IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Josefina Figueroa
July 01, 2024

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection using SpaceX API, Data Collection with Web Scraping, Data Wrangling, Exploratory Data Analysis using SQL, EDA DataViz Using Python Pandas and Matplotlib, Launch Sites Analysis with Folium-Interactive Visual Analytics and Ploty Dash , and Machine Learning Landing Prediction.
- Summary of all results
  - Exploratory Data Analysis results, Interactive Visual Analytics and Dashboards, and Predictive Analysis.

Introduction

- Project background and context
  - Analyze SpaceX launch data to identify factors influencing landing success and build a predictive model to assist competitors in making informed bids against SpaceX.
  - SpaceX's $62 million Falcon 9 launches are cheaper than competitors' $165 million launches due to reusable first stages. This project analyzes SpaceX launch data to predict Falcon 9 first stage landing success, aiding competitors in bidding against SpaceX.
- Problems you want to find answers
  - Predict the success of SpaceX Falcon 9 first stage landings to determine launch costs.

Section 1

Methodology

Methodology

Executive Summary

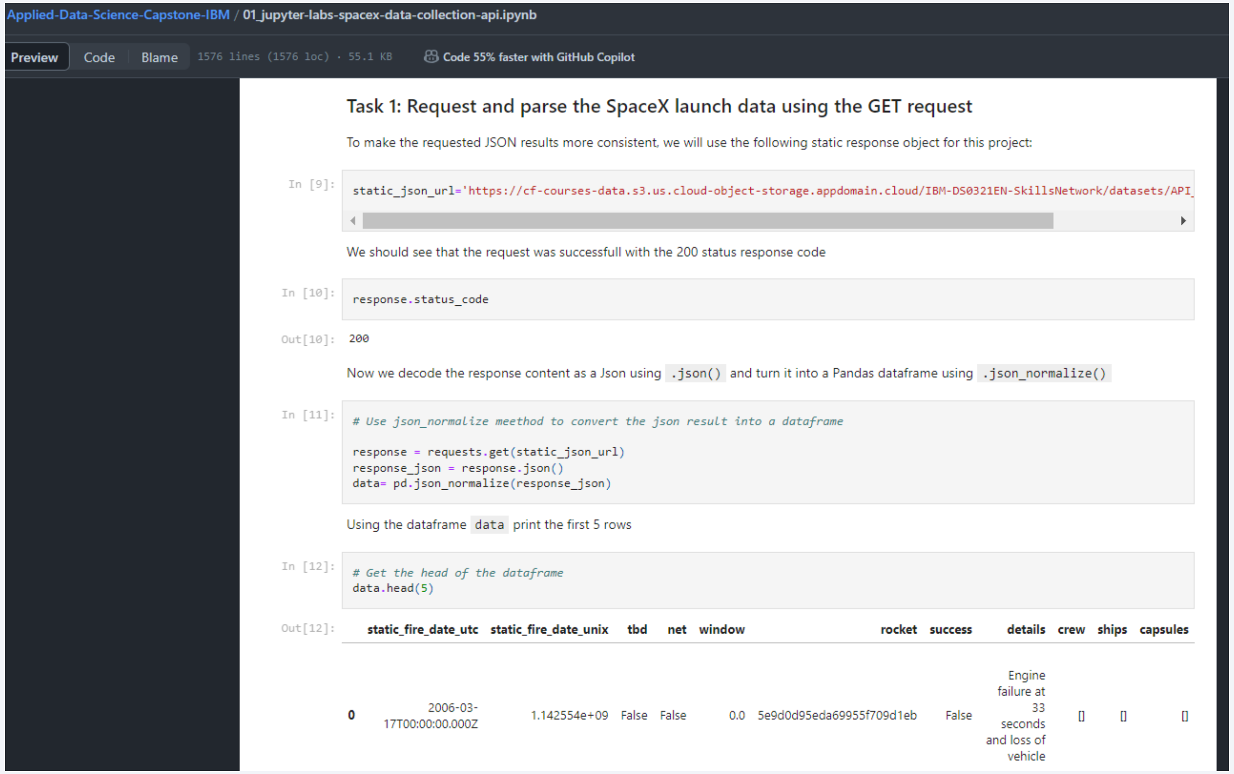
- Data collection methodology:
  - Describes how data was collected
- Perform data wrangling
  - Describes how data was processed
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- Data for this project was collected using the SpaceX API and additional sources like Wikipedia. The API provided detailed information about each Falcon 9 launch, including rocket details, payload, launch site, and core data. Steps to collect and prepare the data:
  - API Requests: Data was requested from the SpaceX API, retrieving information on past launches.
  - Data Extraction: Specific details were extracted using the API, such as booster version, payload mass, orbit, launch site details (longitude and latitude), and core information (landing outcome, gridfins usage, reuse count, etc.).
  - Data Wrangling: The extracted data was cleaned and transformed into a structured format suitable for analysis. This included handling missing values and filtering out irrelevant data.
  - Data Integration: The cleaned data was combined into a single DataFrame, ensuring all relevant information was available for analysis.
- The final dataset included comprehensive details on Falcon 9 launches, enabling thorough analysis and predictive modeling.

Data Collection – SpaceX API

- Data was collected using the SpaceX RESTful API by making GET requests. The launch data was retrieved and parsed, then the JSON response content was decoded and converted into a Pandas DataFrame for further analysis.
- GitHub URL of the completed SpaceX API calls notebook: https://github.com/a-hognose-snake/Applied-Data-Science-Capstone-IBM/blob/main/01_jupyter-labs-spacex-data-collection-api.ipynb



Data Collection - WebScraping

- The process scraped data from the Wikipedia page titled "List of Falcon 9 and Falcon Heavy launches" using BeautifulSoup. The relevant launch records were stored in an HTML table, which was parsed to extract data on launch dates, times, sites, payloads, orbits, customers, and outcomes. The parsed data was organized into a dictionary and then converted into a Pandas DataFrame for further analysis.
- GitHub URL of the completed web scraping notebook: https://github.com/a-hognose-snake/Applied-Data-Science-Capstone-IBM/blob/main/02_jupyter-labs-webscraping.ipynb

Data Wrangling

- The process included data wrangling, exploratory data analysis, and creating training labels for our machine learning model.
  - Data Wrangling:
    - Import Libraries: Pandas and NumPy.
    - Load Data: SpaceX dataset into a DataFrame.
    - Identify Missing Values: Calculated missing value percentages.
    - Determine Data Types: Identified numerical and categorical columns.
  - Exploratory Data Analysis (EDA):
    - Launch Sites: Counted launches at each site.
    - Orbits: Counted occurrences of each orbit type.
    - Mission Outcomes: Analyzed mission outcomes.
  - Creating Training Labels:
    - Define Landing Outcomes: Categorized as successful or unsuccessful.
    - Label Creation: Binary label (Class) with 1 for success and 0 for failure.
    - Success Rate: Averaged the Class column.
- These steps prepare the data for training models to predict Falcon 9 landing success.
- GitHub URL of your completed data wrangling related notebook: https://github.com/a-hognose-snake/Applied-Data-Science-Capstone-IBM/blob/main/03_labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- Performed data analysis and visualization using Pandas and Matplotlib.
  - Exploratory Data Analysis
  - Preparing Data for Plotting
- Used scatter plots to Visualize the relationship between Flight Number and Launch Site, Payload and Launch Site, Flight Number and Orbit type, Payload and Orbit type.
- Used Bar chart to Visualize the relationship between success rate of each orbit type
- Line plot to Visualize the launch success yearly trend.
- GitHub URL of your completed EDA with data visualization notebook: https://github.com/a-hognose-snake/Applied-Data-Science-Capstone-IBM/blob/main/05_edadataviz.ipynb

EDA with SQL

- GitHub URL of your completed EDA with SQL notebook: https://github.com/a-hognose-snake/Applied-Data-Science-Capstone-IBM/blob/main/04_jupyter-labs-eda-sql-coursera_sqllite.ipynb

Task 4

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

In [12]:

```
%sql select avg(payload_mass__kg_) as average from SPACEXTBL where booster_version like 'F9 v
```

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

Out[12]:

average
2534.6666666666665

Task 5

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

Hint:Use min function

In [13]:

```
%sql select min(date) as date from SPACEXTBL where mission_outcome like 'Success'
```

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

Out[13]:

date
2010-06-04

Task 6

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

In [14]:

```
%sql select booster_version from SPACEXTBL where (mission_outcome like 'Success') and (payloa
```

Build an Interactive Map with Folium

- Various map objects such as markers, circles, and lines were created and added to a Folium map to enhance the visualization of SpaceX launch sites and their success rates. Markers were used to pinpoint the exact locations of each launch site, providing a clear geographical context. Circles were added around these markers to represent the payload range, with the size of the circles indicating the payload capacity at each site. Lines were used to connect the launch sites to their respective landing zones, illustrating the flight paths and distances traveled by the rockets.
  - These objects were added to provide an intuitive and interactive way for users to explore the data. The markers help users quickly identify and locate the launch sites, while the circles offer a visual representation of the payload capacities, making it easier to understand the distribution and capabilities of each site. The lines connecting launch sites to landing zones further enrich the map by showing the relationship between launch and landing locations, highlighting the complexity and reach of SpaceX missions
- GitHub URL of your completed interactive map with Folium map: https://github.com/a-hognose-snake/Applied-Data-Science-Capstone-IBM/blob/main/06_lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- Launch Site Drop-down: Allows selection of specific launch sites or all sites to filter data.
- Success Pie Chart: Displays total successful launches, updating based on the selected site.
- Payload Range Slider: Enables selection of payload mass range for detailed analysis.
- Success-Payload Scatter Chart: Shows correlation between payload mass and launch success, updating with selected site and payload range.
- These interactive components provide a user-friendly interface for exploring SpaceX launch data. The drop-down and slider allow flexible filtering, the pie chart offers a quick success rate overview, and the scatter plot details the impact of payload on launch outcomes. Together, they facilitate comprehensive and insightful analysis.
- GitHub URL of your completed Plotly Dash lab: https://github.com/a-hognose-snake/Applied-Data-Science-Capstone-IBM/blob/main/07_build_a_dashboard_application_with_plotly_dash

Predictive Analysis (Classification)

- Building the Model
  - Data Preparation: Collect and clean the dataset, engineer features, and split into training, validation, and test sets.
  - Model Selection: Choose initial algorithms (e.g., Logistic Regression, Decision Trees, Random Forest) and train a baseline model.
- Evaluating the Model
  - Train Models: Train on the training set using cross-validation for robust evaluation.
  - Performance Metrics: Assess models using accuracy, precision, recall, F1-Score, ROC-AUC, and confusion matrix.
- Improving the Model
  - Hyperparameter Tuning: Use Grid Search, Random Search, or Bayesian Optimization.
  - Feature Engineering: Enhance features, reduce dimensionality, and address imbalanced data with techniques like SMOTE.
  - Ensemble Methods: Improve performance with Bagging (e.g., Random Forest), Boosting (e.g., XGBoost), and Stacking.
- Selecting the Best Model
  - Final Evaluation: Retrain the best models on combined training and validation sets and evaluate on the test set.
  - Compare Metrics: Choose the model with the best overall performance.
  - Deployment: Save and document the model, set up monitoring for ongoing performance.

- GitHub URL of your completed predictive analysis lab: https://github.com/a-hognose-snake/Applied-Data-Science-Capstone-IBM/blob/main/08_SpaceX_Machine%20Learning%20Prediction.ipynb

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

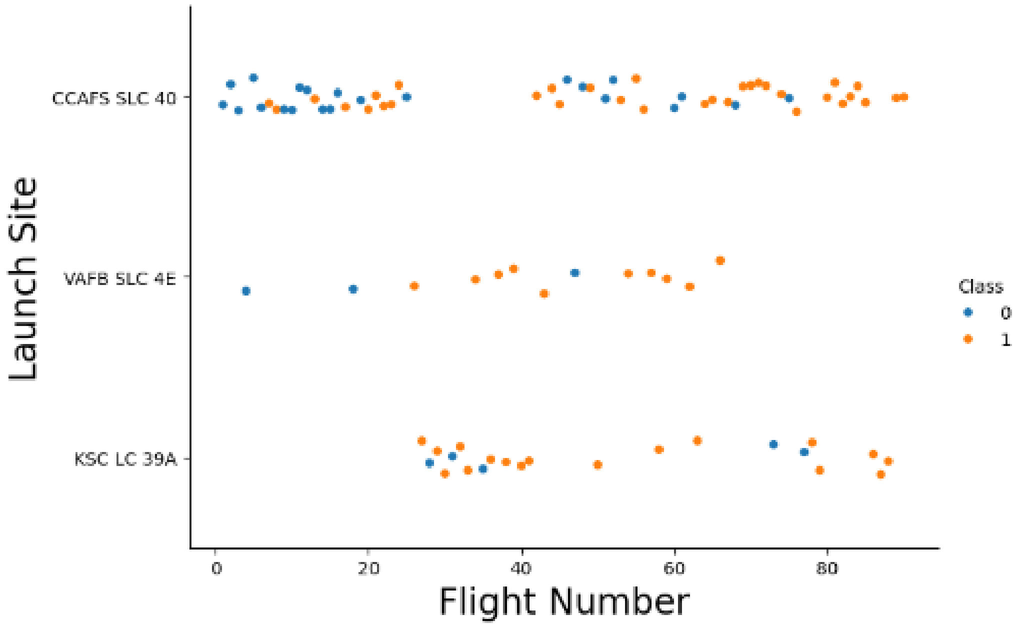
- Explanation:
  - We can deduce that, as the flight number increases, so does the success rate. In the case of the launch site VAFB SCL 4E, the success rate is a 100% after the 50th flight. Also, the other 2 launch sites show a 100% success rate after the 80th flight.

TASK 1: Visualize the relationship between Flight Number and Launch Site

Use the function `catplot` to plot `FlightNumber` vs `LaunchSite`, set the parameter `x` parameter to `FlightNumber`,set the `y` to `Launch Site` and set the parameter `hue` to `'class'`

[5]:

```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 1.5 ,height=5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



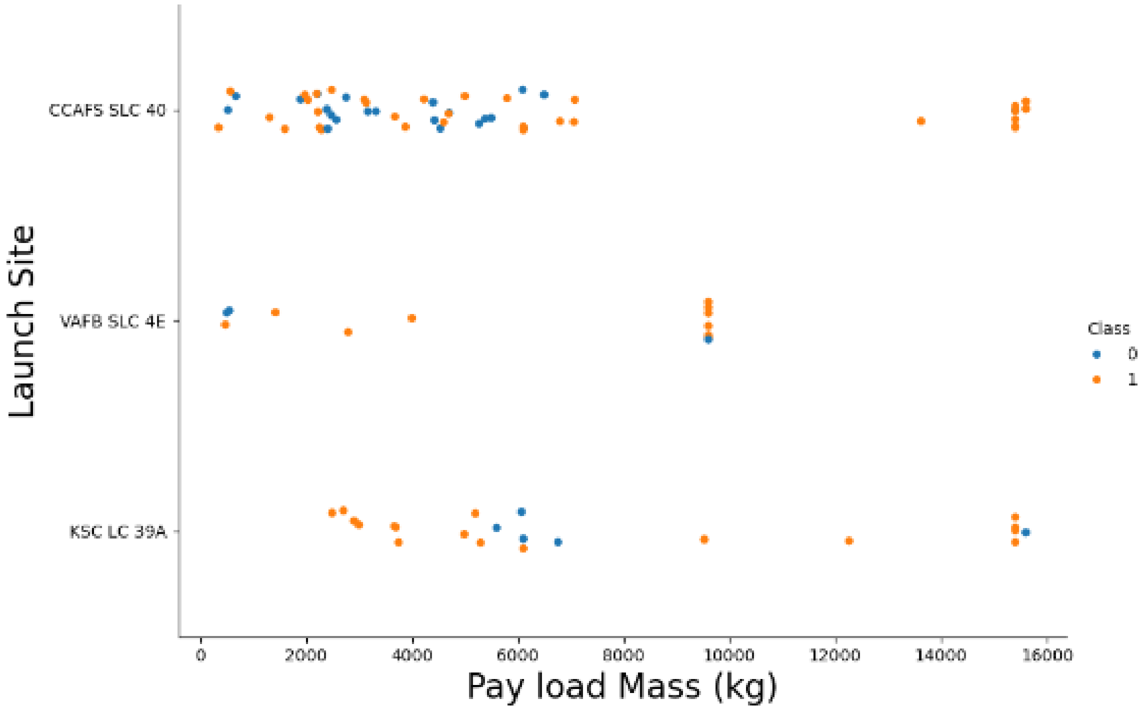
Payload vs. Launch Site

TASK 2: Visualize the relationship between Payload and Launch Site

We also want to observe if there is any relationship between launch sites and their payload mass.

In [6]:

```
# 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
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 1.5 ,height=6)
plt.xlabel("Pay load Mass (kg)",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



Now if you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

Success Rate vs. Orbit Type

- Explanation:
  - Orbits to the left have the highest success rates at 100%, while the one at the right has the least success at 0%.

TASK 3: Visualize the relationship between success rate of each orbit type

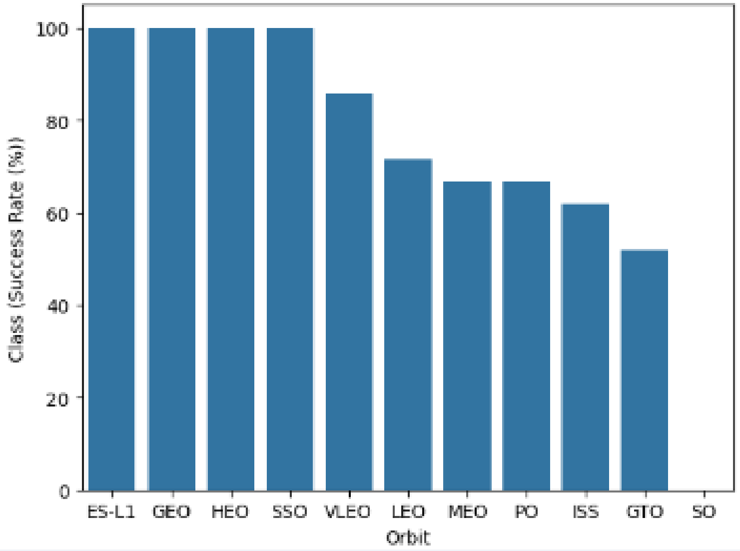
Next, we want to visually check if there are any relationship between success rate and orbit type.

Let's create a `bar chart` for the sucess rate of each orbit

In [7]:

```
# HINT use groupby method on Orbit column and get the mean of Class column
sr_df = df.groupby('Orbit')['Class'].mean().reset_index().sort_values(by='Class', ascending=False)
sr_df['Class'] = sr_df['Class'] * 100


sns.barplot(data=sr_df, x='Orbit', y='Class')
plt.xlabel('Orbit')
plt.ylabel('Class (Success Rate (%))')
plt.show()
```



Flight Number vs. Orbit Type

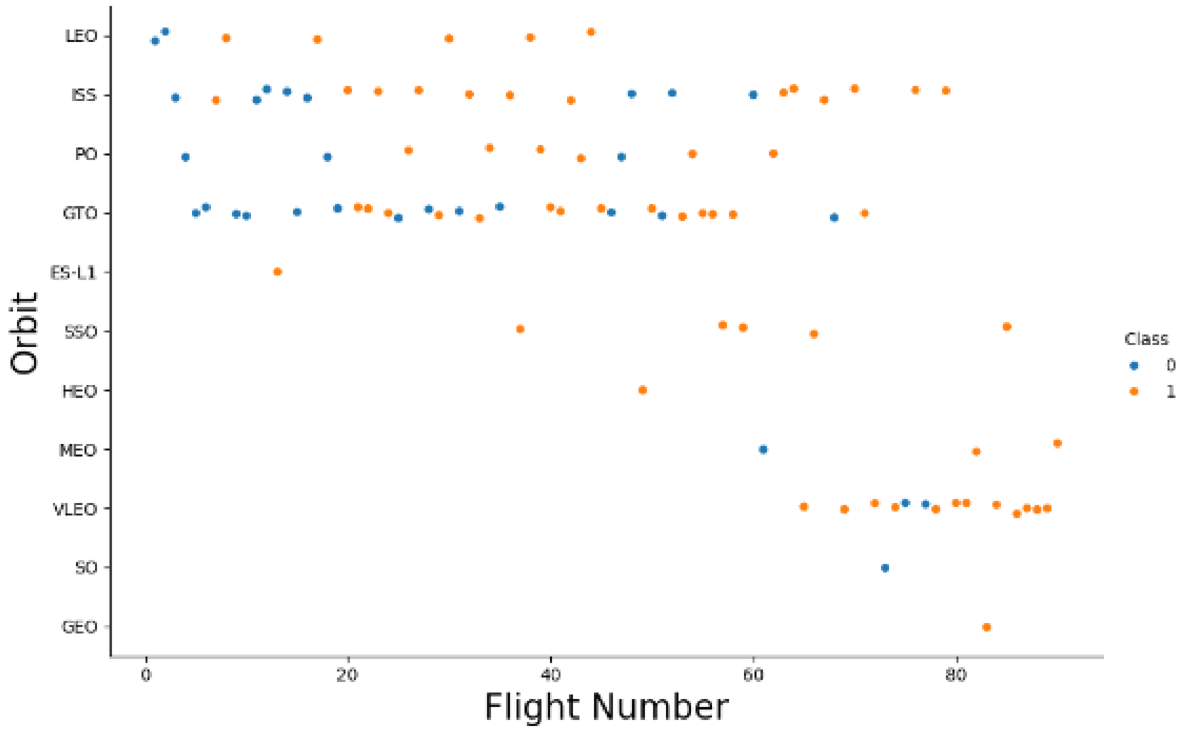
TASK 4: Visualize the relationship between FlightNumber and Orbit type

For each orbit, we want to see if there is any relationship between FlightNumber and Orbit type.

In [8]:

```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect = 1.5 ,height=6)

plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



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.

Payload vs. Orbit Type

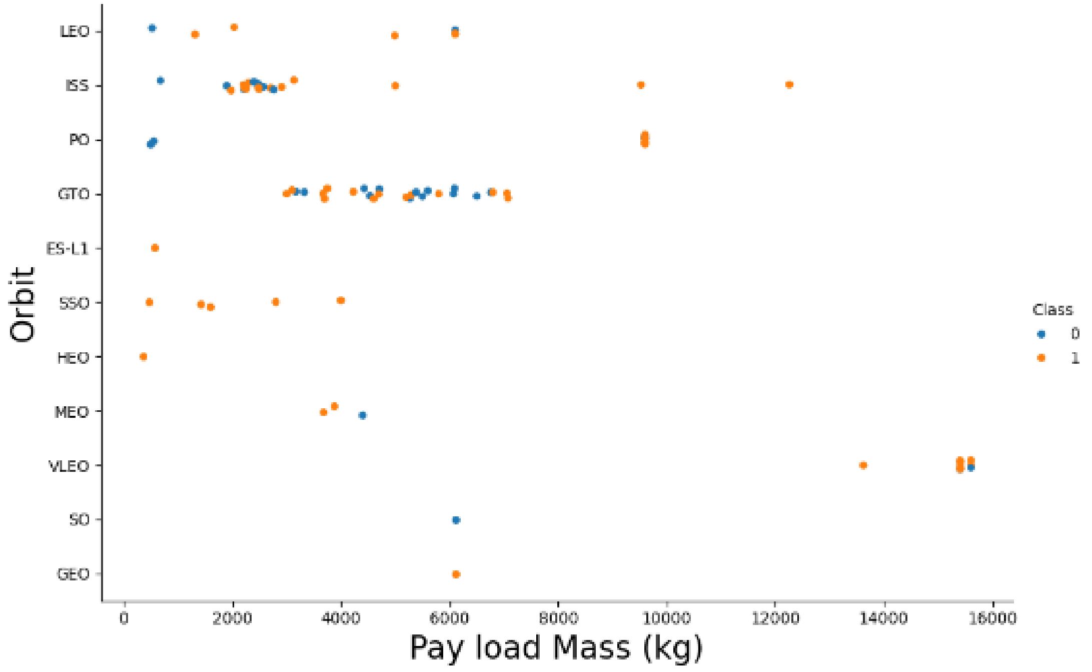
TASK 5: Visualize the relationship between Payload and Orbit type

Similarly, we can plot the Payload vs. Orbit scatter point charts to reveal the relationship between Payload and Orbit type

In [9]:

```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect = 1.5 ,height=6)

plt.xlabel("Pay load Mass (kg)",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.

Launch Success Yearly Trend

TASK 6: Visualize the launch success yearly trend

You can plot a line chart with x axis to be `Year` and y axis to be average success rate, to get the average launch success trend.

The function will help you get the year from the date:

In [10]:

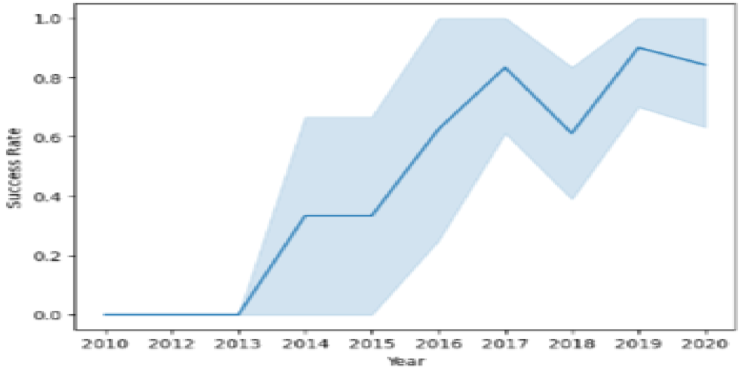
```
# A function to Extract years from the date
year=[]
def Extract_year():
    for i in df["Date"]:
        year.append(i.split("-")[0])
    return year
Extract_year()
df['Date'] = year
df.head()
```

Out[10]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad
0	1	2010	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN
1	2	2012	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN
2	3	2013	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN
3	4	2013	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN
4	5	2013	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN

In [11]:

```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
sns.lineplot(data=df, x="Date", y="Class")
plt.xlabel("Year")
plt.ylabel("Success Rate")
plt.show()
```



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

All Launch Site Names

Task 1

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

In [9]:

```
%sql select distinct Launch_Site from SPACEXTBL
```

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

Out[9]:

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

Launch Site Names Begin with 'CCA'

Task 2

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

In [10]:

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

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

Out[10]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	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)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

Task 3

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

In [11]:
```
%sql select sum(payload_mass__kg_) as sum from SPACEXTBL where customer like 'NASA (CRS)'
```

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

Out[11]:

sum
45596

Average Payload Mass by F9 v1.1

Task 4

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

In [12]:
```
%sql select avg(payload_mass__kg_) as average from SPACEXTBL where booster_version like 'F9 v1.1%'
```

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

Out[12]:

average
2534.6666666666665

First Successful Ground Landing Date

Task 5

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

Hint:Use min function

In [13]:
```
%sql select min(date) as date from SPACEXTBL where mission_outcome like 'Success'
```

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

Out[13]:

date
2010-06-04

Successful Drone Ship Landing with Payload between 4000 and 6000

Task 6

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

In [14]:

```
%sql select booster_version from SPACEXTBL where (mission_outcome like 'Success') and (payload_mass__kg_ between 4000 and 6(
```

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

Out[14]:

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

Total Number of Successful and Failure Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

In [15]:

```
%sql select mission_outcome, count(*) as count from SPACEXTBL group by mission_outcome order by mission_outcome
```

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

Out[15]:

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

Boosters Carried Maximum Payload

Task 8

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

In [16]:
```
%sql select booster_version from SPACEXTBL where payload_mass__kg_ = (select max(payload_mass__kg_) from SPACEXTBL)
```

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

Out[16]:

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

2015 Launch Records

Task 9

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

In [17]:

```
%sql select substr(DATE, 6,2) as month, Landing_Outcome, booster_version, launch_site from SPACEXTBL where date like '2015%
```

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

Out[17]:

month	Landing_Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

Task 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.

In [18]:

```
%sql select landing_outcome, count(*) as count from SPACEXTBL where date >= '2010-06-04' and date <= '2017-03-20' group by l
```

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

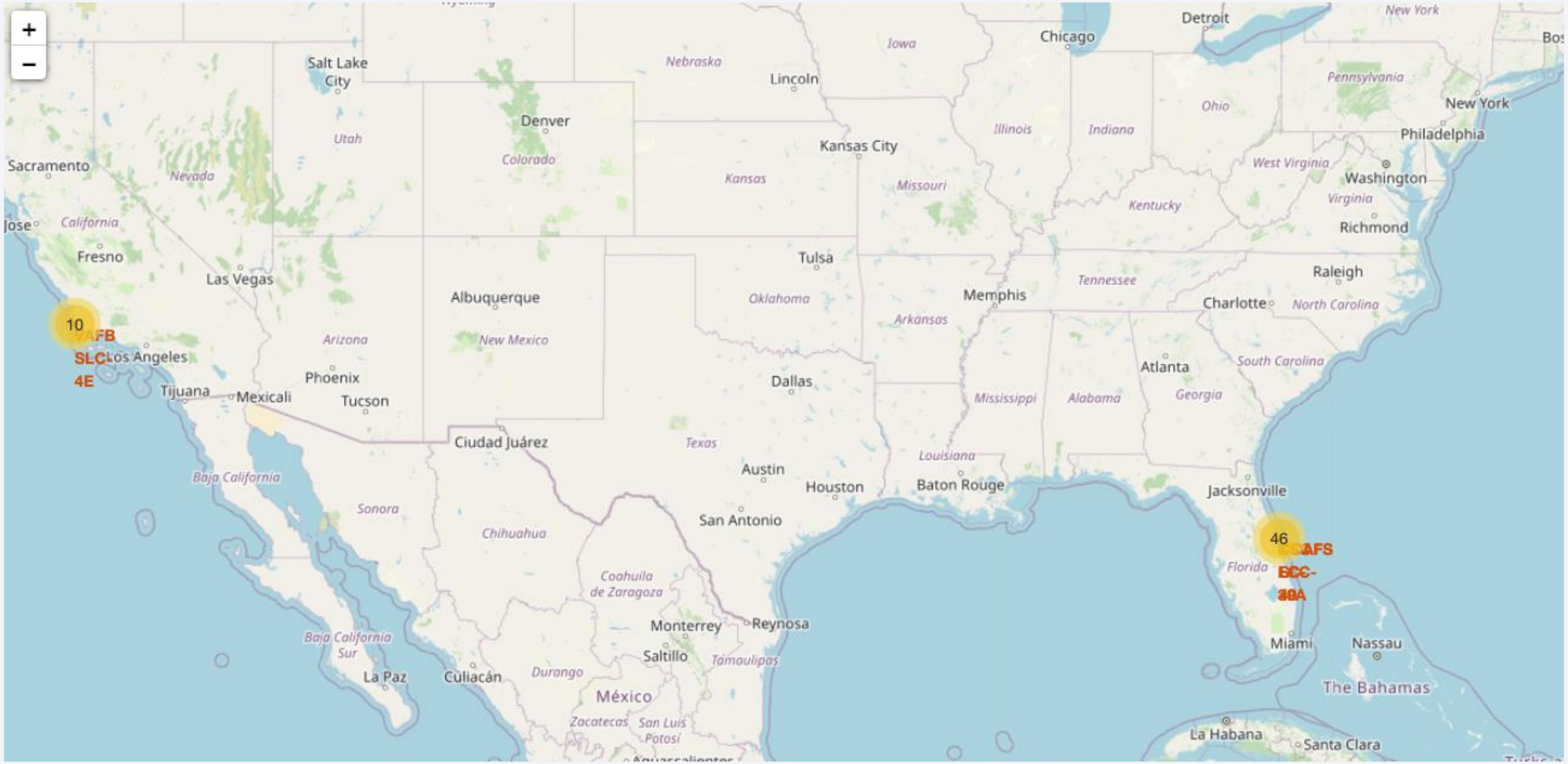
Out[18]:

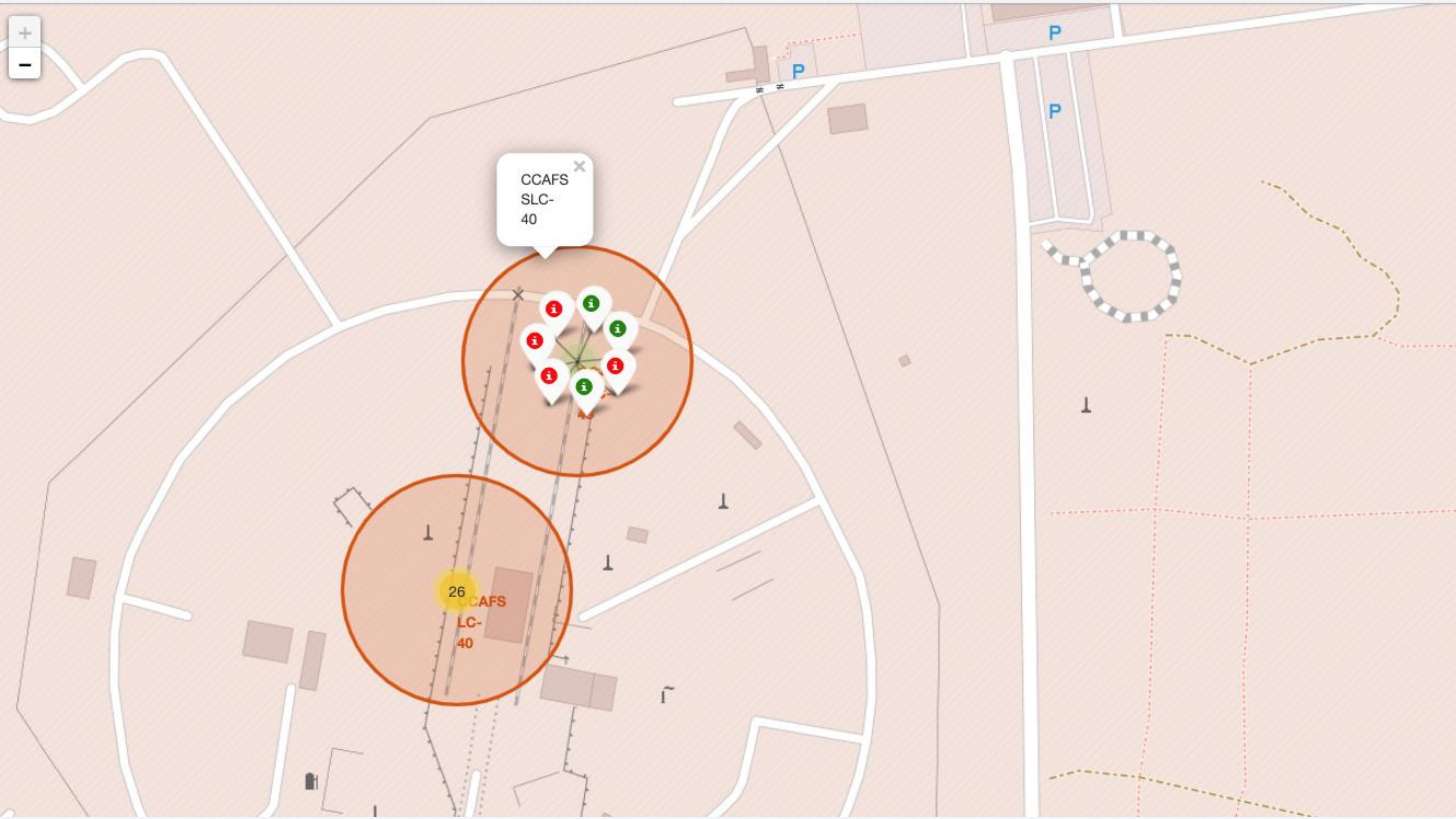
Landing_Outcome	count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

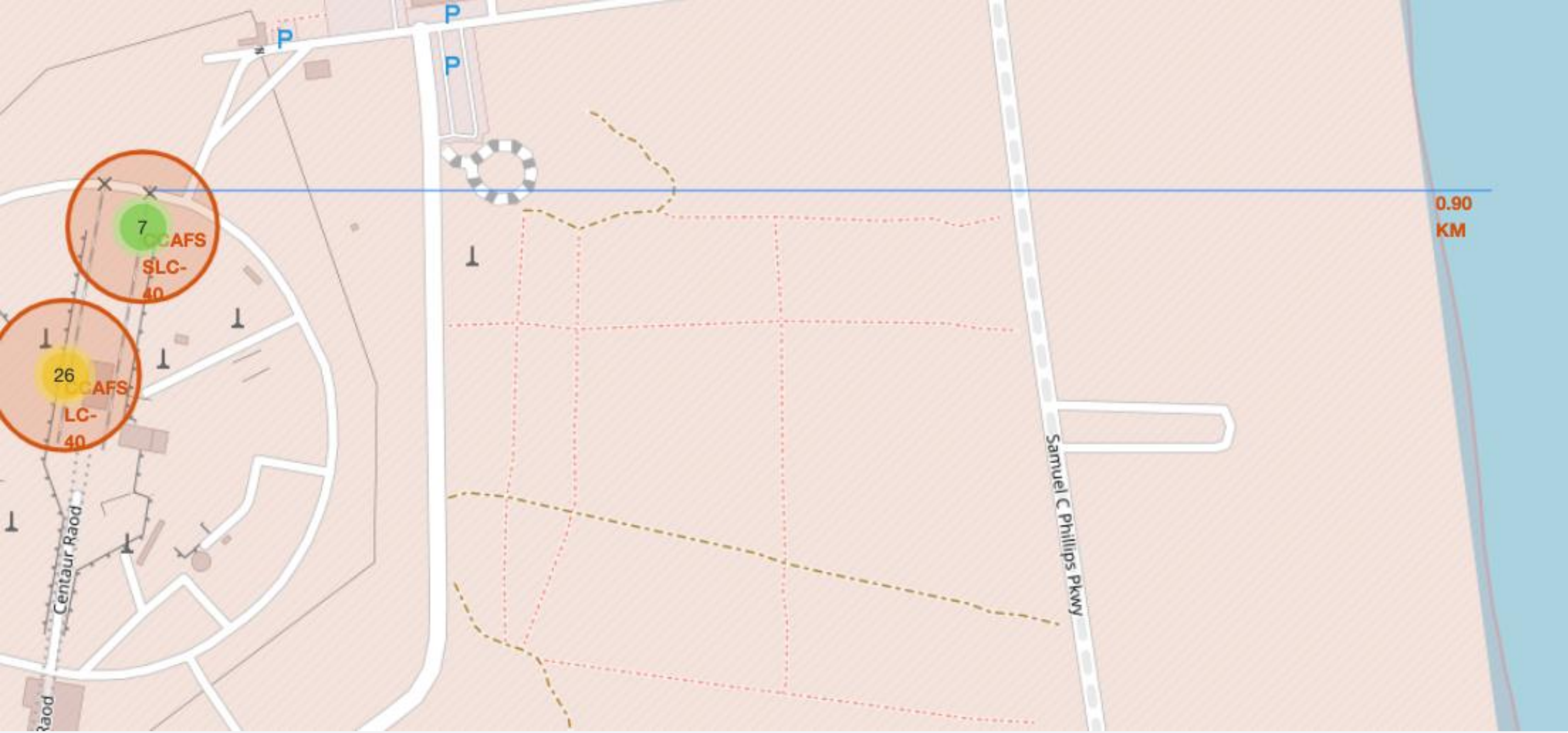
Section 3

Launch Sites Proximities Analysis

Markers of all launch sites




CCAFS
SLC-
40
26
CCAFS
LC-
40
P
P
P


P
P
P
7
CCAFS
SLC-
40
26
CCAFS
LC-
40
0.90
KM
Centaur Raod
Raod
Samuel C Phillips Pkwy

Section 4

Build a Dashboard with Plotly Dash

Dashboard

Applied-Data-Science-Capstone-IBM / 07_build_a_dashboard_application_with_plotly_dash

a-hognose-snake Update 07_build_a_dashboard_application_with_plotly_dash 0774464 · 3 hours ago History

Code Blame 88 lines (76 loc) · 3.63 KB Code 55% faster with GitHub Copilot Raw

```
# Import required libraries
import pandas as pd
import dash
import dash_html_components as html
import dash_core_components as dcc
from dash.dependencies import Input, Output
import plotly.express as px

# Read the airline data into pandas dataframe
spacex_df = pd.read_csv("spacex_launch_dash.csv")
max_payload = spacex_df['Payload Mass (kg)'].max()
min_payload = spacex_df['Payload Mass (kg)'].min()

# Create a dash application
app = dash.Dash(__name__)

# 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 Launch Site Drop-down Input Component
    dcc.Dropdown(id='site-dropdown',
                 options=[
```

SpaceX Launch Records Dashboard

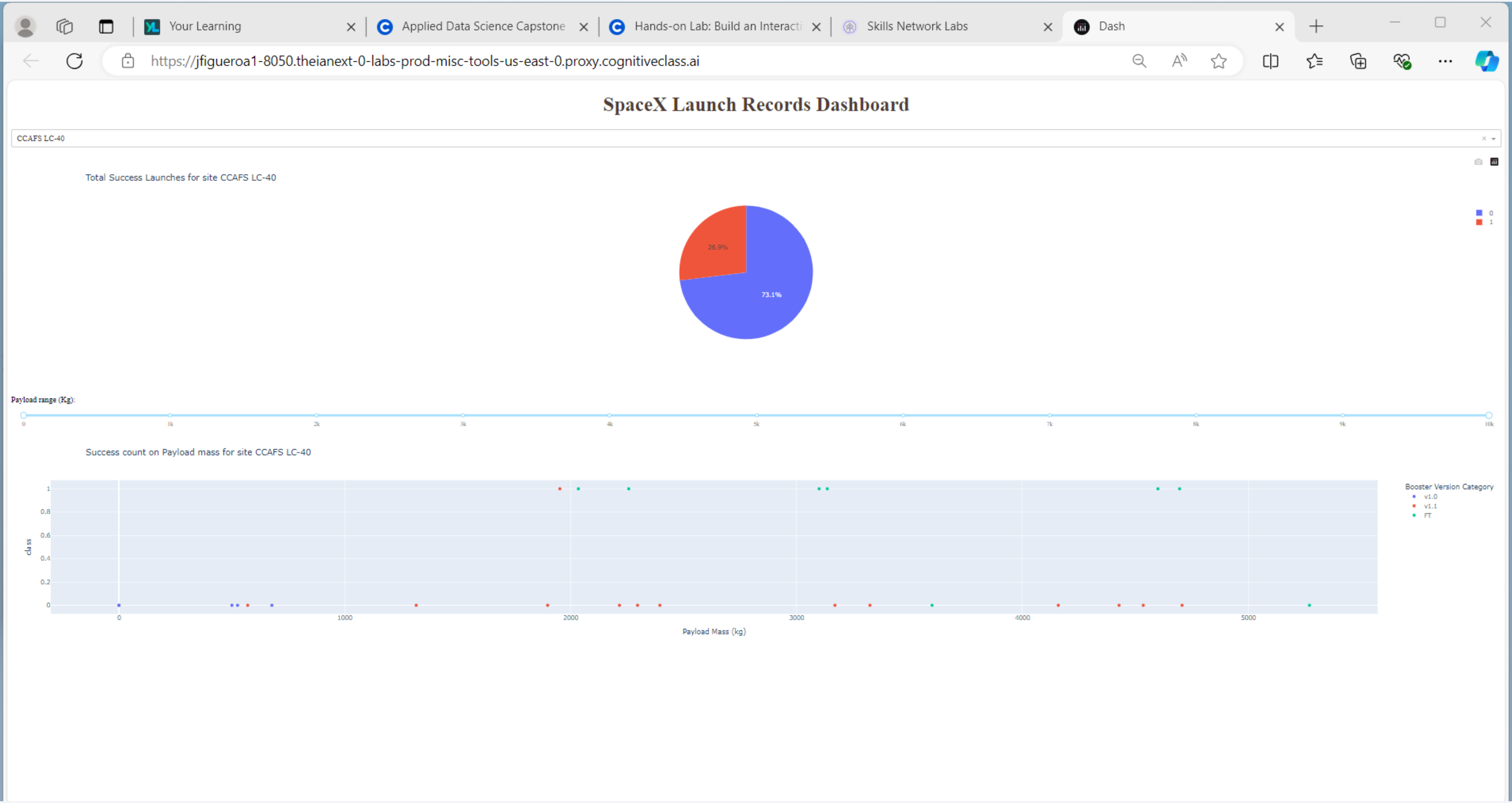
All Sites

Success Count for all launch sites

Payload range (Kg):

0 1k 2k 3k 4k 5k 6k 7k 8k 9k 10k

Success count on Payload mass for all sites


Your Learning
Applied Data Science Capstone
Hands-on Lab: Build an Interacti
Skills Network Labs
Dash
https://jfigueroa1-8050.theianext-0-labs-prod-misc-tools-us-east-0.proxy.cognitiveclass.ai
SpaceX Launch Records Dashboard
CCAFS LC-40
Total Success Launches for site CCAFS LC-40
26.9%
73.1%
0
1
Payload range (Kg):
0
1k
2k
3k
4k
5k
6k
7k
8k
9k
10k
Success count on Payload mass for site CCAFS LC-40
class
Payload Mass (kg)
Booster Version Category
v1.0
v1.1
FT

Section 5

Predictive Analysis (Classification)

Classification Accuracy

TASK 12

Find the method performs best:

In [37]:

```
Report = pd.DataFrame({'Method' : ['Test Data Accuracy']})

knn_accuracy=knn_cv.score(X_test, Y_test)
Decision_tree_accuracy=tree_cv.score(X_test, Y_test)
SVM_accuracy=svm_cv.score(X_test, Y_test)
Logistic_Regression=logreg_cv.score(X_test, Y_test)

Report['Logistic_Reg'] = [Logistic_Regression]
Report['SVM'] = [SVM_accuracy]
Report['Decision Tree'] = [Decision_tree_accuracy]
Report['KNN'] = [knn_accuracy]




Report.transpose()
```

Out[37]:

	0
Method	Test Data Accuracy
Logistic_Reg	0.833333
SVM	0.833333
Decision Tree	0.888889
KNN	0.833333

Confusion Matrix

TASK 9

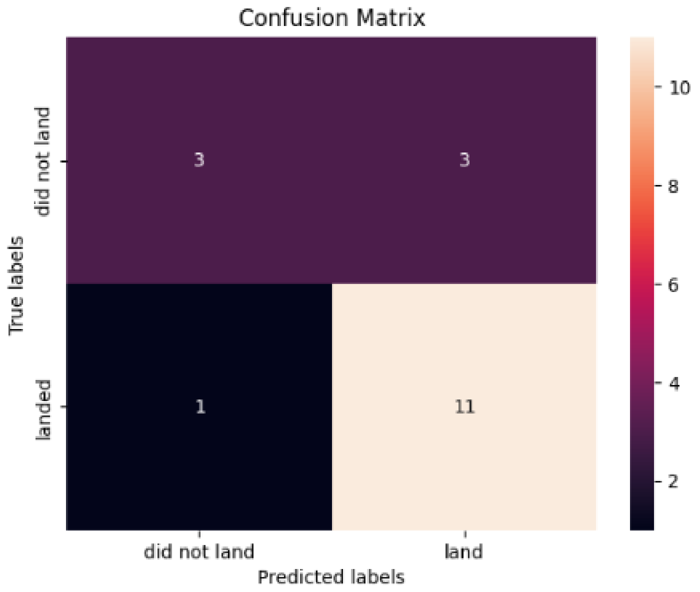
Calculate the accuracy of tree_cv on the test data using the method `score` :

In [26]:
```
print("Decision Tree accuracy on test set :",tree_cv.score(X_test, Y_test))
```

```
Decision Tree accuracy on test set : 0.7777777777777778
```

We can plot the confusion matrix

In [27]:
```
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
plt.show()
```



Conclusions

- Model Performance:
  - All four classification models had similar confusion matrices.
  - Each model was equally capable of distinguishing between the different classes.
  - The major issue across all models was false positives.
- Success Rates:
  - Success rates have been steadily increasing since 2013, reaching higher levels by 2020.
- Additional Insights:
  - Consistent improvement in model accuracy suggests effective feature engineering and model tuning.
  - Further reduction of false positives is crucial for enhancing model reliability.
  - The increasing trend in success rates highlights the potential for further optimizations and refinements in future models.

Thank you!