IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Viktor Gakis

28/11/2022

Outline

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

Executive Summary

- **Summary of methodologies**
  - **Data Collection via API, Web Scraping**
  - **Data Wrangling**
  - **Exploratory Data Analysis**
    - **Visualization**
    - **SQL**
  - **Interactive Map with Folium**
  - **Dashboards with Dash(Plotly)**
  - **Predictive analysis(Classification)**
- **Summary of all results**
  - **Exploratory Data Analysis results**
  - **Interactive maps and dashboard**
  - **Predictive Analysis results**

Introduction

- Project background and context
  - Having entered the era of commercial space travel, Space X advertises Falcon 9 rocket launches to space at the cost of 62 million dollars. The cost for other providers reaches165 million dollars each. The difference in price comes from the ability to reuse the first stage in Space X.
  - If it can be determined whether the first stage lands, the cost of the launch can be estimated beforehand. This information can be used from other companies in case they want to bid against space X.
  - The goal of this project is to predict whether the first stage will land successfully.
- Problems you want to find answers
  - Which factors determine the successful landing of the rocket?
  - What are the effects of the features on the success/failure of landing?
  - What are the optimal parameters in order to achieve the best landing success rate?

Section 1

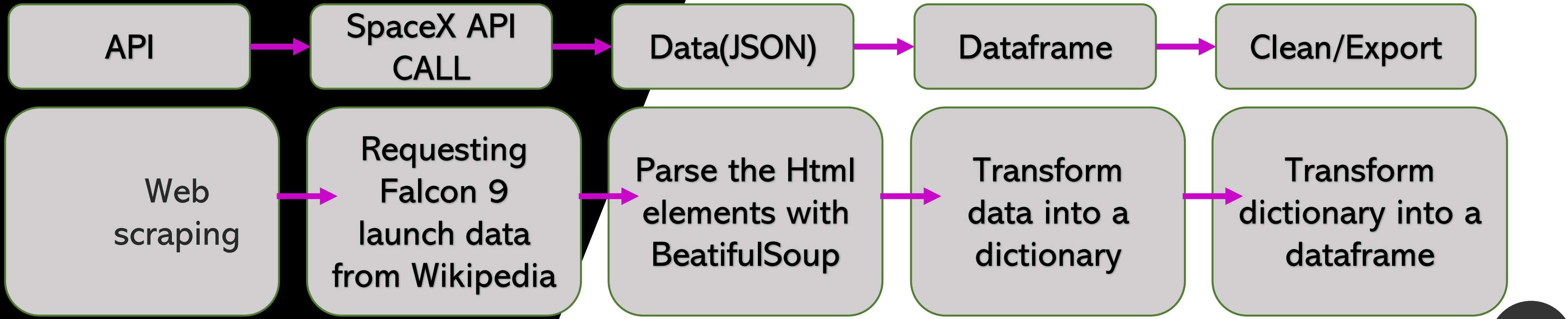
Methodology

Methodology

- Executive Summary
- Data collection methodology:
  - Directly querying the SpaceX REST API
  - Web scrapping data from Wikipedia
- Perform data wrangling
- 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

- There were two methods collecting data:
  - API:
  - Web scraping



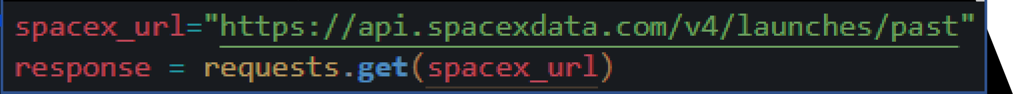
API → SpaceX API CALL → Data(JSON) → Dataframe → Clean/Export

Web scraping → Requesting Falcon 9 launch data from Wikipedia → Parse the Html elements with BeatifulSoup → Transform data into a dictionary → Transform dictionary into a dataframe

Data_Collection_API.ipynb

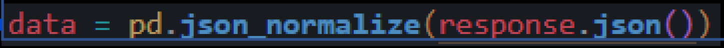
Data Collection – SpaceX API

1. Get API Response for Launches as JSON



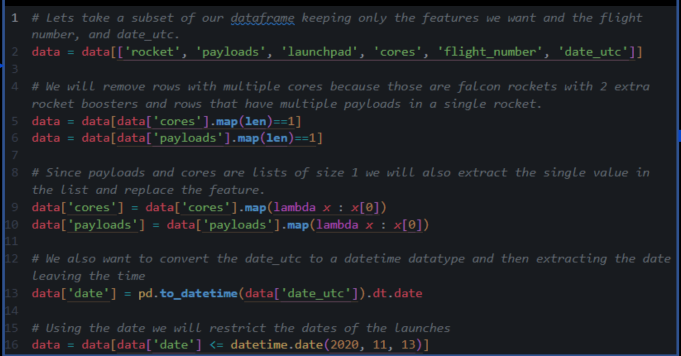
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url)
```

2. Convert JSON to DataFrame using .json()



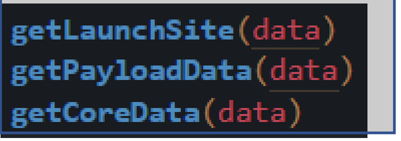
```
data = pd.json_normalize(response.json())
```

3. Transform Data



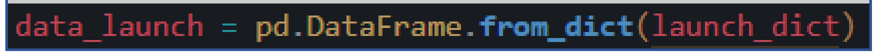
```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# 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 single rocket.
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)]
```

4. Get rest of data



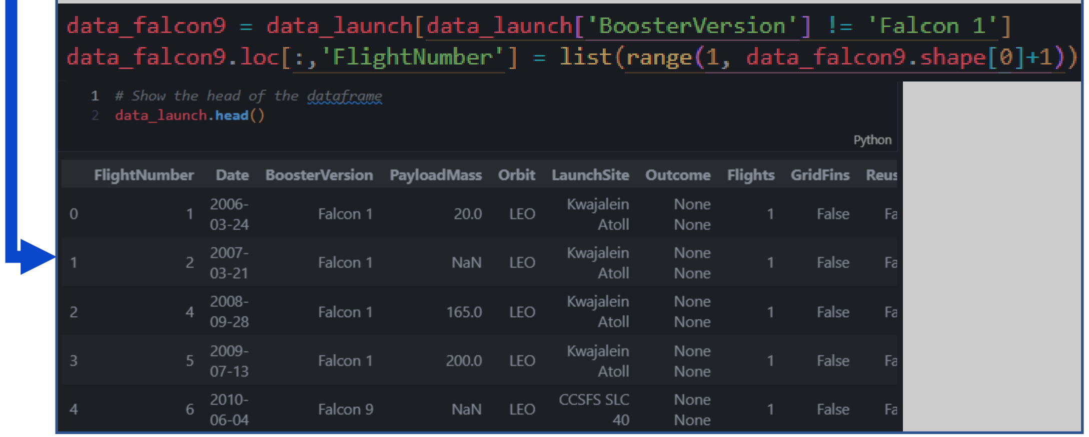
```
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

5. Unify and re-arrange all data into Df



```
data_launch = pd.DataFrame.from_dict(launch_dict)
```

6. Filter the Df for Falcon 9



```
data_falcon9 = data_launch[data_launch['BoosterVersion'] != 'Falcon 1']
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
```

```
# Show the head of the dataframe
data_launch.head()
```

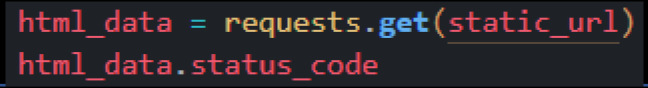
Python

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reus
0	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	Fa
1	2	2007-03-21	Falcon 1	NaN	LEO	Kwajalein Atoll	None None	1	False	Fa
2	4	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	Fa
3	5	2009-07-13	Falcon 1	200.0	LEO	Kwajalein Atoll	None None	1	False	Fa
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	Fa

Data_Collection_API.ipynb

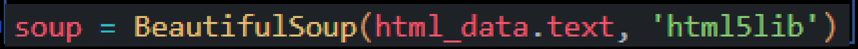
Data Collection – SpaceX Scraping

1. Get HTML Response for Launches



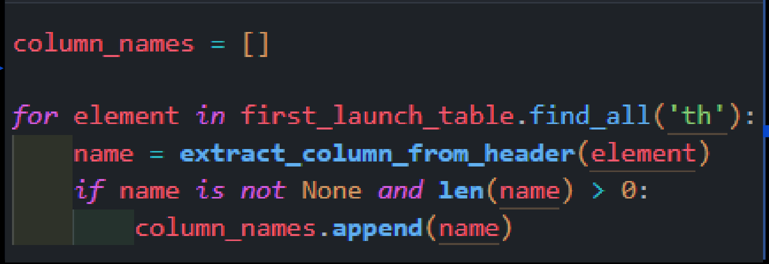
```
html_data = requests.get(static_url)
html_data.status_code
```

2. Initiate BeautifulSoup



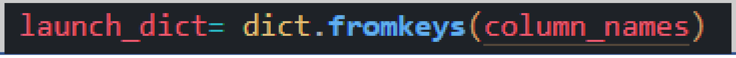
```
soup = BeautifulSoup(html_data.text, 'html5lib')
```

3. Get all tables



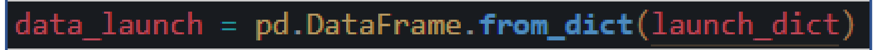
```
column_names = []

for element in first_launch_table.find_all('th'):
    name = extract_column_from_header(element)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

4. Re-arrange and fill in data into a dictionary

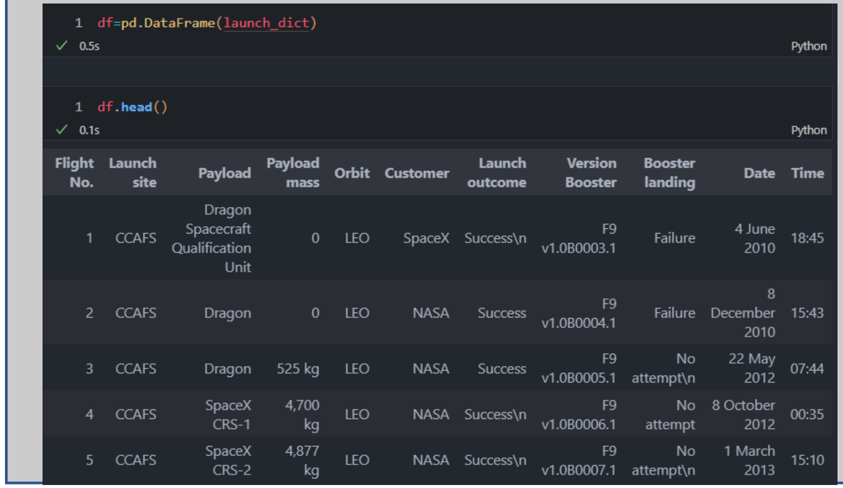


```
launch_dict= dict.fromkeys(column_names)
```

5. Unify and re-arrange all data into Df



```
data_launch = pd.DataFrame.from_dict(launch_dict)
```

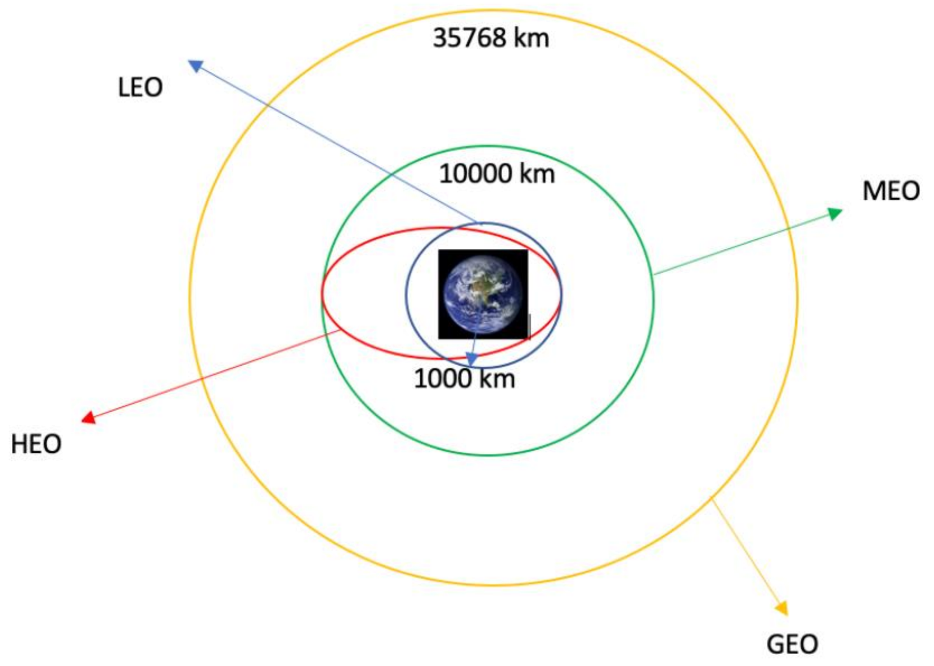


```
df=pd.DataFrame(launch_dict)
```
✓ 0.5s Python

```
df.head()
```
✓ 0.1s Python

Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	F9 v1.0B0003.1	Failure	4 June 2010	18:45
2	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43
3	CCAFS	Dragon	525 kg	LEO	NASA	Success	F9 v1.0B0005.1	No attempt\n	22 May 2012	07:44
4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success\n	F9 v1.0B0006.1	No attempt	8 October 2012	00:35
5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success\n	F9 v1.0B0007.1	No attempt\n	1 March 2013	15:10

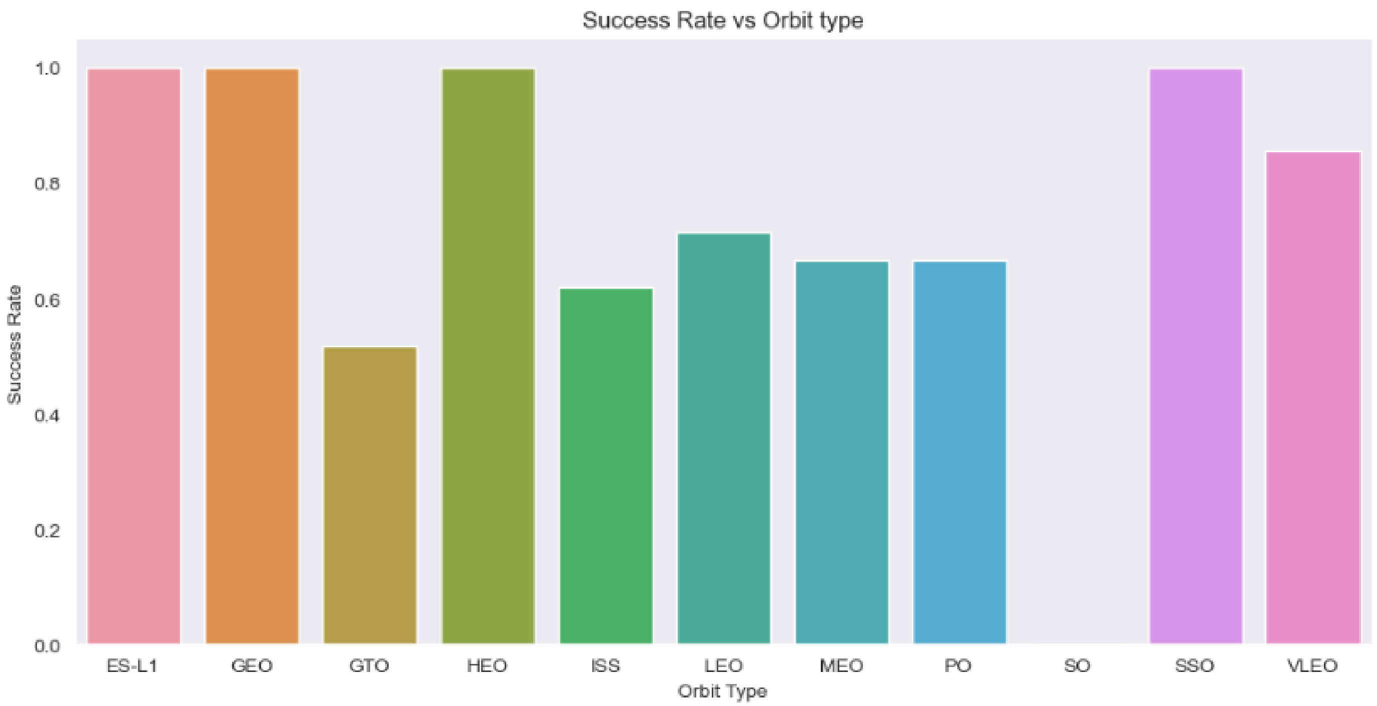
Data_Collection_API.ipynb

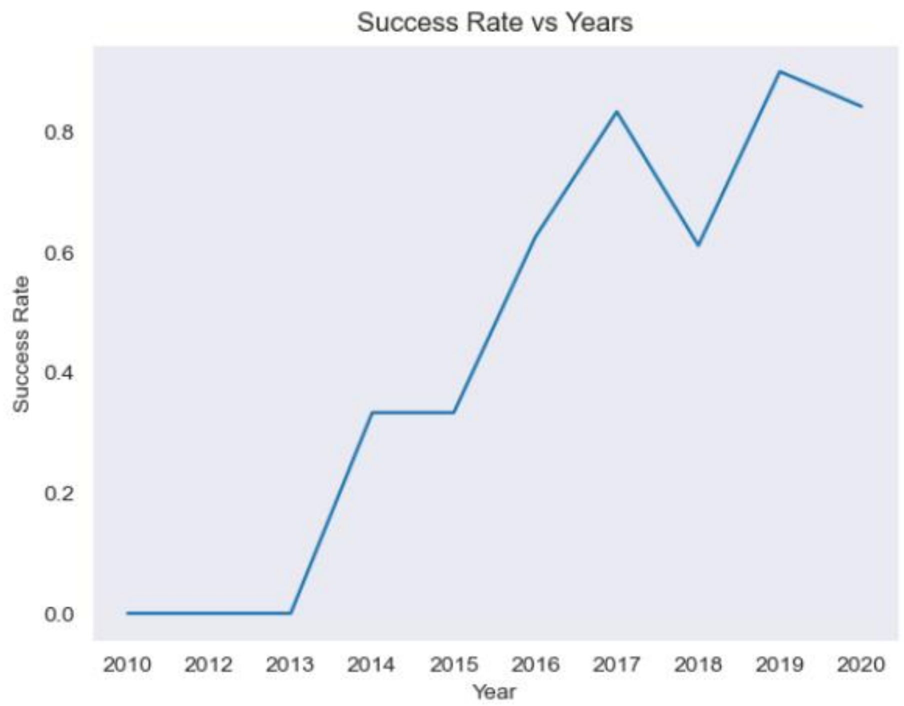


Data Wrangling

- Exploratory data analysis was performed in order to determine the training labels.
- The number of launches at each site as well as the number and occurrence of each orbits where calculated.
- We created landing outcome label from outcome column and exported the results to csv.
- Notebook link

Data_Wrangling.ipynb





EDA with Data Visualization

- The data was explored visually by plotting
  - Left: Success Rate vs Orbit type
  - Right: Success Rate vs Year
- Notebook link

EDA_data_visualization.ipynb

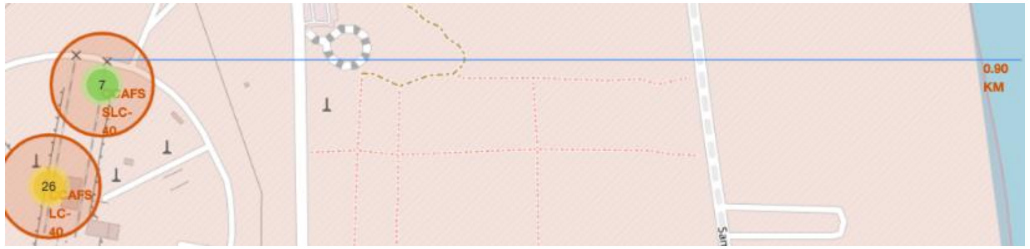
EDA with SQL

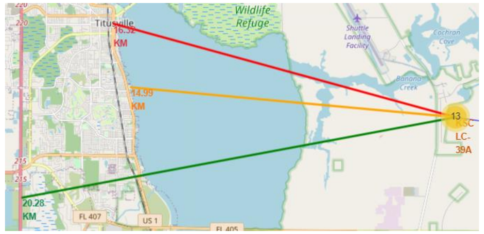
- The SpaceX dataset was loaded into a local SQLite database we created with SQLALchemy.
- Exploratory Data Analysis was then performed by querying the database using raw SQL queries via the ipython-sql in order but not limited to:
    - Find the names of the unique launch sites in the space mission
    - Filter the records for launch sites that begin with the string 'CCA’
    - Find the total payload mass carried by boosters launched by NASA (CRS)
    - Find the date when the first successful landing outcome in ground pad was achieved.
    - Calculate the total number of successful and failure mission outcome

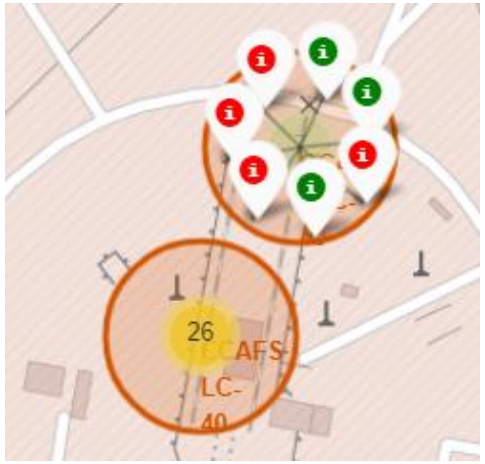
EDA_data_SQL.ipynb

Build an Interactive Map with Folium

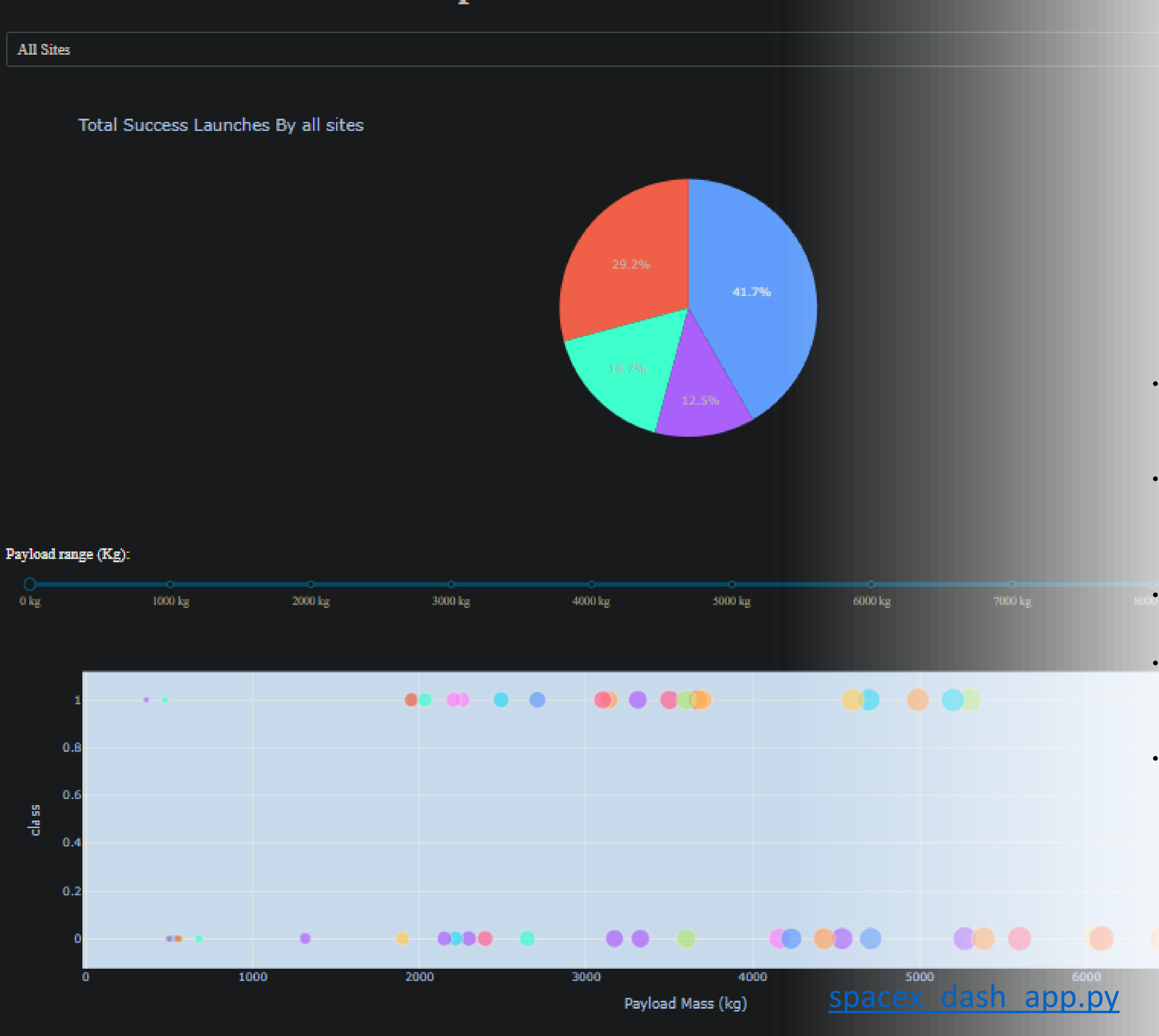
- The Folium map object is a map centered on NASA Johnson Space Center at Houson, Texas
- All sites were marked within map objects such as markets, circles and lines in order to indicate success or failure of launches for each site on a folium map instance.
- Feature outcomes were assigned to 0(failure) and 1(success).
- Color-labeled market clusters were used in order to identify the launch sites that have high success rate.
- The distances between the sites where calculated and some of the questions answered where:
  - Are launch sites near railways, highways and coastlines?
  - Is certain distance kept between the launch sites and the cities?







Launch_site_location_Folium.ipynb



All Sites

spacex_dash_app.py

Build a Dashboard with Plotly Dash

- Launch Sites Dropdown List:
    - Added a dropdown list to enable Launch Site selection.
- Pie Chart showing Success Launches (All Sites/Certain Site):
    - Added a pie chart to show the total successful launches count for all sites and the
- Success vs. Failed counts for the site, if a specific Launch Site was selected.
- Slider of Payload Mass Range:
    - Added a slider to select Payload range.
- Scatter Chart of Payload Mass vs. Success Rate for the different Booster Versions:
    - Added a scatter chart to show the correlation between Payload and Launch Success.

Predictive Analysis (Classification)

1. Loaded data, transformed data using numpy and pandas, split data into training and testing.

2. Created different ML models and tuned them with hyperparameters using GridSearchCV.

3. Using the accuracy as the metric for our models, we picked the best performing parameters for each model.

4. Compared the best performing methods.

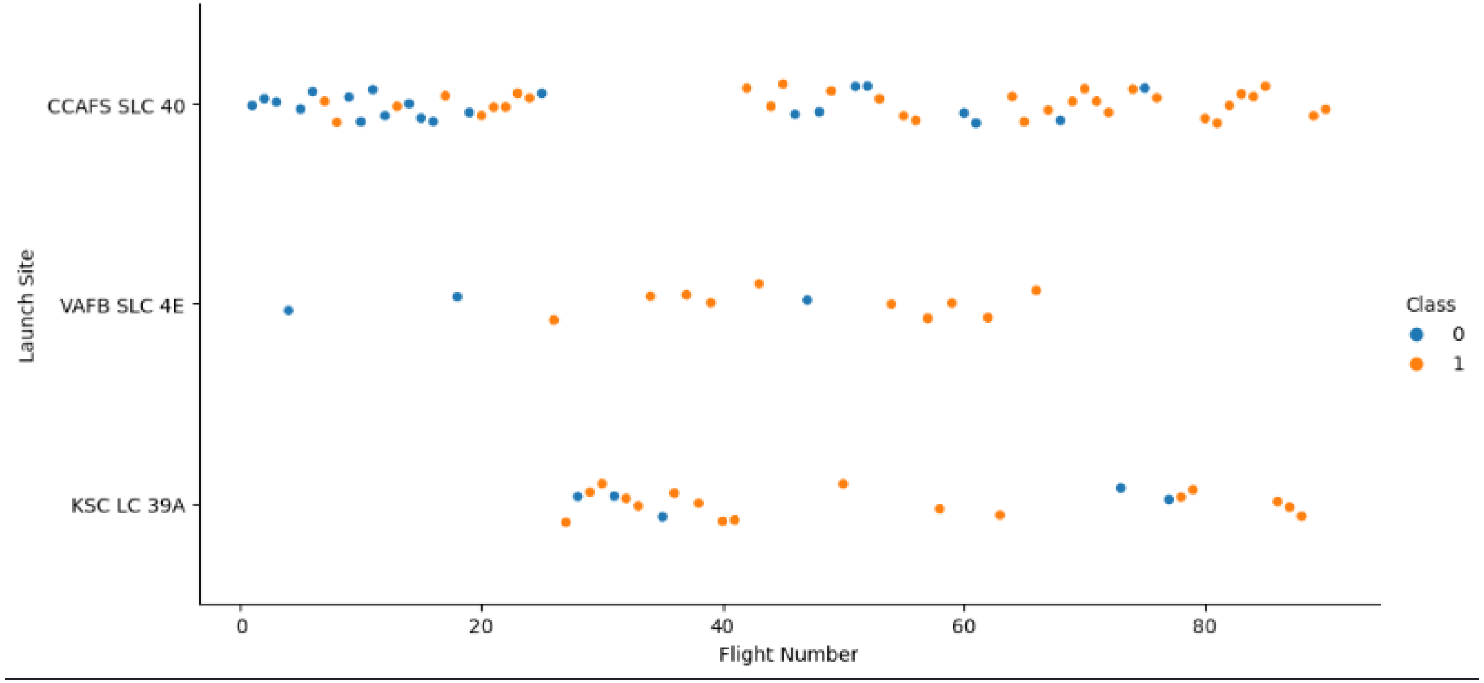
Prediction.ipynb

Results

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

Section 2

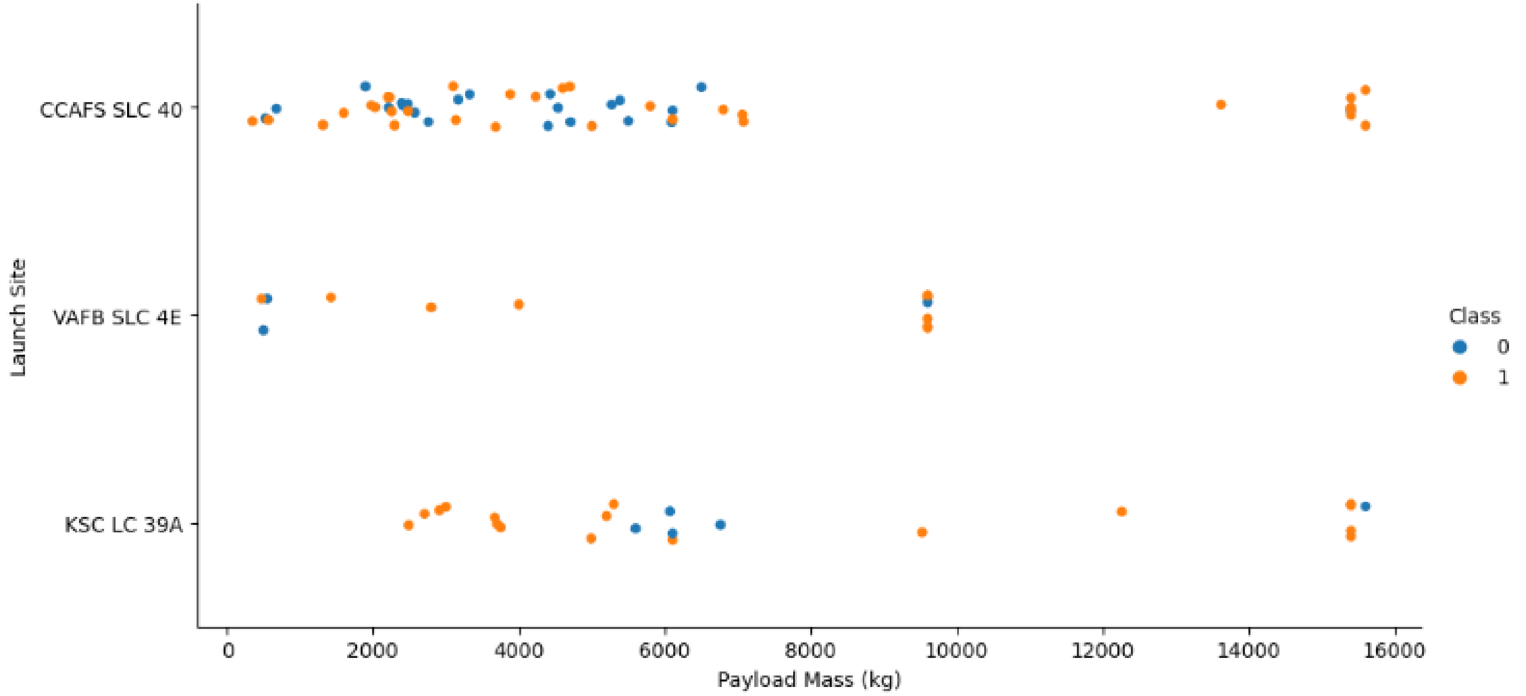
Insights drawn from EDA



Flight Number vs. Launch Site

EDA_data_visualization.ipynb

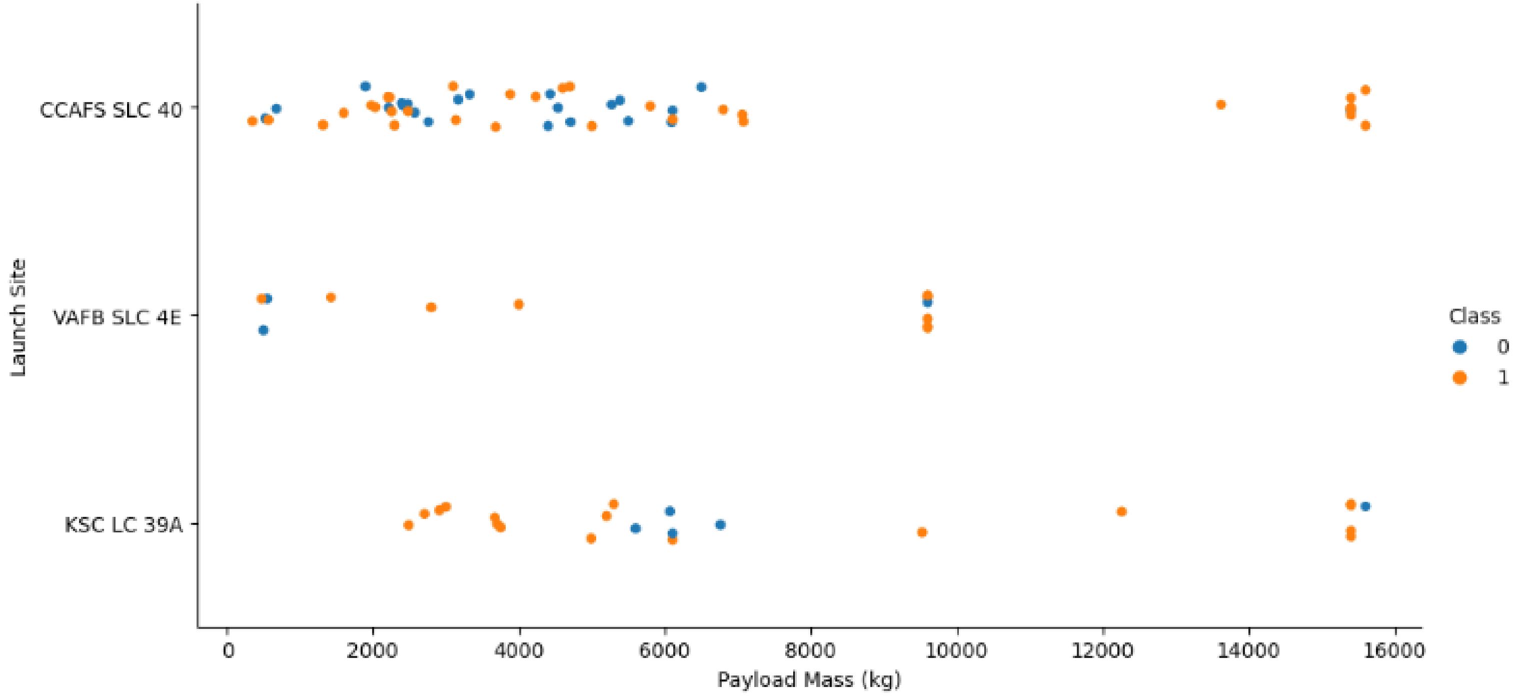
- Most of the earliest flights failed while the latest flights all succeeded.
- The CCAFS SLC 40 launch has the least lanches.
- VAFB SLC 4E and KSC LC 39A have the highest success rates.
- We can loosely assume that higher flight number is linked to higher success rates.



Payload vs. Launch Site

- The higher the payload mass, the higher the success rate.
- Most of the launches with payload mass > 7000 kg were successful.
- KSC LC 39A has almost 100% success rate for payload > 8000kg and payload < 5500kg

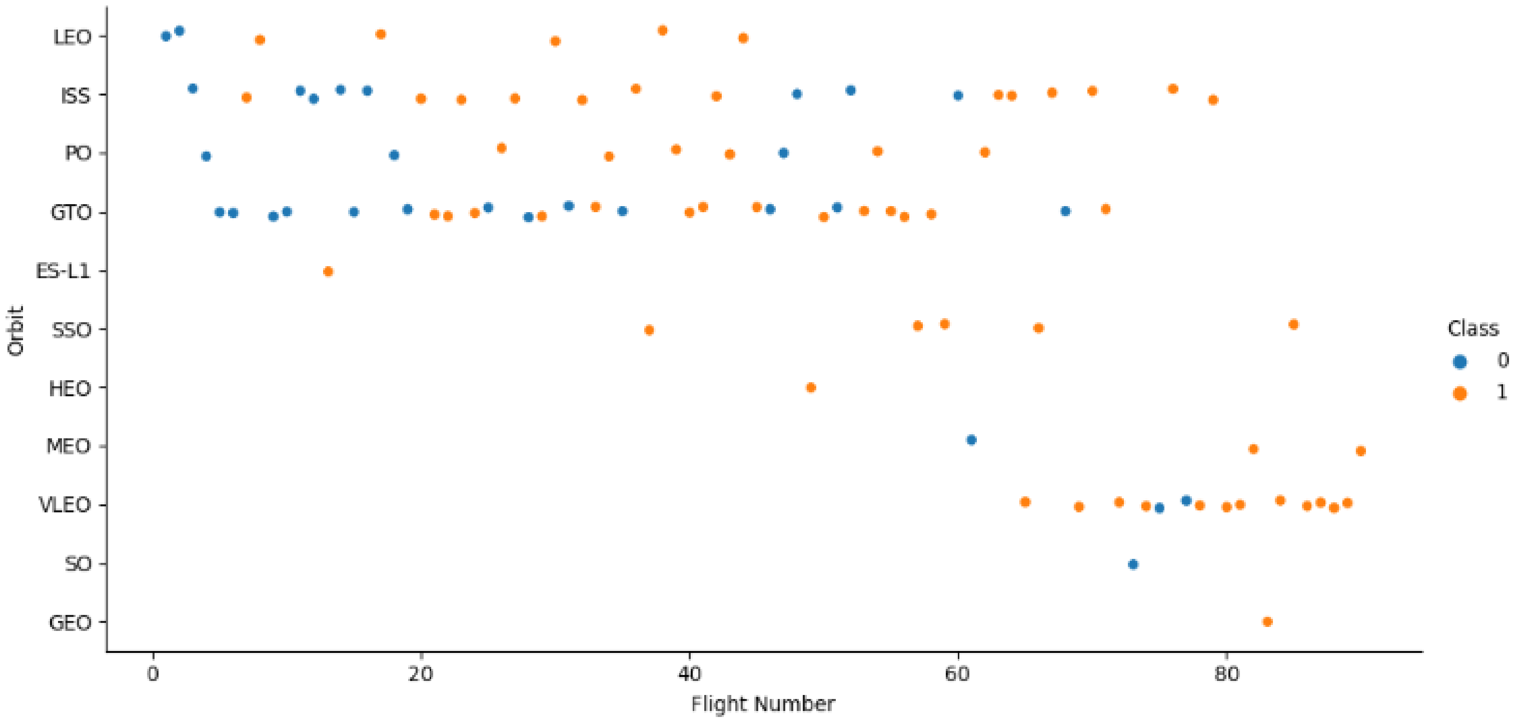
EDA_data_visualization.ipynb



Success Rate vs. Orbit Type

- The higher the payload mass, the higher the success rate.
- Most of the launches with payload mass > 7000 kg were successful.
- KSC LC 39A has almost 100% success rate for payload > 8000kg and payload < 5500kg

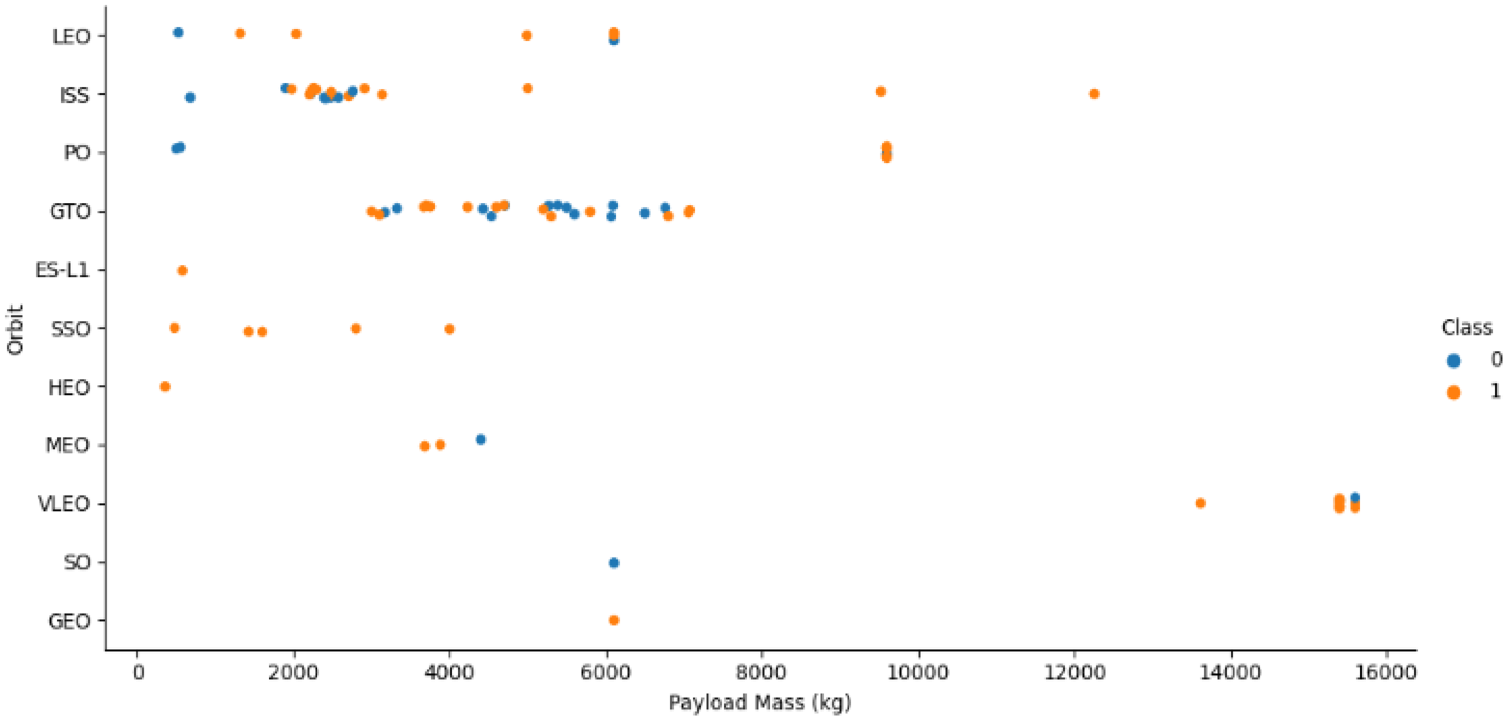
EDA_data_visualization.ipynb



Flight Number vs. Orbit Type

- The larger the flight number per orbit, the greater the success rate.
  - This holds strongly for the LEO
  - weakly for GTO and any other orbit with very few points.

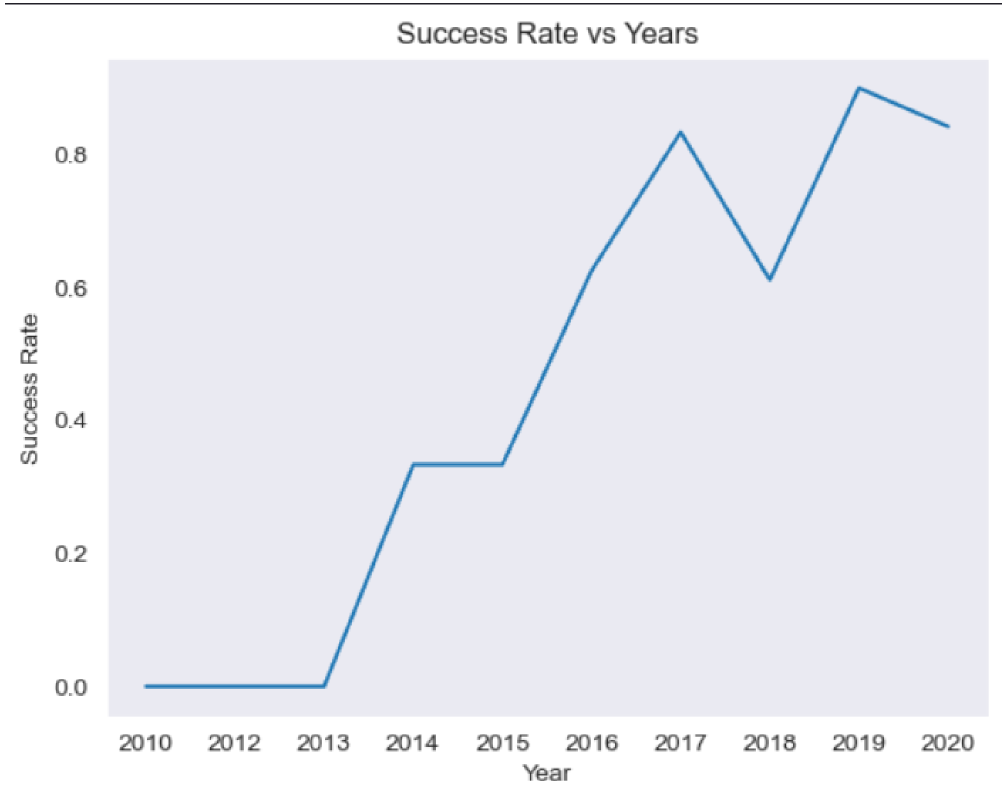
EDA_data_visualization.ipynb



Payload vs. Orbit Type

- Heavy payloads seem to have a negative effect on GTO orbits and positive on GTO/Polar LEO (ISS) orbits.

EDA_data_visualization.ipynb



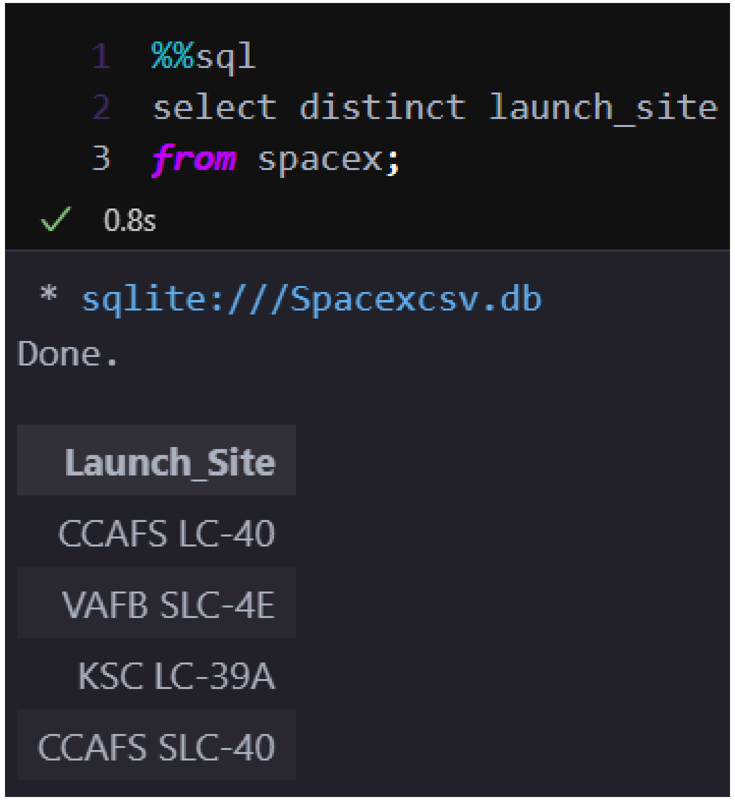
Launch Success Yearly Trend

- The success rate since 2013 followed an increasing trend till 2020.
- There was a dip after 2016 but the momentum was regained after 2018.

EDA_data_visualization.ipynb

All Launch Site Names

- The key word DISTINCT was used in order to show only unique launch sites from the SpaceX data.



```
%%sql
select distinct launch_site
from spacex;
```

✓ 0.8s

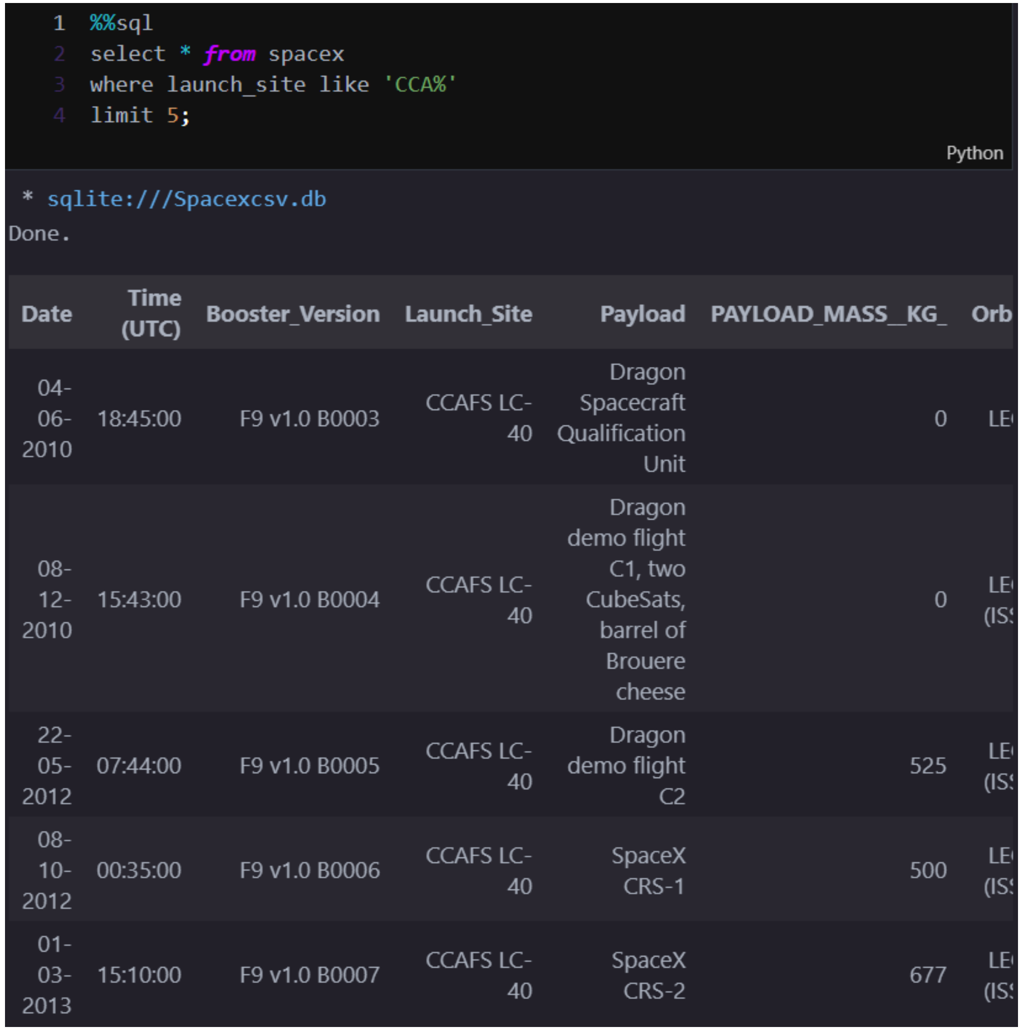
```
 * sqlite:///Spacexcsv.db
Done.
```

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

EDA_data_SQL.ipynb

Launch Site Names Begin with 'CCA'

- The key word limit 5 was used in order to show only the first 5 queries that matches the Where condition.



```
%%sql
select * from spacex
where launch_site like 'CCA%'
limit 5;
```

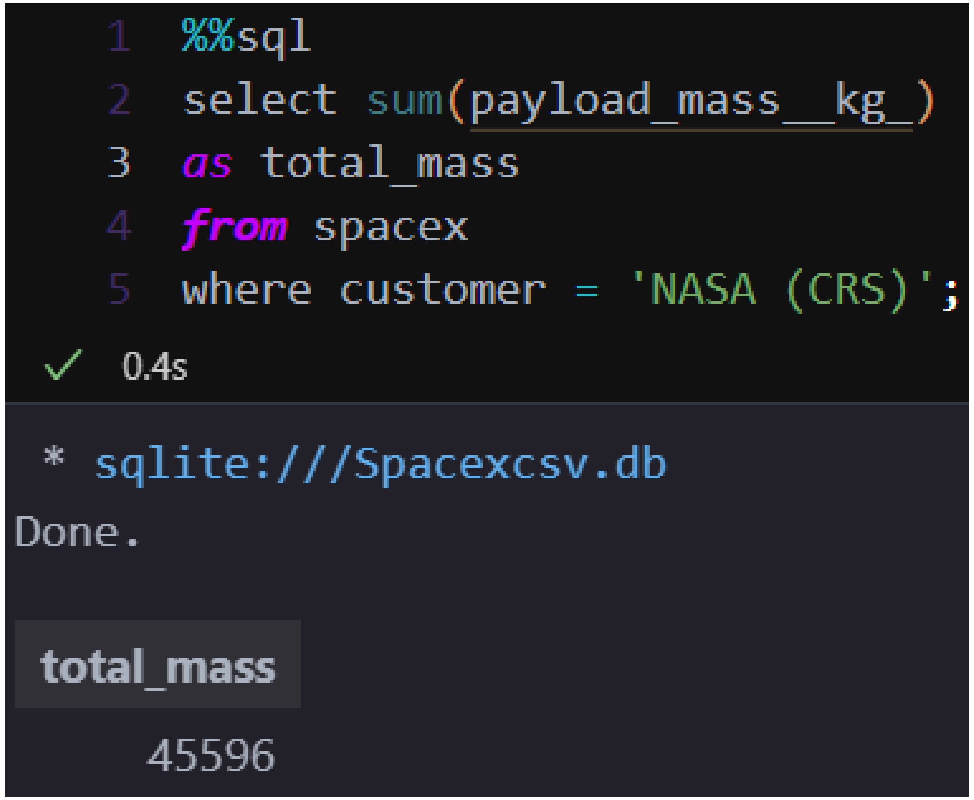
```
* sqlite:///Spacexcsv.db
Done.
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS_KG_	Orb
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LE
08-12-2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LE (IS
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LE (IS
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LE (IS
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LE (IS

EDA_data_SQL.ipynb

Total Payload Mass

- The function sum was used on the column payload_mass__kg_ in order to find the total payload mass.



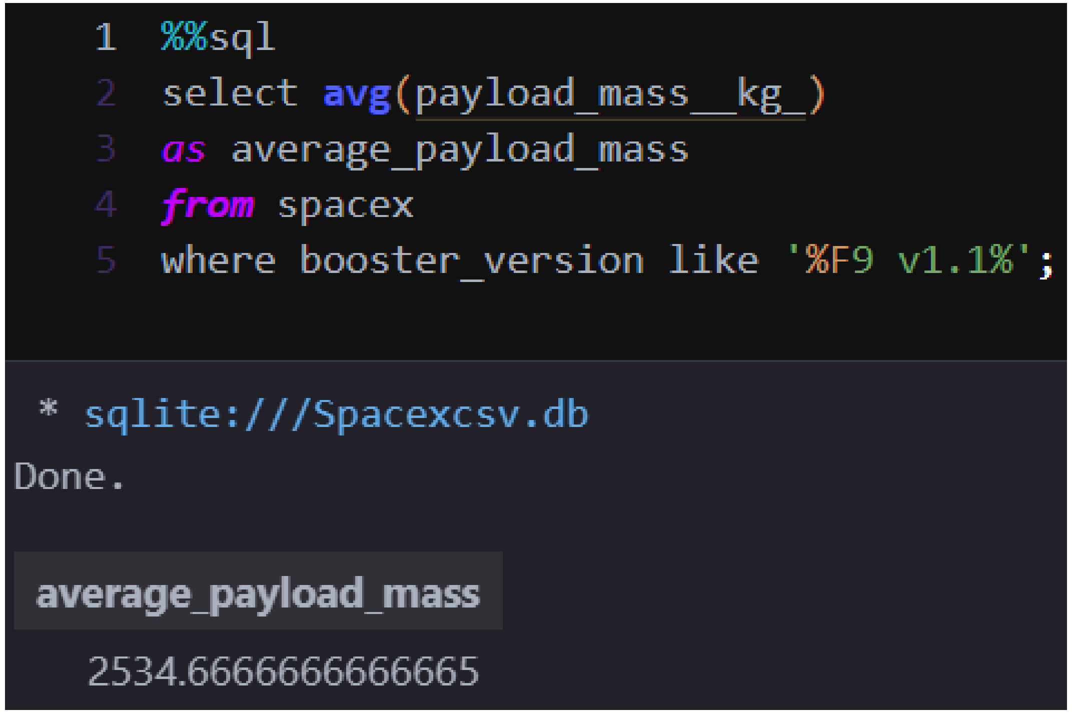
```
%%sql
select sum(payload_mass__kg_)
as total_mass
from spacex
where customer = 'NASA (CRS)';
```

```
✓ 0.4s

 * sqlite:///Spacexcsv.db
Done.

total_mass
     45596
```

EDA_data_SQL.ipynb

Average Payload Mass by F9 v1.1

- The function avg was used on the column payload_mass__kg_ in order to find the average payload mass.
- Then we results where filtered by the wildcard `'%F9 v1.1%'`



```
%%sql
select avg(payload_mass__kg_)
as average_payload_mass
from spacex
where booster_version like '%F9 v1.1%';
```

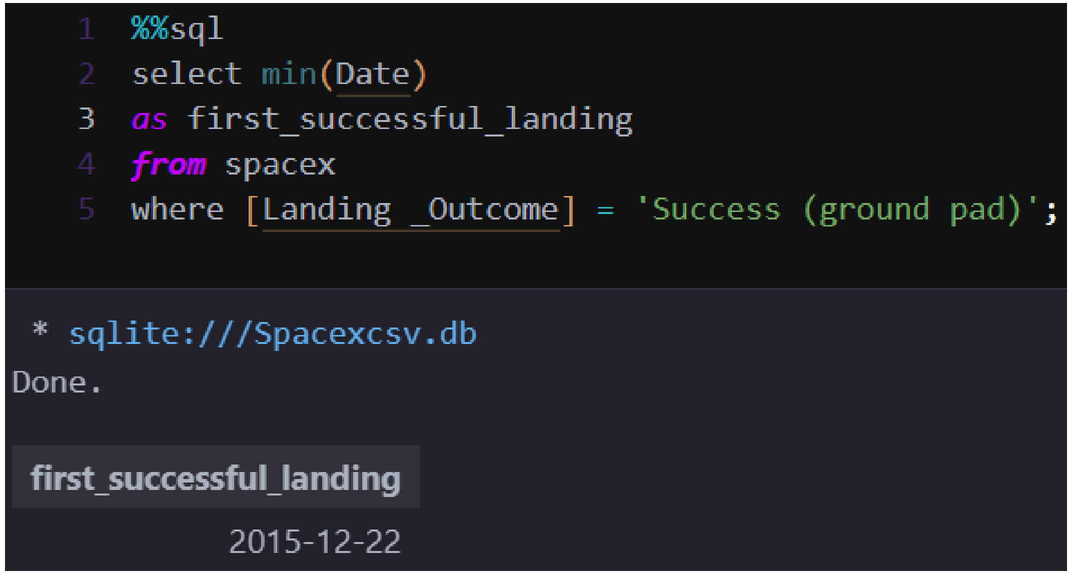
```
 * sqlite:///Spacexcsv.db
Done.
```

average_payload_mass
2534.6666666666665

EDA_data_SQL.ipynb

First Successful Ground Landing Date

- The function min was used on the column Date in order to find the first landing.
- Then we results where filtered by the string 'Success (ground pad)' in order to finally obtain only the successful landing date.



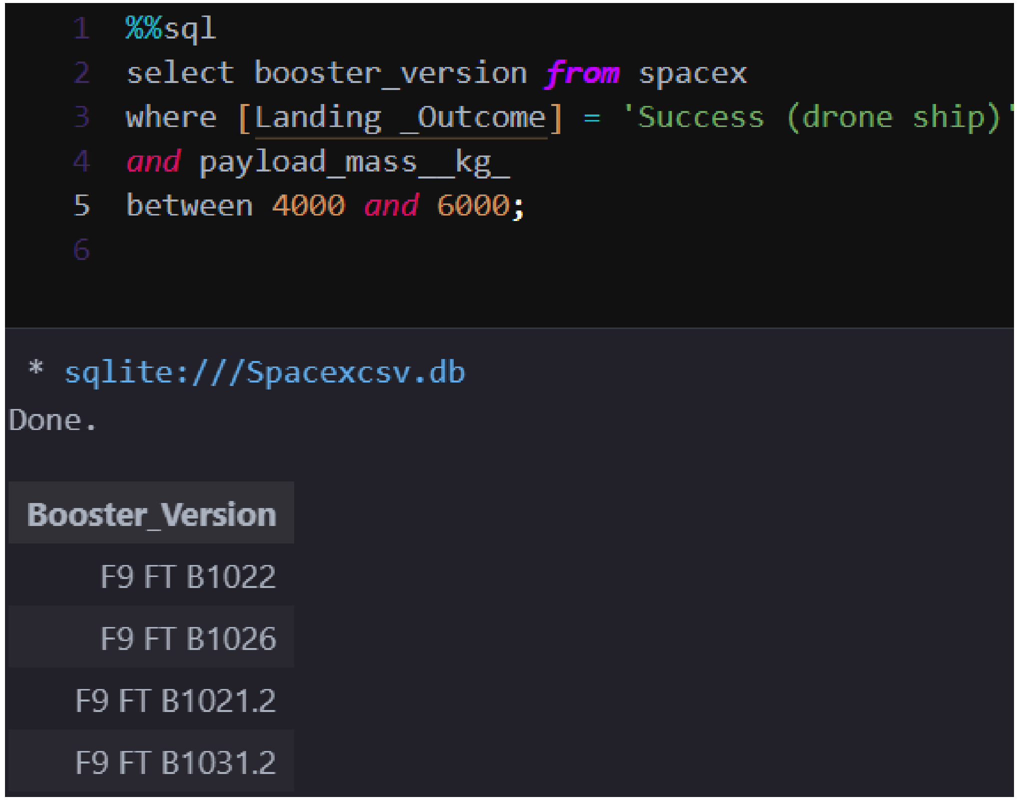
```sql
%%sql
select min(Date)
as first_successful_landing
from spacex
where [Landing _Outcome] = 'Success (ground pad)';
```

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

first_successful_landing
              2015-12-22
```

EDA_data_SQL.ipynb

Successful Drone Ship Landing with Payload between 4000 and 6000

- A subquery was used In conjunction with the between keyword to find the in-between values of 4000 and 6000



```
%%sql
select booster_version from spacex
where [Landing _Outcome] = 'Success (drone ship)'
and payload_mass__kg_
between 4000 and 6000;

```

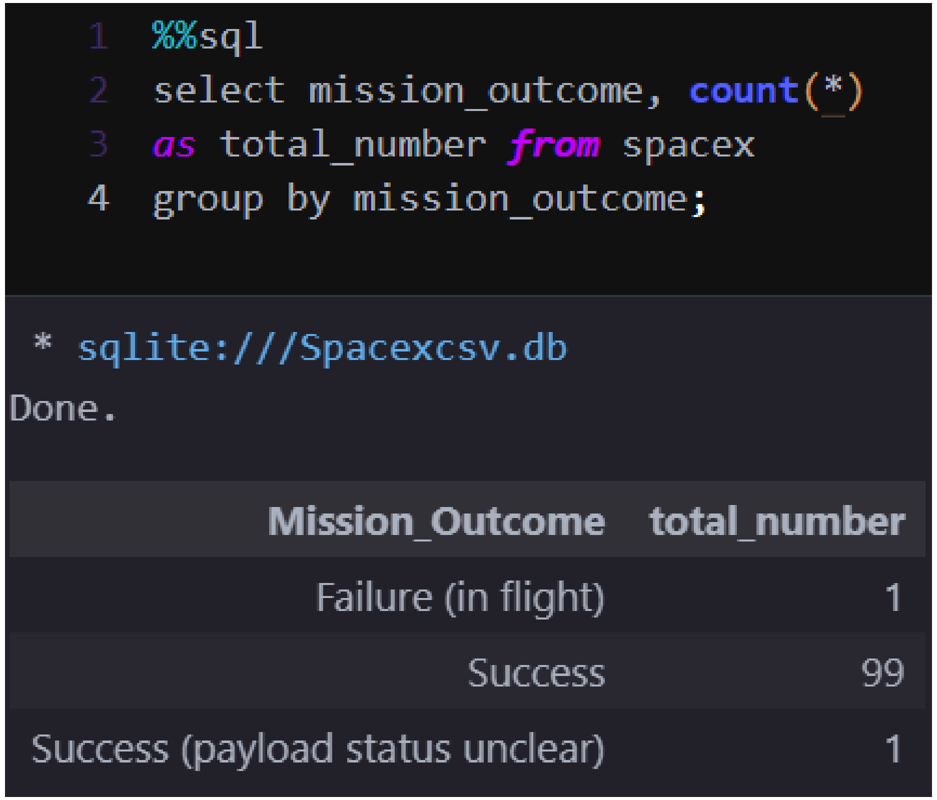
```
 * sqlite:///Spacexcsv.db
Done.
```

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

EDA_data_SQL.ipynb

Total Number of Successful and Failure Mission Outcomes

- The function count was used along with the group by keyword in order to find the total number of successful and failed mission outcomes.



```
%%sql
select mission_outcome, count(*)
as total_number from spacex
group by mission_outcome;
```

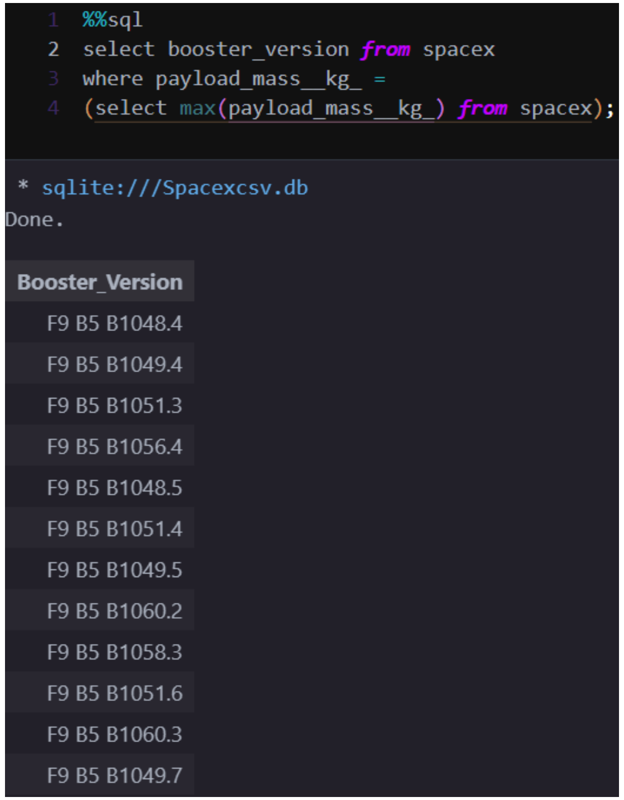
```
 * sqlite:///Spacexcsv.db
Done.
```

Mission_Outcome	total_number
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

EDA_data_SQL.ipynb

Boosters Carried Maximum Payload

- The function max was used within a subquery in order to find the boosters with maximum payload.



```
%%sql
select booster_version from spacex
where payload_mass__kg_ =
(select max(payload_mass__kg_) from spacex);
```

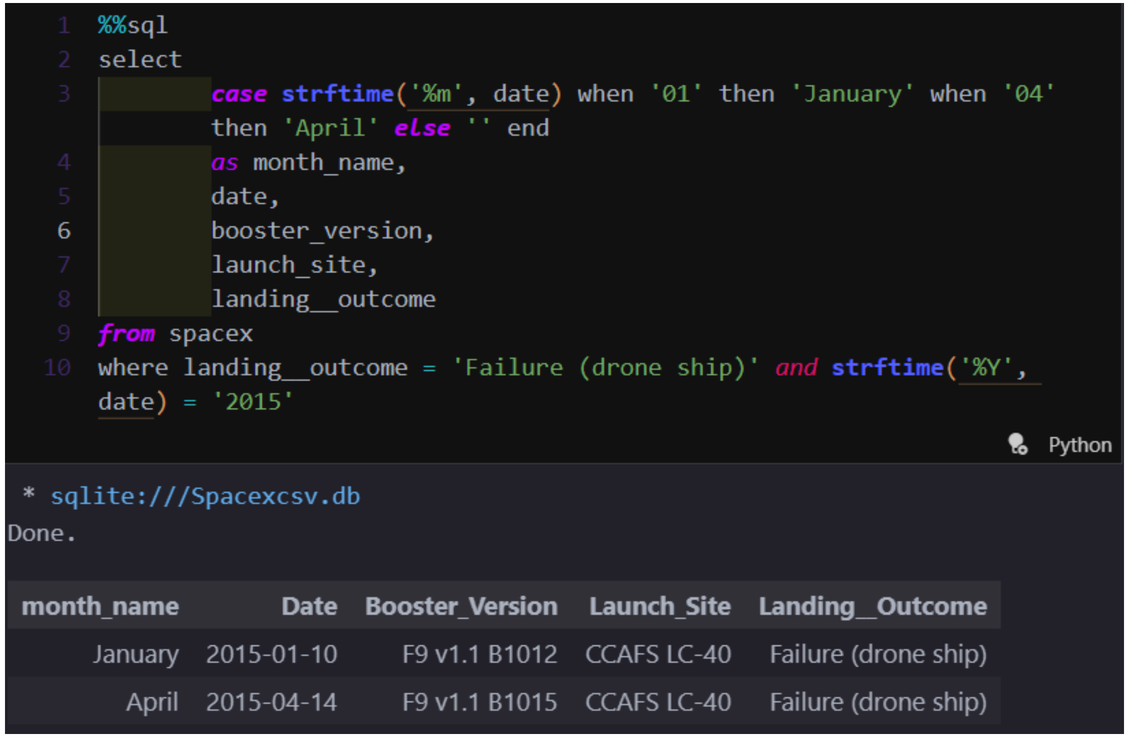
```
 * sqlite:///Spacexcsv.db
Done.
```

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

EDA_data_SQL.ipynb

2015 Launch Records

- Since we used SQLite we did not have access to functions like year, month, day that return strings.
- Strftime was used instead to extract components of dates.
- Keywords case, when, then were used as in order to turn integers to month names.



```
%%sql
select
        case strftime('%m', date) when '01' then 'January' when '04'
        then 'April' else '' end
        as month_name,
        date,
        booster_version,
        launch_site,
        landing__outcome
from spacex
where landing__outcome = 'Failure (drone ship)' and strftime('%Y',
date) = '2015'
```

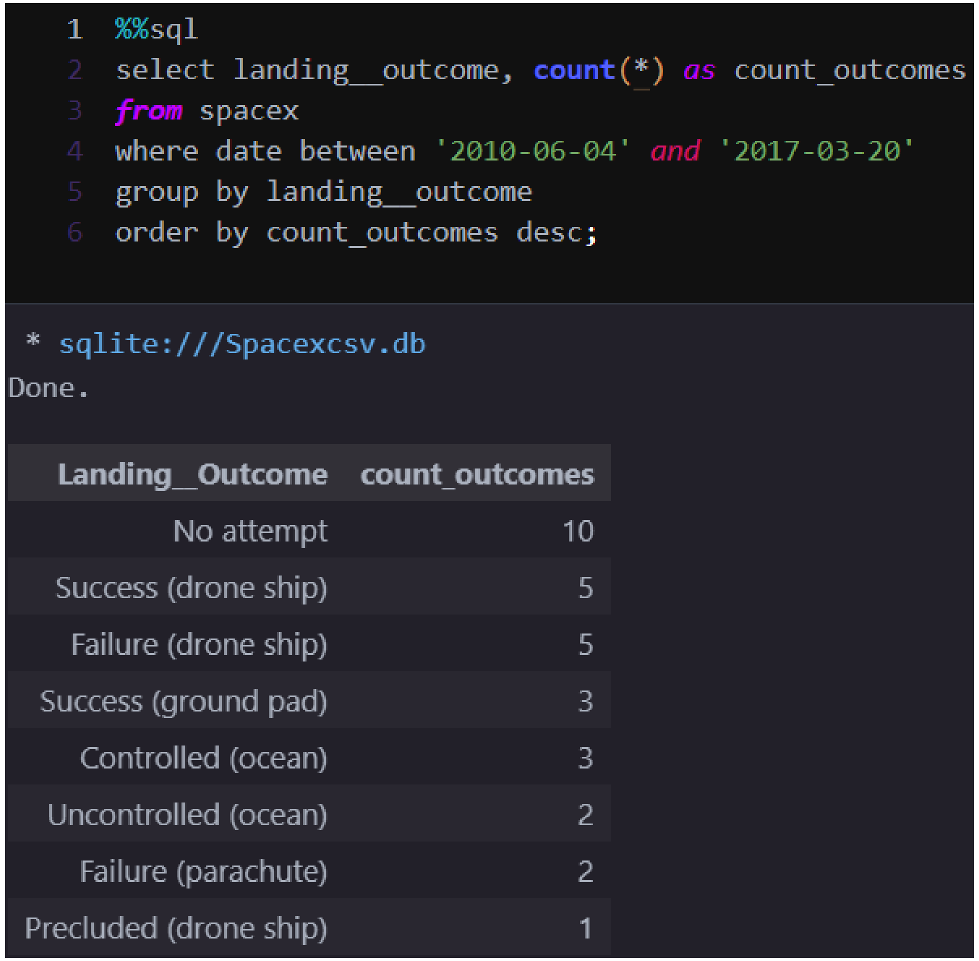
```
 * sqlite:///Spacexcsv.db
Done.
```

month_name	Date	Booster_Version	Launch_Site	Landing__Outcome
January	2015-01-10	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
April	2015-04-14	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

EDA_data_SQL.ipynb

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

- First, we found all dates between 2010-06-04 and 2017-03-20 (line 4).
- We grouped them by landing__outcome (line 5).
- We ordered them in descending mode (line 6).
- Finally, we counted the results



```
1  %%sql
2  select landing__outcome, count(*) as count_outcomes
3  from spacex
4  where date between '2010-06-04' and '2017-03-20'
5  group by landing__outcome
6  order by count_outcomes desc;
```

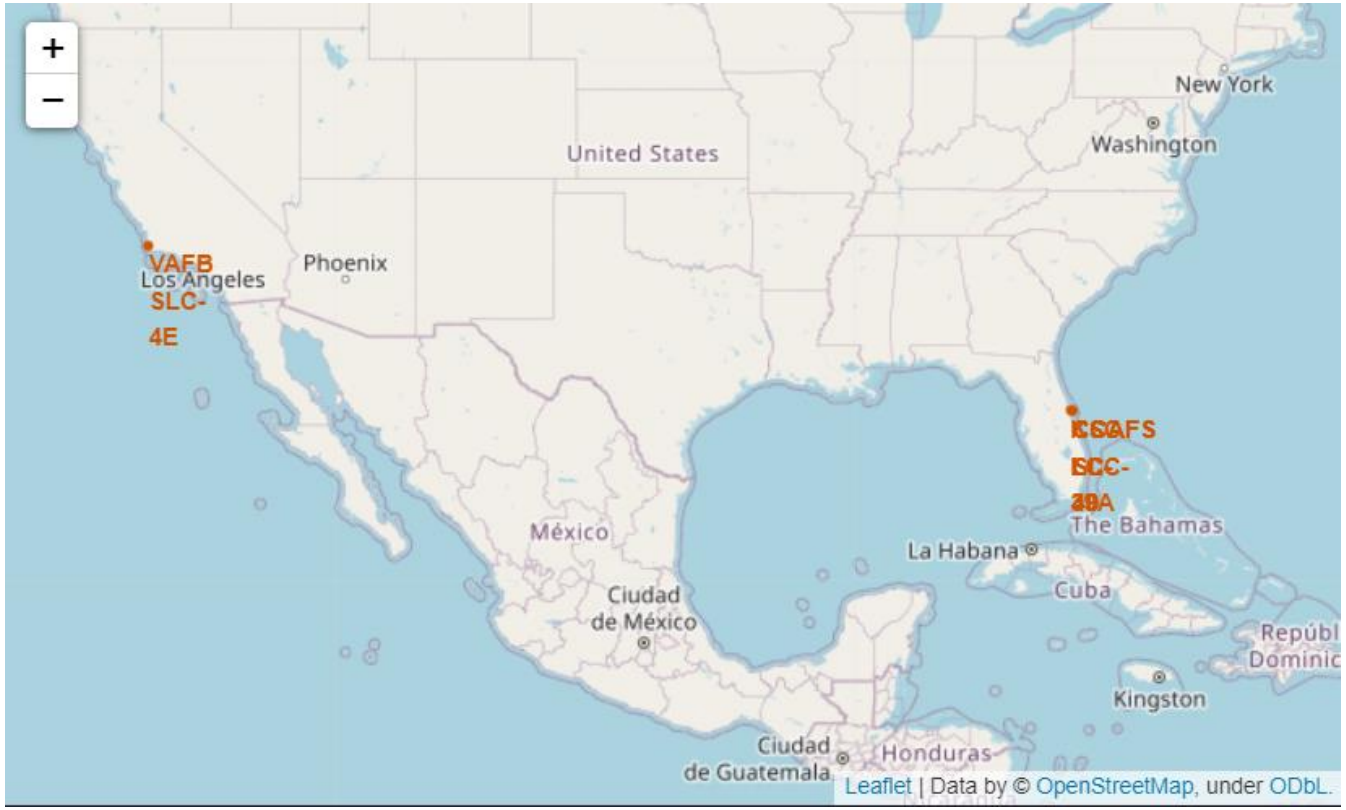
```
 * sqlite:///Spacexcsv.db
Done.
```

Landing__Outcome	count_outcomes
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

EDA_data_SQL.ipynb

Section 3

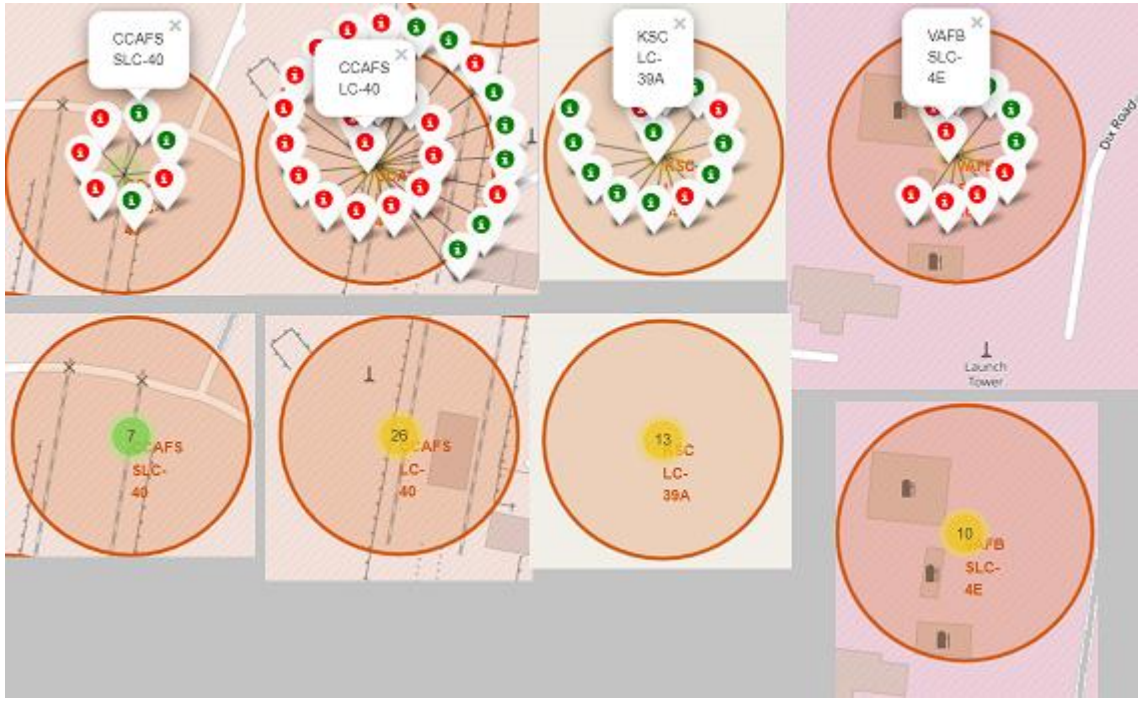
Launch Sites Proximities Analysis



Launch sites in Folium map

- All the launching sites are tangent to coastlines.
- This ensures debris or disengaged part of the rockets are not to fall into habitable areas.

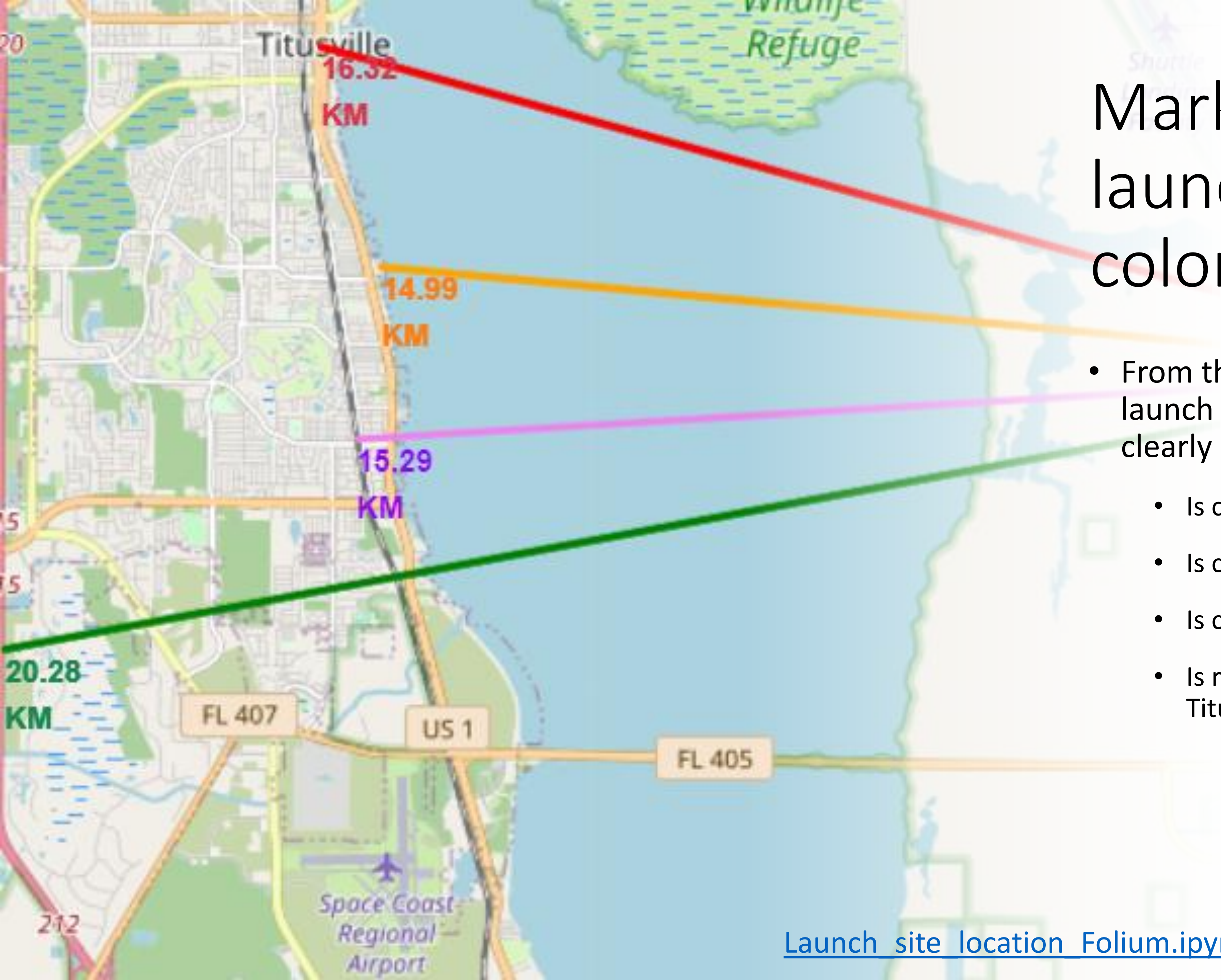
Launch_site_location_Folium.ipynb



Markers showing launch sites with color labels

- Green markers: successful launches
- Red markers: failed launches
- The numbers within green/yellow circles indicate the total mission attempts.

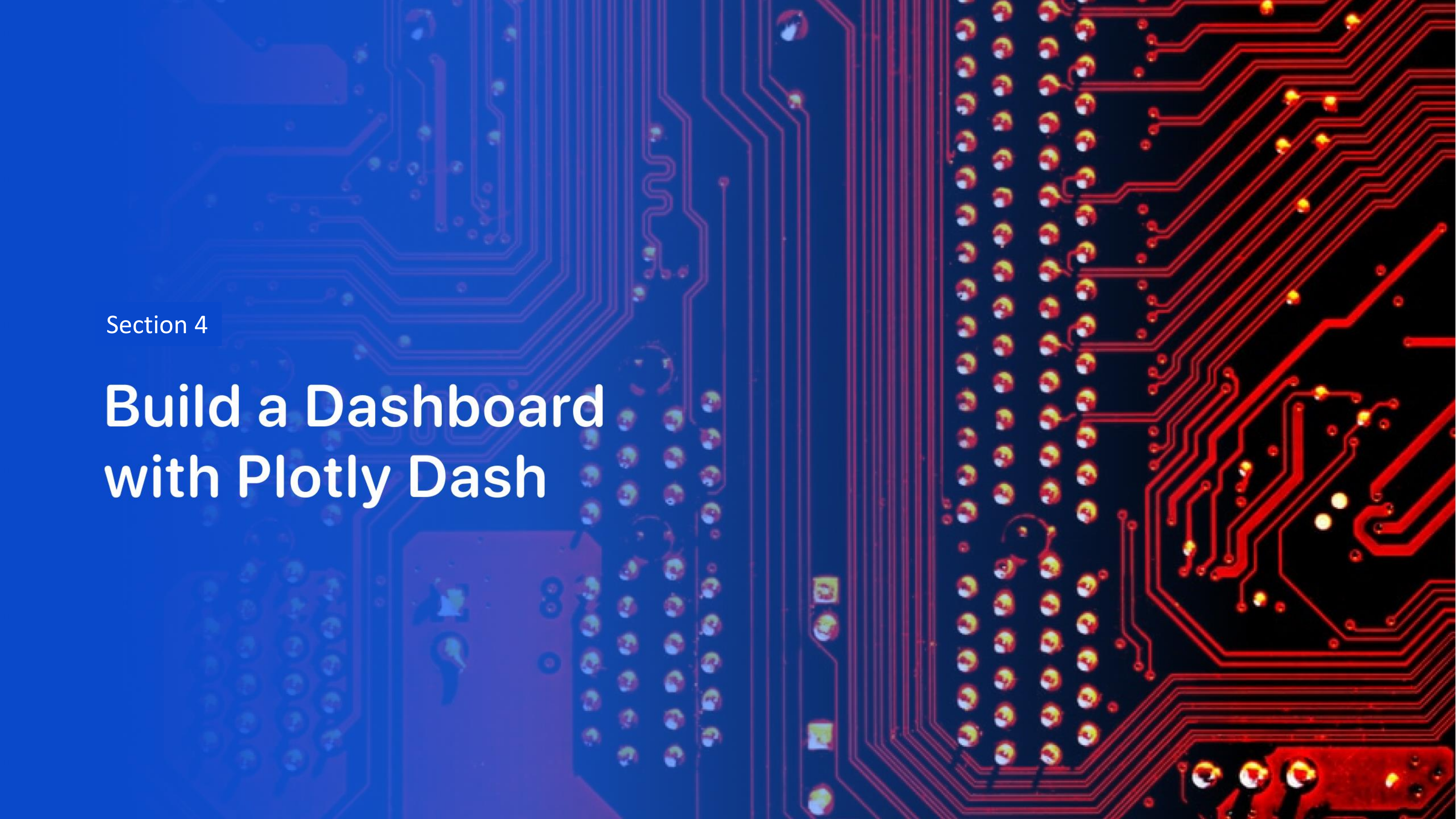
Launch_site_location_Folium.ipynb



Markers showing launch sites with color labels

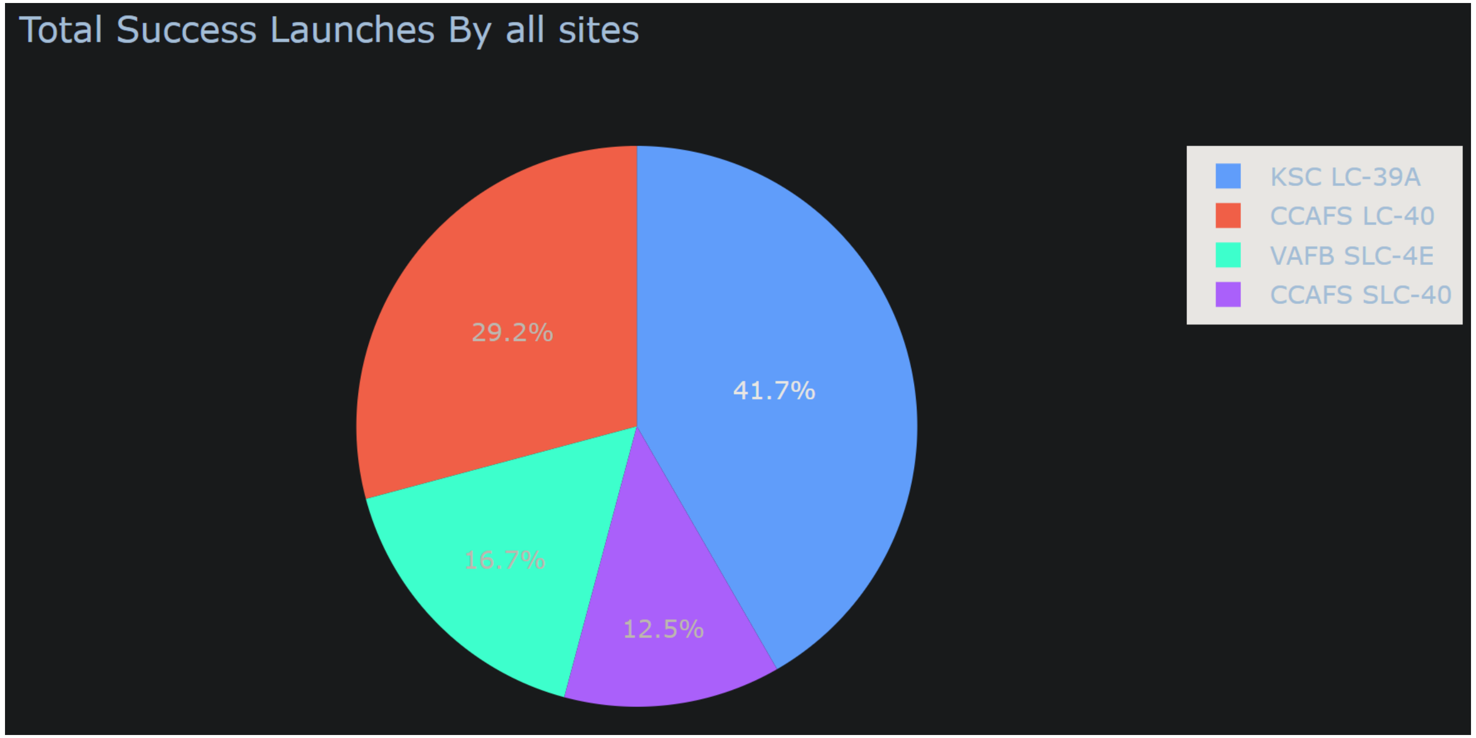
- From the visual analysis of the launch site KSC LC-39A we can clearly see that
  - Is close to railway (15.29 km)
  - Is close to highway (20.28 km)
  - Is close to coastline (14.99 km)
  - Is relative close to its closest city Titusville (16.32 km).

[Launch_site_location_Folium.ipynb]()



Section 4

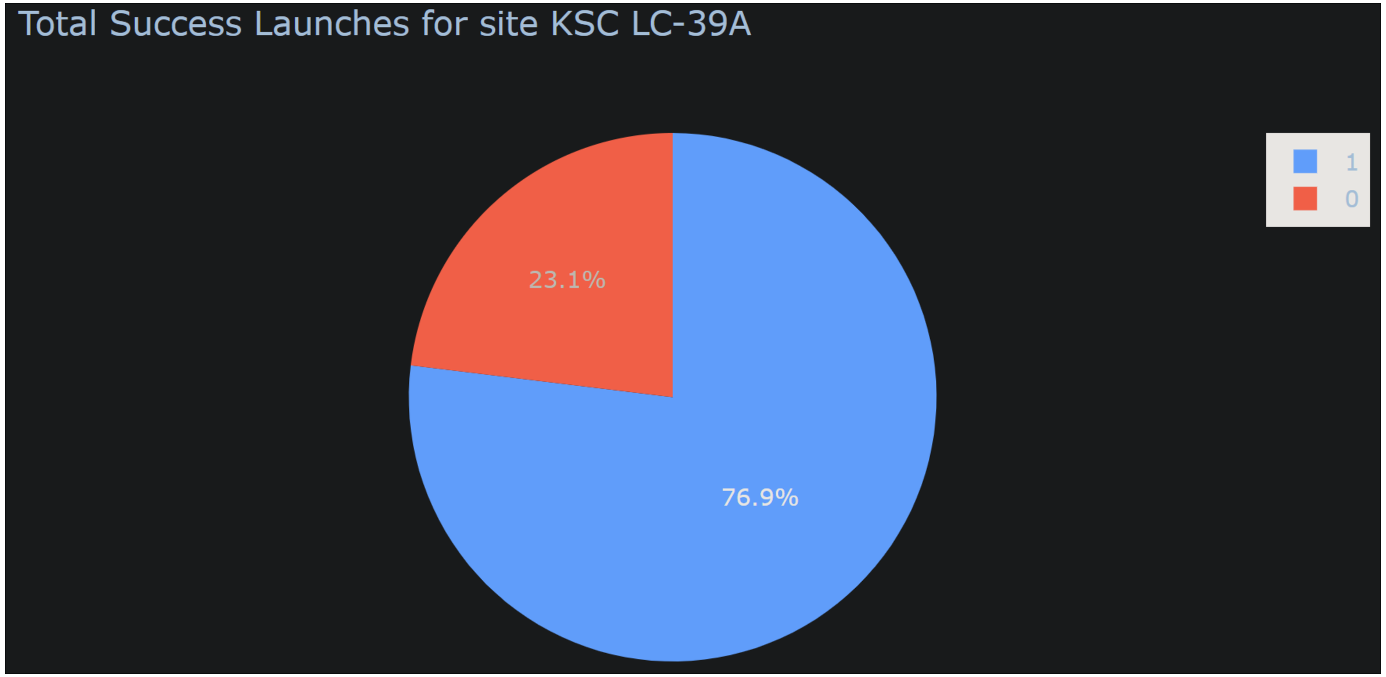
Build a Dashboard with Plotly Dash



Pie chart showing the success percentage achieved by each launch site

- KSC LC-39A had the most successful launches from all sites

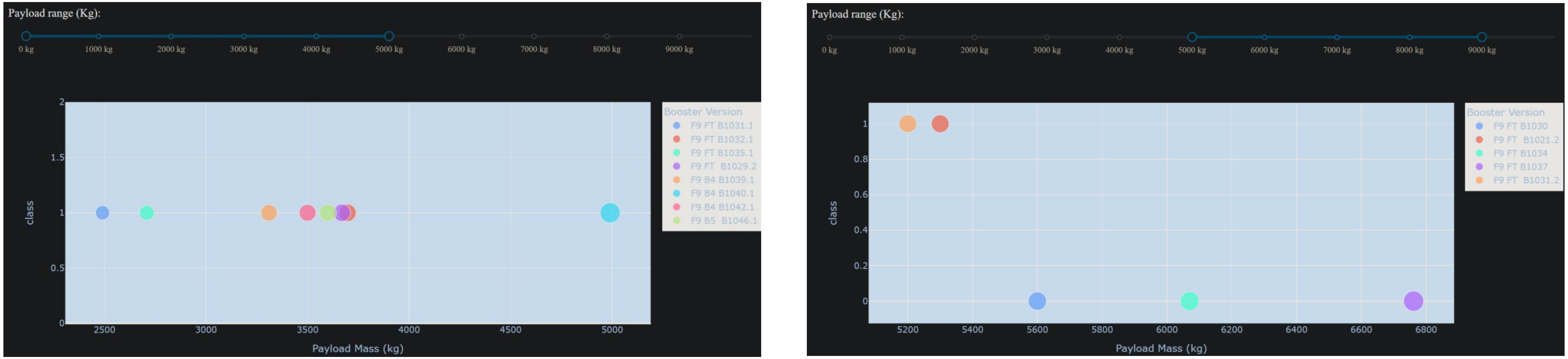
spacex_dash_app.py



Pie chart showing the Launch site with the highest launch success ratio

- KSC LC-39A achieved 76.9% success rate while getting 23.1% failures

spacex_dash_app.py



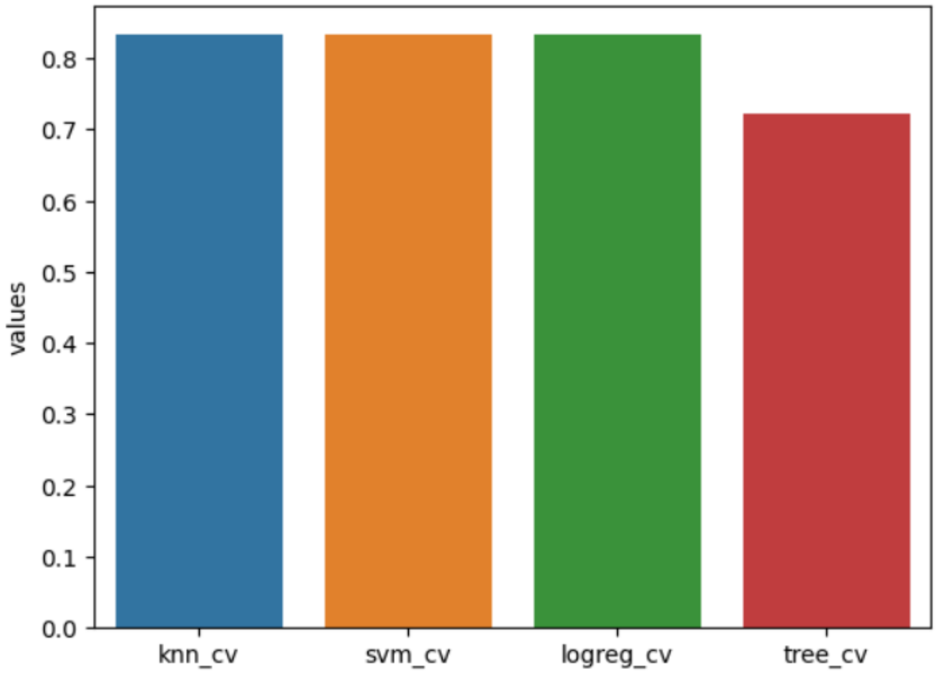
Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider

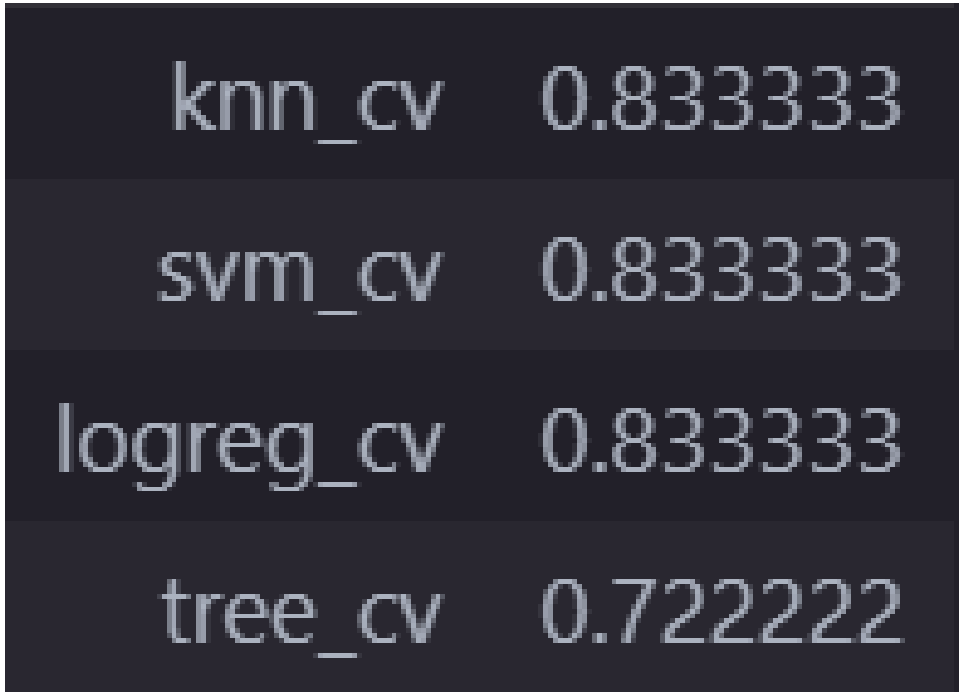
- We can see that there is higher success rate for low weighted payloads.

spacex_dash_app.py

Section 5

Predictive Analysis (Classification)



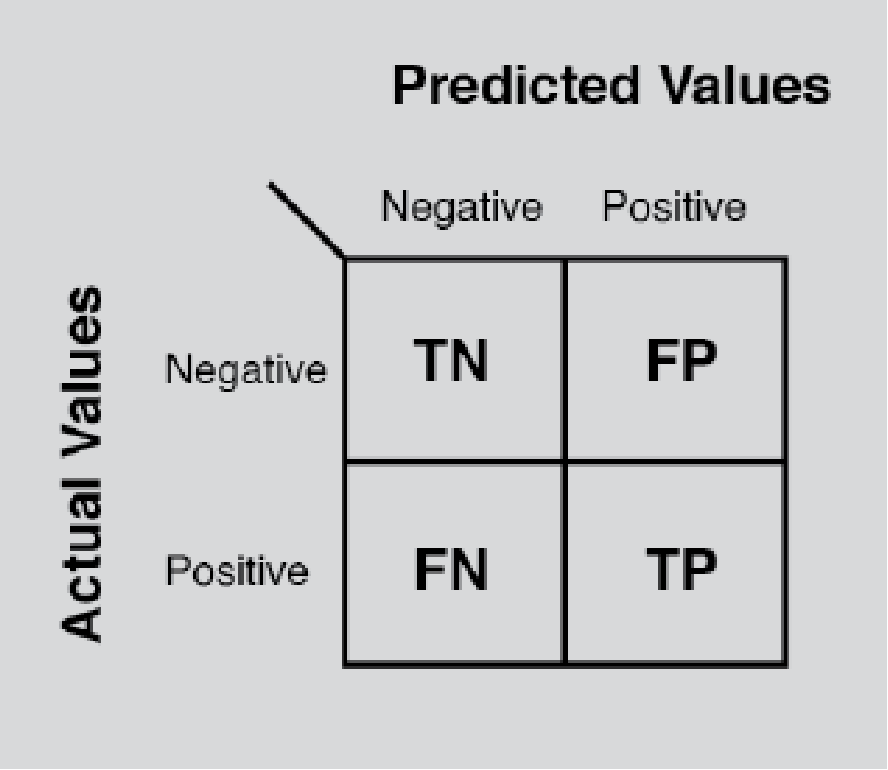


```
   knn_cv   0.833333
   svm_cv   0.833333
logreg_cv   0.833333
  tree_cv   0.722222
```

Classification Accuracy

- There is tie between KNN(knn_cv), Support Vector Machines(svm) and Logistic Regression(logreg_cv) with accuracy of 0.83, which is also the the max value.
- Decision Tree(tree_cv) scored the lowest with 0.72 accuracy.

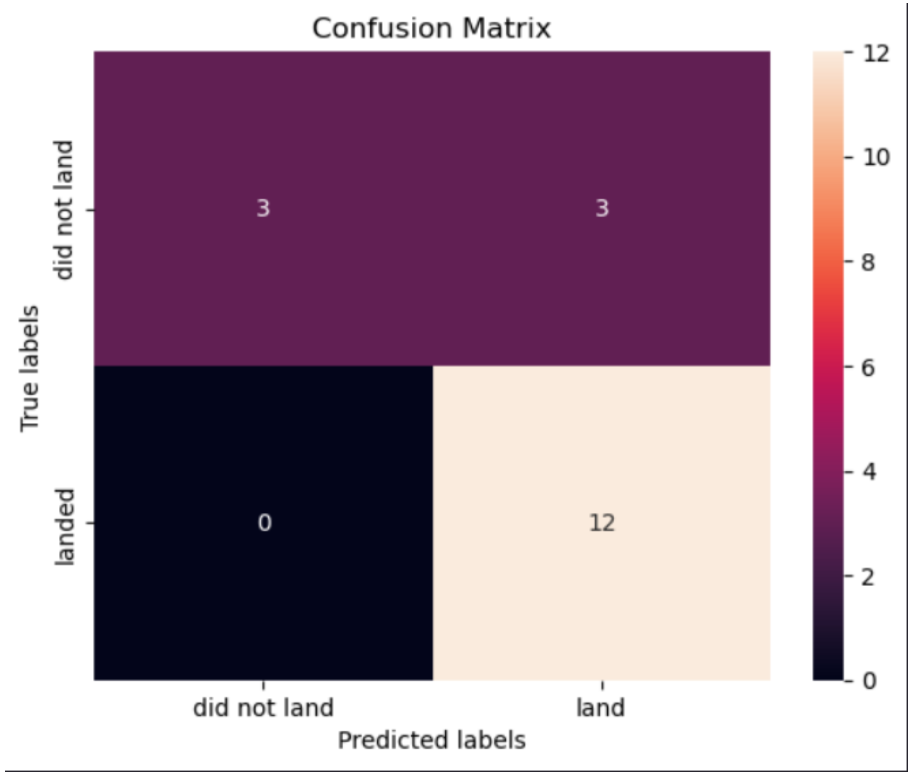
Prediction.ipynb



Predicted Values

Actual Values

	Negative	Positive
Negative	TN	FP
Positive	FN	TP

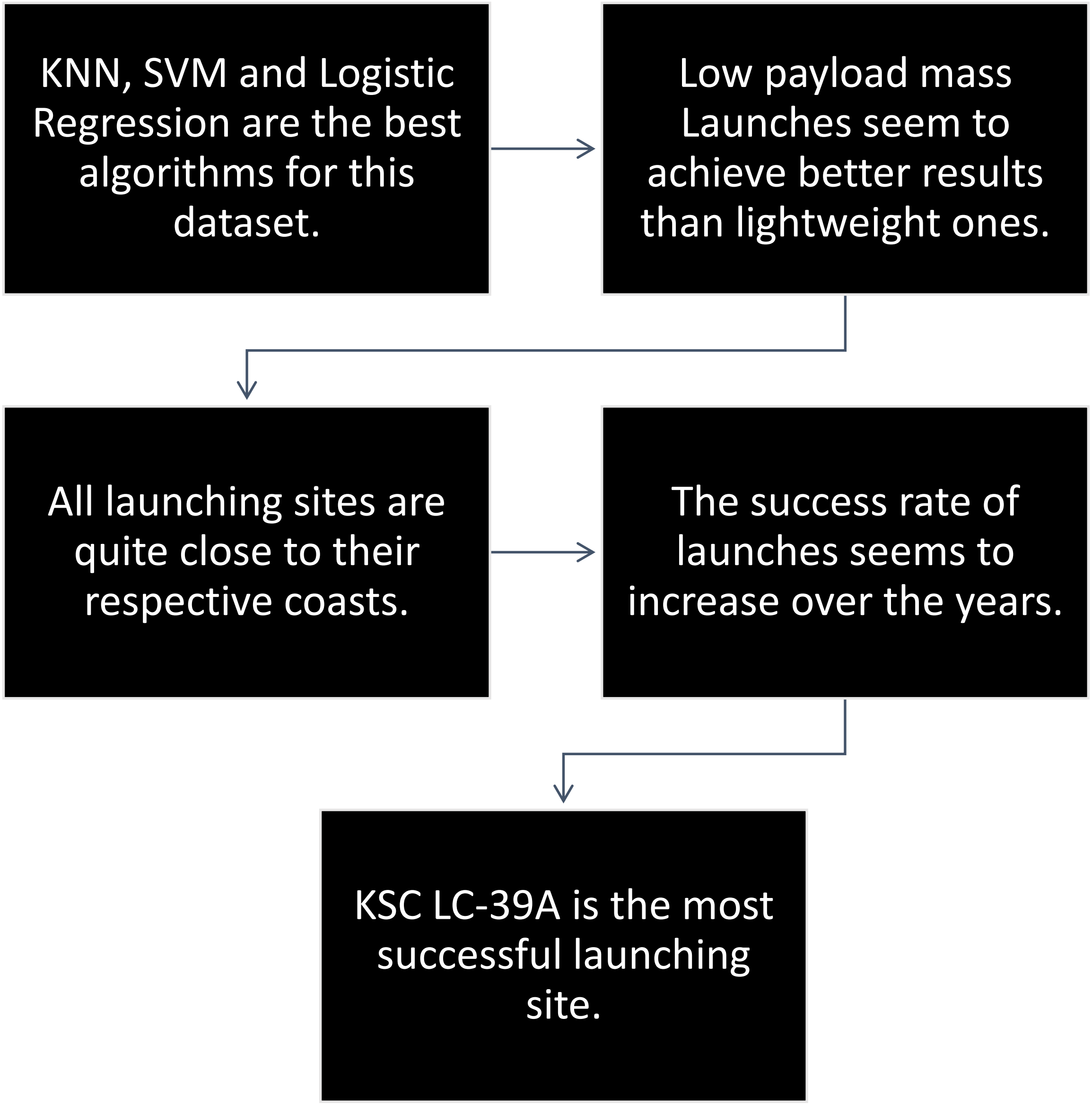


Confusion Matrix

- All KNN(knn_cv), Support Vector Machines(svm) and Logistic Regression(logreg_cv) with accuracy of 0.83 have also the same confusion matrix.
- Only 3 misclassifications

Prediction.ipynb

Conclusions



KNN, SVM and Logistic Regression are the best algorithms for this dataset.

Low payload mass Launches seem to achieve better results than lightweight ones.

All launching sites are quite close to their respective coasts.

The success rate of launches seems to increase over the years.

KSC LC-39A is the most successful launching site.

Thank you!