Space X Falcon 9 First Stage Landing

By

George Sylva

OUTLINE

- Executive Summary
- Introduction
- Methodology
  - Data collection using API
  - Data collection using BeautifulSoup
- Results
  - Visualization – Charts
  - Eda using SQL
  - Insights drawn
  - Interactive maps with Folium
- Discussion
  - Findings & Implications
- Conclusion

EXECUTIVE SUMMARY

- Summary of method
  - Data Collection through API
  - Data Collection using BeautifulSoup
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Interactive Visualization using Folium

- Summary of Results

INTRODUCTION

- **Project Background**

SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch. In this lab, you will collect and make sure the data is in the correct format from an API. The following is an example of a successful and launch.

- **Problem we want to solve**
  - What factors determine if the rocket will land successfully?
  - The interaction amongst various features that determine the success rate of a successful landing.
  - What operating conditions needs to be in place to ensure a successful landing program.

METHODOLOGY

- Data collection methodology:
  - Data was collected using SpaceX API and web scraping from Wikipedia.
  - Data was also collected using BeautifulSoup
- Perform data wrangling
  - One-hot encoding was applied to categorical features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification model

Data collection using API

Load the data

In [4]:
```
from js import fetch
import io

URL1 = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_2.csv
resp1 = await fetch(URL1)
text1 = io.BytesIO((await resp1.arrayBuffer()).to_py())
data = pd.read_csv(text1)
```

In [5]:
```
data.head()
```

Out[5]:

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

In [6]:
```
data.info()
```

Here we try to extract Data using APIs. Typically this involves making a request to a web server using a specific URL or endpoint, along with any necessary parameters or authentication credentials. The server will then respond with the requested data in a standardized format such as JSON or XML.

Data extraction using BeautifulSoup

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

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

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

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

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

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

```
In [7]: # Use soup.title attribute
        page_title = soup.title.string
        print("Page Title:", page_title)

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

Here we us BeautifulSoup which is a Python library to parses HTML and XML documents and provides methods to extract data from them. It allows us to navigate the document structure, locate specific elements, and extract their contents.

BeautifulSoup is a popular tool for web scraping and data extraction used in data science, machine learning, and other fields.

RESULTS

- Data exraction was done using API and BeautifulSoup
- Then Exploratory Data Analysis was then carried out with python and SQL

Data Analysis

```
from js import fetch
import io

URL = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv'
resp = await fetch(URL)
dataset_part_1_csv = io.BytesIO((await resp.arrayBuffer()).to_py())
```

Load Space X dataset, from last section.

```
df=pd.read_csv(dataset_part_1_csv)
df.head(10)
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003

Task 2

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

In [14]:
```
%%sql

SELECT *
FROM SPACEXTBL
WHERE Launch_Site LIKE 'CCA%'
LIMIT 5;
```

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

Out[14]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
06/04/2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX	Success	Failure (parachute)
12/08/2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0.0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
22/05/2012	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525.0	LEO (ISS)	NASA (COTS)	Success	No attempt
10/08/2012	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	NASA (CRS)	Success	No attempt
03/01/2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	NASA (CRS)	Success	No attempt

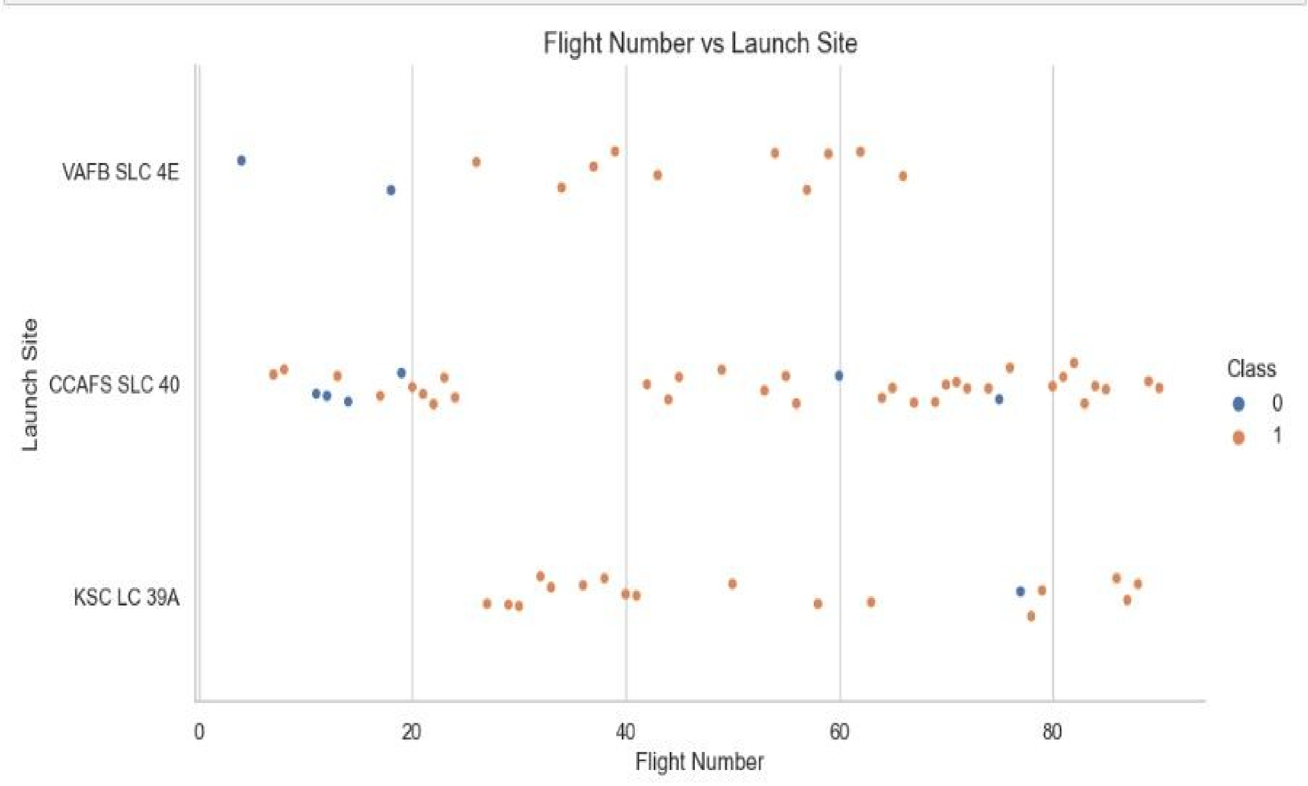
RESULTS

Visualisation Charts

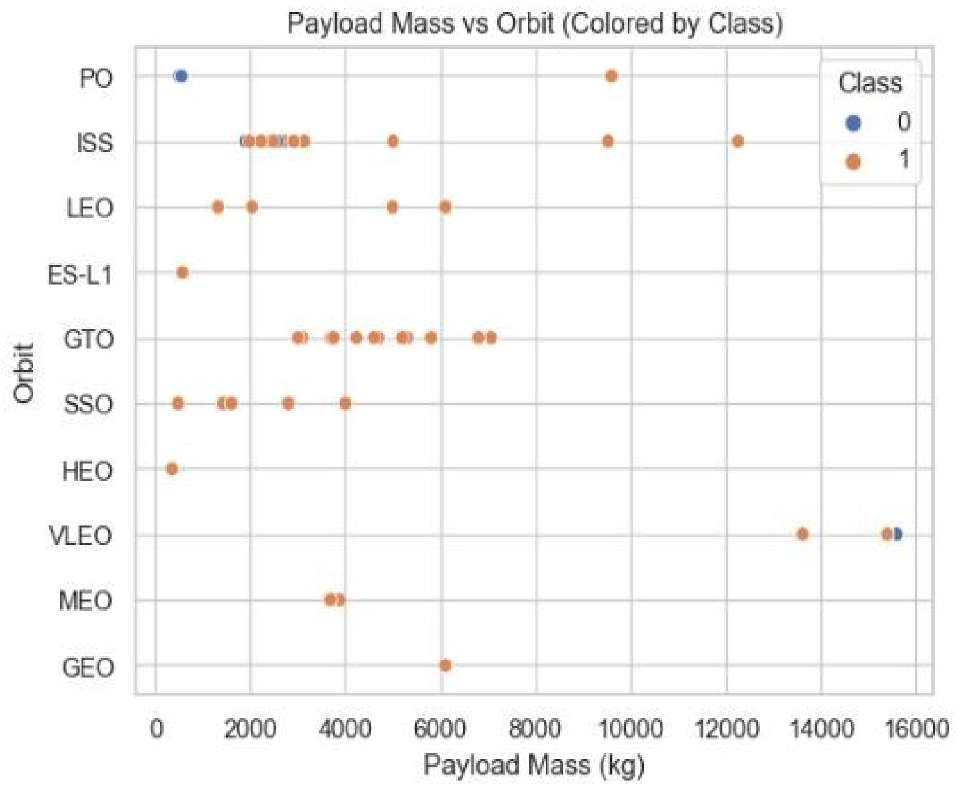
During our Exploratory data we visualized using various types of methods. This is to help us extract different informations from the data.

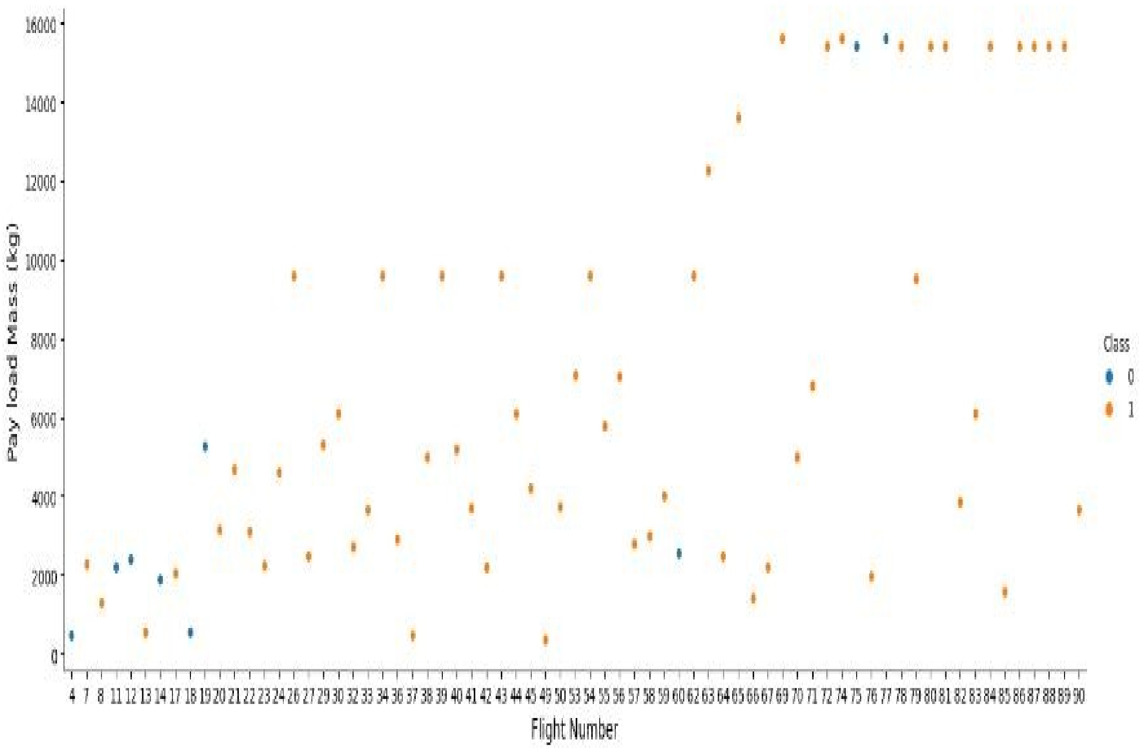
These includes:

- Scatter plots
- Categorical plot
- Bar plot
- Line plot



RESULTS

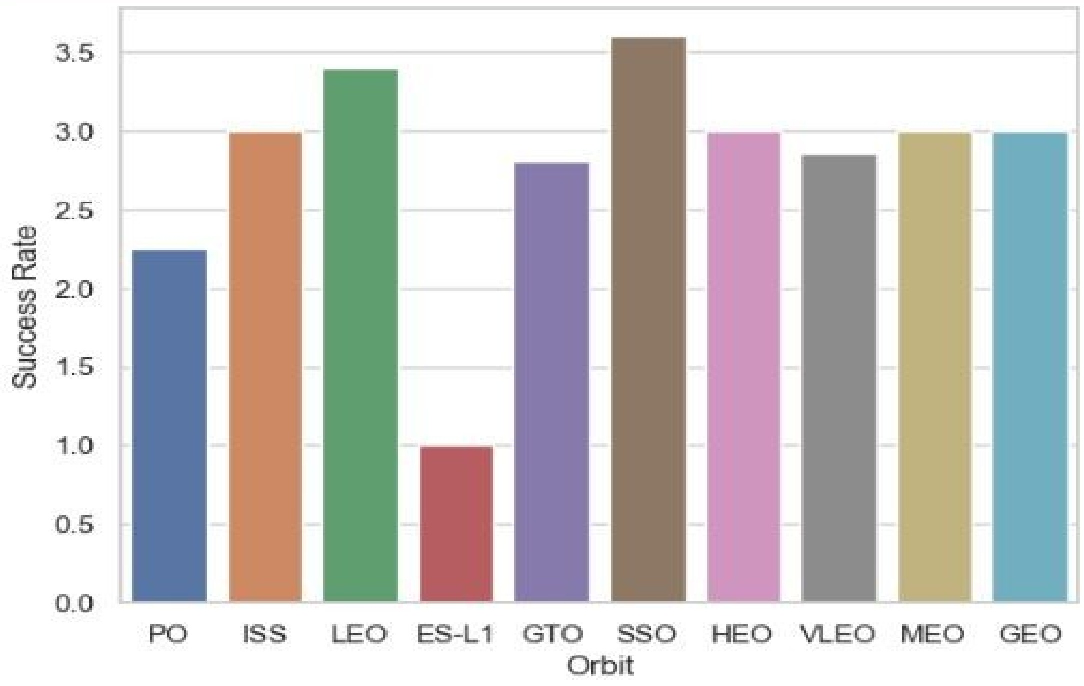




IBM Developer

SKILLS NETWORK

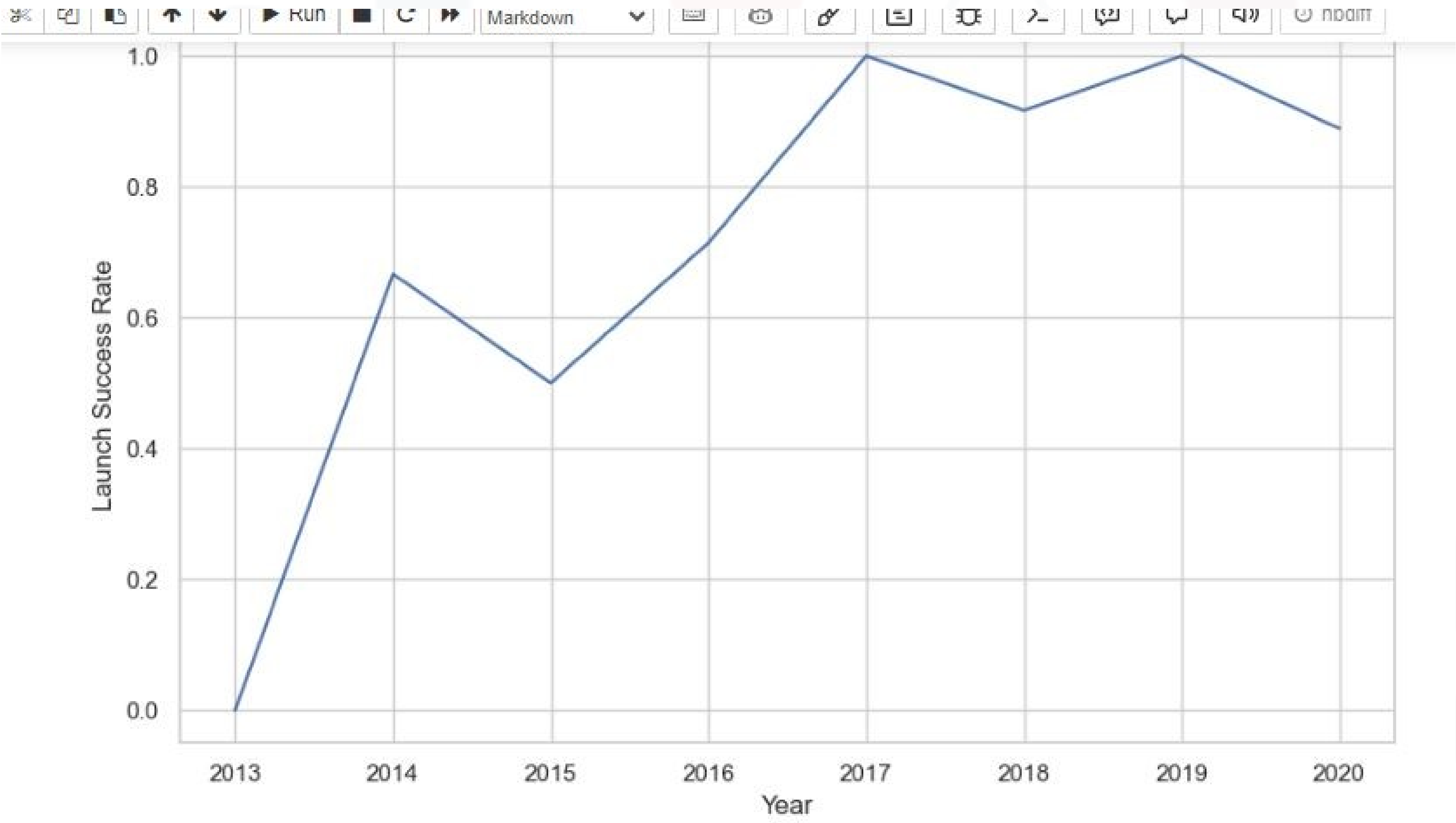
RESULTS



From our various plots we can deduce the following:

- Launch site CCAFS SLC40 has the highest number of flights
- Orbit SSO has the highest number of successful flights

Report



- The year with the Space X had the most successful rate between 2013 to 2020 are both 2017 and 2019

EDA with SQL

List the total number of successful and failure mission outcomes

In [27]:
```
%%sql

SELECT Mission_Outcome, COUNT(*) AS Total
FROM SPACEXTBL
GROUP BY Mission_Outcome;
```

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

Out[27]:

Mission_Outcome	Total
None	898
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

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 [23]:
```
%%sql

SELECT Booster_Version
FROM SPACEXTBL
WHERE Landing_Outcome = "Success (drone ship)" AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```

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

Out[23]:

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

Insights Drawn

Task 7

List the total number of successful and failure mission outcomes

In [27]:
```
%%sql

SELECT Mission_Outcome, COUNT(*) AS Total
FROM SPACEXTBL
GROUP BY Mission_Outcome;
```

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

Out[27]:

Mission_Outcome	Total
None	898
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

- Total number of successful mission outcomes = 98

Insights Drawn

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

Hint:Use min function

In [21]:
```
%%sql

SELECT MIN(Date)
FROM SPACEXTBL
WHERE Landing_Outcome = "Success (ground pad)";
```

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

Out[21]:

MIN(Date)
01/08/2018

In [22]:
```
%%sql

SELECT MIN(Date) AS First_Successful_Landing_Date
FROM SPACEXTBL
WHERE Landing_Outcome = 'Success (ground pad)';
```

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

Out[22]:

First_Successful_Landing_Date
01/08/2018

First successful landing date is on 01/08/2018.

Insights Drawn

TASK 1: Calculate the number of launches on each site

The data contains several Space X launch facilities: Cape Canaveral Space Launch Complex 40 **VAFB SLC 4E** , Vandenberg Air Force Base Space Launch Complex 4E **(SLC-4E)**, Kennedy Space Center Launch Complex 39A **KSC LC 39A** .The location of each Launch Is placed in the column `LaunchSite`

Next, let's see the number of launches for each site.

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

In [7]:
```
# Apply value_counts() on column LaunchSite
# Calculate the number of launches on each site using value_counts()
launch_counts = df['LaunchSite'].value_counts()

# Print the number of launches on each site
print(launch_counts)
```

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

Each launch aims to an dedicated orbit, and here are some common orbit types:

Number of launches on each site

- CCAFS SLC40 = 55
- KSC LC39A = 22
- VAFB SLC4E = 13

Insights Drawn

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

```
In [23]: %%sql

SELECT Booster_Version
FROM SPACEXTBL
WHERE Landing_Outcome = "Success (drone ship)" AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;

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

Out[23]:

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

List of Boosters which have success in drone ship against payload>4000 but <6000 are

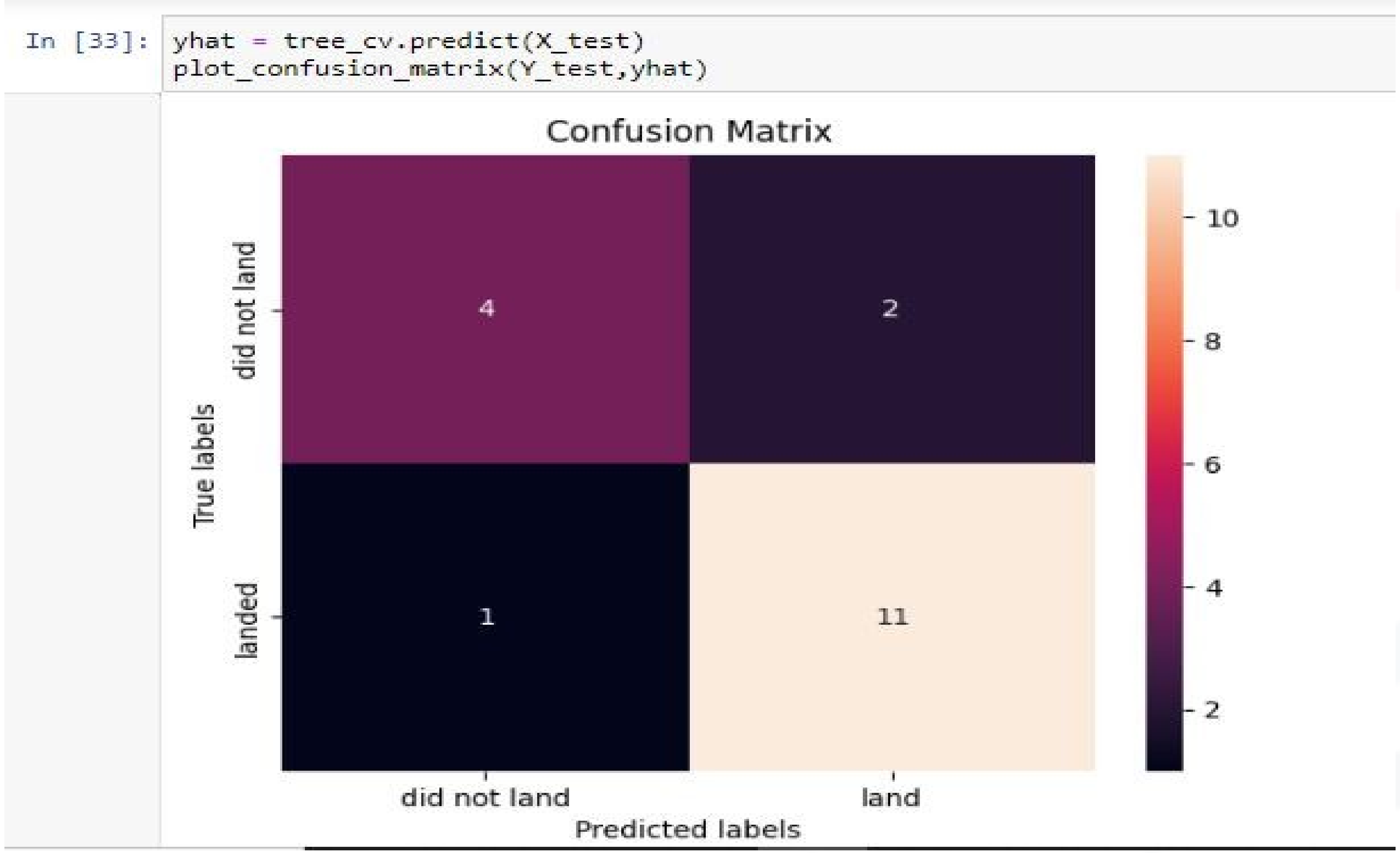
- F9 FT B1022
- F9 FT B1026
- F9 FT B1021.2

Insights Drawn

```
In [30]: tree = DecisionTreeClassifier()

         # Creating a GridSearchCV object with cross-validation of 10
         tree_cv = GridSearchCV(tree, parameters, cv=10)

         # Fitting the GridSearchCV object to find the best parameters
         tree_cv.fit(X_train, Y_train)  # X and y represent the training data and target variable

         # Accessing the best parameters
         best_params = tree_cv.best_params_

In [31]: print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
         print("accuracy :",tree_cv.best_score_)

         tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 12, 'max_features': 'sqrt', 'min_samples_leaf': 2,
         'min_samples_split': 5, 'splitter': 'best'}
         accuracy : 0.8767857142857143
```

The decision Tree model had the highest prediction accuracy score with a best score of 87.6785% or 0.876785

Insights Drawn



Confusion Matrix of the Decision Tree Model is shown here; comparing True labels and Predicted labels.

True Labels: 12 landed, 6 did not land

Predicted Labels: 13 landed, 5 did not land

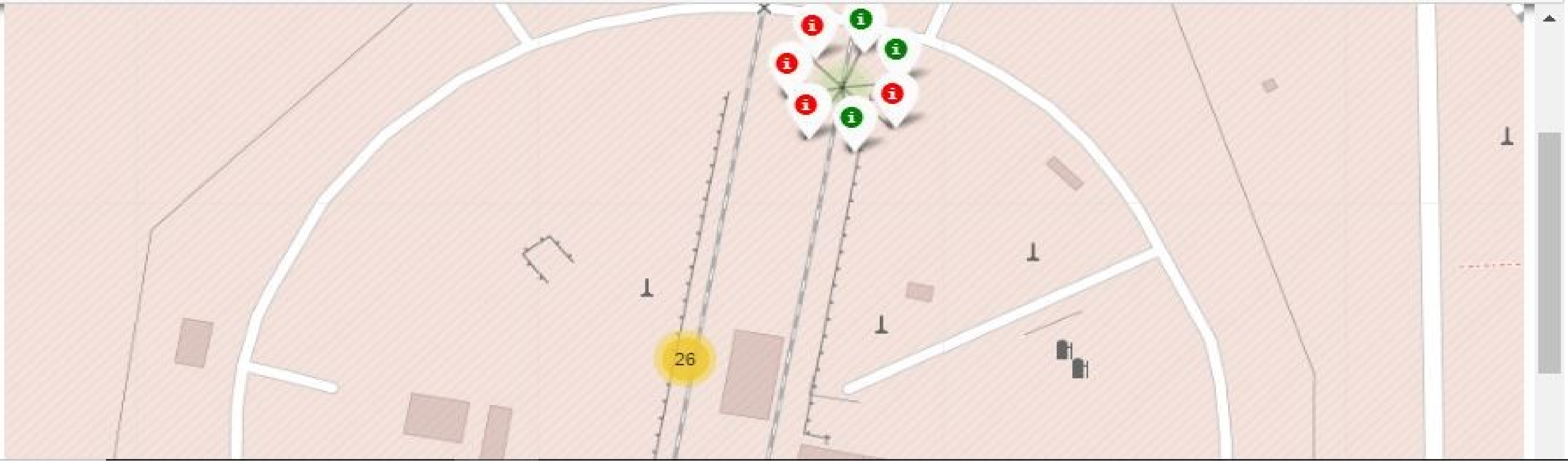
Interactive Maps with Folium results slides

```
In [13]: # Display the map
         site_map
```

Out[13]:

Interactive Maps with Folium results slides

```
# Add marker_cluster to the map
marker_cluster = MarkerCluster().add_to(site_map)

# Add markers to the marker_cluster for each record in the spacex_df dataframe
for index, record in spacex_df.iterrows():
    icon_color = record['marker_color']
    marker = folium.Marker(location=[record['Lat'], record['Long']], icon=folium.Icon(color='white', icon_color=icon_color))
    marker_cluster.add_child(marker)

# Display the map
site_map
```



Conclusion

We can conclude that:

- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase in 2013 till 2020, with peaks at 2017 and 2019.
- Orbits SSO had the most success rate followed by LEO.
- The Decision tree classifier is the best machine learning algorithm for this task.