IBM DATA SCIENCE CAPSTONE PROJECT

Lenard Kristian Tyler

2023

Outline

Executive Summary

Summary of Methodologies

This project follows these steps

- Data Collection
- Data Wrangling
- Exploratory Data Analysis (EDA)
- Interactive Visual Analytics
- Predictive Analysis (Classification)

Summary of Results

This project produced the following outputs and visualizations

- EDA results
- Geospatial analytics
- Interactive dashboard
- Predictive analysis of classification models

Introduction

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 can be used if an alternate company wants to bid against SpaceX for a rocket launch.

In this project, we will predict if the Falcon 9 first stage will land successfully.

Methodology

Data Collection

- Making GET requests to the SpaceX REST API
- Web Scraping SpaceX Wikipedia

Data Wrangling

- Using the .fillna() method to remove NaN values
- Using the .value_counts() method to determine:
  - Number of launches on each site
  - Number and occurrence of each orbit
  - Number and occurrence of mission outcome
- Creating a landing outcome label that shows the following:
  - 0 when the booster did not land successfully
  - 1 when the booster did land successfully

Exploratory Data Analysis

- Using SQL queries to manipulate and evaluate the SpaceX dataset
- Using Pandas and Matplotlib to visualize relationships between variables, and determine patterns

Interactive Visual Analysis

- Geospatial analysis using Folium
- Creating an interactive dashboard using Plotly Dash

Data Modelling and Evaluation

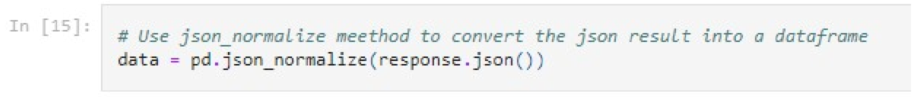
- Using Scikit-Learn to:
  - Pre-process (standardize) the data
  - Split the data into training and testing data using train_test_split
  - Train different classification models
  - Find hyperparameters using GridSearchCV
- Plotting confusion matrices for each classification model
- Assessing the accuracy of each classification model

Data Collection SpaceX Rest API

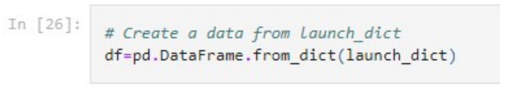
- Request and parse the SpaceX launch data using the GET request

- Use custom logic to collect and store useful data and build a dataset from dictionary

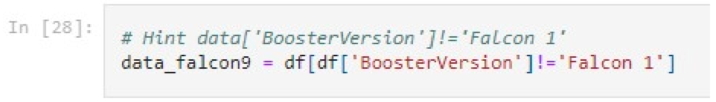
- Clean the dataset to only include Falcon 9 launches, reset the FlightNumber column, replace missing values of PayloadMass with mean PayloadMass value



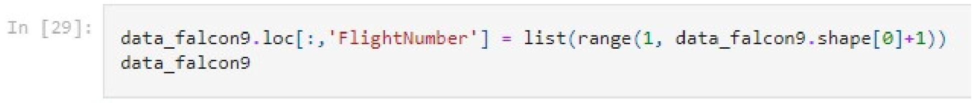
```
In [15]:  # Use json_normalize meethod to convert the json result into a dataframe
          data = pd.json_normalize(response.json())
```



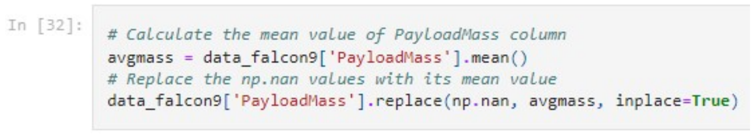
```
In [26]:  # Create a data from launch_dict
          df=pd.DataFrame.from_dict(launch_dict)
```



```
In [28]:  # Hint data['BoosterVersion']!='Falcon 1'
          data_falcon9 = df[df['BoosterVersion']!='Falcon 1']
```



```
In [29]:  data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
          data_falcon9
```

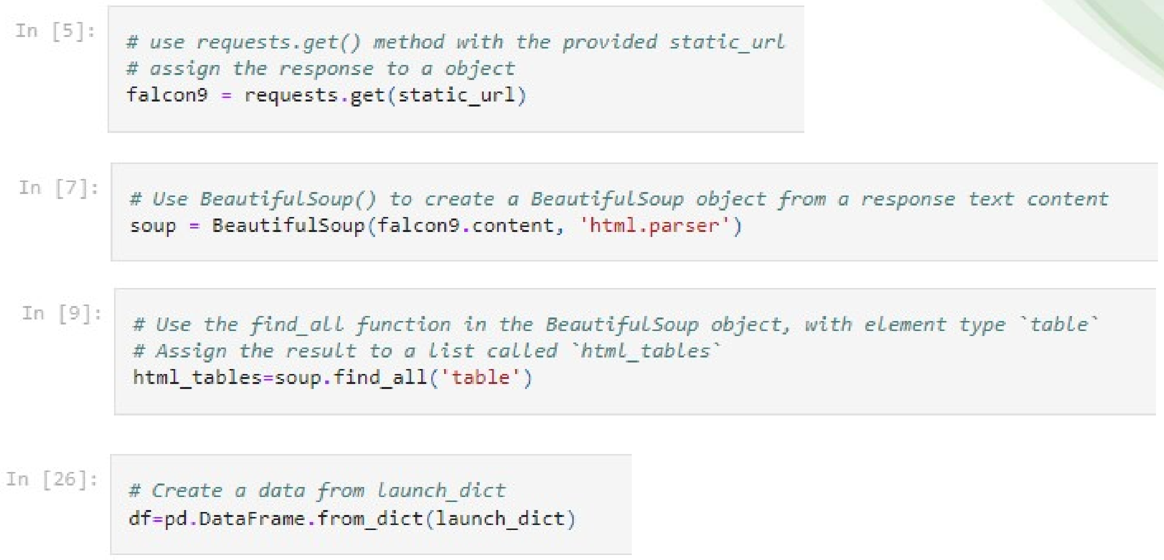


```
In [32]:  # Calculate the mean value of PayloadMass column
          avgmass = data_falcon9['PayloadMass'].mean()
          # Replace the np.nan values with its mean value
          data_falcon9['PayloadMass'].replace(np.nan, avgmass, inplace=True)
```

GitHub Link

Data Collection SpaceX Web Scraping

- Request the Falcon9 Launch Wiki page from its URL
- Extract all column/variable names from the HTML table header
- Create a data frame by parsing the launch HTML tables

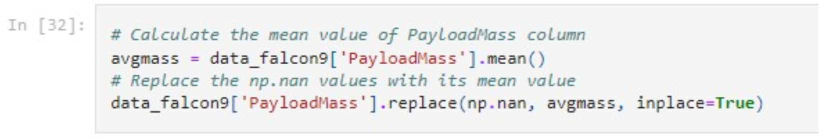


```
In [5]:  # use requests.get() method with the provided static_url
         # assign the response to a object
         falcon9 = requests.get(static_url)

In [7]:  # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
         soup = BeautifulSoup(falcon9.content, 'html.parser')

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')

In [26]: # Create a data from launch_dict
         df=pd.DataFrame.from_dict(launch_dict)
```

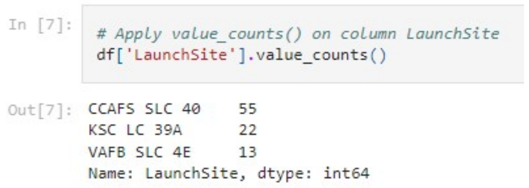
GitHub Link

Data Wrangling

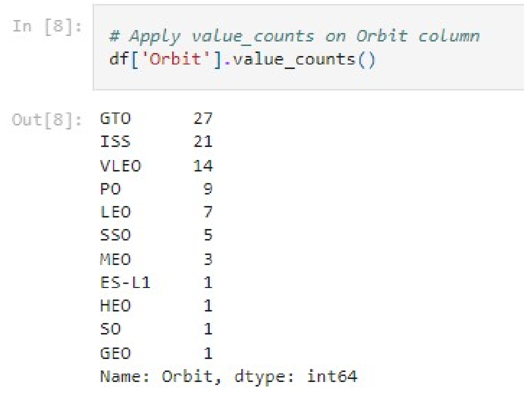
- Replace missing values of PayloadMass with the mean PayloadMass
- Determine the number of launches of each site
- Determine the number and occurrence of orbit type
- Determine the number and occurrence of outcome
- Creating a landing outcome label that shows the following:
  - 0 when the booster did not land successfully
  - 1 when the booster did land successfully



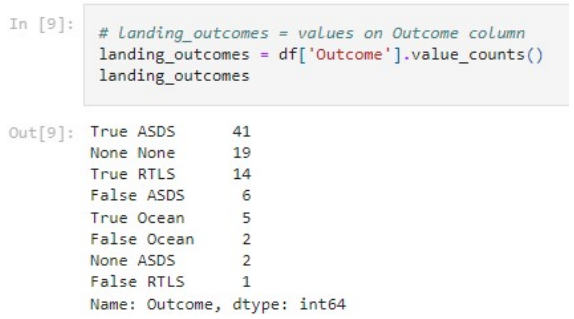
```
In [32]:  # Calculate the mean value of PayloadMass column
          avgmass = data_falcon9['PayloadMass'].mean()
          # Replace the np.nan values with its mean value
          data_falcon9['PayloadMass'].replace(np.nan, avgmass, inplace=True)
```



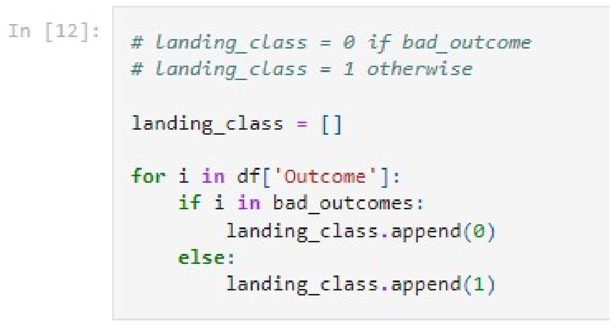
```
In [7]:   # Apply value_counts() on column LaunchSite
          df['LaunchSite'].value_counts()

Out[7]:   CCAFS SLC 40    55
          KSC LC 39A      22
          VAFB SLC 4E     13
          Name: LaunchSite, dtype: int64
```



```
In [8]:   # Apply value_counts on Orbit column
          df['Orbit'].value_counts()

Out[8]:   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
```



```
In [9]:   # landing_outcomes = values on Outcome column
          landing_outcomes = df['Outcome'].value_counts()
          landing_outcomes

Out[9]:   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
```



```
In [12]:  # landing_class = 0 if bad_outcome
          # landing_class = 1 otherwise

          landing_class = []

          for i in df['Outcome']:
              if i in bad_outcomes:
                  landing_class.append(0)
              else:
                  landing_class.append(1)
```

GitHub Link

EDA - Visualization

SCATTER CHARTS

Scatter charts were produced to visualize the relationships between:

- Flight Number and Launch Site
- Payload and Launch Site
- Orbit Type and Flight Number
- Payload and Orbit Type

Scatter charts are useful to observe relationships, or correlations, between two numeric variables.

BAR CHART

A bar chart was produced to visualize the relationship between:

- Success Rate and Orbit Type

Bar charts are used to compare a numerical value to a categorical variable. Horizontal or vertical bar charts can be used, depending on the size of the data.

LINE CHARTS

Line charts were produced to visualize the relationships between:

- Success Rate and Year (i.e. the launch success yearly trend)

Line charts contain numerical values on both axes, and are generally used to show the change of a variable over time.

GitHub Link

EDA - SQL

To further understand the dataset, the following SQL queries were performed:

- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'KSC’
- 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 where the successful landing outcome in drone ship was achieved
- List the names of the boosters which have success in ground pad 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, successful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017
- Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order

GitHub Link

Launch Sites Locations Analysis with Folium

Mark all launch sites

Initialize a Folium Map object

Add a circle and marker object for each launch site

Mark success/failed launches for each site

Cluster the launches according to launch site

Assign a successful launch with a green marker and a failed launch with a red marker

Calculate the distances between a launch site to its proximities

The approximate distances can be calculated by using the latitude and longitude values

Create a Folium marker object for the point of interest

Folium polyline was added to show the distance between the two points

GitHub Link

Interactive Dashboard – Plotly Dash

Build a Plotly Dash application for users to perform interactive visual analytics on SpaceX launch data in real-time. This dashboard application contains input components such as a dropdown list and a range slider to interact with a pie chart and a scatter point chart.

- The pie chart shows the total successful launches per site
  - This makes it clear to see which sites are the most successful
  - This chart can be filtered using the dropdown list to see more details for a specific site.

- Scatter chart shows the correlation between launch outcome and payload mass
  - This can be filtered using a range slider and dropdown list to further compare booster version and payload mass.

GitHub Link

Predictive Analysis - Classification

Create a machine learning pipeline to predict if the first stage will land

Model Development

- Prepare the dataset for model development
  - Load data
  - Standardize and pre-process
  - Split data into training and testing data
- Models
  - Create a GridSeachCV object and dictionary of parameters
  - Fit the object to the parameters
  - Use the training data to train the model

Model Evaluation

- Chosen Model
  - Check tuned hyperparameters
  - Check accuracy
- Plot and examine the confusion matrix

Determine Best Classification Model

- Compare accuracy of models

GitHub Link

Results

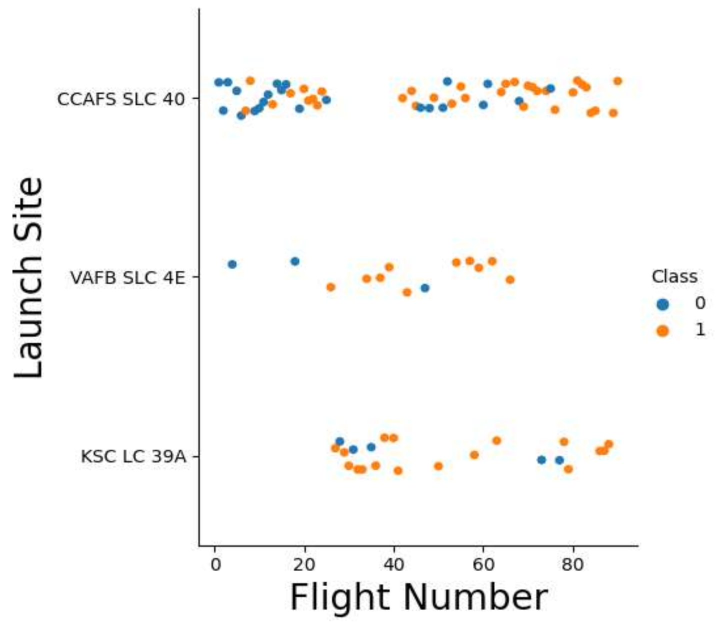
Exploratory Data Analysis

Interactive Analytics

Predictive Analysis

EDA - Visualization

Flight Number vs Launch Site



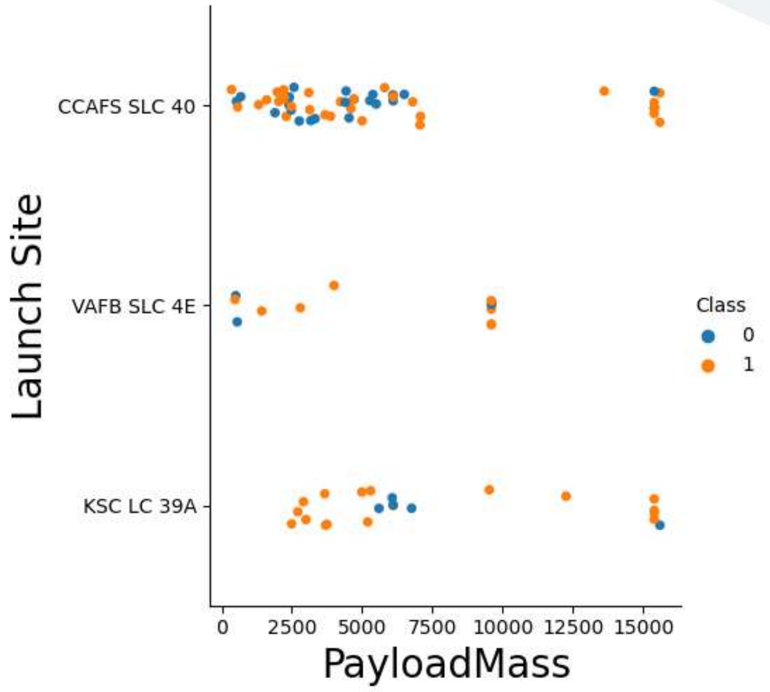
The scatter plot of Launch Site vs. Flight Number shows:

- As the number of flights increases, the rate of success at a launch site increases.
- Most early flights were launched from CCAFS SLC 40 and were generally unsuccessful. (Class = 0)
- The flights from VAFB SLC 4E also show this trend, that earlier flights were less successful.
- No early flights were launched from KSC LC 39A, so the launches from this site are more successful.
- Later flights show there are significantly more successful landings (Class = 1).

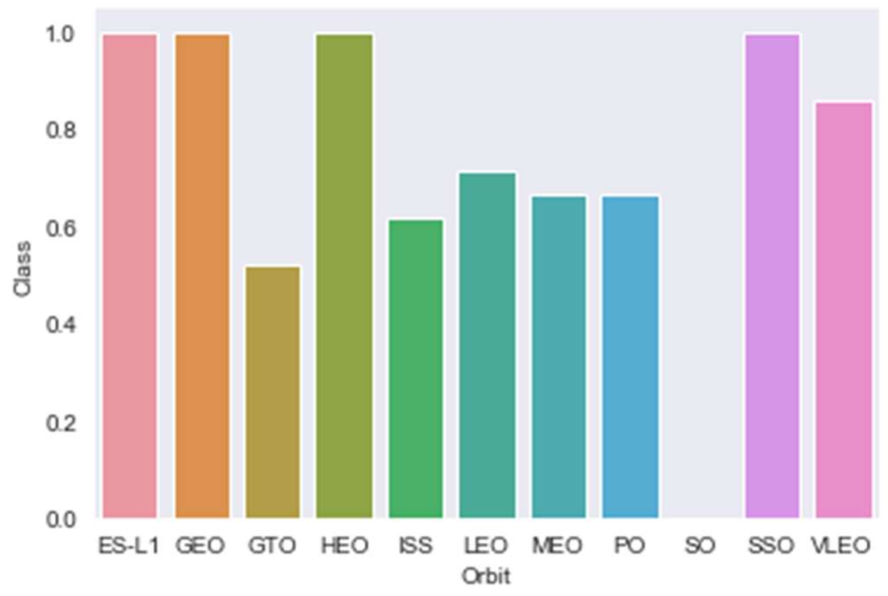
Payload vs Launch Site

The scatter plot of Launch Site vs. Payload Mass shows:

- Above a mass of around 7000 kg, there are very few unsuccessful landings, but there is also less data for heavier launches.
- There is no clear correlation between payload mass and success rate for a given launch site.
- All sites launched a variety of payload masses, with most of the launches from CCAFS SLC 40 being comparatively lighter payloads (with some outliers).



Success Rate vs Orbit Type



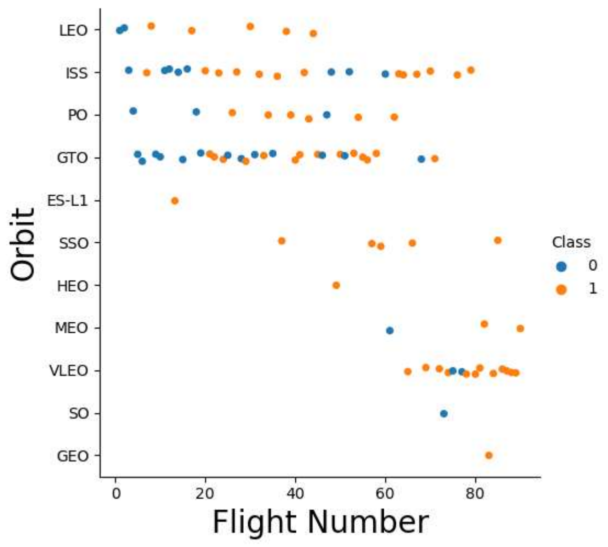
The bar chart of Success Rate vs. Orbit Type shows that the higher altitude orbits have the highest (100%) success rate:

- ES-L1 (Earth-Sun First Lagrangian Point)
- GEO (Geostationary Orbit)
- HEO (High Elliptical Orbit)
- SSO (Sun-synchronous Orbit)

The orbit with the lowest (0%) success rate is:

- SO (Sub Orbital)

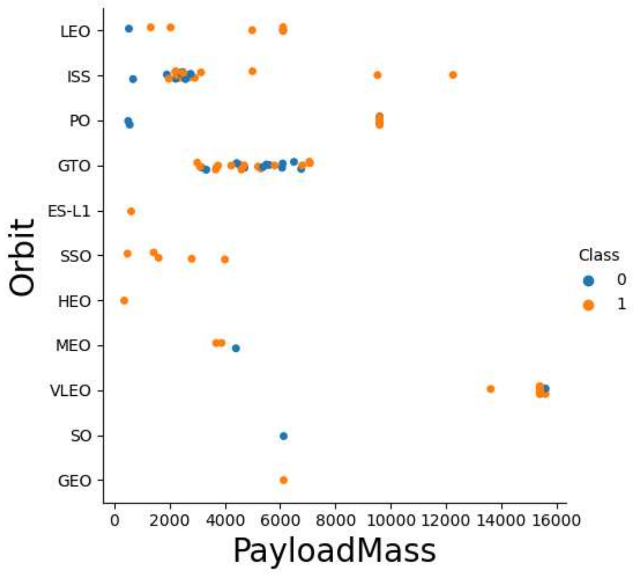
Flight Number vs Orbit Type



This scatter plot of Orbit Type vs. Flight number shows:

- The 100% success rate of GEO, HEO, and ES-L1 orbits can be explained by only having 1 flight into their respective orbits
- The 100% success rate in SSO is more impressive, with 5 successful flights
- There is little relationship between Flight Number and Success Rate for GTO
- Generally, as Flight Number increases, the success rate increases

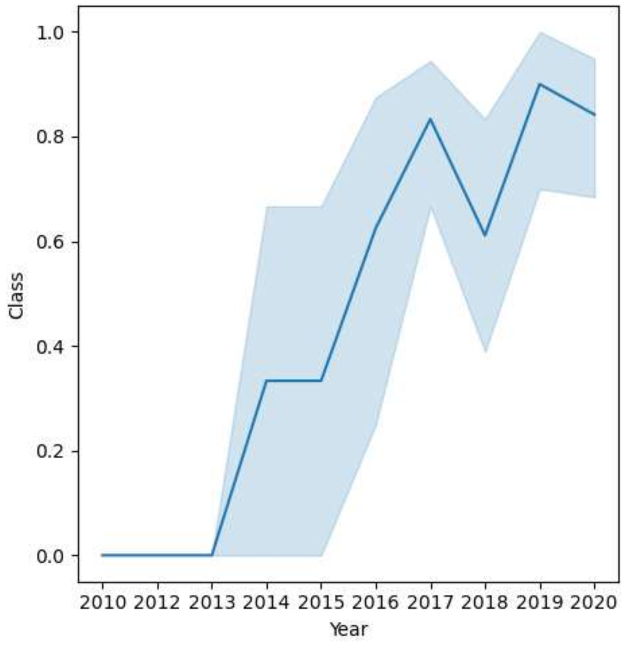
Payload vs Orbit Type



This scatter plot of Orbit Type vs. Payload Mass shows:

- The following orbit types have more success with heavy payloads:
  - PO
  - ISS
  - VLEO
- For the successful, high-altitude orbits, the payload was lighter
- VLEO (Very Low Earth Orbit) launches are associated with heavier payloads.

Launch Success Yearly Trend



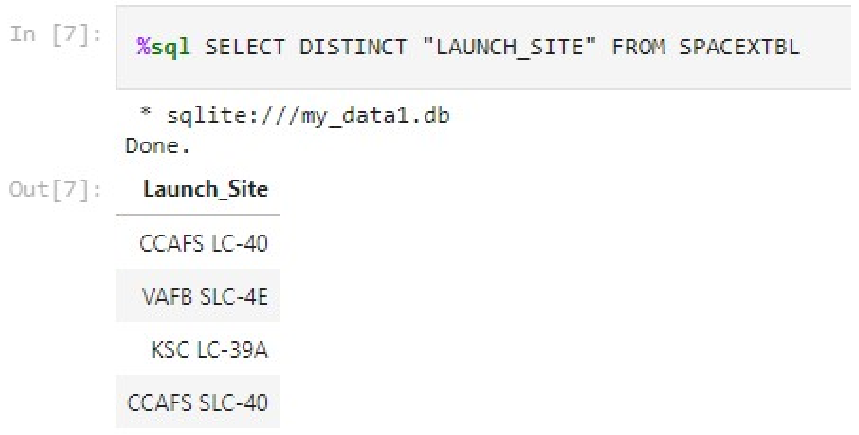
The line chart of yearly average success rate shows:

- Between 2010 and 2013, all landings were unsuccessful (as the success rate is 0).
- After 2013, the success rate generally increased, despite small dips in 2018 and 2020.
- After 2016, there was always a greater than 50% chance of success.

EDA - SQL

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

DISTINCT returns the unique values of the “LAUNCH_SITE” column of the SPACEXTBL table.

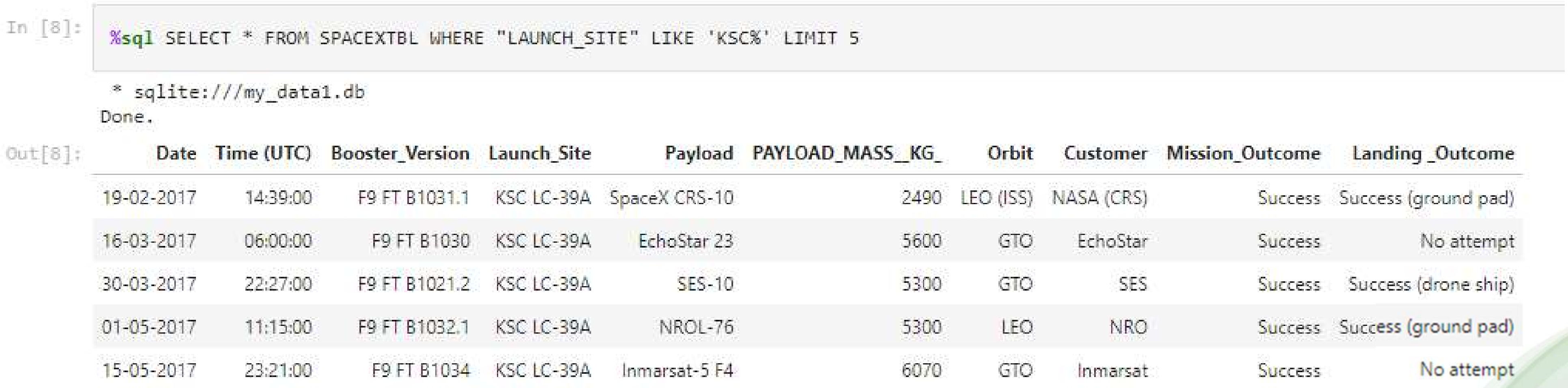


```
In [7]: %sql SELECT DISTINCT "LAUNCH_SITE" FROM SPACEXTBL

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

Out[7]:

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

Display 5 records where launch sites begin with the string 'KSC’



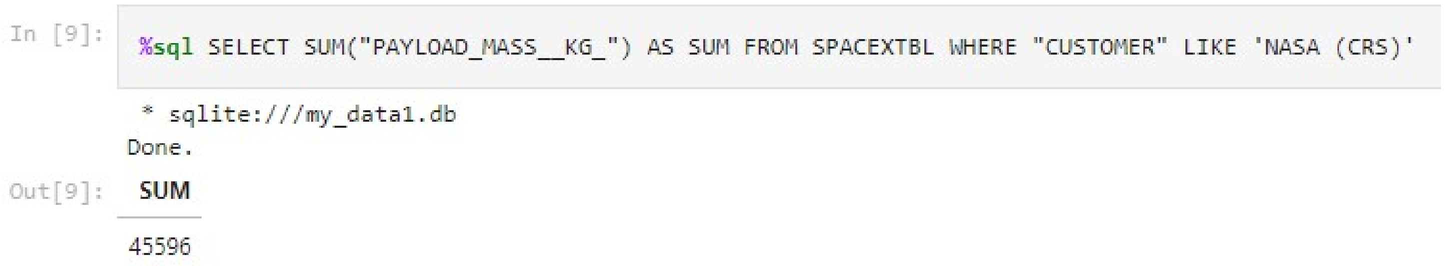
```
In [8]:  %sql SELECT * FROM SPACEXTBL WHERE "LAUNCH_SITE" LIKE 'KSC%' LIMIT 5

 * sqlite:///my_data1.db
Done.
Out[8]:
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
19-02-2017	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
16-03-2017	06:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	5600	GTO	EchoStar	Success	No attempt
30-03-2017	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
01-05-2017	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300	LEO	NRO	Success	Success (ground pad)
15-05-2017	23:21:00	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070	GTO	Inmarsat	Success	No attempt

LIMIT 5 only retrieves 5 records, and LIKE is used with wild card ‘KSC%’ to retrieve string values starting with ‘KSC’

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

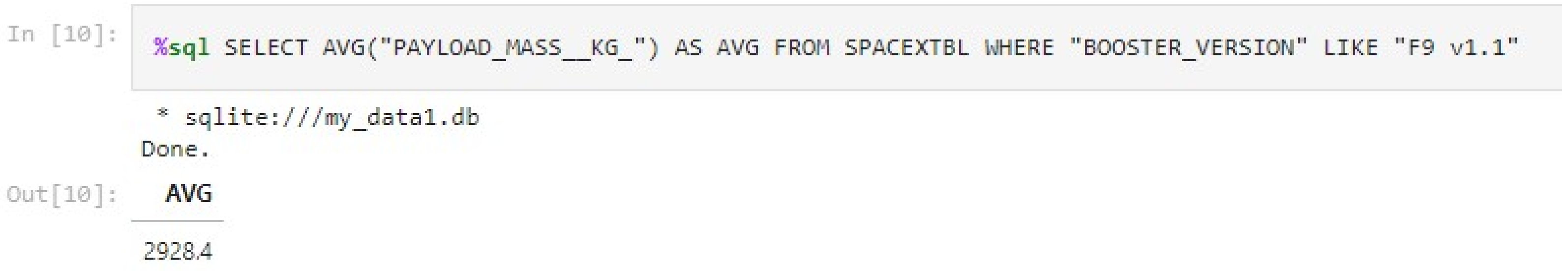
The SUM keyword is used to calculate the total of the LAUNCH column,
and the SUM keyword (and the associated condition) filters the results to only boosters from NASA (CRS)



```
In [9]:  %sql SELECT SUM("PAYLOAD_MASS__KG_") AS SUM FROM SPACEXTBL WHERE "CUSTOMER" LIKE 'NASA (CRS)'

          * sqlite:///my_data1.db
         Done.
Out[9]:   SUM
         45596
```

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

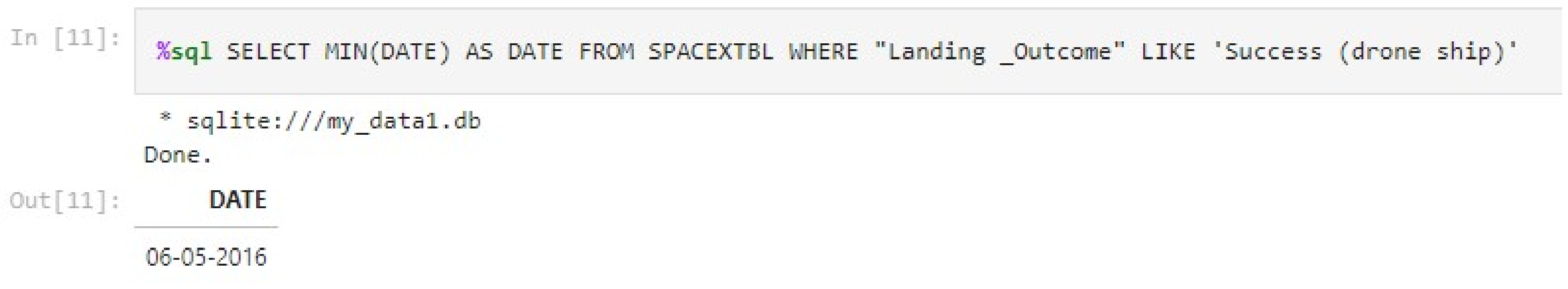
The AVG keyword is used to calculate the average of the PAYLOAD_MASS__KG_ column, and the WHERE keyword (and the associated condition) filters the results to only the F9 v1.1 booster version



```
In [10]: %sql SELECT AVG("PAYLOAD_MASS__KG_") AS AVG FROM SPACEXTBL WHERE "BOOSTER_VERSION" LIKE "F9 v1.1"

 * sqlite:///my_data1.db
Done.

Out[10]:  AVG
        2928.4
```

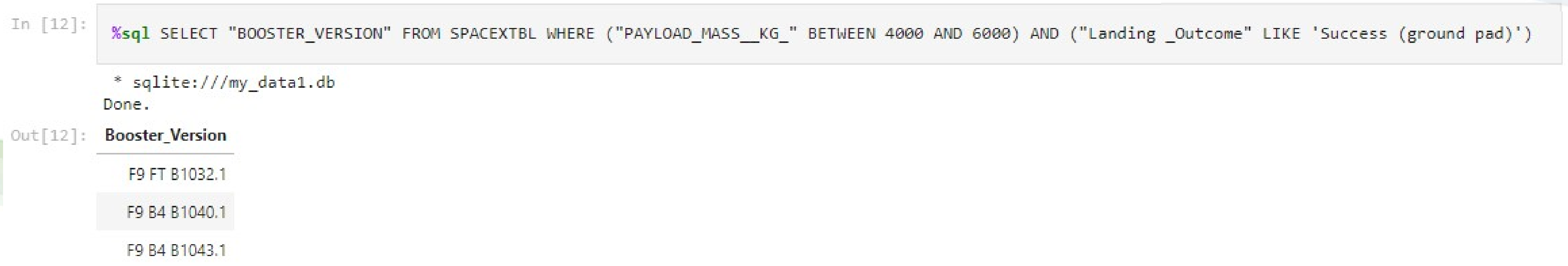
List the date where the successful landing outcome in drone ship was achieved.

The MIN keyword is used to calculate the minimum of the DATE column, i.e., the first date, and the WHERE keyword (and the associated condition) filters the results to only the successful drone ship landings



```
In [11]: %sql SELECT MIN(DATE) AS DATE FROM SPACEXTBL WHERE "Landing _Outcome" LIKE 'Success (drone ship)'

 * sqlite:///my_data1.db
Done.
Out[11]:       DATE
         06-05-2016
```

List the names of the boosters which have success in ground pad and have payload mass between 4000 and 6000



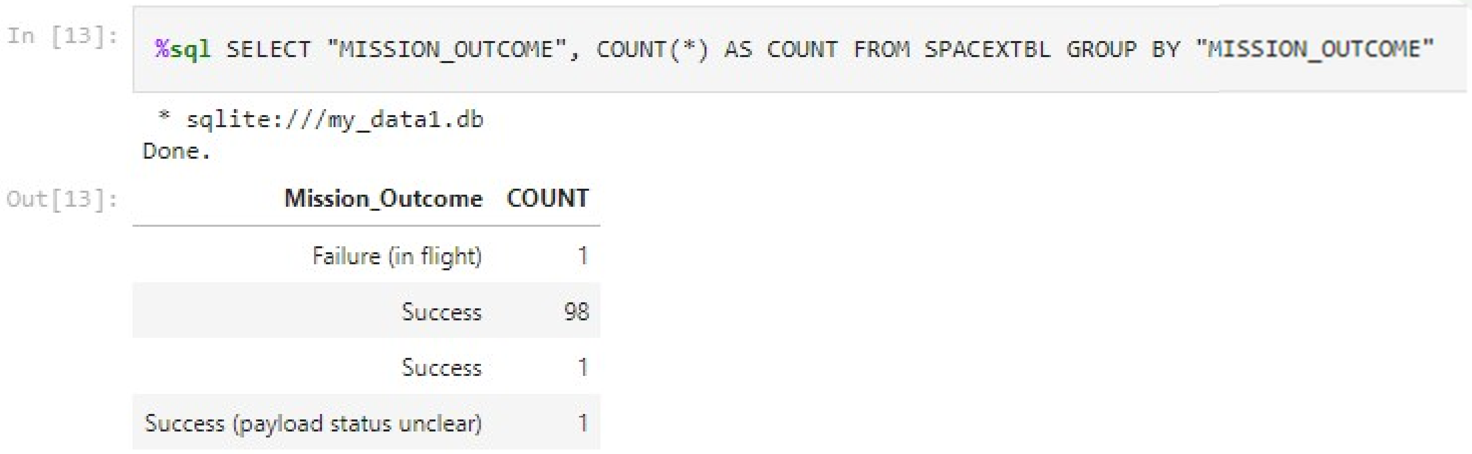
```
In [12]: %sql SELECT "BOOSTER_VERSION" FROM SPACEXTBL WHERE ("PAYLOAD_MASS__KG_" BETWEEN 4000 AND 6000) AND ("Landing _Outcome" LIKE 'Success (ground pad)')

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

Out[12]:

Booster_Version
F9 FT B1032.1
F9 B4 B1040.1
F9 B4 B1043.1

The WHERE keyword is used to filter the results to include only those that satisfy both conditions in the brackets (as the AND keyword is also used). The BETWEEN keyword allows for 4000 < x < 6000 values to be selected

List the total number of successful and failure mission outcomes

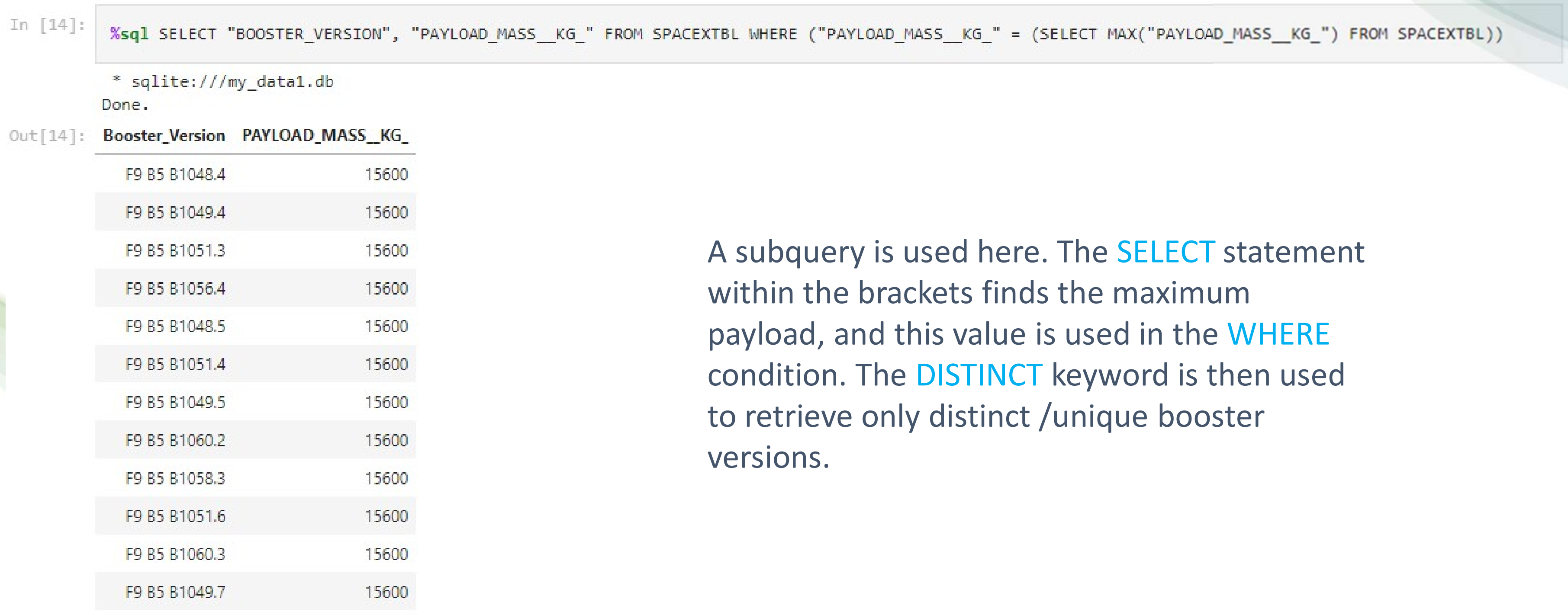


```
In [13]: %sql SELECT "MISSION_OUTCOME", COUNT(*) AS COUNT FROM SPACEXTBL GROUP BY "MISSION_OUTCOME"

 * sqlite:///my_data1.db
Done.
Out[13]:
```

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

The COUNT keyword is used to calculate the total number of mission outcomes, and the GROUPBY keyword is also used to group these results by the type of mission outcome.

List the names of the booster_versions which have carried the maximum payload mass



```
In [14]: %sql SELECT "BOOSTER_VERSION", "PAYLOAD_MASS__KG_" FROM SPACEXTBL WHERE ("PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTBL))

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

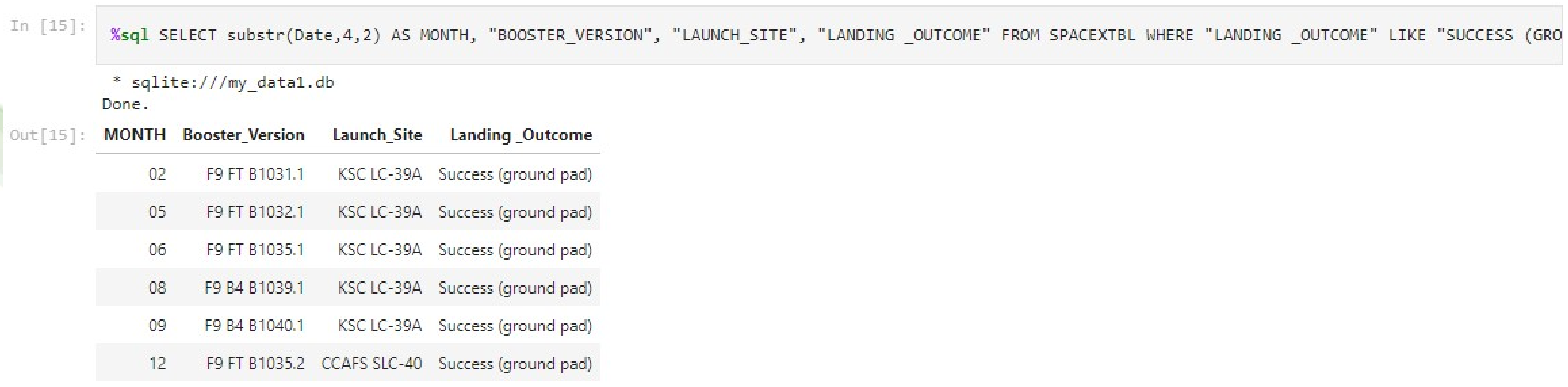
Out[14]:

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

A subquery is used here. The SELECT statement within the brackets finds the maximum payload, and this value is used in the WHERE condition. The DISTINCT keyword is then used to retrieve only distinct /unique booster versions.

List the records which will display the month names, successful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017

The WHERE keyword is used to filter the results for only successful landing outcomes, AND only for the year of 2017.



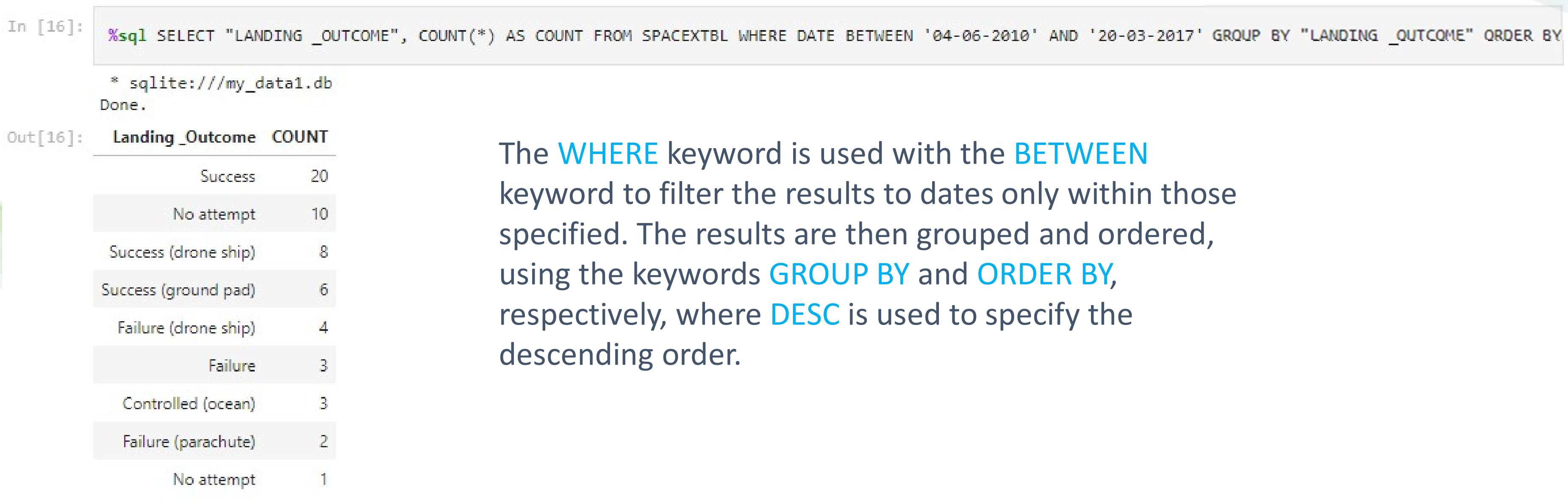
```
In [15]:  %sql SELECT substr(Date,4,2) AS MONTH, "BOOSTER_VERSION", "LAUNCH_SITE", "LANDING _OUTCOME" FROM SPACEXTBL WHERE "LANDING _OUTCOME" LIKE "SUCCESS (GRO
```

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

Out[15]:

MONTH	Booster_Version	Launch_Site	Landing _Outcome
02	F9 FT B1031.1	KSC LC-39A	Success (ground pad)
05	F9 FT B1032.1	KSC LC-39A	Success (ground pad)
06	F9 FT B1035.1	KSC LC-39A	Success (ground pad)
08	F9 B4 B1039.1	KSC LC-39A	Success (ground pad)
09	F9 B4 B1040.1	KSC LC-39A	Success (ground pad)
12	F9 FT B1035.2	CCAFS SLC-40	Success (ground pad)

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



```
In [16]: %sql SELECT "LANDING _OUTCOME", COUNT(*) AS COUNT FROM SPACEXTBL WHERE DATE BETWEEN '04-06-2010' AND '20-03-2017' GROUP BY "LANDING _OUTCOME" ORDER BY

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

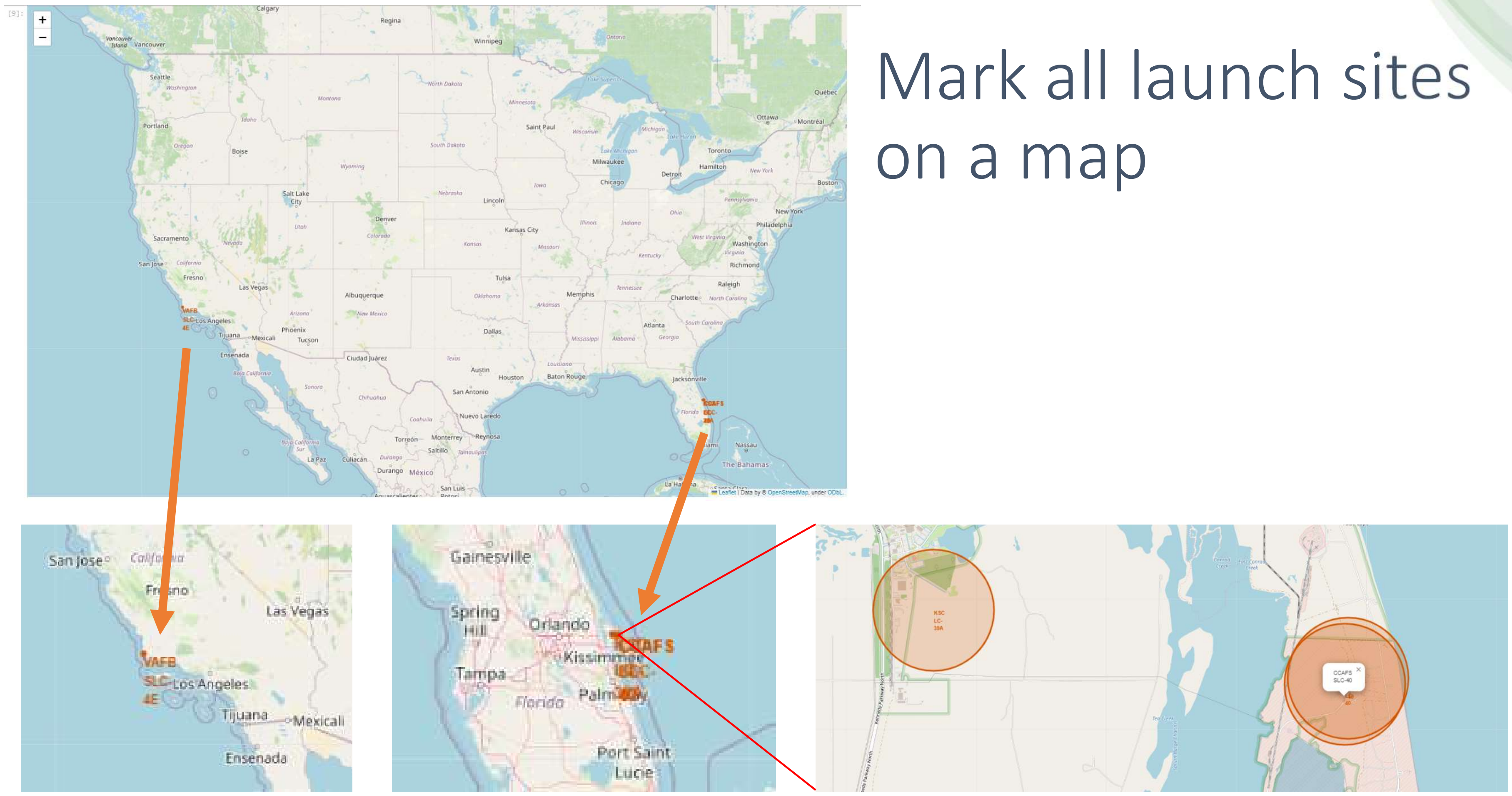
Out[16]:

Landing _Outcome	COUNT
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

The WHERE keyword is used with the BETWEEN keyword to filter the results to dates only within those specified. The results are then grouped and ordered, using the keywords GROUP BY and ORDER BY, respectively, where DESC is used to specify the descending order.

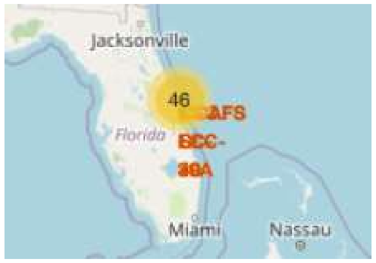
Launch Sites Locations Analysis with Folium

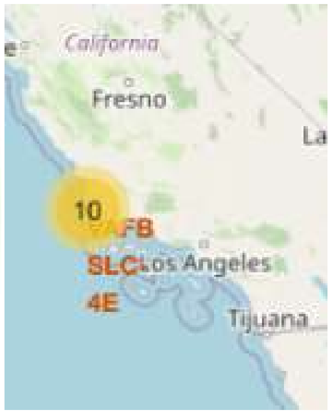
Mark all launch sites on a map

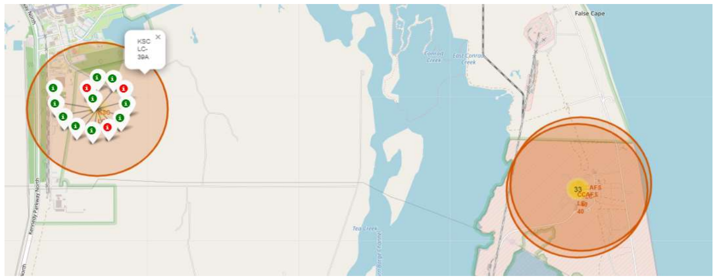


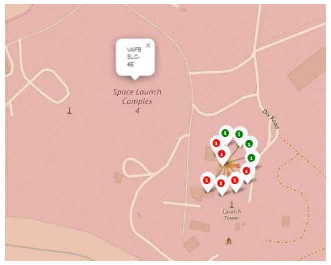
Mark the success/failed launches for each site on the map

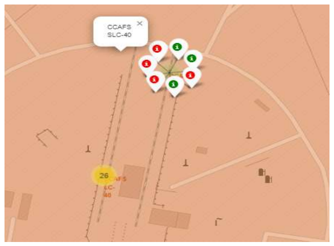
Launches have been grouped into clusters and annotated with green icons for successful launches, and red icons for failed launches.

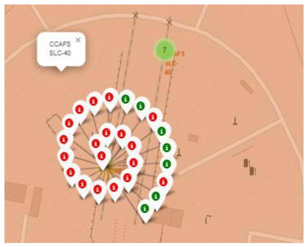








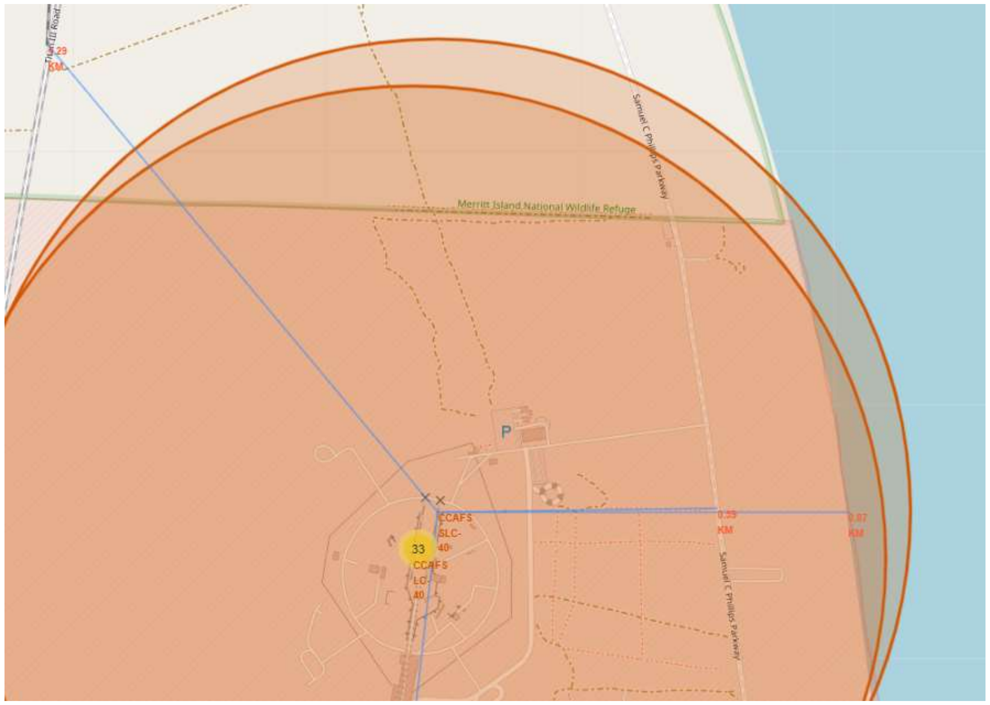




Calculate the distances between a launch site to its proximities

Using the CCAFS SLC-40 launch site, we can understand more about the placement of launch sites.

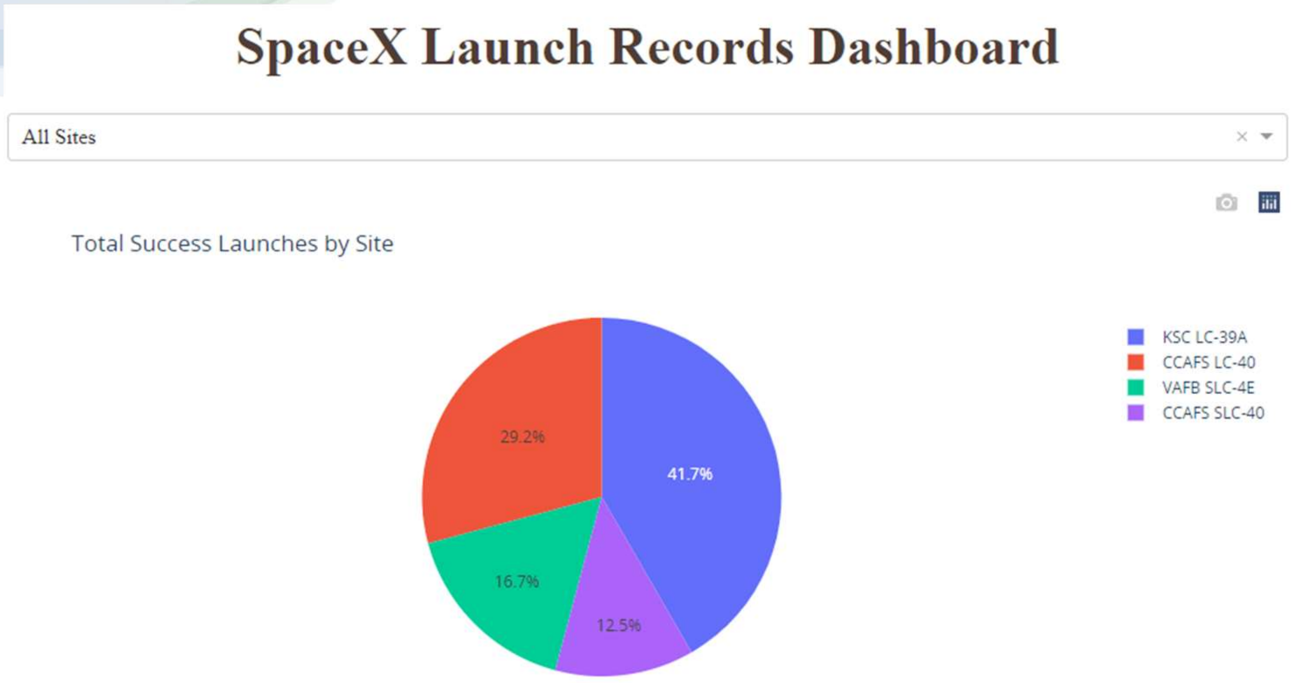
- The nearest city is 51.74 km away.
- The nearest railway is only 1.29 km away.
- The nearest highway is only 0.59km away.
- The coastline is only 0.87 km due East.


1.29 KM
Merritt Island National Wildlife Refuge
Samuel C Phillips Parkway
CCAFS SLC-40
33
CCAFS LC-40
0.59 KM
0.87 KM

CR 507
North Apollo Boulevard
Nieman Avenue
Melbourne
Orlando Melbourne International Airport
51.74 KM

Interactive Dashboard
Plotly Dash

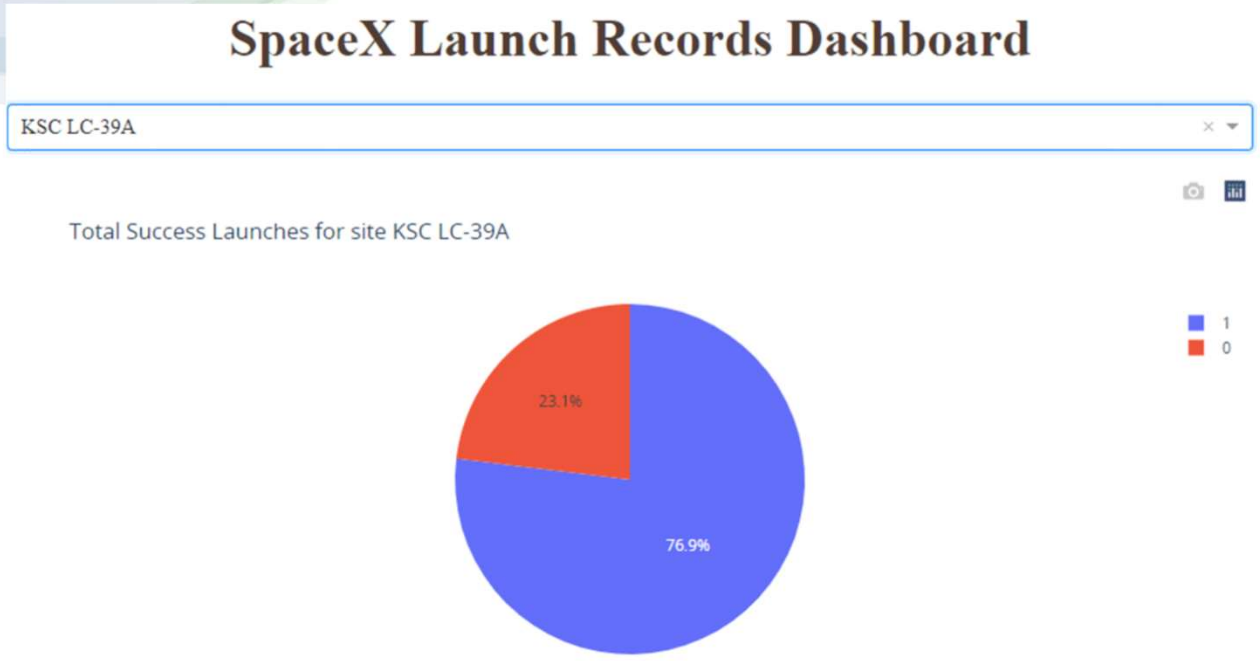
Total Successful Launches by Site


SpaceX Launch Records Dashboard

All Sites

The launch site KSC LC-39 A had the most successful launches, with 41.7% of the total successful launches.

Launch Site with Highest Success

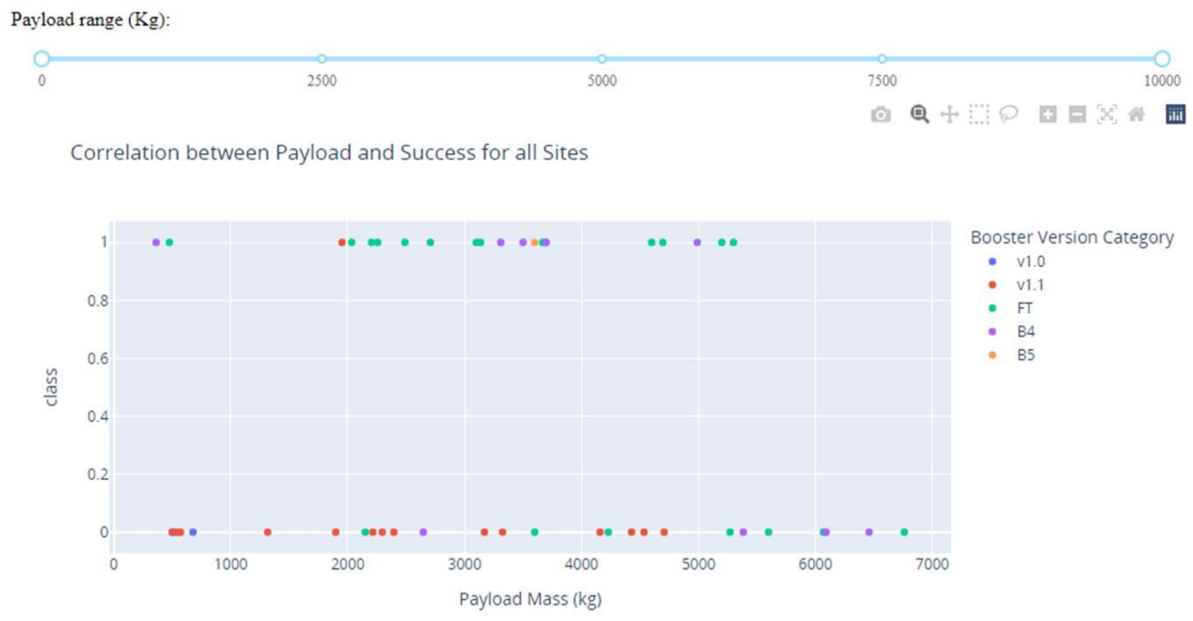

SpaceX Launch Records Dashboard

KSC LC-39A

Total Success Launches for site KSC LC-39A

The launch site KSC LC-39 A also had the highest rate of successful launches, with a 76.9% success rate.

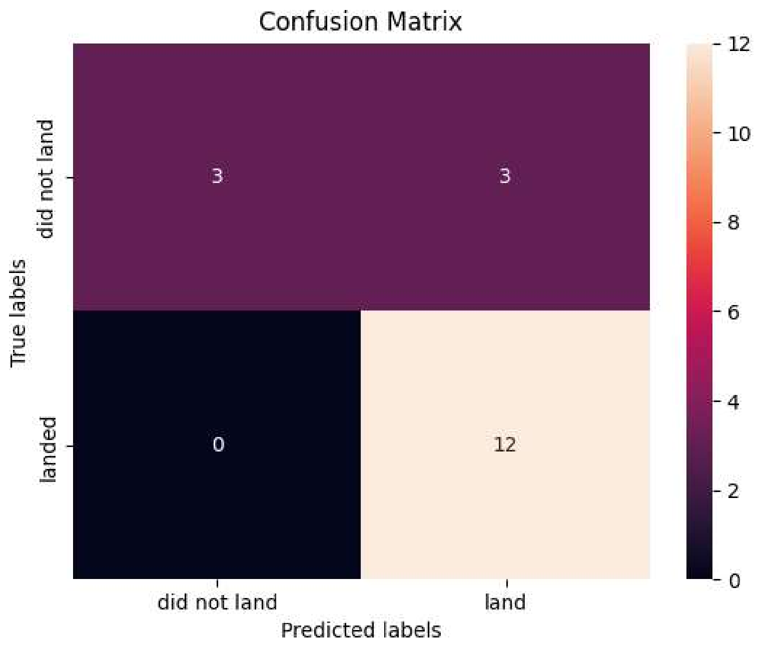
Outcome vs Payload Across All Sites



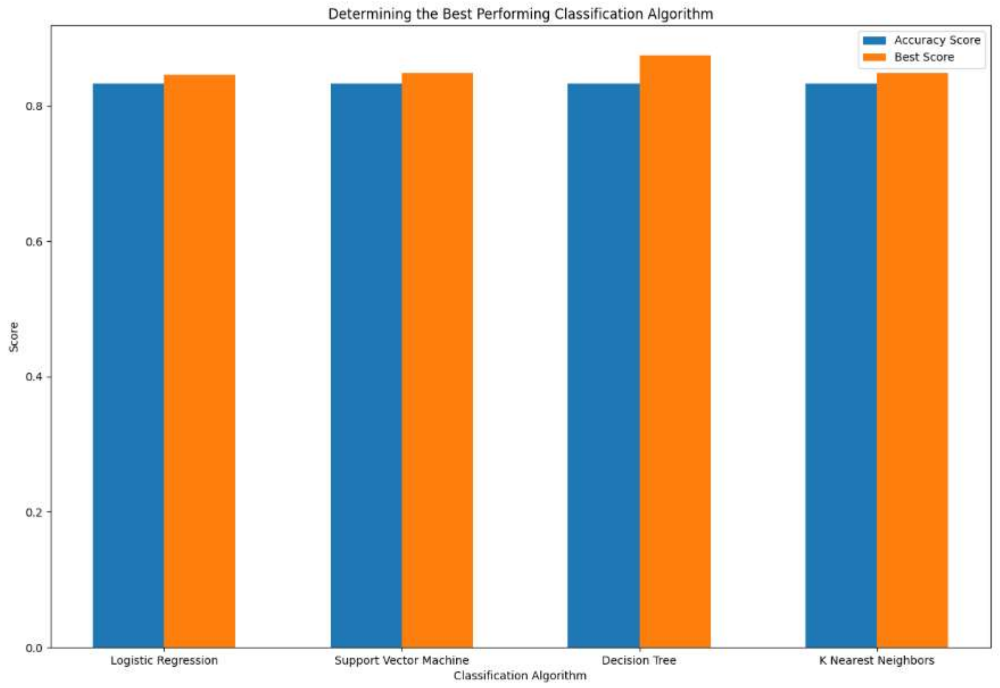
Plotly dashboard has a Payload range selector. However, this is set from 0-10000 instead of the max Payload of 15600. Class indicates 1 for successful landing and 0 for failure. Scatter plot also accounts for booster version category in color

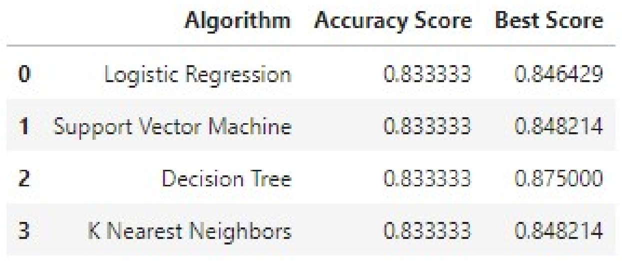
Predictive Analysis Classification

Confusion Matrix



- This confusion matrix shows the predicted labels from the model vs the actual labels, which shows 3 out of 18 total results classified incorrectly (a false positive, shown in the top-right corner).
- The other 15 results are correctly classified (3 did not land, 12 did land).





	Algorithm	Accuracy Score	Best Score
0	Logistic Regression	0.833333	0.846429
1	Support Vector Machine	0.833333	0.848214
2	Decision Tree	0.833333	0.875000
3	K Nearest Neighbors	0.833333	0.848214

Classifier Accuracy

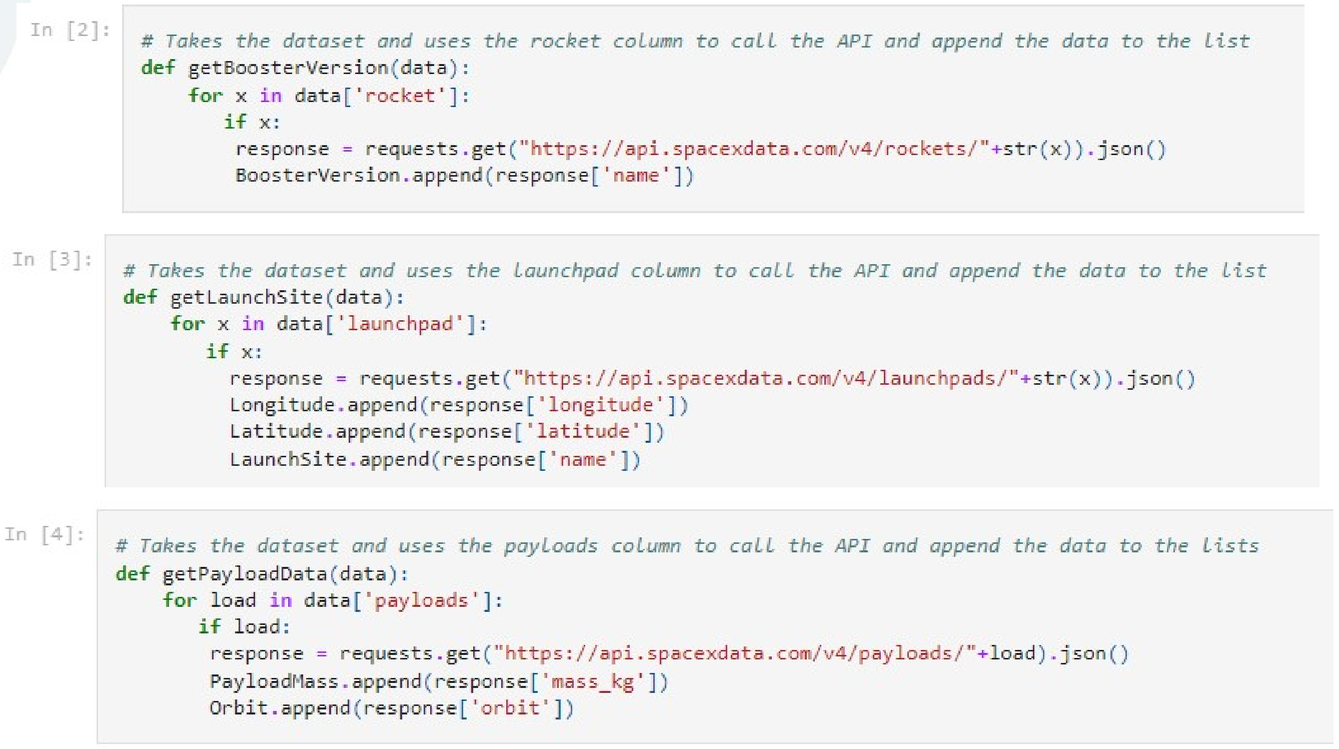
Plotting the Accuracy Score and Best Score for each classification algorithm produces the following result:

- The Decision Tree model has the highest classification accuracy
- The Accuracy Score is 83.33%
- The Best Score is 87.50%

Conclusion

- As the number of flights increases, the rate of success at a launch site increases, with most early flights being unsuccessful. (I.e., with more experience, the success rate increases.)
    - Between 2010 and 2013, all landings were unsuccessful (as the success rate is 0)
    - After 2016, there was always a greater than 50% chance of success
- Orbit types ES-L1, GEO, HEO, and SSO, have the highest (100%) success rate.
    - The 100% success rate of GEO, HEO, and ES-L1 orbits can be explained by only having 1 flight into the respective orbits
    - The 100% success rate in SSO is more impressive, with 5 successful flights
    - The orbit types PO, ISS, and VLEO, have more success with heavy payloads
- The launch site KSC LC-39 A had the most successful launches, with 41.7% of the total successful launches, and the highest rate of successful launches, with 76.9% success rate.
- The success for heavier payloads is lower than that for lighter payloads.
- The best performing classification model is the Decision Tree

Appendix

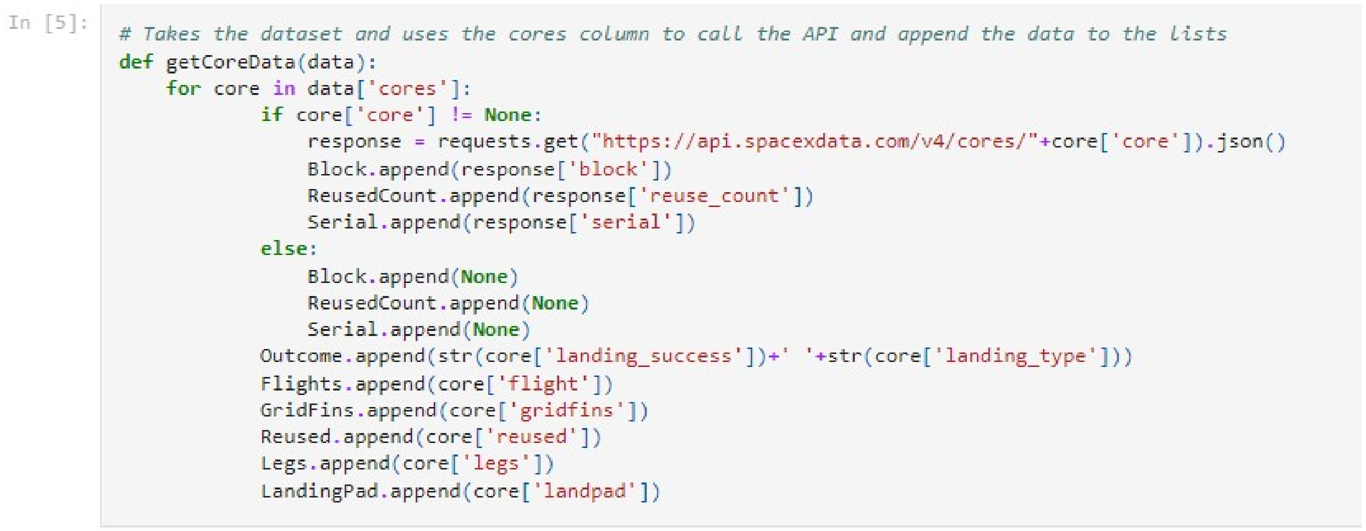


```
In [2]:
# Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
       if x:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])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

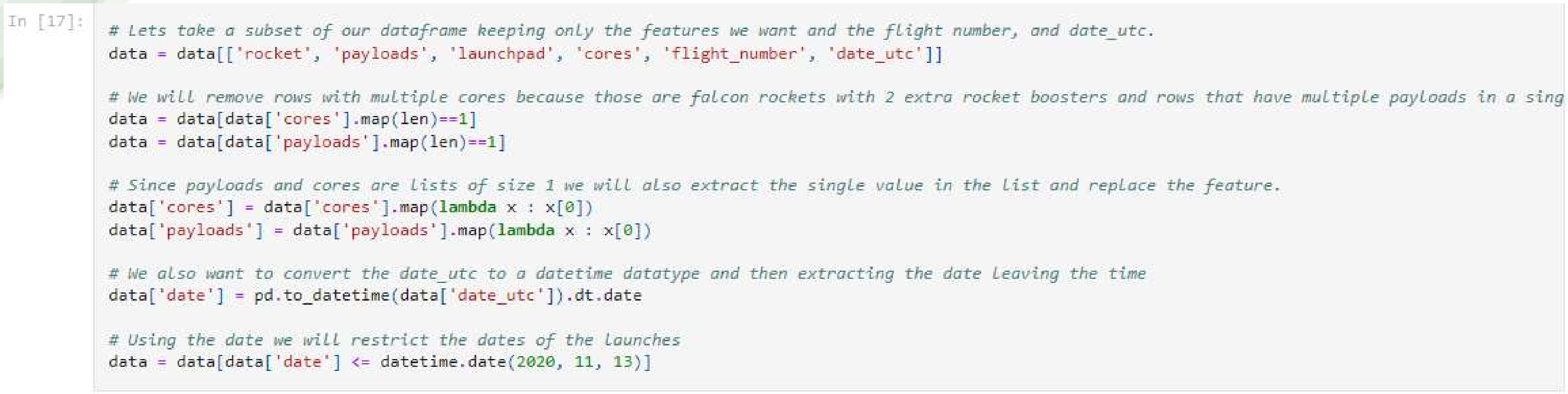
```
In [3]:
# Takes the dataset and uses the launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['launchpad']:
       if x:
         response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
         Longitude.append(response['longitude'])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

```
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']:
       if load:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

Data Collection SpaceX Rest API

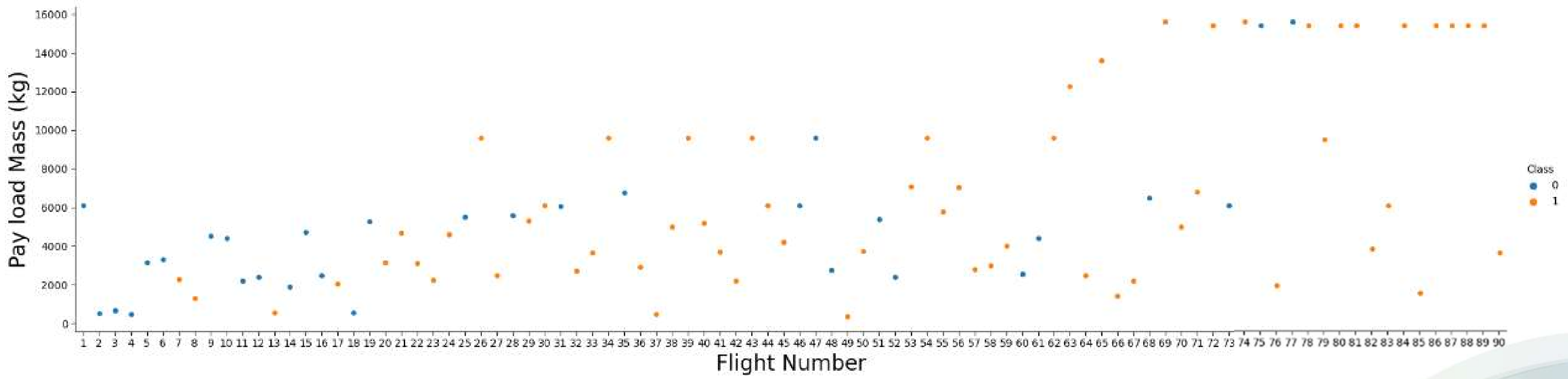

```
In [5]:
# Takes the dataset and uses the cores column to call the API and append the data to the lists
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
                Block.append(response['block'])
                ReusedCount.append(response['reuse_count'])
                Serial.append(response['serial'])
            else:
                Block.append(None)
                ReusedCount.append(None)
                Serial.append(None)
            Outcome.append(str(core['landing_success'])+' '+str(core['landing_type']))
            Flights.append(core['flight'])
            GridFins.append(core['gridfins'])
            Reused.append(core['reused'])
            Legs.append(core['legs'])
            LandingPad.append(core['landpad'])
```

Data Collection SpaceX Rest API



```
In [17]:
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that have multiple payloads in a sing
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

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

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

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

Data Collection SpaceX Rest API



EDA - Visualization