
IBM Developer
SKILLS NETWORK
Winning Space Race
with Data Science
Indrani Dhanisetty
9/20/2021

Outline

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

Executive Summary

- Summary of Methodologies:
  - Data Collection
  - Data Wrangling
  - EDA Data Visualization
  - EDA with SQL
  - Building interactive map with Folium
  - Building a Dashboard with Plotly Dash
  - Predictive Analysis
- Summary of all results:
  - EDA (Visual and SQL results)
  - Interactive Analytics With Screenshots
  - Predictive Analysis Results

Introduction

- **Project background and context**

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

- **Problems you want to find answers**

  - What influences if the rocket will land successfully?
  - The effect each relationship with certain rocket variables will impact in determining the success rate of a successful landing.
  - What conditions does SpaceX have to achieve to get the best results and ensure the best rocket success landing rate.

Section 1

Methodology

Methodology

- Data collection methodology:
  - SpaceX Rest API
  - (Web Scrapping) from Wikipedia
- Performed data wrangling (Transforming data for Machine Learning)

  One Hot Encoding data fields for Machine Learning and dropping irrelevant columns
- Performed exploratory data analysis (EDA) using visualization and SQL - Plotting :

  Scatter Graphs, Bar Graphs to show relationships between variables to show patterns of data.
- Performed interactive visual analytics using Folium and Plotly Dash
- Performed predictive analysis using classification models

  How to build, tune, evaluate classification models

Collection

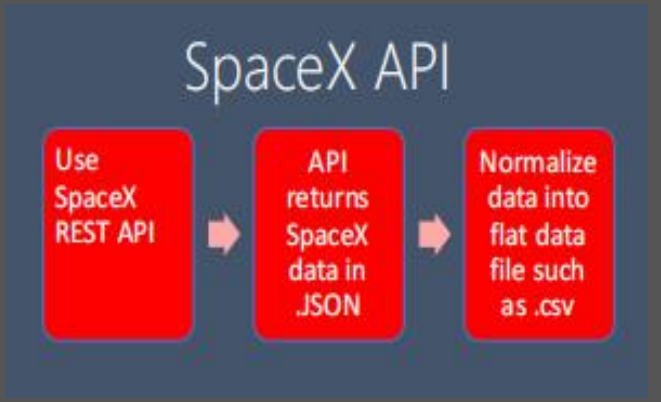
Data Collection

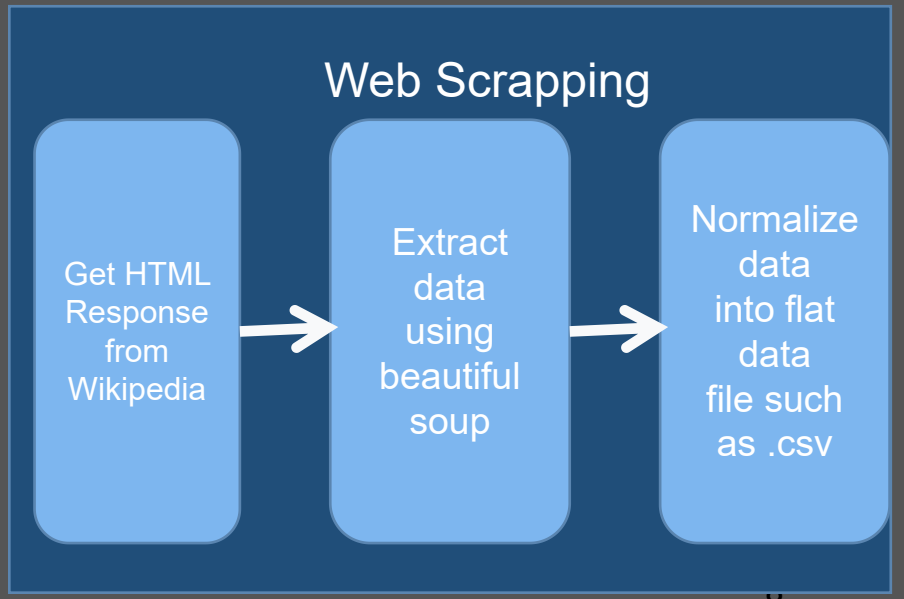
The following datasets was collected by Using Spacex REST API and Wikipedia

- We worked with SpaceX launch data that is gathered from the SpaceX REST API.
- This API will give us data about launches, including information about the rocket used,

payload delivered, launch specifications, landing specifications, and landing outcome.

- Our goal is to use this data to predict whether SpaceX will attempt to land a rocket or not.
- The SpaceX REST API endpoints, or URL, starts with api.spacexdata.com/v4/.
- Another popular data source for obtaining Falcon 9 Launch data is web scraping Wikipedia using BeautifulSoup.





Data Collection – SpaceX API

Use SpaceX Restapi

Filter Data Frame for Falcon9 only and Clean data

Export Data into flat CSV

```
In [48]: spacex_url="https://api.spacexdata.com/v4/launches/past"

In [49]: response = requests.get(spacex_url)
```

Check the content of the response

```
In [50]: print(response.content)
```

```
b'[{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},"links":{"patch":{"small":"https://images2.imgbox
om/3c/0e/T8iJcSN3_o.png","large":"https://images2.imgbox.com/40/e3/GypSkayF_o.png"},"reddit":{"campaign":null,"launch":null,"media":nul
```

Task 1: Request and parse the SpaceX launch data using the GET request

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

```
In [51]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_a
pi.json'
```

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

```
In [52]: response.status_code
Out[52]: 200
```

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

```
In [53]: # Use json_normalize meethod to convert the json result into a dataframe
data=pd.json_normalize(response.json())
```

Using the dataframe `data` print the first 5 rows

```
In [54]: # Get the head of the dataframe
data.head()
```

Out[54]:

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships	capsules	payloads
	2006-03						[{'time': 33, 'altitude': None,	Engine failure at				

Use SpaceX Restapi

Filter Data Frame for Falcon9 only and Clean data

Export Data into flat CSV

```
In [77]: # Hint data['BoosterVersion']!='Falcon 1'
df1.drop(df1[df1['BoosterVersion'] !='Falcon 9'].index,inplace= True)
data_falcon9=df1
data_falcon9.head()
```

Out[77]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Seria
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B000
5	8	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B000
6	10	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B000
7	11	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None	1.0	0	B100
8	12	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B100

Now that we have removed some values we should reset the FlgihtNumber column

Task 3: Dealing with Missing Values

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace `np.nan` values in the data with the mean you calculated.

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

```
[96]: 4      6123.547647
```


```
Latitude            0
dtype: int64
```

```
In [99]: data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Now we should have no missing values in our dataset except for in `LandingPad`.

GitHub:Data_Restapi

Data Collection - Scraping

Get HTML Response from Wikipedia

Extract data using Beautiful soup

Parse HTML into a List or Dictionary

In [4]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

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

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

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

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

Create a `BeautifulSoup` object from the HTML `response`

In [6]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(page.content, "html.parser")
```

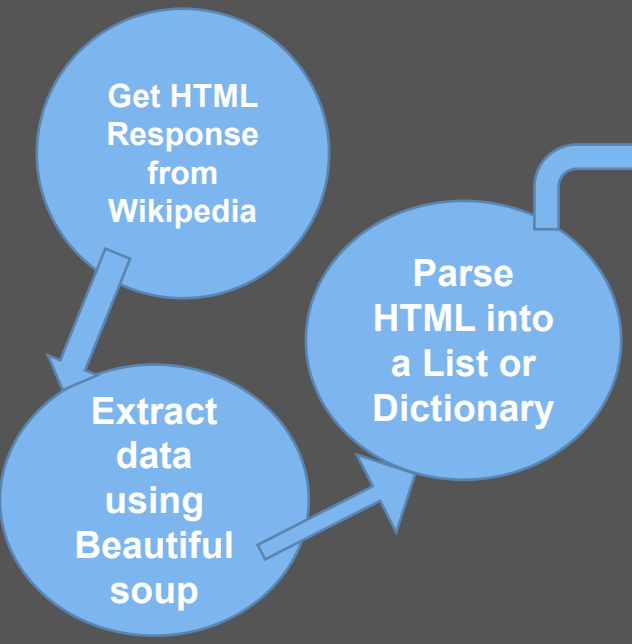
Print the page title to verify if the `BeautifulSoup` object was created properly

In [7]:
```
# Use soup.title attribute
soup.title
```

Out[7]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

In [8]:
```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
print('Classes of each table:')
for table in soup.find_all('table'):
    print(table.get('class'))
# Creating list with all tables
tables = soup.find_all('table')
```

```
Classes of each table:
```




```
In [144]:
column_names = []
th_tags =first_launch_table.find_all('th')

if (th_tags is not None and len(th_tags) > 0):
    for th in th_tags:
        column_names.append(th.get_text())

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_he
# Append the Non-empty column name (`if name is not None and len(name)
column_names
```

```
Out[144]:
['Flight No.\n',
 'Date andtime (UTC)\n',
 'Version,Booster [b]\n',
 'Launch site\n',
 'Payload[c]\n',
 'Payload mass\n',
 'Orbit\n'
```

TASK 3: Create a data frame by parsing the launch HTML tables

We will create an empty dictionary with keys from the extracted column names in the previous task. Later, this dictionary will be converted into a Pandas dataframe

```
In [146]:
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date andtime (UTC)\n']

# 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']=[]
```

```
In [148]:
df=pd.DataFrame(launch_dict)
```

We can now export it to a **CSV** for the next section, but to make the answers consistent and in case you have difficulties finishing this lab.

Following labs will be using a provided dataset to make each lab independent.

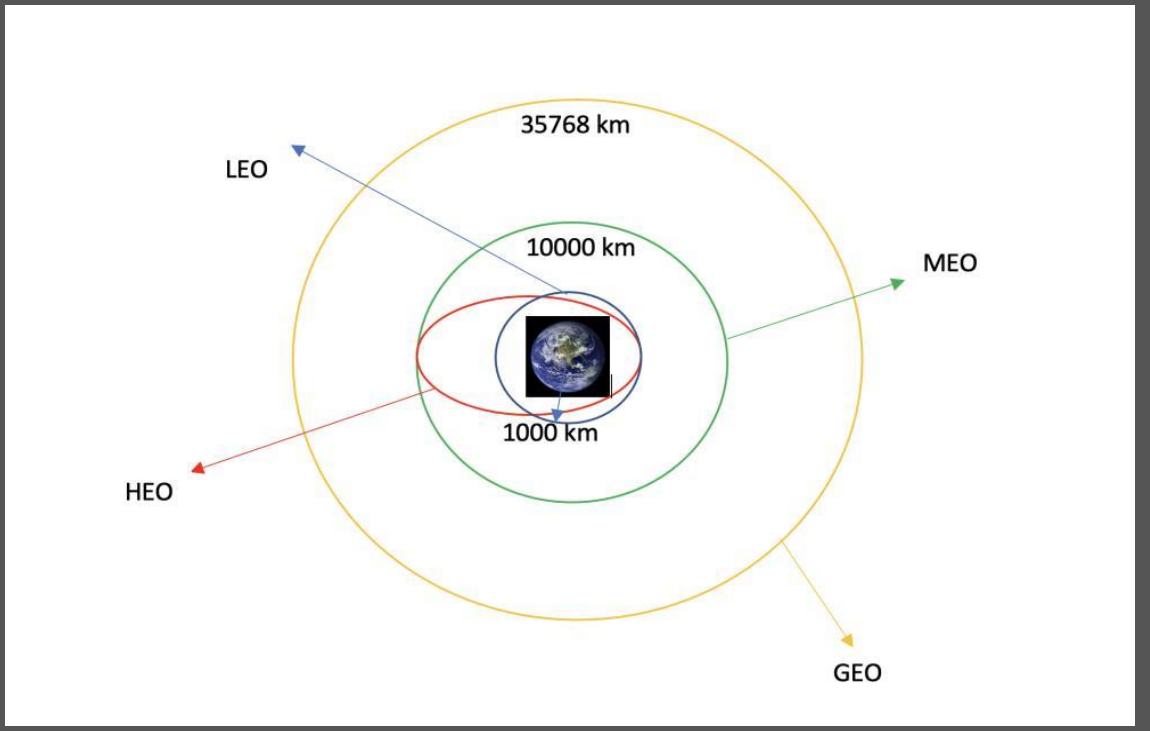
```
df.to_csv('spacex_web_scraped.csv', index=False)
```

GitHub:Webscrape

Data Wrangling

In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident; for example, True Ocean means the mission outcome was successfully landed to a specific region of the ocean while False Ocean means the mission outcome was unsuccessfully landed to a specific region of the ocean. True RTLS means the mission outcome was successfully landed to a ground pad False RTLS means the mission outcome was unsuccessfully landed to a ground pad.

True ASDS means the mission outcome was successfully landed on a drone ship False ASDS means the mission outcome was unsuccessfully landed on a drone ship.
We mainly convert those outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful.



- Perform Exploratory Data Analysis EDA on dataset

Calculate the number of launches at each site

Calculate the number and occurrence of each orbit

Git Hub: Data Wrangling

Calculate the number and occurrence of mission outcome per orbit type

Export dataset as .CSV

Create a landing outcome label from Outcome column

Work out success rate for every landing in dataset

Calculate the number of launches at each site

Use the method value_counts() on the column LaunchSite to determine the number of launches on each site:

```
In [5]: # Apply value_counts() on column LaunchSite
        df['LaunchSite'].value_counts()
```

```
Out[5]: CCAFS SLC 40    55
        KSC LC 39A      22
        VAFB SLC 4E     13
        Name: LaunchSite, dtype: int64
```

Calculate the number and occurrence of each orbit

TASK 2: Calculate the number and occurrence of each orbit

Use the method .value_counts() to determine the number and occurrence of each orbit in the column Orbit

```
In [6]: # Apply value_counts on Orbit column
        df['Orbit'].value_counts()
```

```
Out[6]: GTO      27
        ISS      21
        VLEO     14
        PO        9
        LEO       7
        SSO       5
        MEO       3
        HEO       1
        GEO       1
        SO        1
        ES-L1     1
        Name: Orbit, dtype: int64
```

Calculate the number and occurrence of mission outcome per orbit type

TASK 3: Calculate the number and occurence of mission outcome per orbit type

Use the method .value_counts() on the column Outcome to determine the number of landing_outcomes.Then assign it to a variable landing_outcomes.

```
In [7]: # landing_outcomes = values on Outcome column
        landing_outcomes=df['Outcome'].value_counts()
        landing_outcomes
```

Export dataset as .CSV

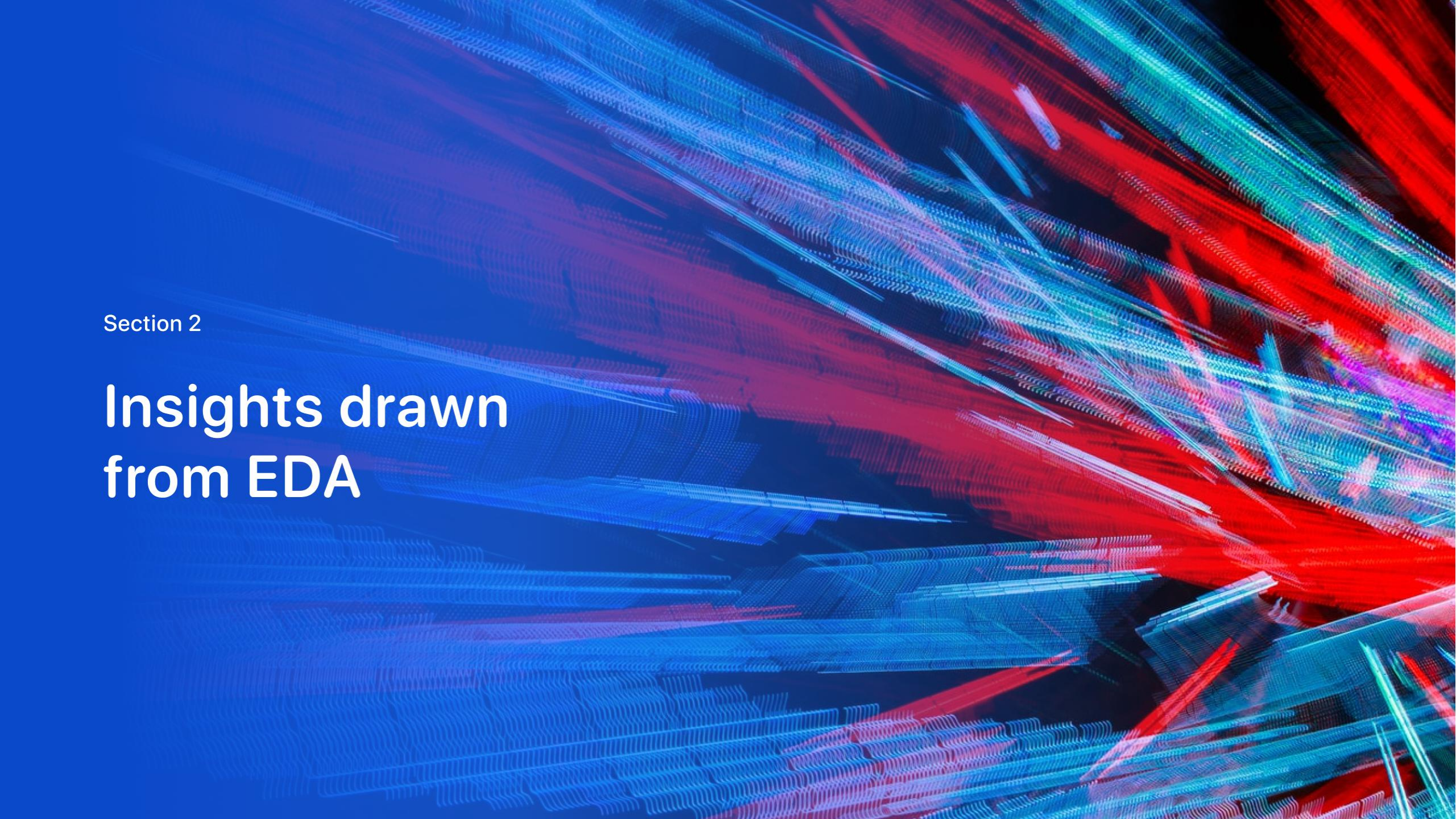
```
In [27]: df.to_csv("dataset_part_2.csv", index=False)
```

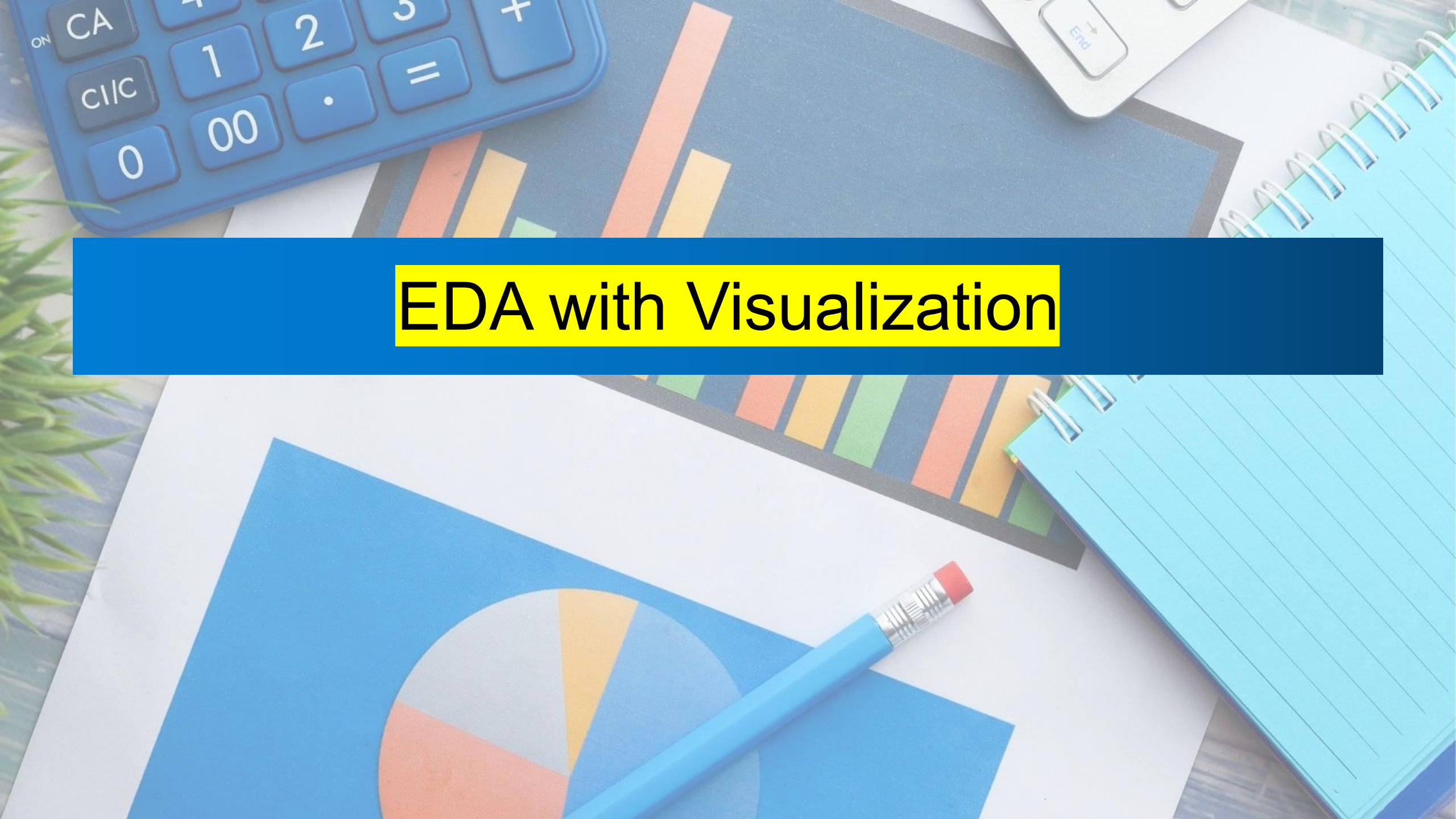
Create a landing outcome label from Outcome column

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

Section 2

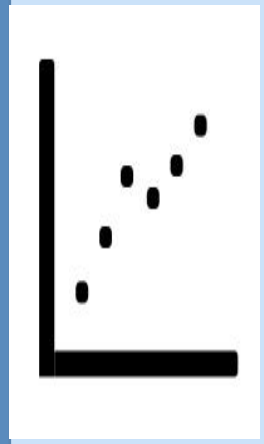
Insights drawn from EDA




EDA with Visualization

Scatter Graphs being drawn:

- 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

Scatter plots show how much one variable is affected by another. The relationship between two variables is called their correlation . Scatter plots usually consist of a large body of data.

Bar Graph being drawn:

- Mean VS. Orbit

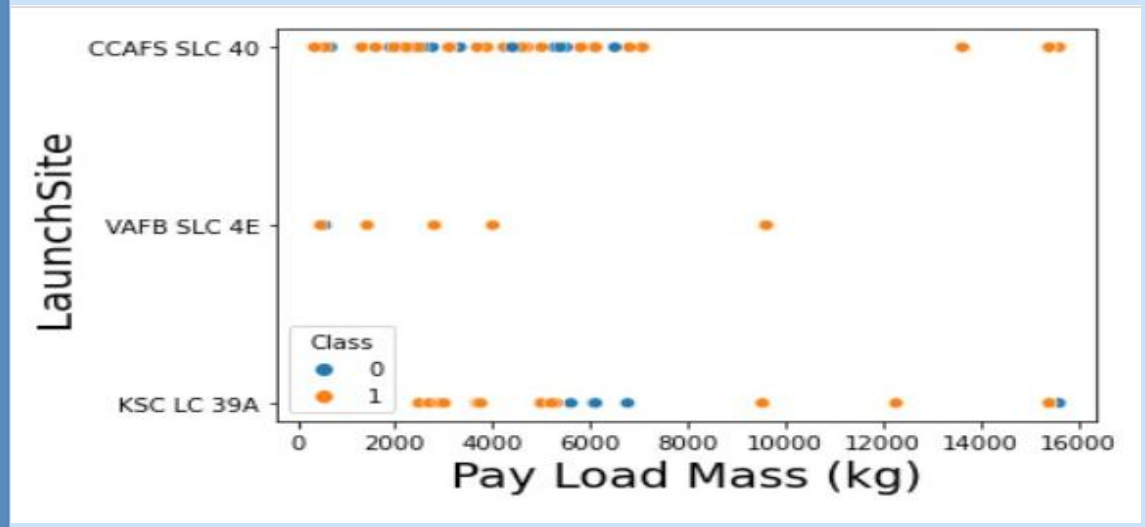
A bar diagram makes it easy to compare sets of data between different groups at aglance.The graph represents categories on one axis and a discrete value in the other. The goal is to show the relationship between the two axes. Bar charts can also show big changes in data over time.

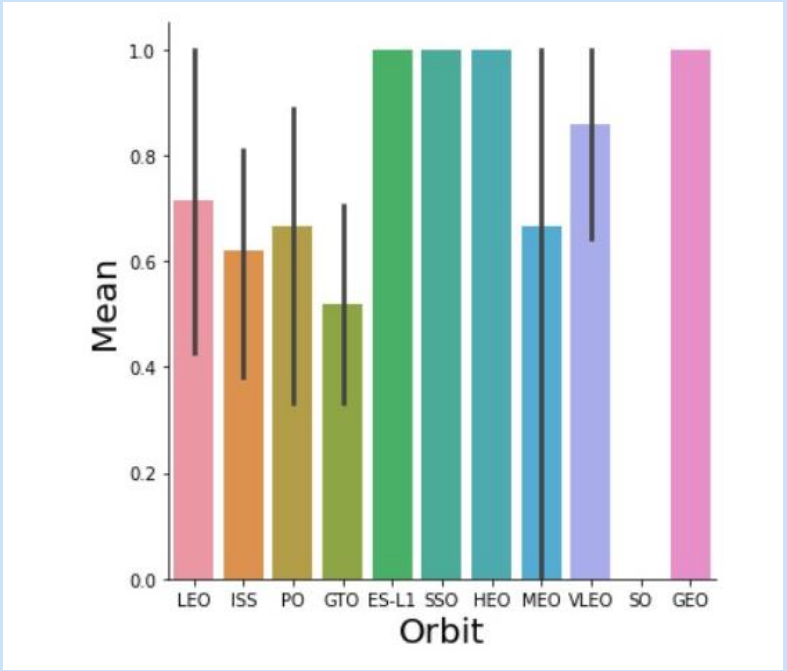
Line Graph being drawn:

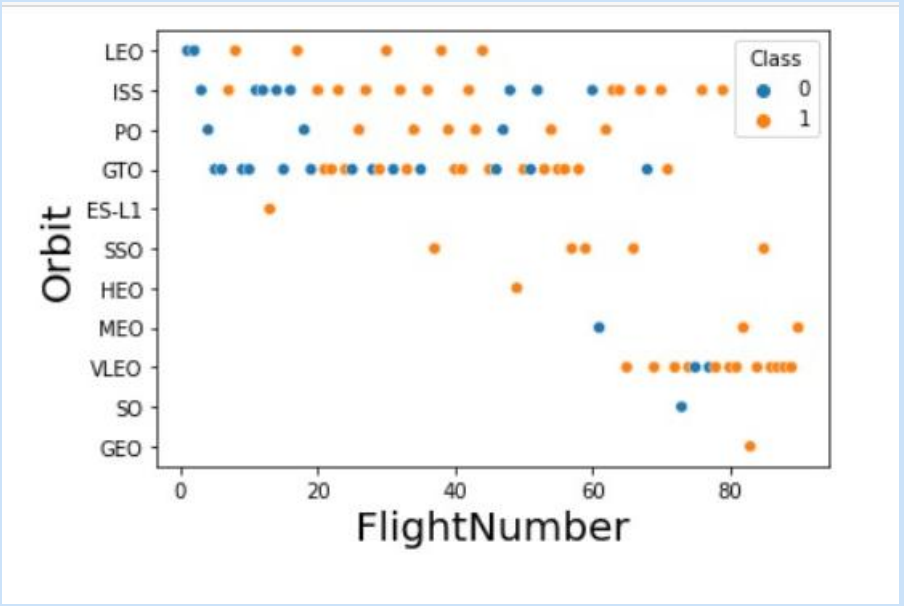
- Success Rate VS. Year

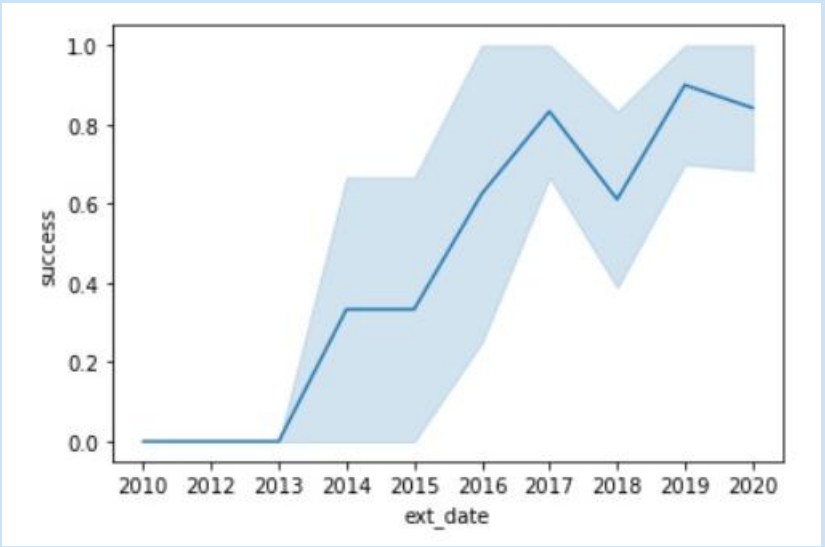
Line graphs are useful in that they show data variables and trends very clearly and can help to make predictions about the results of data not yet recorded

GitHub:EDA_Visualiztion

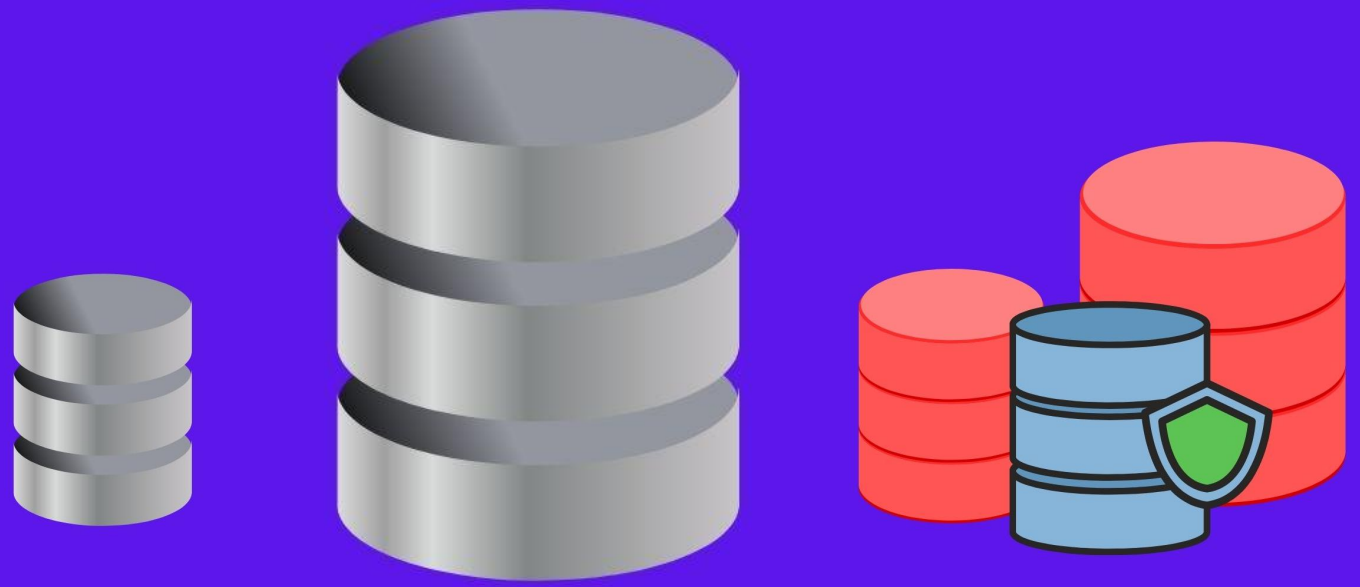

CCAFS SLC 40
VAFB SLC 4E
KSC LC 39A
LaunchSite
Class
0
1
0
2000
4000
6000
8000
10000
12000
14000
16000
Pay Load Mass (kg)


1.0
0.8
0.6
0.4
0.2
0.0
Mean
LEO
ISS
PO
GTO
ES-L1
SSO
HEO
MEO
VLEO
SO
GEO
Orbit


LEO
ISS
PO
GTO
ES-L1
SSO
HEO
MEO
VLEO
SO
GEO
Orbit
Class
0
1
0
20
40
60
80
FlightNumber


1.0
0.8
0.6
0.4
0.2
0.0
success
2010
2012
2013
2014
2015
2016
2017
2018
2019
2020
ext_date

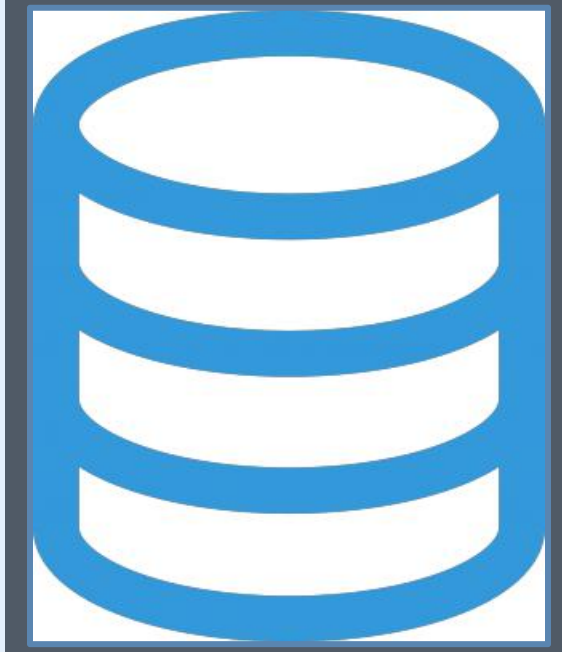
EDA with SQL



Performed SQL queries to gather information about the dataset.
For example of some questions we were asked about the data we needed information about.

Which we are using SQL queries to get the answers in the dataset :

1. Display the names of the unique launch sites in the space mission.
2. Display 5 records where launch sites begin with the string 'CCA'.
3. Display the total payload mass carried by boosters launched by NASA (CRS)
4. Display average payload mass carried by booster version F9 v1.1
5. List the date when the first successful landing outcome in ground pad was acheived.
6. List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
7. List the total number of successful and failure mission outcomes.
8. List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
9. List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
10. Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order



Github: EDA SQL


Build an Interactive Map with Folium

Building an interactive map with Folium

To visualize the Launch Data into an interactive map.

We took the Latitude and Longitude Coordinates at each launch site and added a Circle Marker around each launch site with a label of the name of the launch site.

We assigned the dataframe launch_outcomes (failures, successes) to classes 0 and 1 with Green and Red markers on the map in a MarkerCluster()

Using Haversine's formula we calculated the distance from the Launch Site to various landmarks to find various trends about what is around the Launch Site to measure patterns. Lines are drawn on the map to measure distance to landmarks

Example of some trends in which the Launch Site is situated in.

- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

GitHub:Visualization_Folium

Built an interactive dashboard with Plotly and Dash

Used Python Anywhere to host the website live 24/7 so you can play around with the data and view the data

- The dashboard is built with Plotly and Dash web framework.

Graphs

- **Pie Chart showing the total launches by a certain site/all**
- Display relative proportions of multiple classes of data.
- Size of the circle can be made proportional to the total quantity it represents.

Scatter Graph showing the relationship with Outcome and Payload Mass (Kg) for the different Booster

- It shows the relationship between two variables.
- It is the best method to show you a non-linear pattern.
- The range of data flow, i.e. maximum and minimum value, can be determined.
- Observation and reading are straightforward.

Github_Dash_plotly

URL:for spacex_dash_app

Predictive Analysis

Building Model

- Load our dataset into NumPy and Pandas
- Transform Data
- Split our data into training and test data sets
- Check how many test samples we have
- Decide which type of machine learning algorithms we want to use
- Set our parameters and algorithms to GridSearchCV
- Fit our datasets into the GridSearchCV objects and train our dataset.

Evaluating Model

- Check accuracy for each model
- Get tuned hyperparameters for each type of algorithms
- Plot Confusion Matrix

Improving Model

- Feature Engineering
- Algorithm Tuning

Github: Classification

Finding The Best Performing Classification Model:

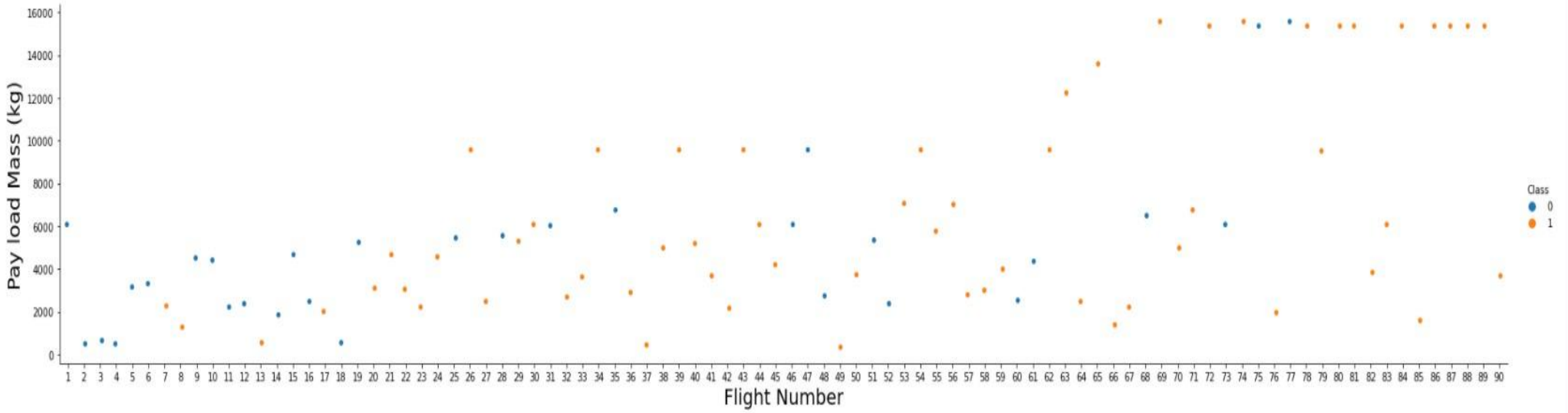
- The model with the best accuracy score wins the best performing model
- In the notebook there is a dictionary of algorithms with scores at the bottom of the notebook

Results

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

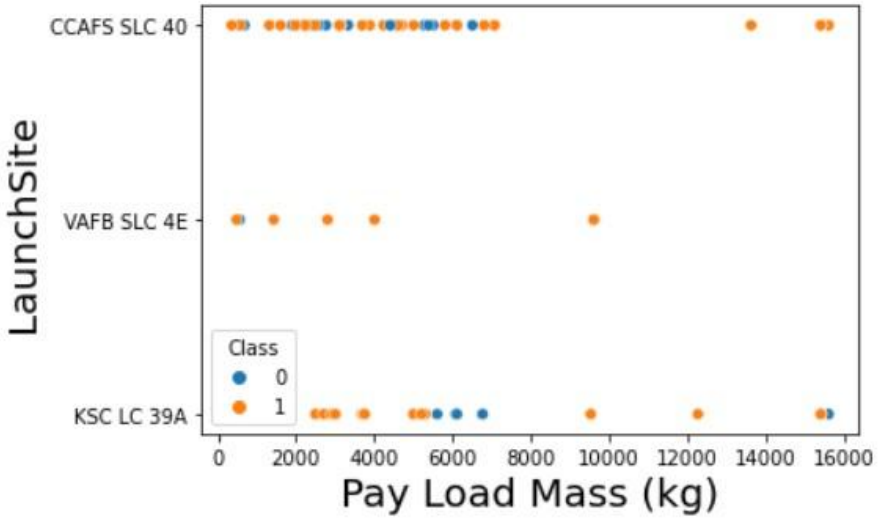
Flight Number Vs Payload Mass

In [3]:
```
sns.catplot(y="PayloadMass", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```



Pay load Mass Vs Launch site

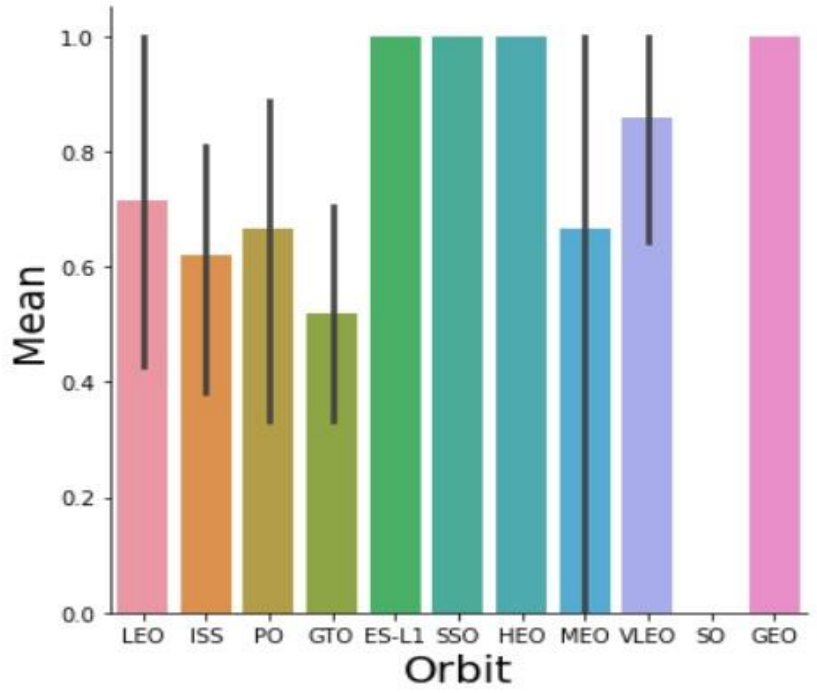
```
In [5]: # Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class value
sns.scatterplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df);
plt.xlabel("Pay Load Mass (kg)",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.show()
```



- The more amount of flights at a launch site CCAFS SLC 40 the greater the success rate at a launch site.
- More success was noticed with pay load mass less than 8000(kg)

Orbit and Mean (Payload Mass)

```
In [8]: sns.catplot(x="Orbit",y="Class", kind="bar",data=df)
        plt.xlabel("Orbit",fontsize=20)
        plt.ylabel("Mean",fontsize=20)
        plt.show()
```

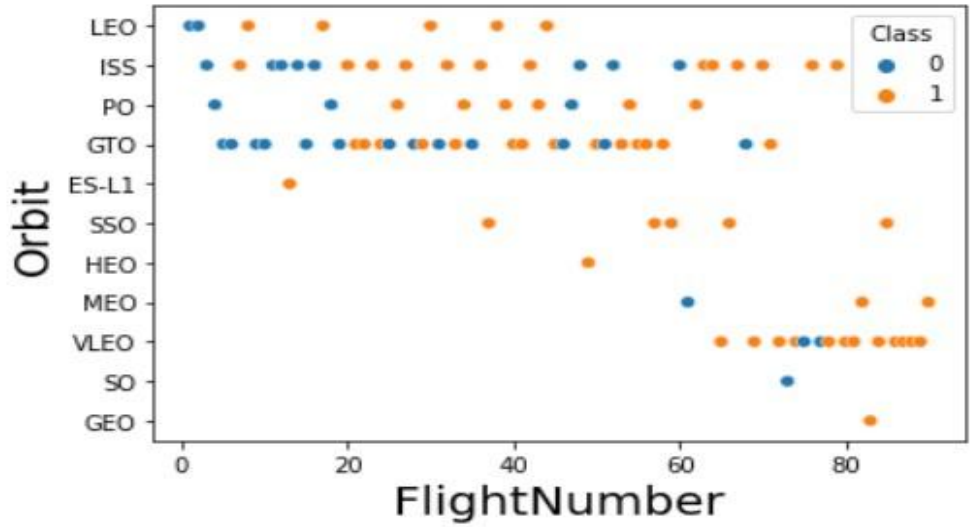


Orbit GEO,HEO,SSO,ES-L1 has the best Success Rate

Flight Number Vs Orbit

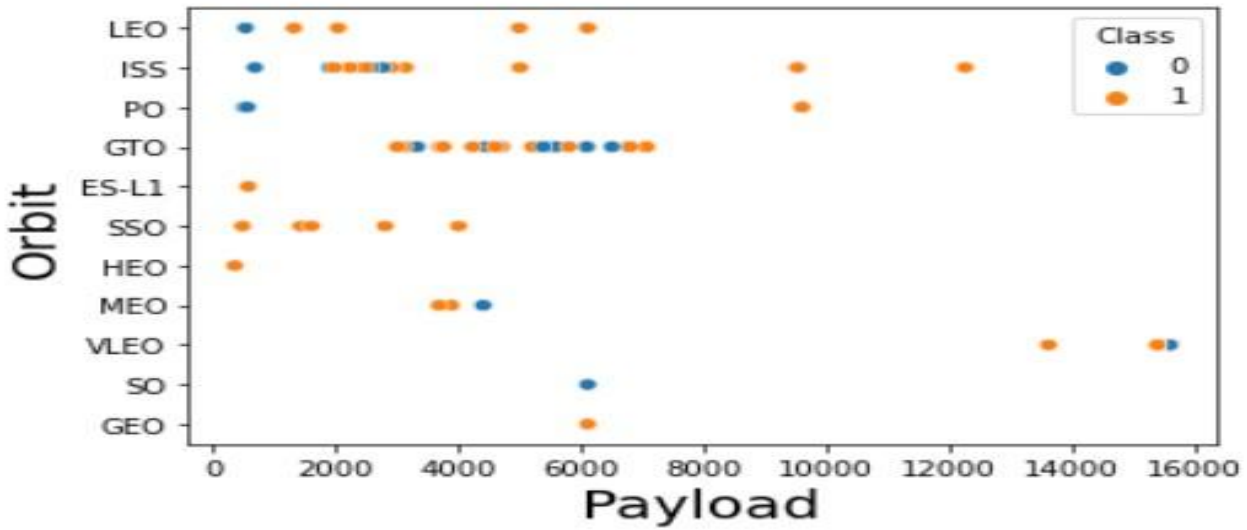
You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

```
:  # Plot a scatter point chart with x axis to be FlightNumber and y axis
   sns.scatterplot(y="Orbit", x="FlightNumber", hue="Class", data=df);
   plt.xlabel("FlightNumber",fontsize=20)
   plt.ylabel("Orbit",fontsize=20)
   plt.show()
```



Pay load Mass Vs Orbit

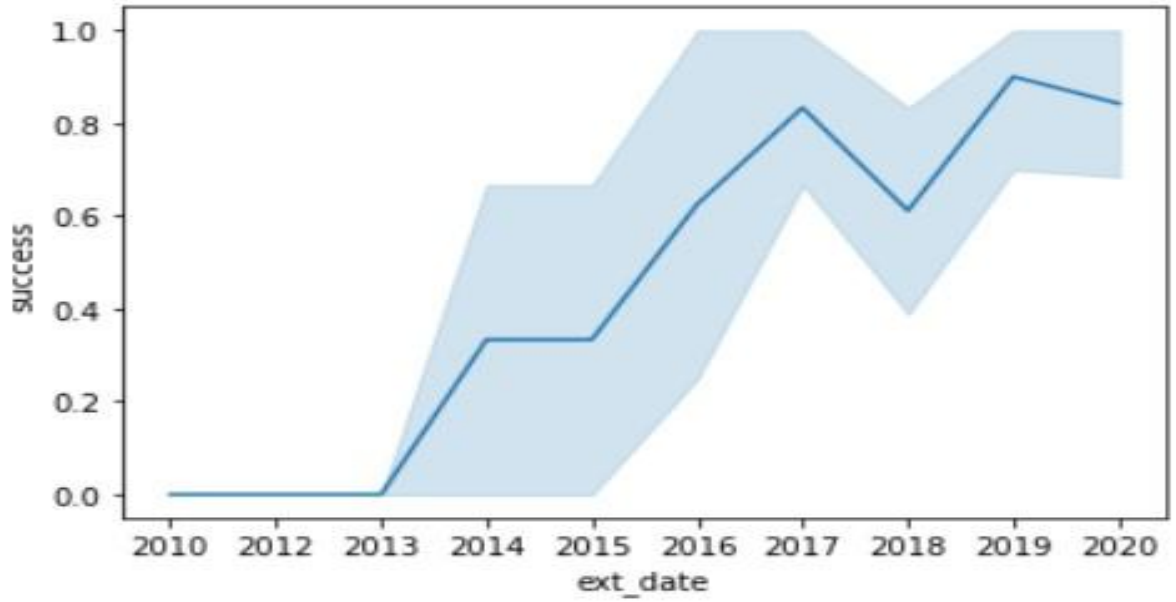
```
# Plot a scatter point chart with x axis to be Payload and y axis to

sns.scatterplot(y="Orbit", x="PayloadMass", hue="Class", data=df);
plt.xlabel("Payload",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



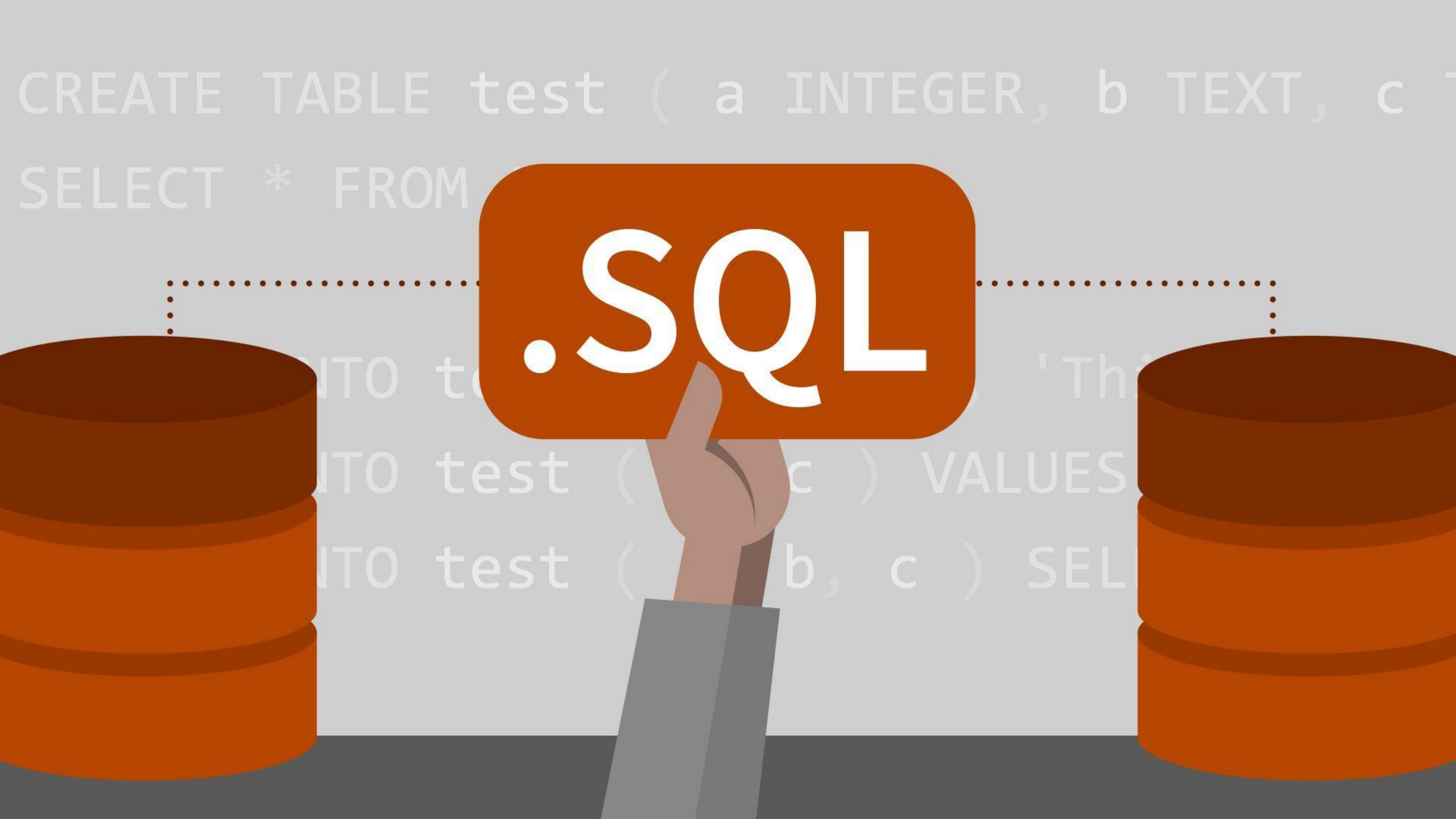
You should observe that Heavy payloads have a negative influence on GTO orbits and positive on GTO and Polar LEO (ISS) orbits.

Launch Success rate Vs Launch Year

```
df.columns
df1=[]
df1=pd.DataFrame(df1)
df1['ext_date']=Extract_year(df['Date'])
df1['success']=df['Class']
sns.lineplot(data=df1,x=df1['ext_date'],y=df1['success'])
plt.show()
```



you can observe that the success rate since 2013 kept increasing till 2020


CREATE TABLE test (a INTEGER, b TEXT, c
SELECT * FROM
.SQL

SQL QUERY

All Launch Site Names

```
select DISTINCT
Launch_Site
from SPACEXTBL
```

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

QUERY EXPLANATION

Using the word DISTINCT in the query means that it will only show Unique values in the Launch_Site column from SPACEXTBL

Launch Site Names Begin with 'CCA'

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

time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
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)
07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

QUERY EXPLANATION

Using the word limit 5 in the query means that it will only show 5 records from tblSpaceX and LIKE keyword has a wild card with the words ‘CCA%’ the percentage in the end suggests that the Launch_Site name must start with CCA

Total Payload Mass

```
%sql select sum(payload_mass__kg_) from SPACEXTBL where customer like 'NASA (CRS)';
```

QUERY EXPLANATION

Using the function SUM summates the total in the column PAYLOAD_MASS_KG_ The WHERE clause filters the dataset to only perform calculations on Customer NASA (CRS)

Average Payload Mass by F9 v1.1

```
%sql SELECT AVG(payload_mass__kg_) FROM SPACEXTBL
where booster_version = 'F9 v1.1' ;
```

QUERY EXPLANATION
Using the function AVG works out the average in the column
payload_mass_kg_
The WHERE clause filters the dataset to only perform calculations on booster_version F9 v1.1

First Successful Ground Landing Date

%sql SELECT MIN(DATE) FROM SPACEXTBL WHERE landing__outcome='Success (ground pad)';

QUERY EXPLANATION
Using the function MIN works out the minimum date in the column Date
The WHERE clause filters the dataset to only perform calculations on Landing_Outcome Success (drone ship)

Successful Drone Ship Landing with Payload between 4000 and 6000

%sql select booster_version from SPACEXTBL where landing__outcome like 'Success (drone ship)' and payload_mass__kg_ BETWEEN 4000 and 6000;

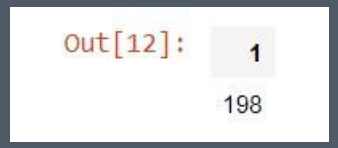
Out[96]:

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

QUERY EXPLANATION
Selecting only Booster_Version
The WHERE clause filters the dataset to Landing_Outcome = Success (drone ship)
The AND clause specifies additional filter conditions
Payload_MASS_KG_ > 4000 AND Payload_MASS_KG_ < 6000

Total Number of Successful and Failure Mission Outcomes

%sql select count (mission_outcome) from SPACEXTBL where mission_outcome like 'Success' or mission_outcome like 'Failure';



QUERY EXPLANATION

Using the function count sums the number of outcomes with “Success” or “Failure “ as result in mission_outcome column

Boosters Carried Maximum Payload

```
%sql SELECT booster_version FROM SPACEXTBL WHERE
payload_mass__kg_ = (select max(payload_mass__kg_) from
SPACEXTBL);
```

Out[13]:

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
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

QUERY EXPLANATION

Selecting only booster_Version from SPACEXTBL where payload_mass_kg_ is maximum

2015 Launch Records

```
%sql select launch_site,booster_version,landing__outcome from SPACEXTBL
where landing__outcome like 'Failure (drone ship)'and DATE like '2015%';
```

Out[14]:

launch_site	booster_version	landing__outcome
CCAFS LC-40	F9 v1.1 B1012	Failure (drone ship)
CCAFS LC-40	F9 v1.1 B1015	Failure (drone ship)
CCAFS LC-40	F9 v1.1 B1012	Failure (drone ship)
CCAFS LC-40	F9 v1.1 B1015	Failure (drone ship)

Selecting only booster_Version, landing_outcome and launch_site from SPACEXTBL where clause filters landing_outcome as ‘Failure(drone ship) and Date (year) as 2015

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

```
%sql select landing__outcome, Date from SPACEXTBL where
landing__outcome like 'Failure (drone ship)'or'Success (ground
pad)' in (select landing__outcome from SPACEXTBL where DATE
between '2010-06-04'  and '2017-03-20' )order by DATE DESC;
```

Out[15]:

landing__outcome	DATE
Success	2020-12-06
Success	2020-12-06
Success	2020-11-25
Success	2020-11-25
Success	2020-11-21
Success	2020-11-21
Success	2020-11-16
Success	2020-11-16
Success	2020-11-05
Success	2020-11-05

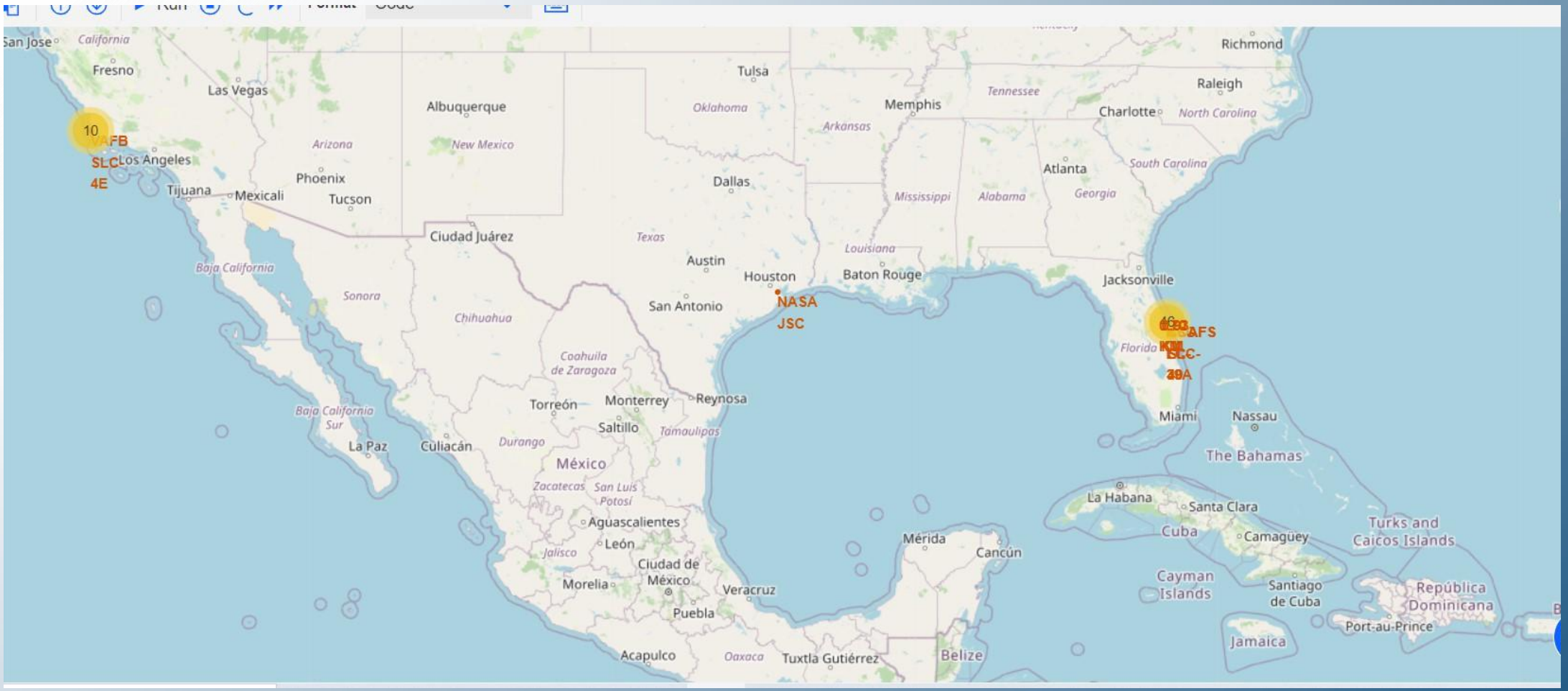
Selecting only landing_outcome and Date from SPACEXTBL where clause filters landing_outcome as ‘Failure(drone ship) or Success(ground pad) and Date between '2010-06-04' and '2017-03-20')order by DATE DESC

Section 4

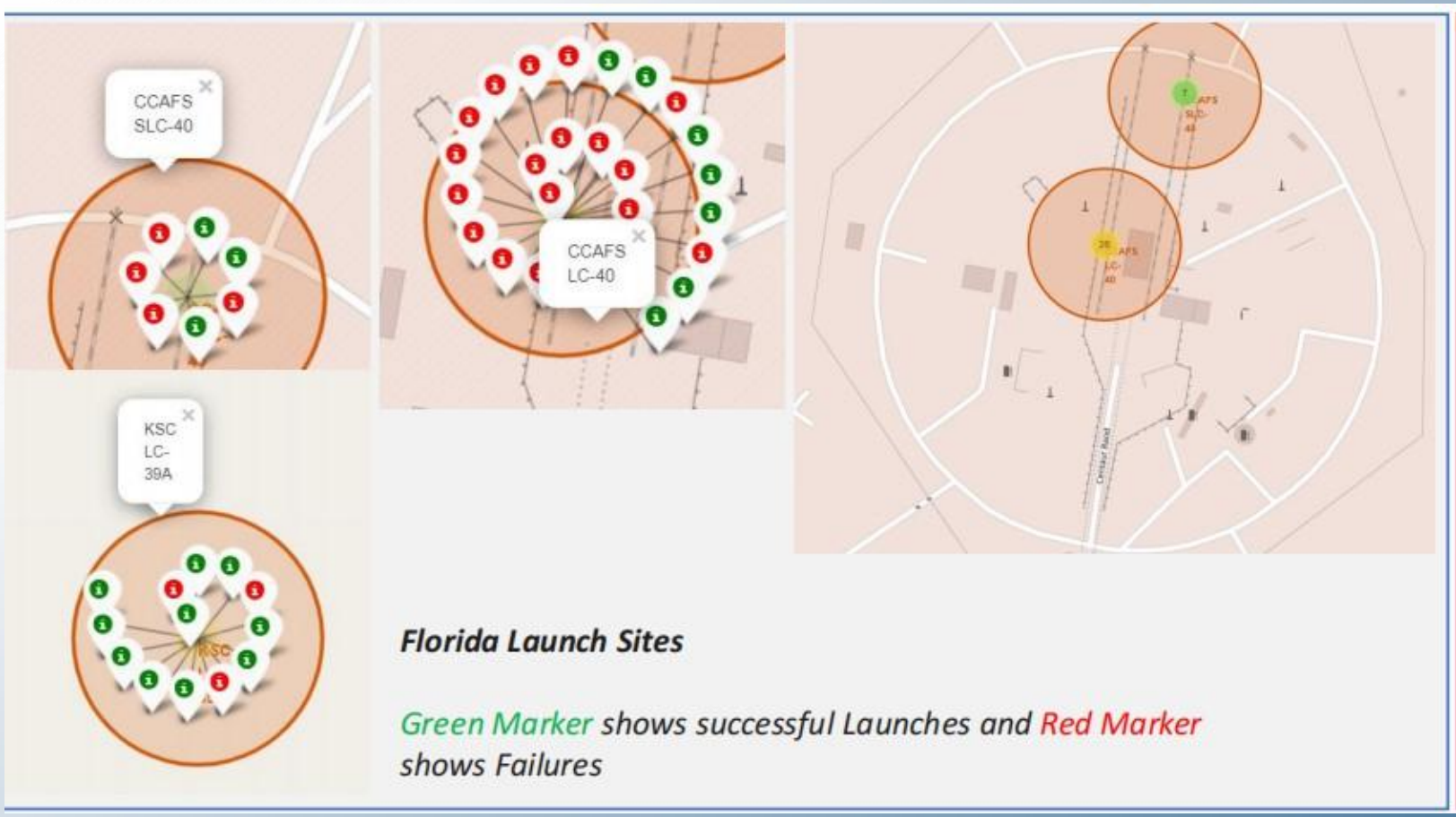
Launch Sites Proximities Analysis

Interactive map with Folium

All launch sites global map markers



Colour Labelled Markers

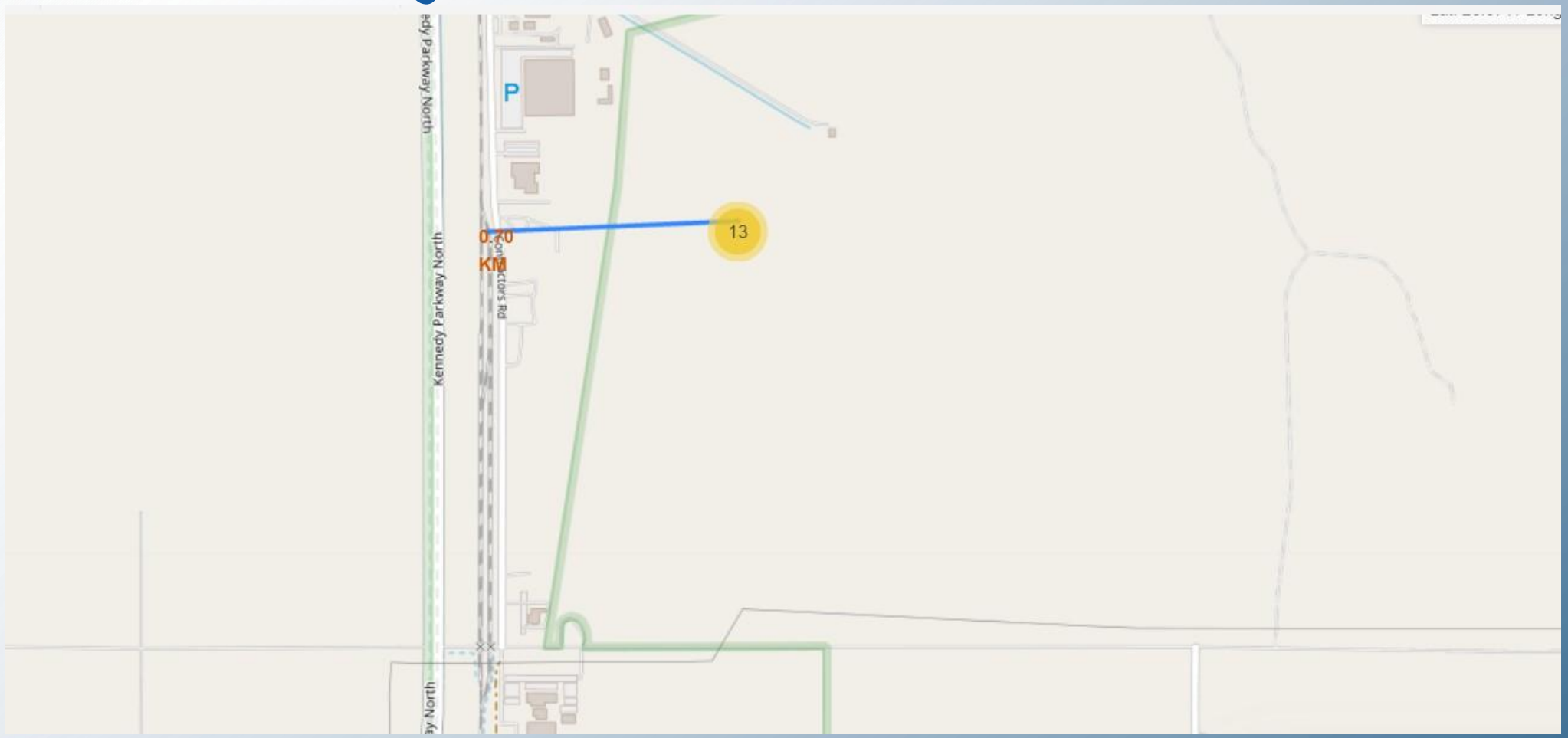


Florida Launch Sites

Green Marker shows successful Launches and Red Marker shows Failures

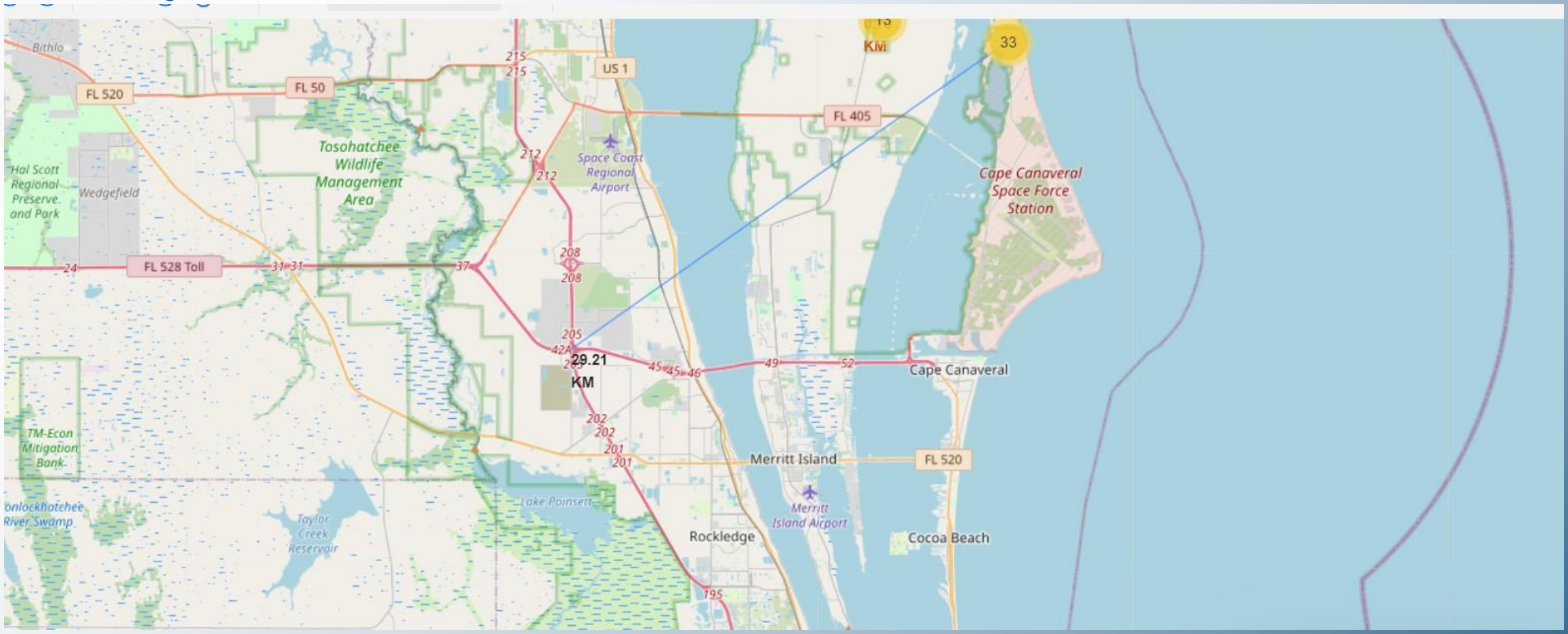
California Launch Site

Working out Launch Sites distance to landmarks to find trends with Haversine formula using KSCLC-39A as a reference



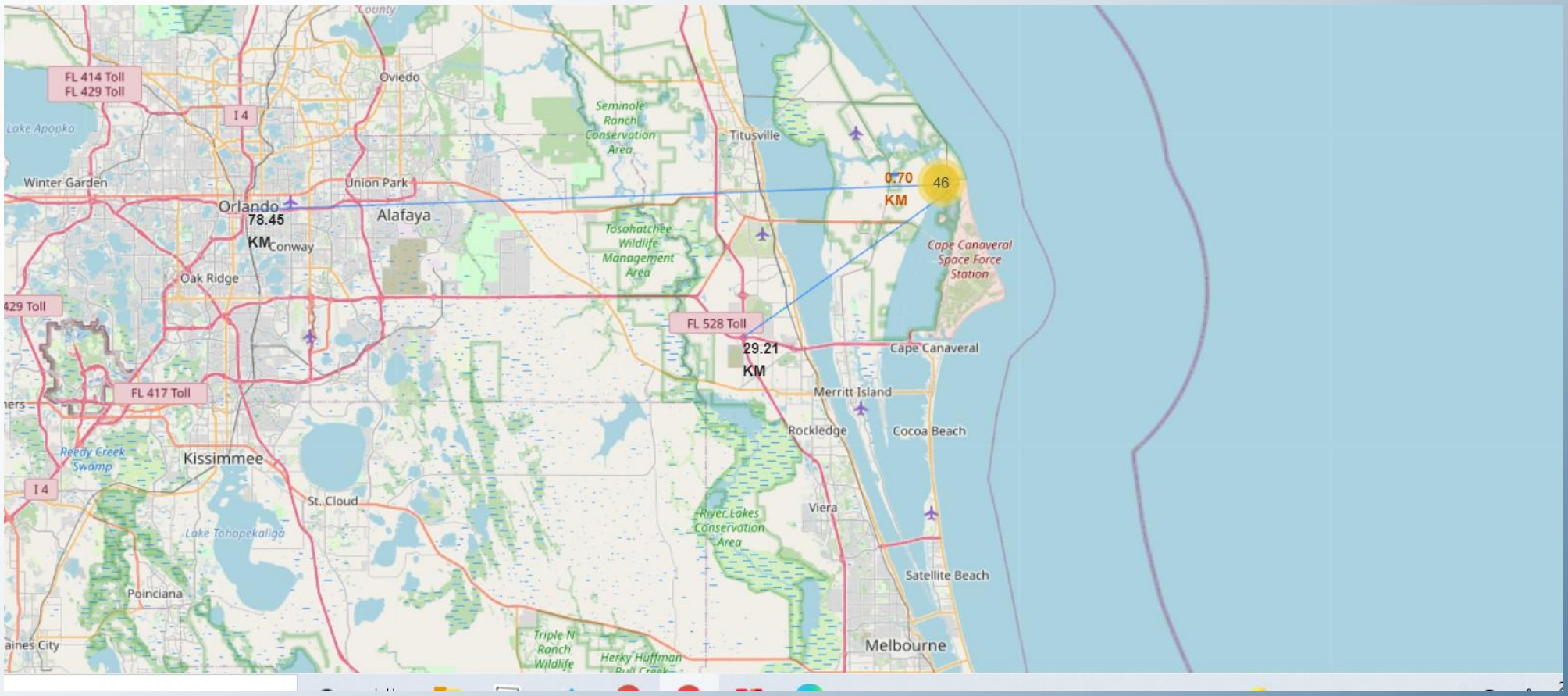
Nearest Railway Point to Florida Lauch site is 0.7 km

Working out Launch Sites distance to landmarks to find trends with Haversine formula using KSCLC-39A as a reference



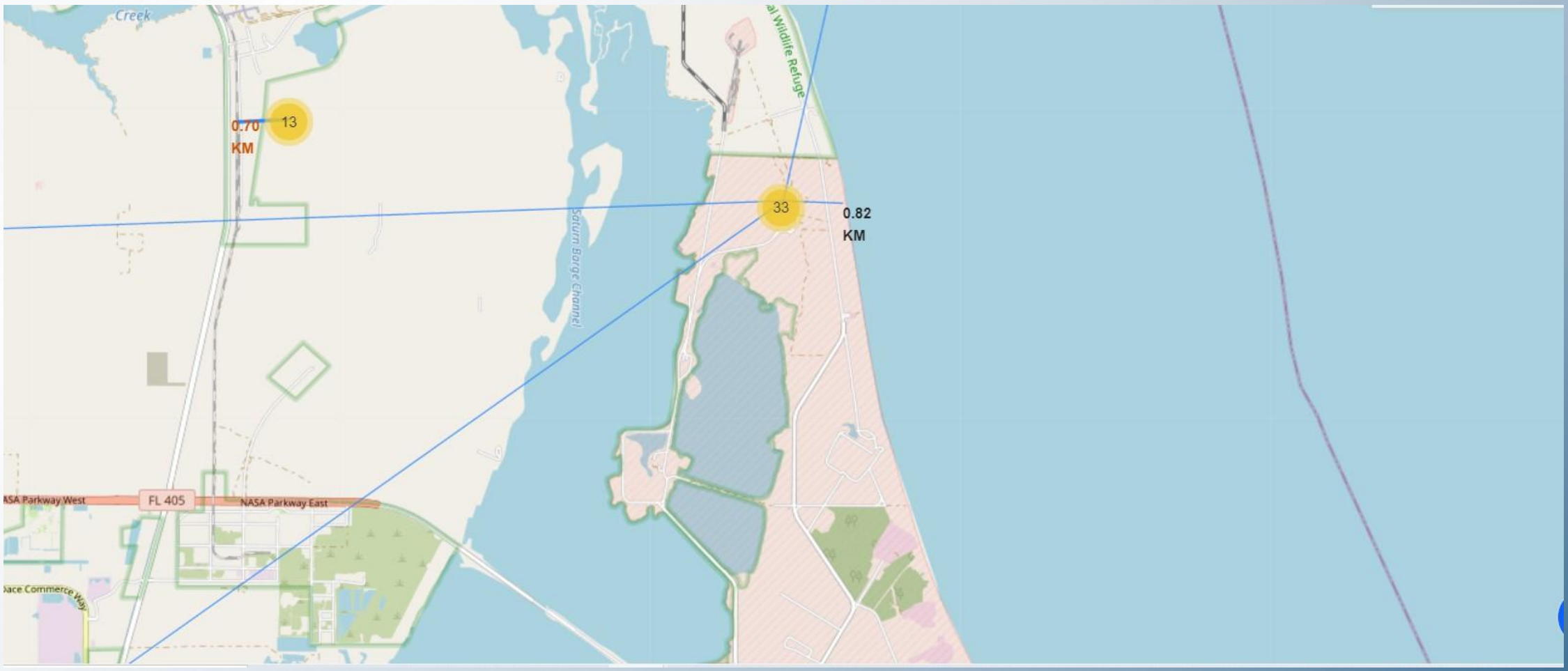
Nearest Highway Point to Florida Lauch site is 29.21 km

Working out Launch Sites distance to landmarks to find trends with Haversine formula using KSCLC-39A as a reference

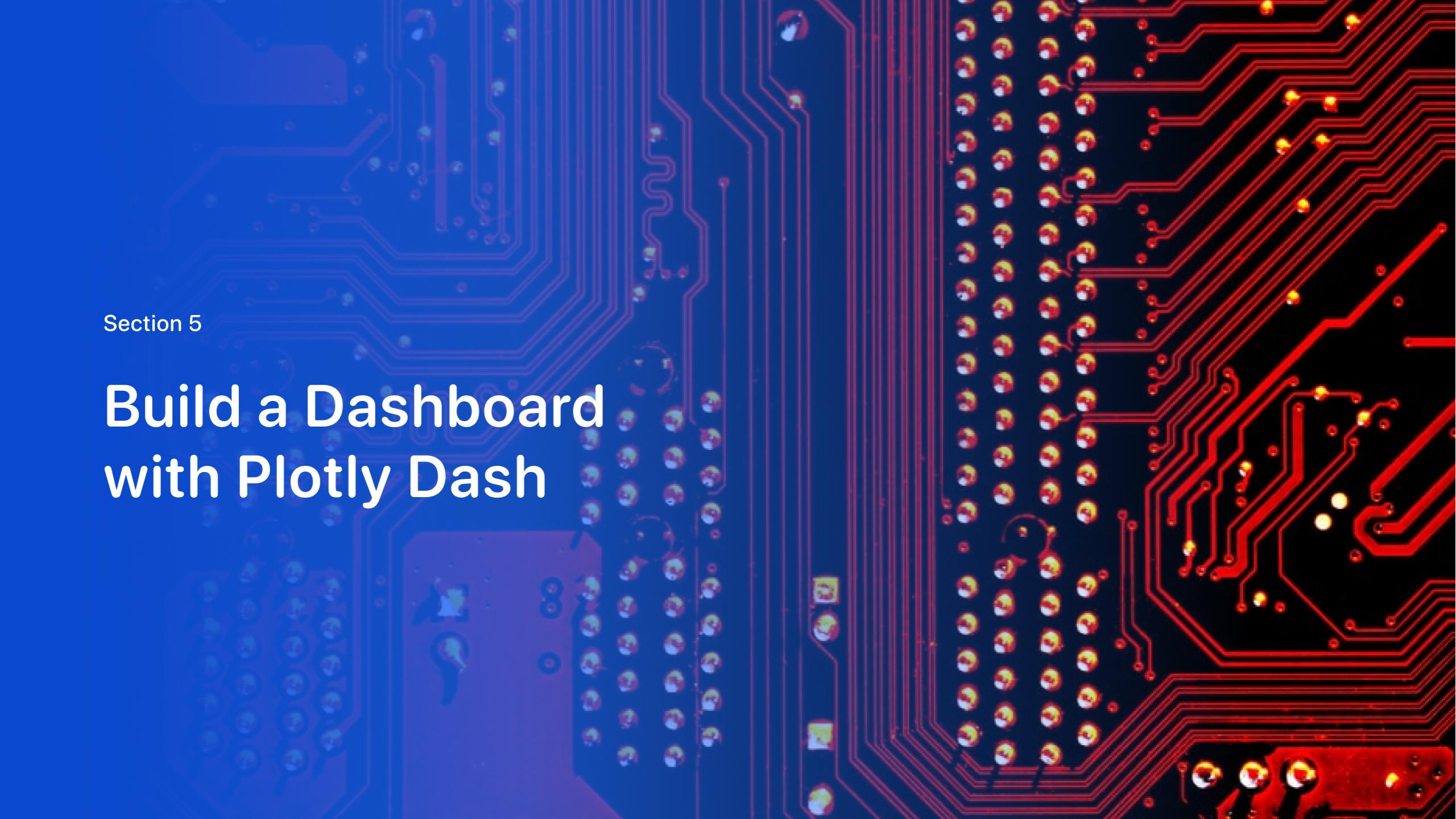


Nearest City to Florida Lauch site is 78.45 km

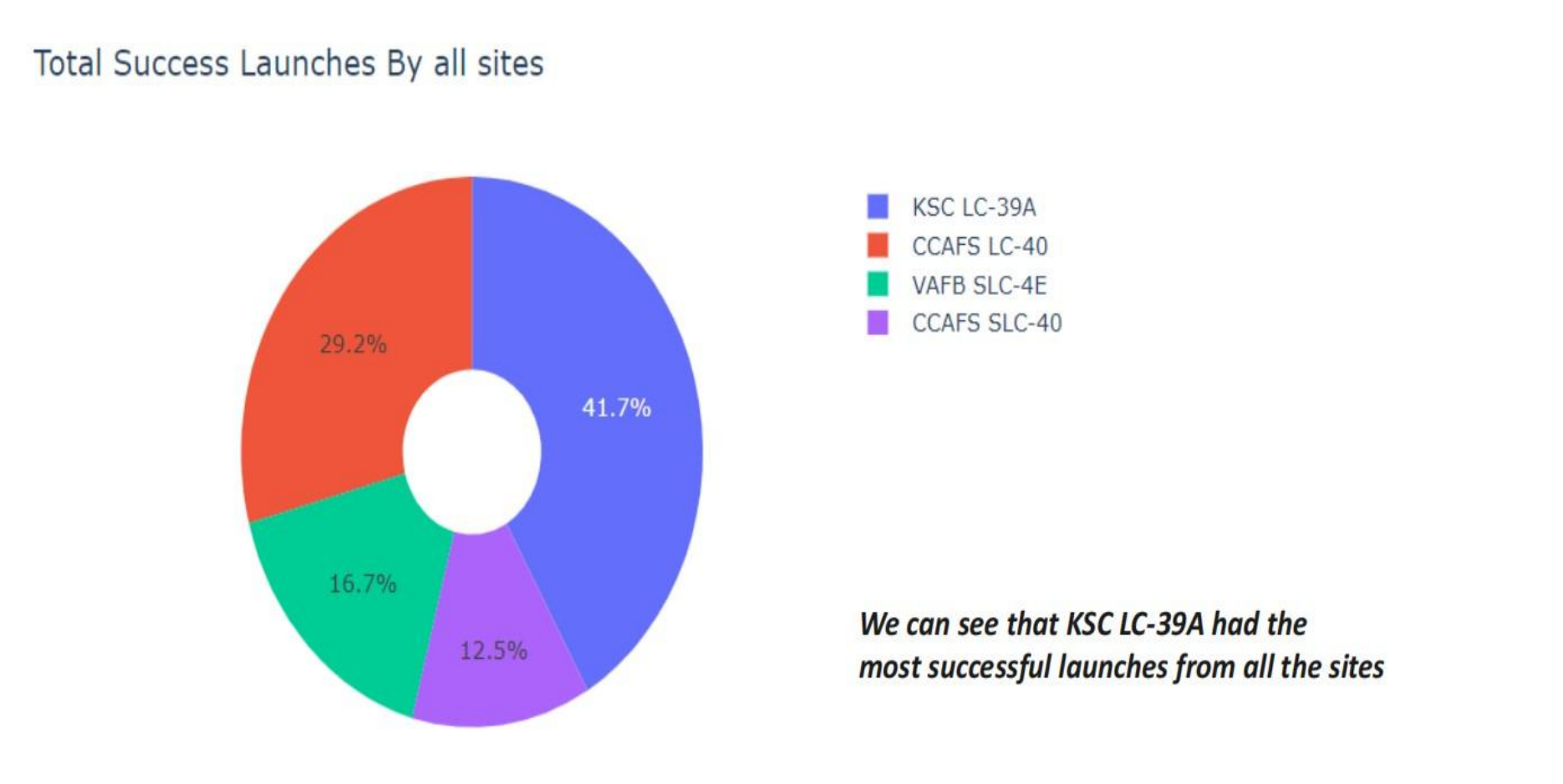
Working out Launch Sites distance to landmarks to find trends with Haversine formula using KSCLC-39A as a reference



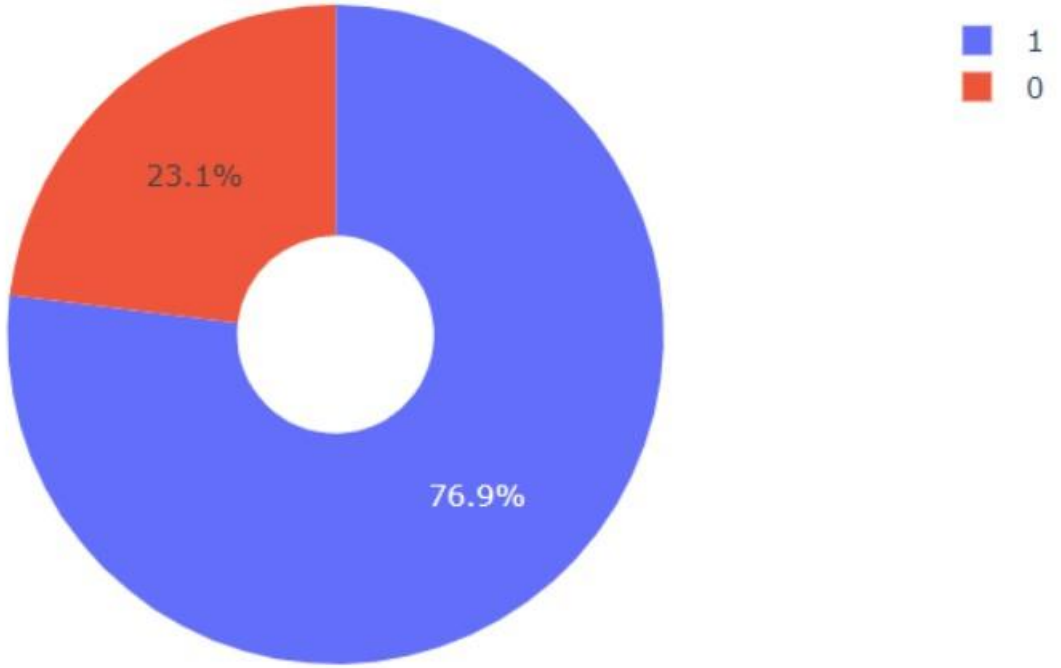
Nearest Coastline to Florida Lauch site is 0.82 km


Section 5
Build a Dashboard with Plotly Dash

DASHBOARD – Pie chart showing the success percentage achieved by each launch site

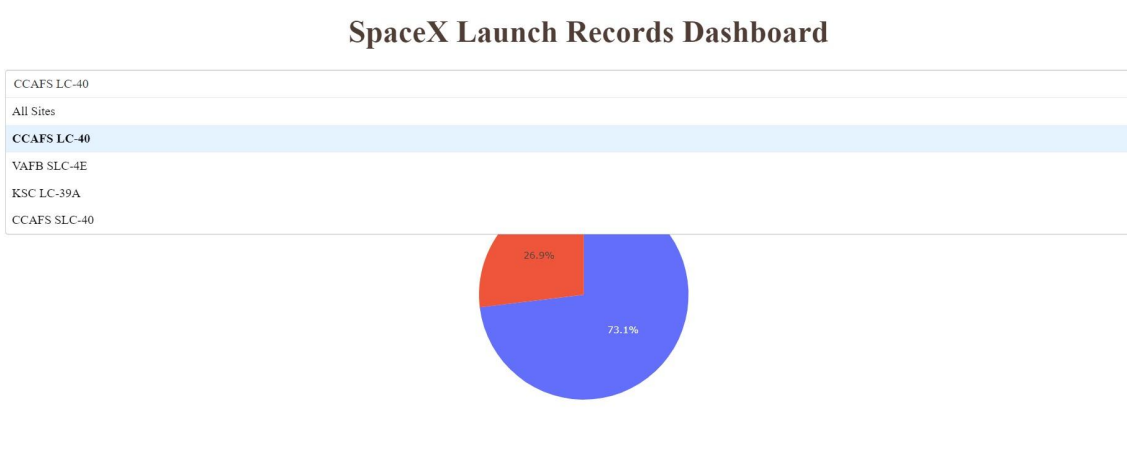


DASHBOARD – Pie chart for the launch site with highest launch success ratio

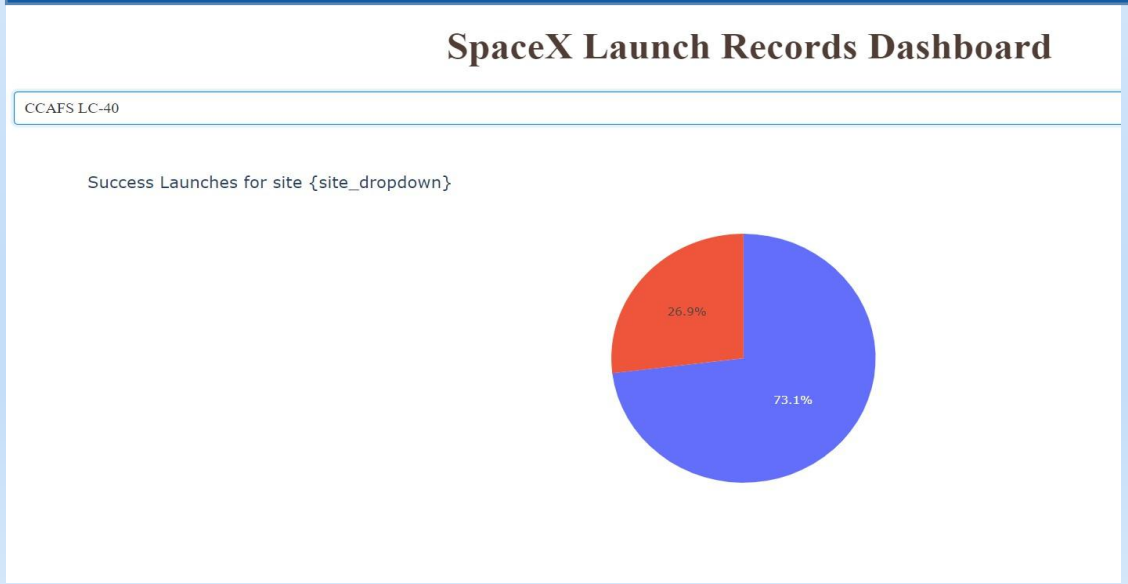


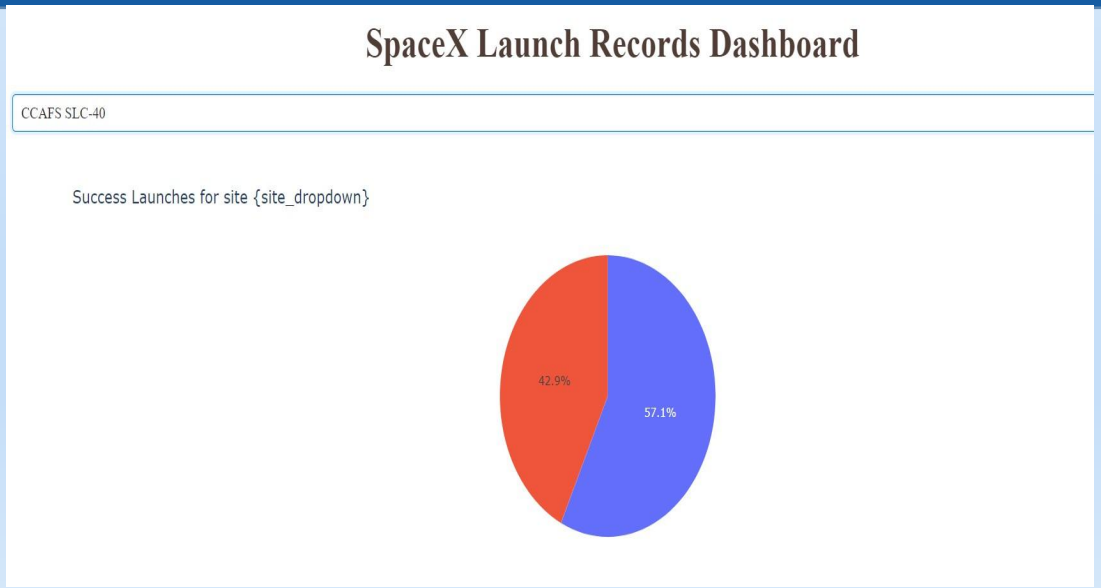
KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

DASHBOARD – Dropdown list of launchsites

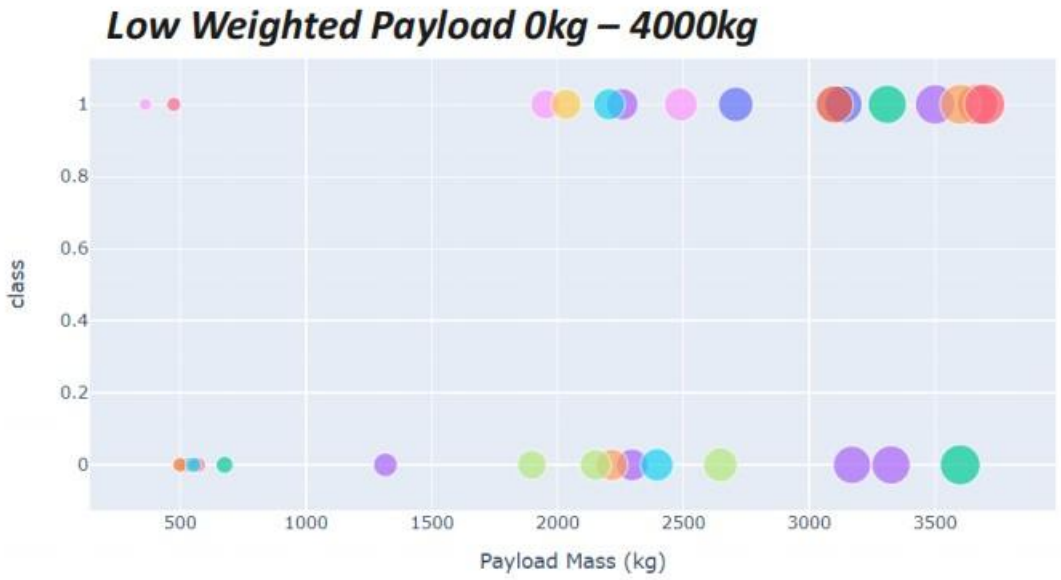


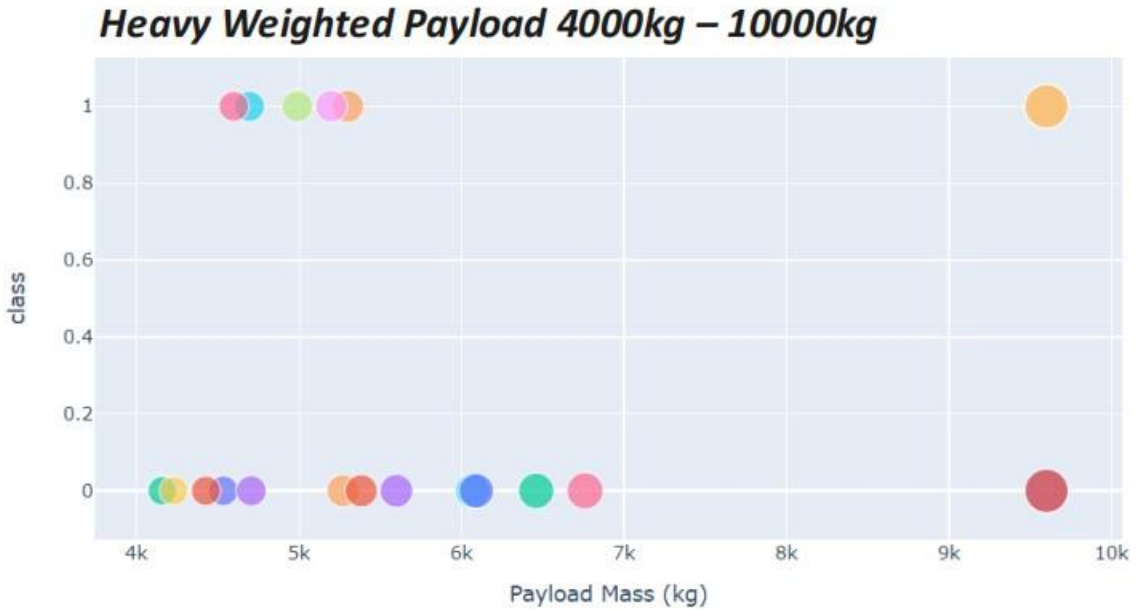
DASHBOARD – Pie chart for the launch site CCAFS LC-40 and CCAF SLC-40 launch success ratio





DASHBOARD – Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider





We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 6

Predictive Analysis (Classification)

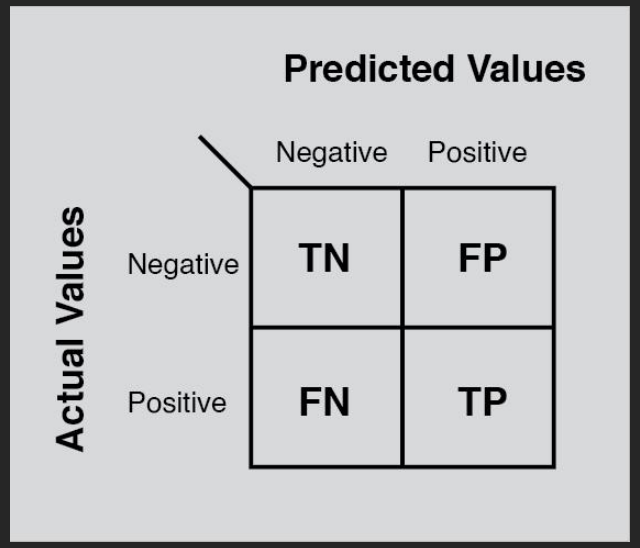
Classification Accuracy

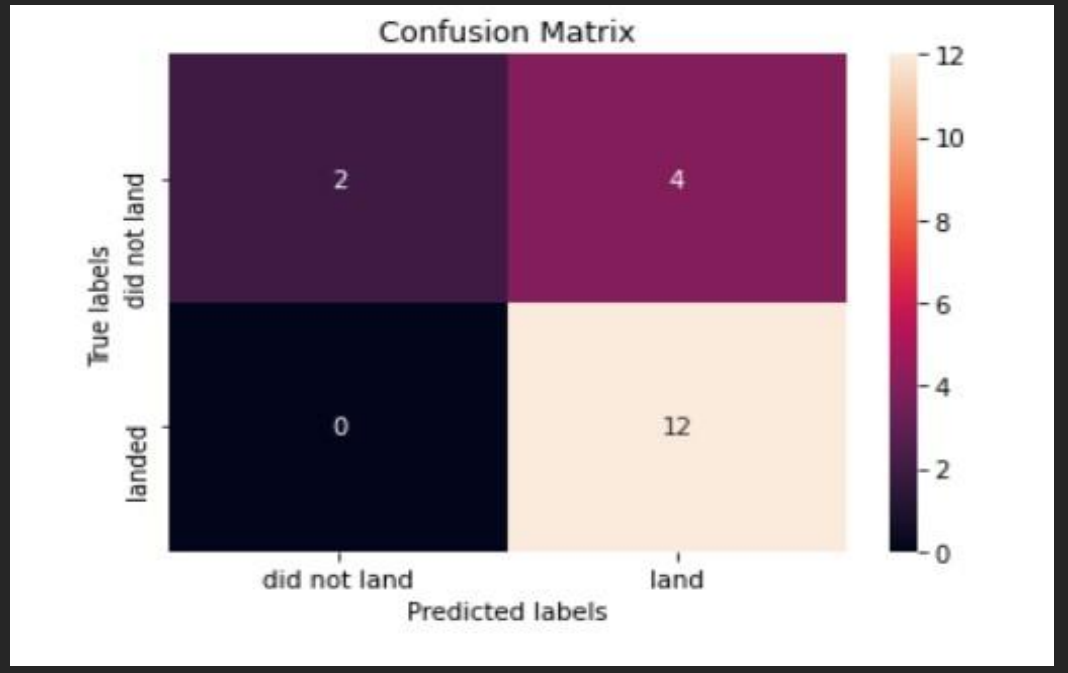
As you can see our accuracy is extremely close but we do have a winner its down to decimal places! Our Best Algorithm is Decision Tree with 90.178% accuracy.

```
KNN: 0.8482142857142858
LogisticRegression 0.8464285714285713
Decision tree 0.9017857142857144
Best Algorithm is Tree with a score of 0.9017857142857144
Best Params is : {'criterion': 'gini', 'max_depth': 14, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'random'}
```

Confusion Matrix

- Examining the confusion matrix, we see that Tree can distinguish between the different classes. We see that the major problem is false positives.





Conclusions

The Tree Classifier Algorithm is the best for Machine Learning for this dataset

- Low weighted payloads perform better than the heavier payloads
- The success rates for SpaceX launches is directly proportional time in years they will eventually perfect the launches
- We can see that KSC LC-39A had the most successful launches from all the sites
- Orbit GEO,HEO,SSO,ES-L1 has the best Success Rate

Appendix

Haversine formula

Introduction

The haversine formula determines the great-circle distance between two points on a sphere given their longitudes and latitudes. Important in navigation, it is a special case of a more general formula in spherical trigonometry, the law of haversines, that relates the sides and angles of spherical triangles.

Usage

Why did I use this formula? First of all, I believe the Earth is round/elliptical. I am not a Flat Earth Believer! Jokes aside when doing Google research for integrating my ADGGoogleMaps API with a Python function to calculate the distance using two distinct sets of {longitudinal, latitudinal} list sets. Haversine was the trigonometric solution to solve my requirements above.

Formula:

$$a = \sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos\varphi 1 \cdot \cos\varphi 2 \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right)$$

$$c = 2 \cdot \operatorname{atan2}\left(\sqrt{a}, \sqrt{(1-a)}\right)$$

$$d = R \cdot c$$

Thank you!