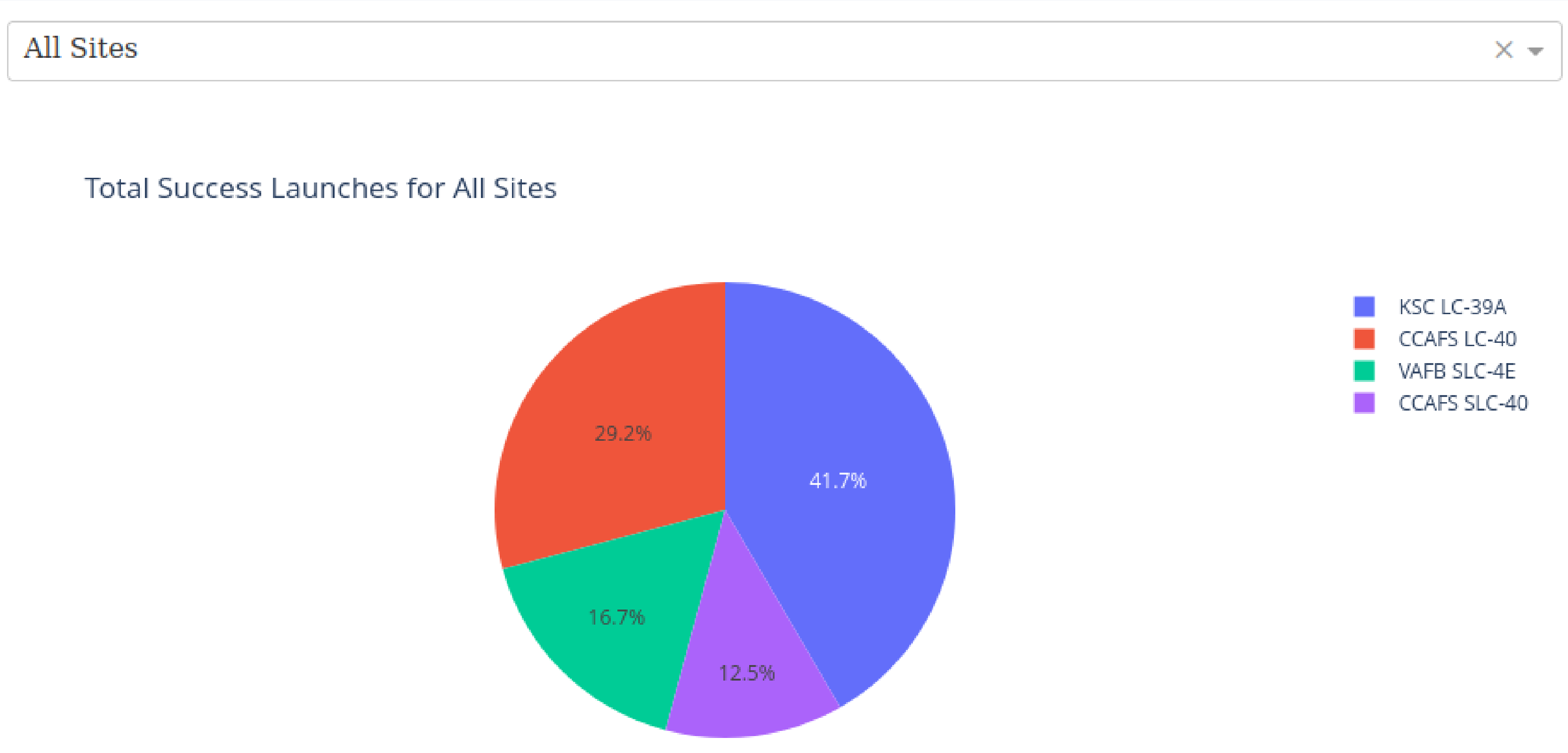
- Distance to other proximities are calculated and shown on the map
- Distance of CCAFS LC-40 launch site to the coastline is less than 1km

Section 4

Build a Dashboard with Plotly Dash

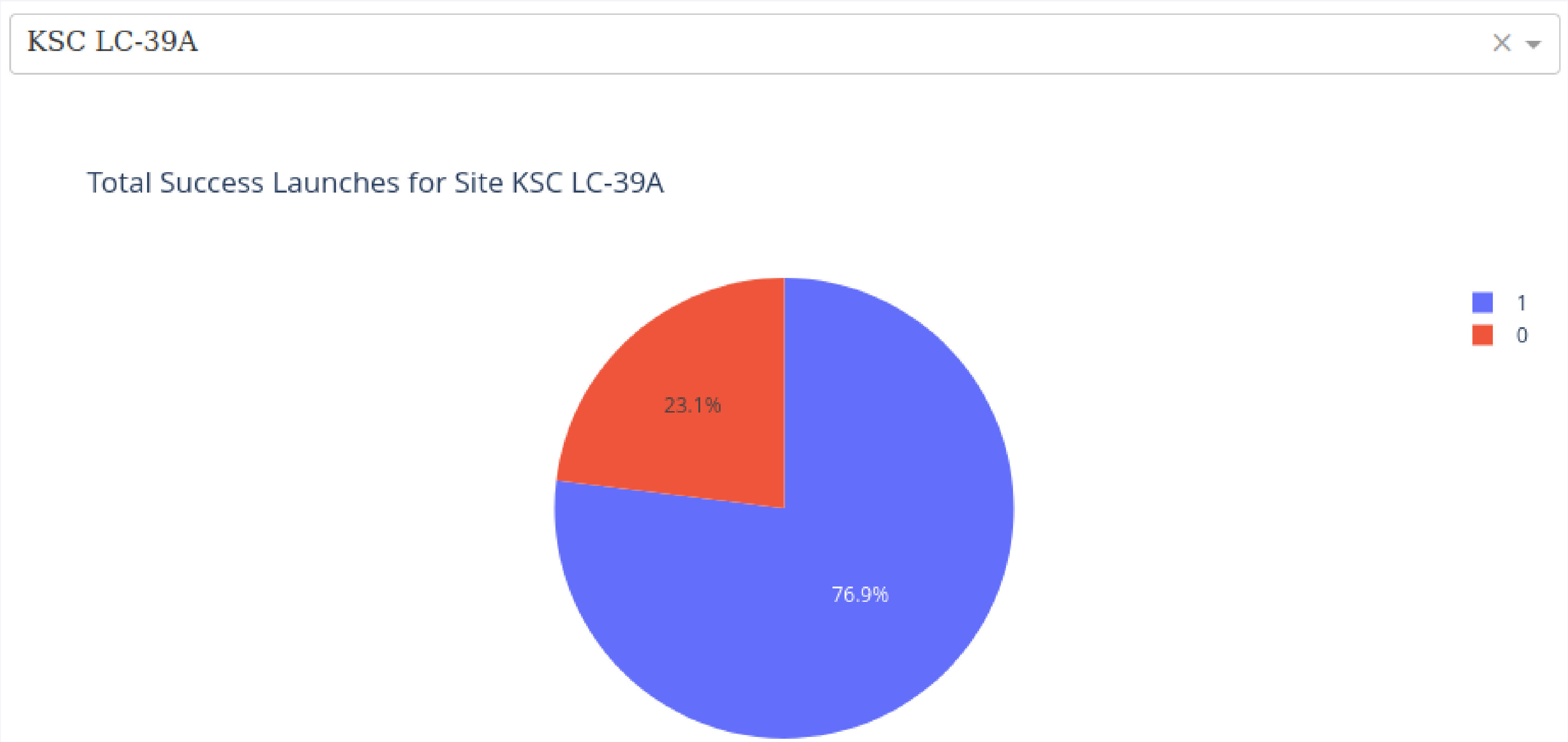
Success in Different Launch Sites

- KSC LC-39A has the highest successful launches, followed by CCAFS LC-40



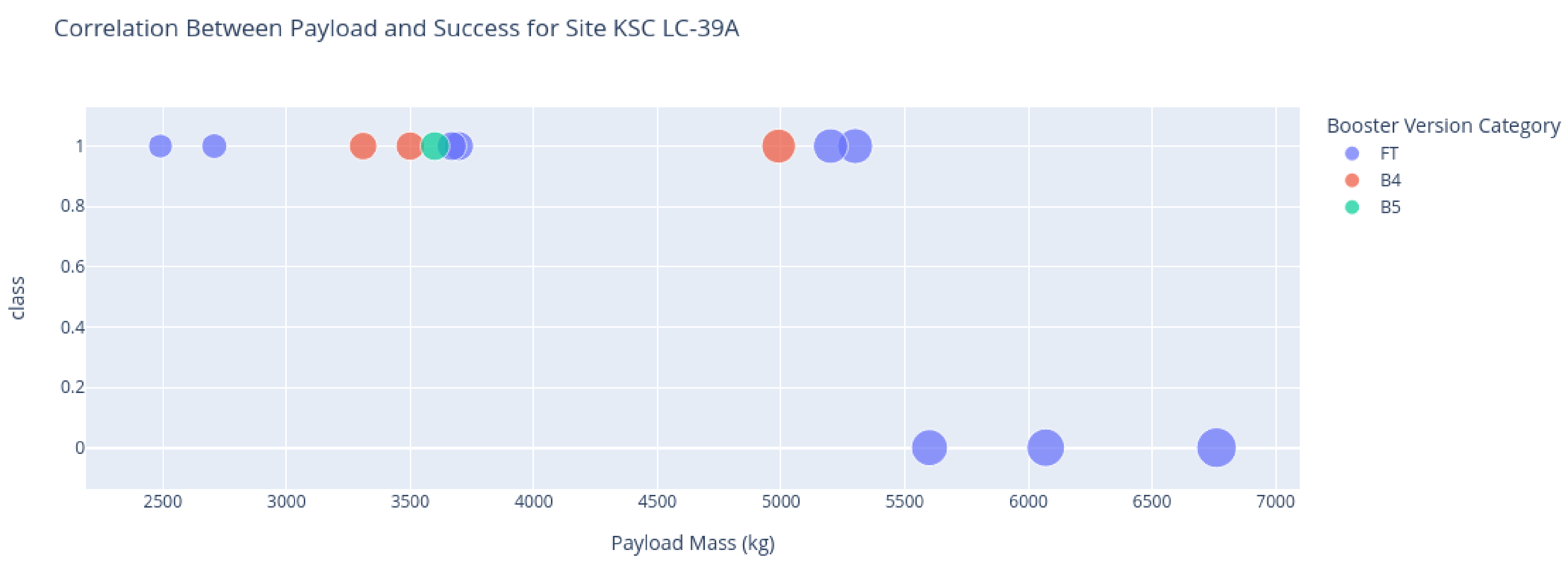
Launch Site with highest successful launches

- KSC LC-39A has 76.9% successful launches



Payload vs. Launch Outcome

- Very low success rate (7,6%) for heavy weighted payload (6001 –10000 kg)
- Highest success rate (50%) for medium weighted payload



Section 5

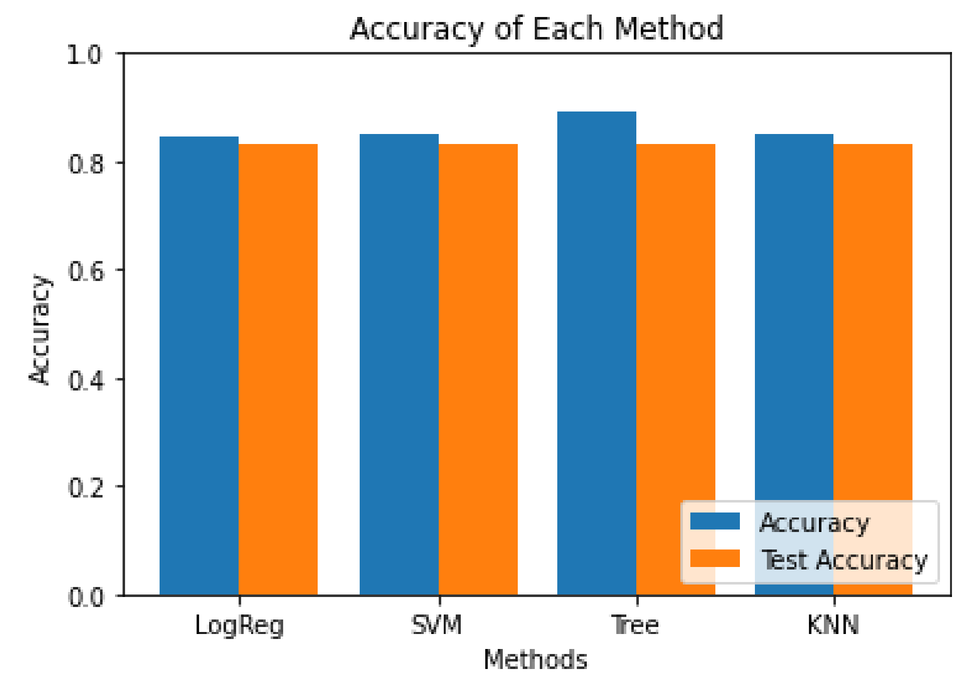
Predictive Analysis (Classification)

Classification Accuracy

Four model was used for prediction as follows:

- logistic regression
- support vector machine
- decision tree
- k nearest neighbors

	Best score	Prediction score
Logistic regresssion	0.846429	0.833333
SVM	0.848214	0.833333
Decision tree	0.900000	0.833333
KNN	0.848214	0.833333



Result of all models are comparative to each other

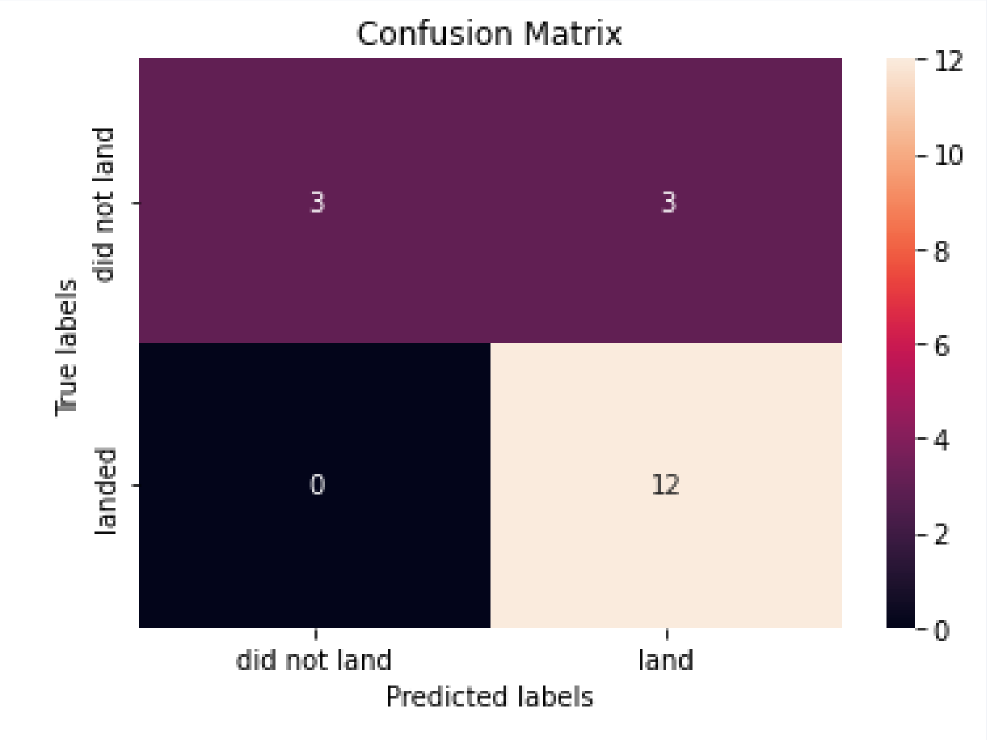
Decision tree has the best accuracy among all models

Confusion Matrix of decision tree

- 12 successful landing and 6 unsuccessful landing in data set

Decision tree confusion matrix:

- 12 successful landing correctly (true positive)
- 3 unsuccessful landing incorrectly (true negative)
- 3 unsuccessful landing incorrectly as successful landing (false positive)



Conclusions

- KSC has the highest successful launches
- Overall sucess rate kept increasing since 2013 till 2020
- SSO orbit has the highest success rate with a significant amount of data points.
- Heavy weighted payloads had lower chance of success than low weighted payloads
- All machine learning model results are very close to each other (around 83%)
- Decision tree performed a bit better than other models

Appendix

- Github link to the project repository:

https://github.com/mojister/IBM_Applied-Data_Science_Capstone

Thank you!