IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Name>Opeyemi David
<Date>23/6/2023

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection via API, Web Scrapping
  - Data Wrangling
  - Exploratory Data Analysis (EDA), with SQL and Visualization
  - Interactive Visual Analytic with Dashboards, Folium, and Ploty
  - Predictive Analysis
- Summary of all results
  - Data Collection via API, Web Scrapping
  - Exploratory Data Analysis Results
  - Interactive Maps and Dashboard
  - Predictive Results

Introduction

- Project background and context

To predict if the Falcon 9 first stage will land successfully. SpaceX 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 SpaceX can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information is needed by Space Y to compete with Space X for a rocket launch

- Problems you want to find answers
  - Data Collection via API, Web Scrapping
  - What is the outcome of the landing?
  - What are the determinants of failed and successful landing?
  - What are the conditions for success rate landing?
  - What will determine the best launch cost?

Section 1

Methodology

Methodology

Executive Summary

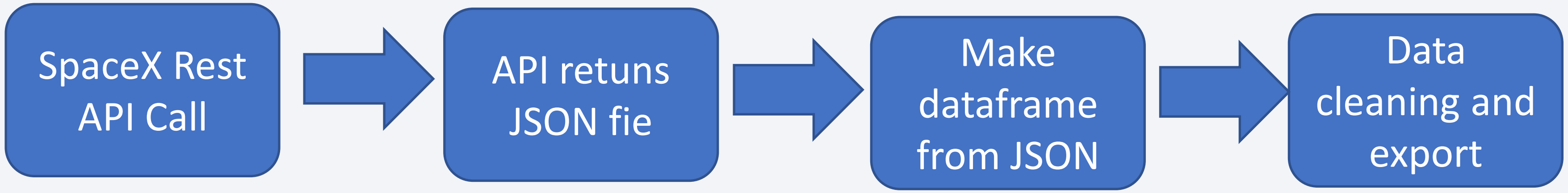
- Data collection methodology:
  - Space X Rest API
  - Web Scraping
- Perform data wrangling
  - Identified and calculated the missing values
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- Describe how data sets were collected.

Data sets were collected via API, and, Webscrapping (Wikipedia)

Some columns such as rocket, payloads, and launchpad were used to obtain useful information such as launch site, booster name, payload mass, and so on
https://api.spacexdata.com



Some of the information obtained from the Websrapping (Wikipedia) are launch site, payload mass, orbit, customer e.t.c
"https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"

Data Collection – SpaceX API

1. Requesting launch data from SpaceX API

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

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

2. Convert response to json file

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

3. Data transformation

```
# 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)
```

4. Create a 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}
```

5. Create dataframe

```
# Create a data from launch_dict
data_falcon9 = pd.DataFrame.from_dict(launch_dict)
```

6. Filter dataframe

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

7. Export to CSV file

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

Link to code

Data Collection - Scraping

1. Getting html response

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

2. Create beautifulsoup object

```
# Use BeautifulSoup() to create a BeautifulSc
soup = BeautifulSoup(data, "html.parser")
```

3. Find the tables

```
# Assign the result to a list called
html_tables=soup.find_all('table')
```

4. Get column names

```
column_names = []

for row in first_launch_table.find_all('th'):
    name = extract_column_from_header(row)
    if (name != None and len(name) > 0):
        column_names.append(name)
```

5. Create a 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.`
            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)
```

7. Create a dataframe from the dictionary

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

8. Export to file

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

[Link to code](#)

Data Wrangling

- Finding the missing data

```
data_falcon9.isnull().sum()

FlightNumber       0
Date               0
BoosterVersion     0
PayloadMass        5
Orbit              0
LaunchSite         0
Outcome            0
Flights            0
GridFins           0
Reused             0
Legs               0
LandingPad        26
Block              0
ReusedCount        0
Serial             0
Longitude          0
Latitude           0
dtype: int64
```

- Dealing with missing data

```
# Calculate the mean value of PayloadMass column
PayloadMass_mean = data_falcon9["PayloadMass"].mean()
# Replace the np.nan values with its mean value
data_falcon9["PayloadMass"]= data_falcon9["PayloadMass"].replace(np.nan, PayloadMass_mean)

/tmp/wsuser/ipykernel_164/1083235087.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/i
-view-versus-a-copy
  data_falcon9["PayloadMass"]= data_falcon9["PayloadMass"].replace(np.nan, PayloadMass_mean)

data_falcon9.isnull().sum()

FlightNumber       0
Date               0
BoosterVersion     0
PayloadMass        0
Orbit              0
LaunchSite         0
Outcome            0
Flights            0
GridFins           0
Reused             0
Legs               0
LandingPad        26
Block              0
```

[Link to code]

EDA with Data Visualization

Scatter Plot

Flight Number vs Payload Mass

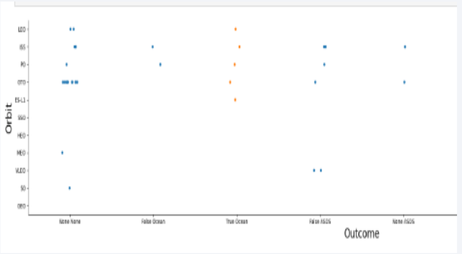
Flight Number vs Launch Site

Payload Mass vs Launch Site

Orbit vs Flight Number

Orbit vs Outcome

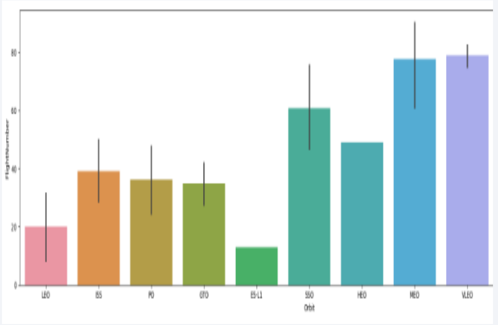
Orbit vs Payload Mass



Bar Chat

Orbit vs Flight Number

Orbit vs Payload Mass



Line Chart

Date vs Success Launch Set

[Link to code]

EDA with SQL

- SQL queries were used to perform the following

✓ 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 succesful 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. Use a subquery

✓ 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

- The launch sites were marked with red markers
- Success and failed launches are also marked on the map
- Distance between launch sites and their proximity is calculated using the coordinate

The objects on the maps are added for easy reading

[Link to code]

Build a Dashboard with Plotly Dash

- Dashboard has a dropdown such as pie chart, rangeslider and scatter plot components
- Dropdown allow users to choose launch site
- Pie chart shows the total success and failure for the launch site chosen with the dropdown components
- Rangeslider allows a user to select a payload mass in fixed range
- Scatter chart displays the relationship between two variables, Success and Payload Mass

[Link to code]

Predictive Analysis (Classification)

- Data Preparation

  Load dataset

  Data normalization

  Splitting data into training and test sets

- Model preparation

  Selecting a machine learning algorithm

  Set parameter for each algorithm to GrivsearchCV

  Train the Gridsearch model with the training dataset

- Model evaluation

  Get the best hyperparameter for each type of model

  Compute accuracy for each model with the test dataset

  Plot confusion matrix

- Model comparison

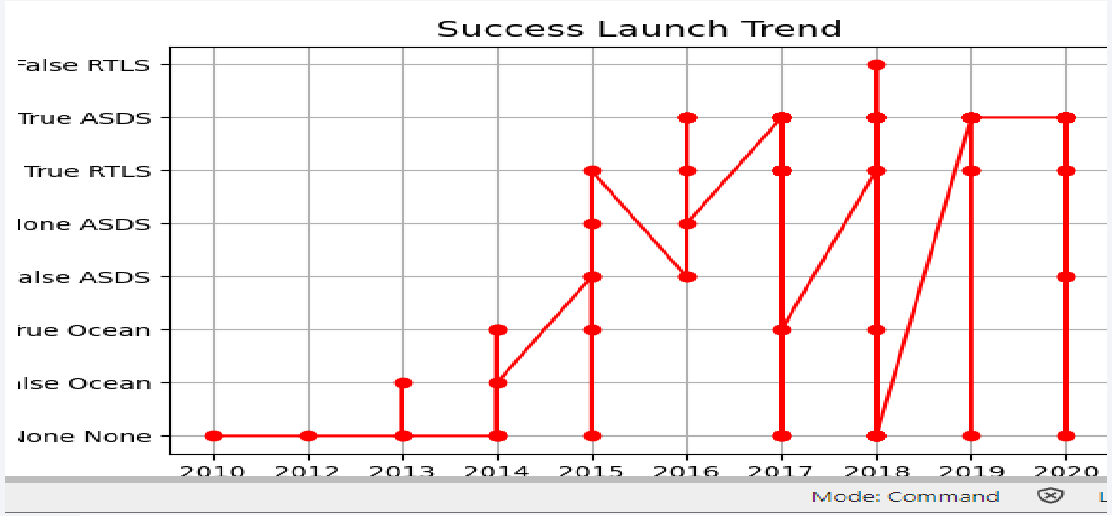
  Comparison of models according to accuracy

  The model with the best accuracy will be chosen

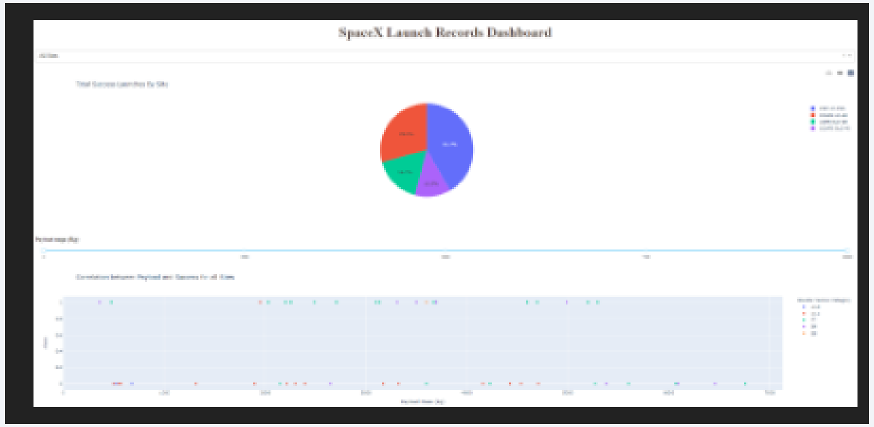
[Link to code]

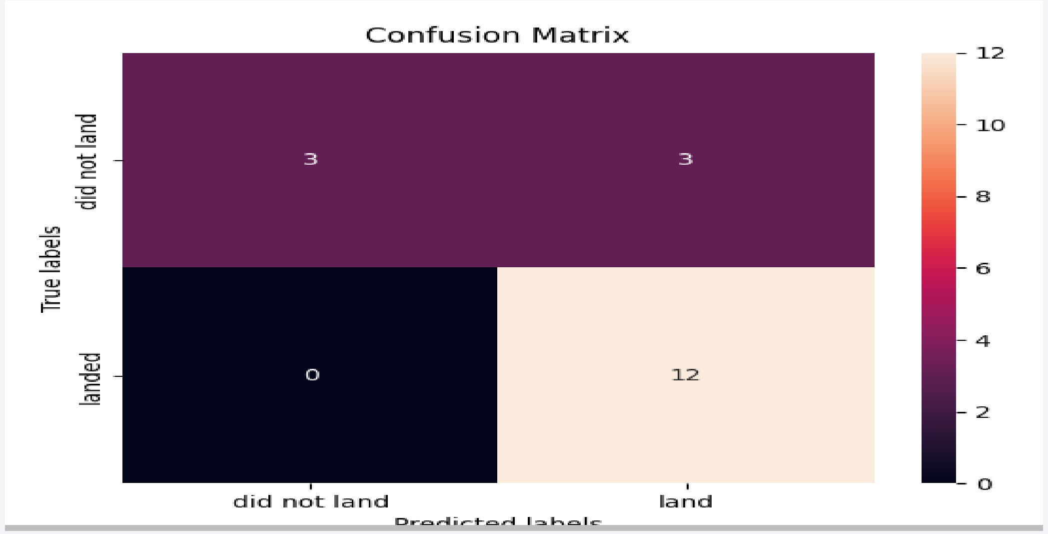
Results

Exploratory data analysis result



Interactive analytics



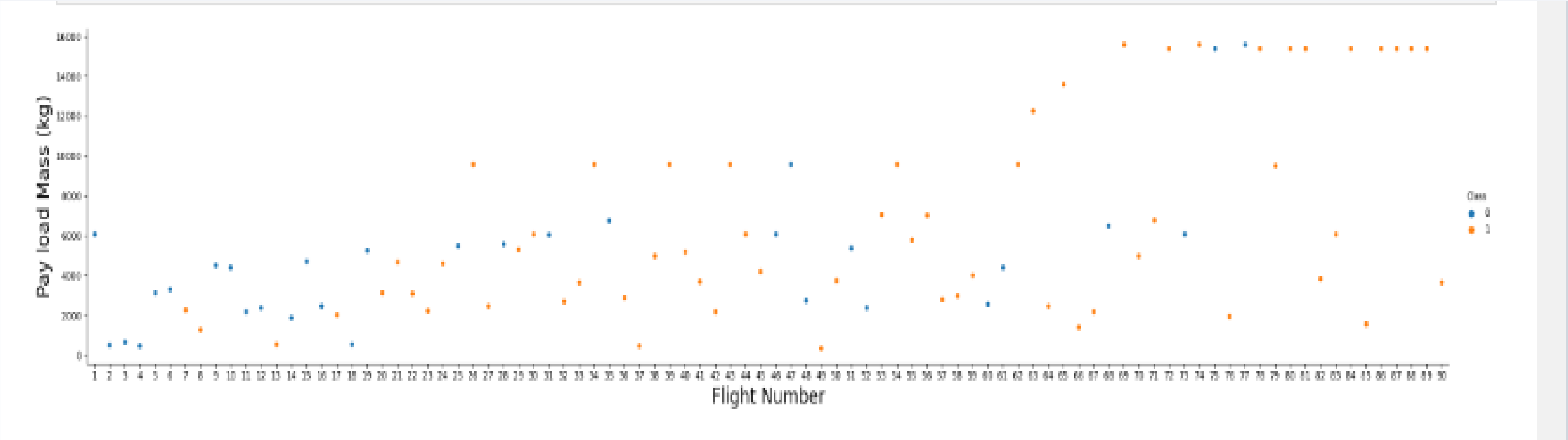


Predictive analysis results

Section 2

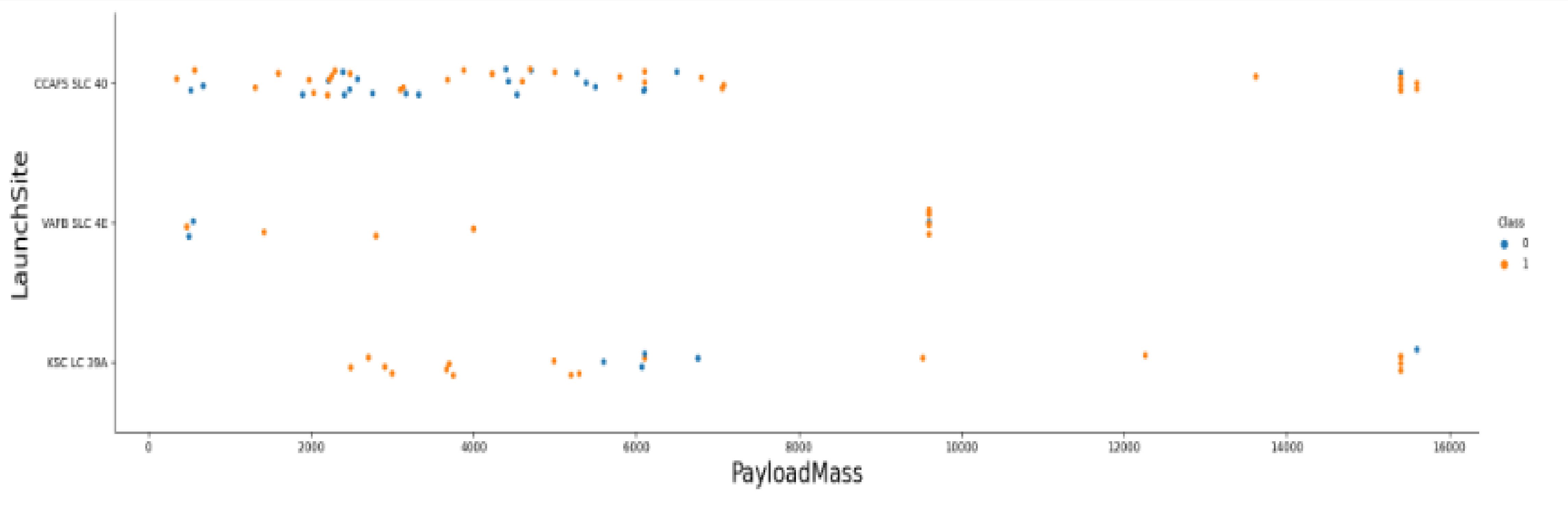
Insights drawn from EDA

Flight Number vs. Launch Site



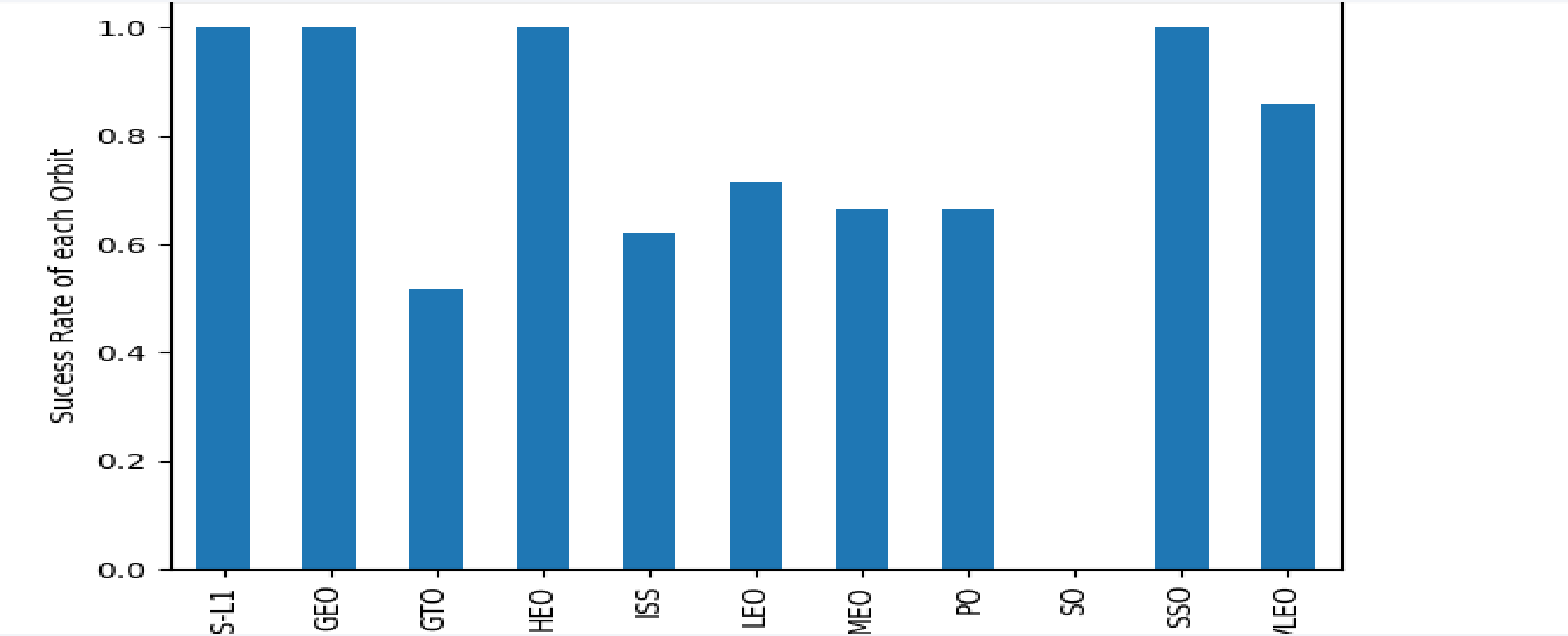
The success rate is increasing for each site

Payload vs. Launch Site



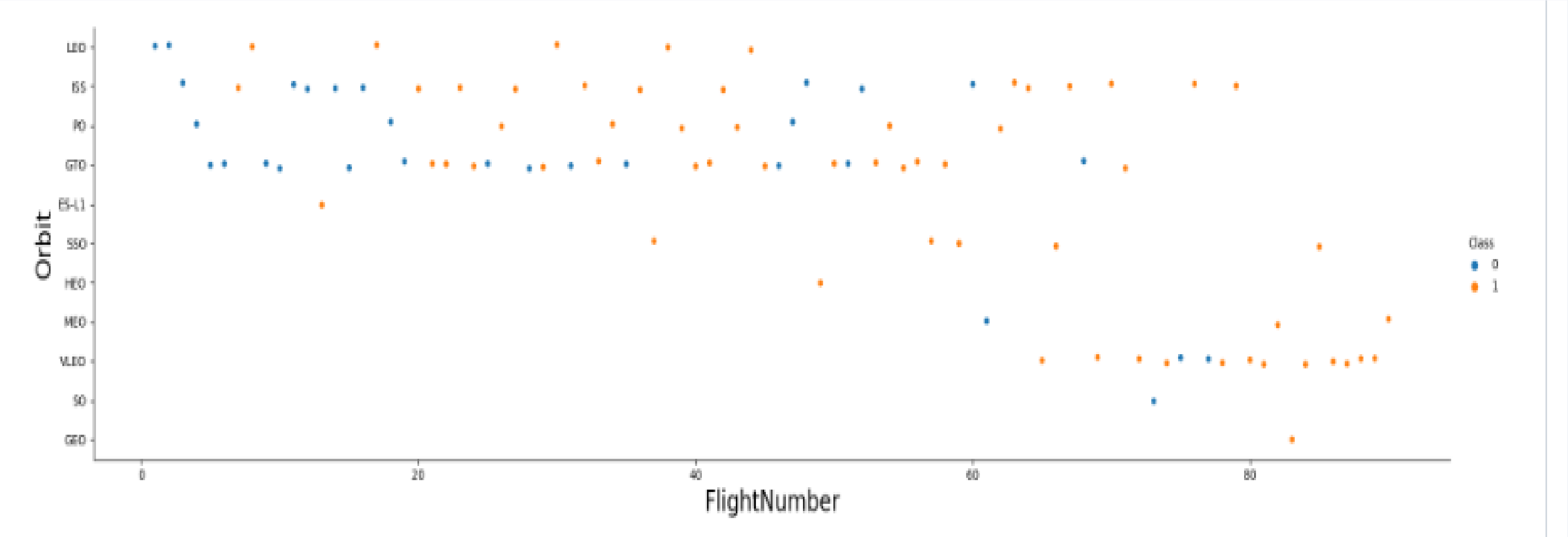
Depending on the launch site, heavy payload may lead to successful landing or failure in landing

Success Rate vs. Orbit Type



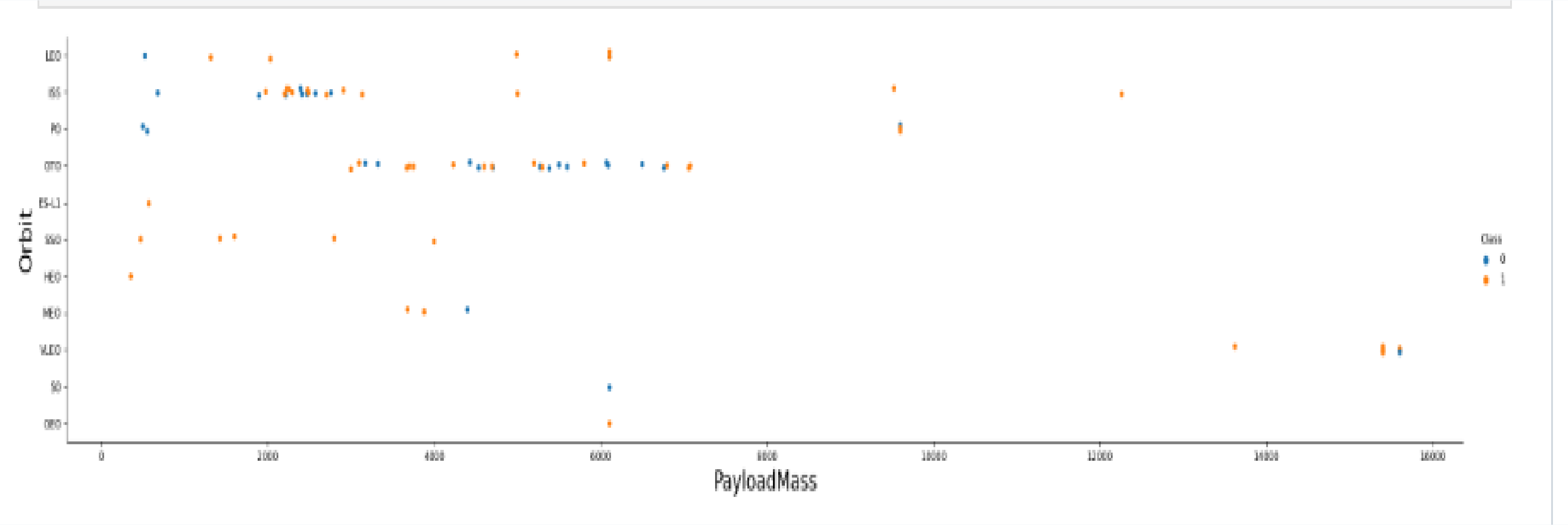
It can be deduced that ES-L1, GEO, HEO, and SSO have the best success rate

Flight Number vs. Orbit Type



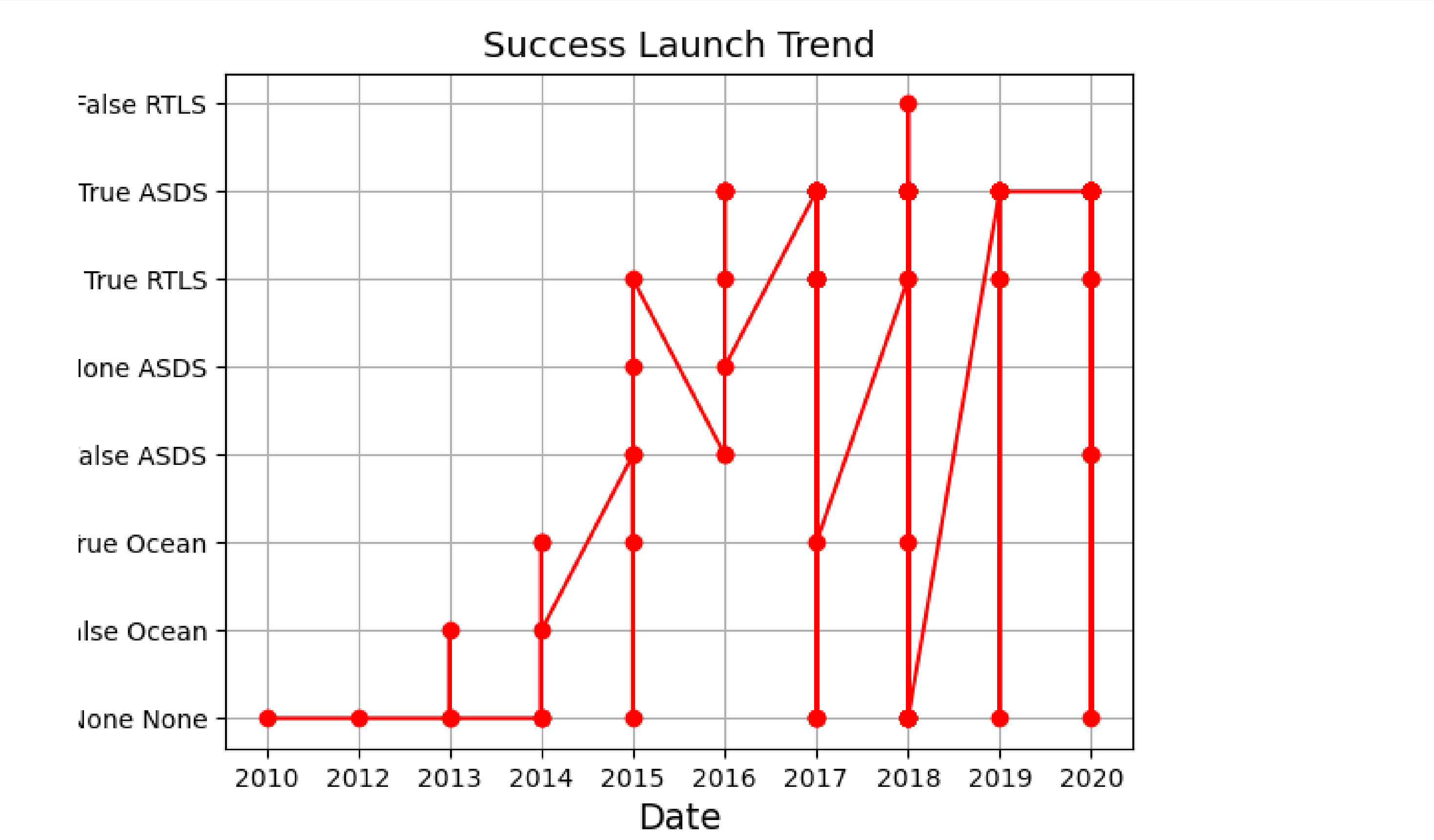
The success rate improves with the number of flights for the GS, LEO orbit, the success rate of HEO and SSO may be due to the knowledge learned during former launches

Payload vs. Orbit Type



The weight of the payload has an effect on the success rate of launches in some orbits. Heavy payload increases the success rate of GTO

Launch Success Yearly Trend



The success rate increases from year 2013 till 2020

All Launch Site Names

```
%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
None

The query SELECT DISTINCT allows you to selection Launch_Site without duplicate

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-	15:10:00	F9 v1.0 B0007	CCAFS LC-	SpaceX	677	LEO	NASA	Success	No

The Like clause filters the Launch_Sites that have the substring CCA, while Limit 5 reduced the outcome to 5

Total Payload Mass

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

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

Customer	sum(PAYLOAD_MASS__KG_)
NASA (CRS)	45596

The sum clause in the query sum and gives the total payload_mass, and the where clause specifies the customer NASA(CRS)

Average Payload Mass by F9 v1.1

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

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

Booster_Version	avg(PAYLOAD_MASS__KG_)
F9 v1.1	2928.4

The avg clause calculated the average Payload_mass, and the where clause specified the Booster_version in the query

First Successful Ground Landing Date

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

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

min(Date)
01/08/2018

The min function in the query was used to specify the date, and the where clause specified the landing_outcome needed

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_ between
```

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

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

Clause and was used to pick the columns success(drone ship) and payload_mass_kg together and clause between to filter between 4000 and 6000

Total Number of Successful and Failure Mission Outcomes

```
%sql select count(Mission_Outcome) as Success_Mission_Outcome from SPACEXTBL where Mission_Outcome='Success'
```

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

Success_Mission_Outcome
98

```
%sql select count(Mission_Outcome) as Failure_Mission_Outcome  from SPACEXTBL where Mission_Outcome='Failure (in flig
```

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

Failure_Mission_Outcome
1

Clause count in the query was used to count the frequency of the values that we are looking for

Boosters Carried Maximum Payload

```
%sql SELECT BOOSTER_VERSION,PAYLOAD_MASS__KG_ FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_)
```

```
 * 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

Subquery was used to get the required output

2015 Launch Records

```
%sql SELECT substr(Date,4,2) as month, DATE,BOOSTER_VERSION, LAUNCH_SITE, [Landing _Outcome] FROM SPACEXTBL where [La
```

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

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

substr(Date, 4, 2) was used in the query to get the months and year

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

```
%sql SELECT [Landing _Outcome], count(*) as count_outcomes FROM SPACEXTBL WHERE DATE between '04-06-2010' and '20-03-
```

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

Landing _Outcome	count_outcomes
Success	20
No attempt	10
Success (drone ship)	8
Success (ground pad)	6
Failure (drone ship)	4
Failure	3
Controlled (ocean)	3
Failure (parachute)	2
No attempt	1

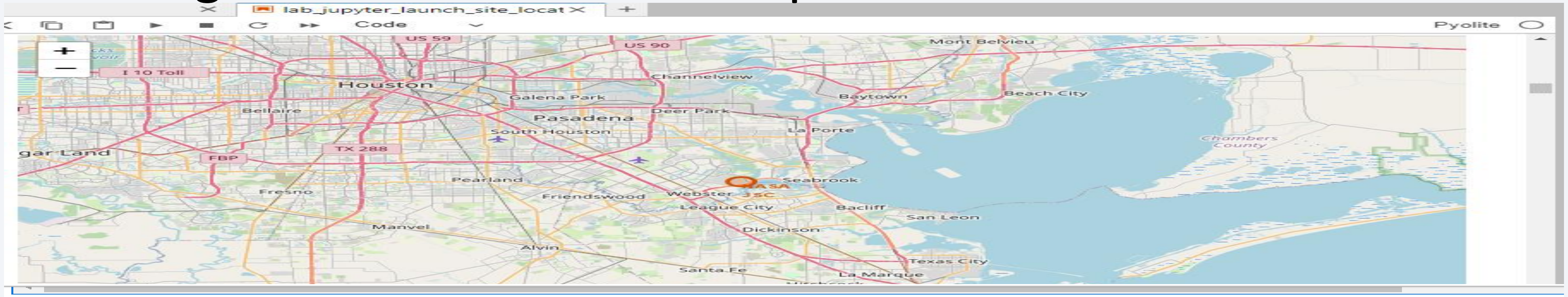
Clause count was used to get the frequency of the values, group by used to group them together, and order by clause is used to streamline the desired output

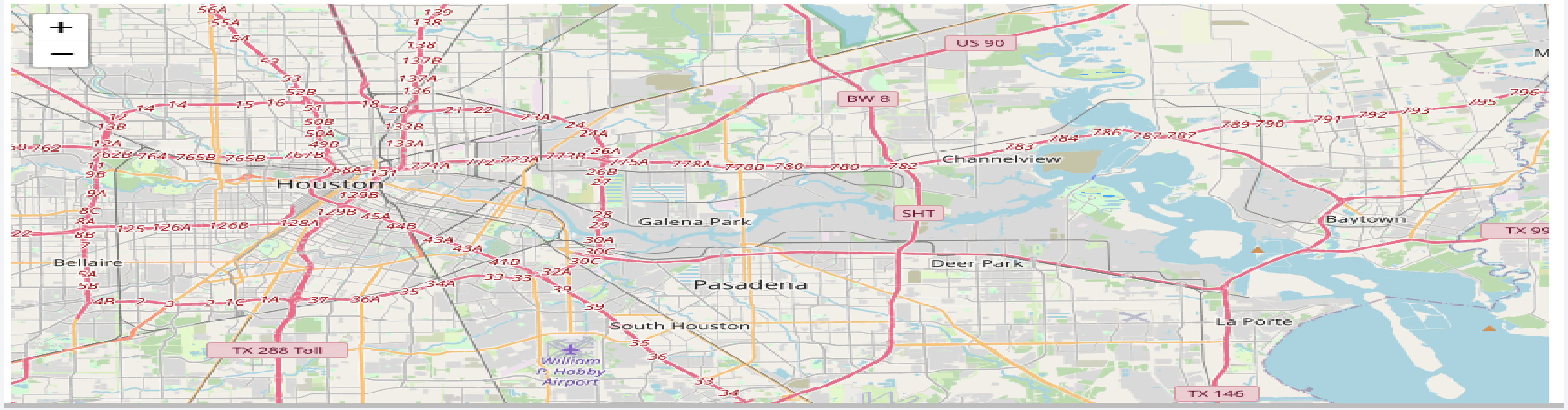
Section 3

Launch Sites Proximities Analysis

<Folium Map Screenshot 1>

Marking launch sites on the map

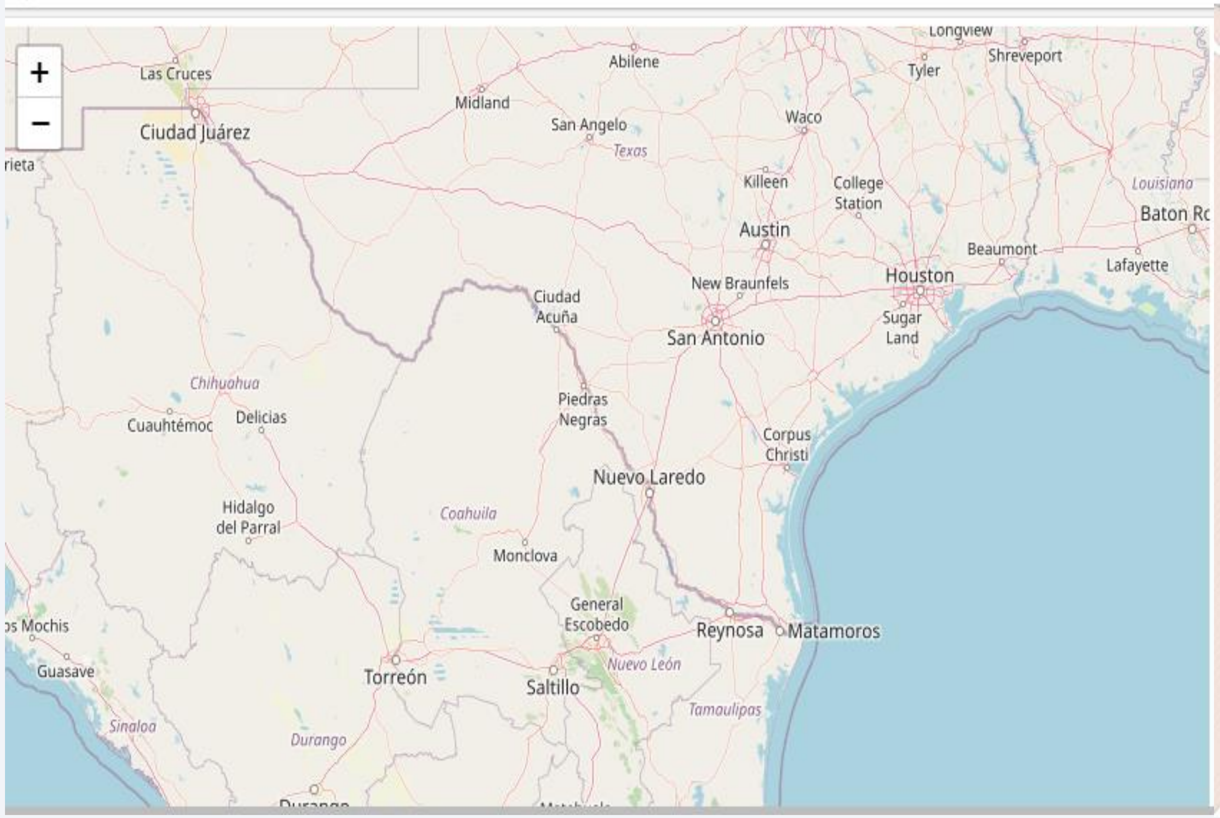




The maps display the launch sites

<Folium Map Screenshot 2>

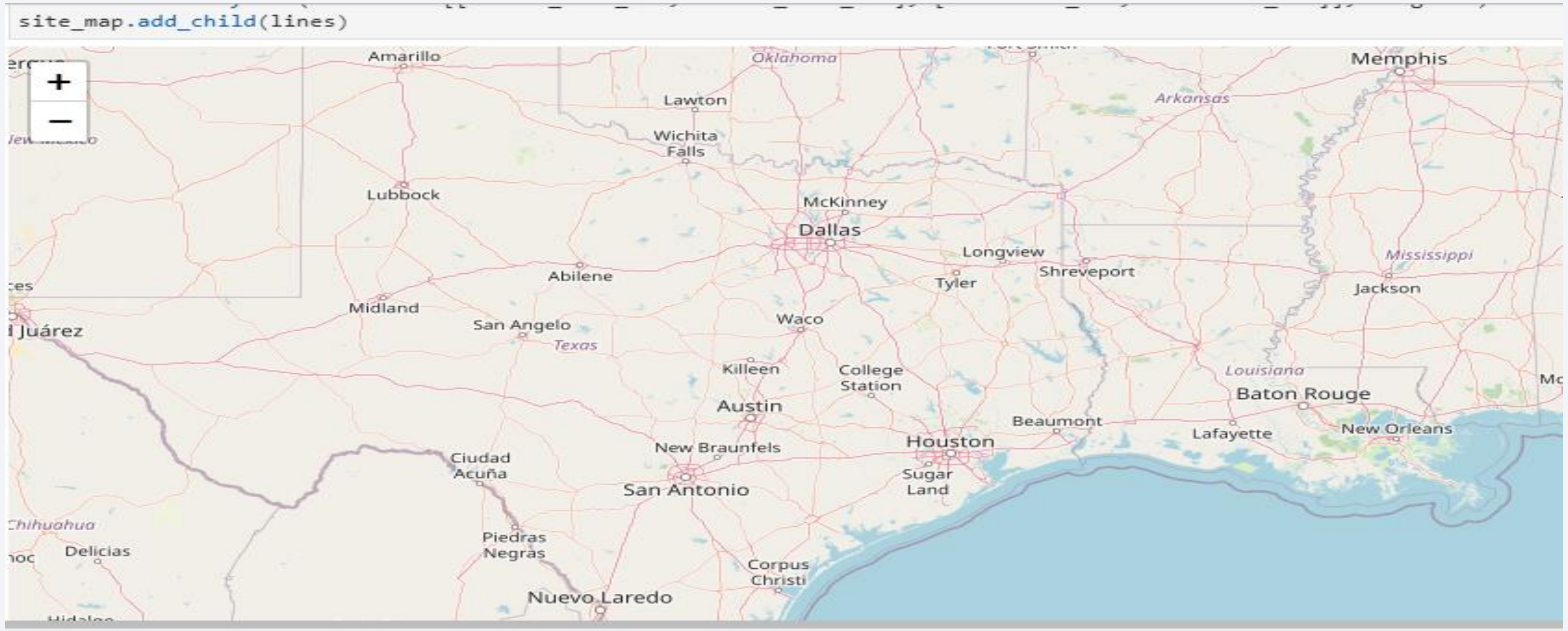
Success/Failed launches



The color labeled shows the launch outcome

<Folium Map Screenshot 3>

- Distance between Launch sites and its proximity



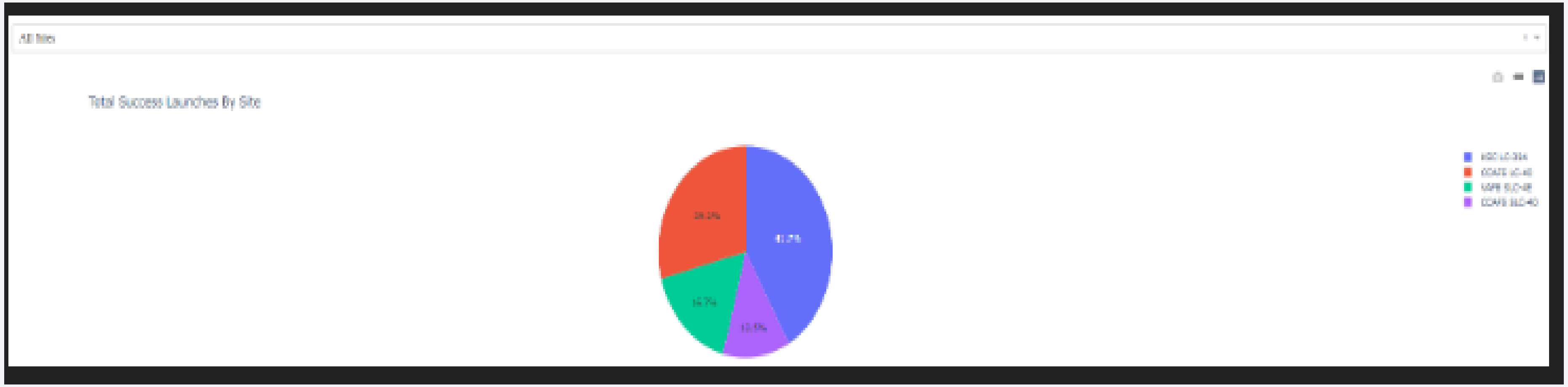
The launch sites and the proximities are displayed on the map

Section 4

Build a Dashboard with Plotly Dash

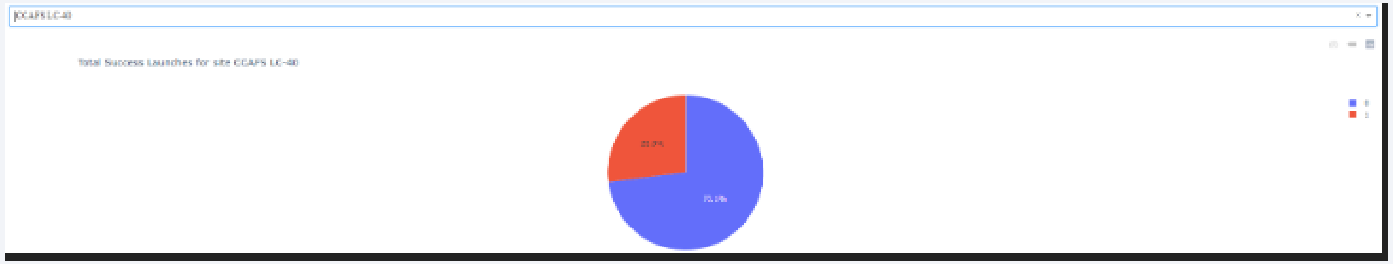
<Dashboard Screenshot 1>

- Piechart of success launch for all sites



<Dashboard Screenshot 2>

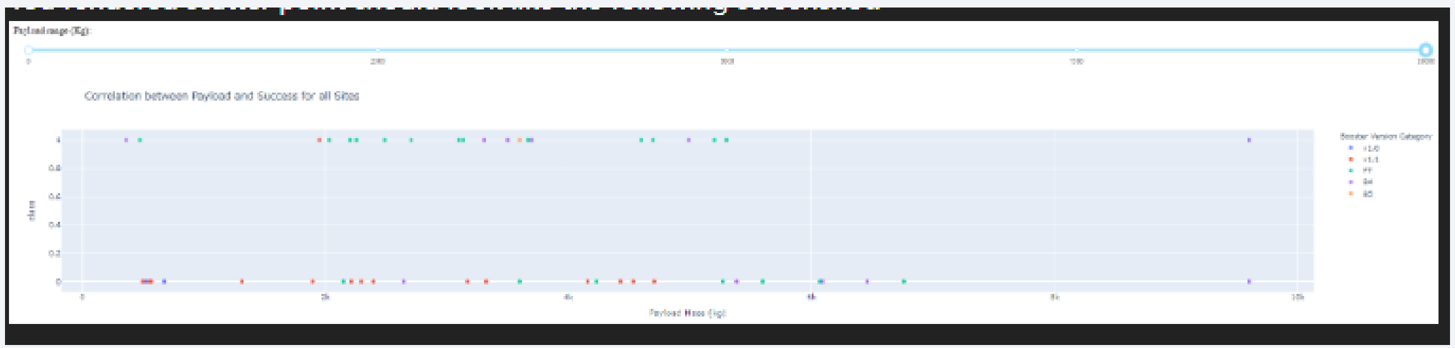
Piechart with the highest launch success ratio



The blue shade shows the highest success launch ratio

<Dashboard Screenshot 3>

- Payload vs Launch outcome plot

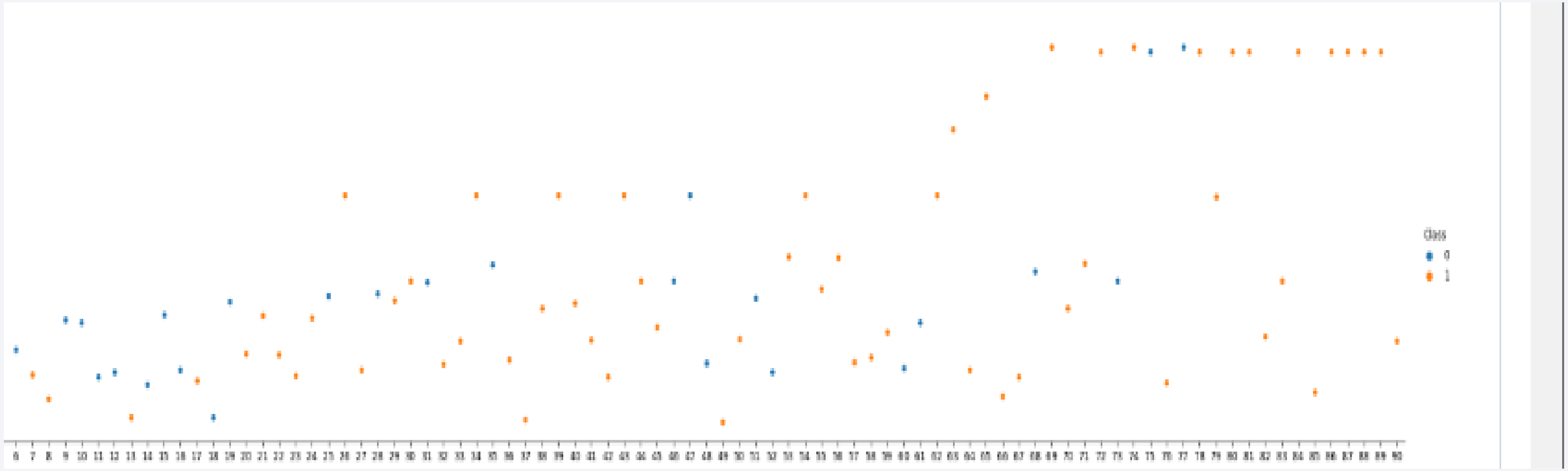


- It can be observed that the size of the payloads affects the success rate of launch sites differently

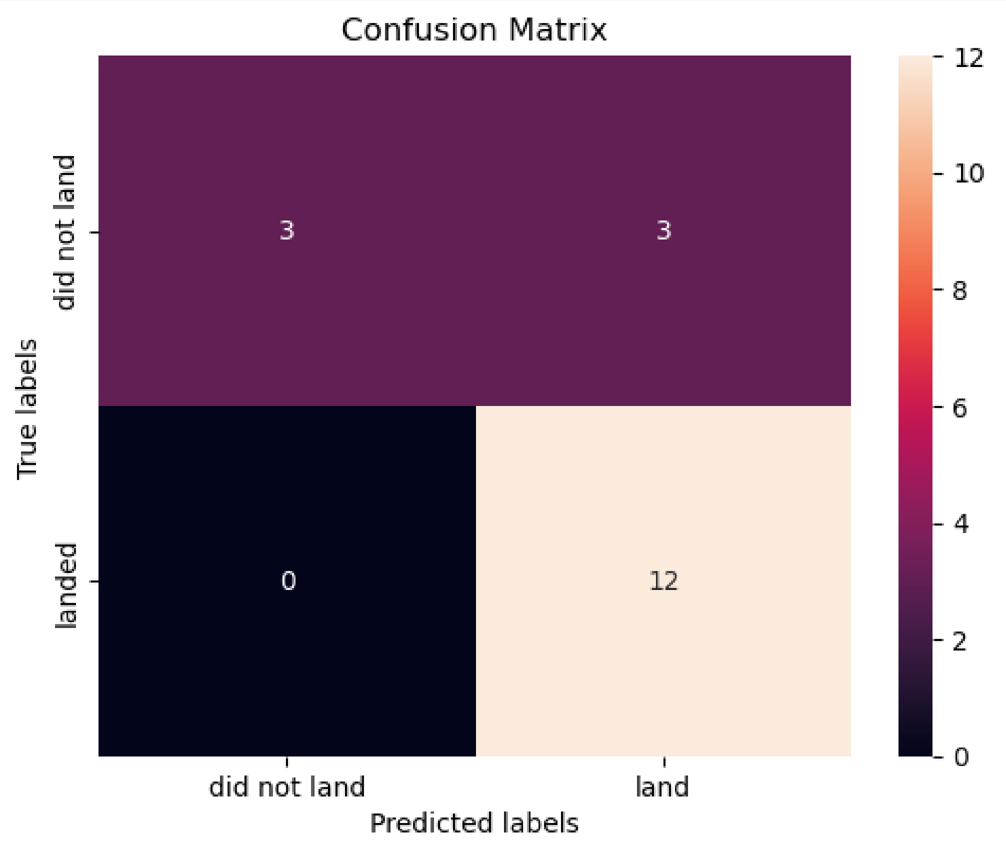
Section 5

Predictive Analysis (Classification)

Classification Accuracy



Confusion Matrix



The confusion matrix shows the prediction of success and failed landing via the labeling.

Conclusions

- Point 1 Success landing
- Point 2 Success landing
- Point 3 Failed
- Point 4
- ...

Appendix

CourseraDataScience_Capstone_Project / Complete the Data Collection API Lab.ipynb

Preview | Code | Blame 1 lines (1 loc) · 486 KB

To make the requested JSON results more consistent, we will use the following static response object for this proj

In [9]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-
```

We should see that the request was successfull with the 200 status response code

In [10]:
```
response.status_code
```

Out[10]: 200

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.jso`

In [11]:
```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
print(data.head())
```

In [22]:
```
#%sql select Date from SPACEXTBL where Landing _Outcome = 'Success (ground pad)' limit 1
%sql select min(Date) from SPACEXTBL where `Landing_Outcome` = 'Success (ground pad)'
```

Thank you!