IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<May Wong>
<03/02/2023>

Outline

Executive Summary

- Summary of methodologies

1. Collect data via SpaceX Rest API and Web Scraping from Wikipedia (SpaceX)
2. Preparation the dataset and load the dataset to IBM Watson Studio
3. Exploratory Data Analysis including identify and calculate the % of the missing values in each attribute, calculate the number of launches on each site, the number and occurrence of each orbit, Create a label for the data
4. Exploratory Data Analysis using SQL (the total payload mass carried by boosters, average payload mass etc)
5. Exploratory Data Analysis using visualization tool such as Matplotlib
6. Perform interactive data analysis by interactive Map with Folium
7. Perform interactive data analysis by dashboards with Plotly Dash
8. Predictive analysis using classification models

- Summary of all results

1. Exploratory data results
2. Interactive dashboard and maps
3. Predictive results

Introduction

- Project background and context

The objective of this project to collect information about Space X, using machine learning algorithms to predict and determine whether Space X can reuse or recover the first stage and the determine the price of each launch. According to the Space X's website, the Falcon 9 rocket launch cost only 62 million dollars but other providers cost around 165 million dollars each. The price difference is explained by the fact that Space X can reuse the first stage. We are using machine learning model to determine whether the stage will successful land. This information will be interest for Space Y to estimate the cost for rocket launch.

- Problems you want to find answers

✓ What are the main characteristics of a successful or unsuccessful landing ?

✓ What are the effects of each relationship of the rocket variables on the success or unsuccessful of a landing ?

✓ What are the conditions which will allow SpaceX to achieve the best landing success rate ?

✓ What are the major factor to determine the rocket can be successful launch?

Section 1

Methodology

Methodology

Executive Summary

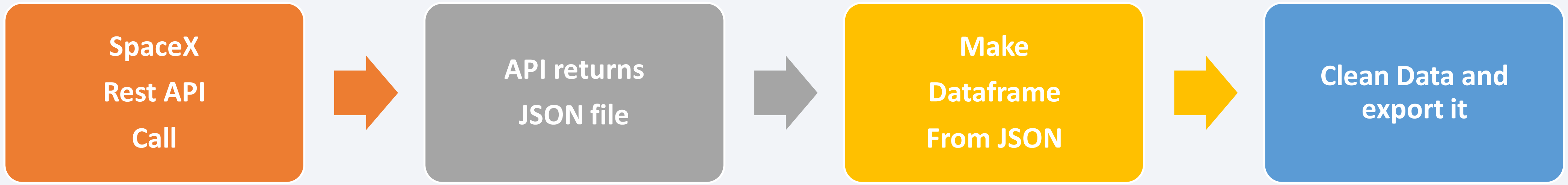
- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

Data were collected from Rest Space X API and web scraping Wikipedia through website.

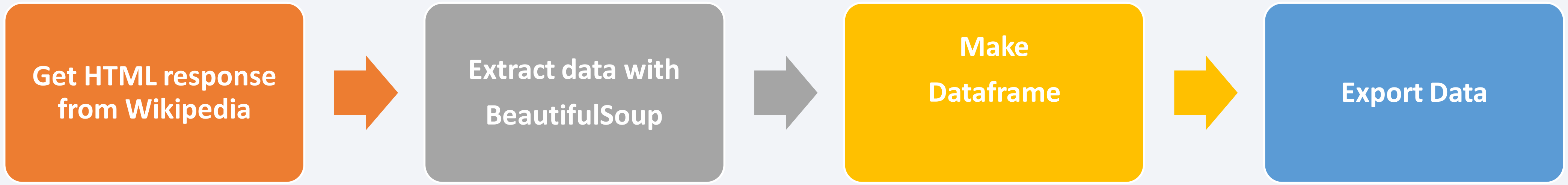
- The information details are collected by the API were included rocket, launches, payload information.

➢ Space X Rest API URL = api.spacexdata.com/v4/



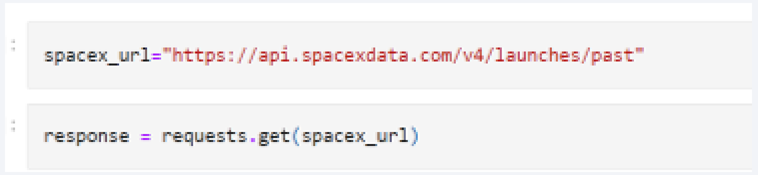
- The information details are collected by the web scraping of Wikipedia are launches, landing, payload information.

➢ https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922


Get HTML response from Wikipedia
Extract data with BeautifulSoup
Make Dataframe
Export Data

Data Collection – SpaceX API

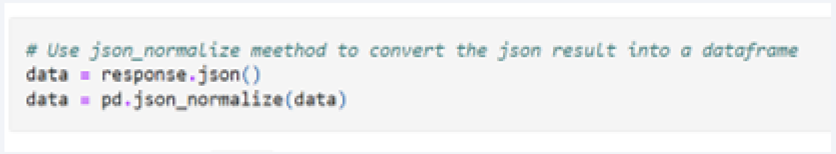
Step 1 Get Response from API



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

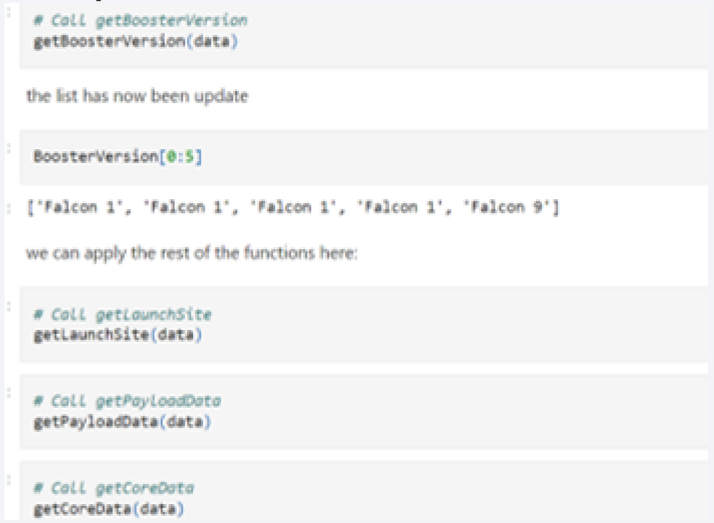
```
response = requests.get(spacex_url)
```

Step 2 Convert Response to JSON File



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

Step 3 Transform data



```
# Call getBoosterVersion
getBoosterVersion(data)
```

the list has now been update

```
BoosterVersion[0:5]
```

```
['Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 9']
```

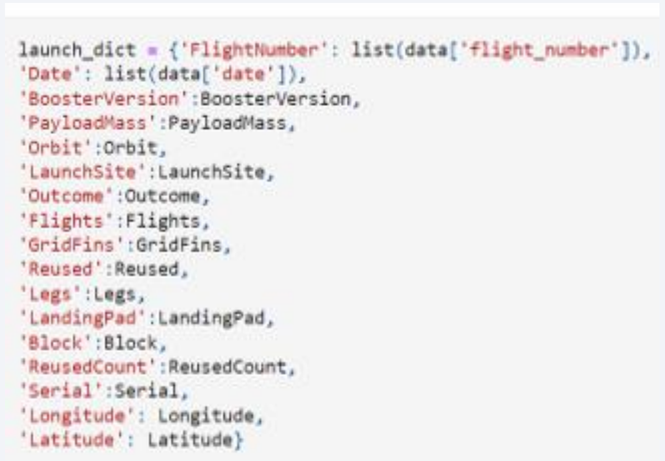
we can apply the rest of the functions here:

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

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

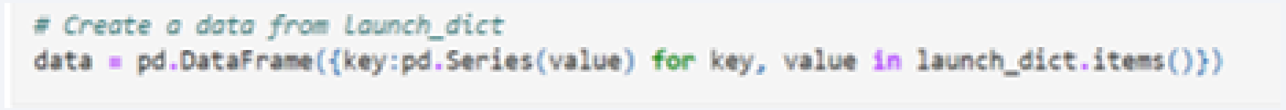
```
# Call getCoreData
getCoreData(data)
```

Step 4 Create dictionary with data



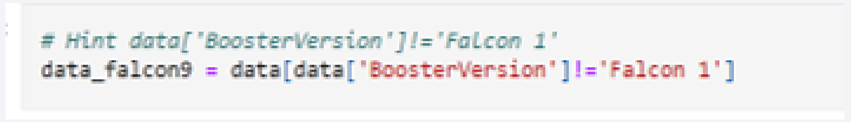
```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

Step 5 Create dataframe



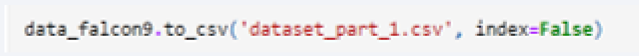
```
# Create a data from launch_dict
data = pd.DataFrame({key:pd.Series(value) for key, value in launch_dict.items()})
```

Step 6 Filter DataFrame



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

Step 7 Export to file

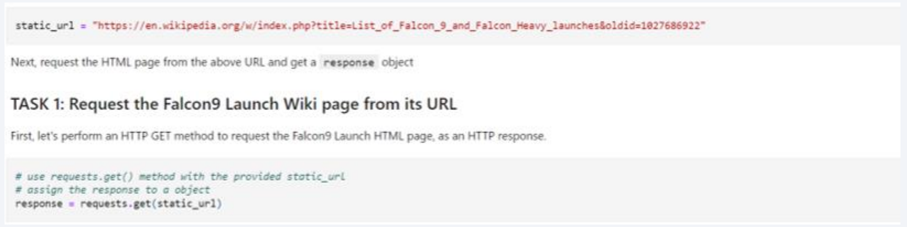


```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Link to code

Data Collection - Scraping

Step 1 Get response from HTML



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

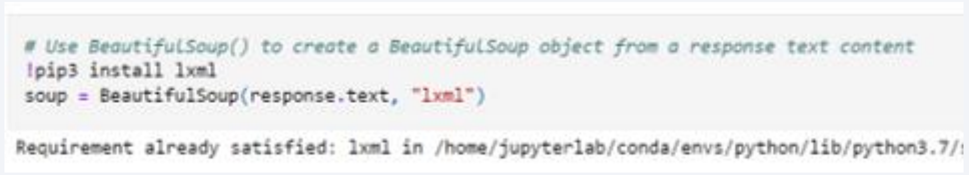
Next, request the HTML page from the above URL and get a `response` object

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.

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

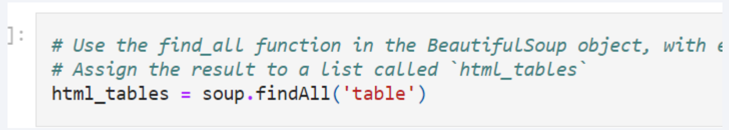
Step 2 Create BeautifulSoup Object



```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
!pip3 install lxml
soup = BeautifulSoup(response.text, "lxml")
```

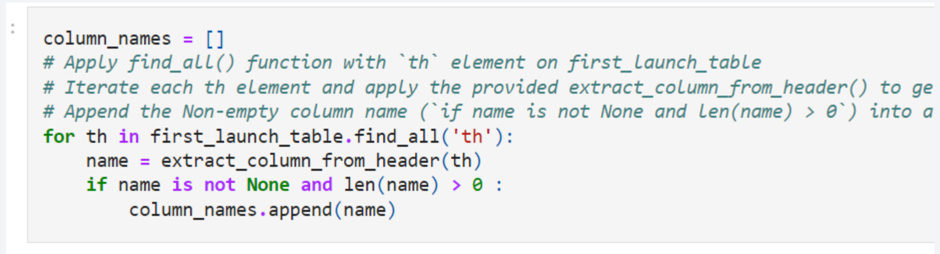
```
Requirement already satisfied: lxml in /home/jupyterlab/conda/envs/python/lib/python3.7/
```

Step 3 Find all tables



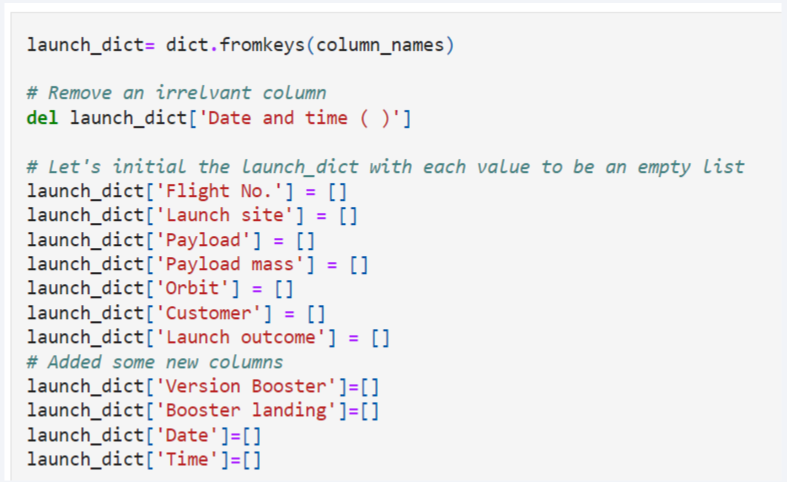
```
]: 
# Use the find_all function in the BeautifulSoup object, with e
# Assign the result to a list called `html_tables`
html_tables = soup.findAll('table')
```

Step 4 Get column names



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

Step 5 Create dictionary

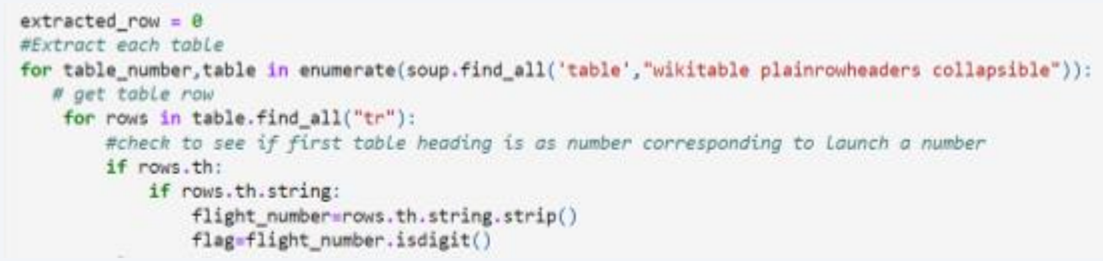


```
launch_dict= dict.fromkeys(column_names)

# 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']=[]
```

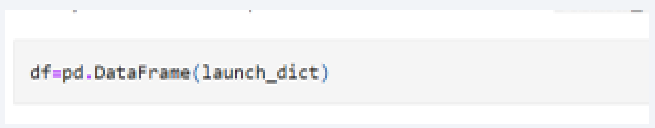
Step 6 Add data to keys



```
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()
```

Data Collection - Scraping

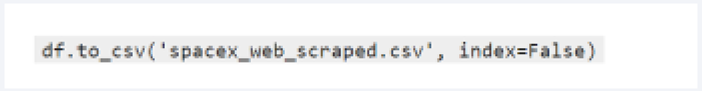
Step 7 Create dataframe from dictionary



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

Link to code

Step 8 Export to file

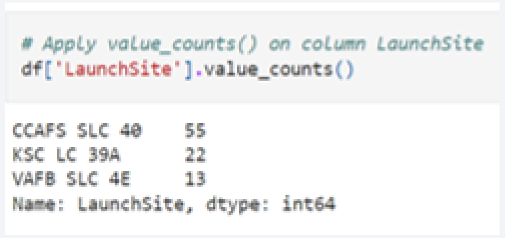


```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Data Wrangling

- In our dataset, there are some cases where the booster didn't land successfully.
  - True Ocean - True RTLS, True ASDS means the mission has been successful.
  - False Ocean – False RTLS, False ASDS means the mission was a failure.

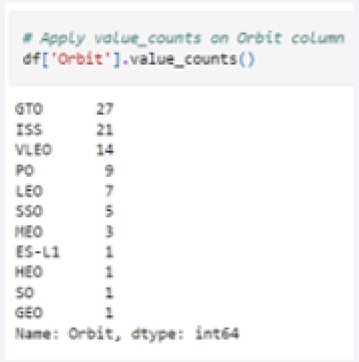
1. Calculate launches number for each site



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

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

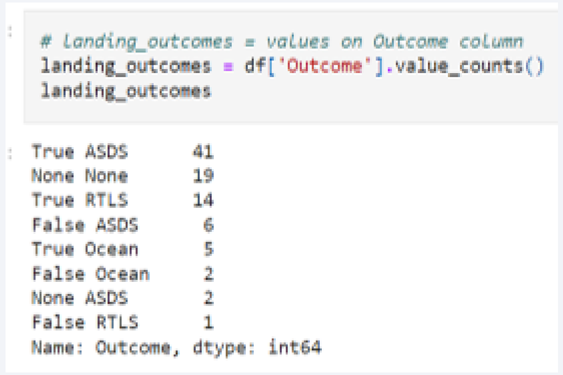
2. Calculate the number and occurrence of each orbit



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

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

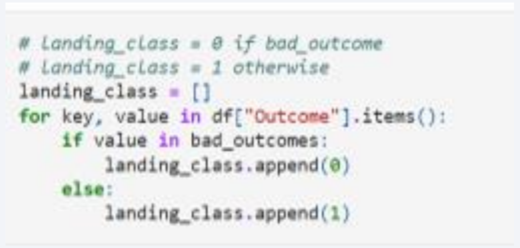
3. Calculate number and occurrence of mission outcome per orbit type



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

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

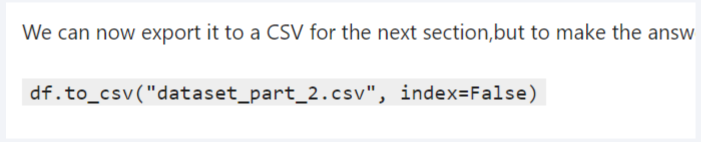
4. Create landing outcome label from outcome column



```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for key, value in df["Outcome"].items():
    if value in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

Data Wrangling

5. Export to file


We can now export it to a CSV for the next section,but to make the answ

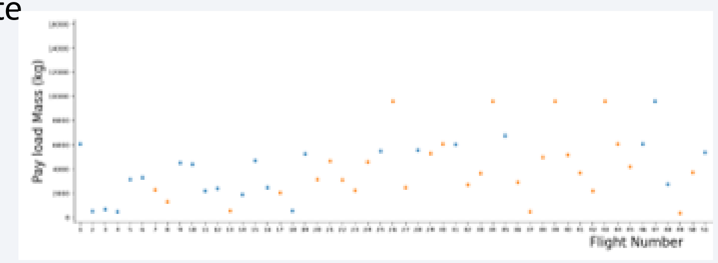
```
df.to_csv("dataset_part_2.csv", index=False)
```

Link to code

EDA with Data Visualization

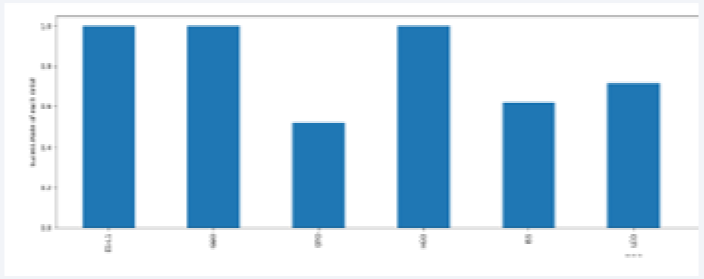
- ✓ **Scatter Graph – Scatter plots show the relationship between variables.**

- Flight number vs PayLoad Mass
- Flight number vs Launch Site
- Payload vs Launch Site
- Orbit vs Flight number
- PayLoad vs Orbit Type
- Orbit vs PayLoad Mass



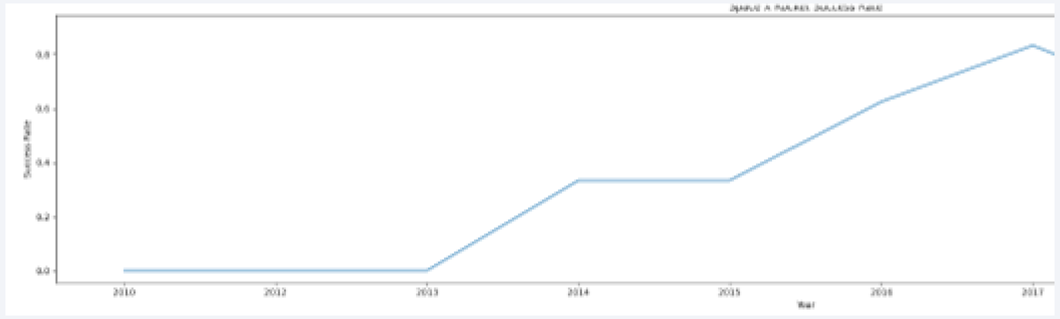
- ✓ **Bar Graph – it is showed the relationship between numeric and categoric variables**

- Success vs Orbit



- ✓ **Line Graph – it is showed the variables, their trends and development. Line graphs can help to show global behavior and make prediction for unseen data.**

- Success rate vs Year



Link to code

EDA with SQL

- Display the names of the unique launch site in the space mission
- Display five records where launch site 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 v 1.1.
- List the date when the first successful landing outcome in ground pad was achieved
- 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
- 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
- Rank the count of successful landing outcomes between the date 04.06.2010 and 20.03.2017 in descending order

Link to code

Build an Interactive Map with Folium

- Interactive Folium map object is a map located on NASA Johnson Space Center in Houson, Texas
- Red circle in NASA Johnson Space Centre coordinate with label show the name
- Red circle at each launch site coordinates with label show launch site name
- The grouping points in a cluster to show many and different information detail for the same coordinates location.
- Markers to show successful and unsuccessful landing. Green for successful landing and red for unsuccessful landing.
- Markers to show distance between launch site to key locations such as railway, city and highway. It plot a line between them.
- These points or objects are developed in order to understand further the problem and data. We can show easily all launch sites, their surroundings and the number of successful and unsuccessful landings.

Link to code

Build a Dashboard with Plotly Dash

- Dashboard had dropdown, pie chart, rangeslider and scatter plot components.
- Dropdown allows a user to choose the launch site or all launch sites
- Pie chart show the total success and the total failure for the launch site chosen with the dropdown component
- Rangeslider allows a user to select a payload mass in a fixed range
- Scatter chat show the relationship between two variables, in particular success vs payload Mass

Link to code

Predictive Analysis (Classification)

Step 1 Data Preparation

- Import Libraries and define functions
- Load the dataset and dataframe
- Normalize the dataset

Step 2 Model Preparation & training

- Describe the dataset
- Select machine learning algorithms
- Set parameters for algorithm to GridSearch CV
- Training GridSearch Models with training dataset

Step 3 Model Evaluation

- Look for the best hyperparameters for each type model training
- Calculate the accuracy of each model
- Plot the confusion matrix to understand and demonstrate the accuracy for each type
- Lot bar chart to compare the accuracy and each type of model

Link to code

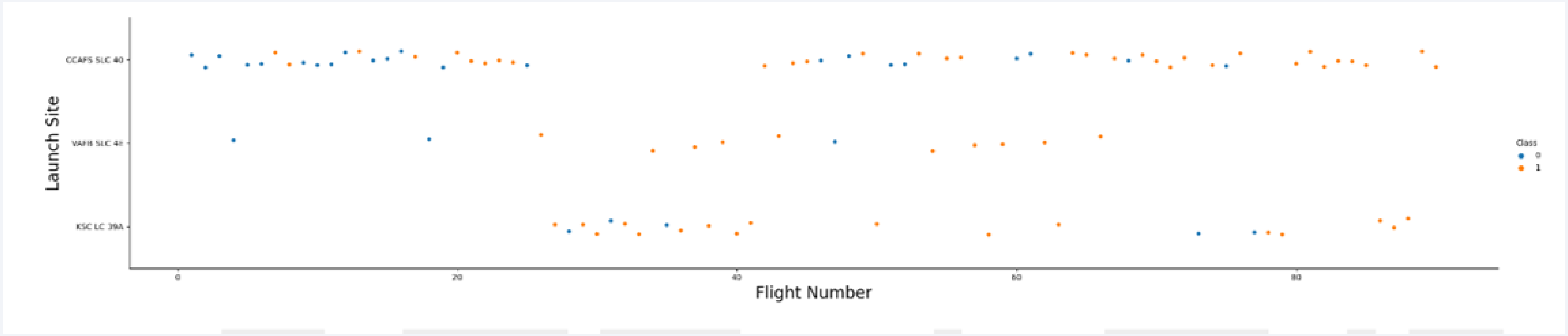
Results

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

Section 2

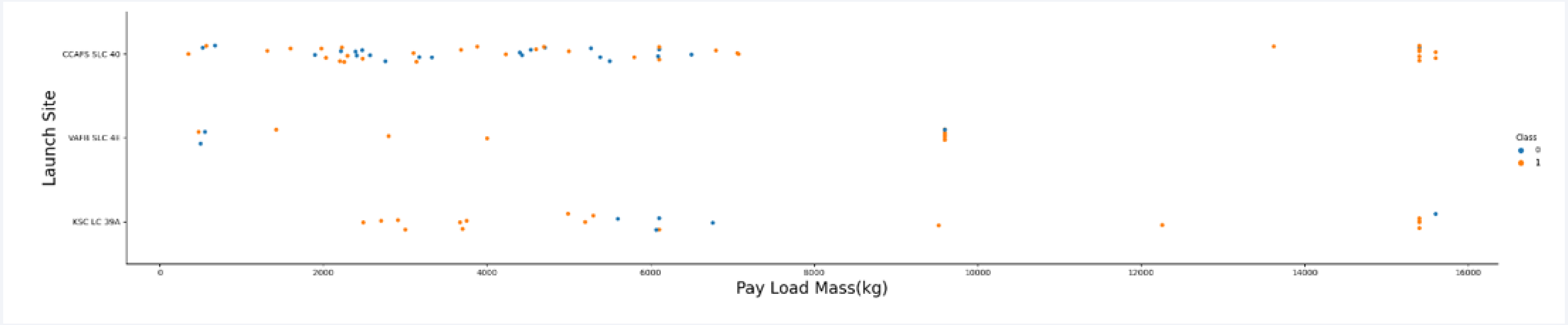
Insights drawn from EDA

Flight Number vs. Launch Site



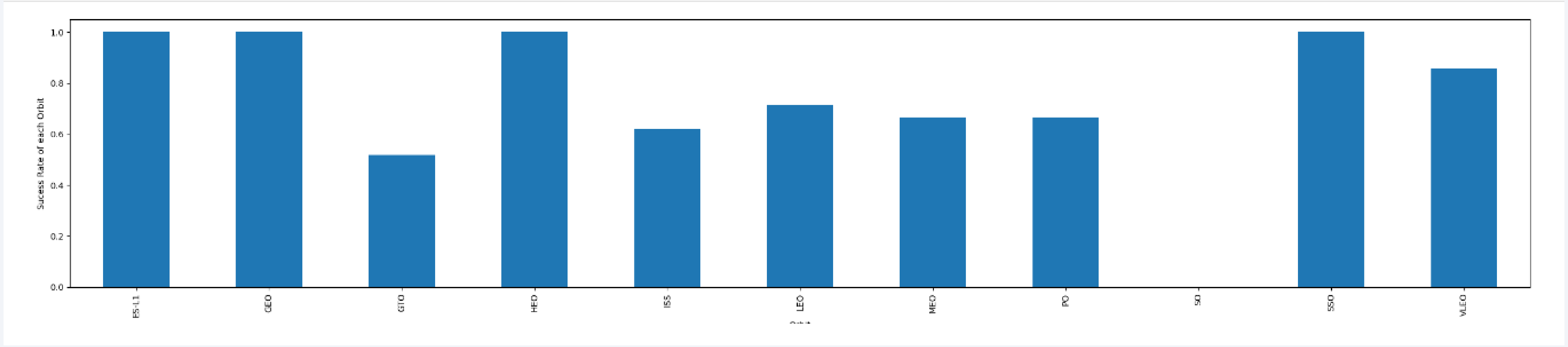
We can each site, the success rate is increasing according to the above figure.

Payload vs. Launch Site



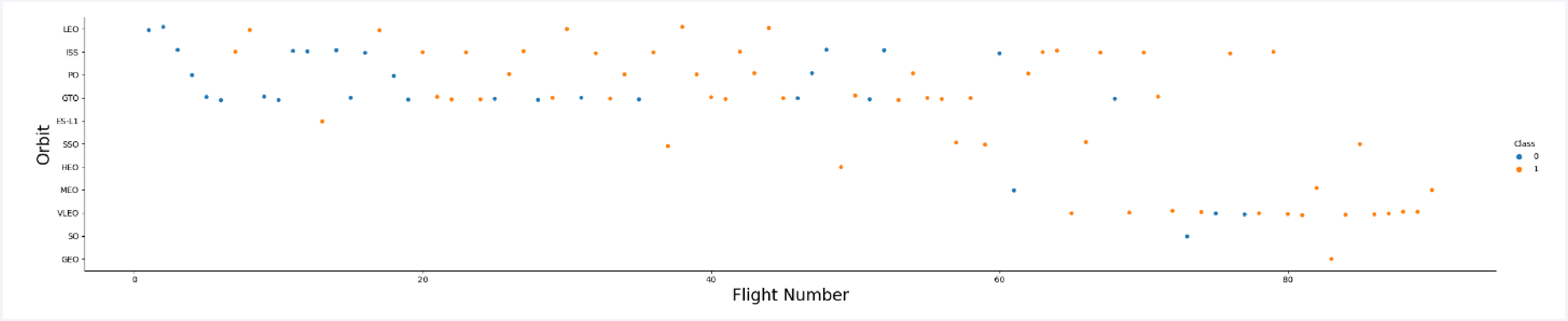
We can see some heavier payload may be a consideration for a successful landing in some site. At the same time, very heavy payload can make a landing fail.

Success Rate vs. Orbit Type



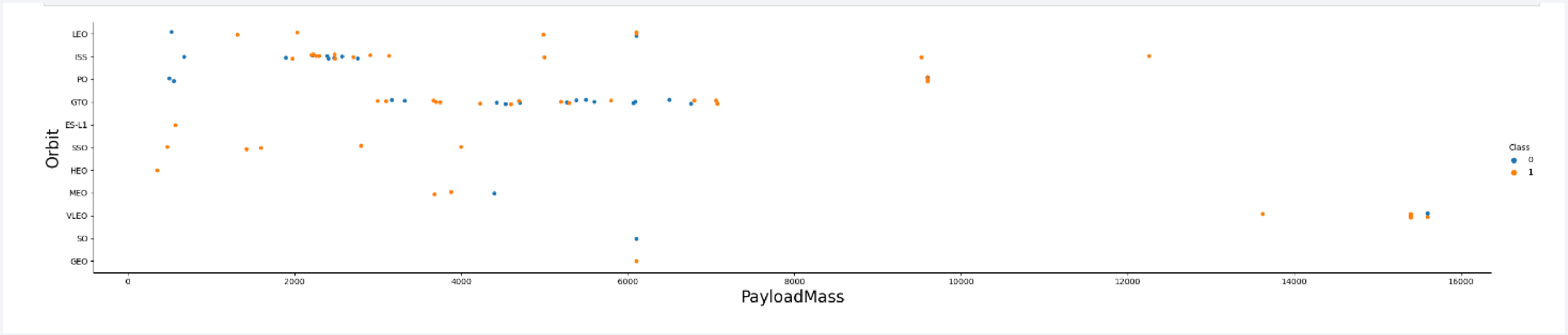
We can see success rate for different orbit types from this bar chart. We can see that ES-L1, GEO, HEO, SSO have the best success rate.

Flight Number vs. Orbit Type



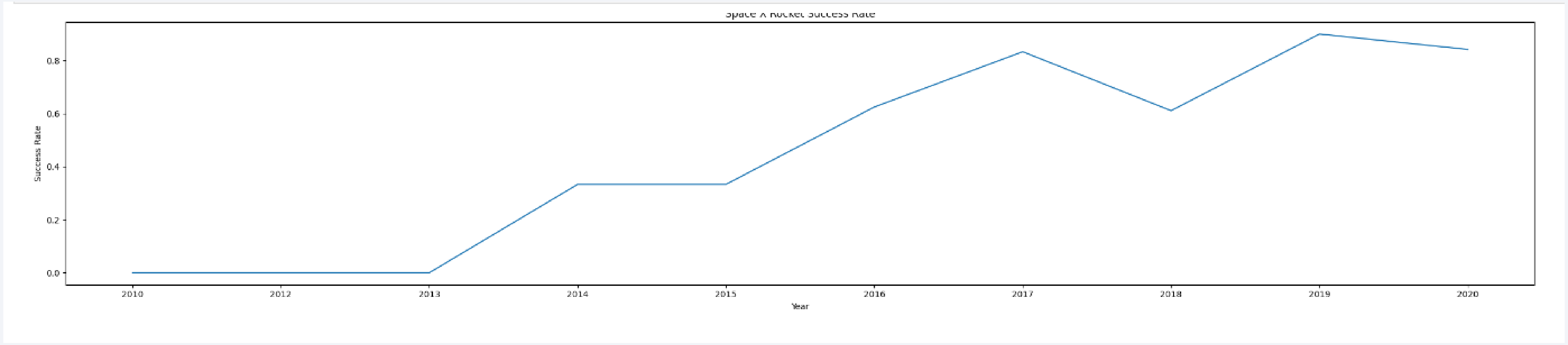
We can see that the success rate increases with the number of flights for the LEO orbit. Regarding orbits like GTO, there is no relation between the success rate and the number of flights. We can assume that the high success rate of SSO or HEO orbits is because of learning experience from previous launches.

Payload vs. Orbit Type



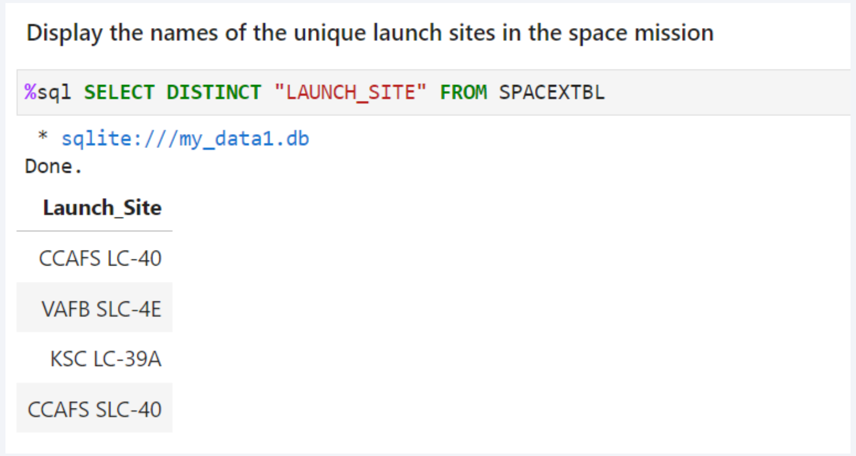
We can see the weight of the payloads have major influence on the success rate of the launches in certain orbits. For example, heavier payloads improve the success rate for the LEO orbit. Another finding is that decreasing the payload weight for a GTO orbit improves the success of a launch. Determining the suitable weight for each type of orbit maybe the key factor to have successful launch.

Launch Success Yearly Trend



The successful rate of Space X rocket is increasing since 2013.

All Launch Site Names


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

```
%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

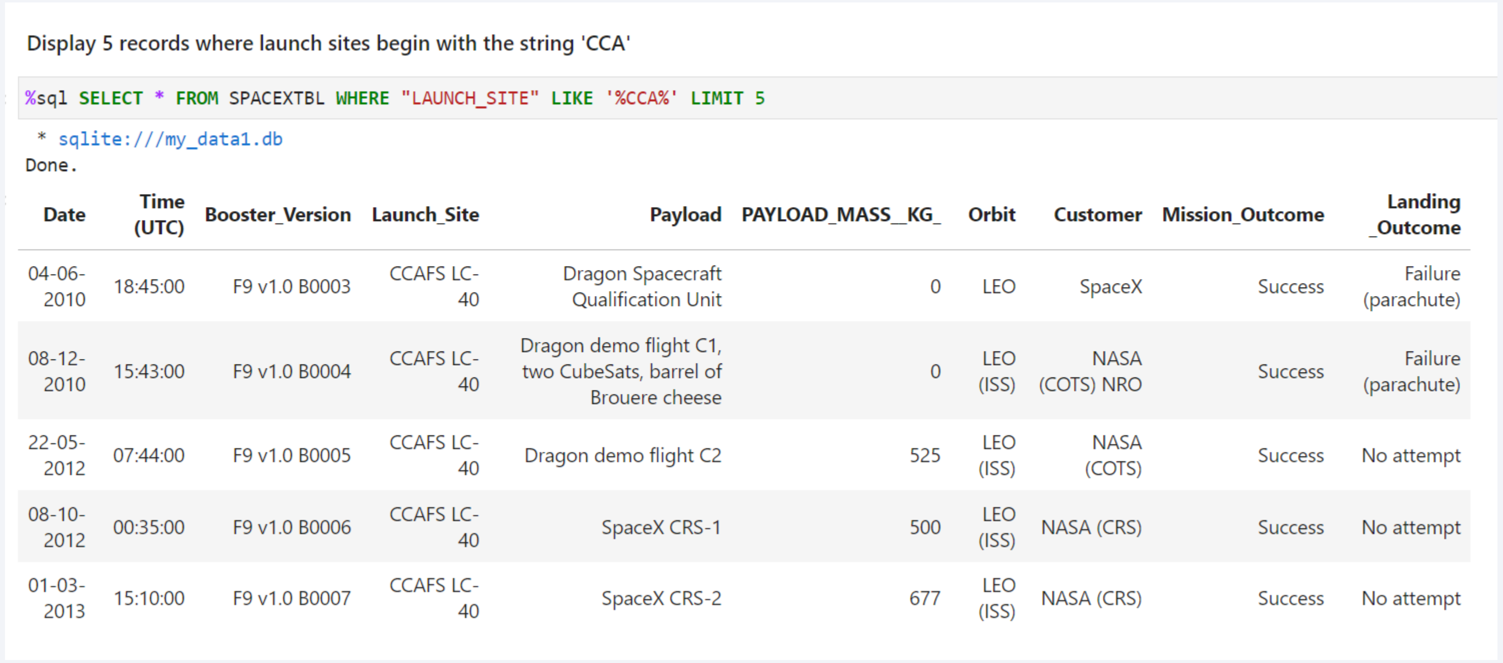
SQL Query

Results

Explanation

The use of DISTINCT command in the query allows to remove duplicate “LAUNCH_SITE”.

Launch Site Names Begin with 'CCA'



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

```
%sql SELECT * FROM SPACEXTBL WHERE "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

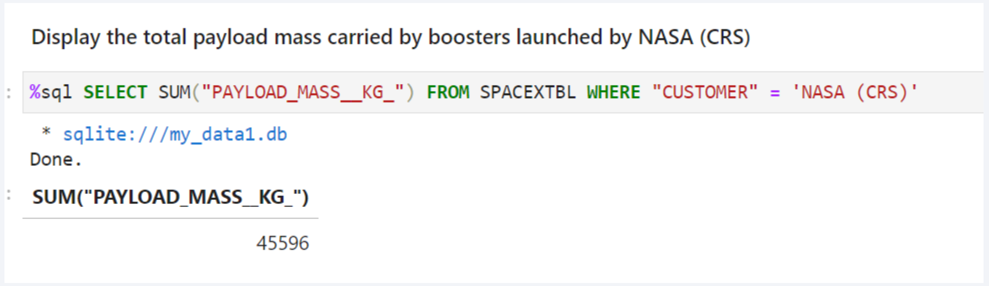
SQL Query

Results

Explanation

The WHERE clause followed by LIKE clause filters launch sites that contain the string “CCA”.
LIMIT 5 can shows 5 records from filtering.

Total Payload Mass



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

```
%sql SELECT SUM("PAYLOAD_MASS__KG_") FROM SPACEXTBL WHERE "CUSTOMER" = 'NASA (CRS)'
```

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

SUM("PAYLOAD_MASS__KG_")
45596

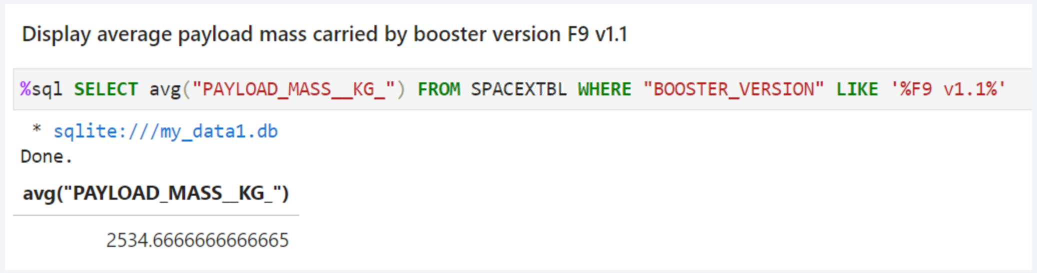
SQL Query

Results

Explanation

This query returns the sum of all payload masses where the customer is NASA (CRS).

Average Payload Mass by F9 v1.1



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

```
%sql SELECT avg("PAYLOAD_MASS__KG_") FROM SPACEXTBL WHERE "BOOSTER_VERSION" LIKE '%F9 v1.1%'
```

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

avg("PAYLOAD_MASS__KG_")
2534.6666666666665

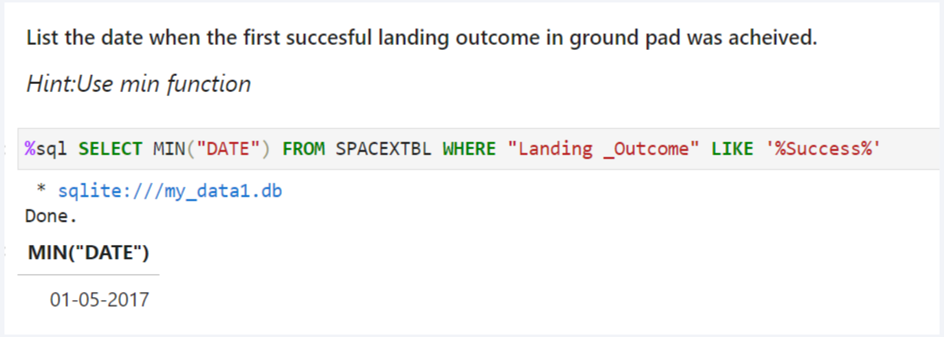
SQL Query

Results

Explanation

This query was returned the average of all payload masses where the booster version contains the substring F9 v1.1.

First Successful Ground Landing Date



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

Hint:Use min function

```
%sql SELECT MIN("DATE") FROM SPACEXTBL WHERE "Landing _Outcome" LIKE '%Success%'
```

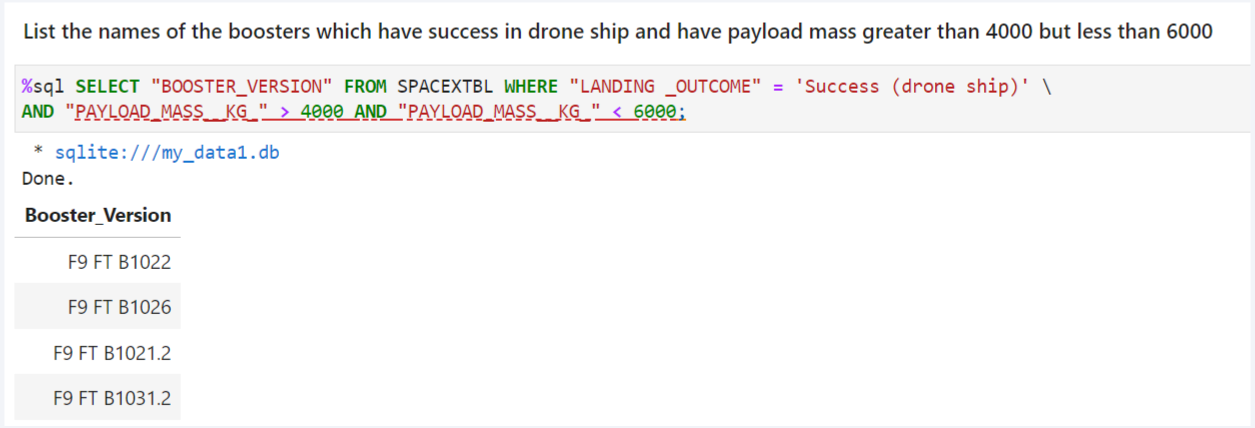
```
 * sqlite:///my_data1.db
Done.
```

MIN("DATE")
01-05-2017

Explanation

This query was returned the first successful ground landing date. The WHERE clause filters dataset in order to keep only records for those successful. Using the MIN function, we can select the record with the oldest date.

Successful Drone Ship Landing with Payload between 4000 and 6000



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

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

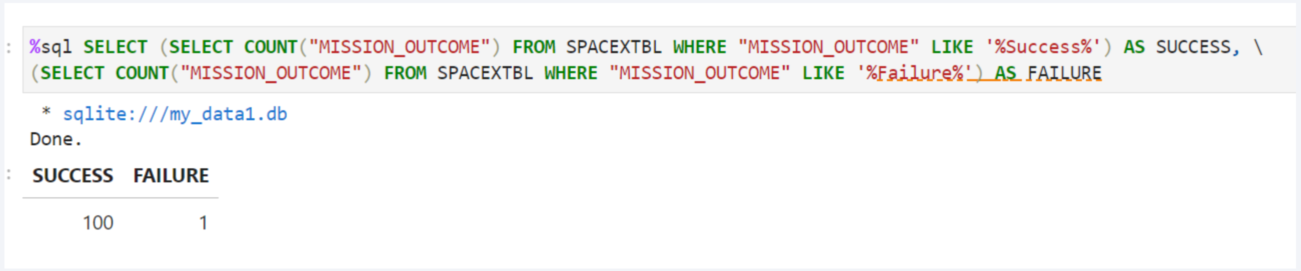
SQL Query

Results

Explanation

This query was returned the booster version where landing was successful and payload mass was between 4000 and 6000 kg. The WHERE and AND clauses filter the dataset.

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 '%Failure%') AS FAILURE
```

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

SUCCESS	FAILURE
100	1

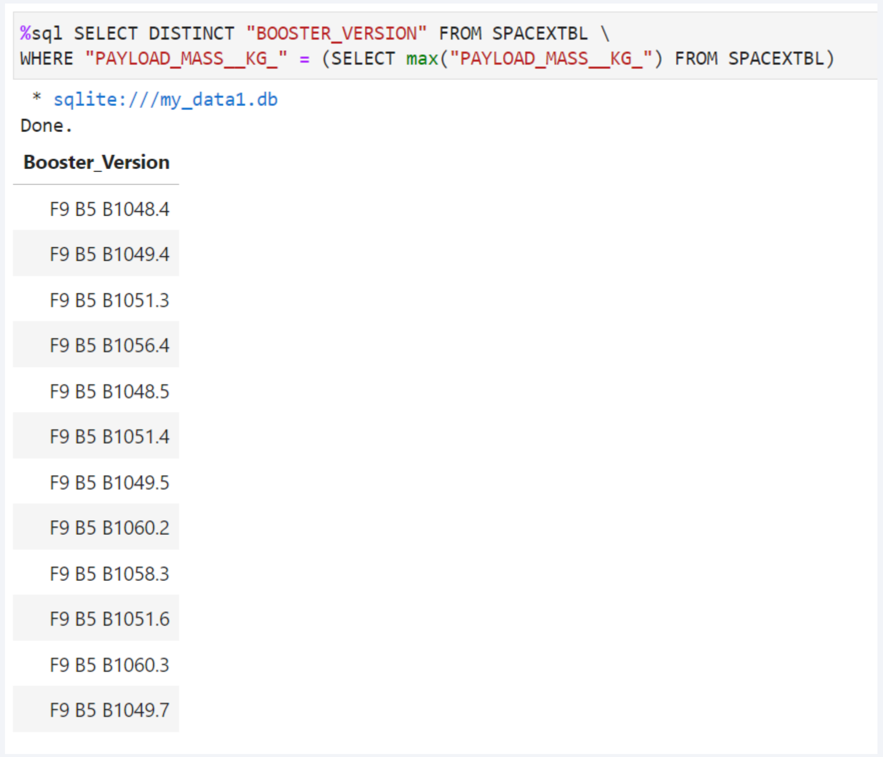
SQL Query

Results

Explanation

- The first SELECT showed the subqueries that return results of the first subquery counts the successful mission.
- The second subquery counts the unsuccessful launch. The WHERE clause followed by LIKE clause filters mission outcome. The COUNT function counts records filtered.

Boosters Carried Maximum Payload



```
%sql SELECT DISTINCT "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

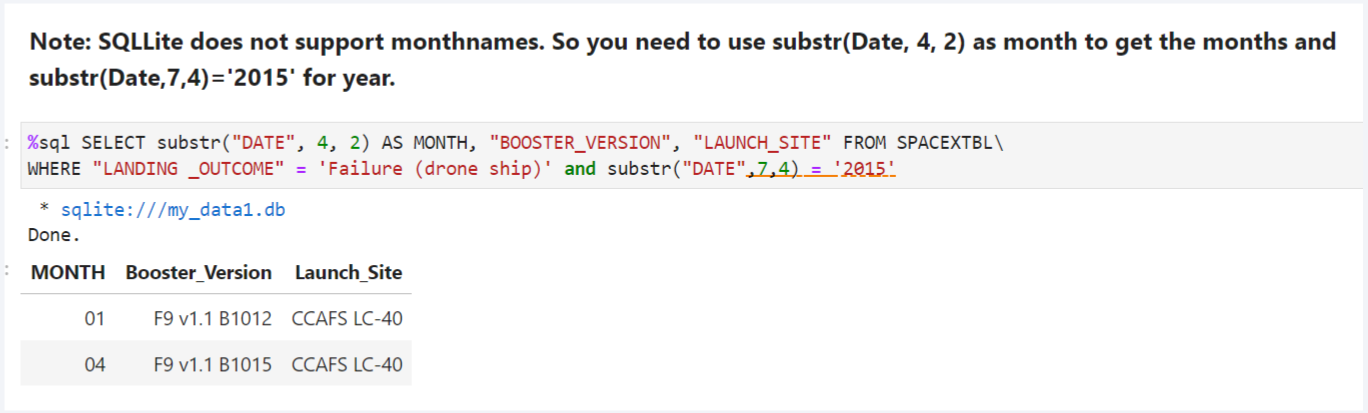
SQL Query

Results

Explanation

- We used a subquery to filter data by returning only the heaviest payload mass with MAX function.
- The main query uses subquery results and returns unique booster version (SELECT DISTINCT) with the heaviest payload mass.

2015 Launch Records



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.

```
%sql SELECT substr("DATE", 4, 2) AS MONTH, "BOOSTER_VERSION", "LAUNCH_SITE" FROM SPACEXTBL\
WHERE "LANDING _OUTCOME" = 'Failure (drone ship)' and substr("DATE",7,4) = '2015'
```

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

MONTH	Booster_Version	Launch_Site
01	F9 v1.1 B1012	CCAFS LC-40
04	F9 v1.1 B1015	CCAFS LC-40

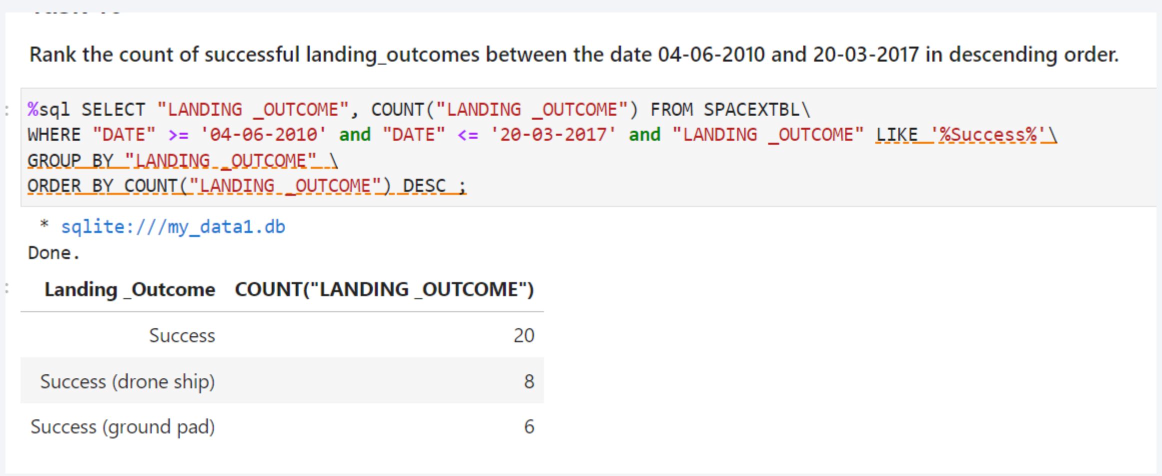
SQL Query

Results

Explanation

- This query was returned month, booster version, launch site where landing was unsuccessful and landing date took place in 2015.
- We used substr function to handle date in order to take month/year.
- Substr(Date, 4, 2) showed months
- Substr(Date, 7, 4) showed years.

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



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

```
%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 ;
```

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

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

SQL Query

Results

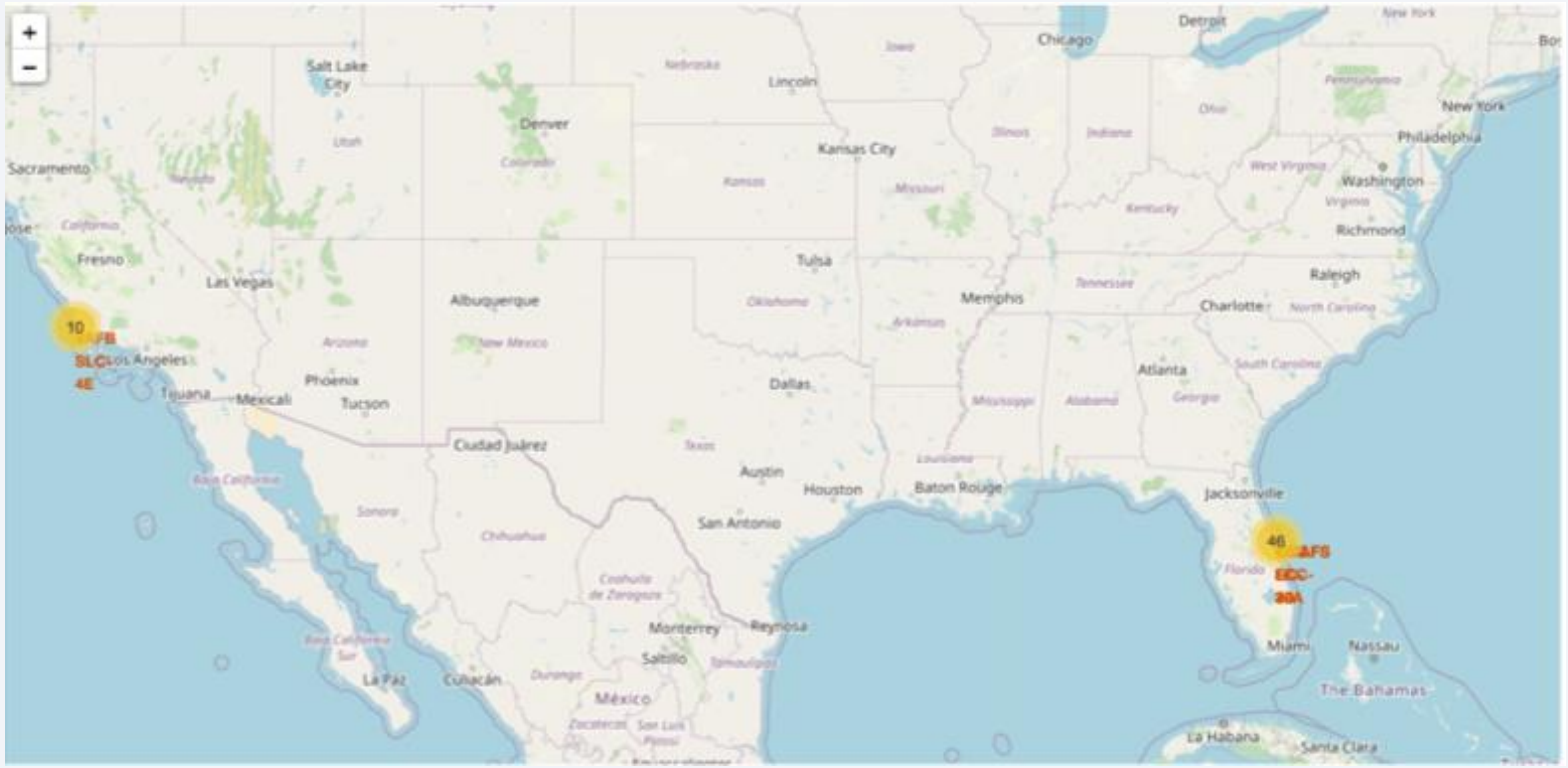
Explanation

- This query was returned landing outcomes and their count where mission was successful and date is between 04/06/2010 and 20/03/2017.
- The GROUP BY function groups results by landing outcome.
- The ORDER BY COUNT DESC showed the results in decreasing order.

Section 3

Launch Sites Proximities Analysis

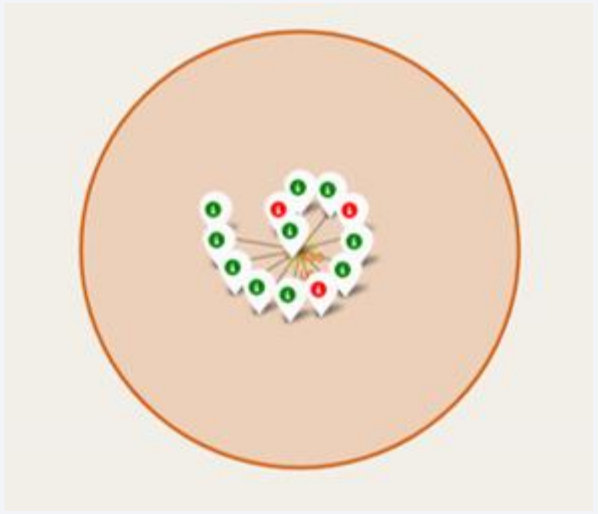
Folium Map – Ground Sites



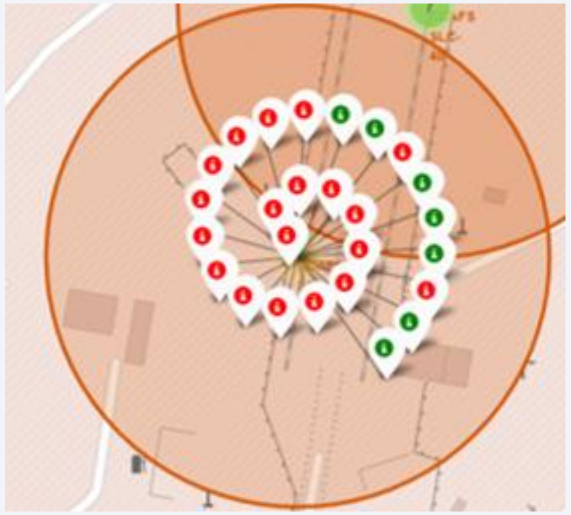
- We can see Space X launch sites are located in the east and west coastline of US.

Folium Map – Color Labeled Markers

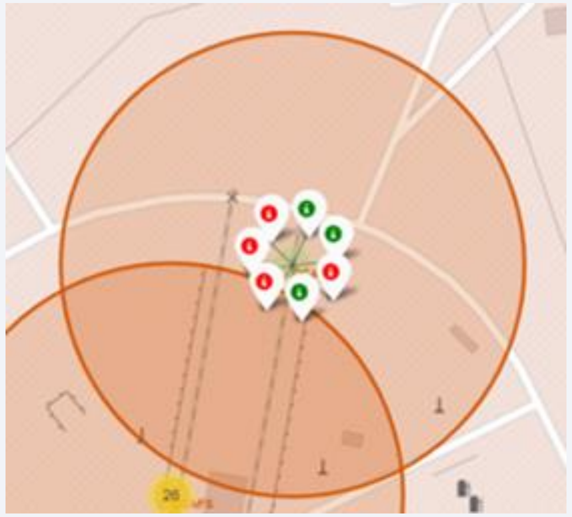
KSC LC-39A



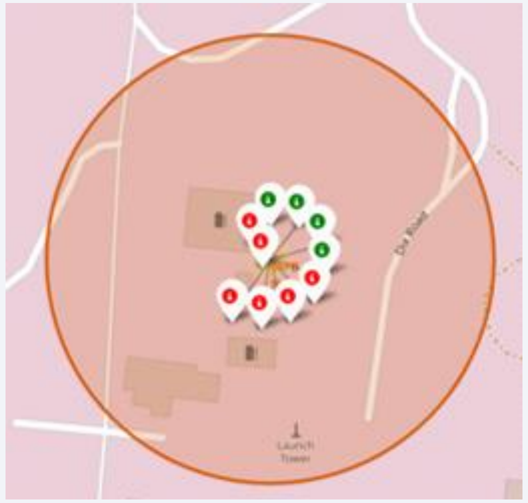
CCAFS LC-40



CCAFS SLC-40

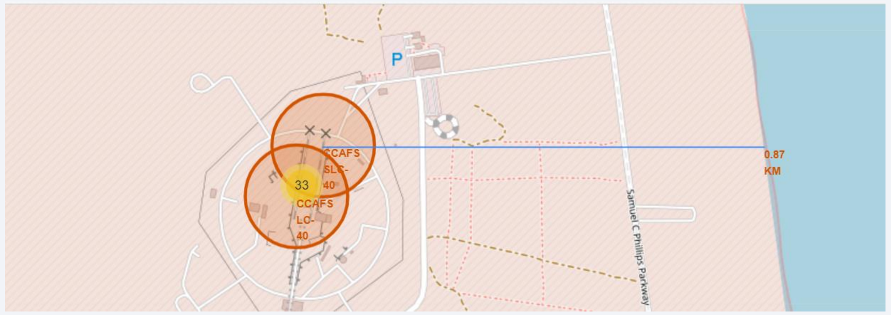


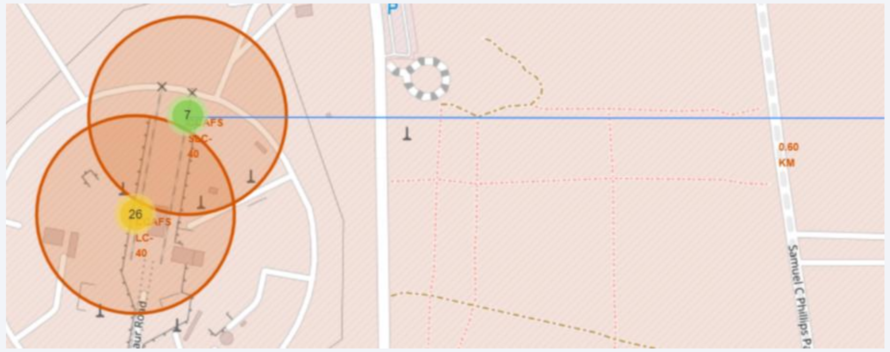
VAFB SLC-4E

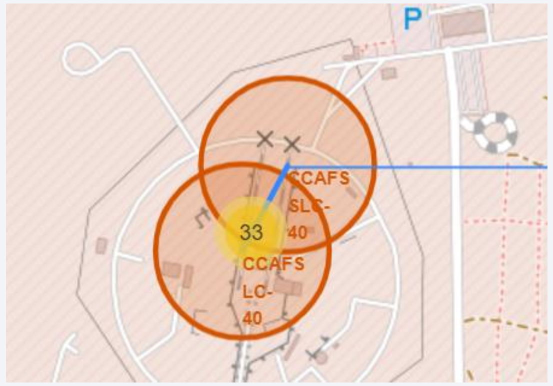


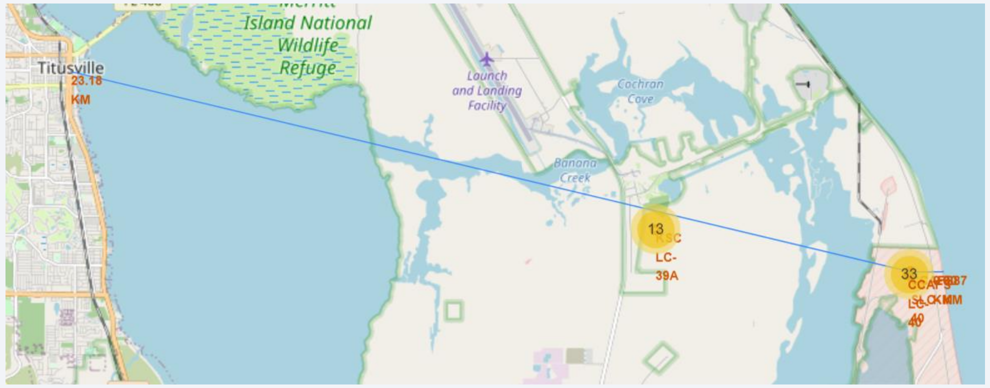
Green label represent successful launch and Red label represent the unsuccessful launch. We can see from the Markers that note that KSC LC-39A has a higher launch success rate.

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







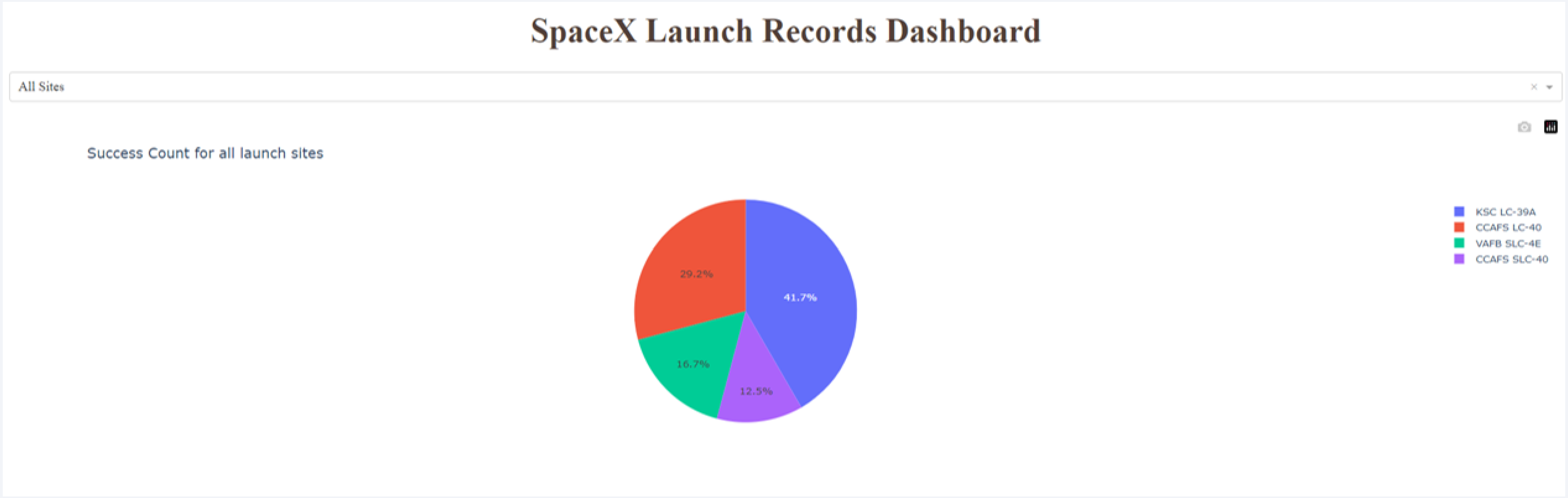


Is CCAFS SLC-40 in close proximity to railways ? Yes
Is CCAFS SLC-40in close proximity to highways ? Yes
Is CCAFS SLC-40in close proximity to coastline ? Yes
DoCCAFS SLC-40keeps certain distance away from cities ? No

Section 4

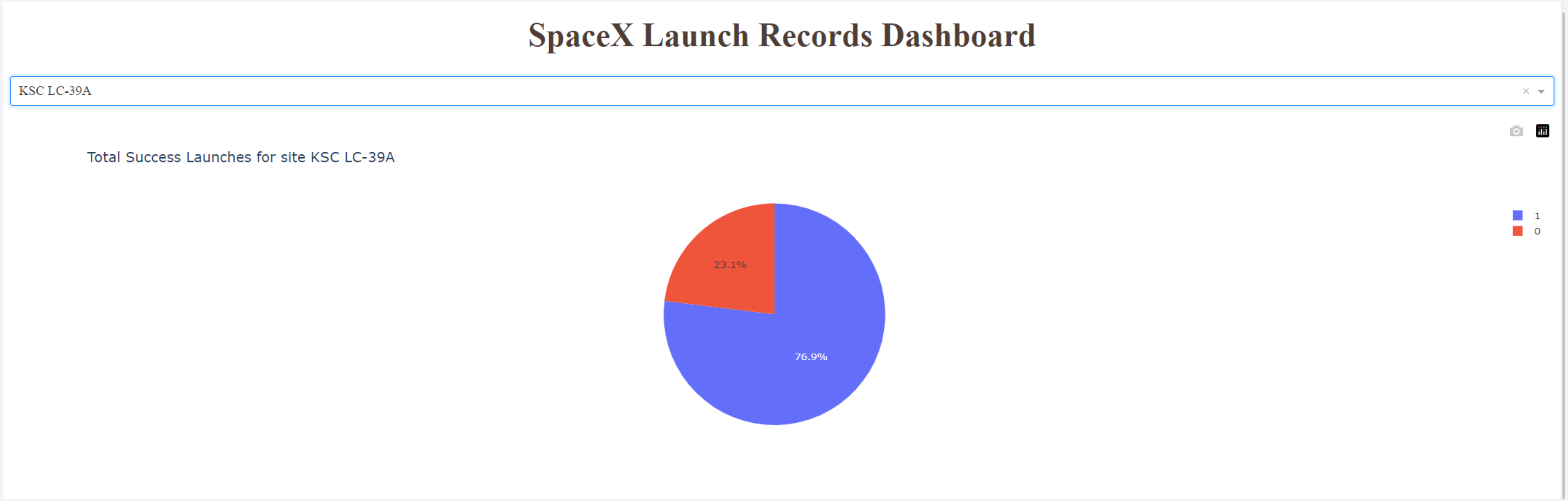
Build a Dashboard with Plotly Dash

Dashboard – Total success by Site



- The above picture showed that KSC LC – 39A site has the highest successful rate.

Dashboard – Total Success for Site KSC LC-39A



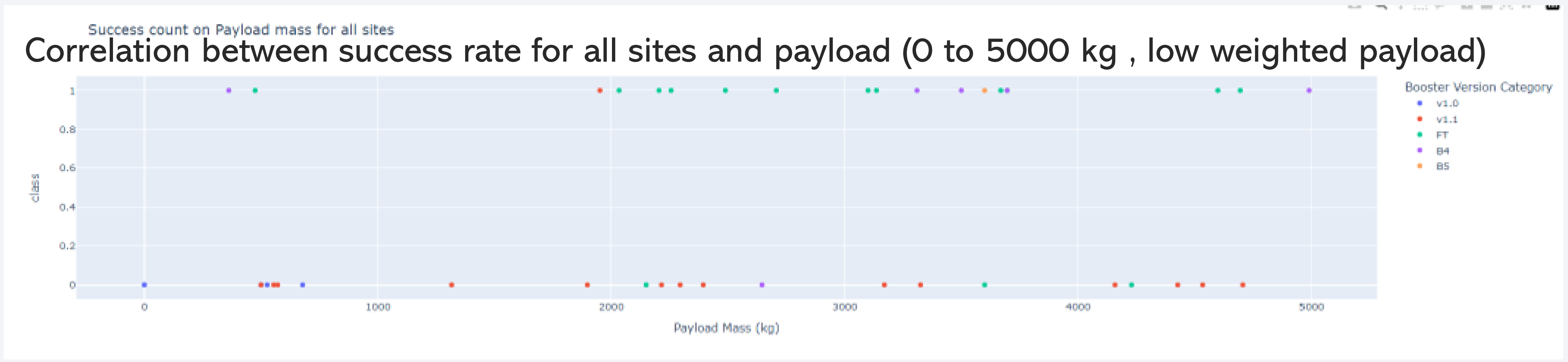
SpaceX Launch Records Dashboard

KSC LC-39A

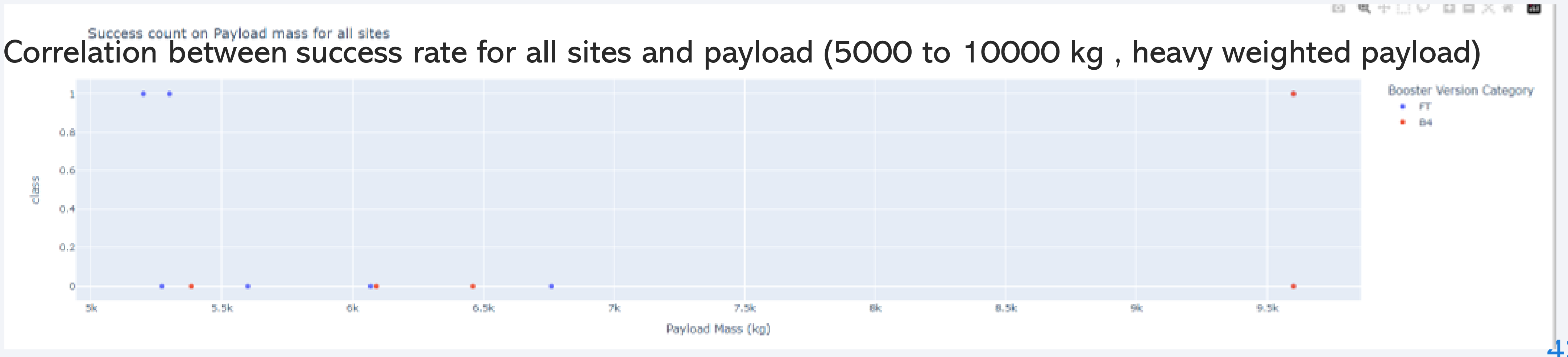
Total Success Launches for site KSC LC-39A

- The above picture showed that KSC LC – 39A site has the highest successful rate and it achieved 76.9% success rate, only it getted 23.1% failure rate.

Dashboard – Payload mass vs Outcome for all sites with different payload mass selected



Correlation between success rate for all sites and payload (0 to 5000 kg , low weighted payload)



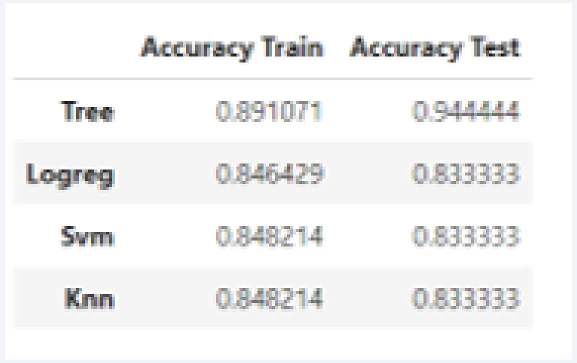
Correlation between success rate for all sites and payload (5000 to 10000 kg , heavy weighted payload)

The successful rate of low weighted payload were better than heavy weighted payload in all site

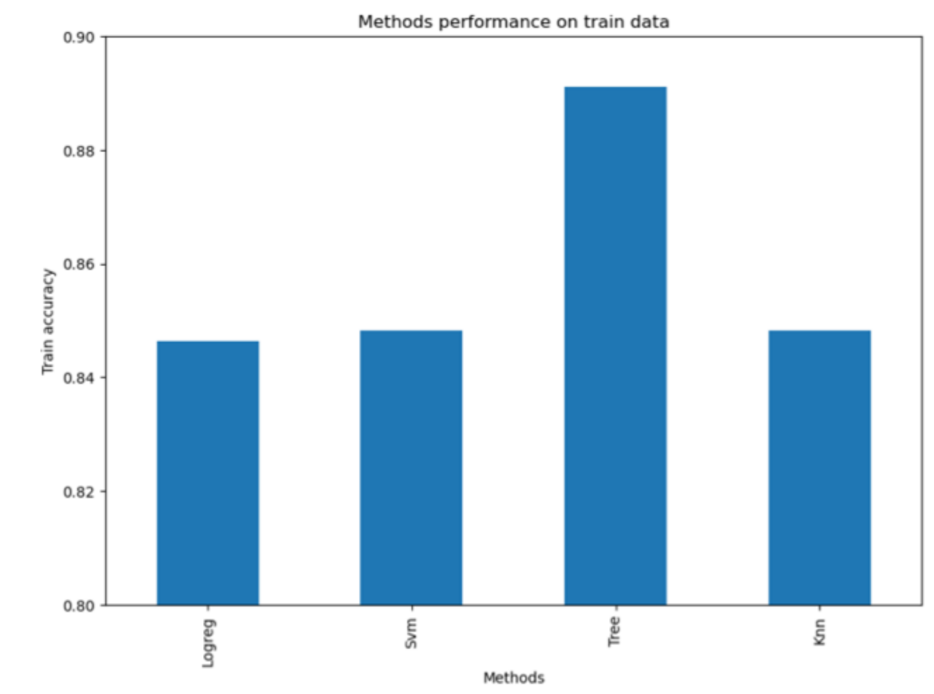
Section 5

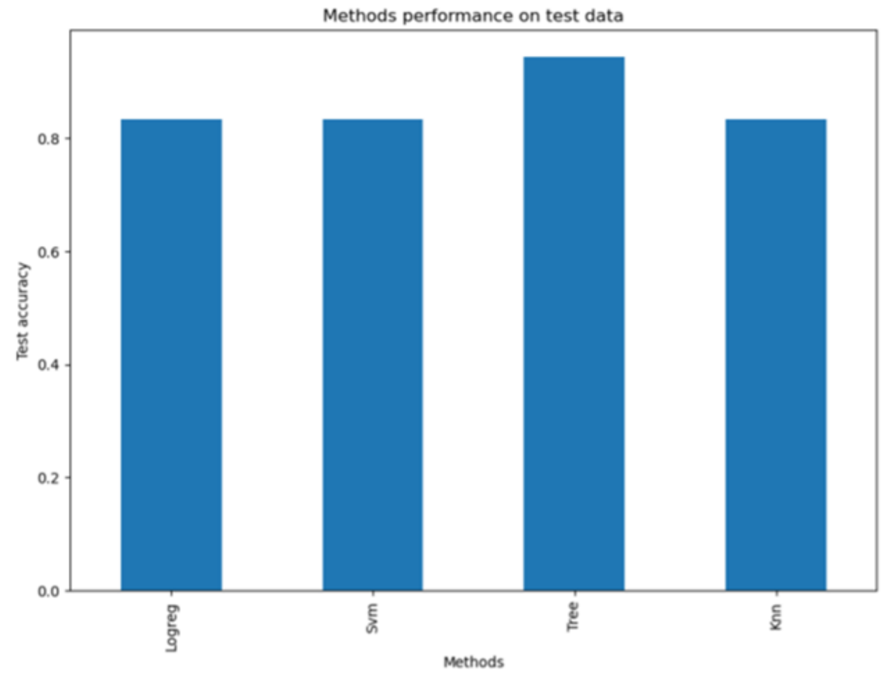
Predictive Analysis (Classification)

Classification Accuracy



	Accuracy Train	Accuracy Test
Tree	0.891071	0.944444
Logreg	0.846429	0.833333
Svm	0.848214	0.833333
Knn	0.848214	0.833333

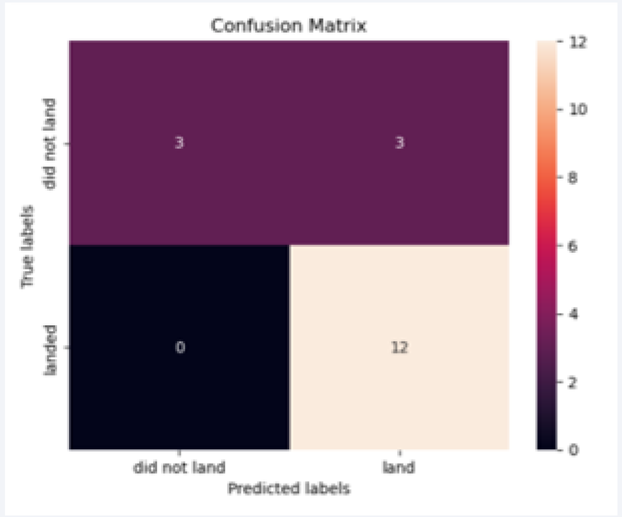




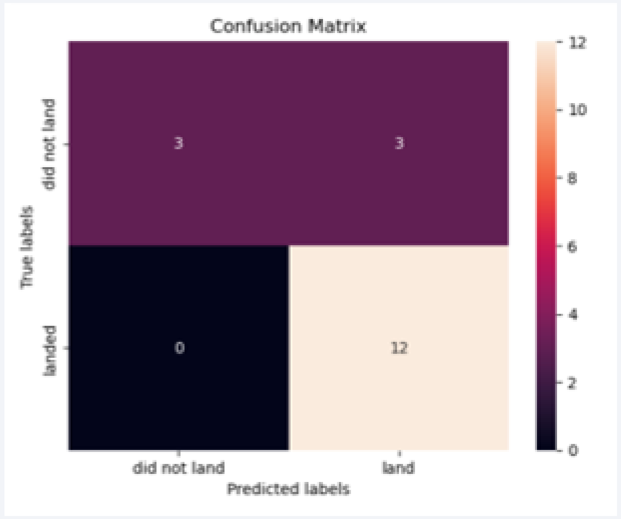
- The bar chats and the table showed that decision Tree model got the highest accuracy both in training dataset and test dataset.

Confusion Matrix

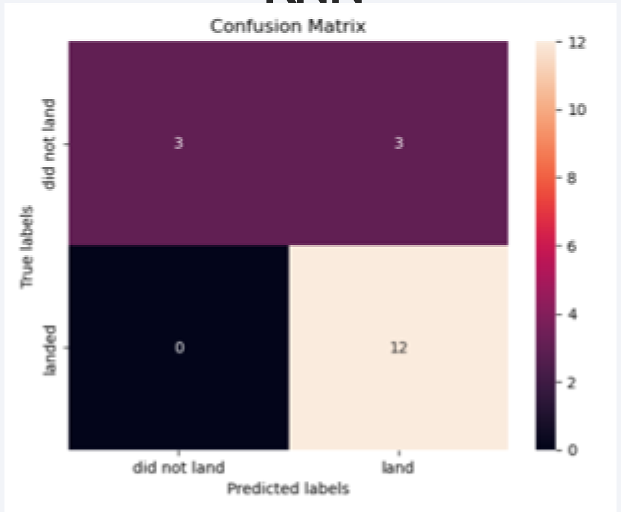
Logistic Regression



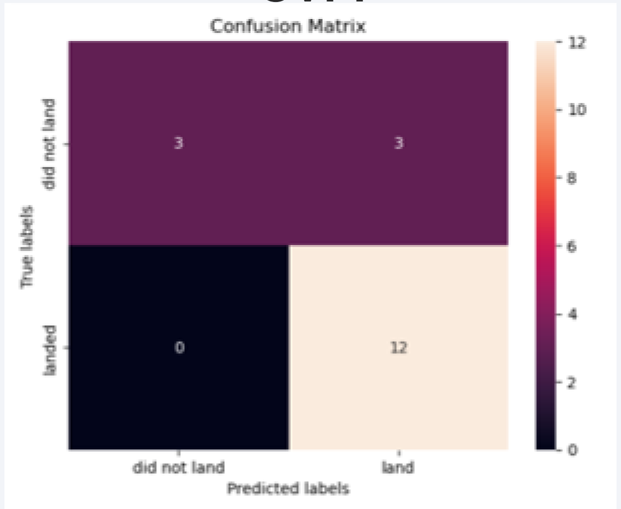
Decision Tree



KNN



SVM



We can see accuracy result are the same in four training models.

The common problems in these models were false positive.

Conclusions

- The successful launch of rocket can be influenced by number of factors such as the launch site, the orbit type and the number of previous launches in the same site and type. We can assume the accumulation of experience was solid ground to have a successful launch of rocket.
- The orbits with the best success rates were GEO, HEO, SSO, ES-L1.
- Depending on the type of orbits, the payload mass can be major factor for the successful launch. Some orbits require a light or heavy payload mass. Therefore, the suitability of how much weight payload mass for different type of orbit are important.
- In general speaker, the lighter payload mass got better performance than the heavy one.
- We cannot explain why some launch sites are better than others (KSC LC-39A is the best launch site) according to the current dataset and the source of data. We need to collect the climate and environment data of different sites to answer this enquiry.
- According to current dataset and predictive analysis result, the Decision Tree Algorithm was the best model because it has a better accuracy both in training and test dataset.

Thank you!