IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Sherri Nelson
September 29,2024

Outline

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

Executive Summary

- Data Collection: Collected data using SpaceX API and web scraping from Wikipedia.
- Data Wrangling: Applied one-hot encoding for categorical features like booster version and handled missing values.
- Exploratory Data Analysis: Conducted EDA using SQL and data visualization to explore relationships between flight number, launch sites, payload mass, and orbit types.
- Interactive Analytics: Created interactive visual analytics using Folium maps and Plotly Dash.
- Predictive Analysis: Built and tuned machine learning models (Logistic Regression, Decision Tree, SVM) to predict landing success.

Introduction

- Context: SpaceX aims to reduce costs by reusing the first stage of the Falcon 9 rocket, which can drastically lower the price of space missions.
- Problem: Predicting whether the Falcon 9 first stage will land successfully based on factors such as payload mass, orbit, and booster version.
- Questions: What factors influence landing success?
- How do various features interact to determine landing success?

Section 1

Methodology

Methodology

Executive Summary

- Data Collection:
  - Used SpaceX API to collect information on rocket launches.
  - Web scraped Wikipedia to gather historical launch records.
- Data Wrangling:
  - Cleaned data and filled missing values.
  - Performed one-hot encoding for categorical data.
- EDA:
  - Used SQL and Python to explore the dataset and identify key trends.
- Interactive Visualizations:
  - Created interactive maps with Folium to display launch locations and success rates.
  - Built a Plotly Dash dashboard to visualize the relationship between payload and success.
- Machine Learning:

  Built classification models (Logistic Regression, Decision Trees) and used GridSearchCV for hyperparameter tuning.

Data Collection

- API Data: Collected launch details such as booster version, landing outcome, and payload mass using the SpaceX API.
- Web Scraping: Parsed HTML tables from Wikipedia to retrieve Falcon 9 launch records using BeautifulSoup.

Data Collection – SpaceX API

- Process:
  - Used Python's requests library to fetch data from SpaceX API.
  - Cleaned and converted data to pandas DataFrame for further analysis.
- Notebook Link: https://github.com/capponelson/Applied-Data-Science-Capstone/blob/main/Collecting%20Date%20Reboot.ipynb

Below we will define a series of helper functions that will help us use the API to extract information using identification numbers in the launch data.

From the `rocket` column we would like to learn the booster name.

[54]:
```python
# Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
       if x:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

From the `launchpad` we would like to know the name of the launch site being used, the logitude, and the latitude.

[55]:
```python
# Takes the dataset and uses the launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['launchpad']:
       if x:
         response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
         Longitude.append(response['longitude'])
         Latitude.append(response['latitude'])
         LaunchSite.append(response['name'])
```

From the `payload` we would like to learn the mass of the payload and the orbit that it is going to.

[56]:
```python
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
       if load:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

From `cores` we would like to learn the outcome of the landing, the type of the landing, number of flights with that core, whether gridfins were used, w
specific core has been reused, and the serial of the core.

[57]:
```python
# Takes the dataset and uses the cores column to call the API and append the data to the lists
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
                Block.append(response['block'])
                ReusedCount.append(response['reuse_count'])
                Serial.append(response['serial'])
            else:
                Block.append(None)
                ReusedCount.append(None)
                Serial.append(None)
            Outcome.append(str(core['landing_success'])+' '+str(core['landing_type']))
            Flights.append(core['flight'])
            GridFins.append(core['gridfins'])
            Reused.append(core['reused'])
            Legs.append(core['legs'])
            LandingPad.append(core['landpad'])
```

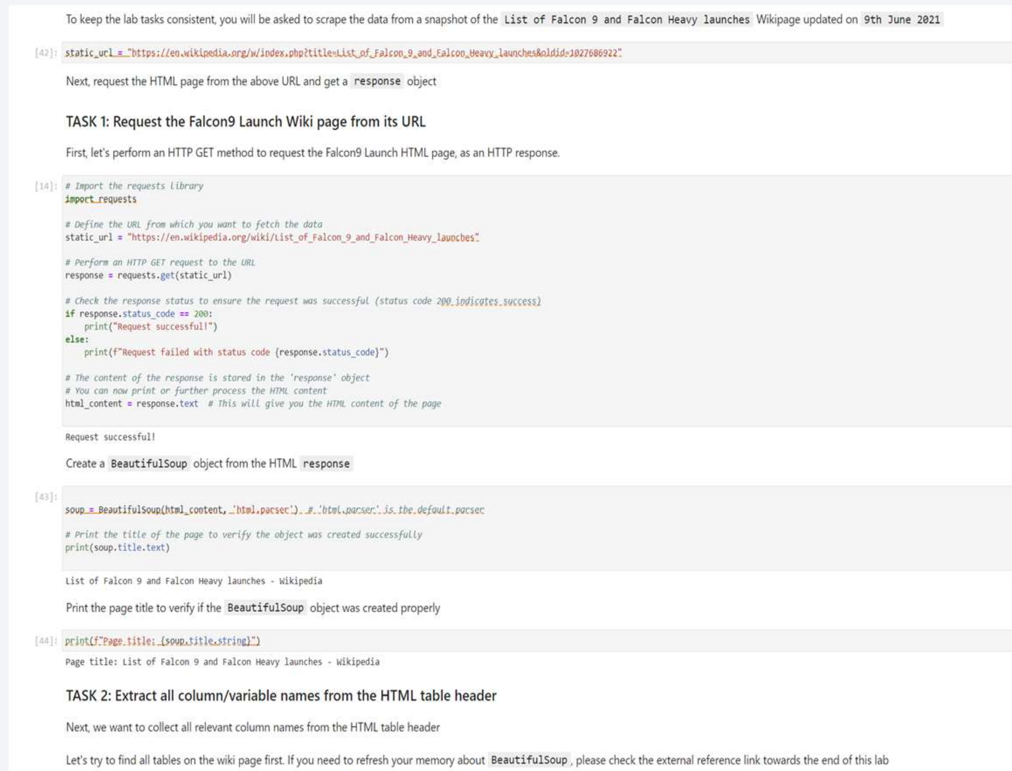
Now let's start requesting rocket launch data from SpaceX API with the following URL:

[58]:
```python
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

[60]:
```python
response = requests.get(spacex_url)
```

Data Collection - Scraping

- Process:
  - Used BeautifulSoup to extract launch records from Wikipedia.
  - Parsed HTML tables and converted them to DataFrame for analysis.
- Notebook Link: https://github.com/capponelson/Applied-Data-Science-Capstone/blob/main/Web%20scraping%20Falcon%209%20and%20Falcon%20Heavy.ipynb


To keep the lab tasks consistent, you will be asked to scrape the data from a snapshot of the `List of Falcon 9 and Falcon Heavy launches` Wikipage updated on `9th June 2021`

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

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

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

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

```
[14]: # Import the requests library
import requests

# Define the URL from which you want to fetch the data
static_url = "https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches"

# Perform an HTTP GET request to the URL
response = requests.get(static_url)

# Check the response status to ensure the request was successful (status code 200 indicates success)
if response.status_code == 200:
    print("Request successful!")
else:
    print(f"Request failed with status code {response.status_code}")

# The content of the response is stored in the 'response' object
# You can now print or further process the HTML content
html_content = response.text  # This will give you the HTML content of the page
```

```
Request successful!
```

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

```
[43]:
soup = BeautifulSoup(html_content, 'html.parser')  # 'html.parser' is the default parser

# Print the title of the page to verify the object was created successfully
print(soup.title.text)
```

```
List of Falcon 9 and Falcon Heavy launches - Wikipedia
```

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

```
[44]: print(f"Page title: {soup.title.string}")
```

```
Page title: List of Falcon 9 and Falcon Heavy launches - Wikipedia
```

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

Data Wrangling

- Process:
  - Cleaned the data by checking for missing values and applying transformations.
  - Created the landing outcome column and exported results to CSV for further use.
- Notebook Link: https://github.com/capponelson/Applied-Data-Science-Capstone/blob/main/Data%20Wrangling.ipynb

Identify and calculate the percentage of the missing values in each attribute

```
[6]: df.isnull().sum()/len(df)*100
```

```
[6]: 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
```

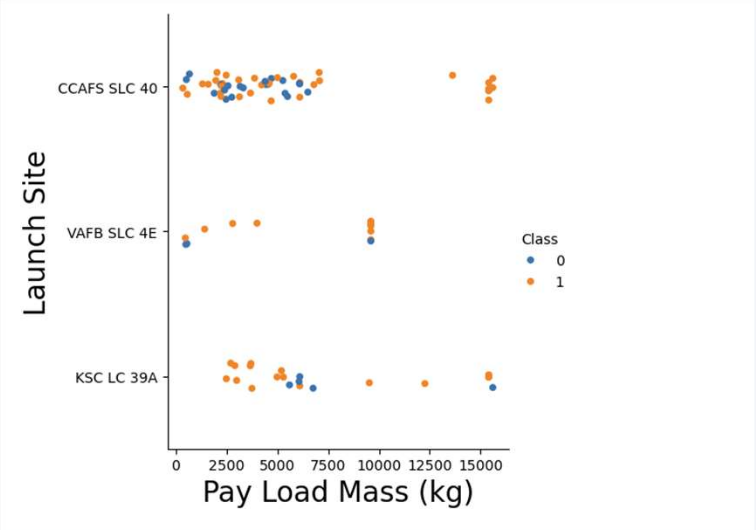
Identify which columns are numerical and categorical:

```
[7]: df.dtypes
```

```
[7]: FlightNumber        int64
     Date               object
     BoosterVersion     object
     PayloadMass       float64
     Orbit              object
     LaunchSite         object
     Outcome            object
     Flights             int64
     GridFins             bool
     Reused               bool
     Legs                 bool
     LandingPad         object
     Block             float64
     ReusedCount         int64
     Serial             object
     Longitude         float64
     Latitude          float64
     dtype: object
```

EDA with Data Visualization

- Queries:
- Found unique launch sites, calculated payload mass for different boosters, and identified mission outcomes.
- Notebook Link: https://github.com/capponelson/Applied-Data-Science-Capstone/blob/main/EDA%20with%20Visualization.ipynb



EDA with SQL

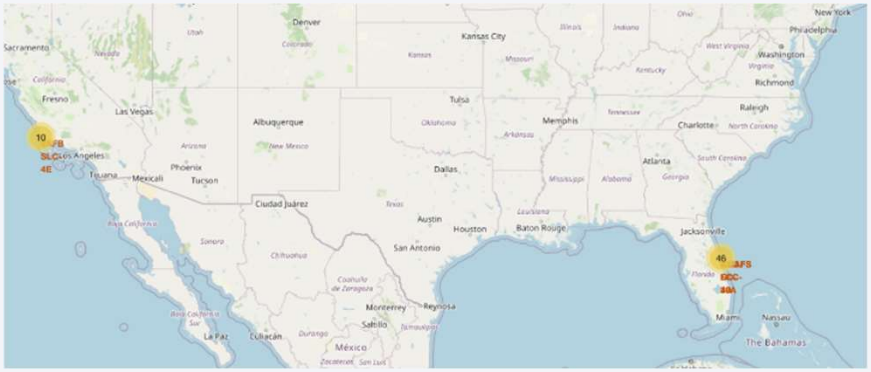
- Queries:
  - Found unique launch sites, calculated payload mass for different boosters, and identified mission outcomes.
  - Notebook Link: https://github.com/capponelson/Applied-Data-Science-Capstone/blob/main/SQL%20Notebook%20for%20Peer%20Assignment.ipynb

```
In [1]: !pip install sqlalchemy==1.3.9

Collecting sqlalchemy==1.3.9
  Downloading SQLAlchemy-1.3.9.tar.gz (6.0 MB)
     ---------------------------------------- 6.0/6.0 MB 77.9 MB/s eta 0:00:00:00:010:01
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: sqlalchemy
  Building wheel for sqlalchemy (setup.py) ... done
  Created wheel for sqlalchemy: filename=SQLAlchemy-1.3.9-cp311-cp311-linux_x86_64.whl size=1142923 sha256=f6
cfd2769ca175e2765b88bb92cb5df1a34500df3eda725306b9be6daeb4e7d9
  Stored in directory: /home/jupyterlab/.cache/pip/wheels/3a/7c/1e/12404784a68083eb969f877a1808a1847bab897684
b56ddc55
Successfully built sqlalchemy
Installing collected packages: sqlalchemy
  Attempting uninstall: sqlalchemy
    Found existing installation: SQLAlchemy 2.0.30
    Uninstalling SQLAlchemy-2.0.30:
      Successfully uninstalled SQLAlchemy-2.0.30
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. Th
is behaviour is the source of the following dependency conflicts.
jupyterhub 4.1.5 requires SQLAlchemy>=1.4, but you have sqlalchemy 1.3.9 which is incompatible.
Successfully installed sqlalchemy-1.3.9
```

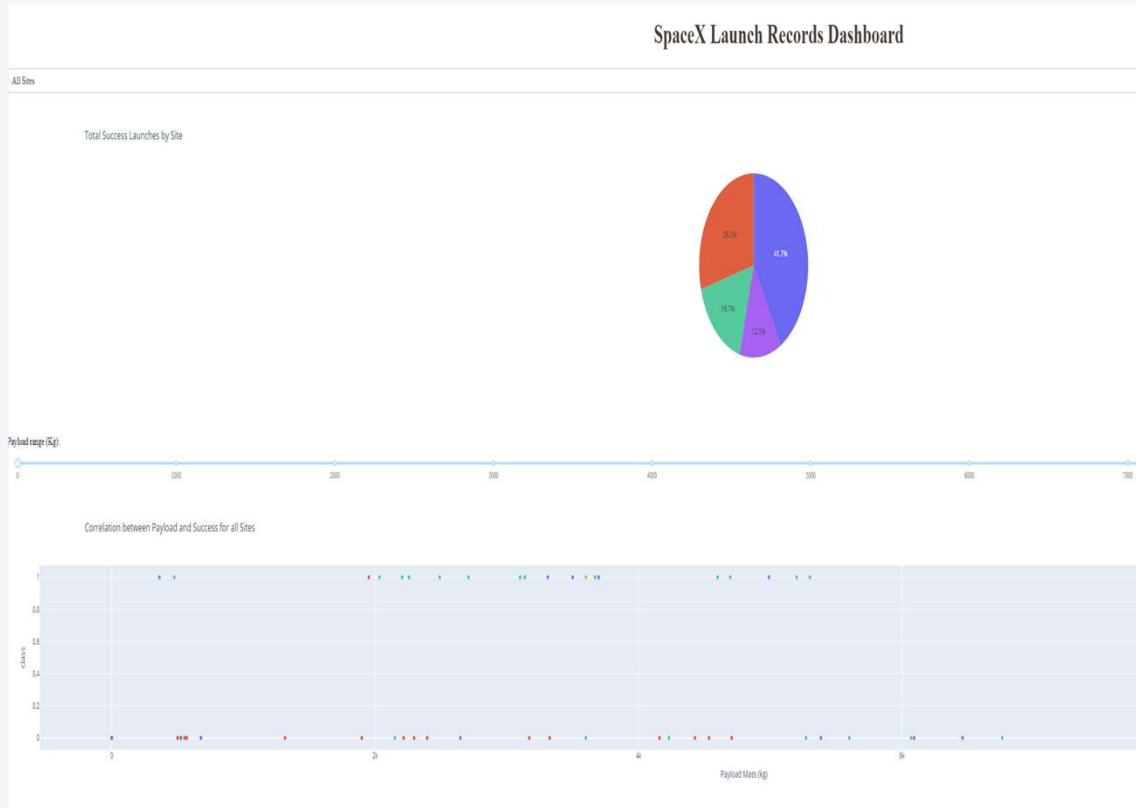
Build an Interactive Map with Folium

- Process:
- Mapped all launch sites and marked the success or failure of launches.Analyzed launch site proximity to highways, railways, and cities.
- Create a `folium.PolyLine` object using the coastline coordinates and launch site coordinate
- Create a marker with distance to a closest city, railway, highway relative to CCAFS SLC-40
- Notebook Link: https://github.com/capponelson/Applied-Data-Science-Capstone/blob/main/Launch%20Sites%20Locations%20Analysis%20with%20Folium.ipynb



Build a Dashboard with Plotly Dash

- Visuals:
- Plotted success rates and payload vs. outcome relationships across launch sites.
- Notebook Link: https://github.com/capponelson/Applied-Data-Science-Capstone/blob/main/Dashboard%20with%20Plotly%20Dash.py



Predictive Analysis (Classification)

- Process:
- Built classification models to predict landing success, using GridSearchCV for tuning.
- Evaluated models based on accuracy and selected the best performing one.
- Notebook Link: https://github.com/capponelson/Applied-Data-Science-Capstone/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

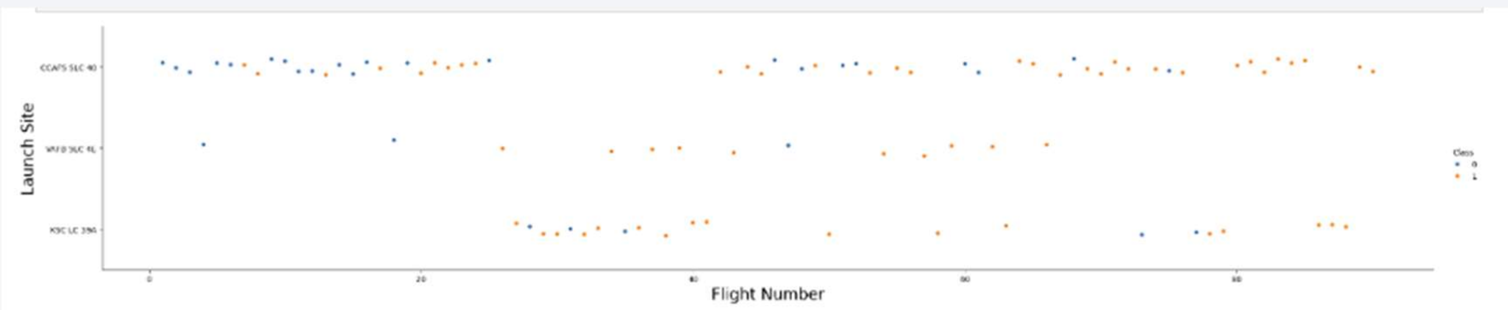
- Key Findings:
- Larger payloads are associated with lower success rates.
- Certain orbits (e.g., LEO, GEO) have higher success rates.
- The Decision Tree classifier had the highest accuracy for landing prediction.

Section 2

Insights drawn from EDA

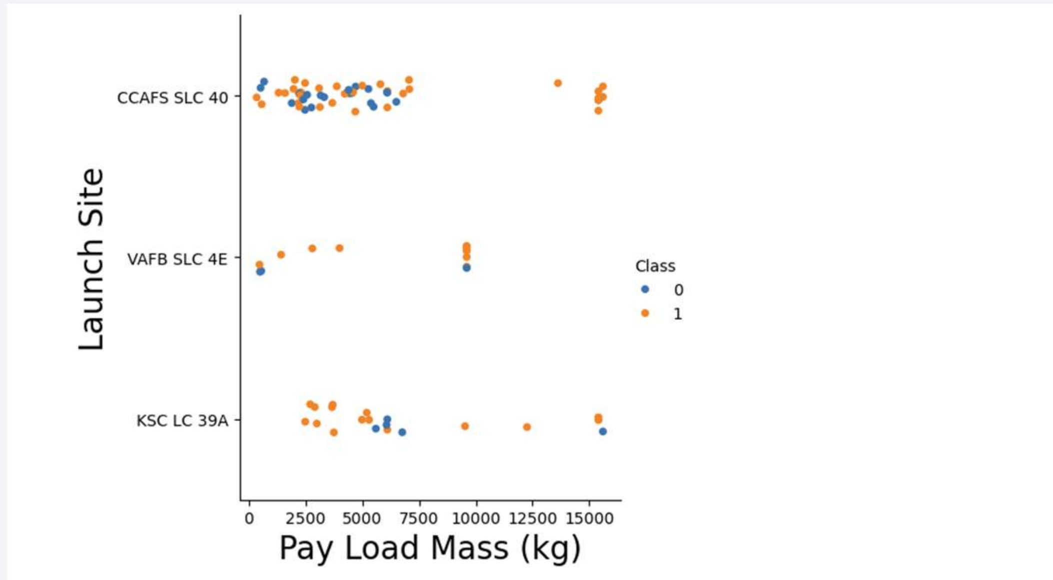
Flight Number vs. Launch Site

- Show a scatter plot of Flight Number vs. Launch Site
- Show the screenshot of the scatter plot with explanations
  - Flight Number vs. Launch Site shows a positive trend in landing success over time, particularly at the KSC LC-39A and CCAFS SLC-40 sites



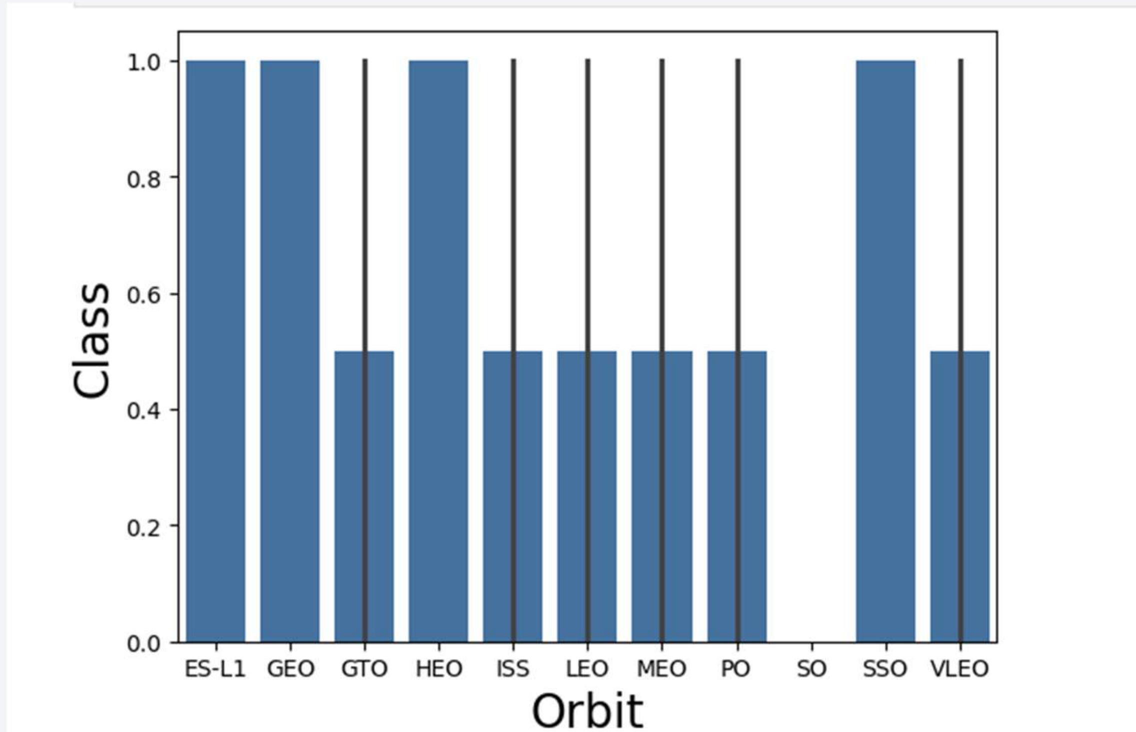
Payload vs. Launch Site

- Show a scatter plot of Payload vs. Launch Site
- Show the screenshot of the scatter plot with explanations
  - CCAFS SLC-40 is the most reliable site for launching heavy payloads, whereas KSC LC-39A excels with lighter and moderately heavy payloads.



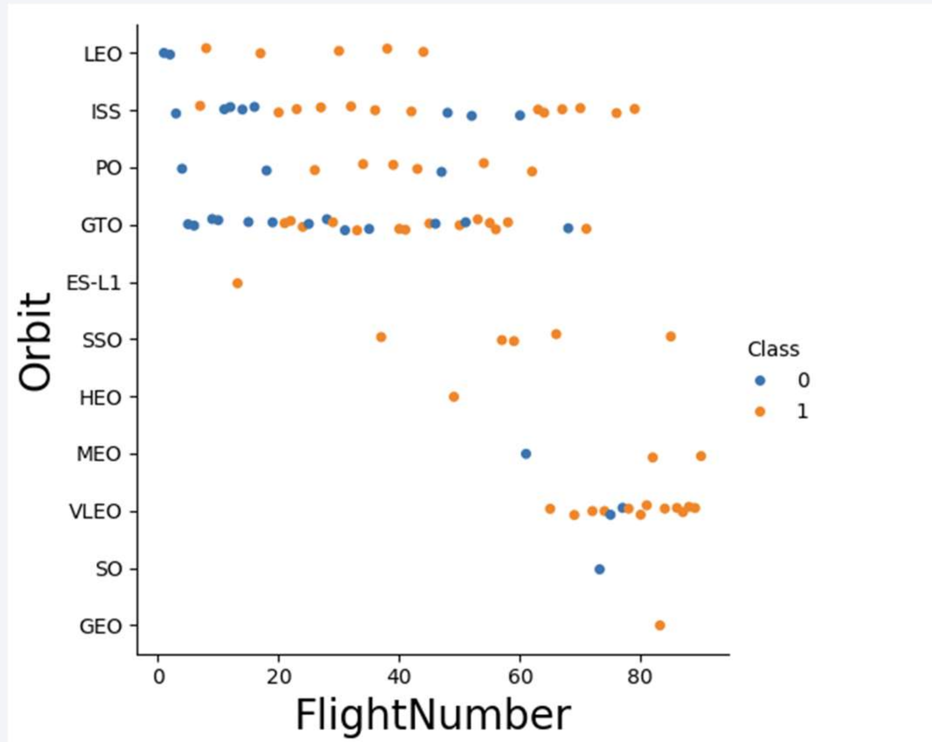
Success Rate vs. Orbit Type

- Show a bar chart for the success rate of each orbit type
- Show the screenshot of the scatter plot with explanations
  - Orbits such as ES-L1, GEO, HEO, PO, SO, and SSO have near-perfect success rates
  - GTO, LEO, and VLEO show lower success rates, suggesting that these orbits may involve more challenging landing condition



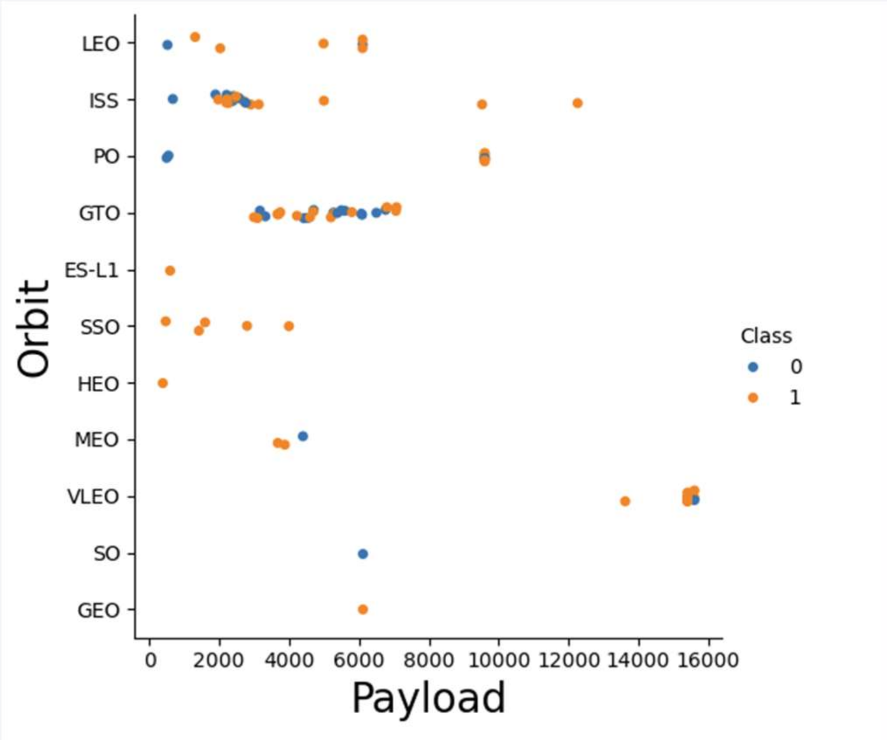
Flight Number vs. Orbit Type

- Show a scatter point of Flight number vs. Orbit type
- Show the screenshot of the scatter plot with explanations
  - In orbits such as LEO and ISS, SpaceX's success rate improves with higher flight numbers
  - GTO orbit seems to pose challenges, as success rates remain mixed across all flight numbers



Payload vs. Orbit Type

- Show a scatter point of payload vs. orbit type
- Show the screenshot of the scatter plot with explanations
  - With heavy payloads, successful landing rates are higher for Polar, LEO, and ISS missions. However, for GTO missions, it is challenging to differentiate between successful and unsuccessful landings, as both outcomes are observed.



Launch Success Yearly Trend

- Show a line chart of yearly average success rate
- Show the screenshot of the scatter plot with explanations

All Launch Site Names

- Find the names of the unique launch sites
- DISTINCT was used to show unique sites

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

```
[27]: %sql SELECT DISTINCT(LAUNCH_SITE) FROM SPACEXTBL;
 * sqlite:///my_data1.db
Done.
```

[27]:

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

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`
- SQL query was used with a limit 5 to display 5 records

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

[29]:
```
# Display 5 records where launch sites begin with "CCA"
%sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

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

[29]:

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

- Calculate the total payload carried by boosters from NASA
  - Total Payload and Average showed

Task 3

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

In [30]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE CUSTOMER = 'NASA (CRS)';
```

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

Out[30]:

SUM(PAYLOAD_MASS__KG_)
45596

Task 4

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

In [31]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE BOOSTER_VERSION = 'F9 v1.1';
```

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

Out[31]:

AVG(PAYLOAD_MASS__KG_)
2928.4

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1
- Average Payload shown

```
In [31]: %sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE BOOSTER_VERSION = 'F9 v1.1';

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

Out[31]:

AVG(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad
- First successful landing in December 2015

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

Hint:Use min function

In [32]:
```
%sql SELECT MIN(DATE) FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (ground pad)';
```

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

Out[32]:

MIN(DATE)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000
- Retrieves a list of booster versions that have successfully landed on a drone ship and carried a payload mass between 4000 kg and 6000 kg

```
In [33]: %sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 AND P
```

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

Out[33]:

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

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes
  - There were 38 successful landings in total (without specifying where they landed), 14 of which were successful landings on drone ships, and 9 on ground pads. This shows that more than half of the missions had a successful recovery.

List the total number of successful and failure mission outcomes

In [34]:
```
%sql SELECT LANDING_OUTCOME, COUNT(*) AS Total_Missions FROM SPACEXTBL GROUP BY LANDING_OUTCOME;
```

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

Out[34]:

Landing_Outcome	Total_Missions
Controlled (ocean)	5
Failure	3
Failure (drone ship)	5
Failure (parachute)	2
No attempt	21
No attempt	1
Precluded (drone ship)	1
Success	38
Success (drone ship)	14
Success (ground pad)	9
Uncontrolled (ocean)	2

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass
  - The results show multiple boosters (e.g., F9 B5 B1048.4, F9 B5 B1049.4, etc.) that carried the maximum payload.
  - The booster version format (e.g., F9 B5 B1048.4) refers to SpaceX's Falcon 9 rocket (F9), Block 5 (B5), and the specific booster number (B1048.4).

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

In [35]:
```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

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

Out[35]:

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

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
  - The query only includes records where the landing outcome was a "Failure (drone ship)"

```
In [36]: %sql SELECT substr(DATE, 6, 2) AS Month, BOOSTER_VERSION, LAUNCH_SITE, LANDING_OUTCOME \
FROM SPACEXTBL \
WHERE LANDING_OUTCOME = 'Failure (drone ship)' AND substr(DATE, 1, 4) = '2015';

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

Out[36]:

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

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

- 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
  - "No attempt" is the most common outcome during this period, reflecting that SpaceX didn't attempt to recover the booster for 10 missions.
  - The equal distribution of success and failure on drone ships (5 each) shows the early experimental nature of SpaceX's attempts to land on drone ships.

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 [37]:

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

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

Out[37]:

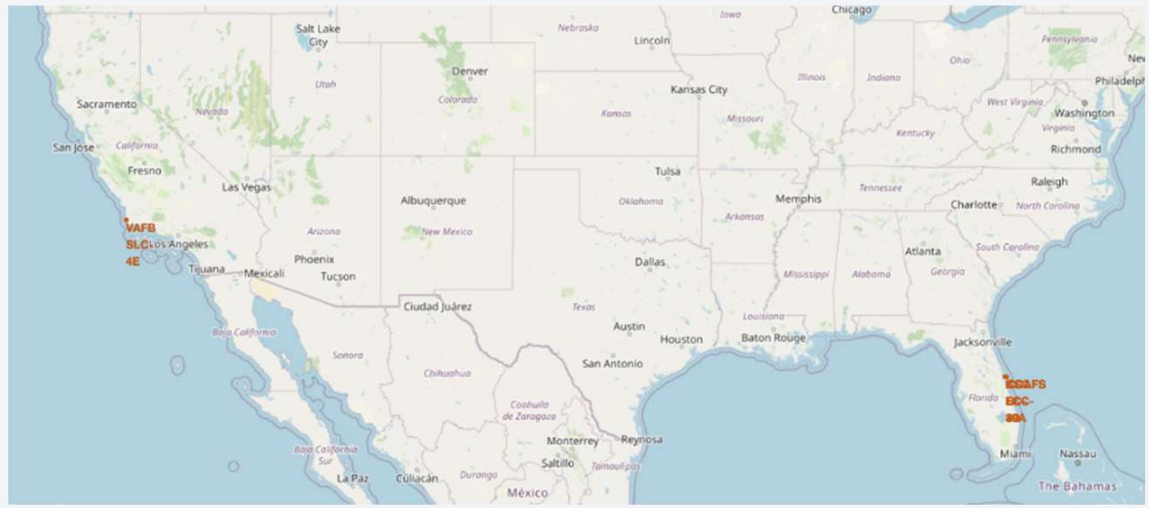
Landing_Outcome	Outcome_Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

Location for Launch Sites

- The Equator runs at 0° latitude, and none of the launch sites shown on the map are near this line. The United States is far north of the Equator, with the closest launch sites being located in Florida, which is still around 28° latitude north of the Equator.
- launch sites are located very close to the coast.

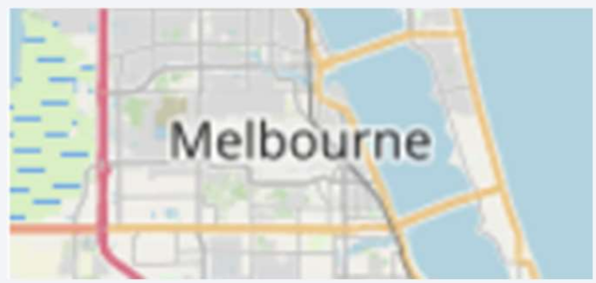


Markers showing sites (Color coded)

- Green – Successful Launches
- Red – Failed Launches

Launch Site Proximity

- SpaceX launch site (LC-40 at Cape Canaveral Air Force Station) with a distance line indicating a distance of 0.90 km to a certain feature on the map
- Railway and highway map symbols to indicate nearby transportation infrastructure.

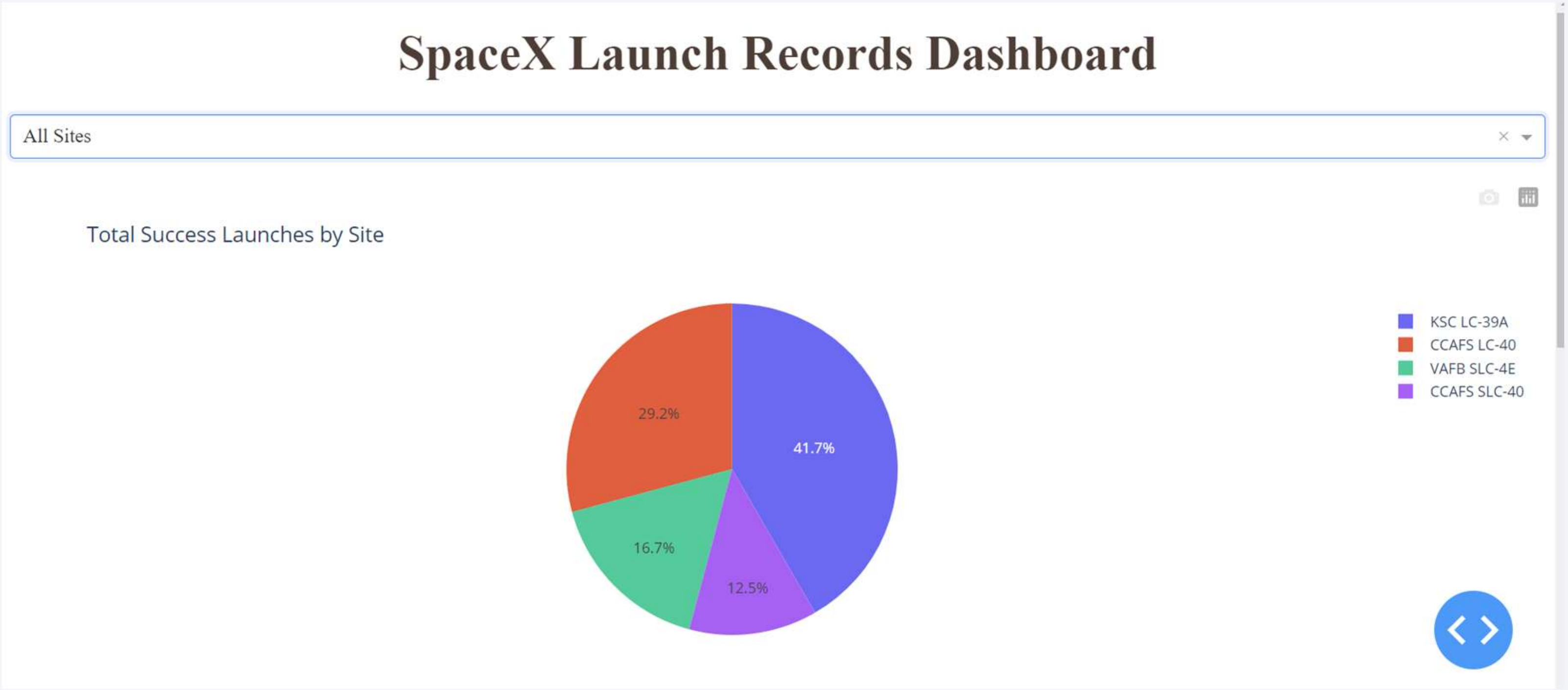


Section 4

Build a Dashboard with Plotly Dash

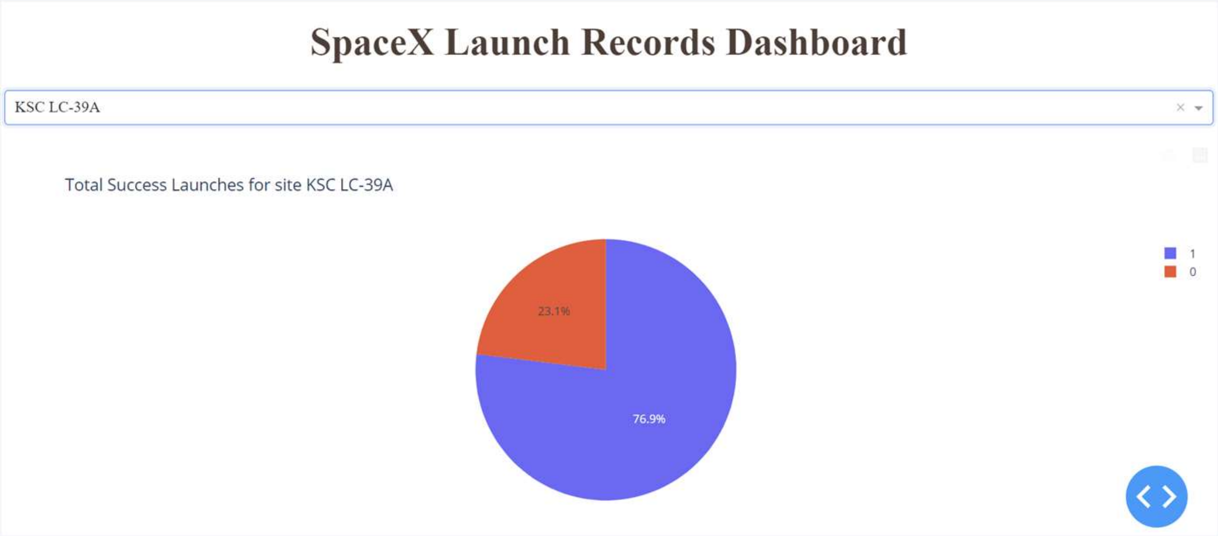
Total Success of Launches by Site

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



Piechart for the launch site with highest launch success ratio

- Piechart for the launch site with highest launch success ratio



Payload vs. Launch Outcome scatter plot

- payload mass range slider at the top allows the user to filter the payloads based on weight, ranging from 0 kg to 9000 kg. In the current view, the graph displays payloads between around 2000 kg and 7000 kg.
- The y-axis represents the success rate, where:1: Indicates a successful mission.0: Indicates a failure.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- Decision tree classifier that finds the best parameters from the dictionary parameters.

```
In [41]: 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()

In [42]: tree_cv = GridSearchCV(tree,parameters,cv=10)
         tree_cv.fit(X_train, Y_train)

Out[42]: GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
                      param_grid={'criterion': ['gini', 'entropy'],
                                  'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18],
                                  'max_features': ['auto', 'sqrt'],
                                  'min_samples_leaf': [1, 2, 4],
                                  'min_samples_split': [2, 5, 10],
                                  'splitter': ['best', 'random']})
```

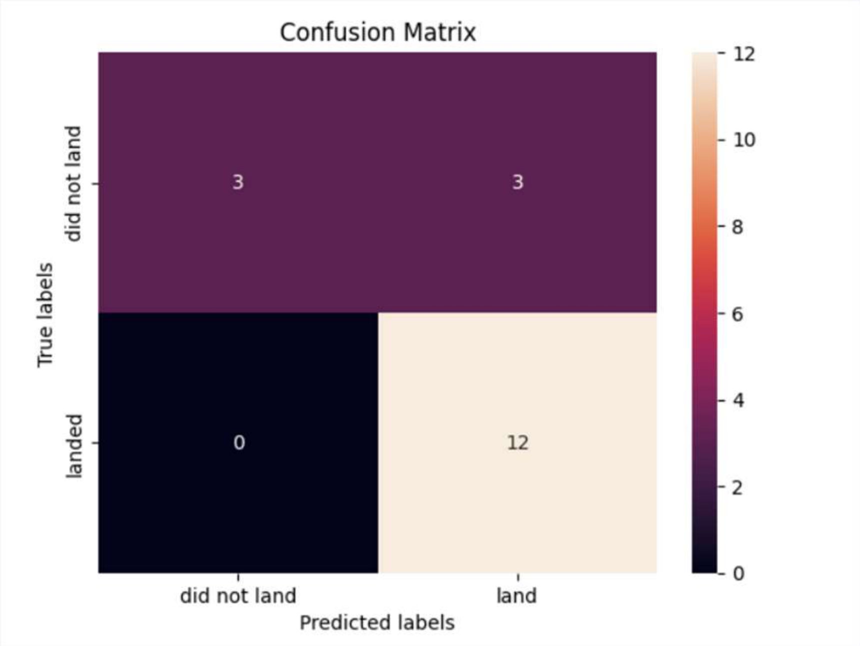
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [43]: print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
         print("accuracy :",tree_cv.best_score_)

tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 2, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
accuracy : 0.875
```

Confusion Matrix

- Logistic regression successfully differentiates between the various classes, but the main issue lies in the false positives.
- Overview:
  - True Positive (TP) - 12: The true label is "landed," and the model correctly predicts "landed.“
  - False Positive (FP) - 3: The true label is "not landed," but the model incorrectly predicts "landed."



Conclusions

- Successful Reusability of Boosters: The data shows that SpaceX majority of the launches have been successful, with a significant number of missions having successful booster recoveries on both ground pads and drone ships.
- Payload Mass and Landing Success: Analysis revealed a trend where larger payloads tend to have lower success rates.
- Importance of Launch Site: Certain launch sites, particularly KSC LC-39A and CCAFS SLC-40, have shown higher success rates.
- Predictive Model Performance: Among the machine learning models tested, the Decision Tree classifier showed the highest accuracy in predicting landing success.


Thank you!