IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Mateusz Zatylny
09.07.2022

Outline

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

Executive Summary

- Summary of methodologies
  - Data were collected with several methods
  - EDA was made
  - Visualization analytics was made
  - Predictive analyses were made
- Summary of all results
  - All EDA results
  - Visualization analytics results
  - Predictive analyses were made

Introduction

- Project background and context

In this project, I will try to predict the probability of a successful landing of Falcon 9 a failure could be a financial kill for SpaceX

- Problems you want to find answers

1. Which factors are crucial for a successful landing?
2. How high is the rate of failure?
3. How high is the rate of success?

Section 1

Methodology

Methodology

Executive Summary

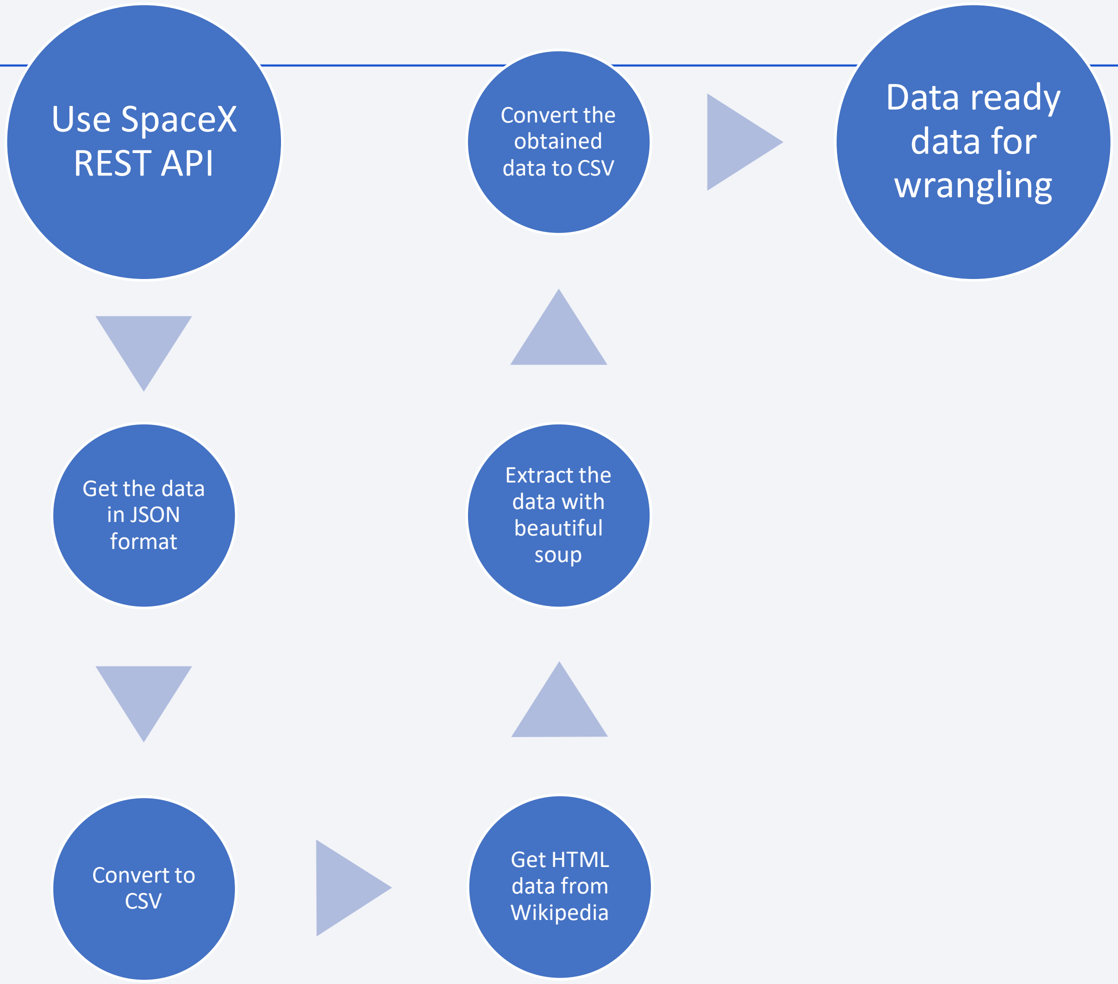
- Data collection methodology:
  - With Rest API and Web Scrapping
- Perform data wrangling
  - Data were transformed and one hot encoded to be applied later on the Machine Learning models
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Classification machine learning models were built, evaluate and the best one were chosen

Data Collection

The data were collected in the following way:

- SpaceX launch data is gathered from the SpaceX REST API. This API will give us data about launches, including information about the rocket used, payload delivered, launch specifications, landing specifications, and landing outcome.

- Another popular data source for obtaining Falcon 9 Launch data is web scraping Wikipedia using BeautifulSoup.



Data Collection – SpaceX API

1. Get the datasets from SpaceX API
2. Save it in JSON format
3. Convert to CSV file

[IBM-SpaceX-Project/jupyter-labs-spacex-data-collection-api.ipynb at main · mzatylny/IBM-SpaceX-Project (github.com)]

From the `rocket` column we would like to learn the booster name.

```
In [2]: # Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

From the `launchpad` we would like to know the name of the launch site being used, the logitude, and the latitude.

```
In [3]: # Takes the dataset and uses the launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['launchpad']:
        response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
        Longitude.append(response['longitude'])
        Latitude.append(response['latitude'])
        LaunchSite.append(response['name'])
```

From the `payload` we would like to learn the mass of the payload and the orbit that it is going to.

```
In [4]: # Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

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)`

```
In [30]: data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection - Scraping

1. Get HTML data from Wikipedia

2. Extract the data with beautiful soup

3. Convert the obtained data to CSV

[IBM-SpaceX-Project/jupyter-labs-webscraping.ipynb at main · mzatylny/IBM-SpaceX-Project (github.com)]()

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

In [6]:
```
!pip3 install html5lib
```
```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: html5lib in c:\users\mateusz zatylny\appdata\roaming\python\python39\site-packages (1.1)
Requirement already satisfied: webencodings in c:\programdata\anaconda3\lib\site-packages (from html5lib) (0.5.1)
Requirement already satisfied: six>=1.9 in c:\programdata\anaconda3\lib\site-packages (from html5lib) (1.16.0)
```

In [7]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response, 'html5lib')
```

Print the page title to verify if the `BeautifulSoup` object was created properly

In [8]:
```
# Use soup.title attribute
soup.title
```

Out[8]:
```
<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 reference link towards the end of this lab

In [9]:
```
# 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.

In [10]:
```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

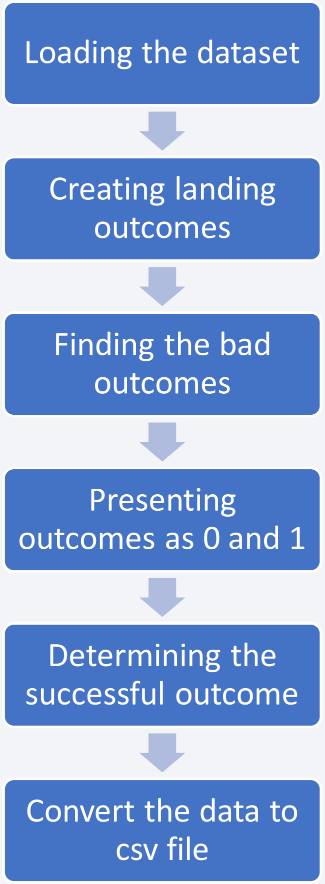
We can now export it to a **CSV** for the next section, but to make the answers consistent and in case you have difficulties finishing this lab.

Following labs will be using a provided dataset to make each lab independent.

In [16]:
```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Data Wrangling

[IBM-SpaceX-Project/labs-jupyter-spacex-Data wrangling.ipynb at main · mzatylny/IBM-SpaceX-Project (github.com)]



EDA with Data Visualization

1. Plot of the relationship between Flight Number and Launch Site
2. Plot of the relationship between Payload and Launch Site
3. Plot of the relationship between success rate of each orbit type
4. Plot of the relationship between FlightNumber and Orbit type
5. Plot of the relationship between Payload and Orbit type
6. Plot of the launch success yearly trend

It helps to find the crucial factors for a successful mission

IBM-SpaceX-Project/jupyter-labs-eda-dataviz.ipynb at main · mzatylny/IBM-SpaceX-Project (github.com)

EDA with SQL

SQL queries performed include:

- Displaying the names of the unique launch sites in the space mission
- Displaying 5 records where launch sites begin with the string 'KSC'
- Displaying the total payload mass carried by boosters launched by NASA (CRS)
- Displaying average payload mass carried by booster version F9 v1.1
- Listing the date where the successful landing outcome in drone ship was achieved.
- Listing the names of the boosters which have success in ground pad and have payload mass greater than 4000

but less than 6000

- Listing the total number of successful and failure mission outcomes
- Listing the names of the booster_versions which have carried the maximum payload mass.
- Listing the records which will display the month names, successful landing_outcomes in ground pad, booster versions, launch_sitefor the months in year 2017
- Ranking the count of successful landing_outcomes between the date 2010 06 04 and 2017 03 20 in descending order.

[IBM-SpaceX-Project/jupyter-labs-eda-sql-coursera_sqllite.ipynb at main · mzatylny/IBM-SpaceX-Project (github.com)]

Build an Interactive Map with Folium

- folium.Marker() was used to create marks on the maps.
- folium.Circle() was used to create a circles above markers on the map.
- folium.Icon() was used to create an icon on the map.
- folium.PolyLine() was used to create polynomial line between the points.
- folium.plugins.AntPath() was used to create animated line between the points.
- markerCluster() was used to simplify the maps which contain several markers

with identical coordination.

[IBM-SpaceX-Project/lab_jupyter_launch_site_location.ipynb at main · mzatylny/IBM-SpaceX-Project (github.com)]

Build a Dashboard with Plotly Dash

- Dash and HTML components were used as they are the most important steps and almost everything depends on them, such as graphs, tables, dropdowns, etc.
- Pandas was used to simplify the work by creating a dataframe.
- Plotly was used to plot the graphs.
- Pie chart and scatter chart were used for plotting purposes.
- Rangeslider was used for payload mass range selection.
- Dropdown was used for launch sites.

IBM-SpaceX-Project/spacex_dash_app.py at main · mzatylny/IBM-SpaceX-Project (github.com)

Predictive Analysis (Classification)

[IBM-SpaceX-Project/SpaceX_Machine Learning Prediction_Part_5.ipynb at main · mzatylny/IBM-SpaceX-Project (github.com)]

Building a model

- Create a column for the class
- Standardize the data
- Split the data info train and test sets
- Build GridSearchCV model and fit the data

Evaluating the model

- Calculating the accuracies
- Calculating the confusion matrixes
- Plot the results

Finding the optimal model

- Find the best hyperparameters for the models
- Find the best model with the highest accuracy
- Acquire the optimal model

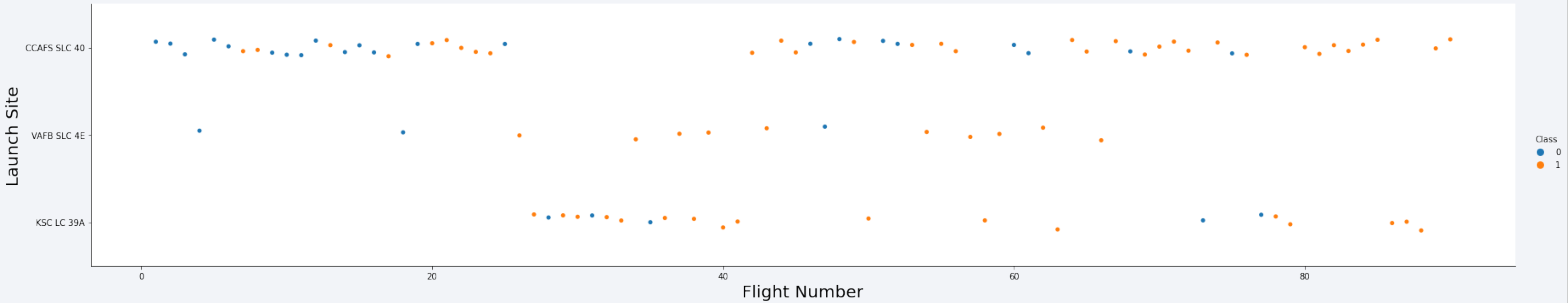
Results

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

Section 2

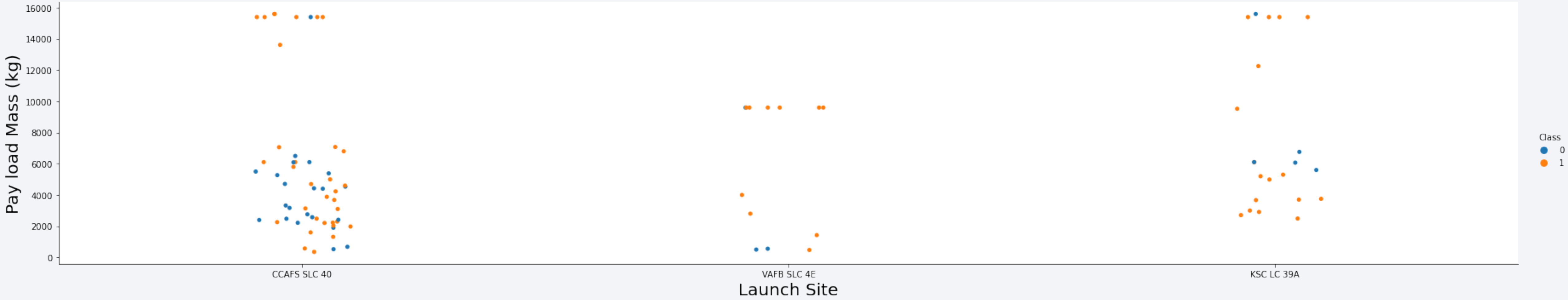
Insights drawn from EDA

Flight Number vs. Launch Site



From the plot, we observe that the larger the flight amount at a launch site, the greater the success rate at a launch site.

Payload vs. Launch Site

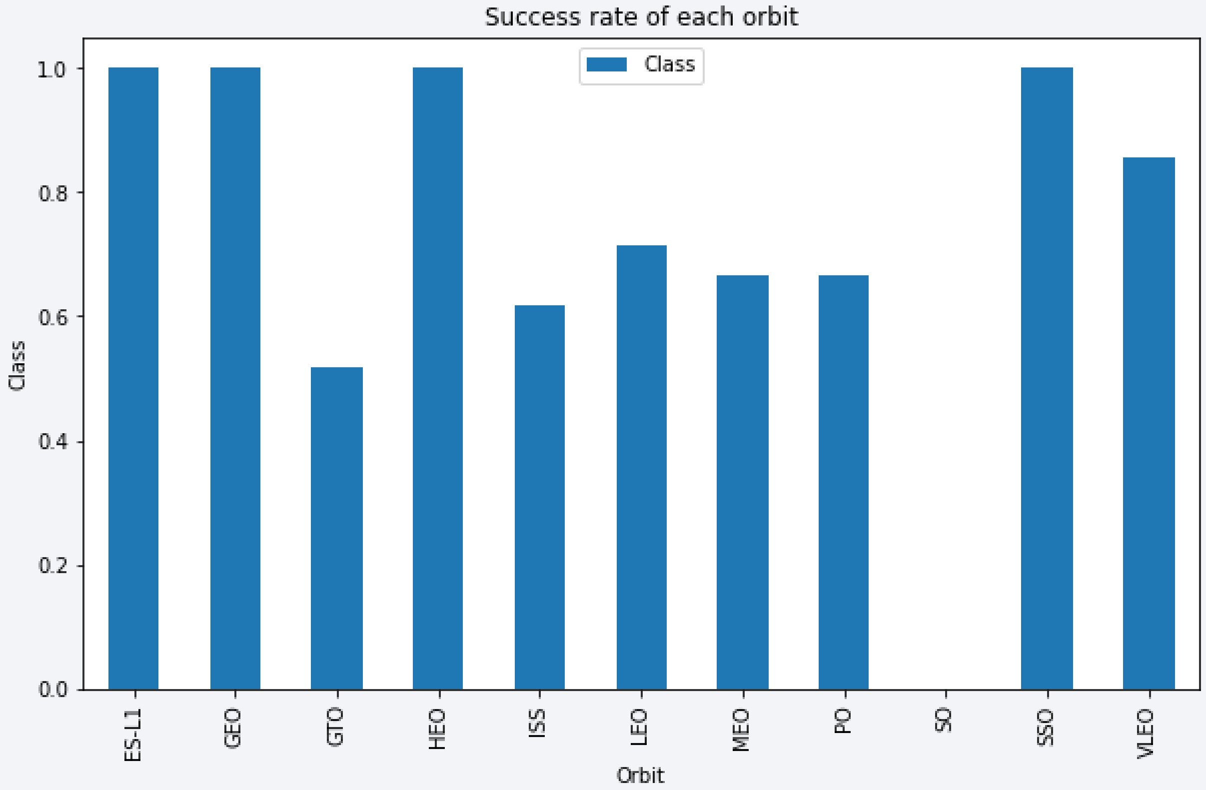


With the increase in Payload Mass, the success rate is increasing as well in the launch sites

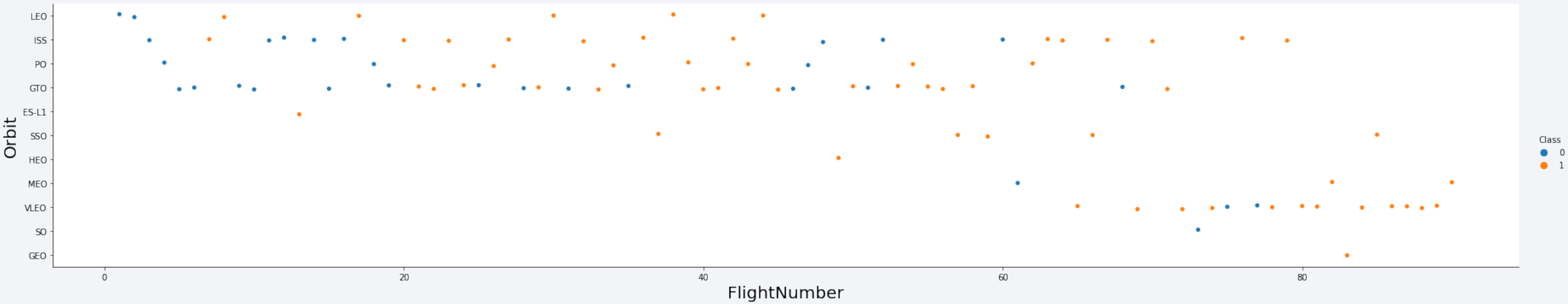
Success Rate vs. Orbit Type

ES-L1, GEO, HEO, and SSO have a success rate of 100%

SO has a success rate of 0%

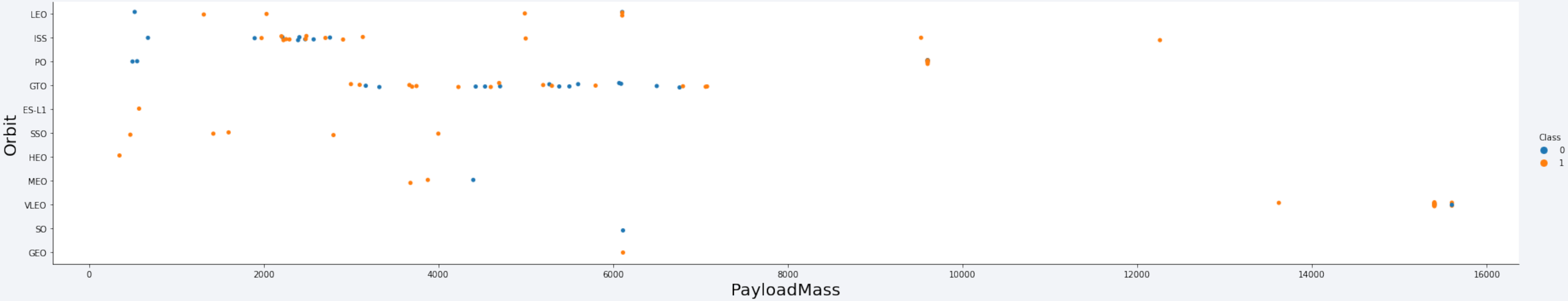


Flight Number vs. Orbit Type



The plot above shows the Flight Number vs. Orbit type. We observe that in the LEO orbit, success is related to the number of flights whereas, in the GTO orbit, there is no relationship between flight number and the orbit.

Payload vs. Orbit Type



The first thing to see is how the Payload Mass between 2000 and 3000 involves ISS.

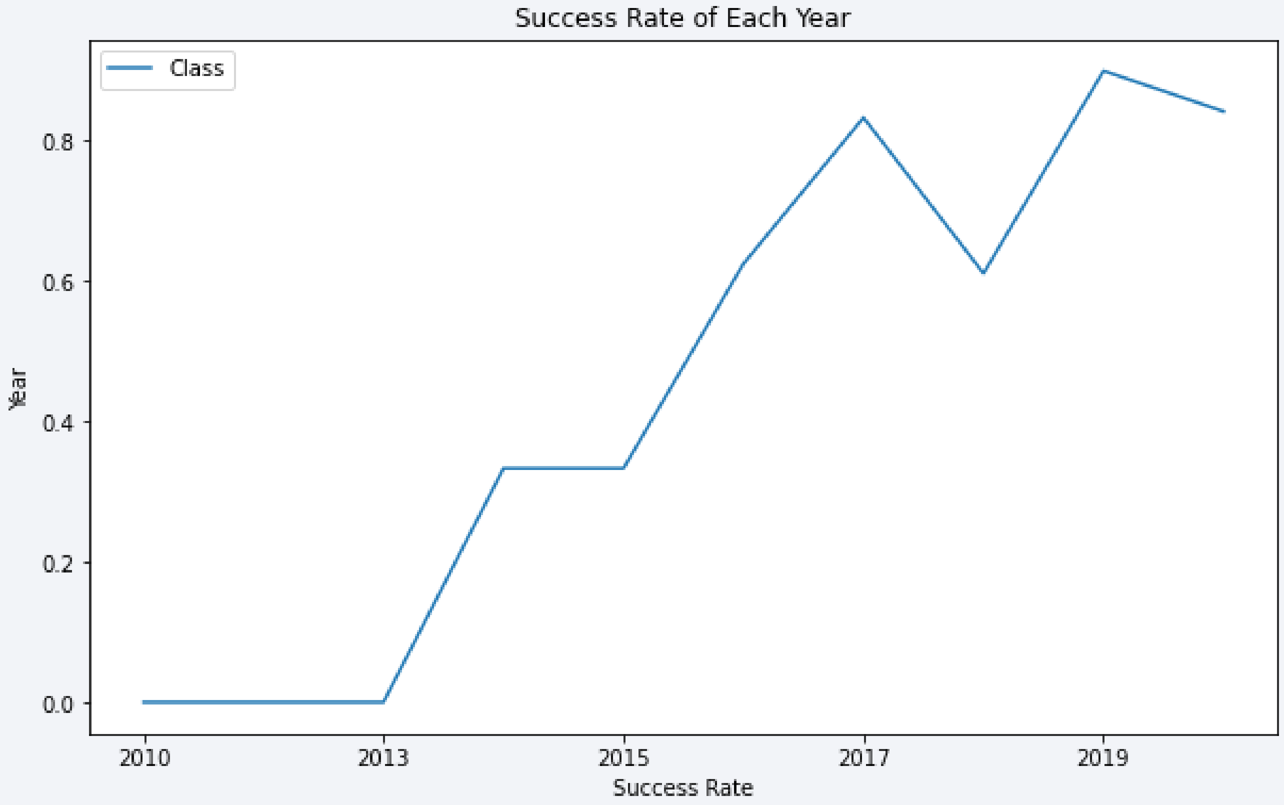
Similarly, Payload Mass between 3000 and 7000 involves GTO.

Launch Success Yearly Trend

Since the year 2013, there was

a massive increase in the success rate. However, it dropped a little

in 2018 but later it got stronger than before.



All Launch Site Names

We can get the unique values by using "DISTINCT"

Display the names of the unique launch sites in the space mission

```
In [7]: %sql select distinct LAUNCH_SITE from SPACEXTBL

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

Out[7]:

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

Launch Site Names Begin with 'CCA'

We can get 5 rows by using “LIMIT”

Display 5 records where launch sites begin with the string 'CCA'

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

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

Out[8]:

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

Total Payload Mass

We can get the sum of all values by using "SUM"

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

```
In [9]: %sql select sum(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL;

 * sqlite:///my_data1.db
Done.

Out[9]:
```

payloadmass
619967

Average Payload Mass by F9 v1.1

We can get the average of all values by using “AVG”

Display average payload mass carried by booster version F9 v1.1

```
In [10]: %sql select avg(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL;

 * sqlite:///my_data1.db
Done.

Out[10]:
```

payloadmass
6138.287128712871

First Successful Ground Landing Date

We can get the first successful data by using “MIN” IMPORTANT DON’T FORGET []

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

```
In [21]: %sql select min(DATE) from SPACEXTBL where [Landing _Outcome] = 'Success (ground pad)';

 * sqlite:///my_data1.db
Done.

Out[21]:
```

min(DATE)
01-05-2017

Successful Drone Ship Landing with Payload between 4000 and 6000

The payload mass data was taken between 4000 and 6000 only, and the landing outcome was determined to be a “success drone ship” important []

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

In [12]:
```
%sql select BOOSTER_VERSION from SPACEXTBL where [Landing _Outcome] ='Success (drone ship)' and PAYLOAD_MASS__KG_ BETWEEN 4000 AN
```

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

Out[12]:

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

Total Number of Successful and Failure Mission Outcomes

Two separate queries:

We can get the number of all the successful mission by using “COUNT” and LIKE “Success%”

We can get the number of all the failure mission by using “COUNT” and LIKE “Failure%”

List the total number of successful and failure mission outcomes

```
In [29]: %sql select count(MISSION_OUTCOME) AS 'Successful Mission' from SPACEXTBL where MISSION_OUTCOME like 'Success%';

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

Out[29]:

Successful Mission
100

```
In [30]: %sql select count(MISSION_OUTCOME) AS 'Failure Mission' from SPACEXTBL where MISSION_OUTCOME like 'Failure%';

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

Out[30]:

Failure Mission
1

Boosters Carried Maximum Payload

We can get the maximum payload masses by using “MAX”

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

In [14]:

```
%sql select BOOSTER_VERSION as boosterversion from SPACEXTBL where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACEXTE
```

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

Out[14]:

boosterversion
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

2015 Launch Records

Unfortunately, I have got an error – don’t know how to fix it

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.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

In [33]:
```
%sql SELECT month(DATE) as "Month", BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE year(DATE) = '2015' AND \
[LANDING _OUTCOME] = 'Failure (drone ship)';
```

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such function: month
[SQL: SELECT month(DATE) as Month , BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE year(DATE) = '2015' AND [LANDING _OUTCOM
E] = 'Failure (drone ship)' ;]
(Background on this error at: http://sqlalche.me/e/e3q8)
```

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

The output is wrong I don’t know how to fix it

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

In [35]:
```
%sql SELECT [LANDING _OUTCOME] as 'Landing Outcome', COUNT([LANDING _OUTCOME]) AS 'Total Count' FROM SPACEXTBL \
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
GROUP BY  [LANDING _OUTCOME] \
ORDER BY COUNT([LANDING _OUTCOME]) DESC ;
```

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

Out[35]:

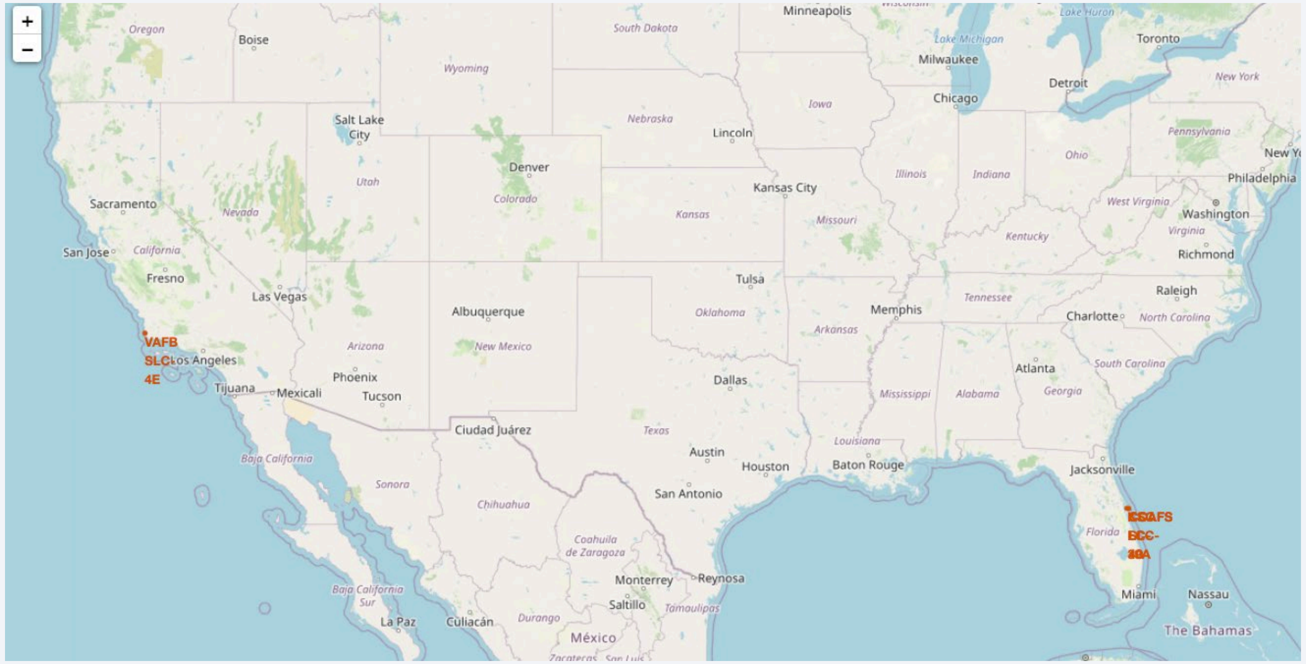
Landing Outcome	Total Count

Section 3

Launch Sites Proximities Analysis

All Launch Sites’ Location Markers

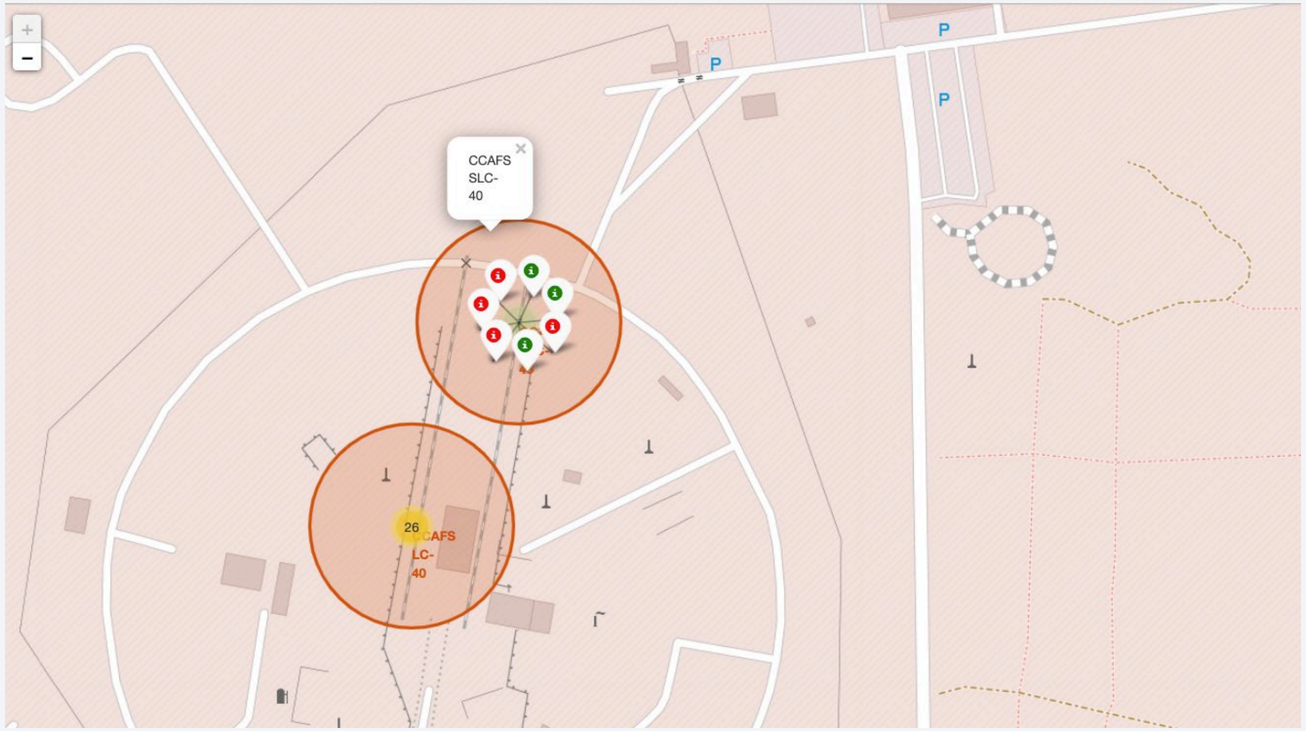
All the launches are in Florida and California



Color-labeled Launch Outcomes

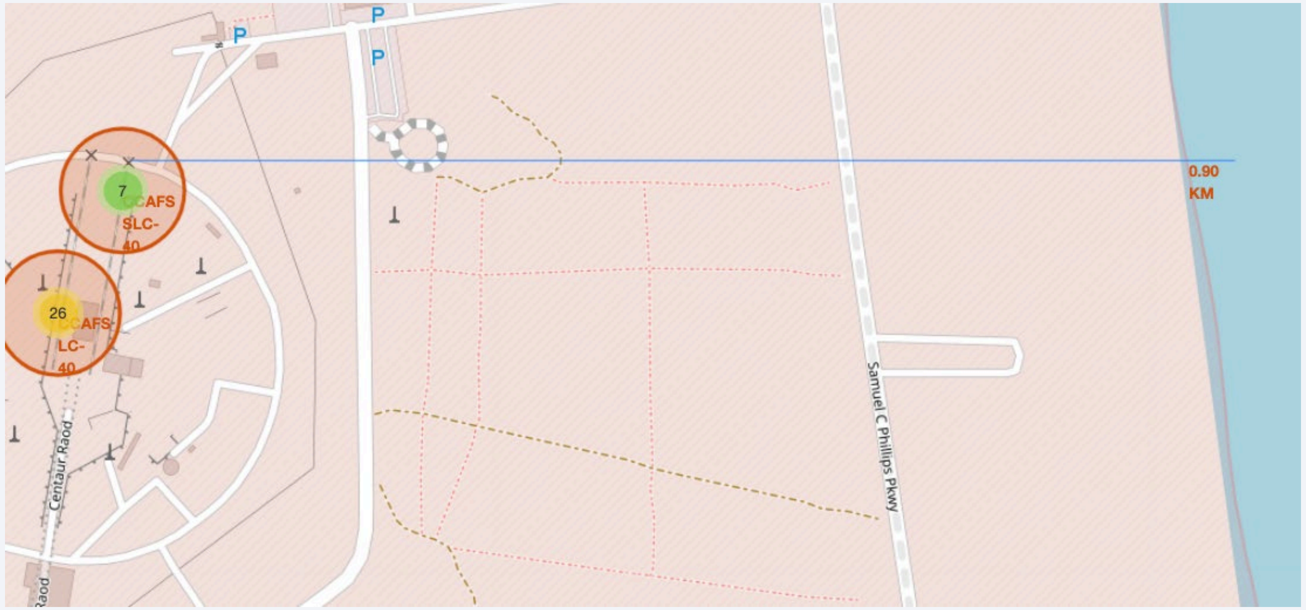
Green means successful

Red means failure



Distances between a launch site to its proximities

All distances from launch sites to its proximities, they weren’t far from the railway tracks.



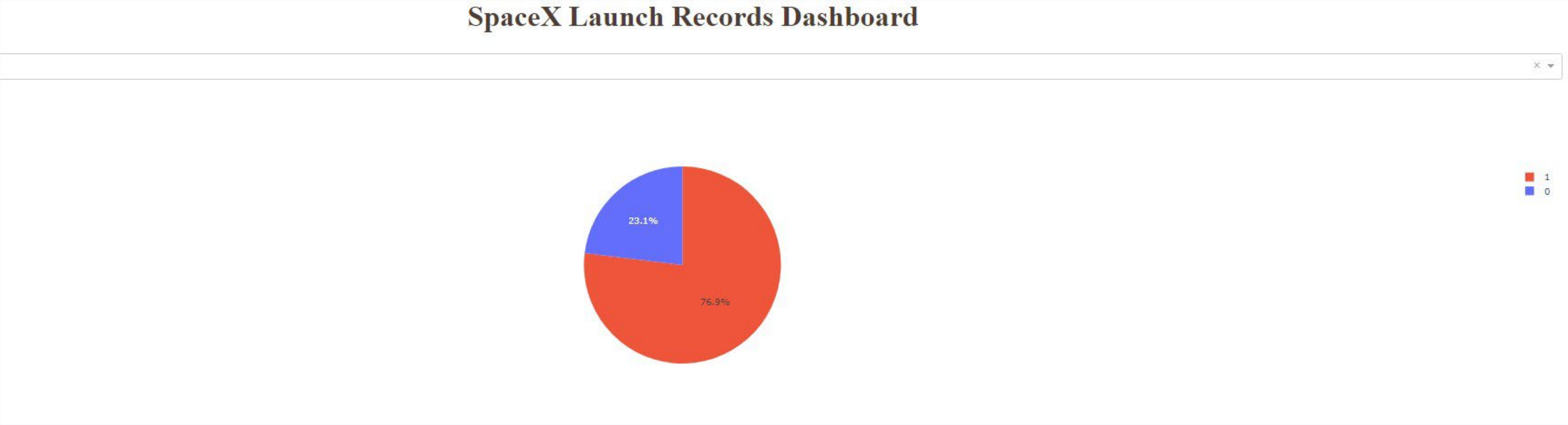
Section 4

Build a Dashboard with Plotly Dash

Total success launches by all sites

KSC LC-39A has the highest success score with 41.7%
CCAFS LC-40 comes next with 29.2%
Finally, VAFB SLC-4E and CCAFS SLC-40 with 16.7% and 12.5% respectively

Success rate by site



KSC LC-39A has the highest score with 76.9% with a payload range of 2000 kg – 10000 kg, and the FT booster version has the highest score

Payload vs launch outcome

The low weighted payloads have higher success rates

Section 5

Predictive Analysis (Classification)

Classification Accuracy

The best model is the decision three model

Out[34]:

	Algorithm	Accuracy
0	Logistic Regression	0.846429
1	SVM	0.848214
2	KNN	0.848214
3	Decision Tree	0.876786

Confusion Matrix

The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives. i.e., an unsuccessful landing is marked as a successful landing by the classifier.

Conclusions

- KSC LC 39A had the most successful launches from all the sites.
- Low weighted payloads perform better than heavier payloads.
- The decision tree classifier is the best machine learning algorithm for this task.

Appendix

All codes can be found here:

[mzatylny/IBM-SpaceX-Project (github.com)]

Thank you!