Executive Summary

OBJECTIVE: ANALYZE SPACEX DATA, VISUALIZE , PREDICT OUTCOMES

IN THIS PROJECT, WE PERFORMED A COMPREHENSIVE ANALYSIS OF **SPACEX LAUNCH DATA** TO UNCOVER INSIGHTS ABOUT LAUNCH SUCCESS PATTERNS, PAYLOAD CORRELATIONS, AND BOOSTER PERFORMANCE. THE PROJECT IS DESIGNED TO DEMONSTRATE **DATA COLLECTION, EXPLORATORY DATA ANALYSIS (EDA), PREDICTIVE MODELING, AND INTERACTIVE VISUALIZATION TECHNIQUES** USING PYTHON, SQL, FOLIUM, AND DASH.)

KEY POINTS:

DATA COLLECTION

DALA ANALYSIS

DATA VISUALIZATION

DATA PREDICTION

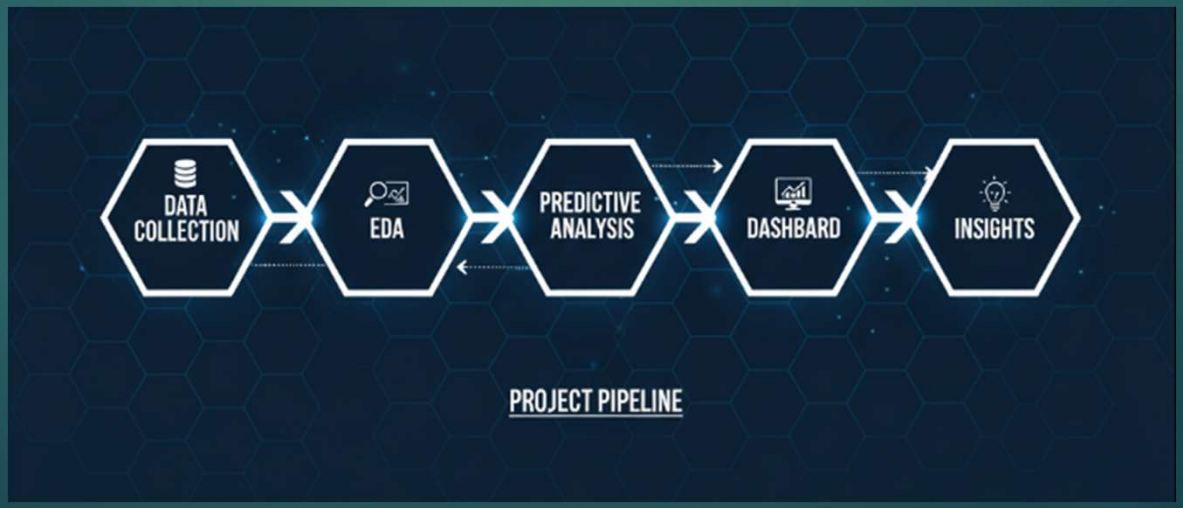
Introduction – SpaceX Launch Data Analysis

- **SpaceX** is a private aerospace company that launches rockets for satellites, cargo, and crew missions.
- Mission success is critical due to high costs and safety concerns.
- Understanding factors affecting launch success helps in planning and risk mitigation.
- This project analyzes historical SpaceX launch data to answer:
- Which launch sites are most successful?
- How does payload affect launch outcomes?
- Which booster versions perform better?

- **Scope**: Analyzing launch records, payload, boosters, and landing outcomes from historical data.

Project Objective

- Analyze SpaceX launch history to find patterns and correlations.
- Build predictive models to classify mission success.
- Develop interactive dashboards for visual exploration of launch data.
- Provide insights that could support decision-making in launch planning.



DATA COLLECTION

- The first step of this project is obtaining the right data and I have taken data by using two very effective methods.
- 1. **API Data Collection** – This methos is very common as we use API's
- to collect the data we needed.
- Here is the image of the input code and the resultant data I got from there .

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

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

        Check the content of the response

In [12]: print(response.content)
```

```
In [77]: launch_df.head()
```

Out[77]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block
0	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN
1	2	2007-03-21	Falcon 1	NaN	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN
2	4	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN
3	5	2009-07-13	Falcon 1	200.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0

DATA COLLECTION

- 2. **Webscraping** – This is also one of the most common tech to
- collect the data, in which we directly scrapes the data from the websites of the data provider legally.
- I have used request and beautiful soul to scrape the data from the websites and presented it in the form of data frame.

In [26]:
```
response = requests.get(static_url, headers=headers)
print(response.status_code)
```

```
200
```

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

In [27]:
```
soup = BeautifulSoup(response.text, 'html.parser')
```

Out[74]:

	Flight No.	Date	Time	Version Booster	Launch Site	Payload	Payload mass	Orbit	Customer	Launch outcome	Booster landing
0	1	4 June 2010	18:45	F9 v1.07B0003.18	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure
1	2	8 December 2010	15:43	F9 v1.07B0004.18	CCAFS	Dragon	0	LEO	NASA	Success	Failure
2	3	22 May 2012	07:44	F9 v1.07B0005.18	CCAFS	Dragon	525 kg	LEO	NASA	Success	No attempt\n
3	4	8 October 2012	00:35	F9 v1.07B0006.18	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success	No attempt
4	5	1 March 2013	15:10	F9 v1.07B0007.18	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success	No attempt\n

DATA WRANGLING

- Standardized column names for consistency.
- Converted categorical variables into numerical or encoded features (e.g., One-Hot Encoding).
- Created new features:
- class column: 1 = success, 0 = failure.
- marker_color for map visualization.
- Prepared data for both **visualization** and **machine learning models**.

```
In [28]: df.head(8)
```

Out[28]:

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

Exploratory Data Analysis (EDA)

- EDA – It is a method to analyze the data that we are working on and find the important insights from it which can be useful in future analysis.
- Like in this I have used sql to gather the important aspects of the data and used them wherever they are required.

```
In [13]: %sql select distinct(Launch_Site) from SPACEXTABLE
 * sqlite:///my_data1.db
Done.
```

Out[13]:

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

```
In [41]: %sql SELECT substr(Date, 6, 2) AS Month, Landing_Outcome, Booster_Version,  La
 * sqlite:///my_data1.db
Done.
```

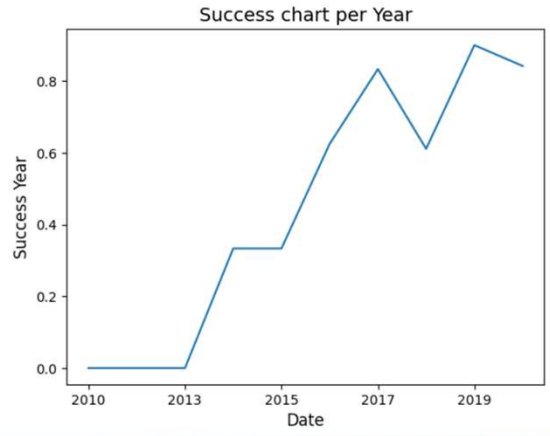
Out[41]:

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

EDA for Visualization

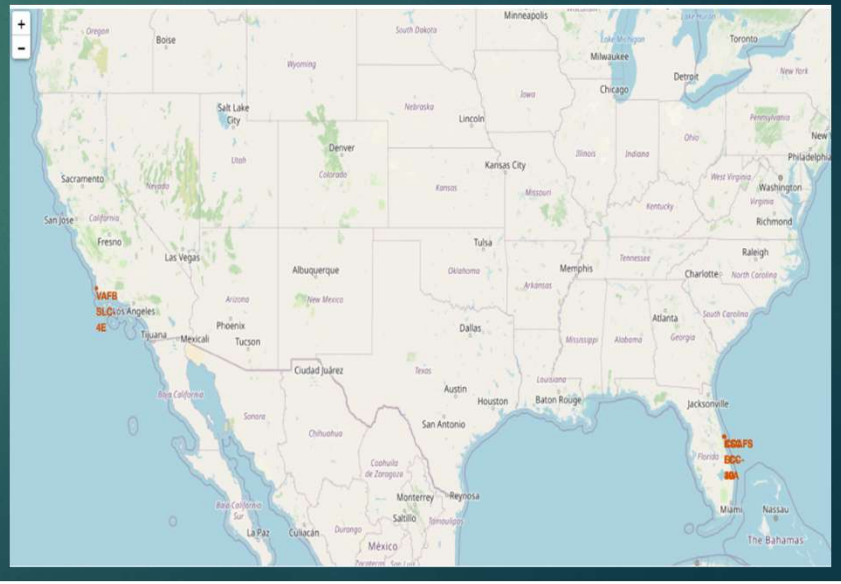
- We have gained many important and powerful insights from our data and now we have to show them in the visual format to find the relationship between the , so that non technical person can also understand the data.
- I have done the same and plot many types of graphs which actually shows the correct info and relationship.
- Like here is the line graph for success rate per year.
- This representation helps to know the incine in the rate of success and what are the peak points.
- There are more plots of different kinds can check on the notebook provided on git.

```
In [12]: # Plot a line chart with x axis to be the extracted year and y axis to be the success rate
         success_by_year = df.groupby('Date')['Class'].mean()
         success_by_year.plot(kind='line')
         plt.title("Success chart per Year", fontsize=14)
         plt.xlabel("Date", fontsize=12)
         plt.ylabel("Success Year", fontsize=12)
         plt.show()
```



Interactive Map Visualization

- **FOLIUM** – In this we are using folium which allows us to plot the geospatial data using the coordinated of the location on the map very easily, supported by many platforms , simple and interactive .
- Marked all launch sites using Folium.
- Color-coded markers indicate success (green) or failure (red).
- Interactive cluster to explore multiple launches per site.
- Generate interactive maps like this on which we can easily mark the location which will show up as a popup that can be easily seen and located.



INTERACTIVE DASHBOARD

- In this section we have used Cloud IDE platform to create a virtual DASH app to represent the dynamic information represented in the form of charts on Dashboard.
- Dropdown to select launch sites.
- Pie chart updates dynamically with selection.
- Range slider for payload mass filtering.
- Scatter chart shows payload vs. success with color-coded boosters.
- Users can explore correlations interactively.

Predictive Analysis

- **Models used: in this we have used many types of ml models like**
- Logistic Regression
- Support Vector Machine
- Decision Tree
- K-Nearest Neighbors
- Hyperparameter tuning via GridSearchCV.

- These all models will help to make machine learning models that will allow us to predict the unknown values and check for the accuracy rate so that these values can be used in further analysis.

Predictive Analysis

- We split the collected data into training and testing data

```
X_train, X_test, Y_train, Y_test
```

```
X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size=0.2, random_state=2)
```

we can see we only have 18 test samples.

```
Y_test.shape
```

```
(18,)
```

We make **linear regression** model of of the data.

```
parameters ={'C':[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}
```

```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()
logreg_cv = GridSearchCV(lr, parameters, cv=10)
logreg_cv.fit(X_train, Y_train)
```

Predictive Analysis

Similarly we have created more types of models or the ways to train the model like **SVM** ,Grid Search.

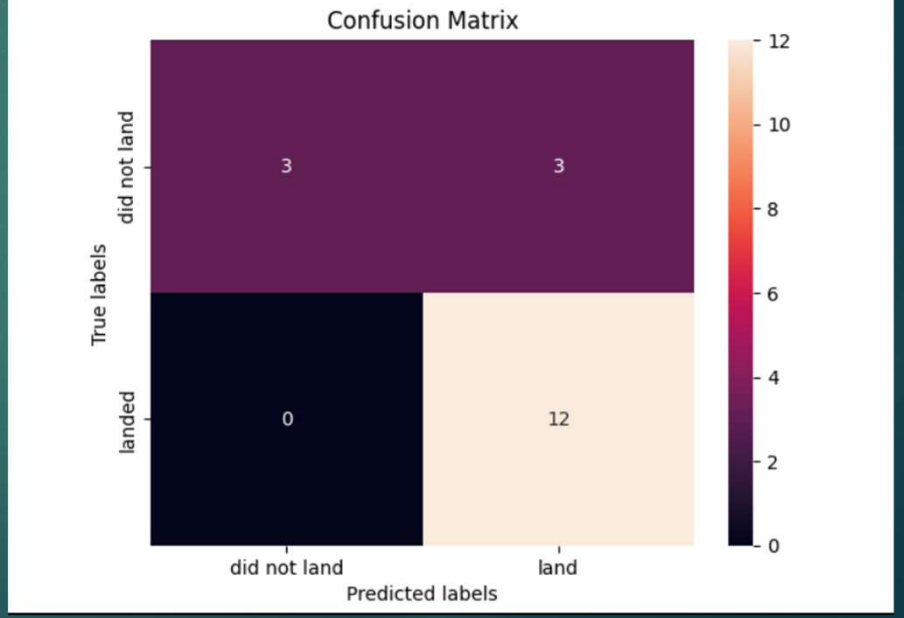
These predictions can be evaluated by using heat maps as confusion matrix.

In [33]:
```
test_accuracy = svm_cv.score(X_test, Y_test)
print("Test set accuracy:", test_accuracy)
```

```
Test set accuracy: 0.8333333333333334
```

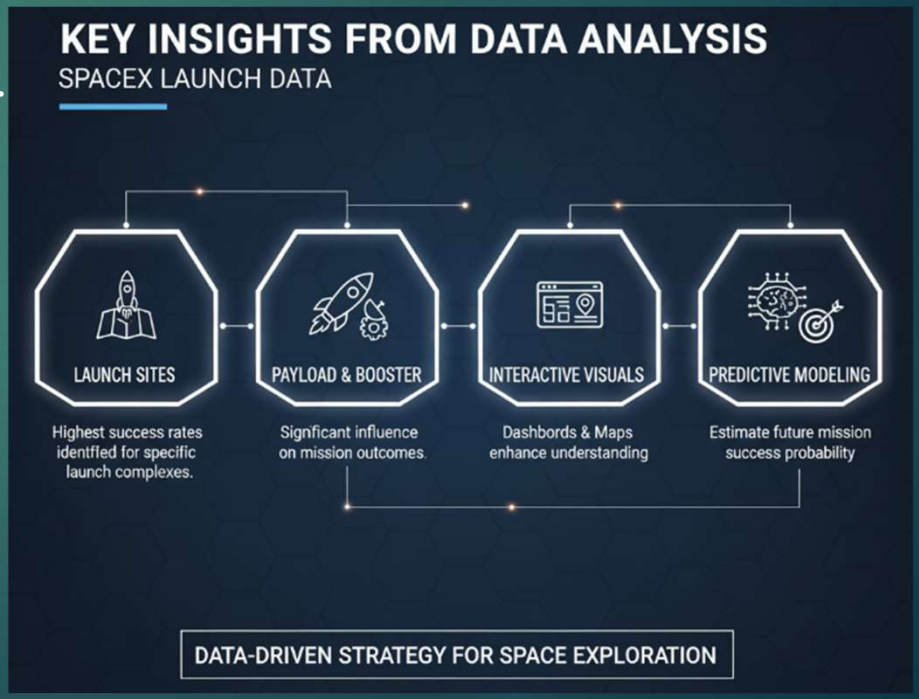
We can plot the confusion matrix

In [34]:
```
yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



CONCLUSION

- **Identified many insights such as .**
- Launch sites with highest success rate identified.
- Payload and booster type significantly influence mission outcomes.
- Interactive dashboards and maps enhance understanding and analysis.
- Predictive models can help estimate mission success probability for future launches.



APPENDIX

- Additional SQL queries used.
- Raw dataset sample.
- Additional charts or exploratory plots not in main slides.

Submitted by:- Chetan Pawar