IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Batuhan Umay>
<22.01.2022>

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection
  - Data Wrangling
  - EDA with Data Visualization
  - EDA with SQL
  - Interactive map with Folium
  - Dashboard with Plotly Dash
  - Predictive Analysis (Classification)
- Summary of all results
  - EDA results
  - Interactive maps and dashboard
  - Predictive results

Introduction

- Project background and context
  - Space X advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because Space X can reuse the first stage.
- Problems you want to find answers
  - The project task is to predict if the first stage of the SpaceX Falcon 9 rocket will land successfully or not.

Section 1

Methodology

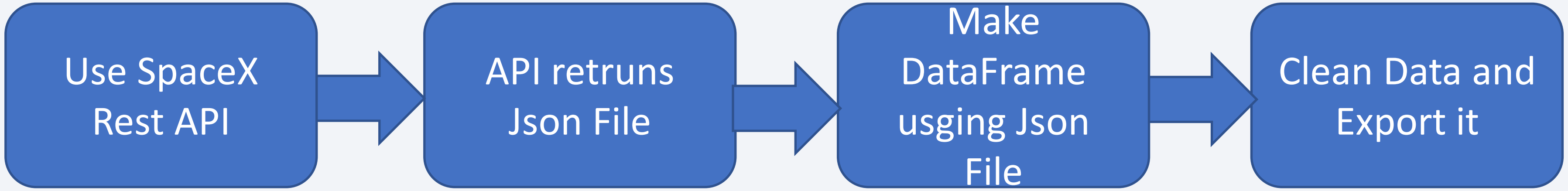
Methodology

Executive Summary

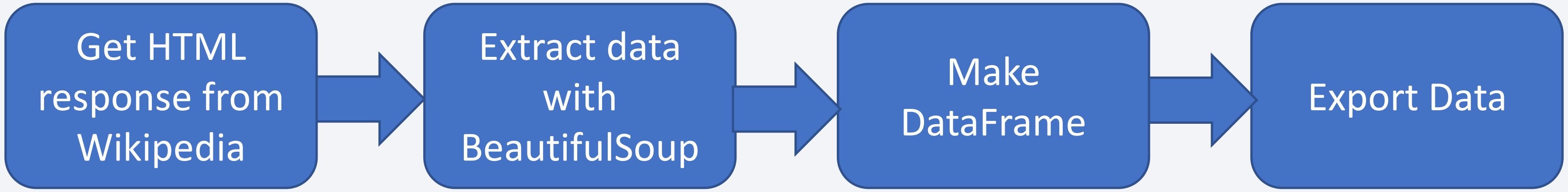
- Data collection methodology:
  - SpaceX Rest API
  - Web Scrapping from Wikipedia
- Perform data wrangling
  - Dropping unnecessary columns
  - One Hot Encoding for classification models
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Logistic Regression, KNeighbors Classifier (KNN), Support Vector Classifier (SVM), Decision Tree Classifier

Data Collection

- Describe how data sets were collected.
  - Datasets are collected from SpaceX Rest API and Webscrapping from Wikipedia.
- You need to present your data collection process use key phrases and flowcharts
  - The SpaceX Rest API url: https://api.spacexdata.com/v4/



  - WebScrapping from Wikipedia url: https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922



Data Collection – SpaceX API

1- Getting response from SpaceX Rest API

```
In [82]: spacex_url="https://api.spacexdata.com/v4/launches/past"

In [83]: response = requests.get(spacex_url)
```

2- Decode response content as a json using .json() then turn it into a pandas dataframe using .json_normalize()

```
In [87]: # Use json_normalize meethod to convert the json result into a dataframe
         context = response.json()
         data = pd.json_normalize(context)
```

3- Transform Data

Now, let's apply `getBoosterVersion` function method to get the booster version

```
In [93]: # Call getBoosterVersion
         getBoosterVersion(data)
```

the list has now been update

```
In [94]: BoosterVersion[0:5]

Out[94]: ['Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 9']
```

we can apply the rest of the functions here:

```
In [95]: # Call getLaunchSite
         getLaunchSite(data)

In [96]: # Call getPayloadData
         getPayloadData(data)

In [97]: # Call getCoreData
         getCoreData(data)
```

- [GitHub Link:]()

4- Assing list to dictionary then pandas dataframe

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

```
In [98]: 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 [99]: # Create a data from launch_dict
         df = pd.DataFrame(launch_dict)
```

5- Use filter dataframe

```
In [111…  # Hint data['BoosterVersion']!='Falcon 1'
          data_falcon9 = df[df['BoosterVersion'] != 'Falcon 1']
```

6- Export it

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

Data Collection - Scraping

- [GitHub Link:]()

1- Getting Response from Html Url

```
In [70]: static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

Next, request the HTML page from the above URL and get a `response` object

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

```
In [71]: # use requests.get() method with the provided static_url
         # assign the response to a object
         responce = requests.get(static_url)
```

2- Creating Beautiful Soup object

```
In [72]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
         soup = BeautifulSoup(responce.text)
```

3- Finding all tables

```
In [74]: # 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')
```

4- Getting column names

```
[76]: column_names = []

      # for i, col in enumerate(first_launch_table.find_all('th')):
      for i in first_launch_table.find_all('th'):

          name = extract_column_from_header(i)

          if name is not None and len(name) > 0:
              column_names.append(name)

      # 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
```

5- Creating dictionary

```
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']=[]
```

6- Add data to keys

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

7- Creating dataframe from dictionary

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

8- Export file as csv

```
df.to_csv('spacex_web_scraped.csv', index = False)
```

Data Wrangling

- [GitHub Link:]()

- In the dataset, there are several cases where the booster didn't land successully.
  - True Ocean, True RTLS, True ASDS means the mission has been successful.
  - False Ocean, False RTLS, False ASDS means the mission was a failure.
- We need to transform string variables into categorical variables where 1 means the mission has been successful and 0 means the mission was a failure.

1- Calculate number of each launch sites

```
# Apply value_counts() on column LaunchSite
df.value_counts("LaunchSite")
```

```
LaunchSite
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
dtype: int64
```

2- Calculate number of each orbit

```
# Apply value_counts on Orbit column
df.value_counts("Orbit")
```

```
Orbit
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
GEO       1
HEO       1
SO        1
dtype: int64
```

3- Calculate number of landing outcomes

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

```
True ASDS      0.455556
None None      0.211111
True RTLS      0.155556
False ASDS     0.066667
True Ocean     0.055556
False Ocean    0.022222
None ASDS      0.022222
False RTLS     0.011111
Name: Outcome, dtype: float64
```

4-Create Class columns and assing 1 or 0 from landing outcome

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise

landing_class = [1 if i not in bad_outcomes else 0 for i in df["Outcome"]]
```

This variable will represent the classification variable that represents the outcome of ea

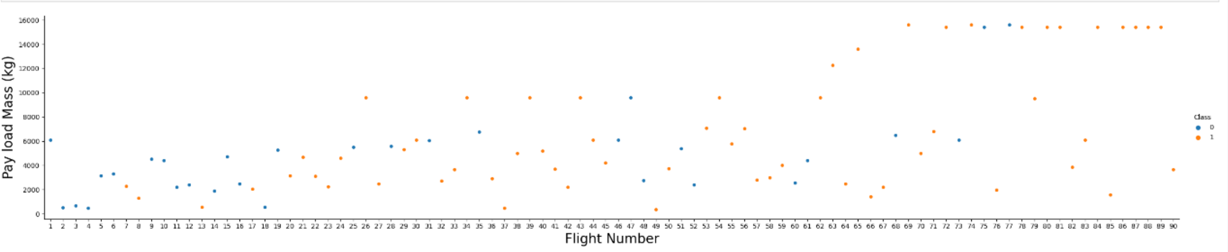
```
df['Class']=landing_class
```

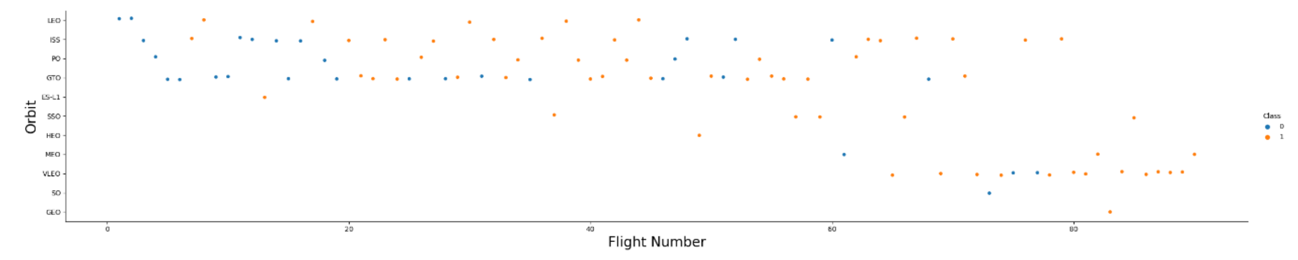
5- Export file

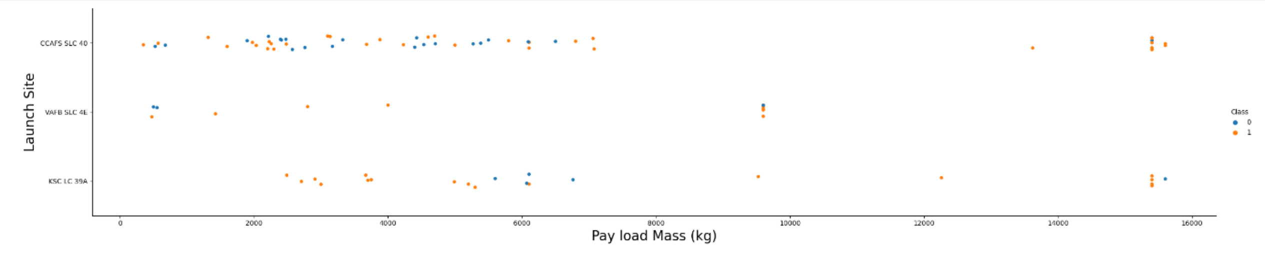
```
df.to_csv("dataset_part2.csv", index = False)
```

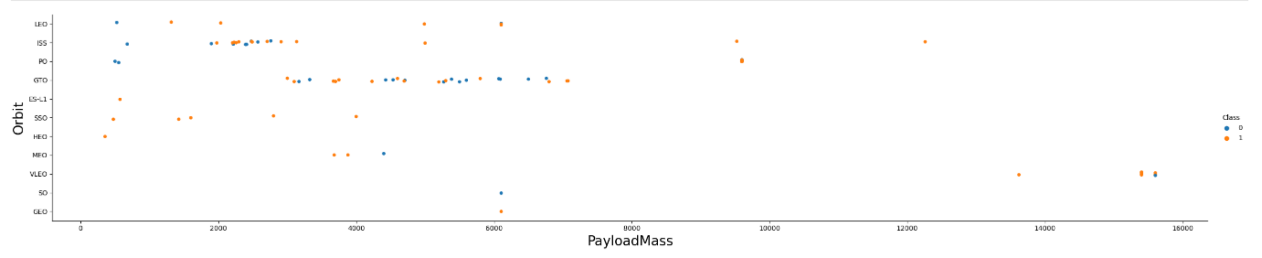
EDA with Data Visualization

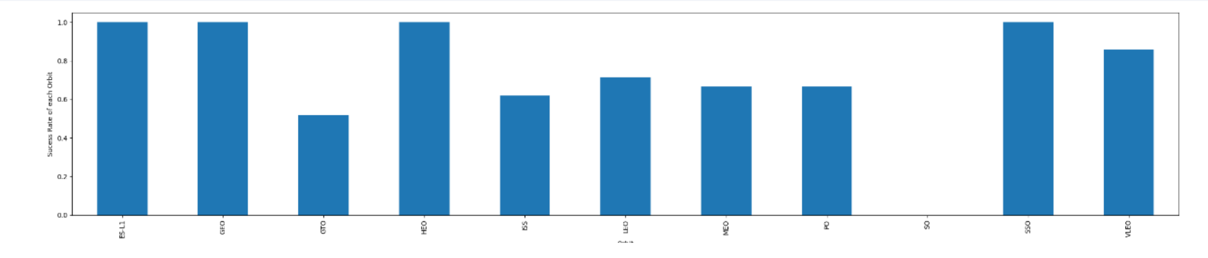
GitHub Link:

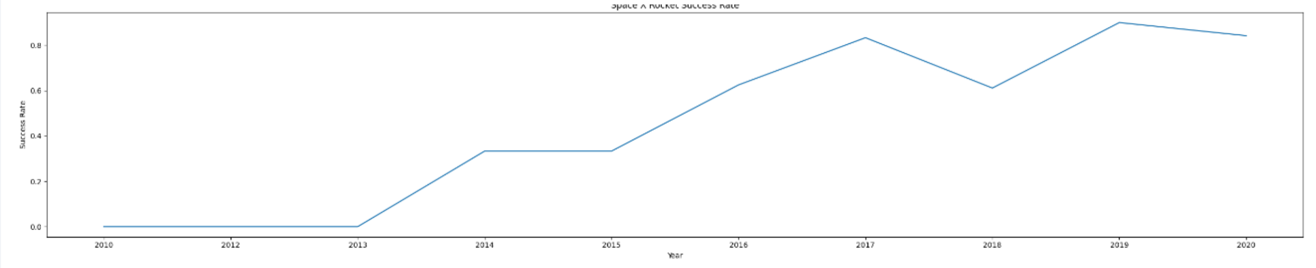












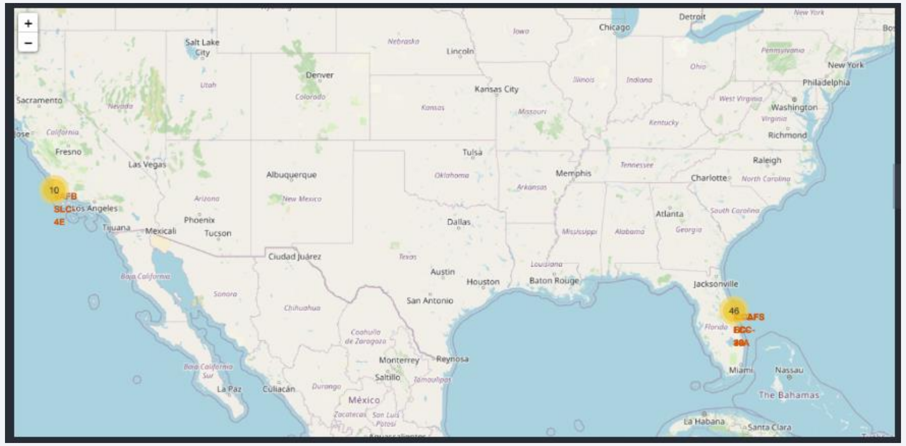
EDA with SQL

GitHub Link:

- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome in ground pad was acheived.
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
- 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

Build an Interactive Map with Folium

GitHub Link:

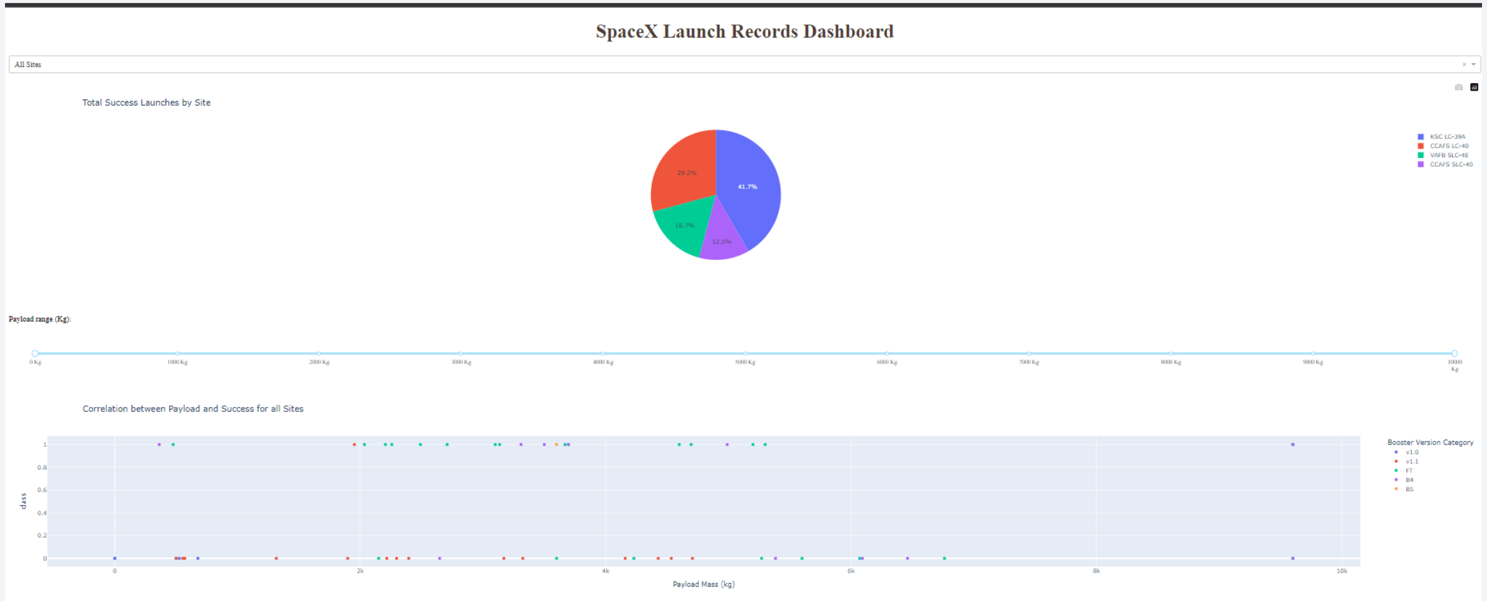


- Folium map object is a map centered on NASA Johnson Space Center at Houston, Texas
  - Red circle at NASA Johnson Space Center's coordinate with label showing its name(folium.Circle, folium.map.Marker).
  - Red circles at each launch site coordinates with label showing launch site name (folium.Circle, folium.map.Marker, folium.features.DivIcon).
  - The grouping of points in a cluster to display multiple and different information for the same coordinates (folium.plugins.MarkerCluster).
  - Markers to show successful and unsuccessful landings. Green for successful landing and Red for unsuccessful landing. (folium.map.Marker, folium.Icon).
  - Markers to show distance between launch site to key locations (railway, highway, coastway, city) and plot a line between them. (folium.map.Marker, folium.PolyLine, folium.features.DivIcon)
- These objects are created in order to understand better the problem and the data. We can show easily all launch sites, their surroundings and the number of successful and unsuccessful landings.

Build a Dashboard with Plotly Dash

[GitHub Link:]

- Dashboard has dropdown, pie chart, rangeslider and scatter plot components
  - Dropdown allows a user to choose the launch site or all launch sites (dash_core_components.Dropdown).
  - Pie chart shows the total success and the total failure for the launch site chosen with the dropdown component(plotly.express.pie).
  - Rangeslider allows a user to select a payload mass in a fixed range (dash_core_components.RangeSlider).
  - Scatter chart shows the relationship between two variables, in particular Success vs Payload Mass (plotly.express.scatter).



Predictive Analysis (Classification)

GitHub Link:

- Data preparation
  - Load dataset
  - Normalize data
  - Split data into training and test sets.
- Model preparation
  - Selection of machine learning algorithms
  - Set parameters for each algorithm to GridSearchCV
  - Training GridSearchModel models with training dataset
- Model evaluation
  - Get best hyperparameters for each type of model
  - Compute accuracy for each model with test dataset
  - Plot Confusion Matrix
- Model comparison
  - Comparison of models according to their accuracy
  - The model with the best accuracy will be chosen

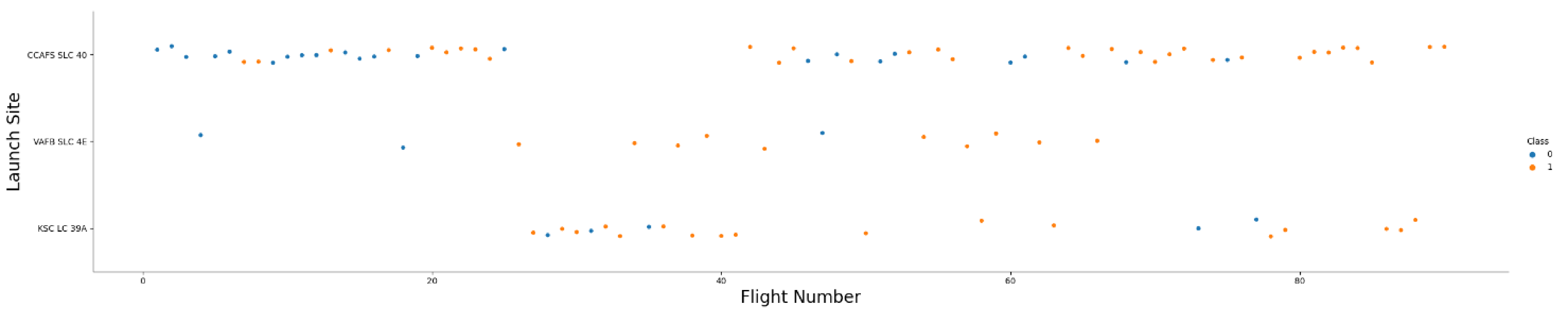
Results

- Decision Tree Classifier has higher test and train accuracy than SVM, KNN and Logistic Regression models.
- Low weighted payloads perform better than heavier payloads.
- The success rates for SpaceX launches is directly proportional time in years they will eventually perfect the launches.
- KSC LC 39A had the most successful launches from all the sites.
- Orbit GEO, HEO, SSO, ES l1 has the best Success Rate.

Section 2

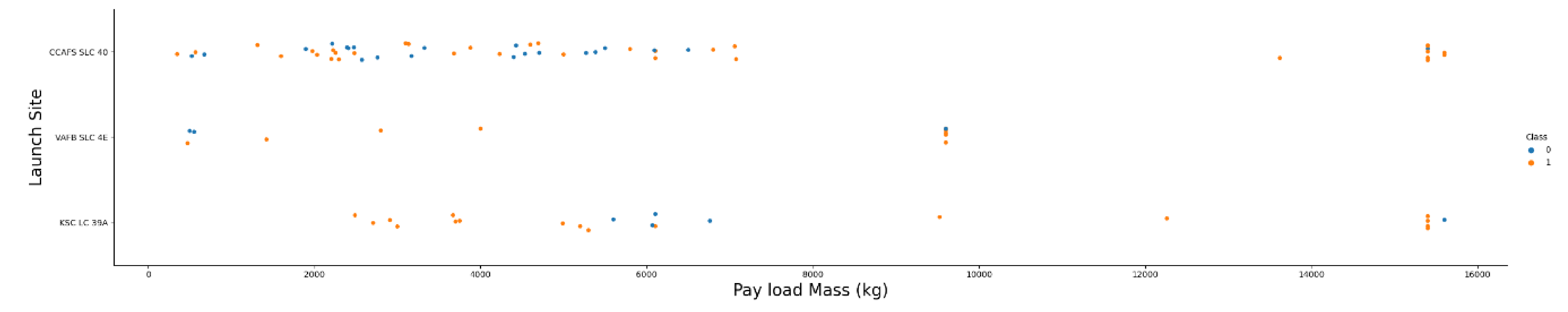
Insights drawn from EDA

Flight Number vs. Launch Site



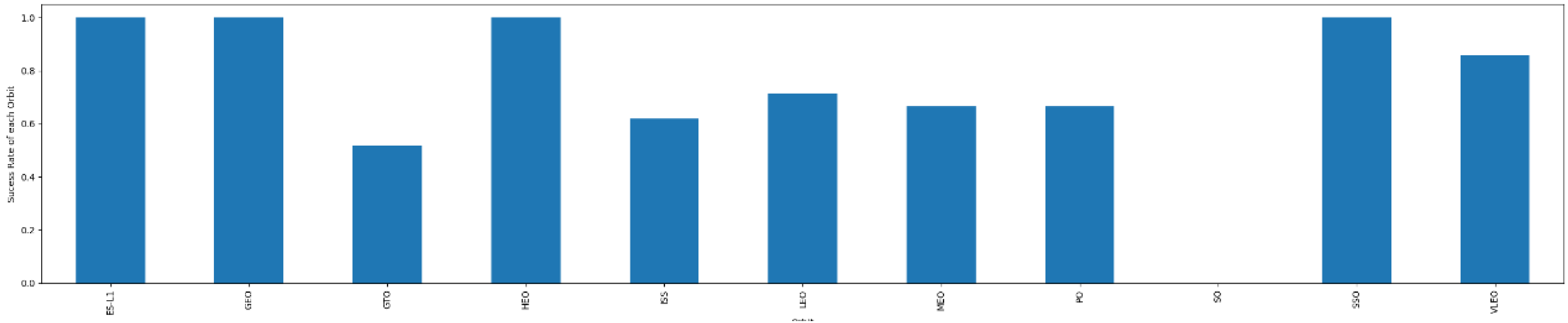
- CCAFS SLC 40 are significantly higher than launches from other sites

Payload vs. Launch Site



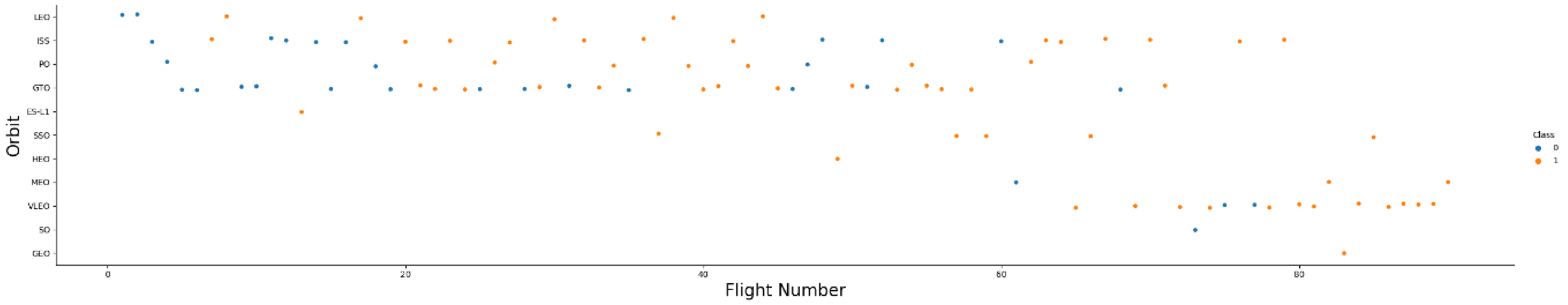
- Depending on the launch site, a heavier payload may be a consideration for a successful landing. On the other hand too heavy payload can make a landing fail.

Success Rate vs. Orbit Type



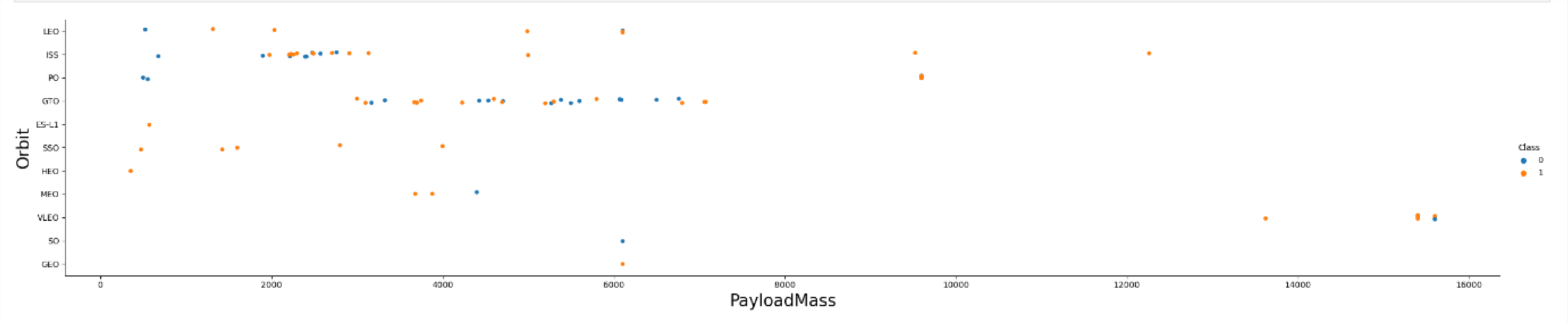
- The orbit types of ES-L1, GEO, HEO, SSO have the best success rate.

Flight Number vs. Orbit Type



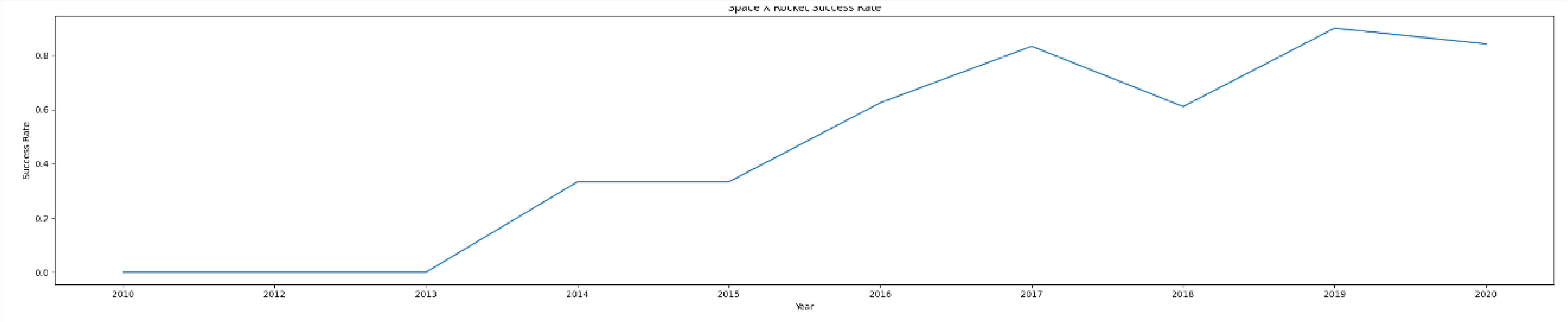
- We notice that the success rate increases with the number of flights for the LEO orbit.

Payload vs. Orbit Type



- The weight of the payloads can have a great influence on the success rate of the launches in certain orbits. For example, heavier payloads improve the success rate for the LEO orbit.

Launch Success Yearly Trend



- Since 2013, we can see an increase in the SpaceX Rocket success rate.

All Launch Site Names

- The use of DISTINCT in the quert allows us to remove duplicate Launch_Site

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

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

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

Launch Site Names Begin with 'CCA'

- The WHERE clause followed by LIKE clause filters launch sites that contain the substring CCA. LIMIT 5 shows 5 records from filltering.

```
%sql select * from SPACEXTBL where Launch_Site like 'CCA%' limit 5;
```

MagicPython

```
 * 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

Total Payload Mass

- This query return the sum of all PAYLOAD_MASS_KG_ where the Customer is NASA (CRS)

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

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

sum(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

- This query returns the average of all PAYLOAD_MASS_KG_ where the Booster_Version contains the substring F9 V1.1

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

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

avg(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

- This query shows the oldest successfel landing.
- Where clause filters dataset

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

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

min(Date)
01-05-2017

Successful Drone Ship Landing with Payload between 4000 and 6000

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

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

- This query returns the Landing_Outcome was successful and PAYLOAD_MASS_KG_ between 4000 and 6000 kg

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

Total Number of Successful and Failure Mission Outcomes

```
%%sql select (select count("MISSION_OUTCOME") FROM SPACEXTBL where "MISSION_OUTCOME" like '%Success%') as SUCCESS, \
    (select count("MISSION_OUTCOME") from SPACEXTBL where "MISSION_OUTCOME" like '%Faliure%') as FAILURE
```

SUCCESS	FAILURE
100	1

- With the first Select, we show the subqueries that return results. The first subquery counts the successful mission. The second subquery counts the unsuccessful mission.

Boosters Carried Maximum Payload

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

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

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

- We used subquery to filter data by returning the heaviest payload mass with Max function. The main query return Booster_Version with the heaviest payload mass

2015 Launch Records

```
%%sql select substr(Date, 4, 2), `Landing _Outcome`, Booster_Version, Launch_Site
from SPACEXTBL
where `Landing _Outcome` = 'Failure (drone ship)' and substr(Date,7,4)='2015';
```

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

substr(Date, 4, 2)	Landing _Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

- Substr(Date, 4, 2) shows month. Substr(Date, 7, 4) shows year.
- This query returns landing was unsuccessfull and landing date in 2015

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

```
%%sql select "LANDING_OUTCOME", count("LANDING_OUTCOME") from SPACEXTBL\
where "DATE" >= '04-06-2010' and "DATE" <= '20-03-2017' and "LANDING_OUTCOME" like '%Success%'\
group by "LANDING_OUTCOME" \
order by count("LANDING_OUTCOME") desc;
```

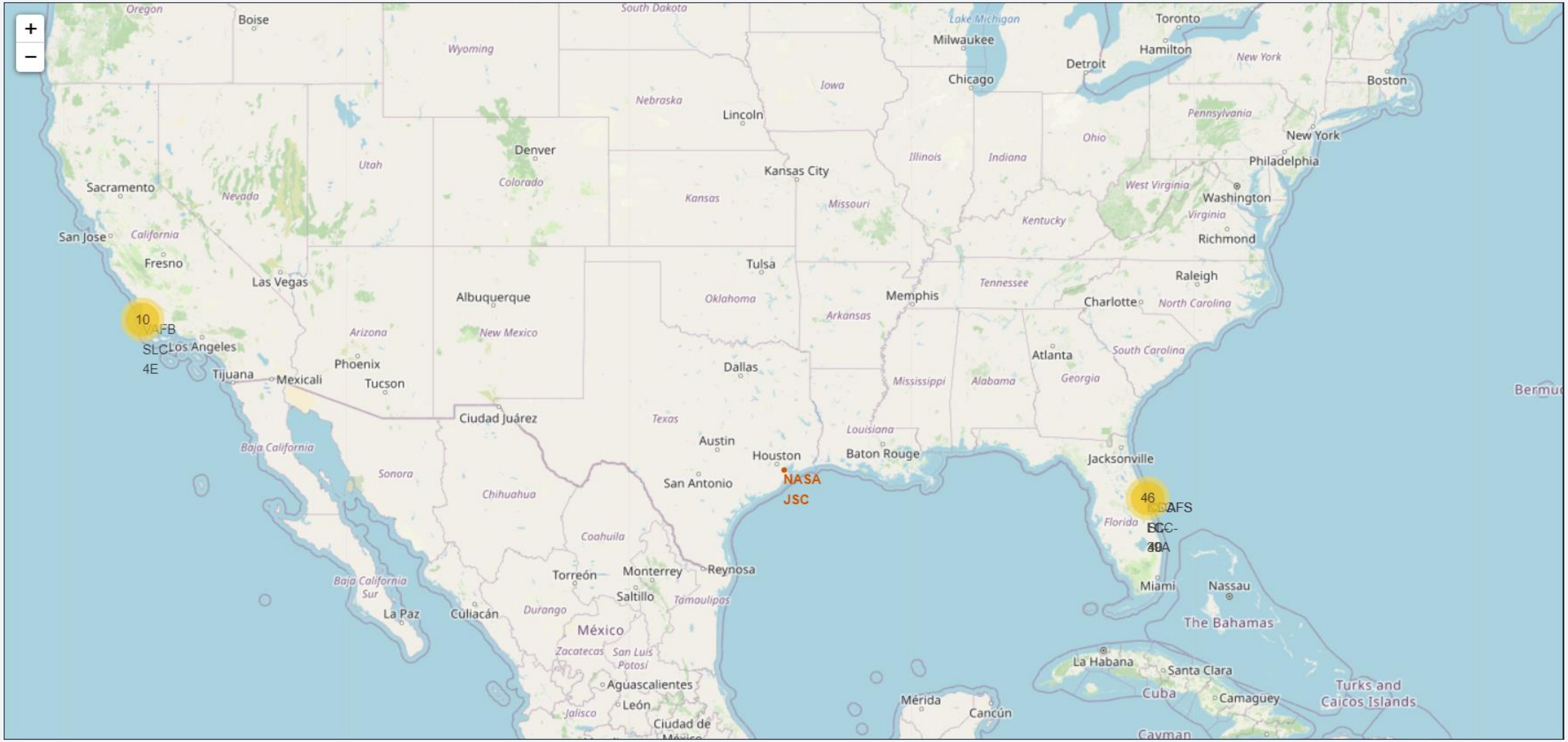
Landing _Outcome	COUNT("LANDING _OUTCOME")
Success	20
Success (drone ship)	8
Success (ground pad)	6

- This query return mission was successful and date between 2010-06-04 and 2017-03-20.

Section 3

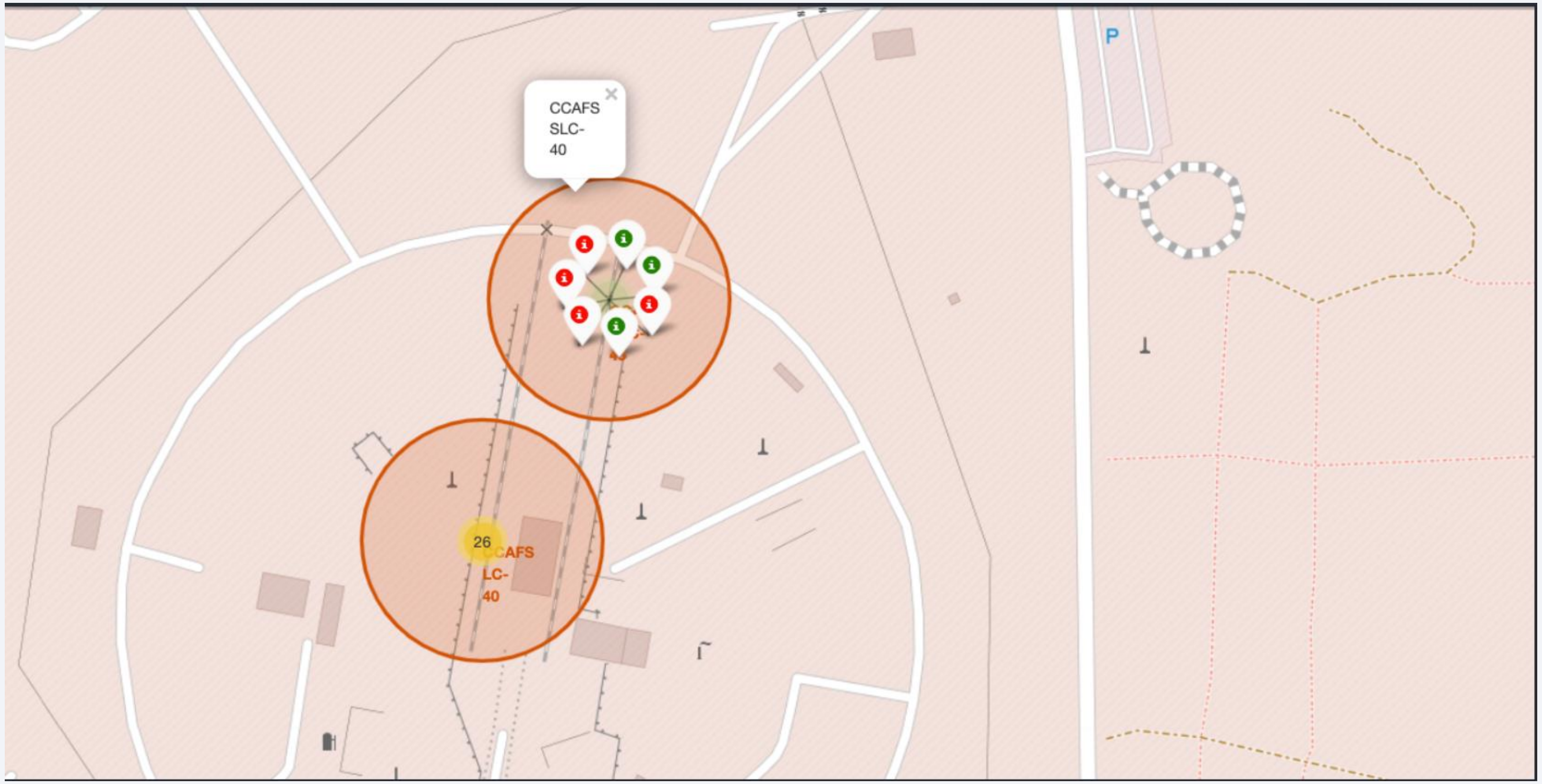
Launch Sites Proximities Analysis

<Folium Map - Ground stations >



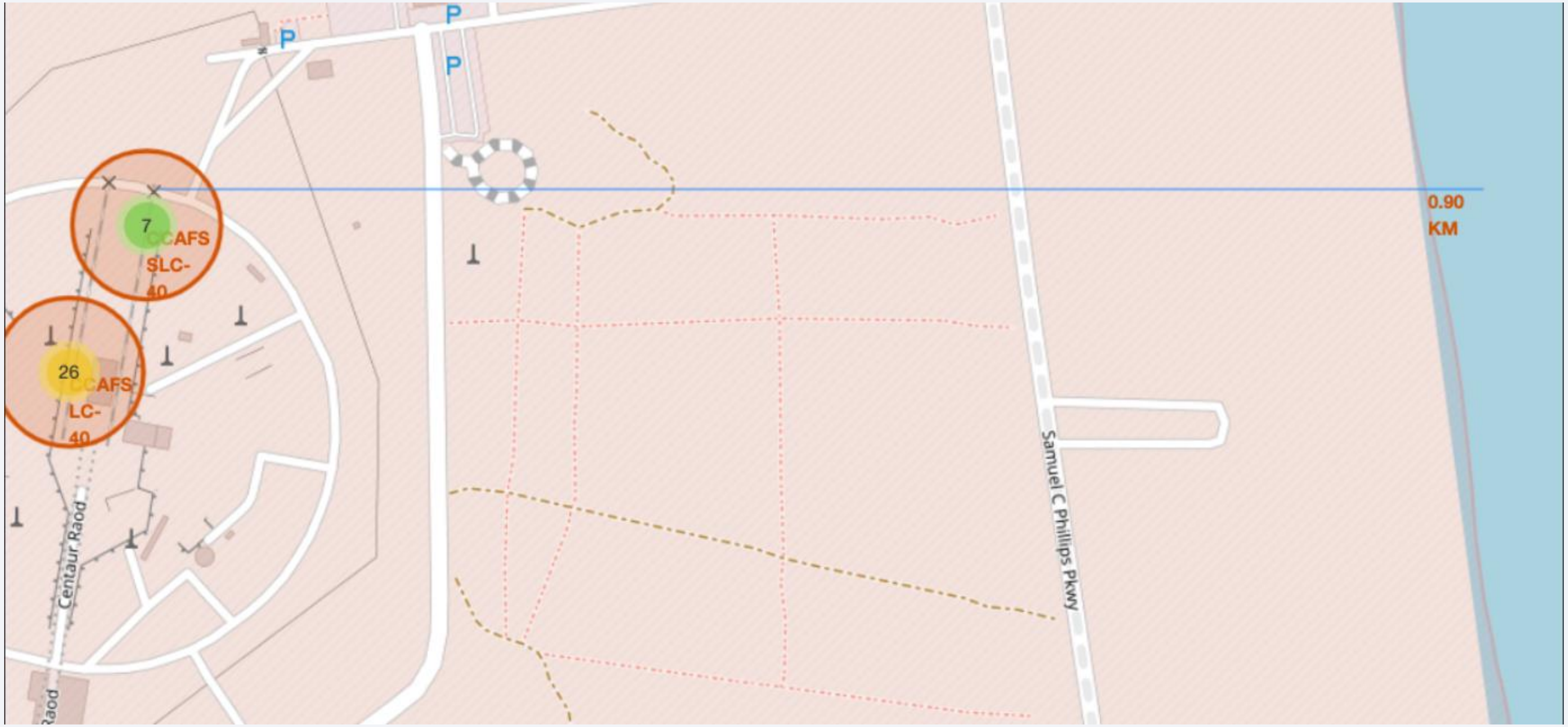
- We see that SpaceX launch sites are located on the coast of the USA

<Folium Map - Color Labeled Markers >



- Green markers represent successful launches.
- Red markers represent unseccessful launches

<Folium Map –Distances between CCAFS SLC-40 and its proximities >

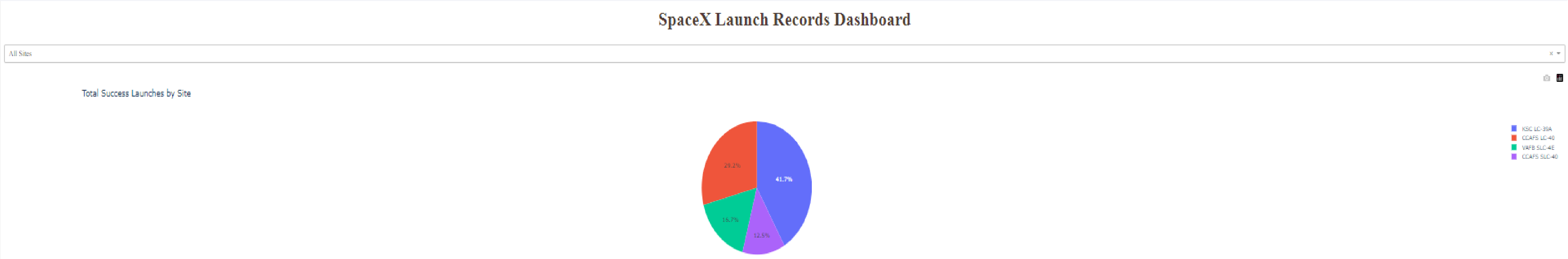


- Is CCAFS SLC-40 in close proximity to coastline?
  - Yes

Section 4

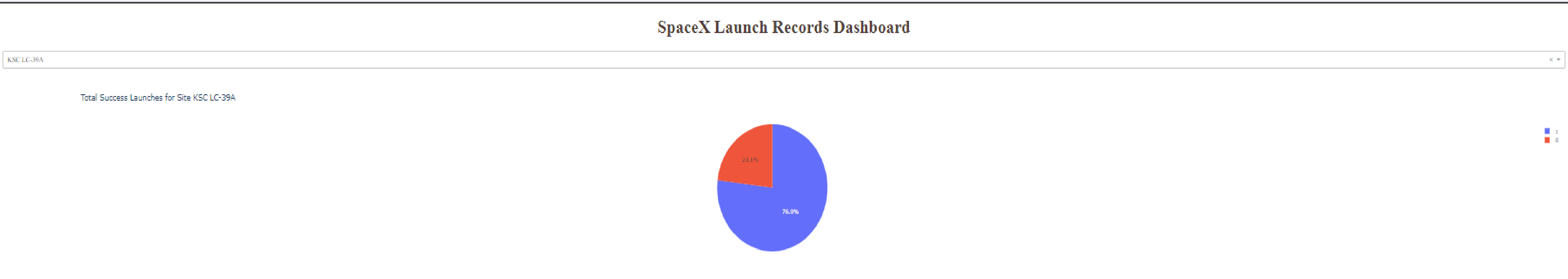
Build a Dashboard with Plotly Dash

<Dashboard - Total success by Site >



- KSC LC-39A has best success rate of launches

<Dashboard –Total success launches for Site KSC LC-39A >



- KSC LC-39A has achieved a 76.9% success rate and 23.1% failure rate.

<Dashboard -Payload mass vs Outcome for all sites with different payload mass selected >

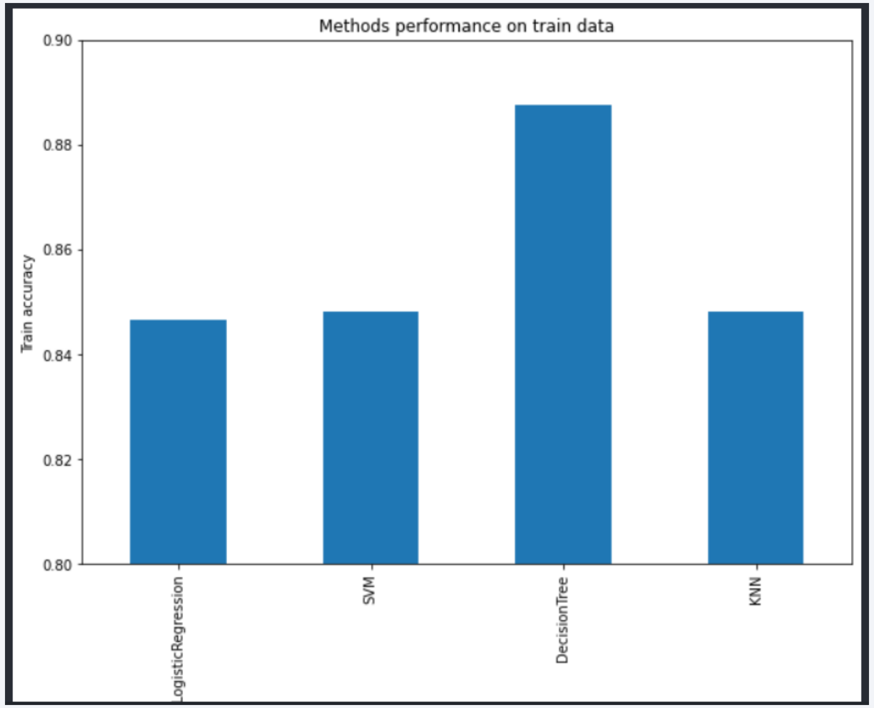
- Low weighted payloads have a better success rate than the heavy weighted payloads

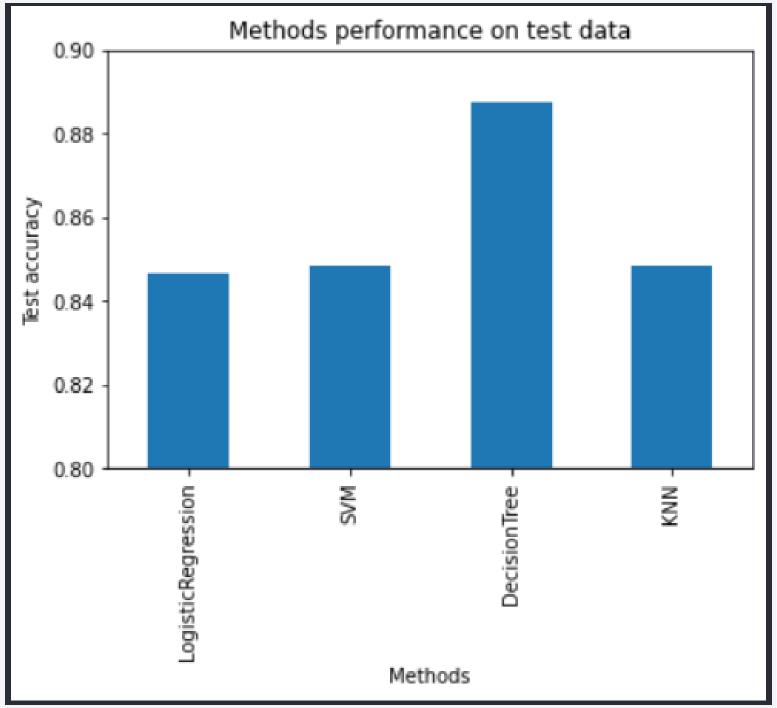
Section 5

Predictive Analysis (Classification)

Classification Accuracy

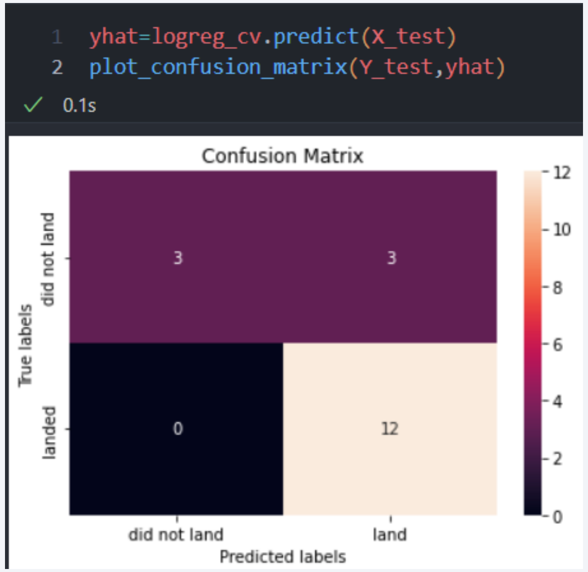
	Accuracy Train	Accuracy Test
DecisionTree	0.887500	0.888889
KNN	0.848214	0.833333
SVM	0.848214	0.833333
LogisticRegression	0.846429	0.833333

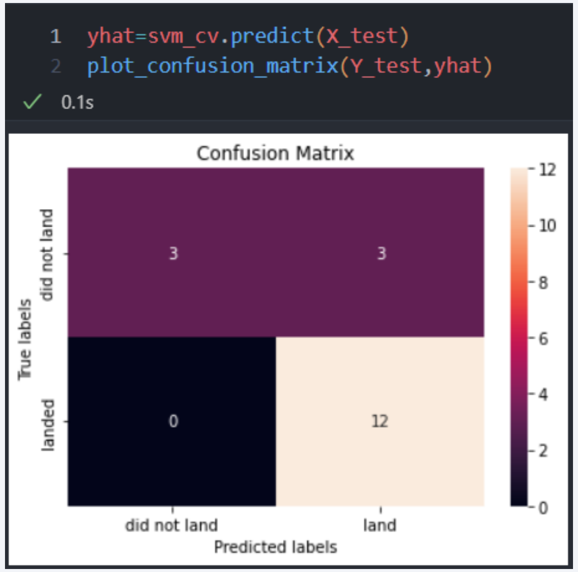


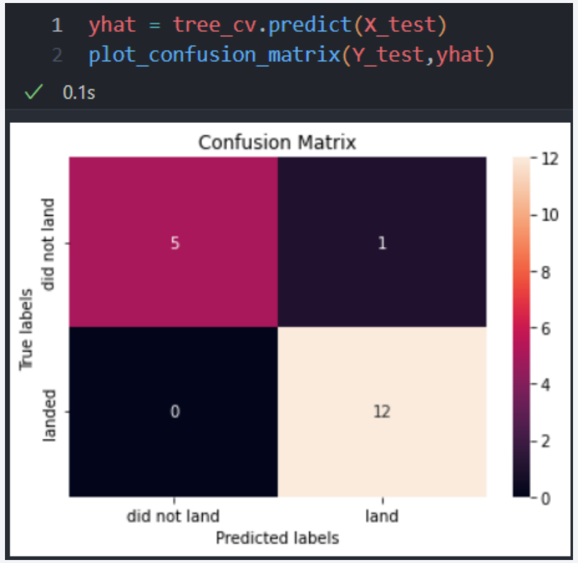


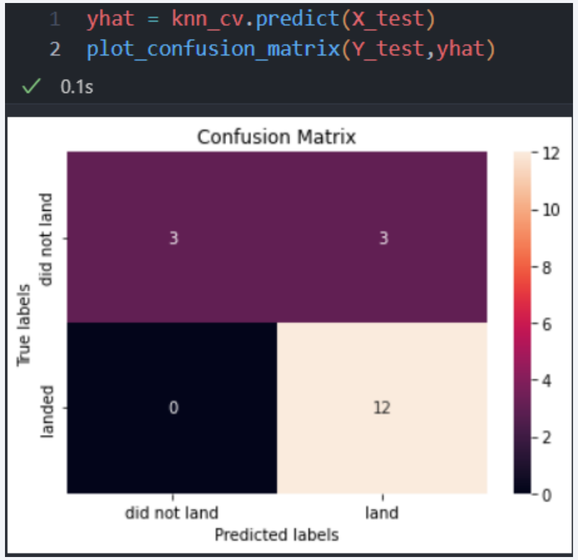
- For accuracy scores are both test and train graphics are so similar. If we want to choose a model we could take the Decision Tree Classifier.

Confusion Matrix

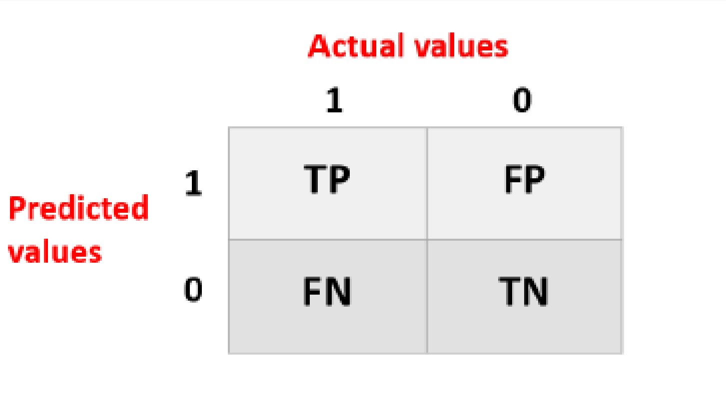








- SVM, KNN, Logistig Regression have same rate on TP, FP, FN, TN
- Decision Tree has better accuracy.Because TP + TN higher than other models.
- The main problem of these 3 models are False Positives (FP)



Conclusions

- Decision Tree model is the best in terms of prediction accuracy for this dataset.
- Low weighted payloads perform better than the heavier payloads.
- KSC LC 39A had the most successful launches from all the sites
- Orbit GEO, HEO, SSO, ES L1 have the best success rate.
- The success of a mission can be explained by several factors such as the launch site, the orbit and especially the number of previous launches. Indeed, we can assume that there has been a gain in knowledge between launches that allowed to go from a launch failure to a success.
- Depending on the orbits, the payload mass can be a criterion to take into account for the success of a mission. Some orbits require a light or heavy payload mass. But generally low weighted payloads perform better than the heavy weighted payloads.
- With the current data, we cannot explain why some launch sites are better than others (KSC LC-39A is the best launch site). To get an answer to this problem, we could obtain atmospheric or other relevant data.

Appendix

```
import warnings
warnings.filterwarnings( 'ignore' )
```

✓ 0.1s

- We add this code block to hide some warnings

Thank you!