IBM Capstone Project

Akshda

OUTLINE

INTRODUCTION

- Project background and context
  - SpaceX advertises Falcon 9 rocket launches for $62 million on its website.
  - Other providers charge upwards of $165 million for each rocket launch.
  - SpaceX's cost savings are mainly due to the ability to reuse the first stage of the Falcon 9 rocket.
  - Predicting whether the first stage will land successfully can determine the cost of a launch.
  - The project's objective is to create a machine learning pipeline for predicting the successful landing of the first stage.
  - The prediction model can be used by an alternate company to compete with SpaceX in bidding for rocket launch contracts.
- Questions Asked
  - What are the determining factors for a successful rocket landing?
  - What are the necessary operating conditions to ensure a successful landing program?

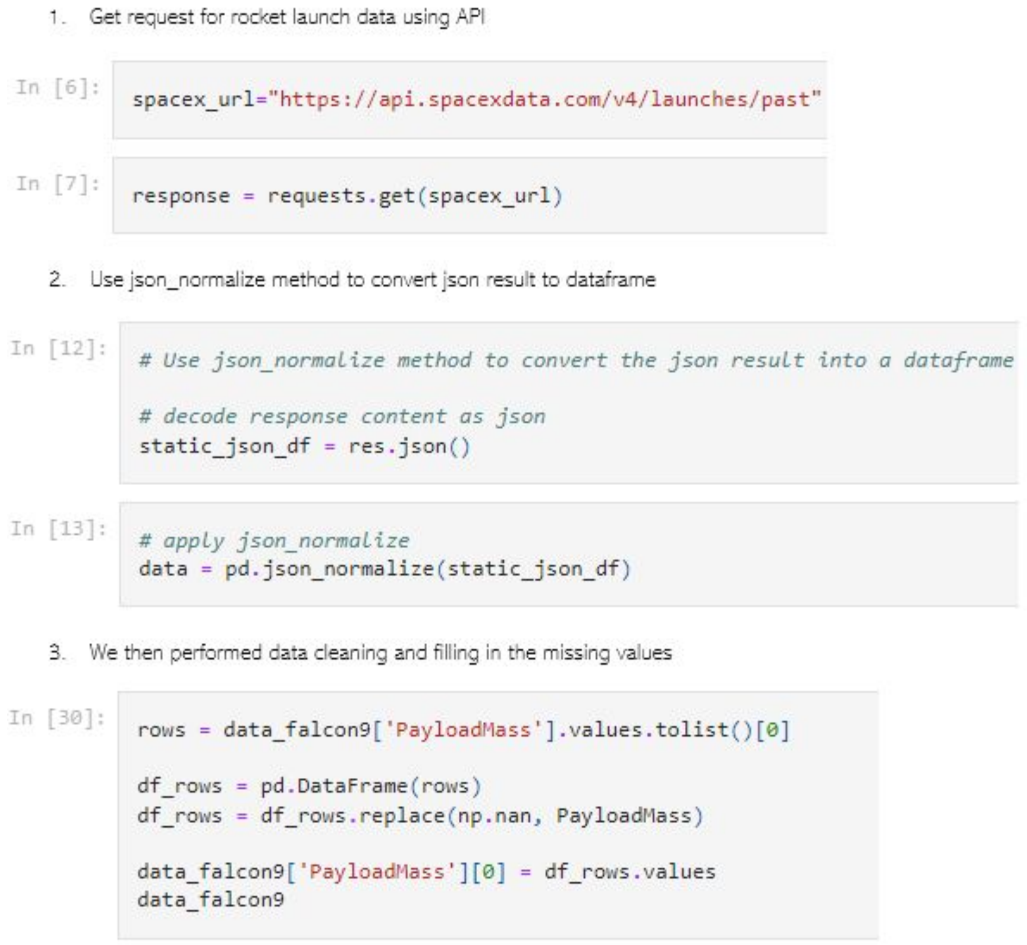
METHODOLOGY

DATA COLLECTION

- The data was collected using various methods
  - The data collection process involved making a GET request to the SpaceX API.
  - The response content was decoded as JSON using the .json() function call.
  - The JSON data was then transformed into a pandas dataframe using the .json_normalize() function.
  - Data cleaning procedures were implemented, including checking for missing values and filling them in as needed.
  - Web scraping techniques were utilized to extract Falcon 9 launch records from Wikipedia.
  - BeautifulSoup was used to scrape the launch records as an HTML table.
  - The scraped table was parsed and converted into a pandas dataframe for further analysis and future use.

DATA COLLECTION CONTD.

- The SpaceX API was utilized to make a GET request for data collection.
- The collected data underwent a cleaning process to ensure its quality and consistency.
- Basic data wrangling and formatting techniques were applied to the cleaned data.



1. Get request for rocket launch data using API

```
In [6]:  spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
In [7]:  response = requests.get(spacex_url)
```

2. Use json_normalize method to convert json result to dataframe

```
In [12]: # Use json_normalize method to convert the json result into a dataframe

         # decode response content as json
         static_json_df = res.json()
```

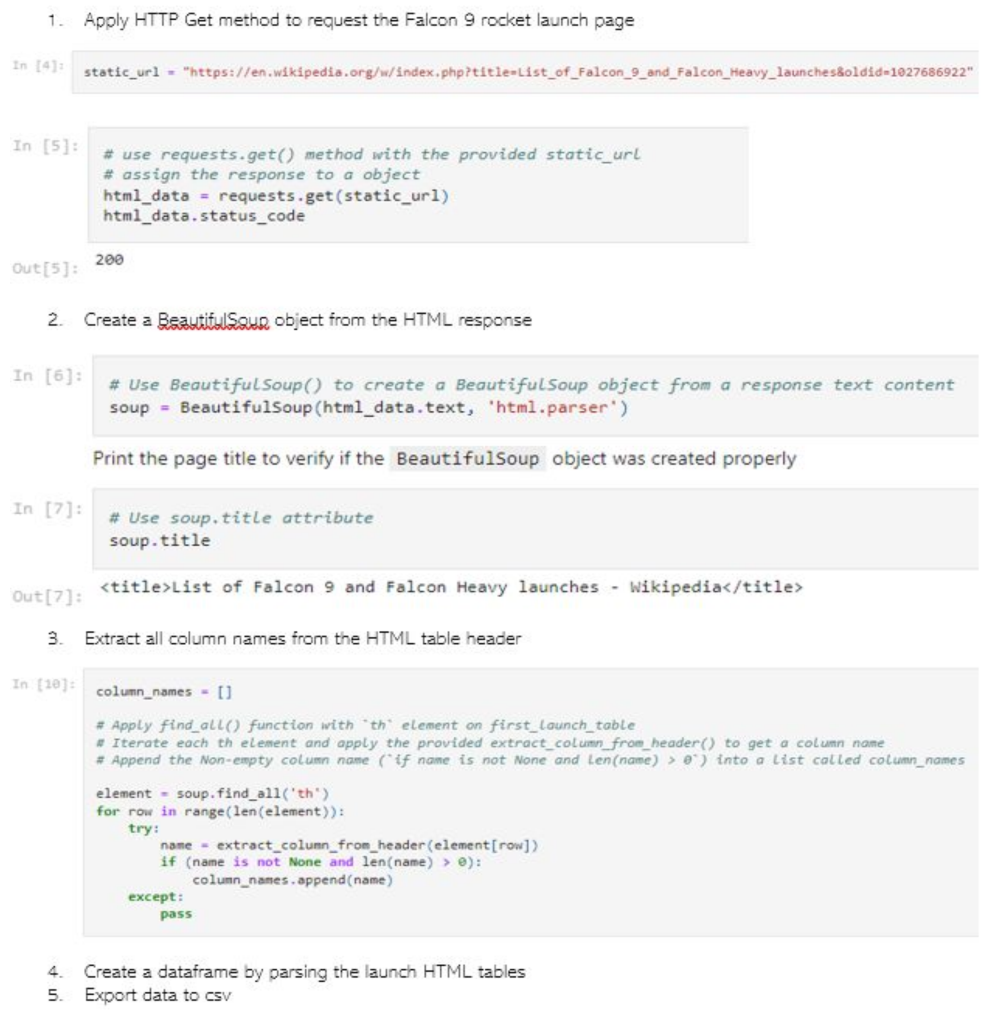
```
In [13]: # apply json_normalize
         data = pd.json_normalize(static_json_df)
```

3. We then performed data cleaning and filling in the missing values

```
In [30]: rows = data_falcon9['PayloadMass'].values.tolist()[0]

         df_rows = pd.DataFrame(rows)
         df_rows = df_rows.replace(np.nan, PayloadMass)

         data_falcon9['PayloadMass'][0] = df_rows.values
         data_falcon9
```

DATA COLLECTION CONTD.

- Web scraping techniques, specifically using BeautifulSoup, were applied to extract Falcon 9 launch records from a website.
- The scraped data, in the form of an HTML table, was parsed and transformed into a pandas dataframe.
- This conversion allowed for easier manipulation and analysis of the Falcon 9 launch records.



1. Apply HTTP Get method to request the Falcon 9 rocket launch page

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

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

Out[5]: 200
```

2. Create a BeautifulSoup object from the HTML response

```
In [6]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
        soup = BeautifulSoup(html_data.text, '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>
```

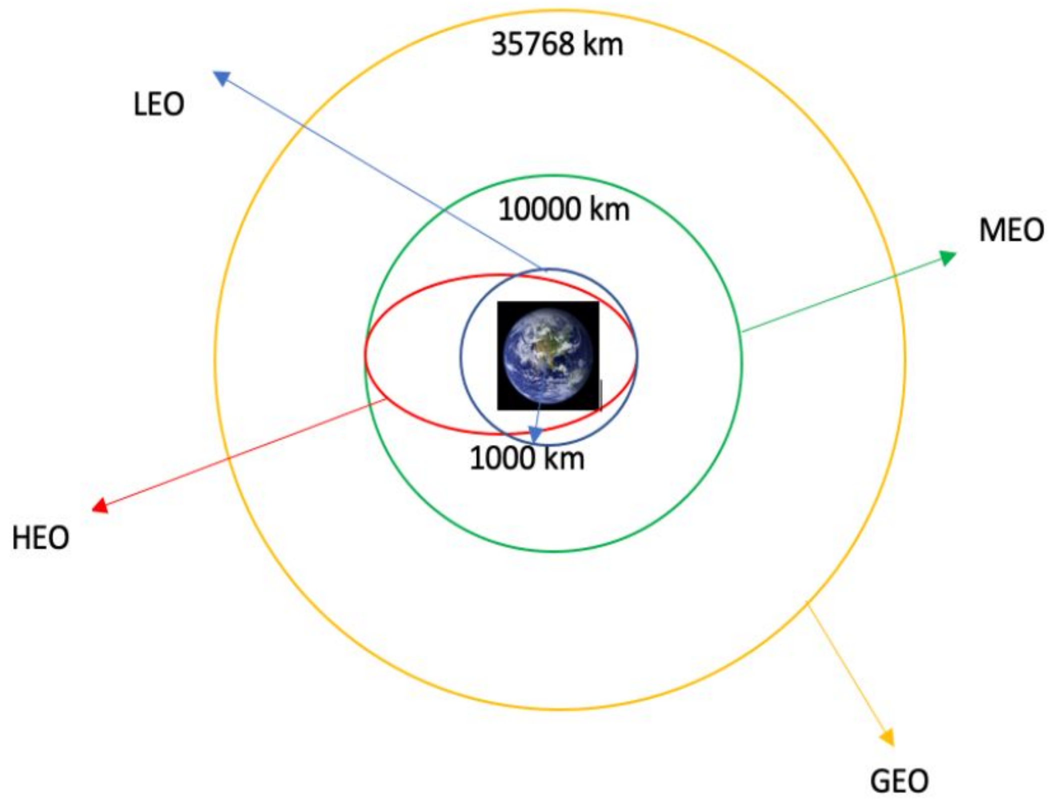
3. Extract all column names from the HTML table header

```
In [10]: column_names = []

         # Apply find_all() function with `th` element on first_launch_table
         # Iterate each th element and apply the provided extract_column_from_header() to get a column name
         # Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names

         element = soup.find_all('th')
         for row in range(len(element)):
             try:
                 name = extract_column_from_header(element[row])
                 if (name is not None and len(name) > 0):
                     column_names.append(name)
             except:
                 pass
```

4. Create a dataframe by parsing the launch HTML tables
5. Export data to csv

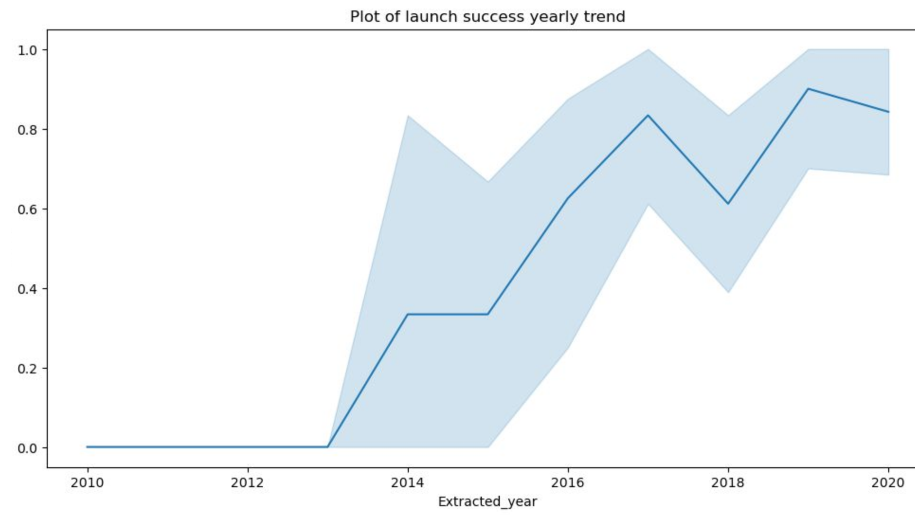
DATA WRANGLING

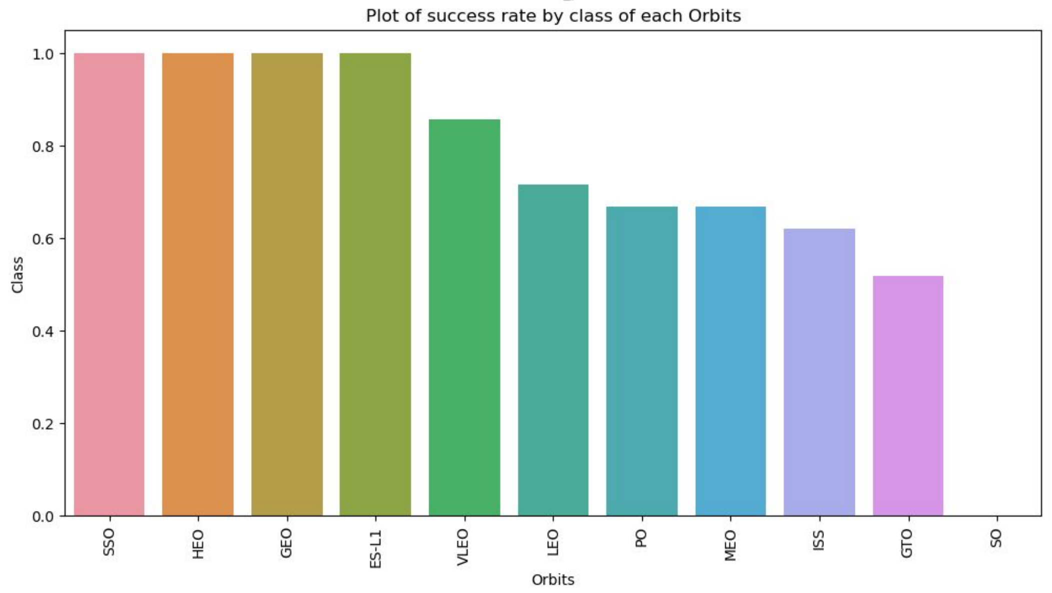
- Exploratory data analysis (EDA) was conducted on the collected data.
- The analysis included calculating the number of launches at each launch site and identifying the number and occurrence of each orbit.
- Based on the findings from the EDA, training labels were determined for the data.
- A landing outcome label was created by processing the information from the outcome column.
- The results of the analysis, including the calculated statistics and labels, were exported to a CSV file for further use and reference.



EDA WITH DATA VISUALIZATION

- The data was explored through various visualizations to understand different relationships and trends.
- Visualizations were created to examine the relationship between flight number and launch site, payload and launch site, as well as the success rate of each orbit type.
- Another visualization was created to analyze the relationship between flight number and orbit type.
- Additionally, a visualization was generated to understand the yearly trend in launch success.
- These visualizations helped uncover insights and patterns within the data, providing valuable information for analysis and decision-making.





EDA WITH SQL

- The SpaceX dataset was seamlessly loaded into a PostgreSQL database within the Jupyter Notebook environment.
- SQL queries were written to perform exploratory data analysis (EDA) and gain insights from the data.
    - Queries were executed to obtain information such as the names of unique launch sites in the space mission.
    - Another query was used to calculate the total payload mass carried by boosters launched by NASA (CRS).
    - The average payload mass carried by booster version F9 v1.1 was determined through a specific query.
    - The total number of successful and failed mission outcomes was computed using relevant queries.
    - To identify failed landing outcomes on drone ships, along with their corresponding booster versions and launch site names, a specific query was written.
    - These SQL queries allowed for in-depth analysis and extraction of insights from the SpaceX dataset without leaving the Jupyter Notebook environment.

BUILDING INTERACTIVE MAP

- Launch sites were marked on a folium map, and map objects such as markers, circles, and lines were added to indicate the success or failure of launches at each site.
- Launch outcomes (failure or success) were assigned class labels, with 0 representing failure and 1 representing success.
- Marker clusters with color labels were used to identify launch sites with relatively high success rates.
- Distances between a launch site and its nearby locations were calculated to gather insights.
- Some questions were addressed, such as whether launch sites are near railways, highways, and coastlines.
  - Additionally, the analysis examined whether launch sites maintain a certain distance from cities.
  - These analyses provided valuable information about the spatial characteristics and success rates of launch sites, contributing to decision-making and further understanding of the data.

Build a Dashboard with Plotly Dash

- An interactive dashboard was developed using Plotly Dash.
- The dashboard included various visualizations, such as pie charts, to display the total launches for specific launch sites.
- Scatter graphs were plotted to showcase the relationship between launch outcomes and payload mass (in kilograms) for different booster versions.
- The interactive nature of the dashboard allowed users to explore and interact with the visualizations, gaining deeper insights into the data.
- The pie charts provided a clear overview of the distribution of launches among different sites.
- The scatter graph facilitated the analysis of how the outcome of launches varied with the payload mass for each booster version.
- Overall, the Plotly Dash dashboard enhanced the presentation and exploration of the SpaceX dataset, enabling a more engaging and informative experience for users.

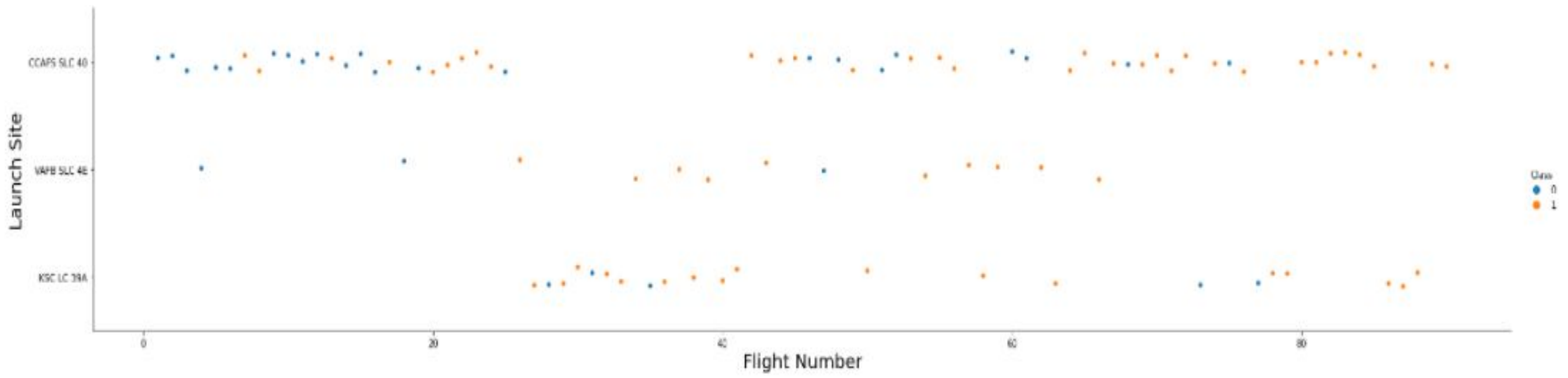
PREDICTIVE ANALYSIS (CLASSIFICATION)

- The data was loaded into the environment using numpy and pandas libraries.
- Data transformation techniques were applied to preprocess the data for machine learning tasks.
- The dataset was split into training and testing sets to evaluate model performance.
- Multiple machine learning models were built, incorporating different algorithms and techniques.
- Hyperparameter tuning was performed using GridSearchCV to optimize the models.
- The accuracy metric was utilized to assess the performance of the models.
- Feature engineering methods were employed to improve the model's predictive power.
- Algorithm tuning was conducted to fine-tune the models for better performance.
- Through these iterative processes, the best performing classification model was identified based on its accuracy and overall performance.

RESULTS

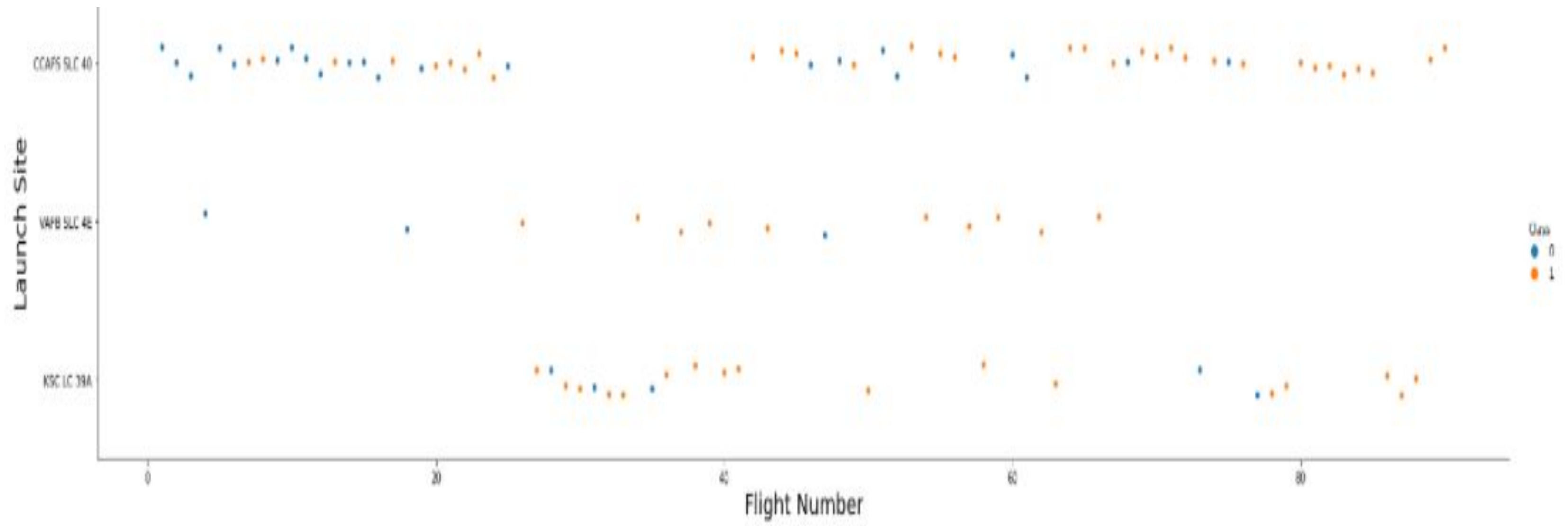
FLISGHT NUMBER vs. LAUNCH SITE

- Based on the plotted data, it was observed that there is a positive correlation between the flight amount at a launch site and the success rate at that site.



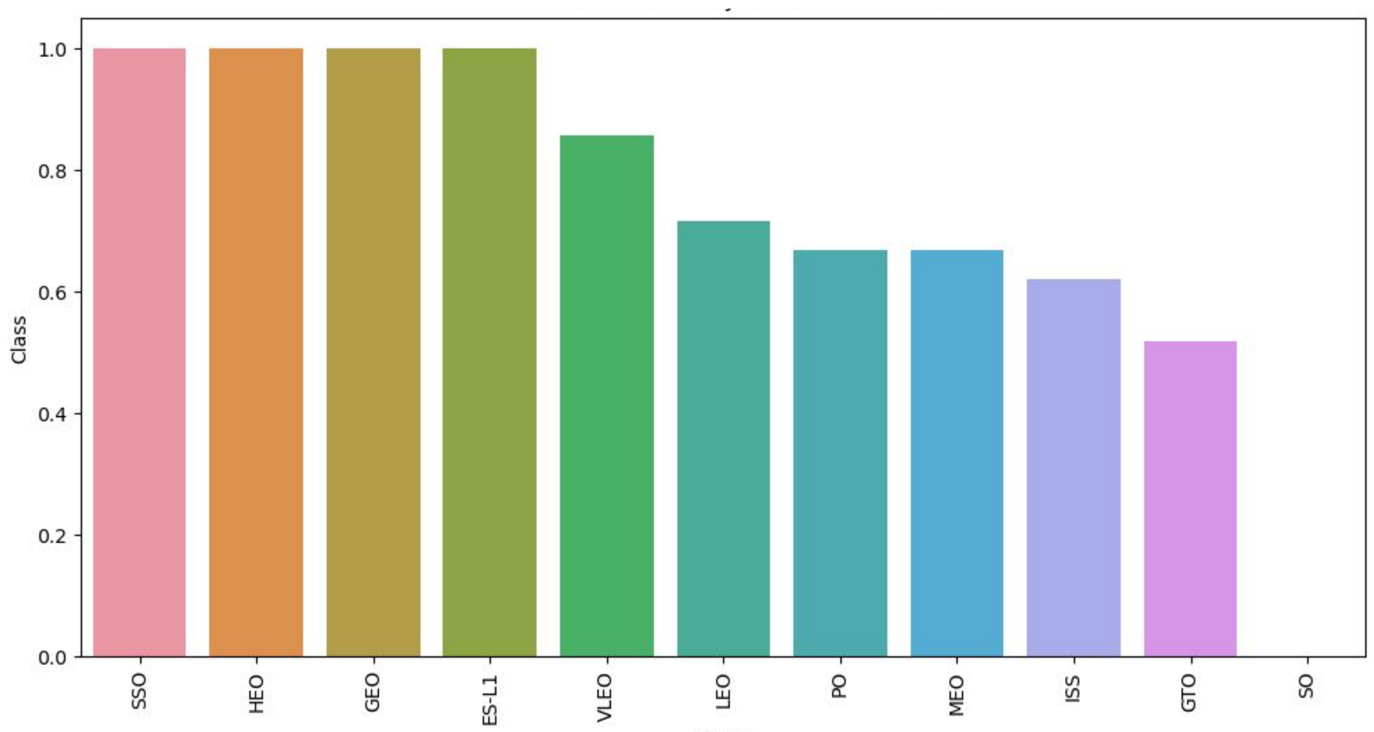
PAYLOAD vs. LAUNCH SITE

The greater the payload mass for launch site CCAFS SLC 40 the higher the success rate for the rocket.



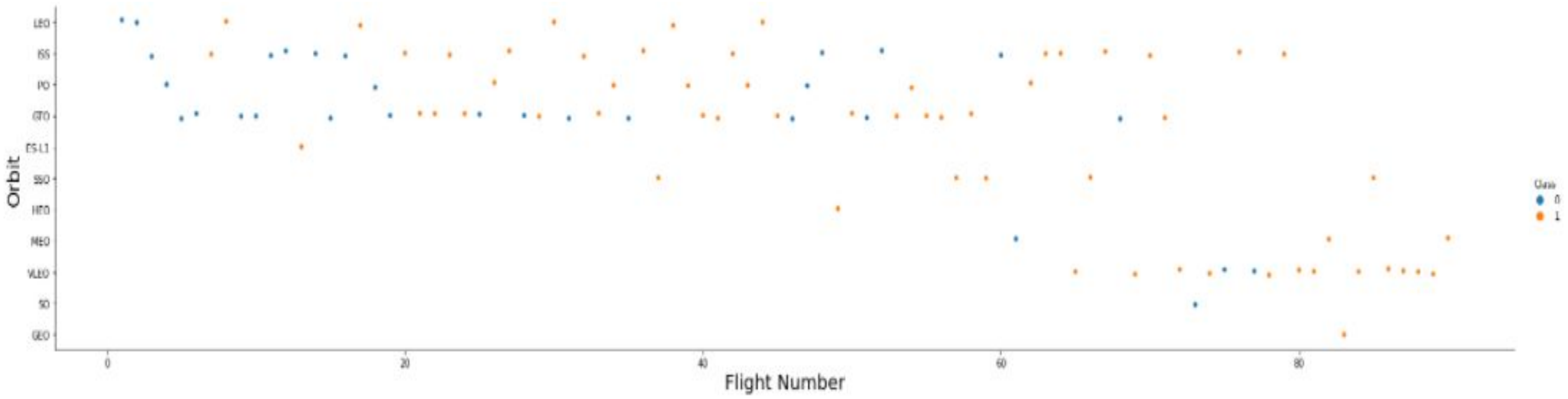
SUCCESS RATE vs. ORBIT TYPE

- Based on the plotted data, it is evident that the orbit types SSO (Sun-Synchronous Orbit), HEO (Highly Elliptical Orbit), GEO (Geostationary Orbit), and ES-L1 (Earth-Sun Lagrange 1 Point) have the highest success rates.
- Following closely behind are VLEO (Very Low Earth Orbit) and LEO (Low Earth Orbit). These orbit types exhibit relatively higher success rates compared to others.



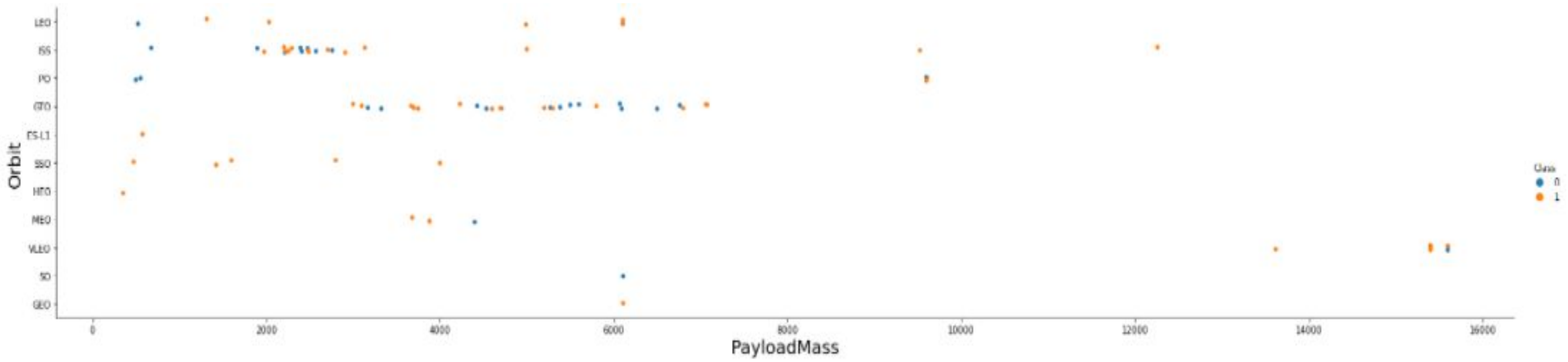
FLIGHT NUMBER vs. ORBIT TYPE

- The plotted data, depicting Flight Number vs. Orbit type, reveals interesting insights. It is observed that in the LEO (Low Earth Orbit), there exists a relationship between the success rate and the number of flights. As the flight number increases in the LEO orbit, the success rate tends to show a positive correlation. However, in the GTO (Geostationary Transfer Orbit) orbit, no discernible relationship between the flight number and the orbit's success rate is apparent.



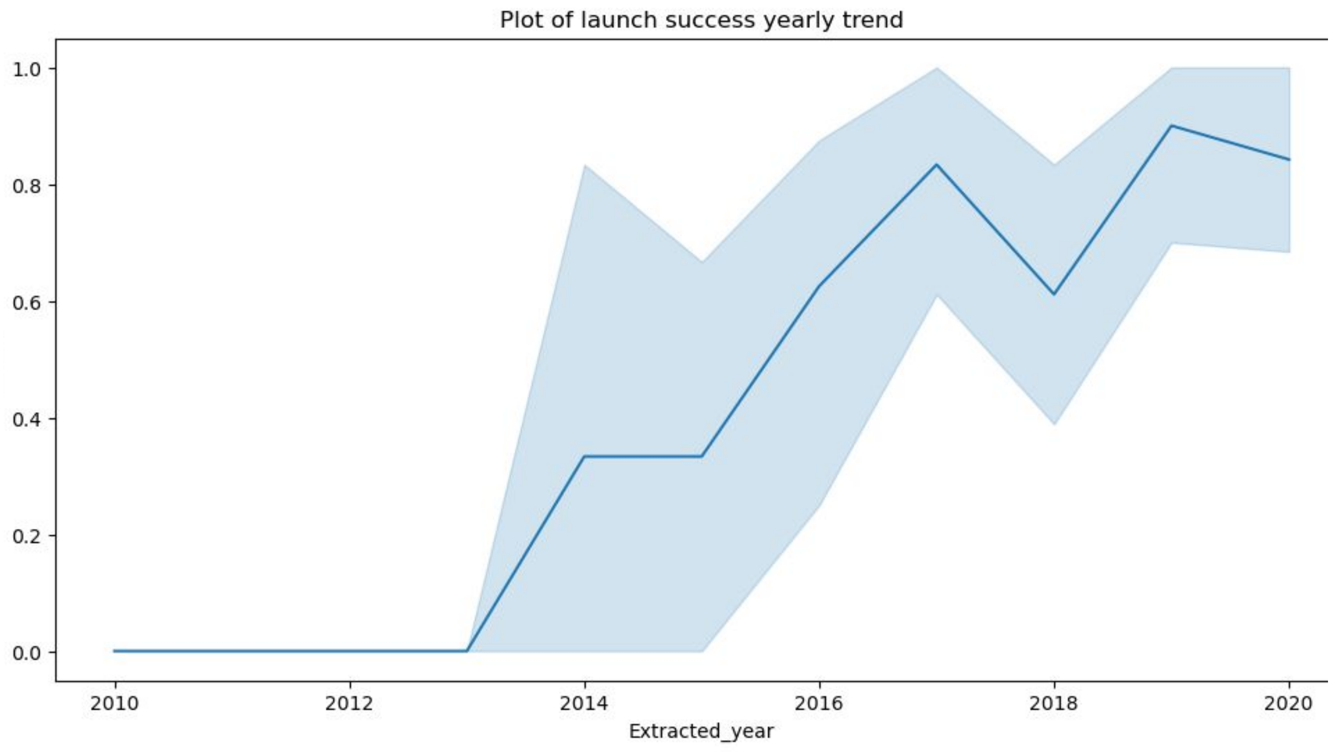
Payload vs. Orbit Type

- Based on the observation from the plot, it is evident that for heavy payloads, the success rate of landings is higher in the PO (Polar Orbit), LEO (Low Earth Orbit), and ISS (International Space Station) orbits. These particular orbits seem to exhibit a stronger correlation between heavy payloads and successful landings compared to other orbit types.



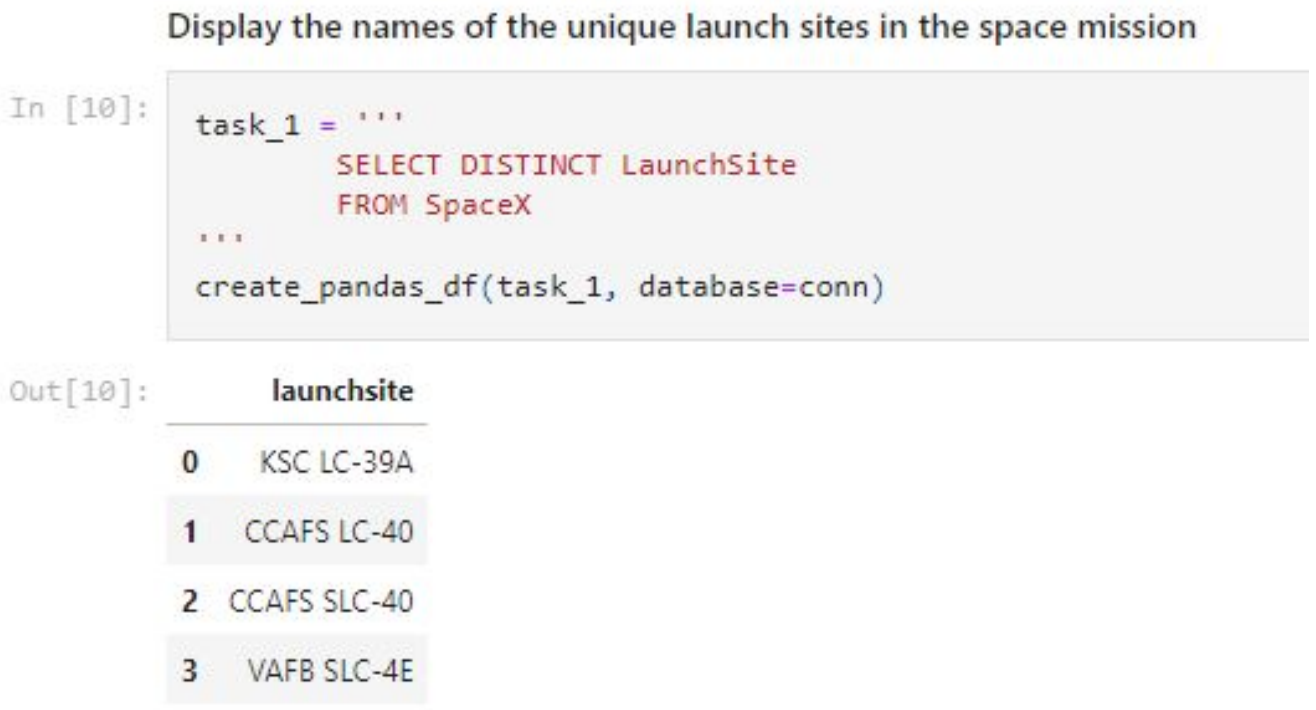
Launch Success Yearly Trend

- Based on the plotted data, it can be observed that the success rate has shown a consistent increase from 2013 until 2020. This indicates a positive trend where the success rate of launches has been progressively improving over the years during this period.



ALL LAUNCH SITE NAMES

- The DISTINCT keyword was utilized to filter and display only the unique launch sites from the SpaceX data. By applying DISTINCT in the query, duplicate launch site entries were eliminated, resulting in a list of distinct launch sites in the dataset.



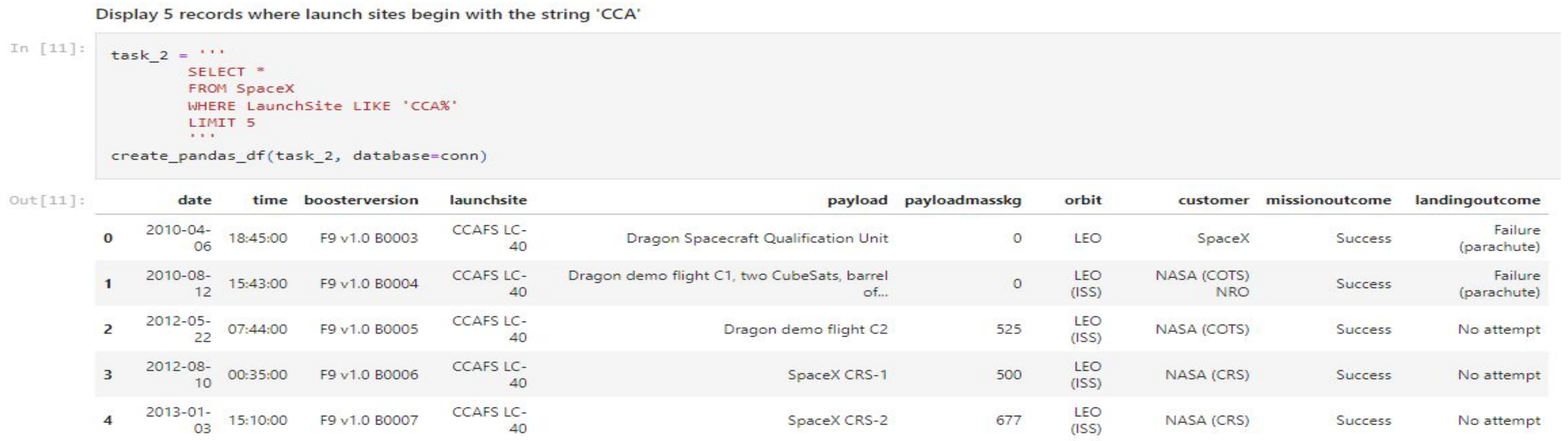
Display the names of the unique launch sites in the space mission

```
In [10]: task_1 = '''
                 SELECT DISTINCT LaunchSite
                 FROM SpaceX
         '''
         create_pandas_df(task_1, database=conn)
```

```
Out[10]:
     launchsite
0   KSC LC-39A
1   CCAFS LC-40
2  CCAFS SLC-40
3   VAFB SLC-4E
```

LAUNCH SITE NAMES BEGIN WITH 'CCA'

The aforementioned query was executed to retrieve five records from the dataset where the launch sites start with "CCA." The query specifically used the DISTINCT keyword to ensure only unique launch sites were displayed, and the WHERE clause with the LIKE operator was employed to filter launch sites beginning with "CCA." The resulting output consisted of five distinct launch sites meeting this criterion.



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

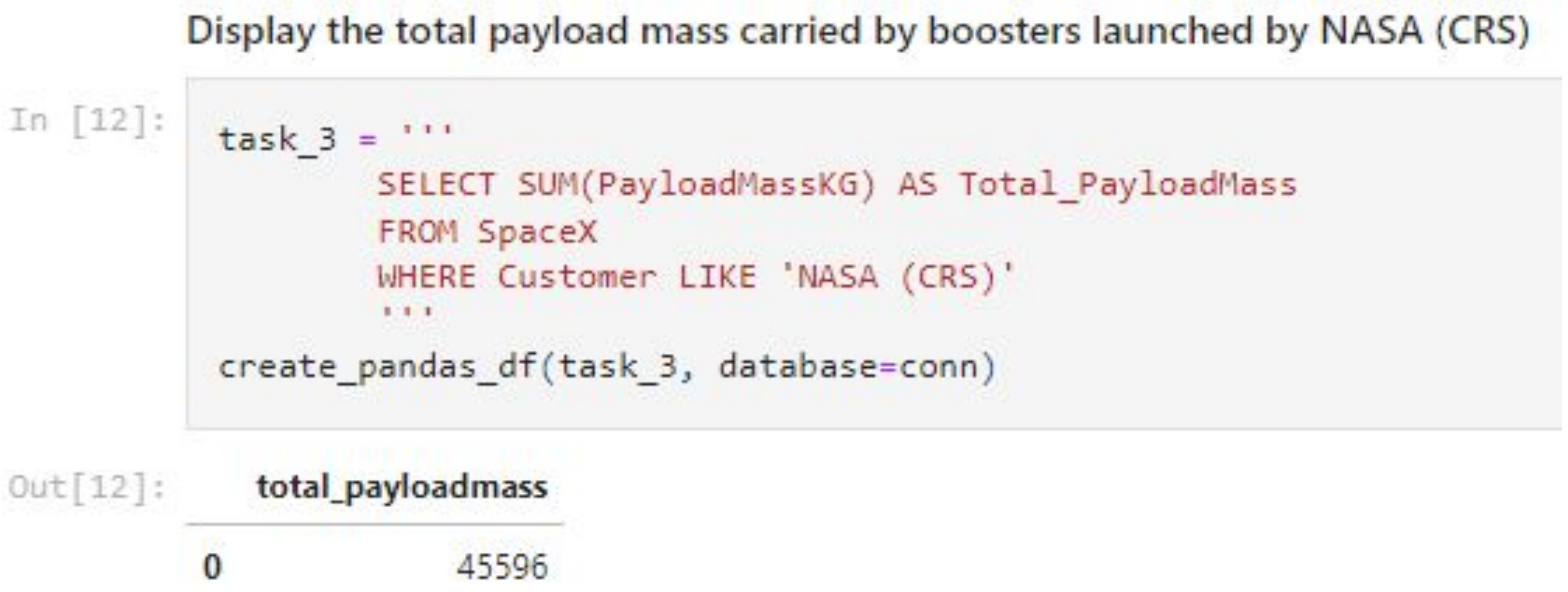
```
In [11]:
task_2 = '''
        SELECT *
        FROM SpaceX
        WHERE LaunchSite LIKE 'CCA%'
        LIMIT 5
        '''
create_pandas_df(task_2, database=conn)
```

Out[11]:

	date	time	boosterversion	launchsite	payload	payloadmasskg	orbit	customer	missionoutcome	landingoutcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
3	2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

TOTAL PAYLOAD MASS

The sum of the payload mass (in kilograms) was calculated, using the SUM function for the records where the customer is 'NASA (CRS)' in the "spacex_data" table. The resulting value of 45596 represents the total payload carried by boosters from NASA.



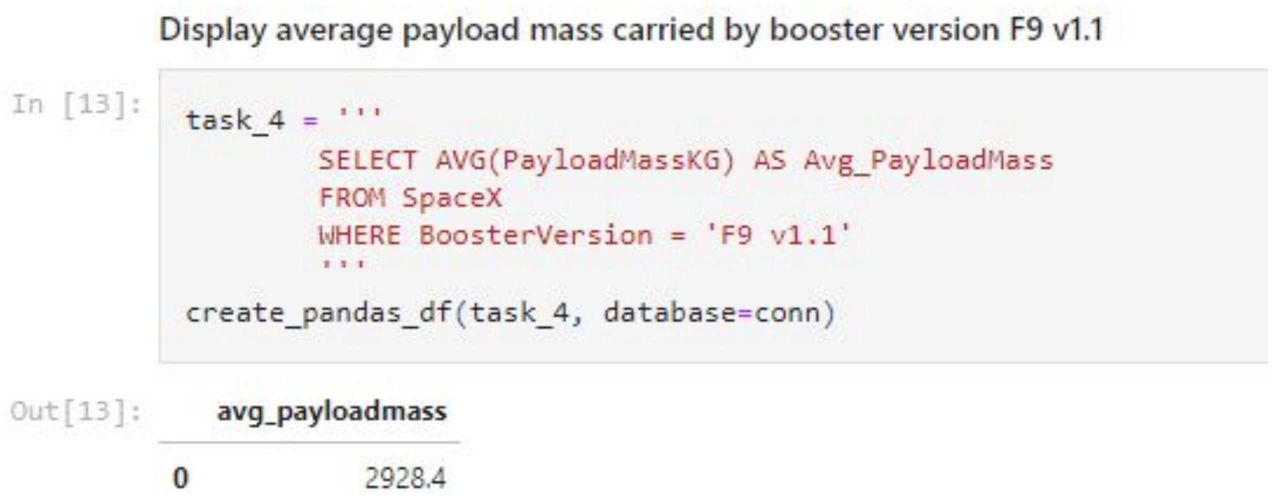
Display the total payload mass carried by boosters launched by NASA (CRS)

```
In [12]:
task_3 = '''
        SELECT SUM(PayloadMassKG) AS Total_PayloadMass
        FROM SpaceX
        WHERE Customer LIKE 'NASA (CRS)'
        '''
create_pandas_df(task_3, database=conn)
```

```
Out[12]:
   total_payloadmass
0              45596
```

AVERAGE PAYLOAD MASS BY F9 v1.1

- Using the provided query, we computed the average payload mass carried by booster version F9 v1.1 as 2928.4. The query utilizes the AVG function to calculate the average payload mass (in kilograms) for records where the booster version is 'F9 v1.1' in the "spacex_data" table. The resulting average payload mass is determined as 2928.4.

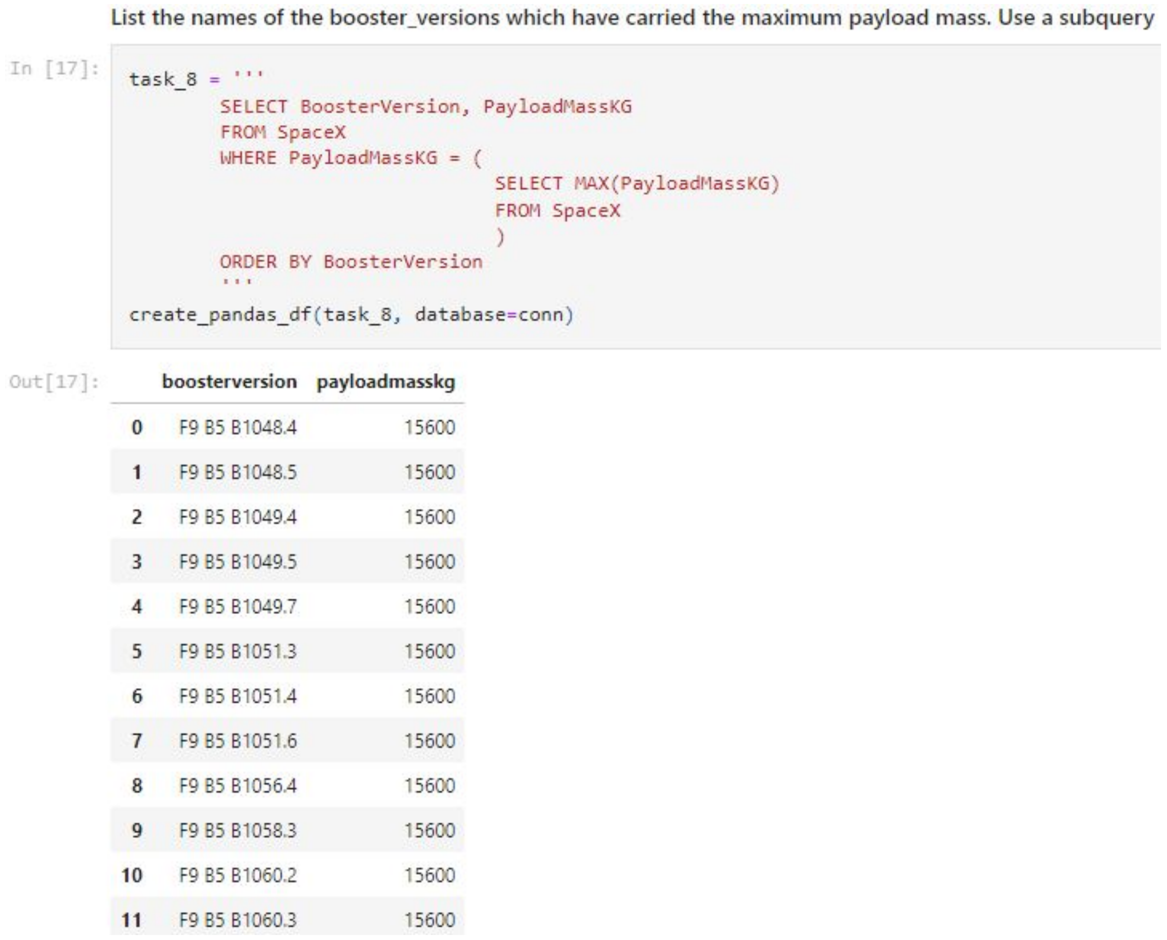


Display average payload mass carried by booster version F9 v1.1

```
In [13]: task_4 = '''
                 SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
                 FROM SpaceX
                 WHERE BoosterVersion = 'F9 v1.1'
                 '''
         create_pandas_df(task_4, database=conn)

Out[13]:    avg_payloadmass
         0           2928.4
```

BOOSTERS CARRIED MAXIMUM PAYLOAD

- IA query was used to retrieve the maximum payload mass from the "spacex_data" table. The outer query then selects all the records where the payload mass matches the maximum value. This query provides the details of the booster that carried the maximum payload.


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

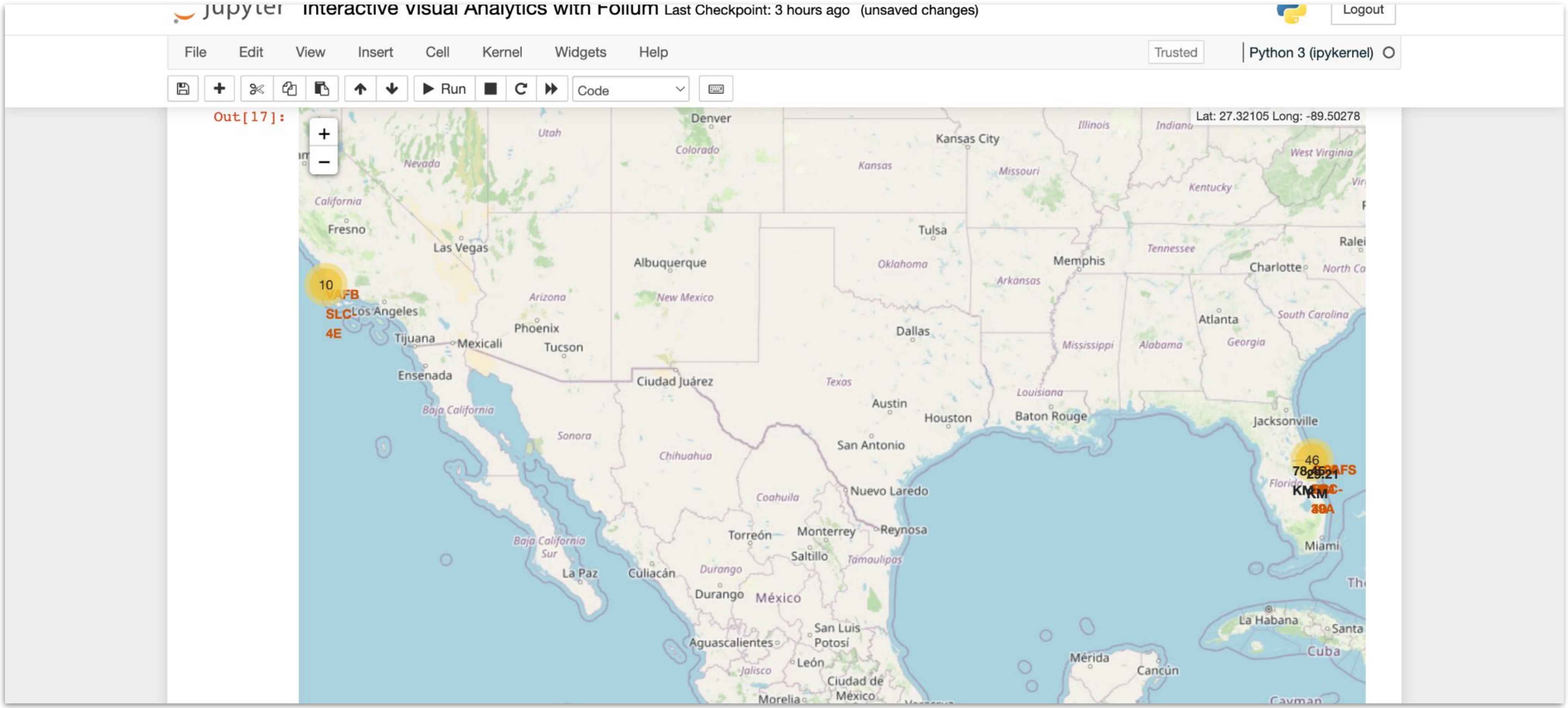
```
In [17]: task_8 = '''
                 SELECT BoosterVersion, PayloadMassKG
                 FROM SpaceX
                 WHERE PayloadMassKG = (
                                         SELECT MAX(PayloadMassKG)
                                         FROM SpaceX
                                         )
                 ORDER BY BoosterVersion
                 '''
         create_pandas_df(task_8, database=conn)
```

Out[17]:

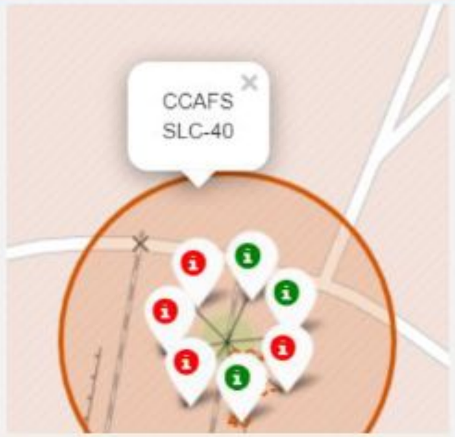
	boosterversion	payloadmasskg
0	F9 B5 B1048.4	15600
1	F9 B5 B1048.5	15600
2	F9 B5 B1049.4	15600
3	F9 B5 B1049.5	15600
4	F9 B5 B1049.7	15600
5	F9 B5 B1051.3	15600
6	F9 B5 B1051.4	15600
7	F9 B5 B1051.6	15600
8	F9 B5 B1056.4	15600
9	F9 B5 B1058.3	15600
10	F9 B5 B1060.2	15600
11	F9 B5 B1060.3	15600

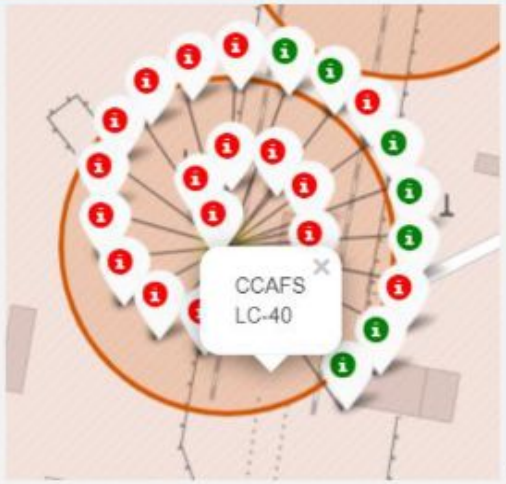
Launch Proximites Analysis

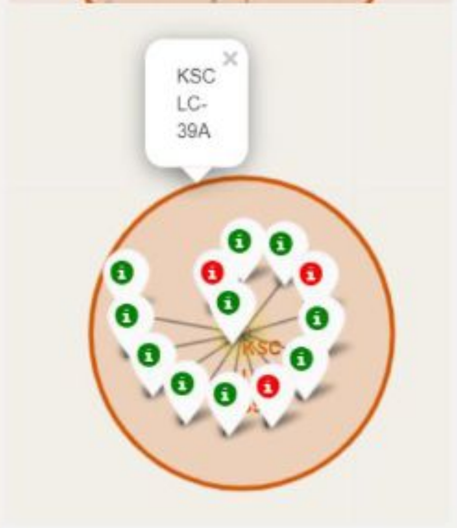
GLOBAL LAUNCH MARKERS

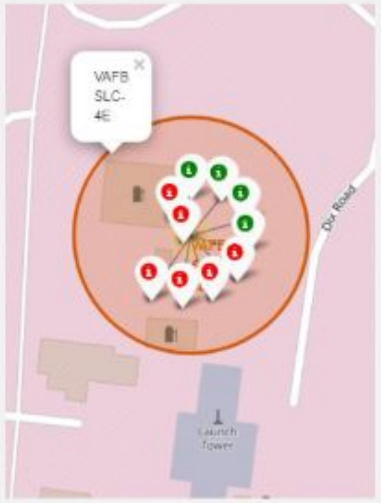


LAUNCH SITES









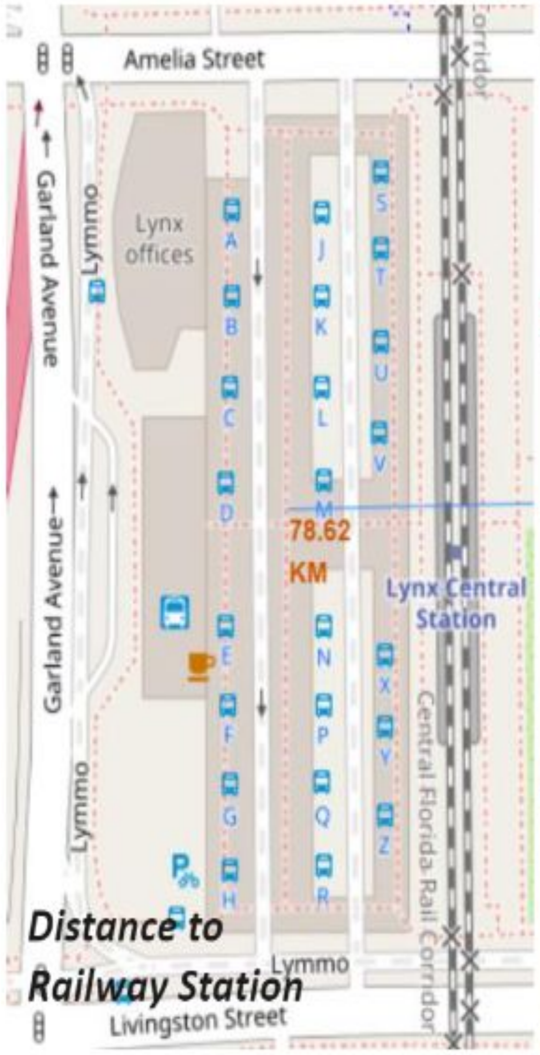
Florida Launch Sites

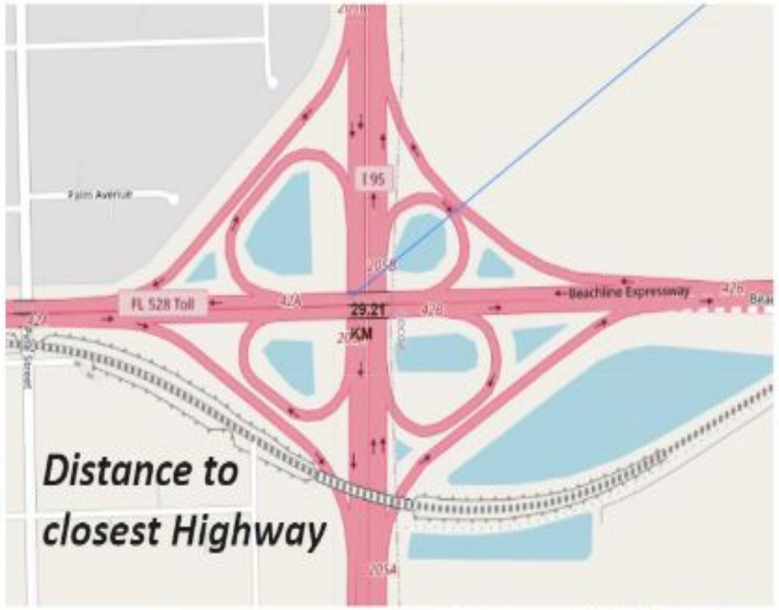
Green Marker shows successful Launches and Red Marker shows Failures

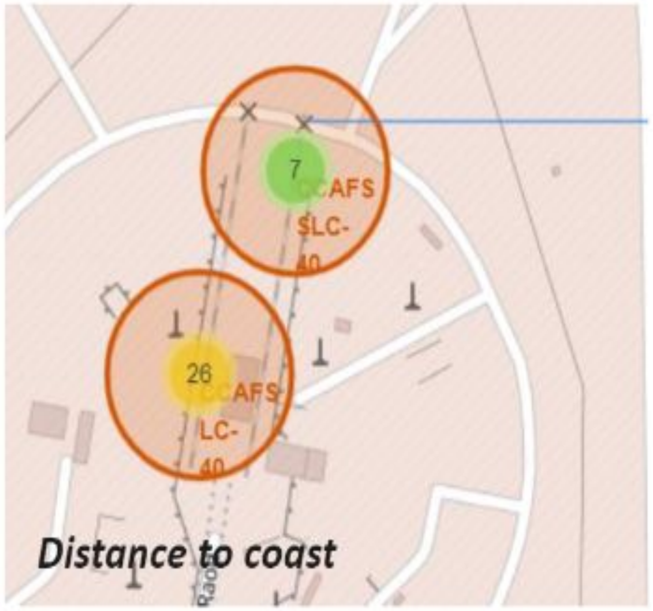
California Launch Site

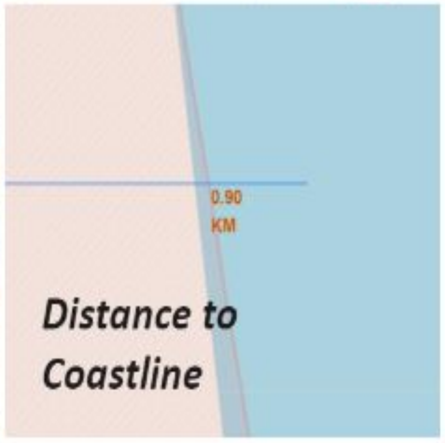
37

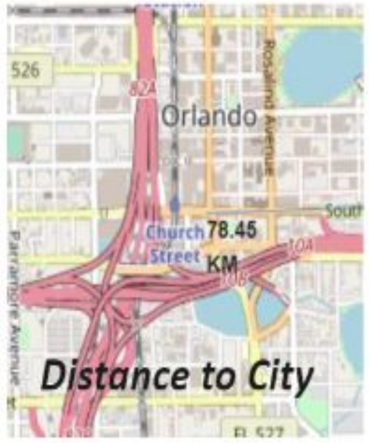
DISTANCE TO LANDMARKS









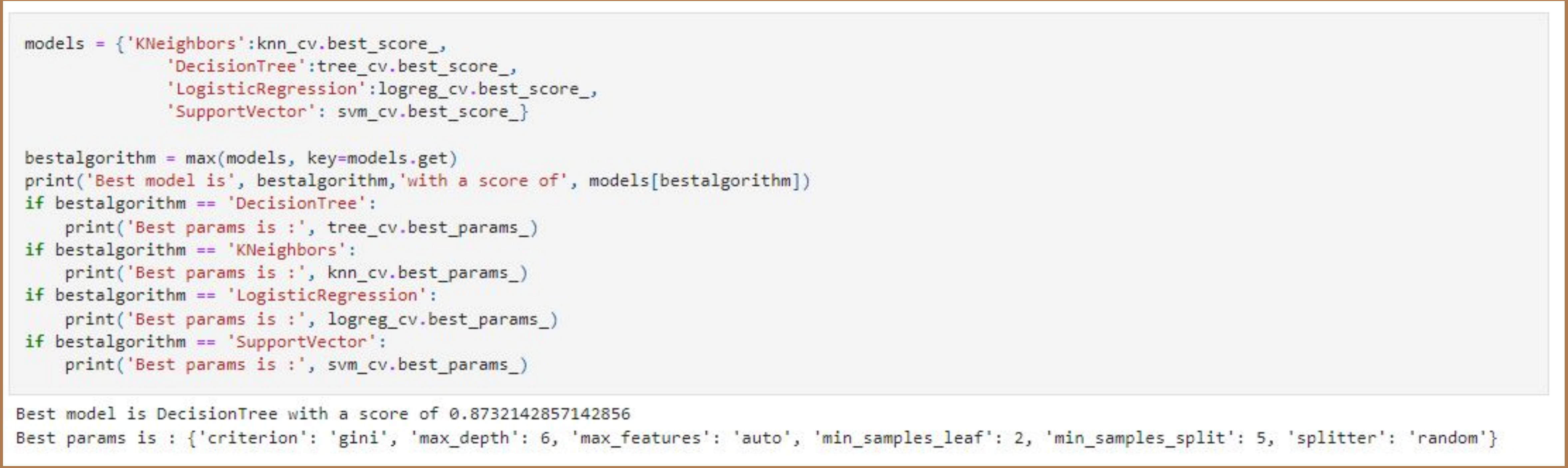


- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

PREDICTIVE ANALYSIS

CLASSIFICATION ACCURACY

- Among the models evaluated, the decision tree classifier achieved the highest classification accuracy.

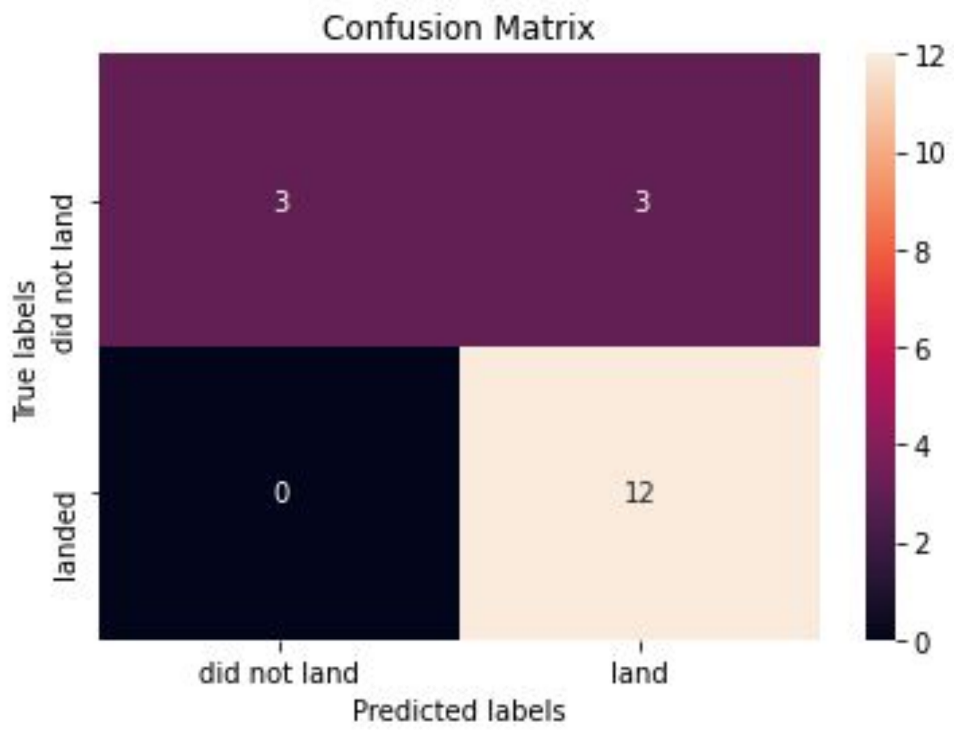


```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

CONFUSION MATRIX

- The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives .i.e., unsuccessful landing marked as successful landing by the classifier.



CONCLUSIONS

- Based on the analysis conducted, the following conclusions can be drawn:
  - There is a positive correlation between the flight amount at a launch site and the success rate at that site, indicating that a higher number of flights is associated with a greater success rate.
  - The launch success rate exhibited a consistent increase from 2013 to 2020, indicating an overall improvement in successful launches during this period.
  - The orbits ES-L1, GEO, HEO, SSO, and VLEO demonstrated the highest success rates compared to other orbit types.
  - KSC LC-39A emerged as the launch site with the most successful launches among all the sites considered.
  - Based on the evaluation of machine learning algorithms, the decision tree classifier emerged as the best-performing algorithm for the given task, achieving the highest classification accuracy.
- These conclusions provide valuable insights into the relationships between flight amount, success rate, launch sites, and machine learning algorithm performance, enabling informed decision-making and further analysis in the context of rocket launches.

THANK YOU