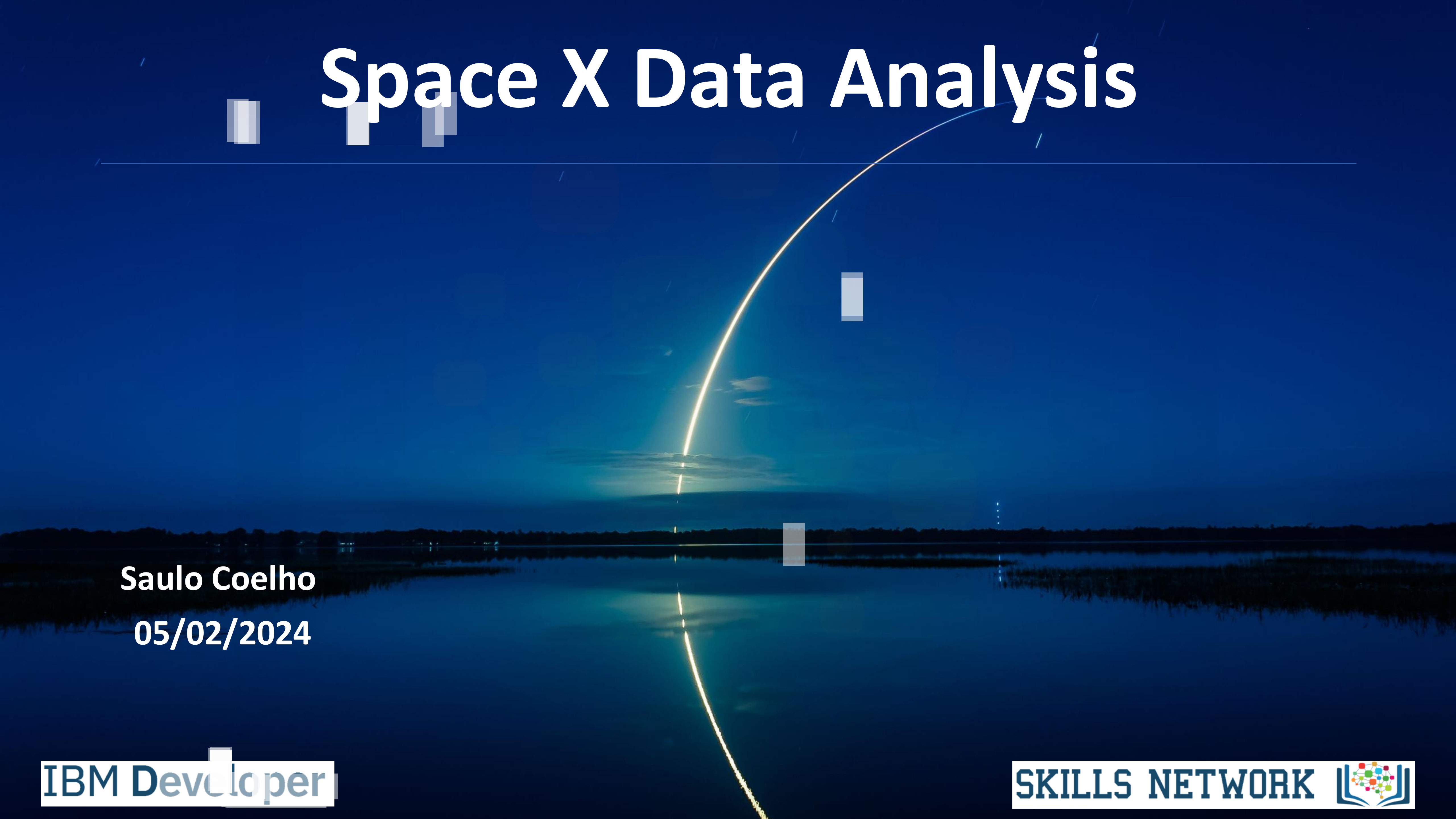
# Space X Data Analysis



Saulo Coelho

05/02/2024

IBM Developer

SKILLS NETWORK

## OUTLINE

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

## EXECUTIVE SUMMARY

- **Methodology**
  - Data Collection
  - Data Wrangling
  - Exploratory Data Analysis(SQLite and Visualization)
  - Predictive Analysis (Classification)
- **Results**
  - Dashboard
  - EAD results
  - Predictive analysis of classification models

IBM Developer

SKILLS NETWORK

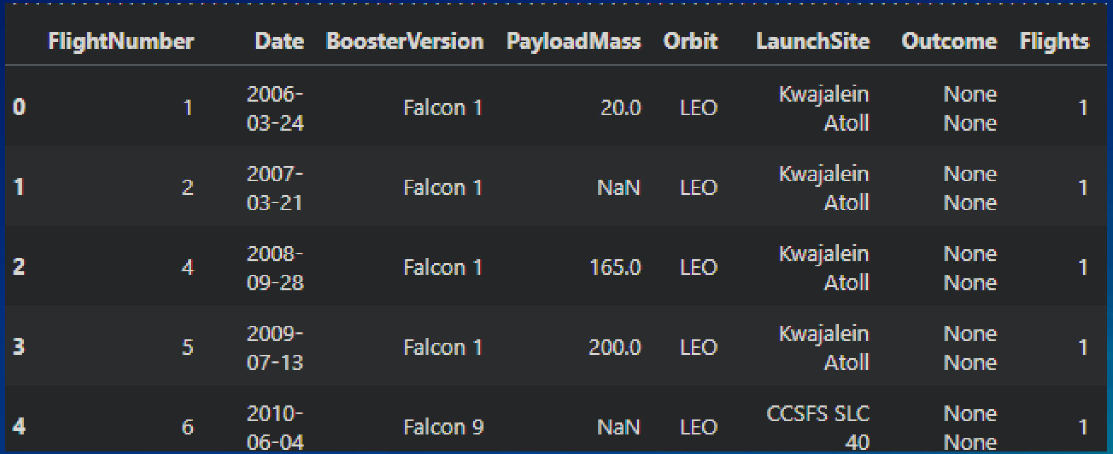
## INTRODUCTION

In the dynamic landscape of space exploration, understanding the patterns and factors influencing launch success is paramount. Our project delves into the correlation between experience, flight numbers, and success rates at launch sites. With a keen focus on historical data, we aim to unravel intriguing insights that can guide future space missions.

IBM Developer

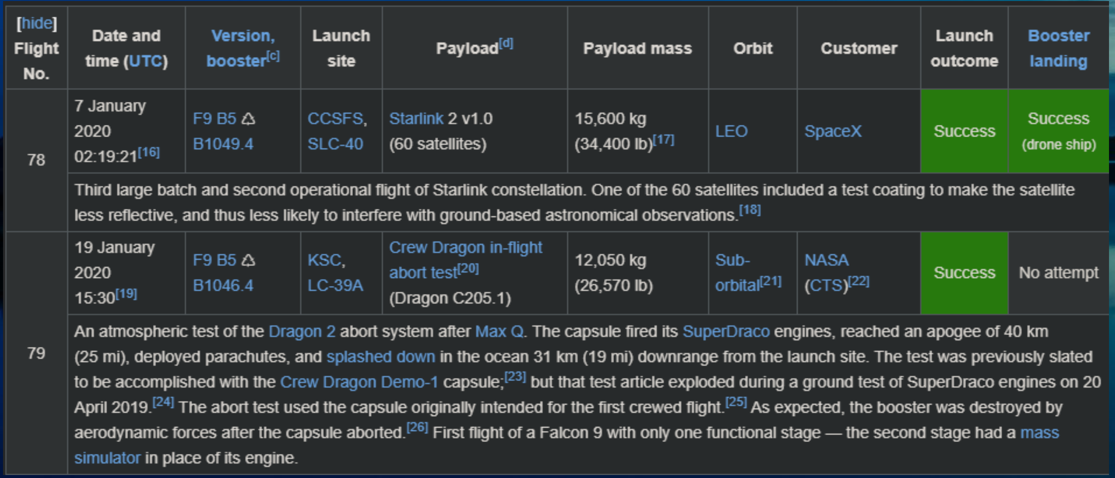
SKILLS NETWORK

# METHODOLOGY



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

*Original Wiki's Table*



| [hide] Flight No. | Date and time (UTC) | Version, booster[c] | Launch site | Payload[d] | Payload mass | Orbit | Customer | Launch outcome | Booster landing |
|---|---|---|---|---|---|---|---|---|---|
| 78 | 7 January 2020 02:19:21[16] | F9 B5 △ B1049.4 | CCSFS, SLC-40 | Starlink 2 v1.0 (60 satellites) | 15,600 kg (34,400 lb)[17] | LEO | SpaceX | Success | Success (drone ship) |
| | Third large batch and second operational flight of Starlink constellation. One of the 60 satellites included a test coating to make the satellite less reflective, and thus less likely to interfere with ground-based astronomical observations.[18] | | | | | | | | |
| 79 | 19 January 2020 15:30[19] | F9 B5 △ B1046.4 | KSC, LC-39A | Crew Dragon in-flight abort test[20] (Dragon C205.1) | 12,050 kg (26,570 lb) | Sub-orbital[21] | NASA (CTS)[22] | Success | No attempt |
| | An atmospheric test of the Dragon 2 abort system after Max Q. The capsule fired its SuperDraco engines, reached an apogee of 40 km (25 mi), deployed parachutes, and splashed down in the ocean 31 km (19 mi) downrange from the launch site. The test was previously slated to be accomplished with the Crew Dragon Demo-1 capsule;[23] but that test article exploded during a ground test of SuperDraco engines on 20 April 2019.[24] The abort test used the capsule originally intended for the first crewed flight.[25] As expected, the booster was destroyed by aerodynamic forces after the capsule aborted.[26] First flight of a Falcon 9 with only one functional stage — the second stage had a mass simulator in place of its engine. | | | | | | | | |

*Data Frame Sample Image*

## DATA COLLECTION

- Libraries used: Pandas, NumPy, BeautifulSoup, Requests
- Steps:

• Collection of data though Space X API.

• Picking the relevant Data from API retrieved data:

Only Falcon 9 Ship.

• Normalizing all the relevant data to reduce noise.

• Data Frame creation for easier handling.

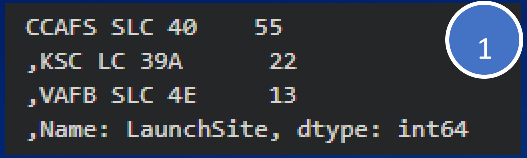
• Web Scraping
  - Used BeautifulSoup to scrape data from Wiki page with relevant data of Space X launch's.
  - Transformed web data into a Data Frame.

*Link: https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches*

IBM Developer

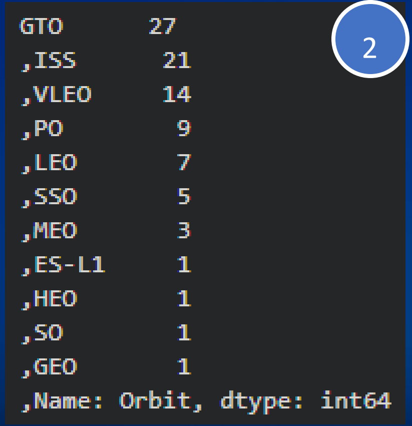
SKILLS NETWORK

# METHODOLOGY



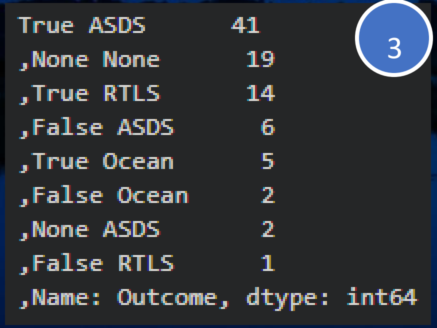
1

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



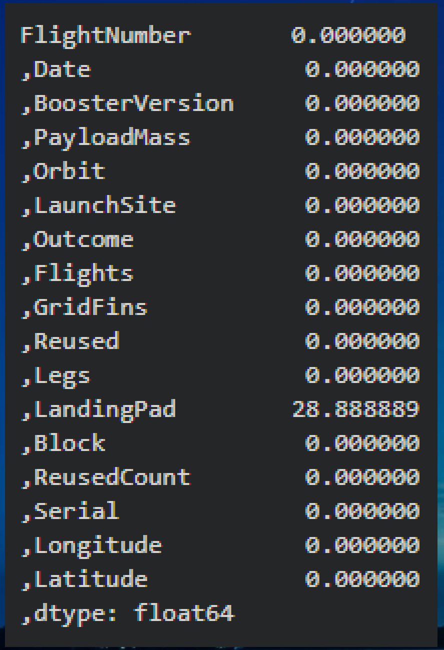
2

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



3

```
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
```



```
FlightNumber       0.000000
,Date               0.000000
,BoosterVersion     0.000000
,PayloadMass        0.000000
,Orbit              0.000000
,LaunchSite         0.000000
,Outcome            0.000000
,Flights            0.000000
,GridFins           0.000000
,Reused             0.000000
,Legs               0.000000
,LandingPad        28.888889
,Block              0.000000
,ReusedCount        0.000000
,Serial             0.000000
,Longitude          0.000000
,Latitude           0.000000
,dtype: float64
```

*Missing Values*

## DATA WRANGLING

1. Libraries used: Pandas, NumPy
2. Steps:

- Finding Missing Data and Replacing if possible
- Dropping Null values or Invalid Columns
- Calculated:
  1. Number of launches on each site
  2. Number and occurrence of each orbit
  3. Number and occurrence of mission outcome of the orbits

**3.Legend:**

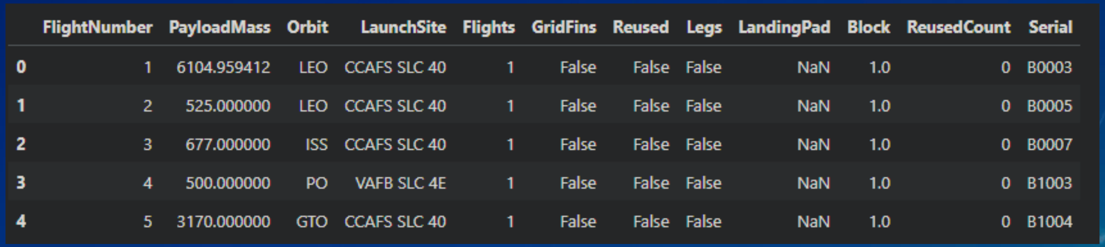
*True Ocean – Successfully landed on the ocean.*
*False Ocean – Unsuccessfully landed on the ocean.*
*True RTLS – Successfully landed to a ground pad*
*False RTLS – Unsuccessfully landed to a ground pad.*
*True ASDS – Successfully landed to a drone ship*
*False ASDS – Unsuccessfully landed to a drone ship.*
*None ASDS and None None – Total failure.*

IBM Developer

SKILLS NETWORK

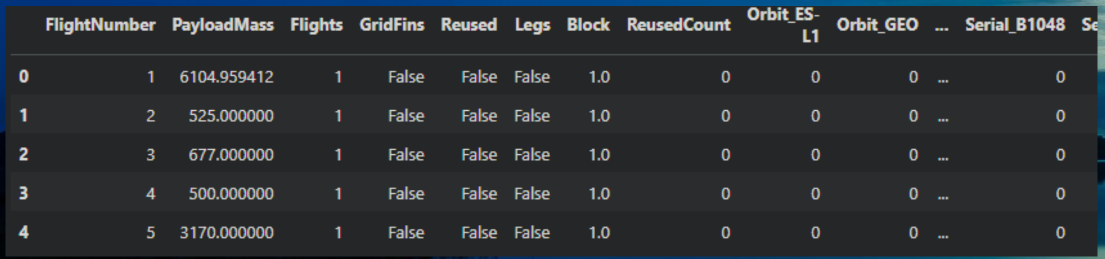
# METHODOLOGY

## Before



| | FlightNumber | PayloadMass | Orbit | LaunchSite | Flights | GridFins | Reused | Legs | LandingPad | Block | ReusedCount | Serial |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 6104.959412 | LEO | CCAFS SLC 40 | 1 | False | False | False | NaN | 1.0 | 0 | B0003 |
| 1 | 2 | 525.000000 | LEO | CCAFS SLC 40 | 1 | False | False | False | NaN | 1.0 | 0 | B0005 |
| 2 | 3 | 677.000000 | ISS | CCAFS SLC 40 | 1 | False | False | False | NaN | 1.0 | 0 | B0007 |
| 3 | 4 | 500.000000 | PO | VAFB SLC 4E | 1 | False | False | False | NaN | 1.0 | 0 | B1003 |
| 4 | 5 | 3170.000000 | GTO | CCAFS SLC 40 | 1 | False | False | False | NaN | 1.0 | 0 | B1004 |

## After



| | FlightNumber | PayloadMass | Flights | GridFins | Reused | Legs | Block | ReusedCount | Orbit_ES-L1 | Orbit_GEO | ... | Serial_B1048 | Se |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 6104.959412 | 1 | False | False | False | 1.0 | 0 | 0 | 0 | ... | 0 | |
| 1 | 2 | 525.000000 | 1 | False | False | False | 1.0 | 0 | 0 | 0 | ... | 0 | |
| 2 | 3 | 677.000000 | 1 | False | False | False | 1.0 | 0 | 0 | 0 | ... | 0 | |
| 3 | 4 | 500.000000 | 1 | False | False | False | 1.0 | 0 | 0 | 0 | ... | 0 | |
| 4 | 5 | 3170.000000 | 1 | False | False | False | 1.0 | 0 | 0 | 0 | ... | 0 | |

## DATA WRANGLING

- **Features Engineering:**

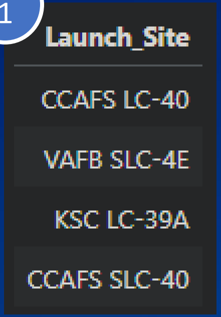
- This methodology consist in "equalize" the data values from different attributes so we reduce noises in the analysis.
- We used OneHotEncoder to it, transforming categorical values into integers
- Pandas.get_dummies were used to simplify and agilize the process.

IBM Developer

SKILLS NETWORK

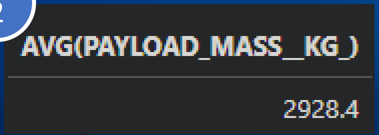
# METHODOLOGY

1



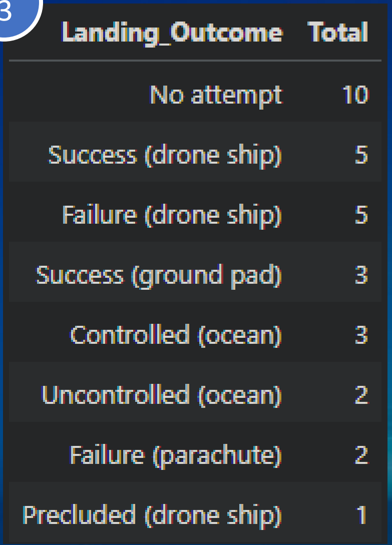
| Launch_Site |
|---|
| CCAFS LC-40 |
| VAFB SLC-4E |
| KSC LC-39A |
| CCAFS SLC-40 |

2



| AVG(PAYLOAD_MASS__KG_) |
|---|
| 2928.4 |

3



| Landing_Outcome | Total |
|---|---|
| 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 |

## EXPLORATORY ANALYSIS(DATABSE-SQLite)

- **SQL queries were used to perform to retrieve several types of data as examples:**

1. **Display the names of the unique launch sites in the space mission:**

*%sql SELECT DISTINCT(LAUNCH_SITE) FROM SPACEXTBL;*

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

*%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE BOOSTER_VERSION = 'F9 v1.1'*

1. **Rank the count of landing outcomes between the date 2010-06-04 and 2017-03-20.:**

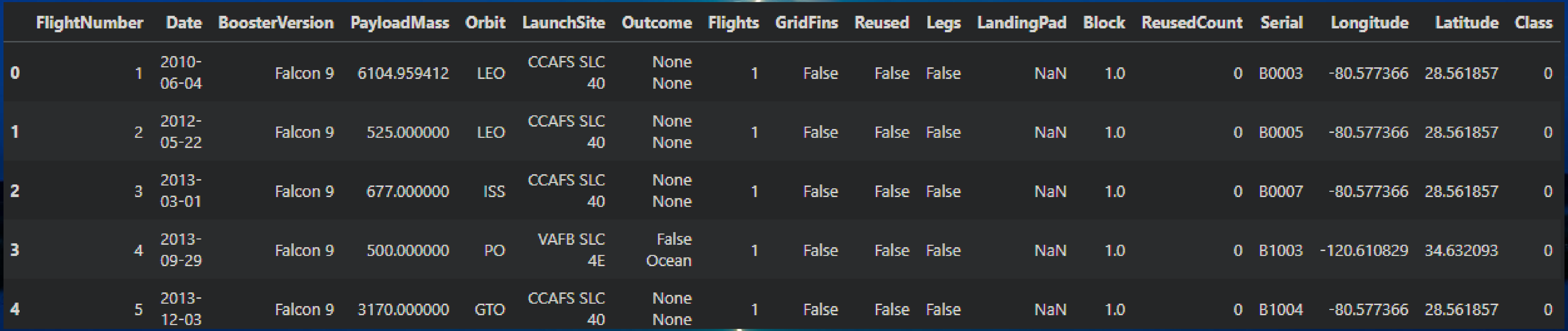
*%sql SELECT LANDING_OUTCOME, COUNT(LANDING_OUTCOME) AS Total FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY LANDING_OUTCOME ORDER BY Total DESC*

## METHODOLOGY

### EXPLORATORY ANALYSIS(Visualization-Plots)

**We used several data visualization tools to visualize the data in different contrasts in order to have a better comprehension of the real mean behind them**

**Clean Data Frame(head) used to plot the visualizations:**



| | FlightNumber | Date | BoosterVersion | PayloadMass | Orbit | LaunchSite | Outcome | Flights | GridFins | Reused | Legs | LandingPad | Block | ReusedCount | Serial | Longitude | Latitude | Class |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 2010-06-04 | Falcon 9 | 6104.959412 | LEO | CCAFS SLC 40 | None None | 1 | False | False | False | NaN | 1.0 | 0 | B0003 | -80.577366 | 28.561857 | 0 |
| **1** | 2 | 2012-05-22 | Falcon 9 | 525.000000 | LEO | CCAFS SLC 40 | None None | 1 | False | False | False | NaN | 1.0 | 0 | B0005 | -80.577366 | 28.561857 | 0 |
| **2** | 3 | 2013-03-01 | Falcon 9 | 677.000000 | ISS | CCAFS SLC 40 | None None | 1 | False | False | False | NaN | 1.0 | 0 | B0007 | -80.577366 | 28.561857 | 0 |
| **3** | 4 | 2013-09-29 | Falcon 9 | 500.000000 | PO | VAFB SLC 4E | False Ocean | 1 | False | False | False | NaN | 1.0 | 0 | B1003 | -120.610829 | 34.632093 | 0 |
| **4** | 5 | 2013-12-03 | Falcon 9 | 3170.000000 | GTO | CCAFS SLC 40 | None None | 1 | False | False | False | NaN | 1.0 | 0 | B1004 | -80.577366 | 28.561857 | 0 |

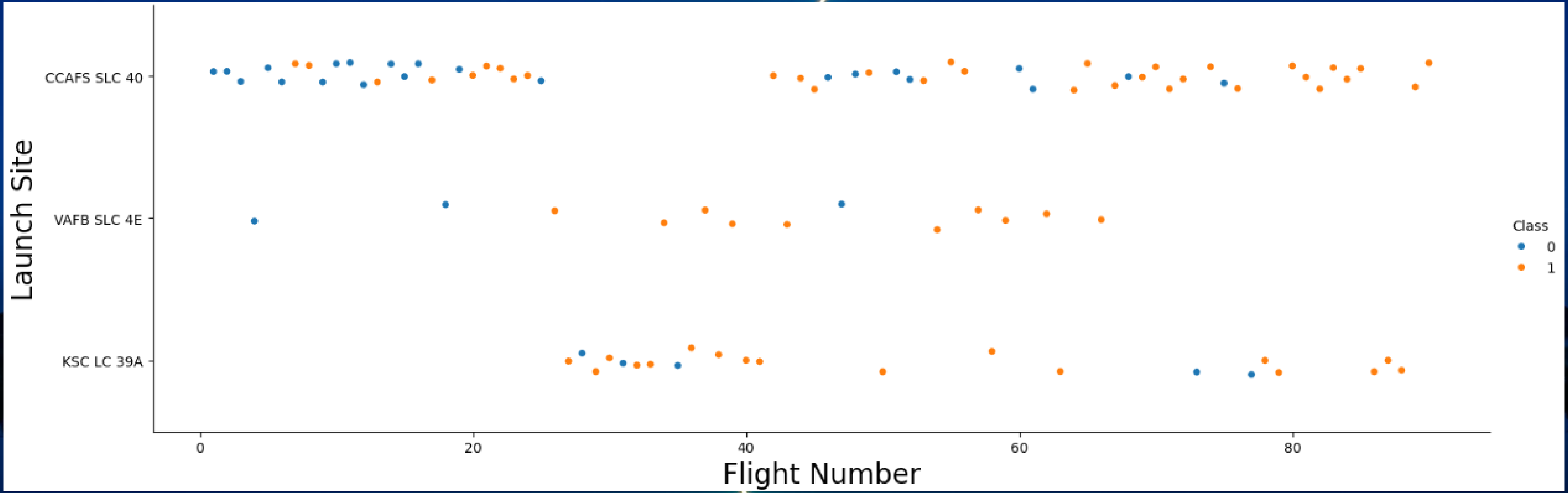
IBM Developer

SKILLS NETWORK

# METHODOLOGY

## EXPLORATORY ANALYSIS(Visualization-Plots)

### Relationship between Flight Number and Launch Site(Scatter PLot)



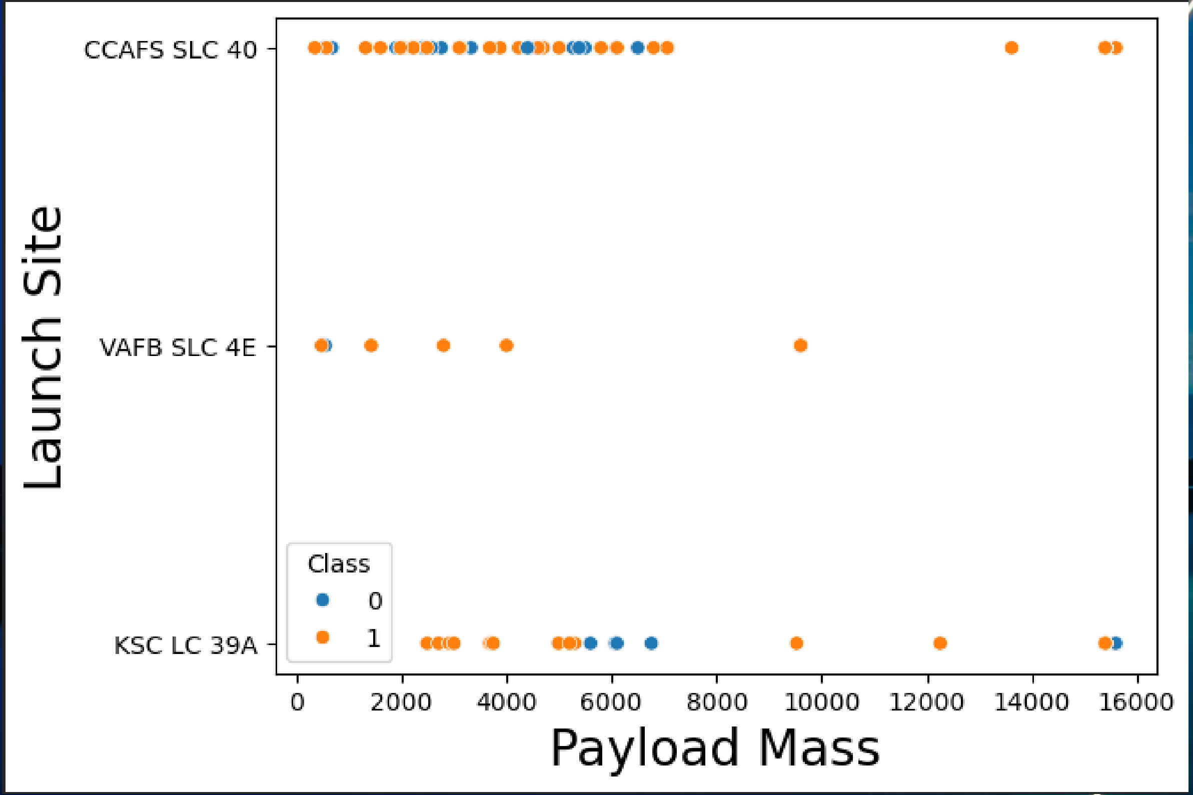
IBM Developer

SKILLS NETWORK

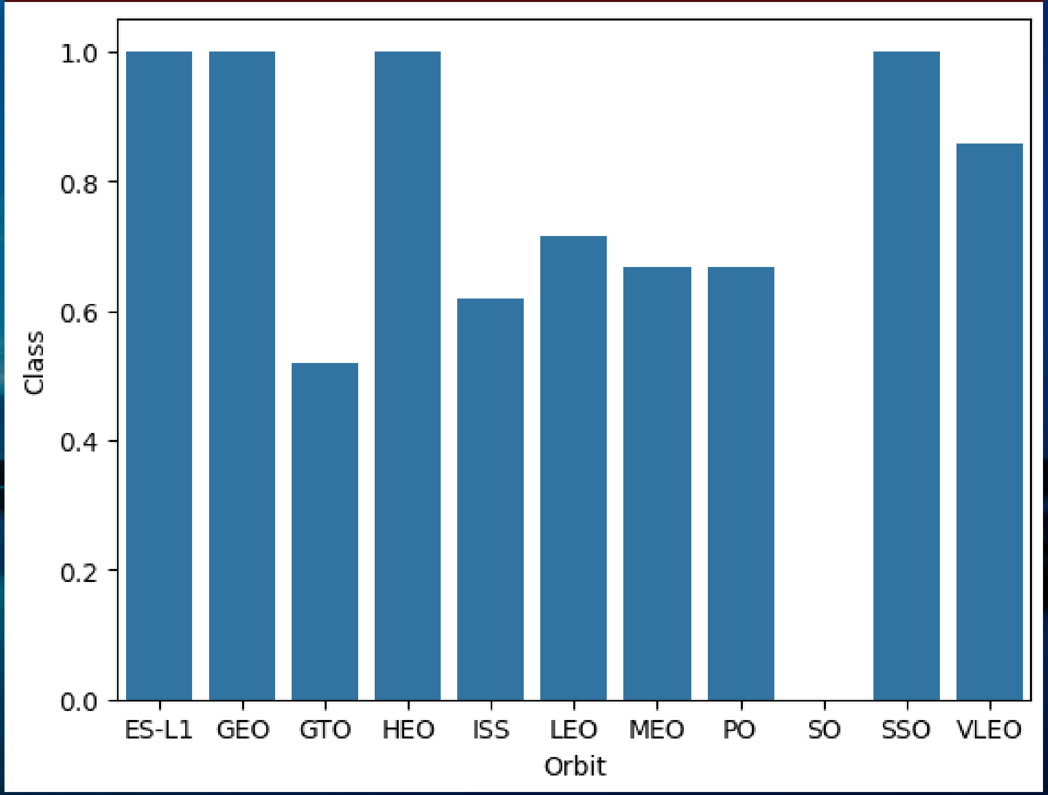
# METHODOLOGY

## EXPLORATORY ANALYSIS(Visualization-Plots)

### Relationship of Payload and Launch Site(Scatter Plot)



### Relationship of Orbit and Success Rate(Bar Plot)



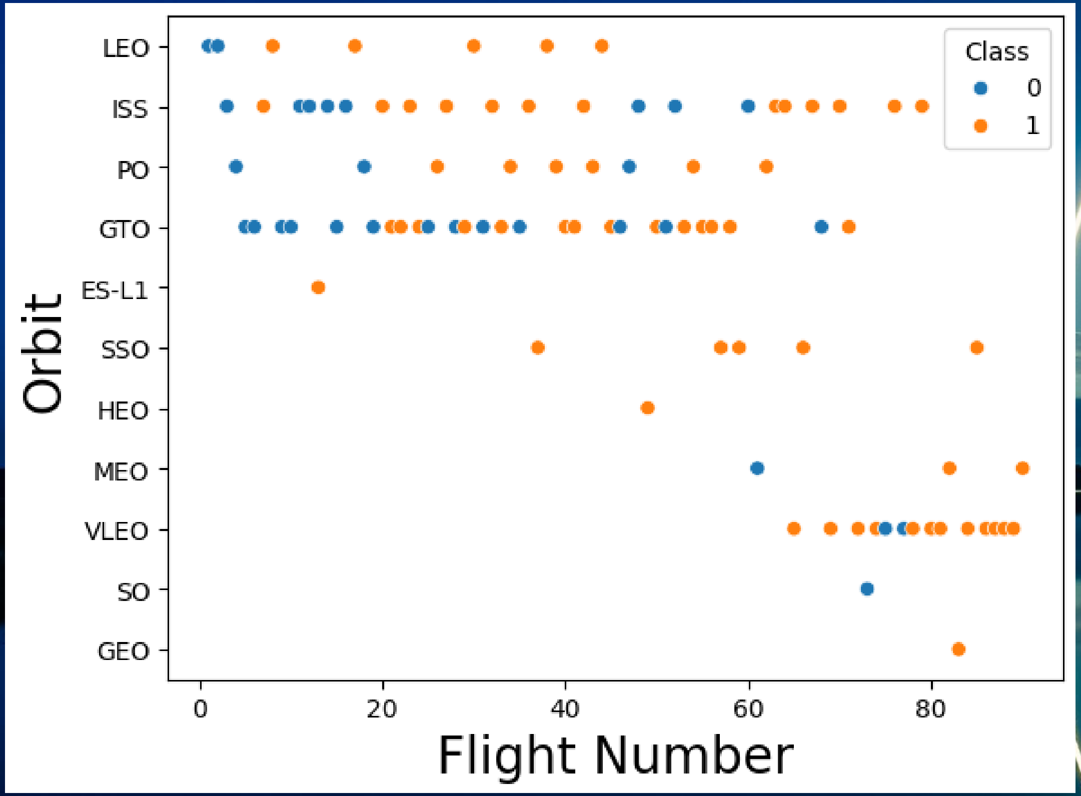
IBM Developer

SKILLS NETWORK

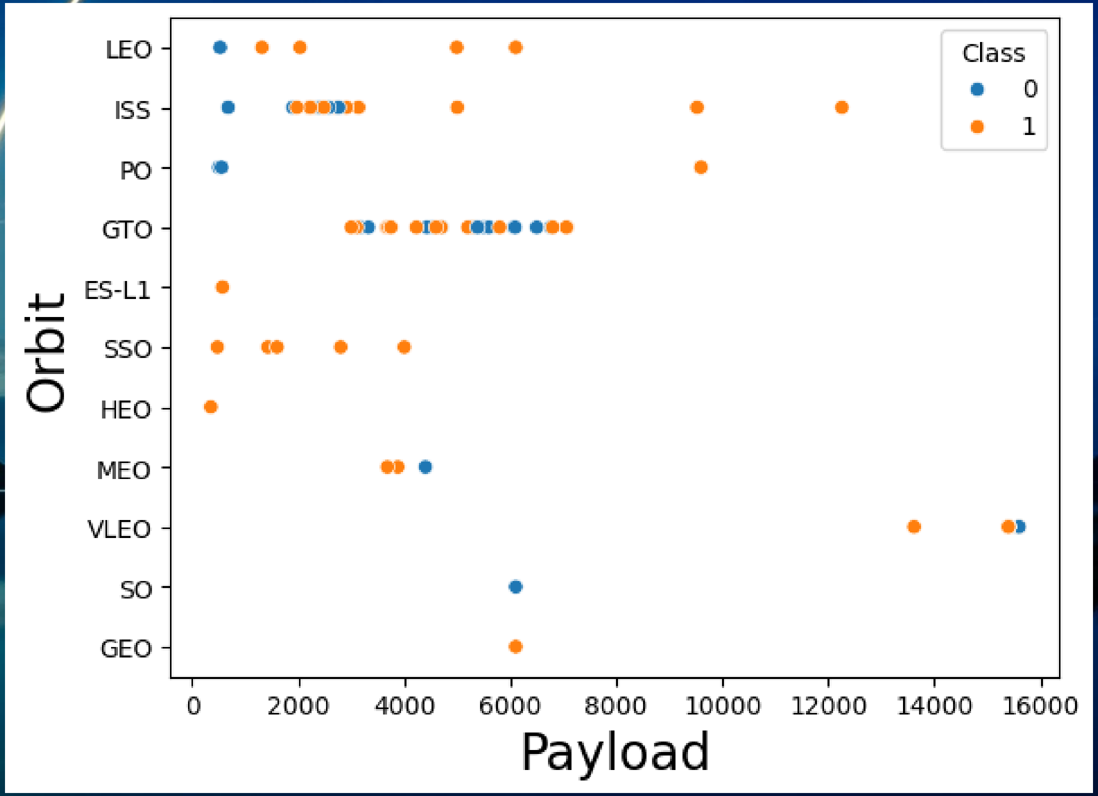
# METHODOLOGY

## EXPLORATORY ANALYSIS(Visualization-Plots)

### Relationship of Flight Number and Orbit(Scatter Plot)



### Relationship of Payload and Orbit(Scatter Plot)



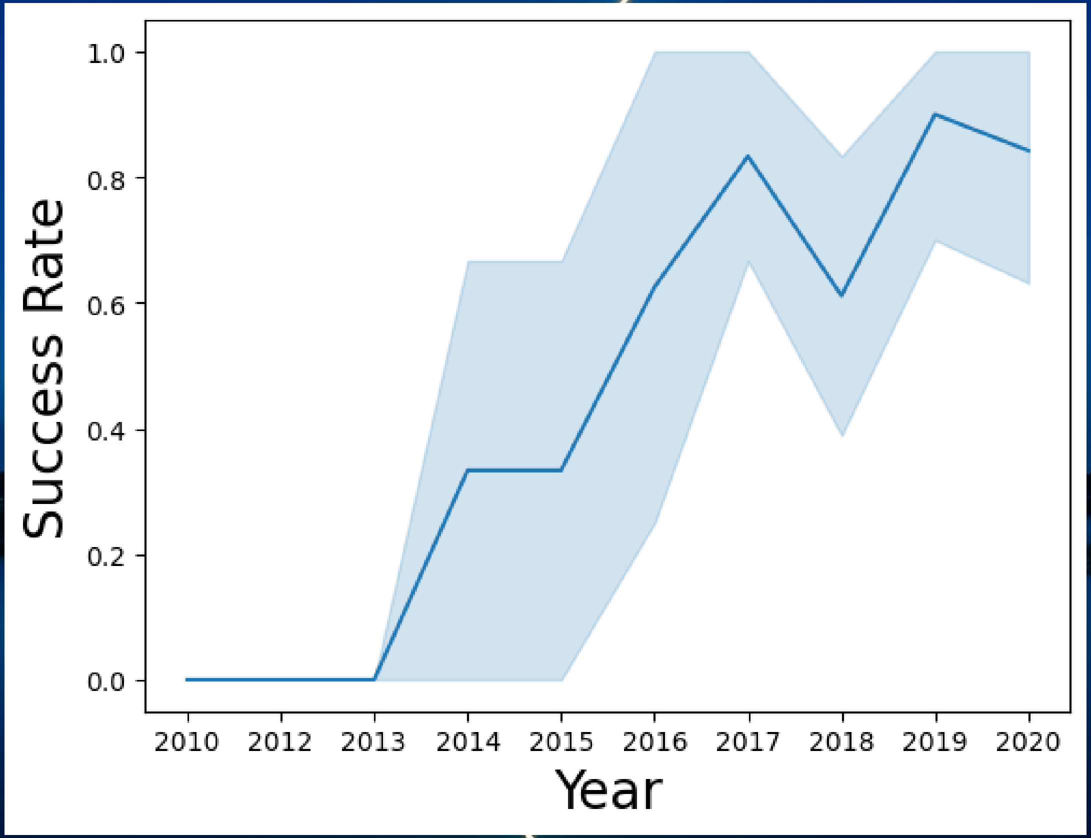
IBM Developer

SKILLS NETWORK

## METHODOLOGY

**EXPLORATORY ANALYSIS(Visualization-Plots)**

**Relationship of Success Rate per Year(LinePlot)**

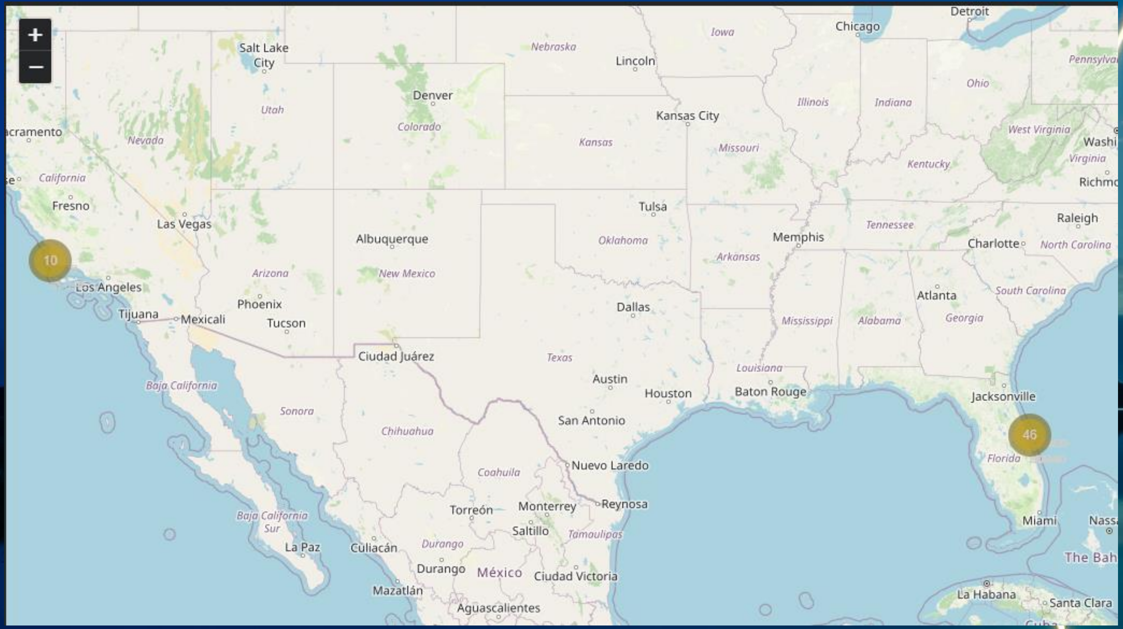


IBM Developer

SKILLS NETWORK

# METHODOLOGY

Sample



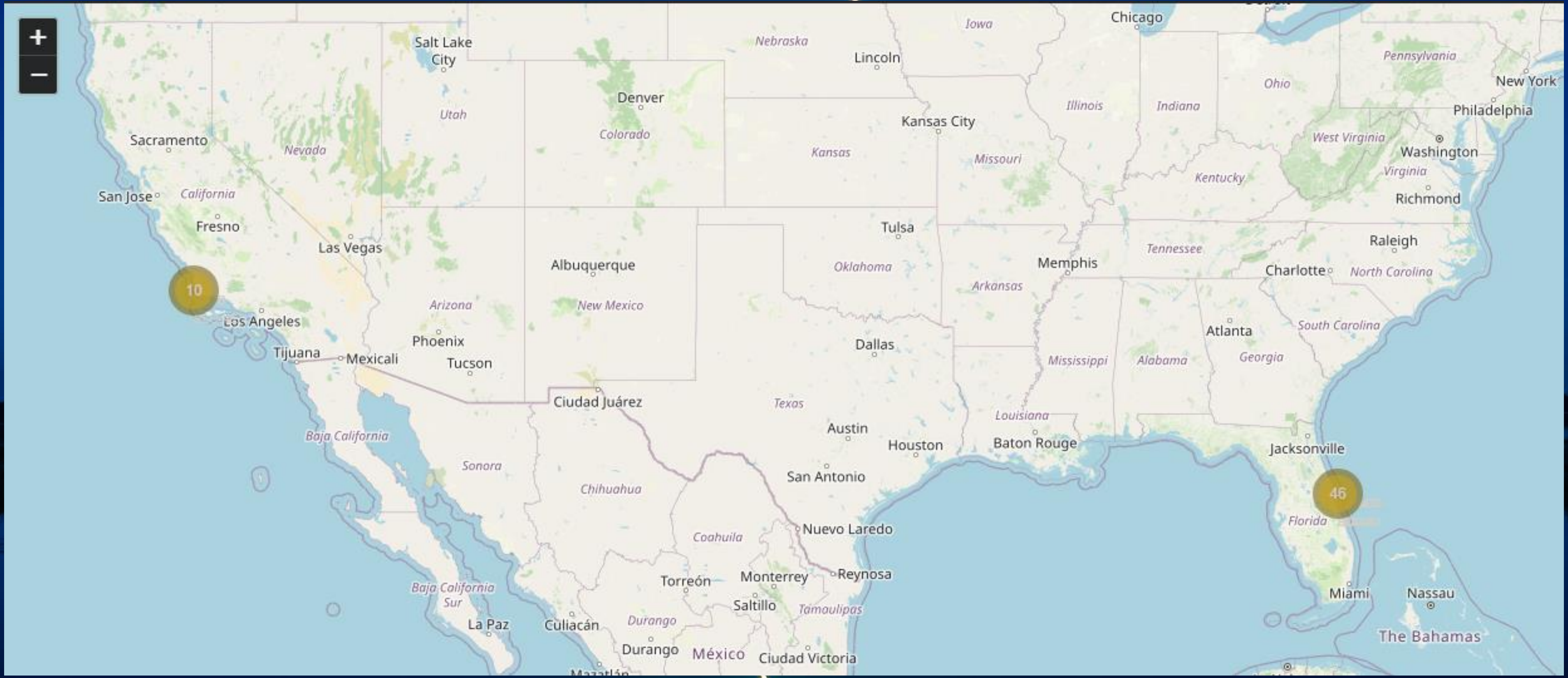
## EXPLORATORY ANALYSIS(Visual analysis with Folium)

- In this step, we used Folium library to add the Lauch sites on the map;
- We marked the success/failure for each launch site;
- Calculate the distances between a launch site to nearest railroads, highways and city.

# METHODOLOGY

## EXPLORATORY ANALYSIS(Visual analysis with Folium)

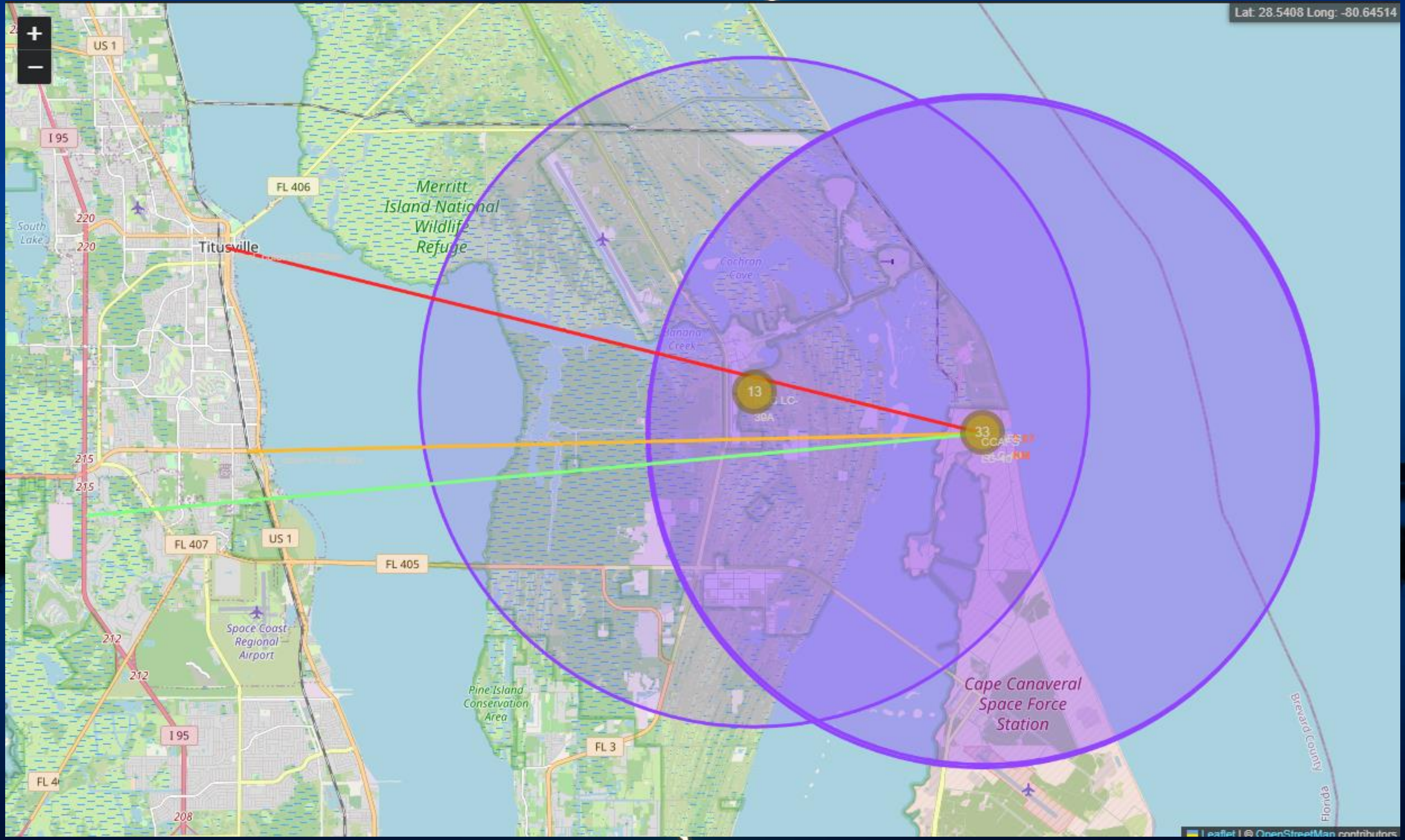
### Relationship of Success Rate per Lauch Site



# METHODOLOGY

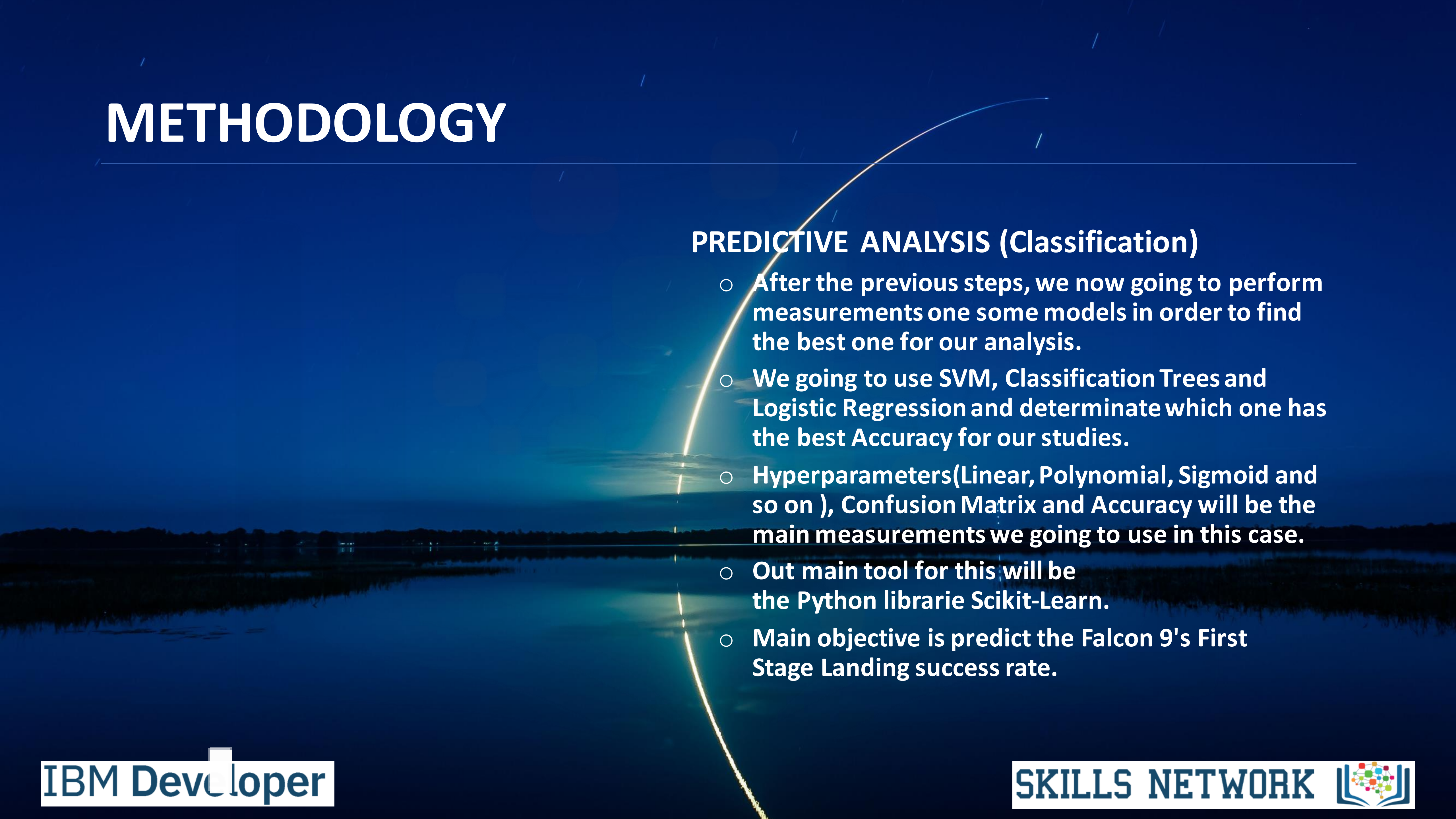
## EXPLORATORY ANALYSIS(Visual analysis with Folium)

**Calculate the distances between a launch site to nearest railroads, highways and city.**


Lat: 28.5408 Long: -80.64514
US 1
I 95
FL 406
Merritt Island National Wildlife Refuge
Titusville
South Lake
220
Cochran Cove
Banana Creek
13
33
CCAFS
215
FL 407
US 1
FL 405
Space Coast Regional Airport
212
Pine Island Conservation Area
FL 3
I 95
208
Cape Canaveral Space Force Station
Brevard County
Florida
Leaflet | © OpenStreetMap contributors

IBM Developer

SKILLS NETWORK



## METHODOLOGY

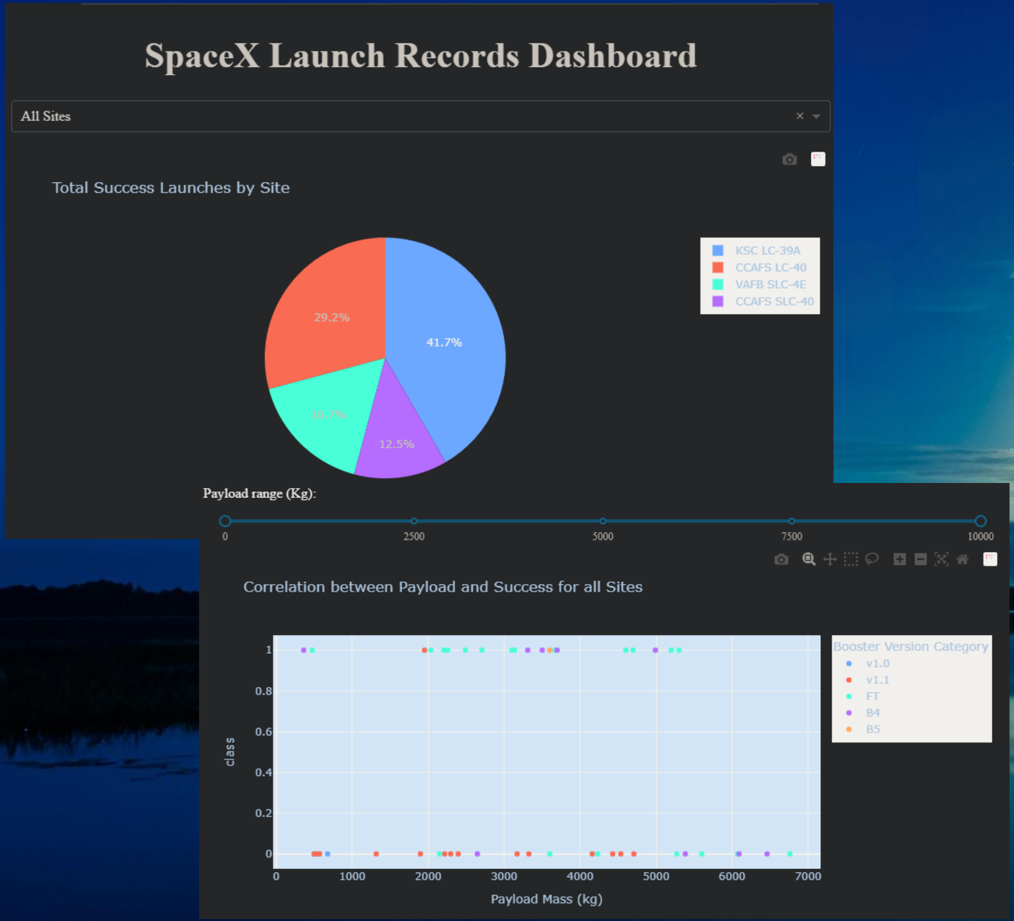
### PREDICTIVE ANALYSIS (Classification)

- After the previous steps, we now going to perform measurements one some models in order to find the best one for our analysis.
- We going to use SVM, Classification Trees and Logistic Regression and determinate which one has the best Accuracy for our studies.
- Hyperparameters(Linear, Polynomial, Sigmoid and so on ), Confusion Matrix and Accuracy will be the main measurements we going to use in this case.
- Out main tool for this will be the Python librarie Scikit-Learn.
- Main objective is predict the Falcon 9's First Stage Landing success rate.

IBM Developer

SKILLS NETWORK

## RESULTS



### DASHBOARD ANALYSIS

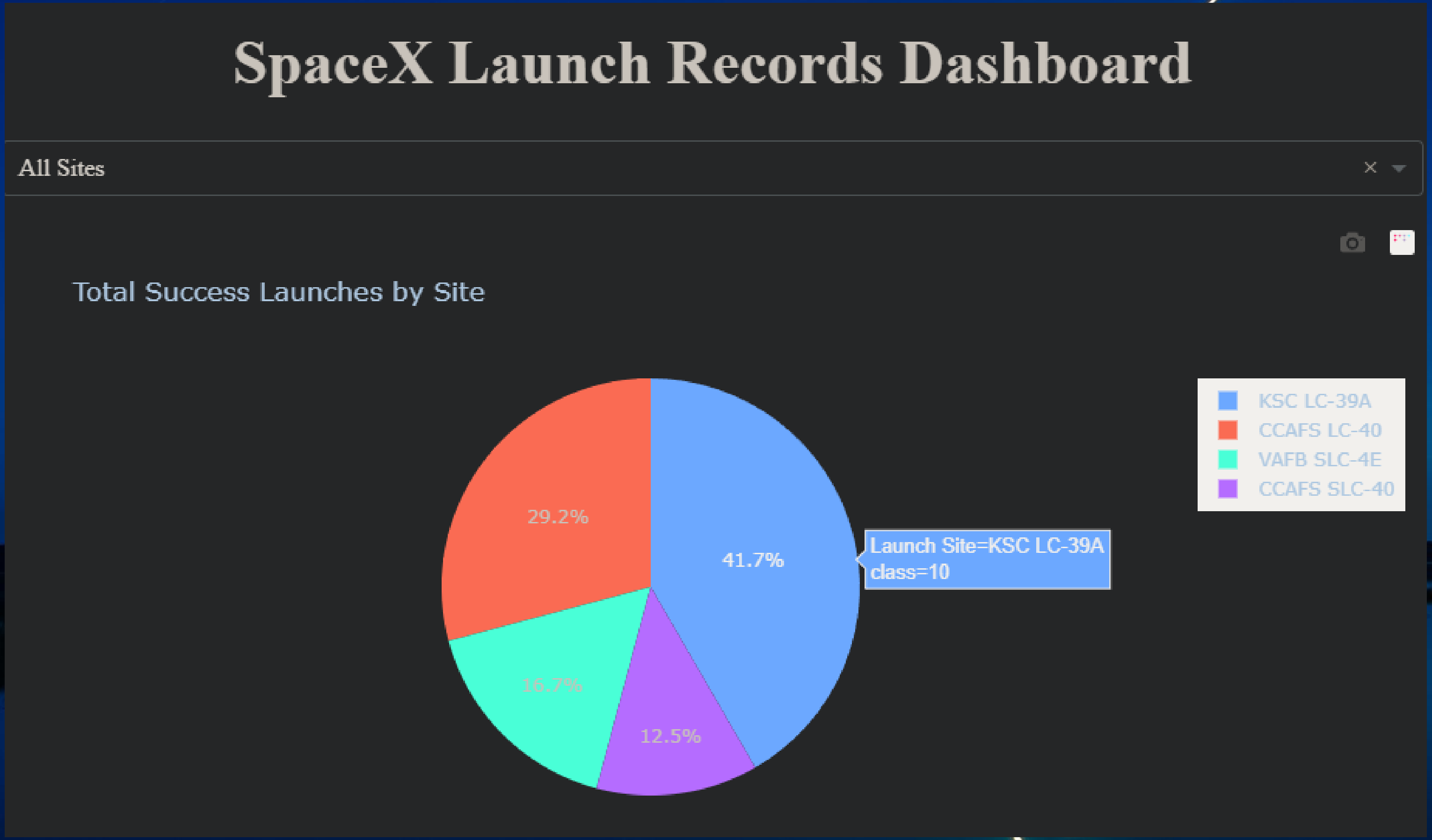
### SpaceX Launch Dashboard

- We created a Dashboard for easier and interactive data analysis of the crucial data we found to be most relevant to determinate the success or failure of launch's per Lanuch Site and Payload.
- We used an interactive dropdown for the Lauch Site, a slider for Payload and Pie Chart and Scatter Plot to cross the information as the user like.

IBM Developer

SKILLS NETWORK

# RESULTS



SpaceX Launch Records Dashboard

All Sites

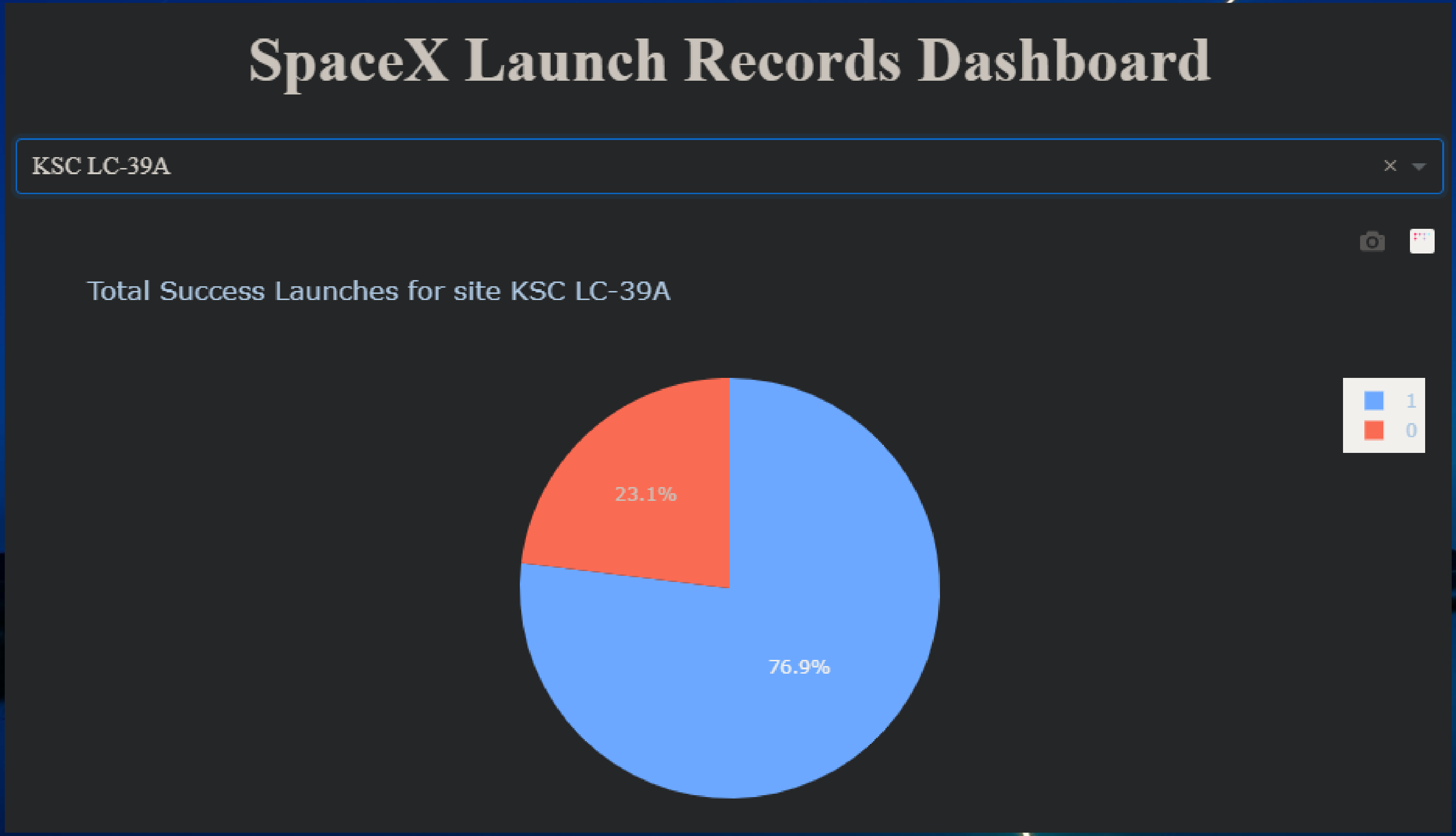
Total Success Launches by Site

## DASHBOARD ANALYSIS

**Conclusions(Pie Chart):**

- Of All Lauch site, the one with most success rate is the KSC LC-39 with 14.7% Success rate.
- The one with least success rate is the CCAFS SLC-40 with only 12.5% success rate.

## RESULTS


SpaceX Launch Records Dashboard

KSC LC-39A

Total Success Launches for site KSC LC-39A

### DASHBOARD ANALYSIS

Conclusions(Pie Chart):

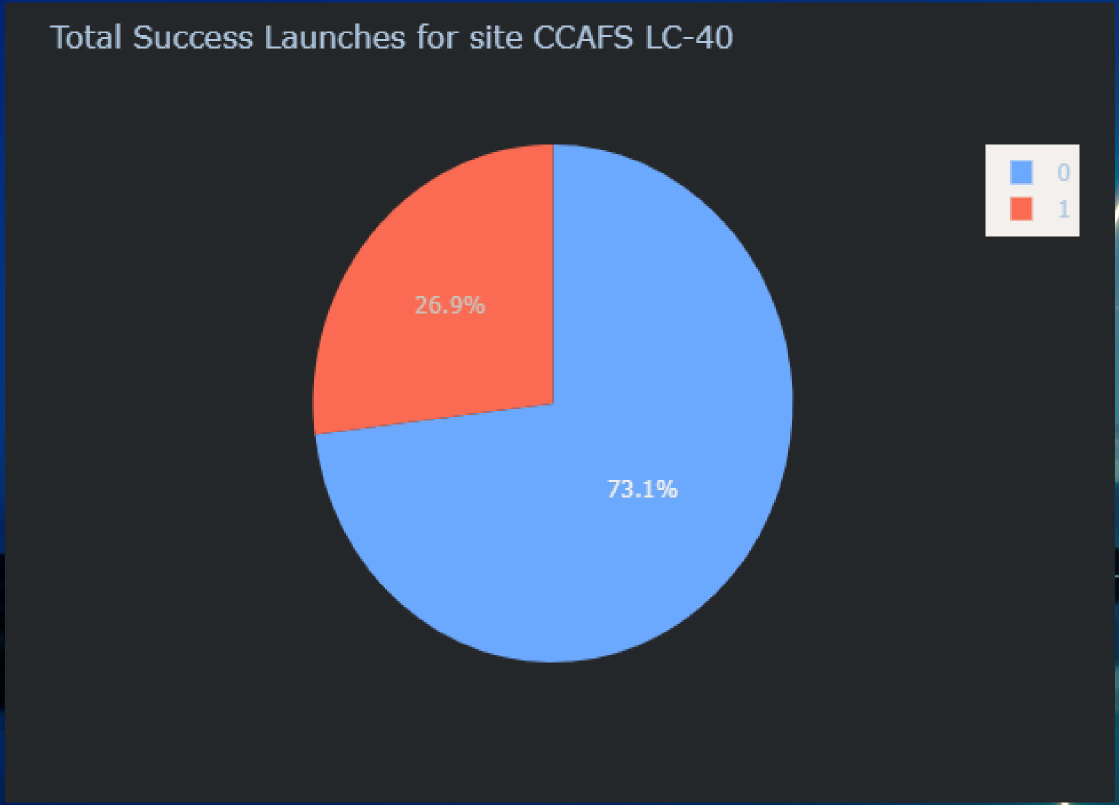
- The Lauch Site with most success rate is the KSC LC-39A with 76.9% s.r..
- The second one was the CCAFS LC-40 with 73.1% s.r..
- And the one with least s.r. was CCAFS SLC-40.

IBM Developer

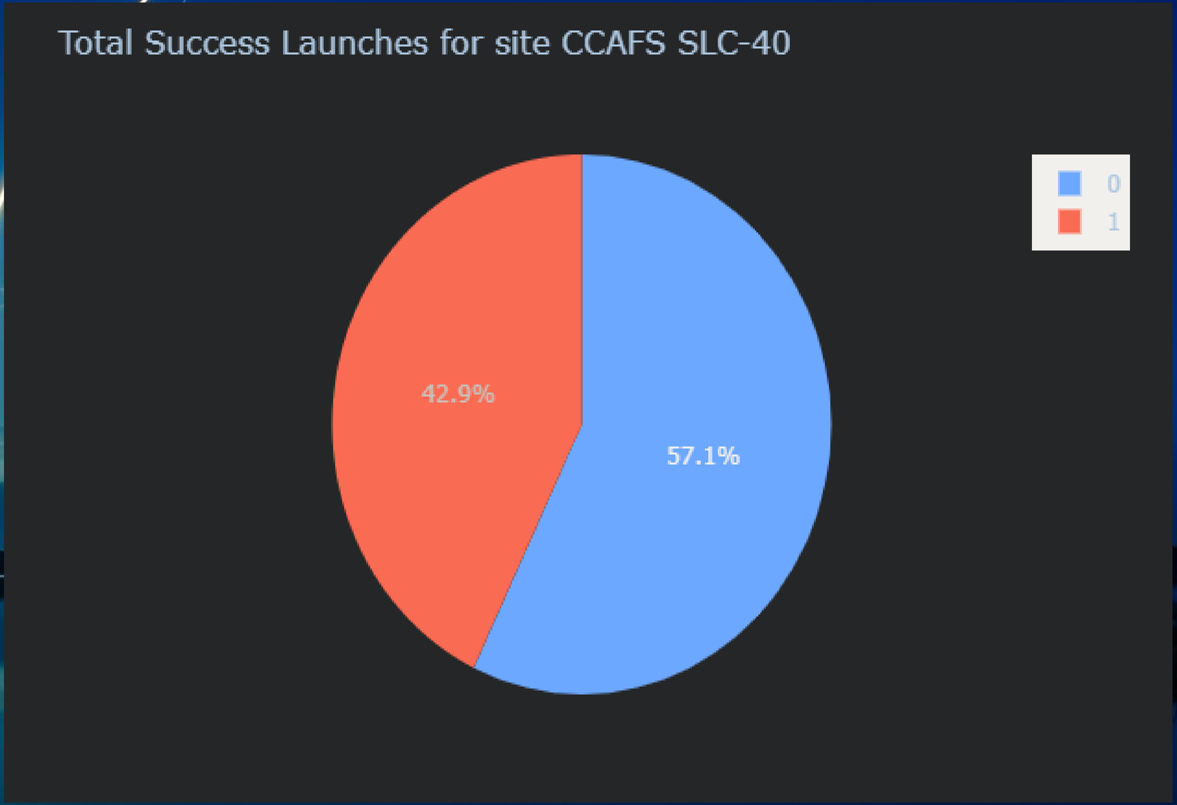
SKILLS NETWORK

## RESULTS

The second one was the CCAFS LC-40 with 73.1% s.r..



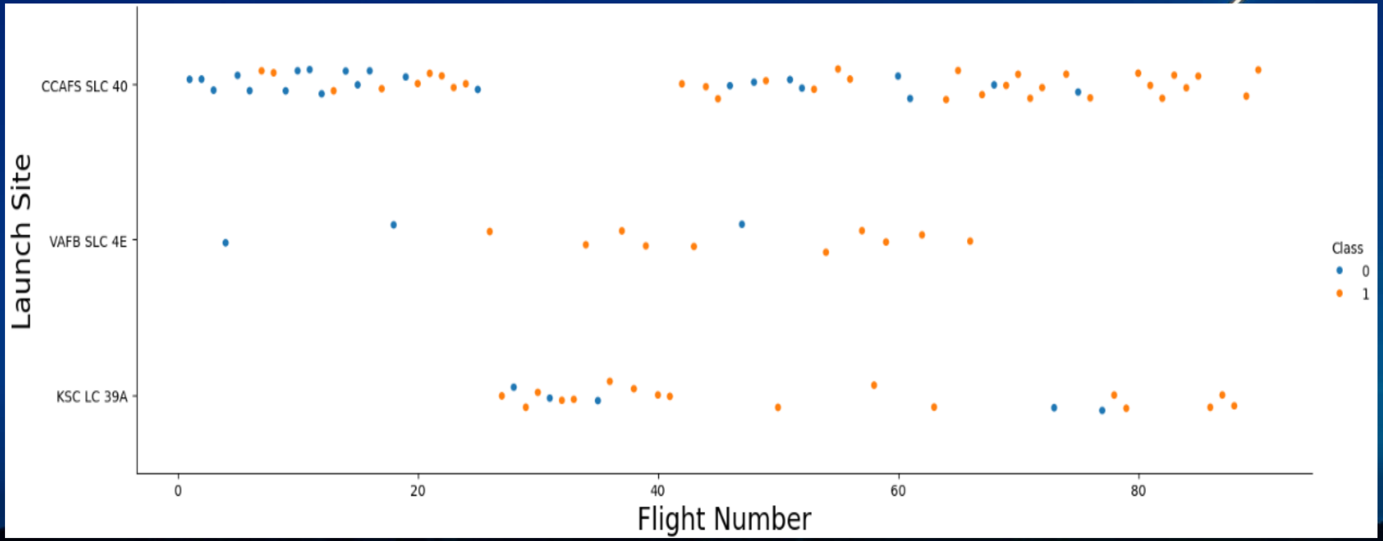
The one with least s.r. was CCAFS SLC-40.



Relationship between Flight Number and Launch Site(Scatter Plot)

# RESULTS

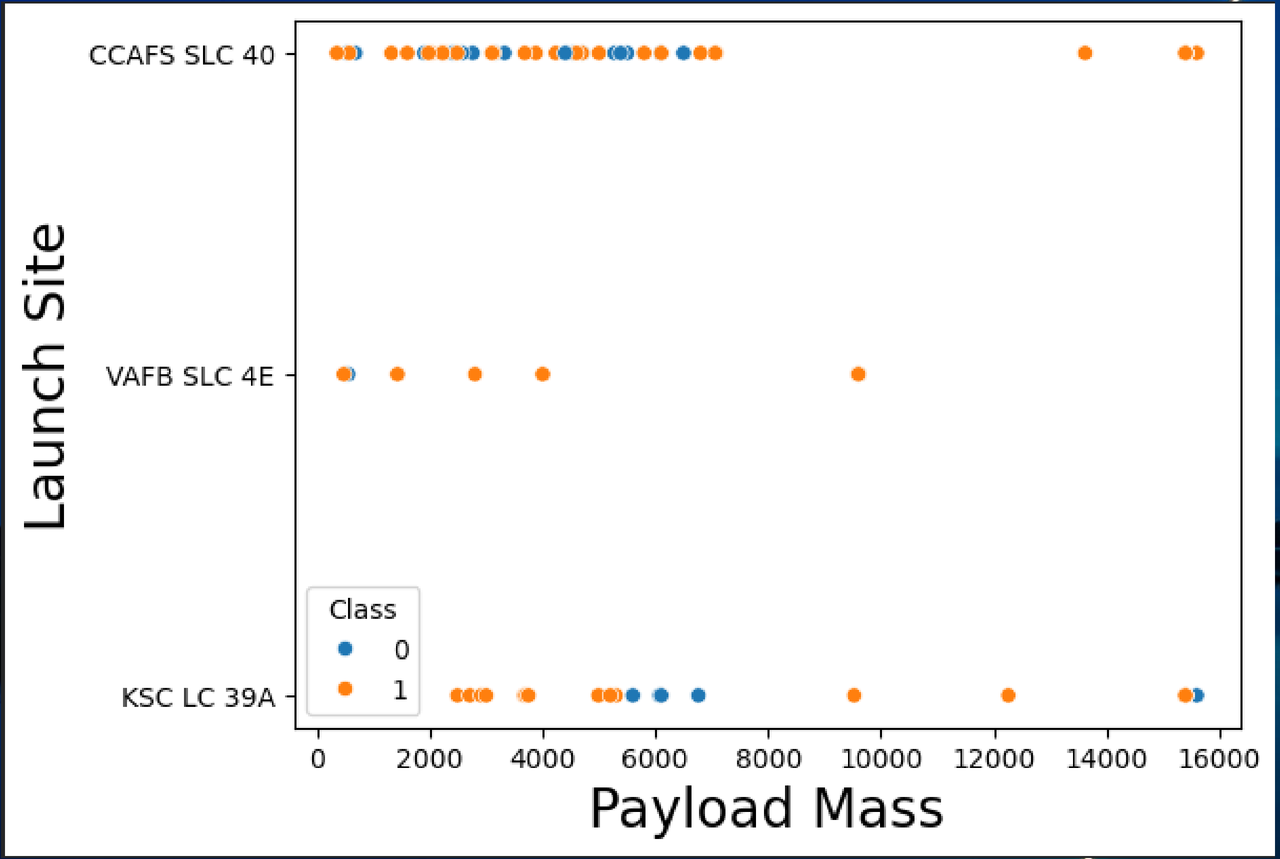
## EXPLORATORY ANALYSIS(Visualization-Plots)



The correlation between Flight Number and Launch Site success is implicit in the analysis. The increasing experience gained over successive flights appears to positively influence success rates, and this is exemplified by the performance of the leading launch site, KSC LC-39 A. The findings suggest that as the number of flights increases, so does the likelihood of success at the chosen launch sites, reflecting the importance of experience in achieving successful space missions.

# RESULTS

## EXPLORATORY ANALYSIS(Visualization-Plots)

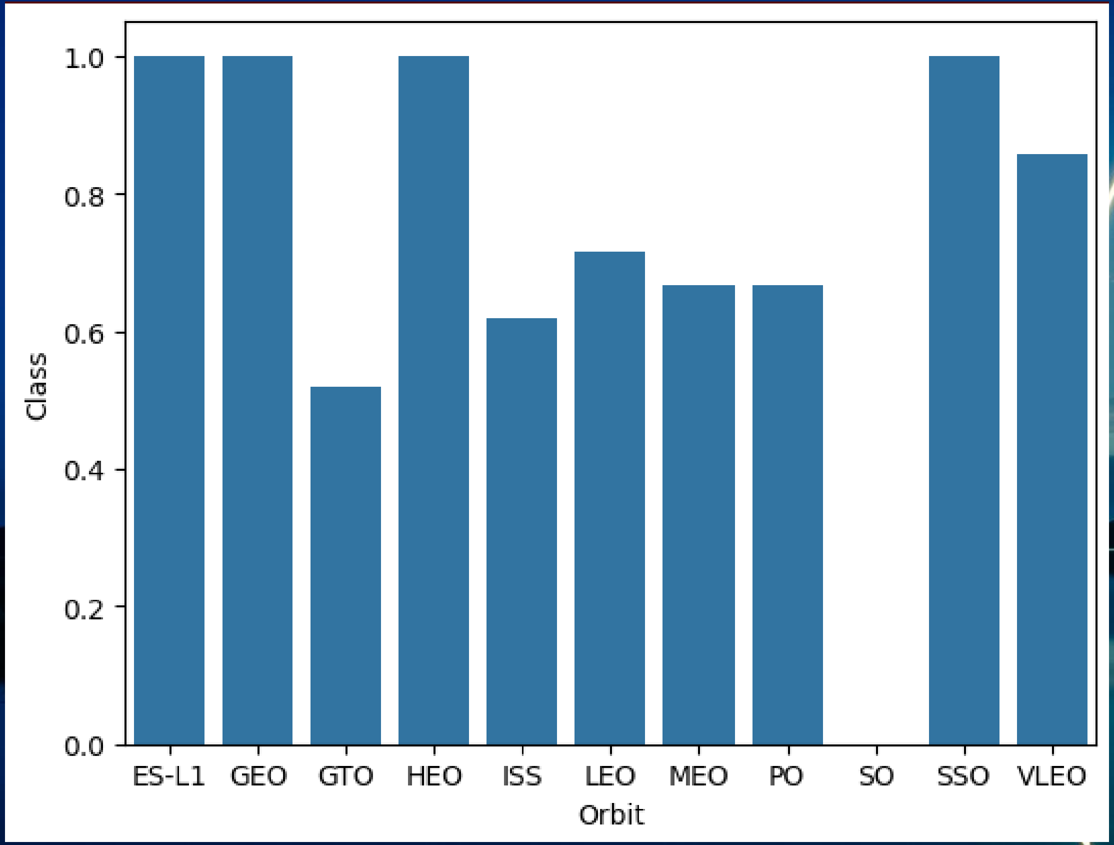


In the context of the text provided, the relationship between Payload Mass and Launch Site is indirectly addressed in Key Finding 8: "Payload Impact on Success." This finding suggests that the success rate for launches with massive payloads (over 4000kg) is comparatively lower than launches with lighter payloads. Although the findings does not explicitly mention a direct correlation between payload mass and launch site, it implies that the choice of launch site may be influenced by the payload mass due to its impact on mission success.

The observation that launches with heavier payloads have lower success rates implies that the selection of a launch site may take into account the payload mass to optimize mission success. Launch sites with certain capabilities or geographical locations may be better suited to accommodate heavier payloads, mitigating the risks associated with launching them.

## RESULTS

### EXPLORATORY ANALYSIS(Visualization-Plots)



- •Orbits like ES-L1, GEO, HEO, and SSO exhibit exceptional 100% success rates.
- •Notably, GEO, HEO, and ES-L1 orbits achieve perfection with only one respective flight each.

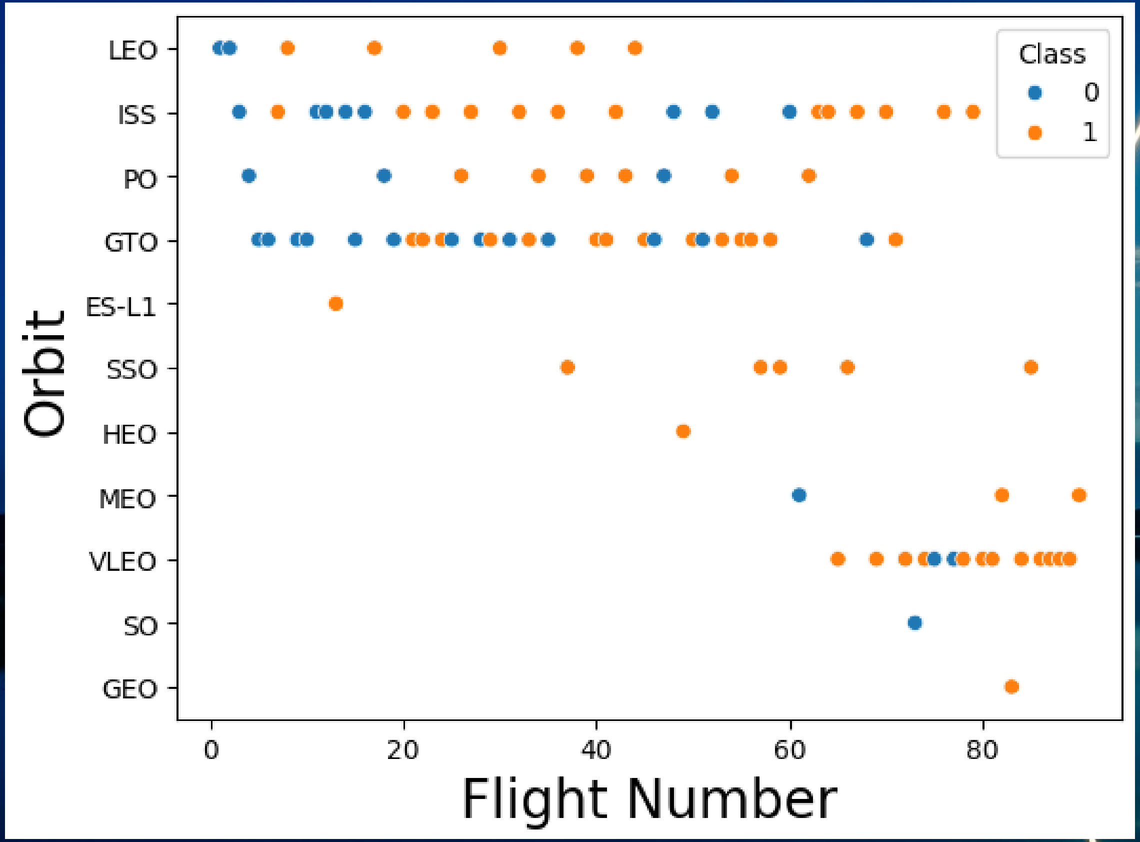
These findings suggest that the success rates are not uniform across all orbits. Certain orbits, such as ES-L1, GEO, HEO, SSO, PO, ISS, and LEO, stand out with higher success rates. Additionally, the correlation between payload characteristics (such as weight) and success rates in specific orbits highlights the nuanced relationship between payload attributes and mission success in different orbital trajectories.

IBM Developer

SKILLS NETWORK

# RESULTS

## EXPLORATORY ANALYSIS(Visualization-Plots)



The 100% success rates observed in GEO, HEO, and ES-L1 orbits can be attributed to the fact that each of these orbits has been attempted only once.

The 100% success rate in SSO is particularly remarkable, given that there have been a total of 5 successful flights into this orbit.

The relationship between Flight Number and Success Rate for GTO appears to be weak, indicating little correlation between these two variables.

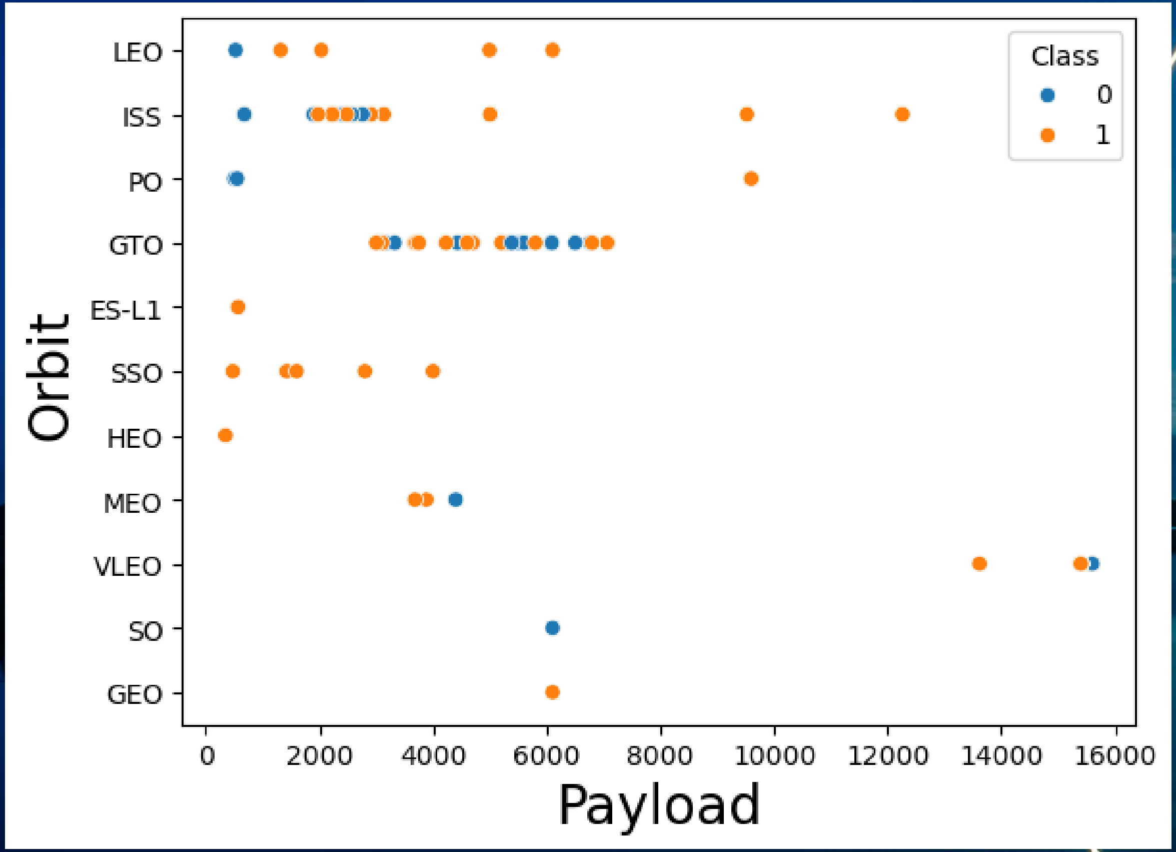
Generally, there is an observable trend: as Flight Number increases, the success rate tends to rise. This trend is most pronounced in the case of LEO, where unsuccessful lanAdings are predominantly associated with lower flight numbers, corresponding to early launches.

IBM Developer

SKILLS NETWORK

## RESULTS

### EXPLORATORY ANALYSIS(Visualization-Plots)



Orbit types, namely PO, ISS, and LEO, exhibit higher success rates when handling heavy payloads. It's essential to note that the dataset for PO is limited, impacting the robustness of this observation.

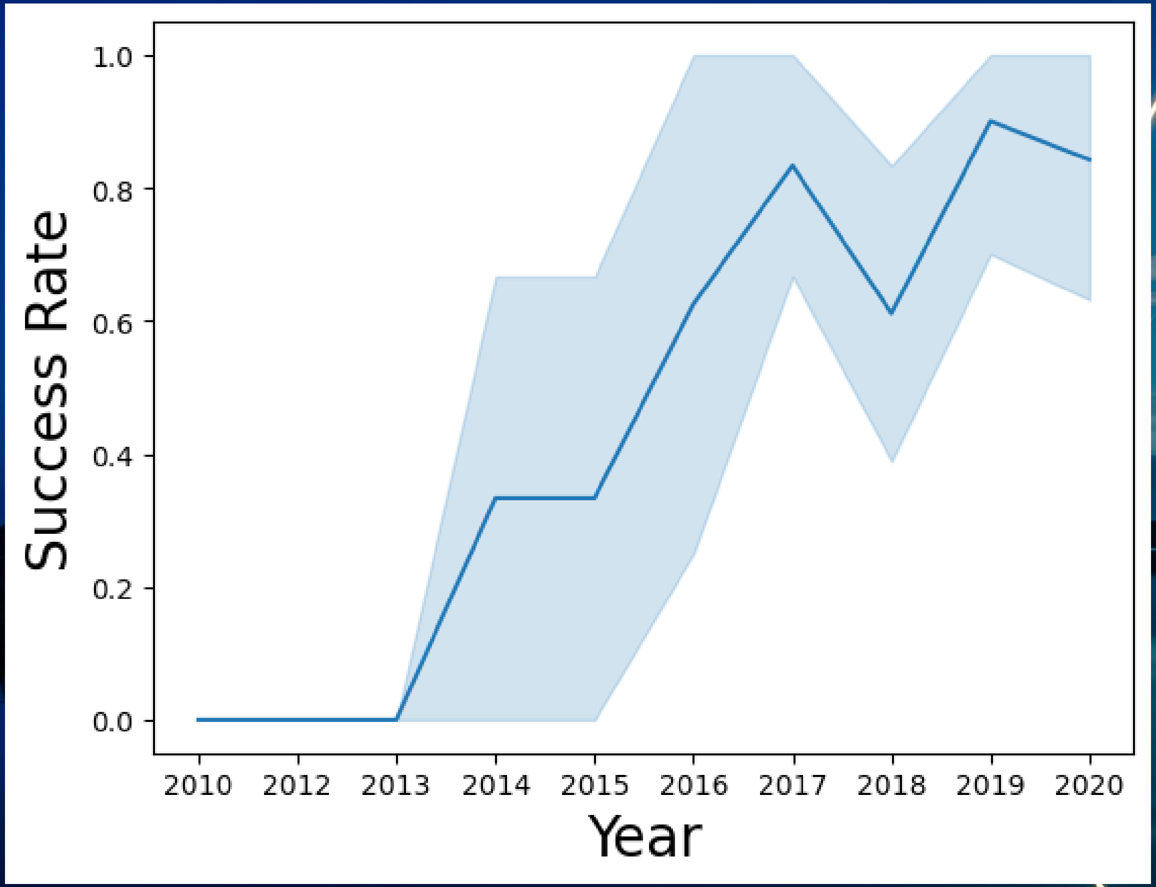
The correlation between payload mass and success rate in the case of GTO remains ambiguous, indicating a need for further exploration to decipher the underlying patterns.

Launches into Very Low Earth Orbit (VLEO) logically involve heavier payloads, aligning with intuitive expectations.

Relationship of Success Rate per Year(LinePlot)

## RESULTS

### EXPLORATORY ANALYSIS(Visualization-Plots)



From 2010 to 2013, all landings experienced a lack of success, resulting in a 0% success rate during this period.

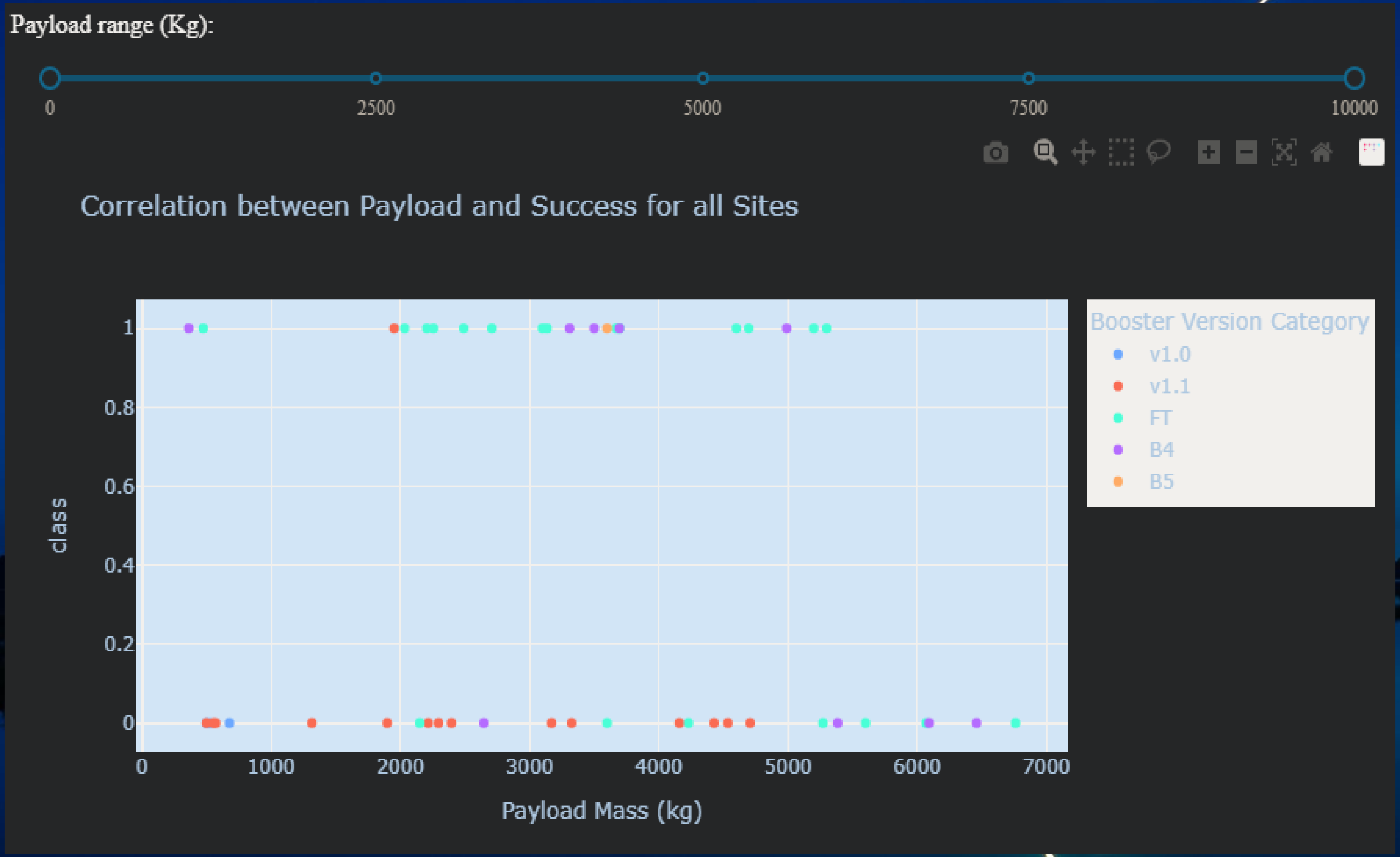
Following 2013, there is a consistent upward trajectory in the success rate, with some fluctuations observed in 2018 and 2020.

Post-2016, the likelihood of success consistently exceeded 50%, indicating a sustained positive trend in mission success probability.

IBM Developer

SKILLS NETWORK

## RESULTS



### DASHBOARD ANALYSIS

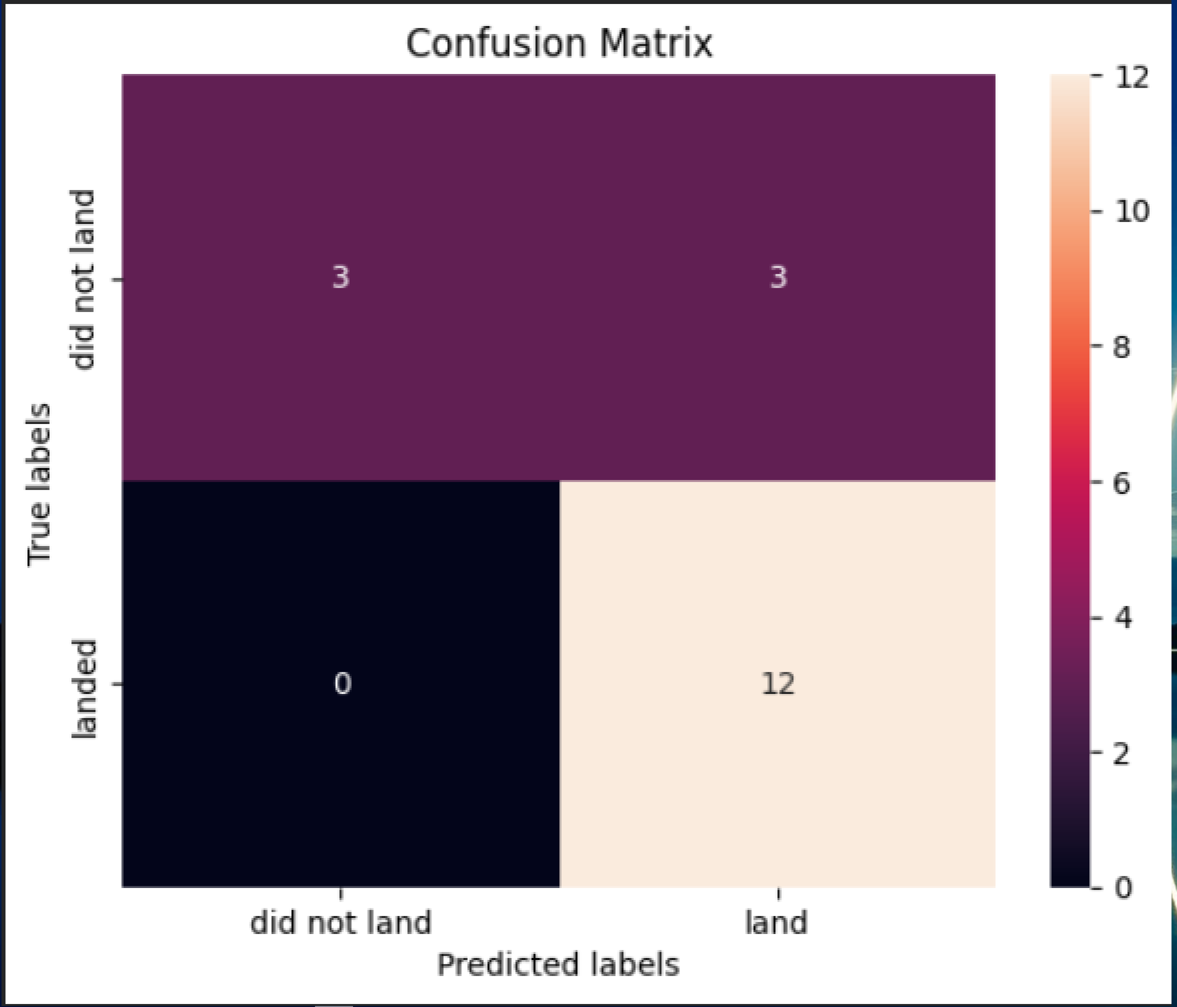
**Conclusions(Scatter Plot):**

- Analyzing the scatter plot, we can clearly see that low Payloads launch have a much better success rate than the ones with high Payload.
- We can also see that some Boosters are not used to high Payload lunch, what can indicate that some Booster can really make a difference in the s.r..

IBM Developer

SKILLS NETWORK

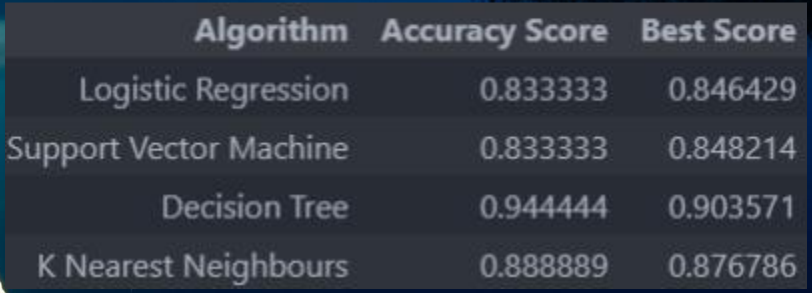
## RESULTS



**PREDICTIVE ANALYSIS (Classification)**

**Conclusions:**

- ❖ **We found out the Decision Tree model has the highest Accuracy and Score(as table below)**
- ❖ **The Confusion Matrix is the same for all the models.**



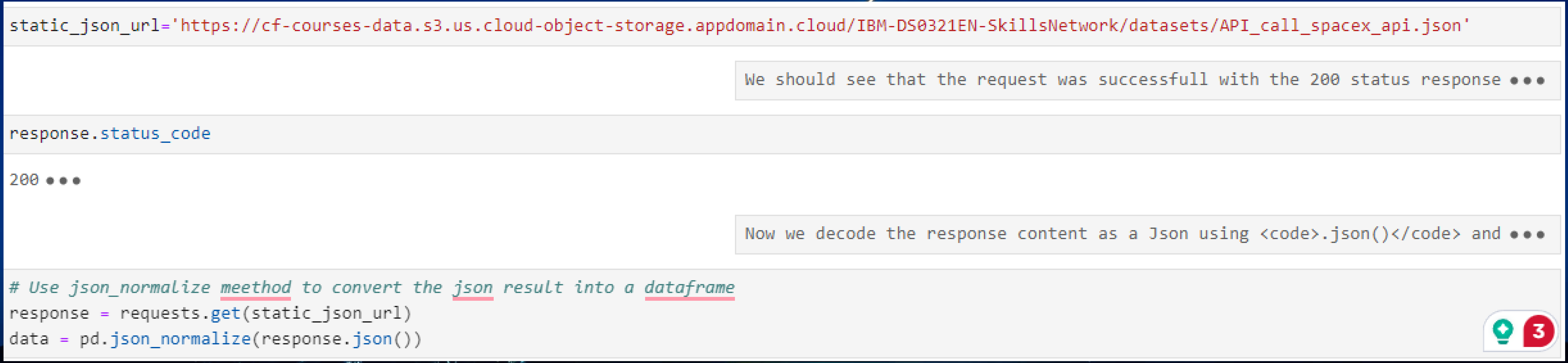
| Algorithm | Accuracy Score | Best Score |
|---|---|---|
| Logistic Regression | 0.833333 | 0.846429 |
| Support Vector Machine | 0.833333 | 0.848214 |
| Decision Tree | 0.944444 | 0.903571 |
| K Nearest Neighbours | 0.888889 | 0.876786 |

IBM Developer

SKILLS NETWORK

## APPENDIX – Data Collection

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



```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

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

```
response.status_code
```

```
200 •••
```

Now we decode the response content as a Json using <code>.json()</code> and •••

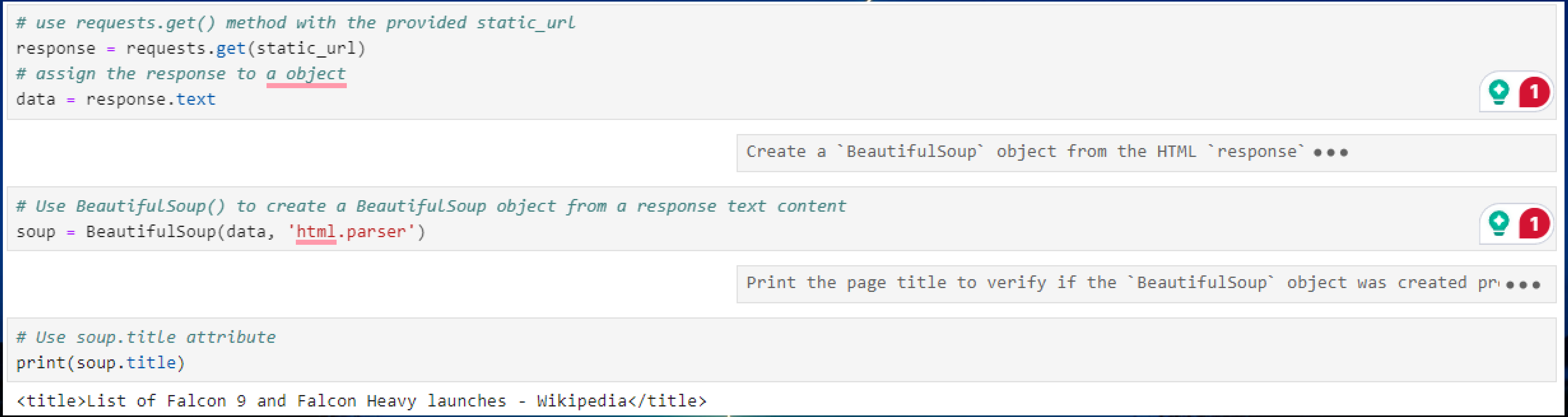
```
# Use json_normalize meethod to convert the json result into a dataframe
response = requests.get(static_json_url)
data = pd.json_normalize(response.json())
```

IBM Developer

SKILLS NETWORK

## APPENDIX – Web Scrapping

### Request the Falcon9 Launch Wiki page from its URL



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

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

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(data, 'html.parser')
```

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

```
# Use soup.title attribute
print(soup.title)
```

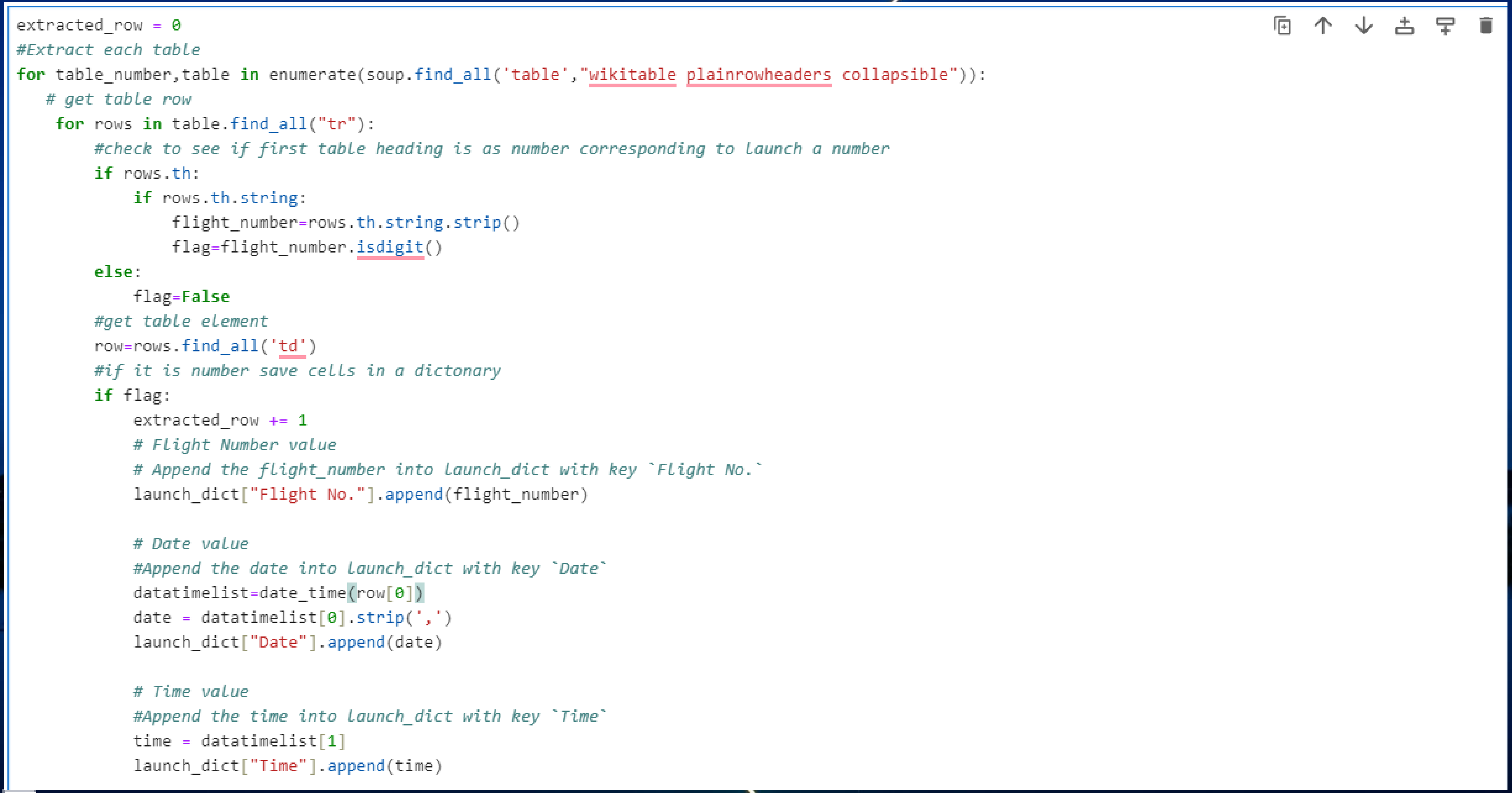
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

IBM Developer

SKILLS NETWORK

# APPENDIX – Web Scrapping

## Creation a data frame by parsing the launch HTML tables(Sample)



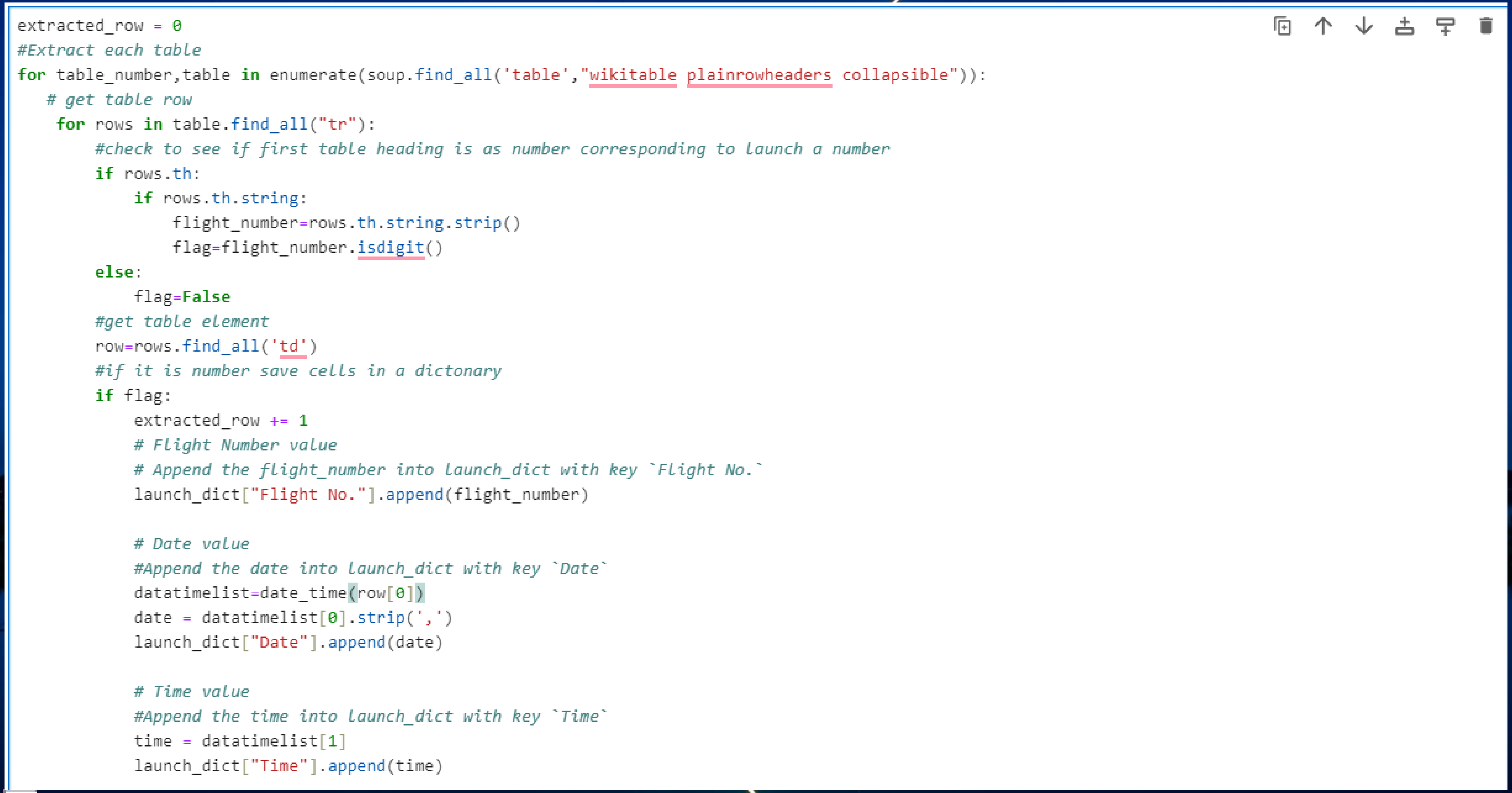
```python
extracted_row = 0
#Extract each table 
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row 
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number 
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element 
        row=rows.find_all('td')
        #if it is number save cells in a dictonary 
        if flag:
            extracted_row += 1
            # Flight Number value
            # Append the flight_number into launch_dict with key `Flight No.`
            launch_dict["Flight No."].append(flight_number)

            # Date value
            #Append the date into launch_dict with key `Date`
            datatimelist=date_time(row[0])
            date = datatimelist[0].strip(',')
            launch_dict["Date"].append(date)

            # Time value
            #Append the time into launch_dict with key `Time`
            time = datatimelist[1]
            launch_dict["Time"].append(time)
```

IBM Developer

SKILLS NETWORK

## APPENDIX – Web Scrapping

### Creation a data frame by parsing the launch HTML tables(Sample)



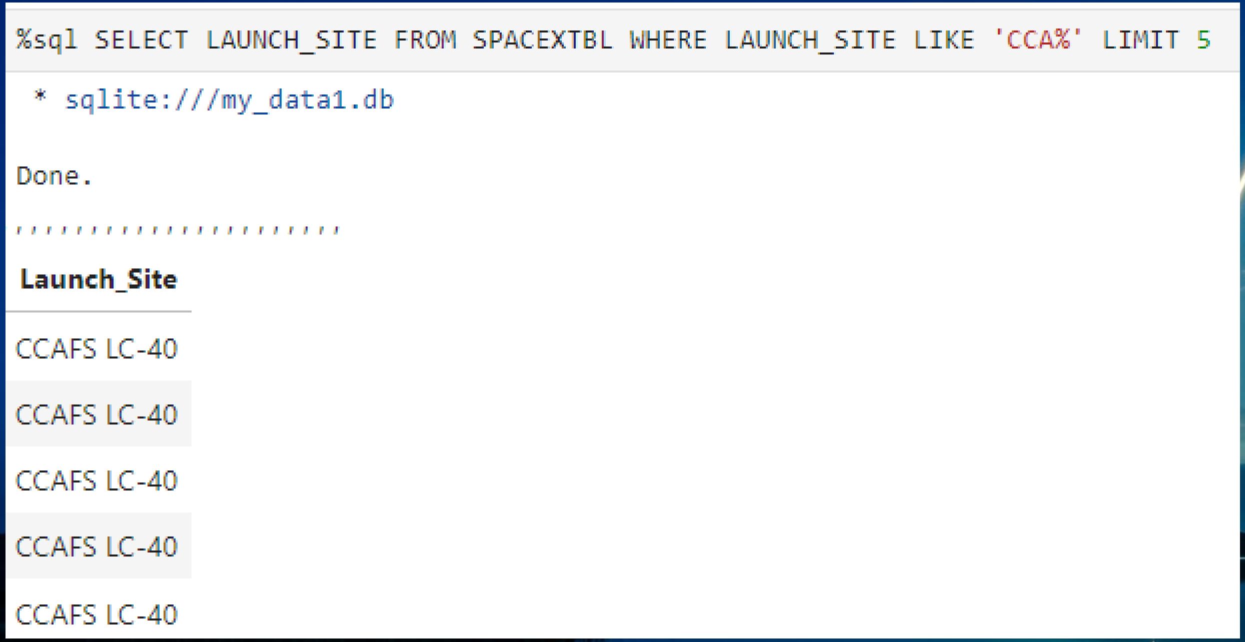
```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # Append the flight_number into launch_dict with key `Flight No.`
            launch_dict["Flight No."].append(flight_number)

            # Date value
            #Append the date into launch_dict with key `Date`
            datatimelist=date_time(row[0])
            date = datatimelist[0].strip(',')
            launch_dict["Date"].append(date)

            # Time value
            #Append the time into launch_dict with key `Time`
            time = datatimelist[1]
            launch_dict["Time"].append(time)
```

IBM Developer

SKILLS NETWORK

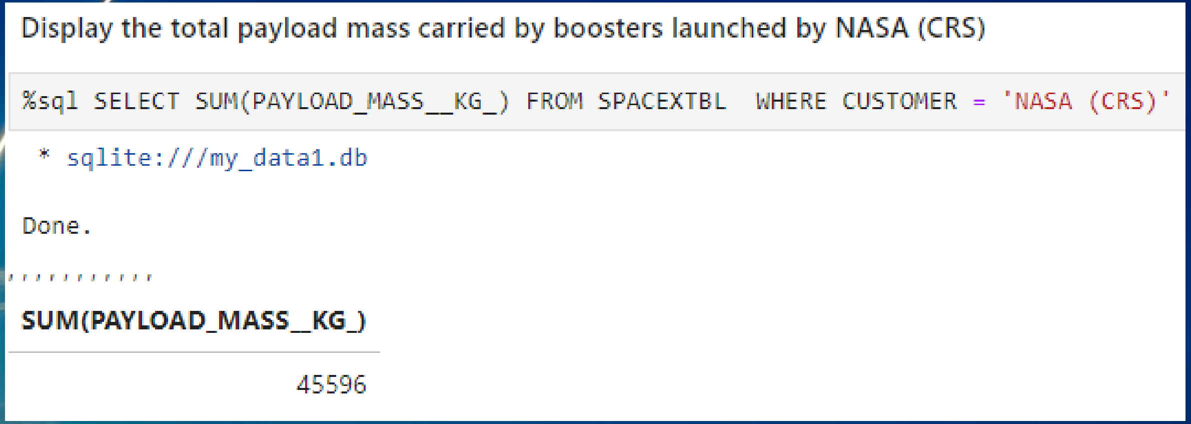
# APPENDIX – EDA SQLite

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



```
%sql SELECT LAUNCH_SITE FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5

 * sqlite:///my_data1.db

Done.
```

| Launch_Site |
|---|
| CCAFS LC-40 |
| CCAFS LC-40 |
| CCAFS LC-40 |
| CCAFS LC-40 |
| CCAFS LC-40 |



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

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

 * sqlite:///my_data1.db

Done.
```

| SUM(PAYLOAD_MASS__KG_) |
|---|
| 45596 |

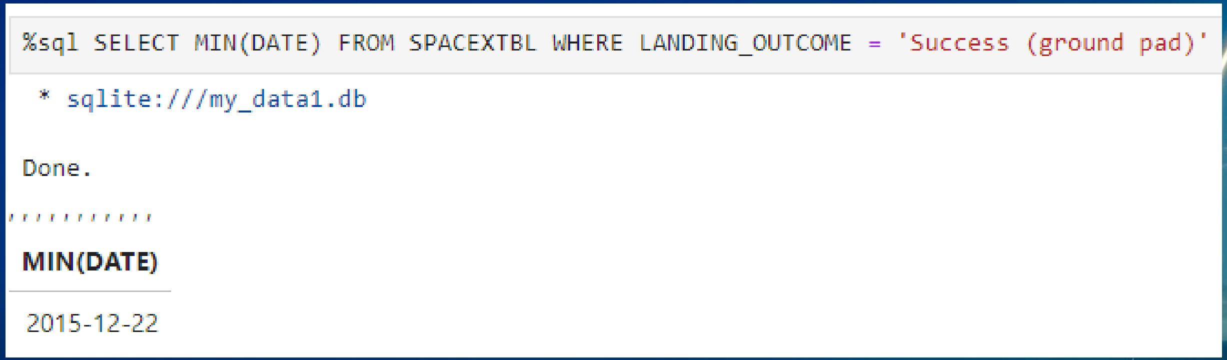
Display the total payload mass carried by boosters launched by NASA (CRS)

IBM Developer

SKILLS NETWORK

# APPENDIX – EDA SQLite

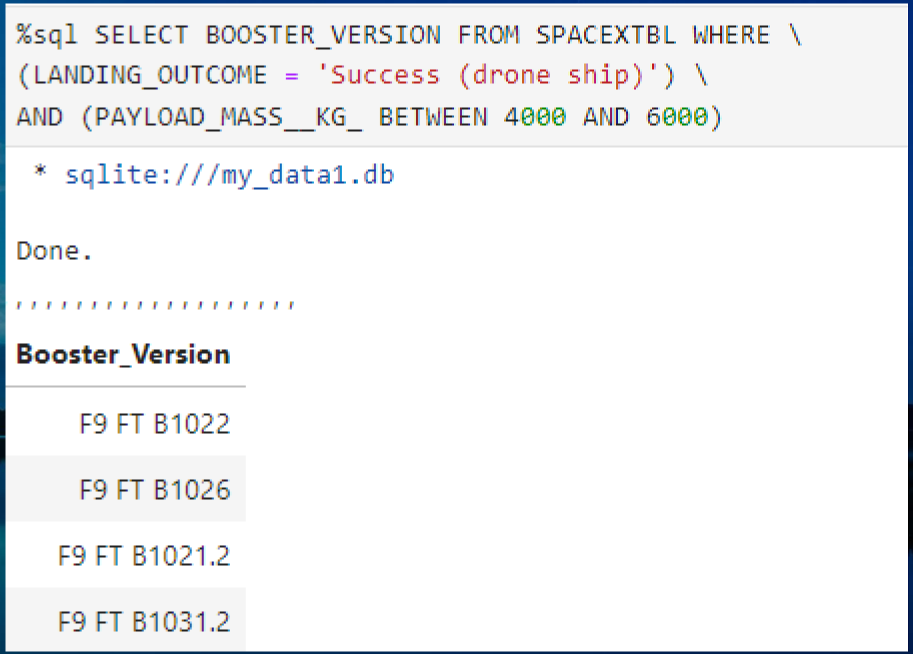
List the date when the first succesful landing outcome in ground pad was acheived



```
%sql SELECT MIN(DATE) FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (ground pad)'

 * sqlite:///my_data1.db

Done.
```

| MIN(DATE) |
|---|
| 2015-12-22 |

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000



```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE \
(LANDING_OUTCOME = 'Success (drone ship)') \
AND (PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000)

 * sqlite:///my_data1.db

Done.
```

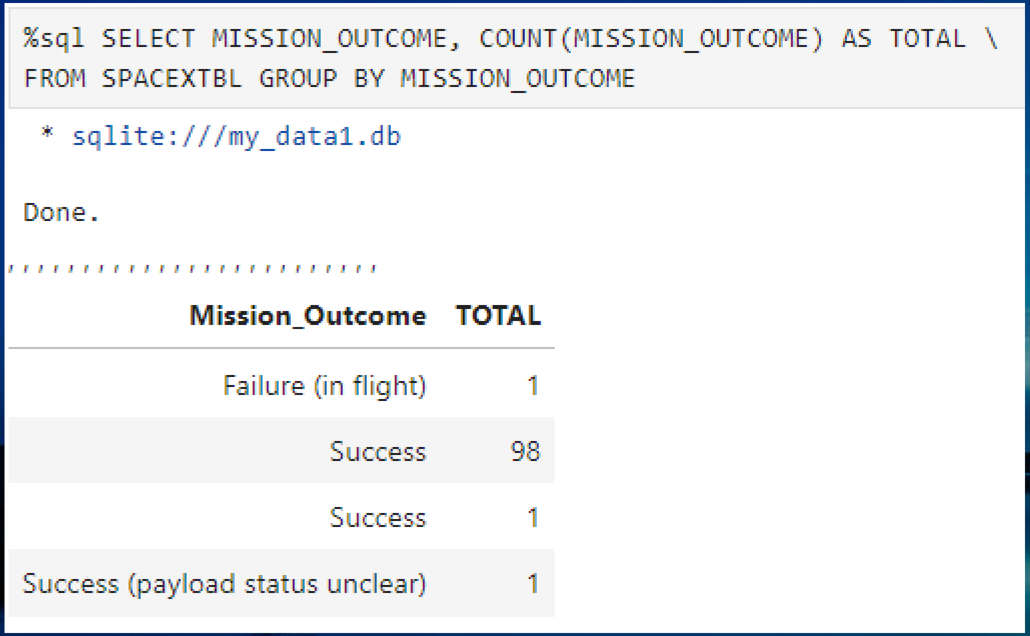
| Booster_Version |
|---|
| F9 FT B1022 |
| F9 FT B1026 |
| F9 FT B1021.2 |
| F9 FT B1031.2 |

IBM Developer

SKILLS NETWORK

# APPENDIX – EDA SQLite

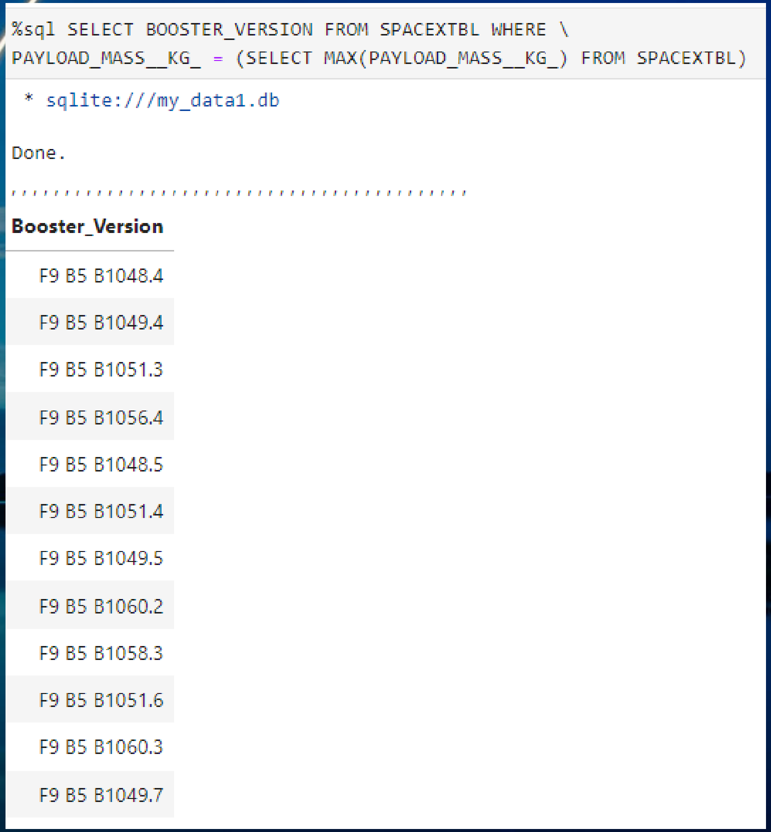
List the total number of successful and failure mission outcomes



```
%sql SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL \
FROM SPACEXTBL GROUP BY MISSION_OUTCOME

 * sqlite:///my_data1.db

Done.
```

| Mission_Outcome | TOTAL |
|---|---|
| Failure (in flight) | 1 |
| Success | 98 |
| Success | 1 |
| Success (payload status unclear) | 1 |

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery



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

 * sqlite:///my_data1.db

Done.
```

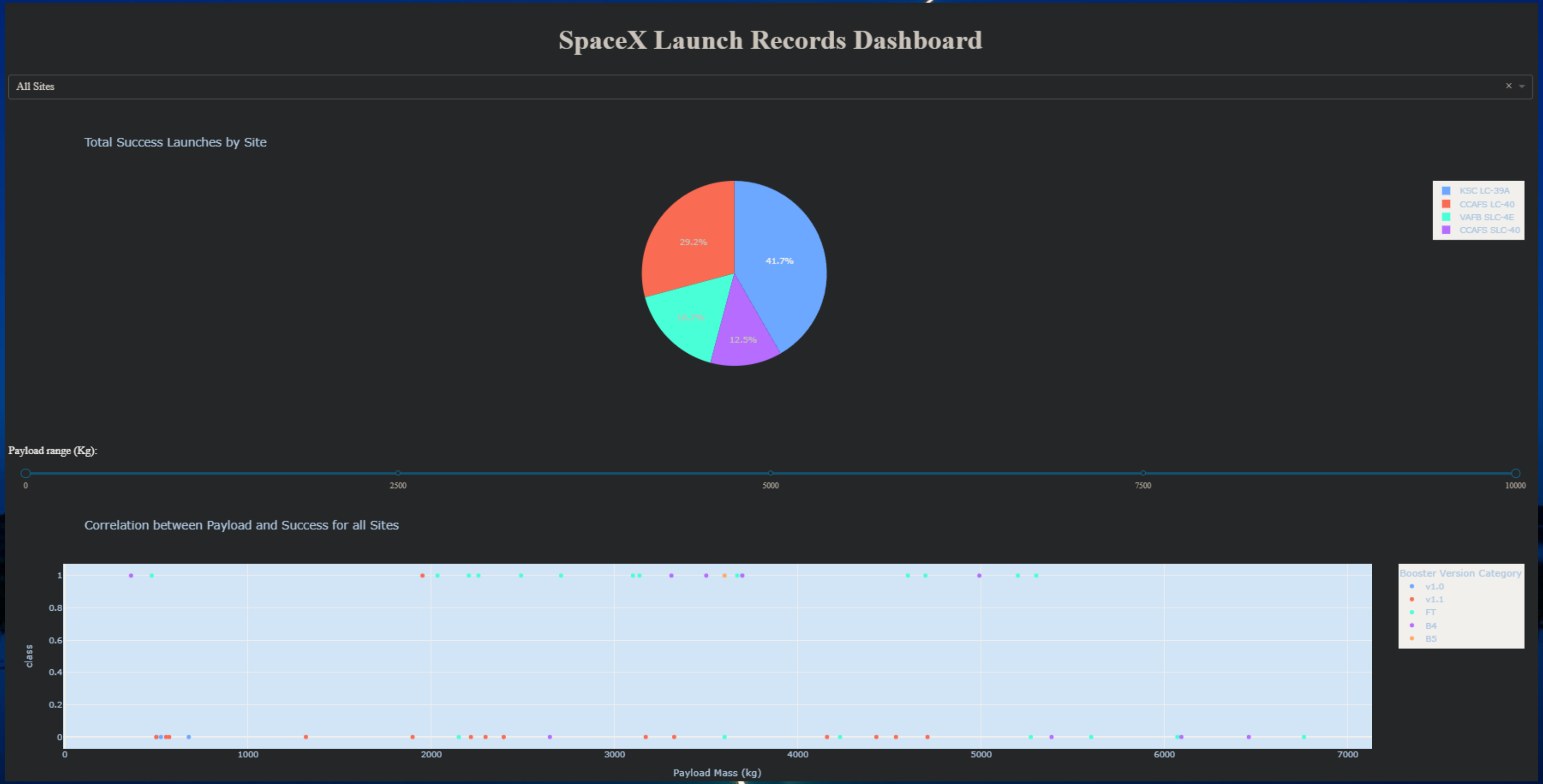
| Booster_Version |
|---|
| F9 B5 B1048.4 |
| F9 B5 B1049.4 |
| F9 B5 B1051.3 |
| F9 B5 B1056.4 |
| F9 B5 B1048.5 |
| F9 B5 B1051.4 |
| F9 B5 B1049.5 |
| F9 B5 B1060.2 |
| F9 B5 B1058.3 |
| F9 B5 B1051.6 |
| F9 B5 B1060.3 |
| F9 B5 B1049.7 |

IBM Developer

SKILLS NETWORK

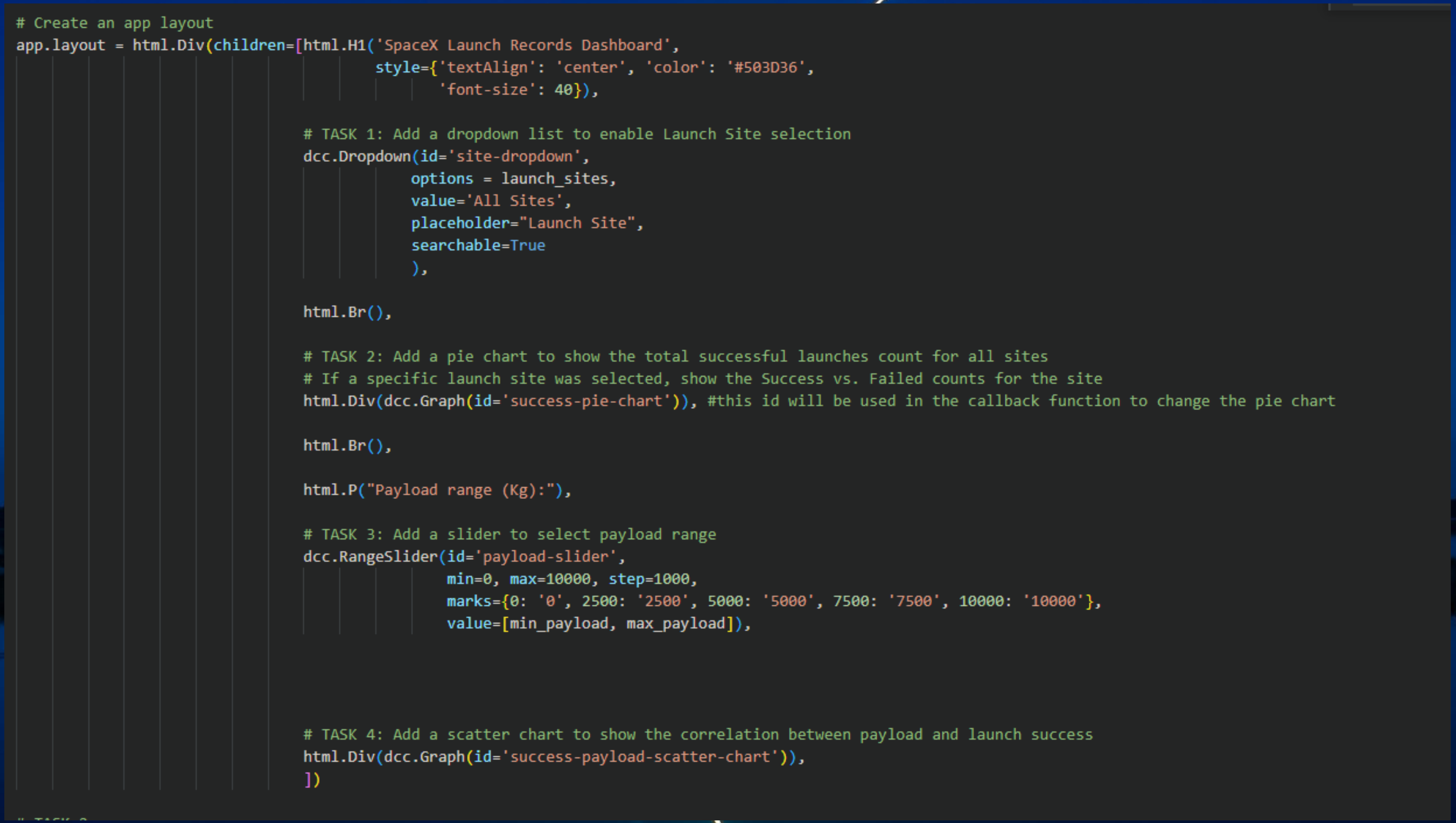
# APPENDIX – EDA With Visualization

## Full Dashboard view



# APPENDIX – EDA With Visualization

## Dashboard Code



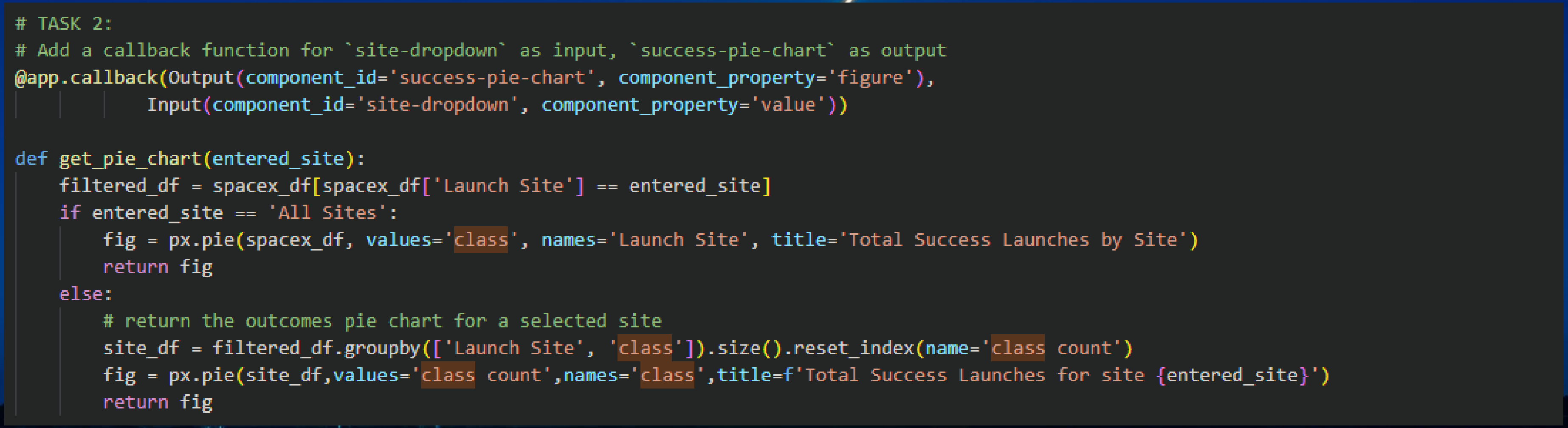
```
# Create an app layout
app.layout = html.Div(children=[html.H1('SpaceX Launch Records Dashboard',
                                        style={'textAlign': 'center', 'color': '#503D36',
                                               'font-size': 40}),

                                # TASK 1: Add a dropdown list to enable Launch Site selection
                                dcc.Dropdown(id='site-dropdown',
                                            options = launch_sites,
                                            value='All Sites',
                                            placeholder="Launch Site",
                                            searchable=True
                                            ),

                                html.Br(),

                                # TASK 2: Add a pie chart to show the total successful launches count for all sites
                                # If a specific launch site was selected, show the Success vs. Failed counts for the site
                                html.Div(dcc.Graph(id='success-pie-chart')), #this id will be used in the callback function to change the pie chart

                                html.Br(),

                                html.P("Payload range (Kg):"),

                                # TASK 3: Add a slider to select payload range
                                dcc.RangeSlider(id='payload-slider',
                                                min=0, max=10000, step=1000,
                                                marks={0: '0', 2500: '2500', 5000: '5000', 7500: '7500', 10000: '10000'},
                                                value=[min_payload, max_payload]),



                                # TASK 4: Add a scatter chart to show the correlation between payload and launch success
                                html.Div(dcc.Graph(id='success-payload-scatter-chart')),
                                ])
```

IBM Developer

SKILLS NETWORK

# APPENDIX – EDA With Visualization

## Dashboard Code



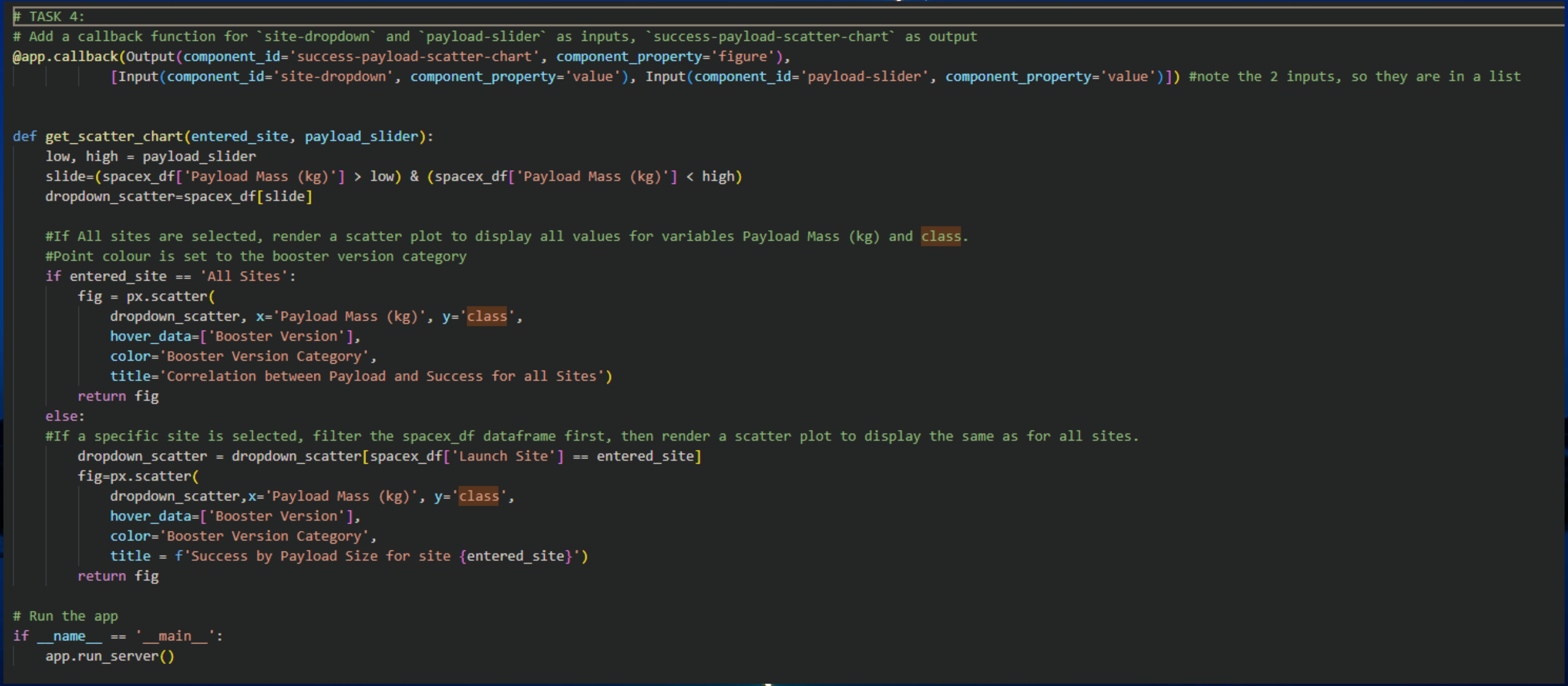
```python
# TASK 2:
# Add a callback function for `site-dropdown` as input, `success-pie-chart` as output
@app.callback(Output(component_id='success-pie-chart', component_property='figure'),
              Input(component_id='site-dropdown', component_property='value'))

def get_pie_chart(entered_site):
    filtered_df = spacex_df[spacex_df['Launch Site'] == entered_site]
    if entered_site == 'All Sites':
        fig = px.pie(spacex_df, values='class', names='Launch Site', title='Total Success Launches by Site')
        return fig
    else:
        # return the outcomes pie chart for a selected site
        site_df = filtered_df.groupby(['Launch Site', 'class']).size().reset_index(name='class count')
        fig = px.pie(site_df,values='class count',names='class',title=f'Total Success Launches for site {entered_site}')
        return fig
```

IBM Developer

SKILLS NETWORK

# APPENDIX – EDA With Visualization

## Dashboard Code



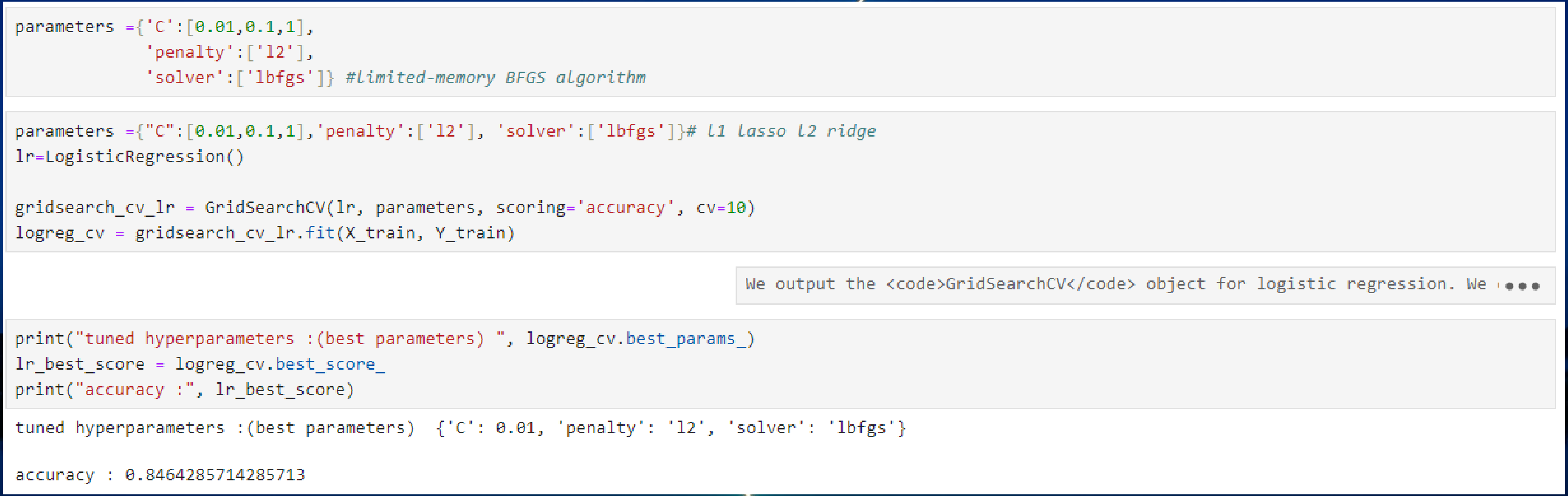
```python
# TASK 4:
# Add a callback function for `site-dropdown` and `payload-slider` as inputs, `success-payload-scatter-chart` as output
@app.callback(Output(component_id='success-payload-scatter-chart', component_property='figure'),
            [Input(component_id='site-dropdown', component_property='value'), Input(component_id='payload-slider', component_property='value')]) #note the 2 inputs, so they are in a list


def get_scatter_chart(entered_site, payload_slider):
    low, high = payload_slider
    slide=(spacex_df['Payload Mass (kg)'] > low) & (spacex_df['Payload Mass (kg)'] < high)
    dropdown_scatter=spacex_df[slide]

    #If All sites are selected, render a scatter plot to display all values for variables Payload Mass (kg) and class.
    #Point colour is set to the booster version category
    if entered_site == 'All Sites':
        fig = px.scatter(
            dropdown_scatter, x='Payload Mass (kg)', y='class',
            hover_data=['Booster Version'],
            color='Booster Version Category',
            title='Correlation between Payload and Success for all Sites')
        return fig
    else:
    #If a specific site is selected, filter the spacex_df dataframe first, then render a scatter plot to display the same as for all sites.
        dropdown_scatter = dropdown_scatter[spacex_df['Launch Site'] == entered_site]
        fig=px.scatter(
            dropdown_scatter,x='Payload Mass (kg)', y='class',
            hover_data=['Booster Version'],
            color='Booster Version Category',
            title = f'Success by Payload Size for site {entered_site}')
        return fig

# Run the app
if __name__ == '__main__':
    app.run_server()
```

IBM Developer

SKILLS NETWORK

# APPENDIX – Predictive Analysis (Classification)

## Logistic Regression Code



```
parameters ={'C':[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']} #limited-memory BFGS algorithm
```

```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()

gridsearch_cv_lr = GridSearchCV(lr, parameters, scoring='accuracy', cv=10)
logreg_cv = gridsearch_cv_lr.fit(X_train, Y_train)
```

```
We output the <code>GridSearchCV</code> object for logistic regression. We
```

```
print("tuned hyperparameters :(best parameters) ", logreg_cv.best_params_)
lr_best_score = logreg_cv.best_score_
print("accuracy :", lr_best_score)
```

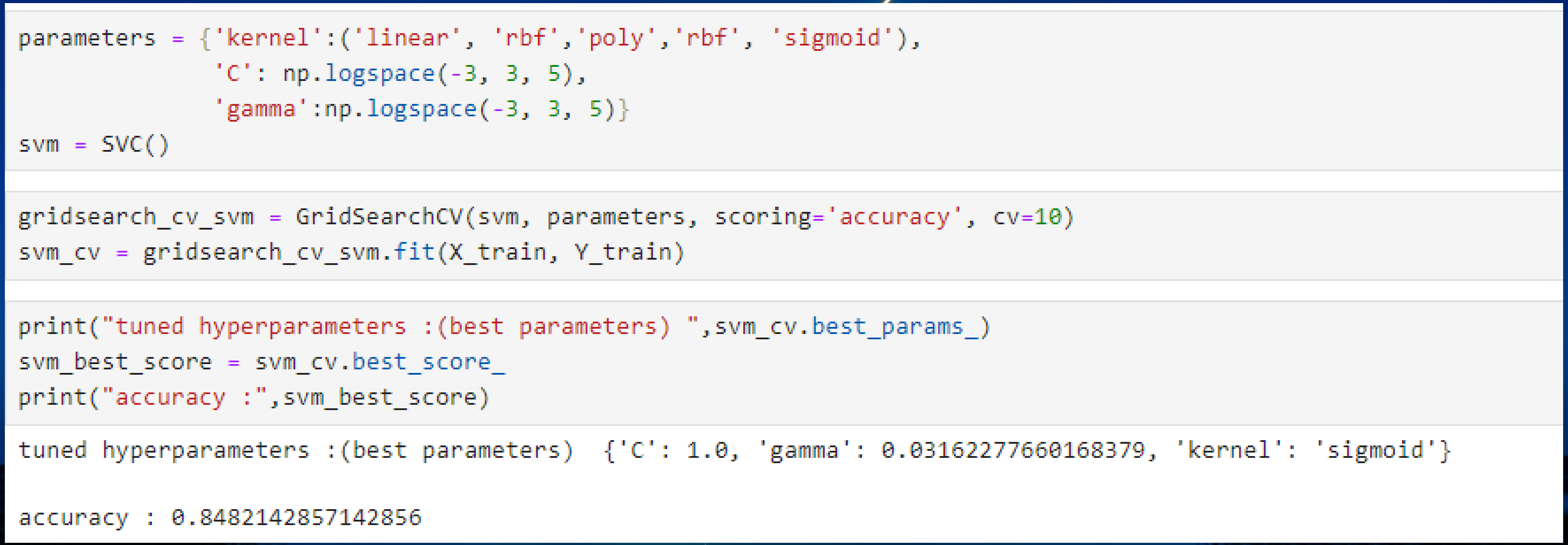
```
tuned hyperparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}

accuracy : 0.8464285714285713
```

IBM Developer

SKILLS NETWORK

# APPENDIX – Predictive Analysis (Classification)

## Support Vector Machine Code



```
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
```

```
gridsearch_cv_svm = GridSearchCV(svm, parameters, scoring='accuracy', cv=10)
svm_cv = gridsearch_cv_svm.fit(X_train, Y_train)
```

```
print("tuned hyperparameters :(best parameters) ",svm_cv.best_params_)
svm_best_score = svm_cv.best_score_
print("accuracy :",svm_best_score)
```

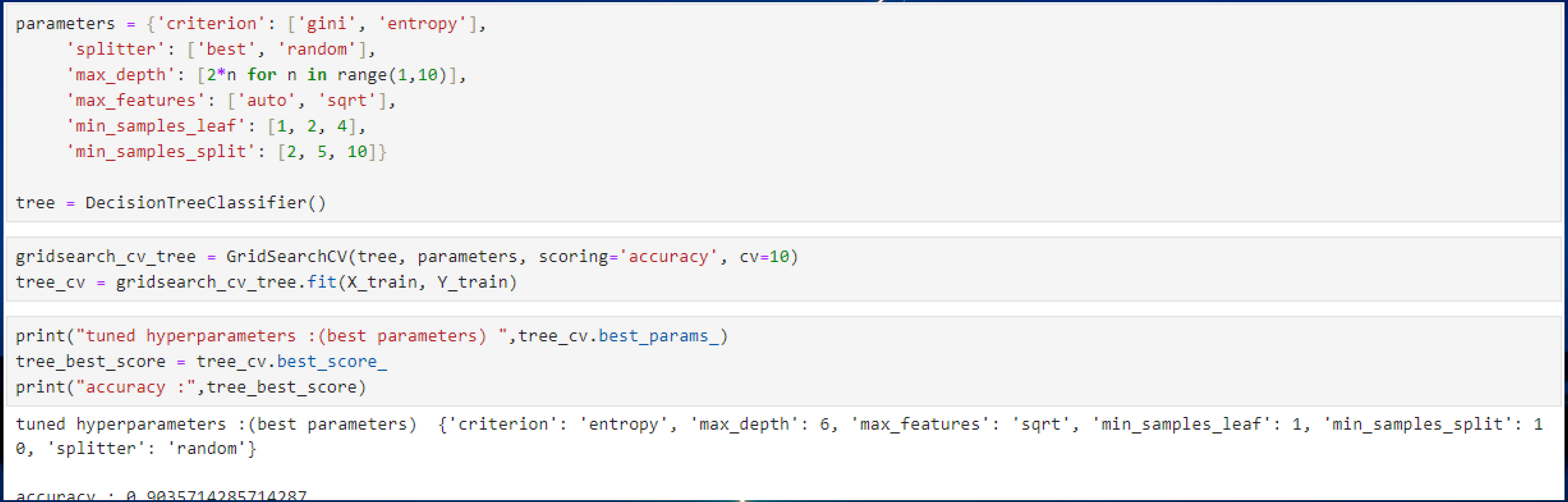
```
tuned hyperparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}

accuracy : 0.8482142857142856
```

IBM Developer

SKILLS NETWORK

# APPENDIX – Predictive Analysis (Classification)

## Decision Tree Code



```
parameters = {'criterion': ['gini', 'entropy'],
     'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],
     'max_features': ['auto', 'sqrt'],
     'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10]}

tree = DecisionTreeClassifier()
```

```
gridsearch_cv_tree = GridSearchCV(tree, parameters, scoring='accuracy', cv=10)
tree_cv = gridsearch_cv_tree.fit(X_train, Y_train)
```

```
print("tuned hyperparameters :(best parameters) ",tree_cv.best_params_)
tree_best_score = tree_cv.best_score_
print("accuracy :",tree_best_score)
```

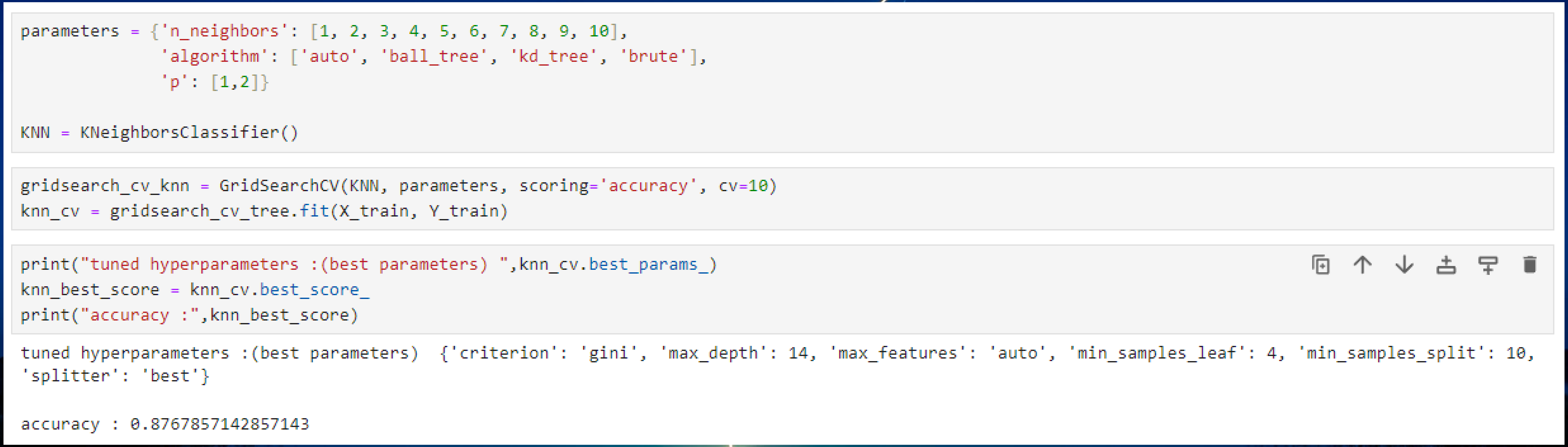
```
tuned hyperparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 1
0, 'splitter': 'random'}

accuracy : 0.9035714285714287
```

IBM Developer

SKILLS NETWORK

# APPENDIX – Predictive Analysis (Classification)

## K-Nearest Neighbour Code



```
parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
              'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
              'p': [1,2]}

KNN = KNeighborsClassifier()
```

```
gridsearch_cv_knn = GridSearchCV(KNN, parameters, scoring='accuracy', cv=10)
knn_cv = gridsearch_cv_tree.fit(X_train, Y_train)
```

```
print("tuned hyperparameters :(best parameters) ",knn_cv.best_params_)
knn_best_score = knn_cv.best_score_
print("accuracy :",knn_best_score)
```

```
tuned hyperparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 14, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 10,
'splitter': 'best'}

accuracy : 0.8767857142857143
```

IBM Developer

SKILLS NETWORK

# CONCLUSION

- Experience and Success: With an increase in the number of flights, there is a notable enhancement in success rates at launch sites. Initial flights tend to have lower success rates, but as experience grows, so does the overall success rate.
- Unsuccessful Landings (2010-2013): During the period from 2010 to 2013, all landings recorded a 0% success rate, indicating challenges and difficulties during that timeframe.
- Post-2013 Success Growth: Subsequent to 2013, there is a general upward trend in success rates, despite minor setbacks in 2018 and 2020.
- Consistent Success Since 2016: From 2016 onwards, the likelihood of success consistently exceeds 50%, reflecting a positive and stable trajectory.
- Orbit Success Rates: ES-L1, GEO, HEO, and SSO orbits showcase an exceptional 100% success rate. Notably, GEO, HEO, and ES-L1 orbits achieve this perfection with only one respective flight each.
- Impressive SSO Success: The 100% success rate in SSO is particularly noteworthy, with a total of five successful flights.
- Payload Influence on PO, ISS, LEO Orbits: PO, ISS, and LEO orbits exhibit higher success rates, especially with heavier payloads. This aligns with the intuitive correlation between launches to Very Low Earth Orbit (VLEO) and heavier payloads.
- Leading Launch Site (KSC LC-39 A): KSC LC-39 A stands out as the most successful launch site, contributing to 41.7% of total successful launches and boasting the highest success rate at 76.9%.
- Payload Impact on Success: The success rate for massive payloads (over 4000kg) is observed to be lower compared to launches with lighter payloads.
- Top-Performing Model: The Decision Tree model emerges as the best-performing classification model, achieving an impressive accuracy of 94.44%.

IBM Developer

SKILLS NETWORK