BEYOND THE STRATOSPHERE:

SPACE X'S DATA-INFUSED TRAJECTORY TO THE STARS

Grayson Keever - 10 December 2023

AGENDA

Executive Summary

Introduction

Methodology & Results

Conclusion

Appendix

EXECUTIVE SUMMARY

- Methodologies
    - Data Collection (API & Web Scraping), Data Wrangling, Exploratory Analysis with SQL and Visualization, Interactive Analytics with Folium, and Machine Learning Prediction
- Results
    - Data Analysis
    - Interactive Analytics
    - Predictive Analytics

INTRODUCTION

SpaceX's Falcon 9 launches at $62 million, far lower than competitors due to first-stage reusability. Our goal is to use Data Science tools to predict first-stage landings, crucial for estimating launch costs. This data will allow us at Space Y to effectively compete with SpaceX.

MISSION CONTROL – MISSION OBJECTIVE

What are the determining factors of a successful landing?

METHODOLOGY & RESULTS

METHODOLOGY

- Data Collection
  - ➢ Space X API
  - ➢ Web Scraping Wikipedia
- Data Wrangling
- Exploratory Data Analysis (EDA)
  - ➢ SQL
  - ➢ Data Visualization
- Interactive Visual Analytics
  - ➢ Folium
  - ➢ Dash
- Predictive Analysis
  - ➢ Machine Learning Prediction

DATA COLLECTION

Space X API

Web Scraping Public Data

DATA COLLECTION

- Utilized the SpaceX API through GET requests for data collection purposes.
- Employed the **.json()** function to decode response content into JSON format, later converting it into a pandas dataframe via **.json_normalize()**.
- Conducted data cleansing procedures, addressing missing values by identifying and filling gaps within the dataset.
- Executed web scraping techniques using **BeautifulSoup** on Wikipedia to extract Falcon 9 launch records.
- Extracted launch records as an HTML table, parsing and transforming the table into a pandas dataframe for subsequent analysis.

SPACE X API

- Utilizing Python libraries Pandas and Numpy, we were able to connect to Space X's historical launch data by calling its API.
  - https://api.spacexdata.com/v4/rockets/
  - https://api.spacexdata.com/v4/launchpads/
  - https://api.spacexdata.com/v4/payloads/
  - https://api.spacexdata.com/v4/cores/
- After normalizing the data using Panda's **.json_normalize()** function and constructing the dataset into a dictionary, we were able to visualize launch data in a table.

Out[59]:

	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

In [60]:

```
data.shape
```

Out[60]: (94, 17)

LINK TO SPACE X API CODE

https://github.com/GraysonKeever/SpaceX-Project/blob/main/SpaceX%20API_Data%20Collection.ipynb

WEB SCRAPING OF WIKIPEDIA

- To accrue more data, we Web Scraped Wikipedia using Python’s **BeautifulSoup** for public data on Space X’s launch data.
- After converting the data in **HTML**, we created a data frame by parsing the the launch HTML tables resulting in 121 extracted rows of data.

In [56]:

```python
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

[59]:

```python
df = pd.DataFrame(launch_dict)


for key, val in launch_dict.items():
    print(f"{key}: #: {len(val)}")
```

```
Flight No.: #: 121
Launch site: #: 121
Payload: #: 121
Payload mass: #: 121
Orbit: #: 121
Customer: #: 121
Launch outcome: #: 121
Version Booster: #: 121
Booster landing: #: 121
Date: #: 121
Time: #: 121
```

LINK TO WEB SCRAPING CODE

https://github.com/GraysonKeever/SpaceX-Project/blob/main/SpaceX%20-%20Web%20Scraping%20Data%20Collection%20(Wiki).ipynb

DATA WRANGLING

Exploratory Data Analysis

Determined Training Labels

DATA WRANGLING

- In the exploratory data analysis phase, we utilized various methods to understand our dataset. This involved employing the **.shape** function to identify the rows and columns within our dataset and utilizing **.types** to determine the data types. We further analyzed the data by computing the launch count at each site, the frequency of different orbits, and the mission outcomes associated with these orbits. To enhance clarity, we simplified the analyzed data by creating an outcome label for the outcome column

CALCULATIONS

of Launches on Each Site

```
In [6]:  # Apply value_counts() on column LaunchSite
         df['LaunchSite'].value_counts()

Out[6]:  CCAFS SLC 40    55
         KSC LC 39A      22
         VAFB SLC 4E     13
         Name: LaunchSite, dtype: int64
```

of Occurrence(s) of Each Orbit

```
In [7]:  # Apply value_counts on Orbit column
         df['Orbit'].value_counts()

Out[7]:  GTO      27
         ISS      21
         VLEO     14
         PO        9
         LEO       7
         SSO       5
         MEO       3
         ES-L1     1
         HEO       1
         SO        1
         GEO       1
         Name: Orbit, dtype: int64
```

of Occurrence(s) of Mission Outcome

```
In [9]:  # landing_outcomes = values on Outcome column
         landing_outcomes = df['Outcome'].value_counts()
         landing_outcomes

Out[9]:  True ASDS      41
         None None      19
         True RTLS      14
         False ASDS      6
         True Ocean      5
         False Ocean     2
         None ASDS       2
         False RTLS      1
         Name: Outcome, dtype: int64
```

LINK TO DATA WRANGLING CODE

https://github.com/GraysonKeever/SpaceX-Project/blob/main/SpaceX%20-%20Data%20Wrangling.ipynb

DATA VISUALIZATION

Exploratory Data Analysis with SQL

Data Visualization

EDA WITH SQL

- Using **SQL**, we were able to identify the unique launch sites, numerical data of payload mass, successful boosters, total number of successes and failures of mission outcomes, and boosters that have carried maximum payload mass.

EDA WITH SQL

Distinct Launch Sites

```
In [63]: %sql select DISTINCT(Launch_Site) from SPACEXTBL;
 * sqlite:///my_data1.db
Done.
```

Out[63]:

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

Successful Boosters (4000 Kg < payload mass < 6000 Kg)

```
In [69]: %sql select Booster_Version from SPACEXTBL
 * sqlite:///my_data1.db
Done.
```

Out[69]:

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

Dominant Boosters

```
In [72]: %sql select Booster_Version
 * sqlite:///my_data1.db
Done.
```

Out[72]:

boosterversion
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

LINK TO SQL EDA CODE

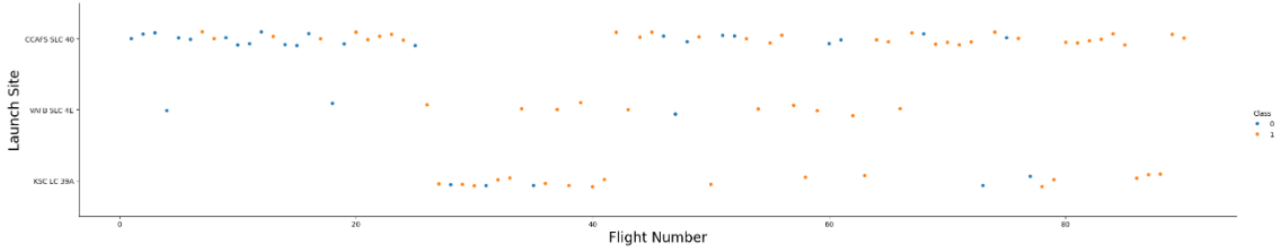
https://github.com/GraysonKeever/SpaceX-Project/blob/main/SpaceX%20-%20SQL.ipynb

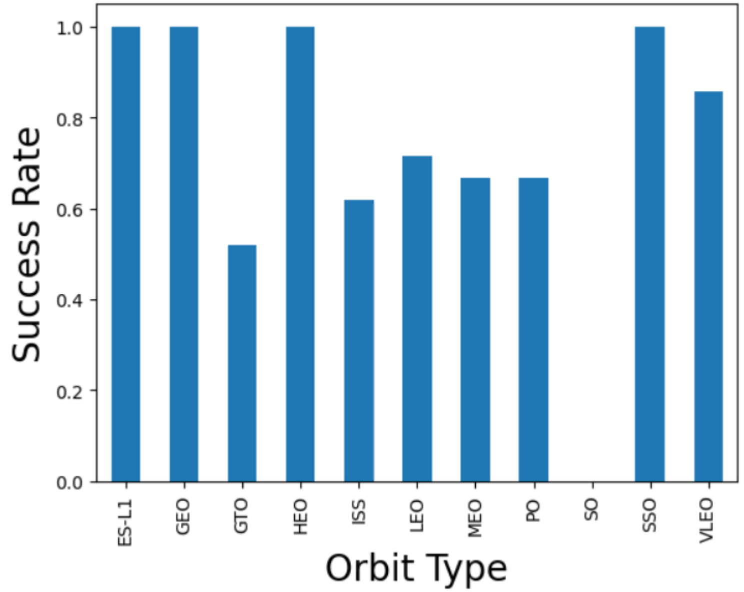
DATA VISUALIZATION

- We conducted exploratory data analysis by visualizing correlations between flight number and launch site, payload and launch site, success rates across various orbit types, flight number relative to orbit type, and the annual trend in launch success.
- We observed a clear correlation between payload mass and orbit achievement. Higher payload masses necessitate more successful rocket performance to reach higher orbits. Moreover, SpaceX has displayed increasing efficiency, leading to higher success rates in their rocket launches.

In [5]:

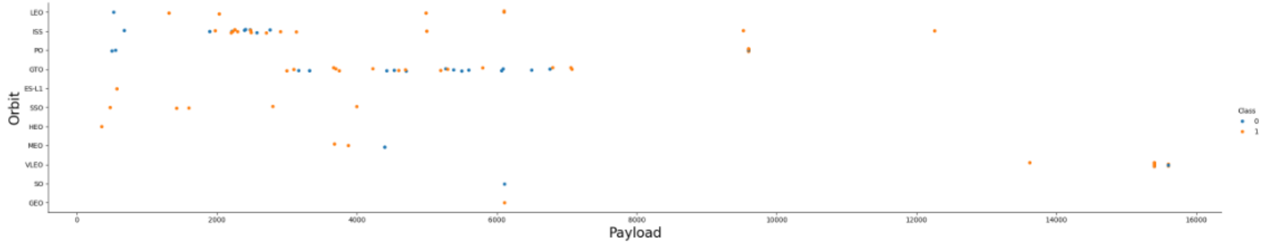
```python
### TASK 1: Visualize the relationship between Flight Number and Launch Site
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```





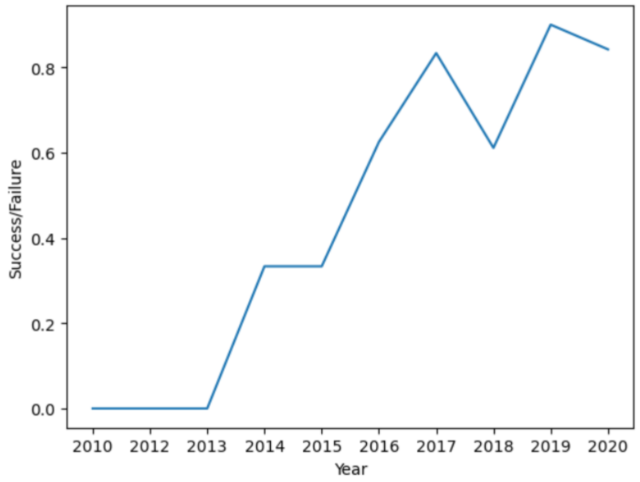
In [11]:

```python
### TASK  5: Visualize the relationship between Payload and Orbit type
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Payload",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



In [19]:

```python
plt.plot(average_by_year["Year"],average_by_year["Class"])
plt.xlabel("Year")
plt.ylabel("Success/Failure")
plt.show()
```



LINK TO DATA VISUALIZATION CODE

https://github.com/GraysonKeever/SpaceX-Project/blob/main/SpaceX%20-%20Data%20Visualization.ipynb

INTERACTIVE MAP

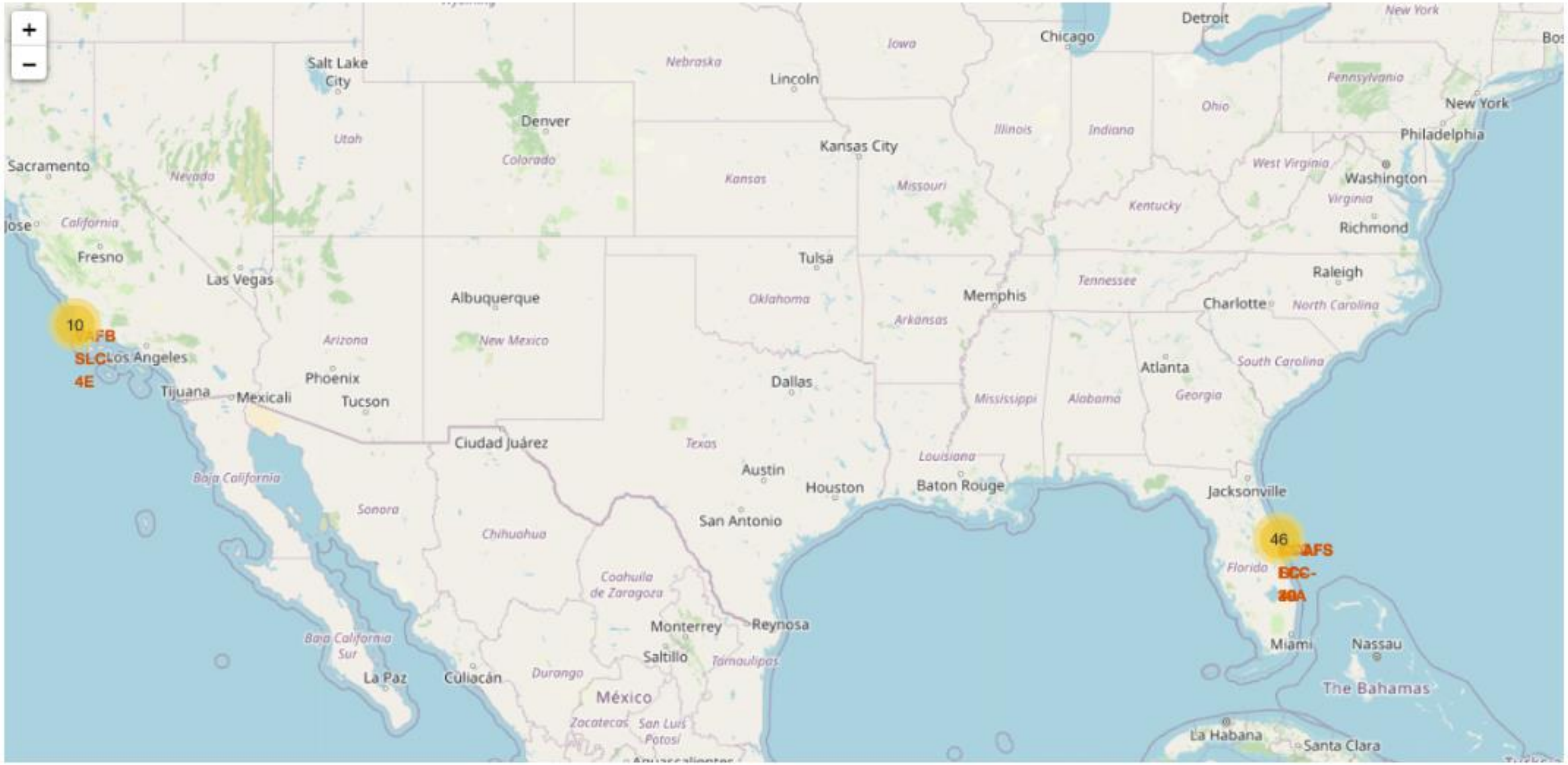
Folium

Plotly

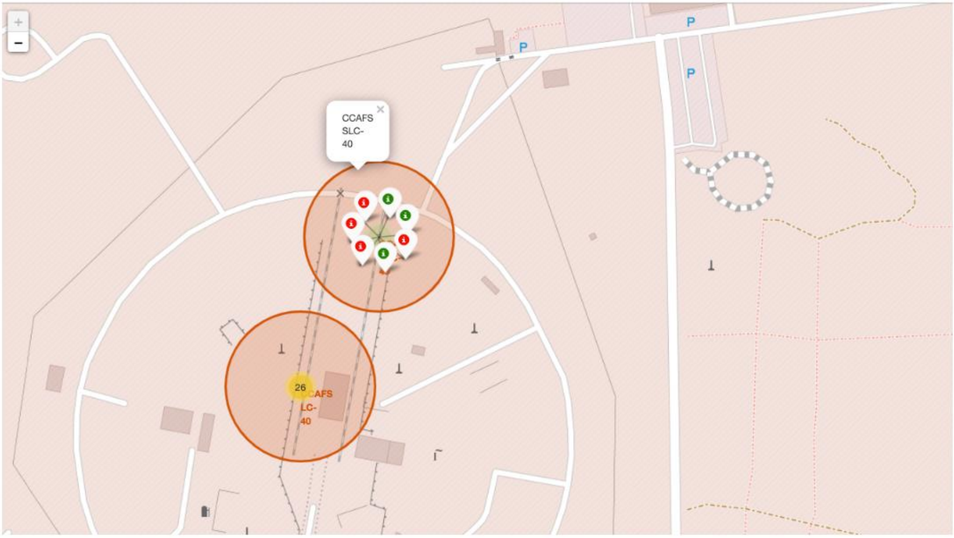
MAPPING LAUNCH SITE LOCATIONS

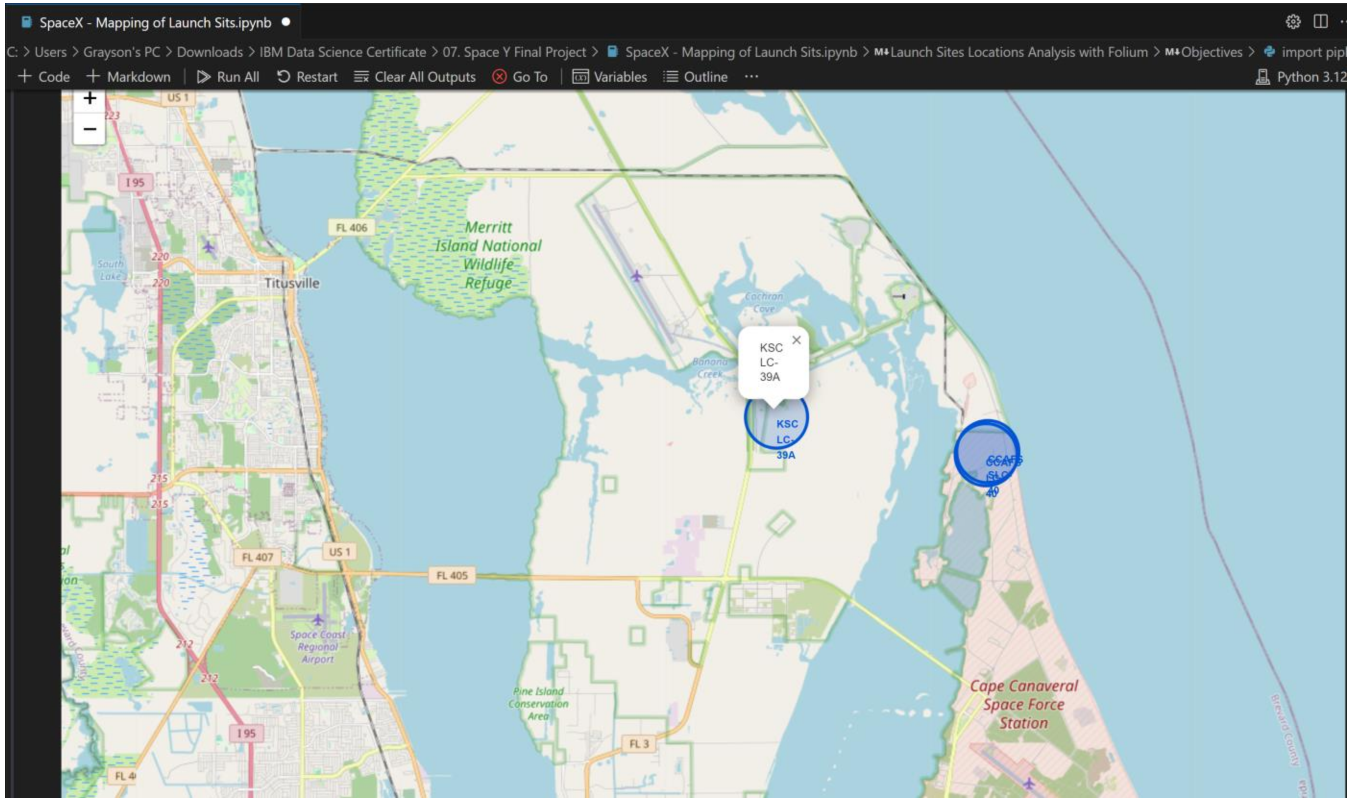
- We used Folium to create an interactive visuals on Space X’s launch sites and success rates.
- We found that Space X launch sites are on the Southwest and Southeast coasts of the U.S. It had 36 more launches in Cape Canaveral, FL than any other location in North America. The sites are not near railways or highways and keep at least 78 Km of distance from cities.

LAUNCH RESULTS



MAPPING LAUNCH SITE LOCATIONS





LINK TO MAP VISUALIZATION CODE

https://github.com/GraysonKeever/SpaceX-Project/blob/main/SpaceX%20-%20Mapping%20of%20Launch%20Sites.ipynb

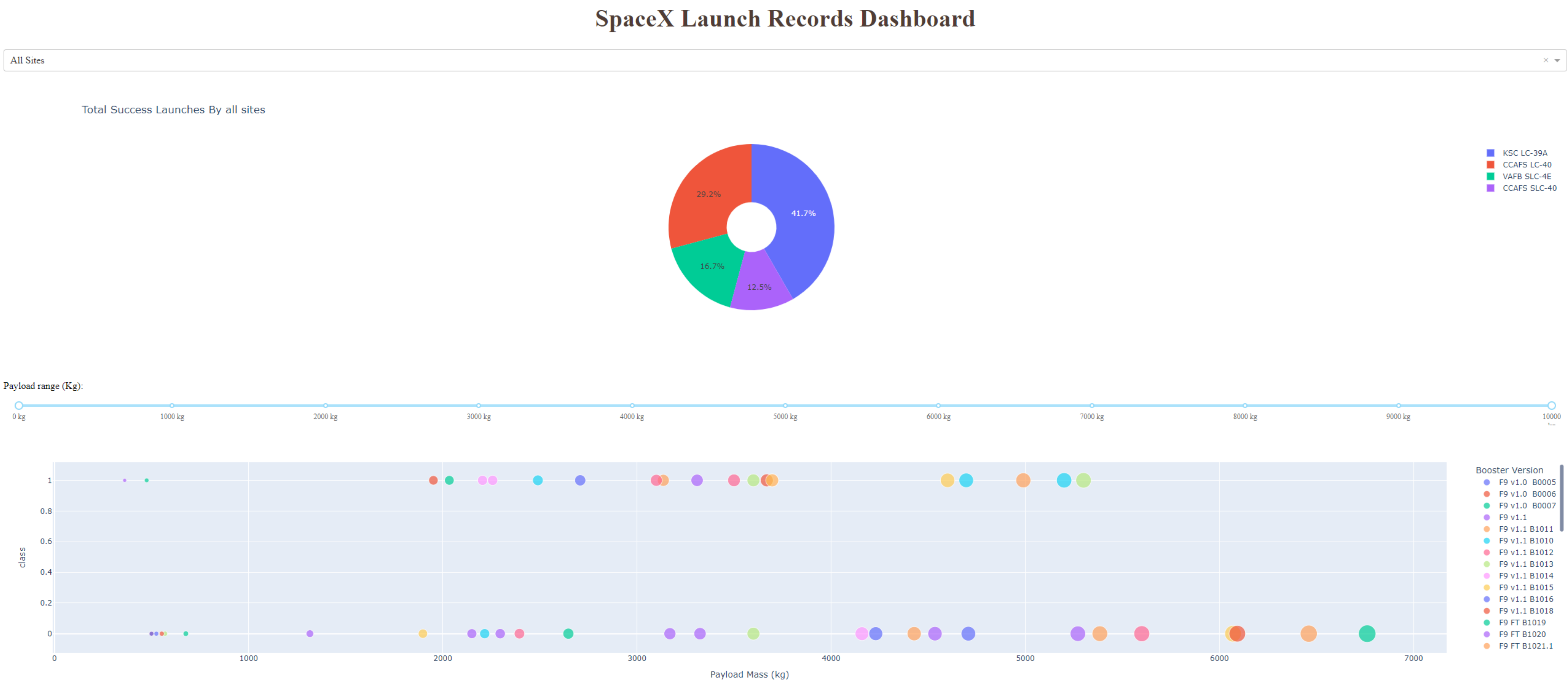
BUILDING A DASHBOARD

Plotly Dash

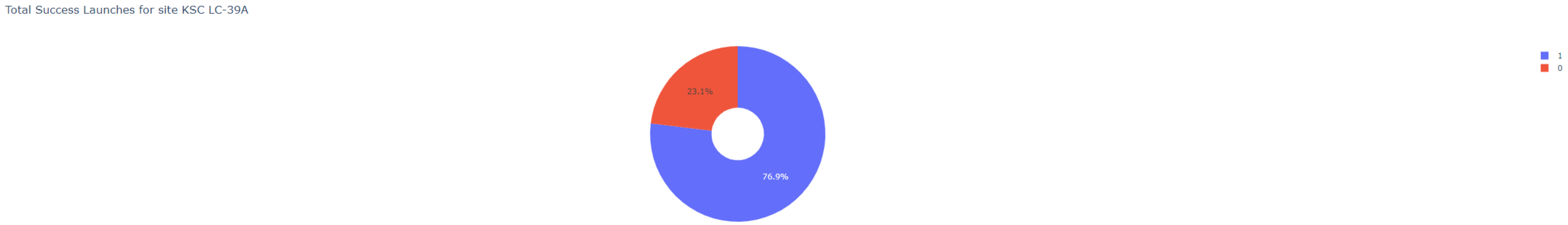
INTERACTIVE DASHBOARDS

- To assist our stakeholders in understanding the relationships between outcomes to payload mass and number of launches per site, we built an interactive dashboard using Plotly Dash.
- Most launches have taken place in Florida, and there is a higher chance of success with payload mass under 6500 Kg. The KSC LC-39A is the dominant rocket with 41.7% of successful launches, and 76.9% of launches being successful.

INTERACTIVE DASHBOARD

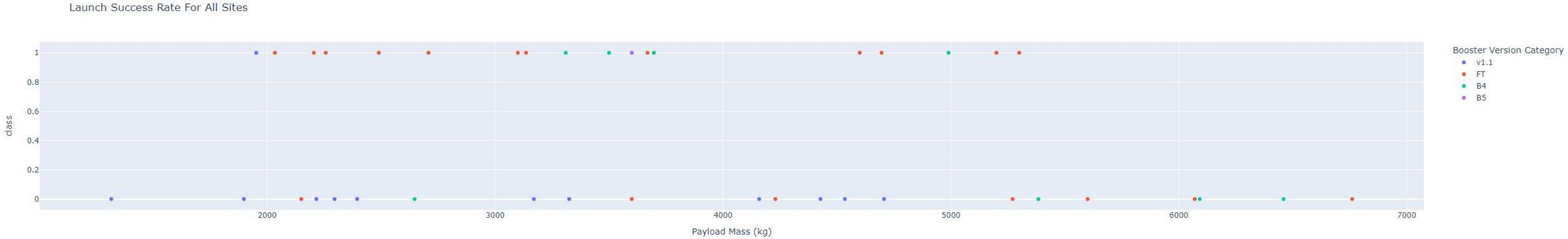


KSC LC-39A



Payload range (Kg):

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



LINK TO PLOTLY DASH CODE

https://github.com/GraysonKeever/SpaceX-Project/blob/main/spacex_dash_app.py

PREDICTIVE ANALYSIS

Machine Learning Models (Logistic Regression, SVM, KNN, Decision Tree Classification)

- Leading Hyperparameters
- Confusion Matrices
- Accuracy Score based on Model

PREDICTIVE ANALYSIS

- After standardizing the data and splitting it into training and testing data, we were able to find the best hyperparameters for Support Vector Machines, Classification Trees, K-Nearest Neighbors and Logistic Regression.
- We found that the best model for our data is a Decision Tree which had a score of 87.3% of accurately predicting a successful first stage landing.

Logistic Regression

```
In [10]: print("tuned hyperparameters :(best parameters) ",logreg_cv.best_params_)
         print("accuracy :", logreg_cv.best_score_)

tuned hyperparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8464285714285713
```

SVM

```
In [14]: print("tuned hyperparameters :(best parameters) ",svm_cv.best_params_)
         print("accuracy :",svm_cv.best_score_)

tuned hyperparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8482142857142856
```

Decision Tree Classifier

```
In [18]: print("tuned hyperparameters :(best parameters) ",tree_cv.best_params_)
         print("accuracy :",tree_cv.best_score_)

tuned hyperparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2,
'min_samples_split': 5, 'splitter': 'random'}
accuracy : 0.8732142857142856
```

KNN

```
In [22]: print("tuned hyperparameters :(best parameters) ",knn_cv.best_params_)
         print("accuracy :",knn_cv.best_score_)

tuned hyperparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
accuracy : 0.8482142857142858
```

BEST PARAMETERS

Logistic Regression

```
In [11]: print('Accuracy on test data is: {:.3f}'.format(logreg_cv.score(X_test, Y_test)))

Accuracy on test data is: 0.833
```

SVM

```
In [15]: print('Accuracy on test data is: {:.3f}'.format(svm_cv.score(X_test, Y_test)))

Accuracy on test data is: 0.833
```

Decision Tree Classifier

```
In [19]: print('Accuracy on test data is: {:.3f}'.format(tree_cv.score(X_test, Y_test)))

Accuracy on test data is: 0.833
```

KNN

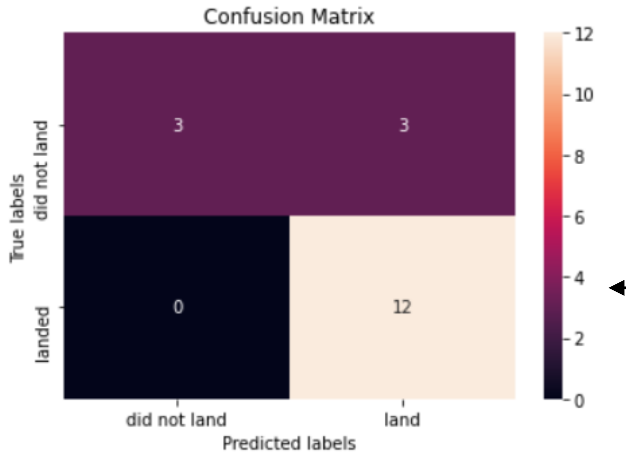
```
In [23]: print('Accuracy on test data is: {:.3f}'.format(knn_cv.score(X_test, Y_test)))

Accuracy on test data is: 0.833
```

ACCURACY OF TEST DATA

CONFUSION MATRICES

In [12]:
```
yhat_lr = logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test, yhat_lr)
```



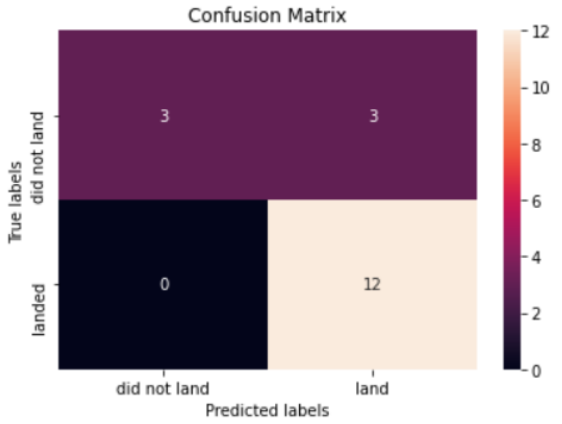
Logistic Regression

Accuracy: 0.833

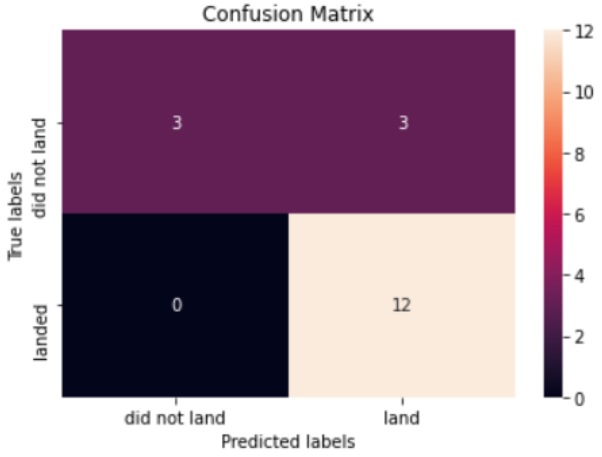
Decision Tree

Accuracy: 0.873

In [20]:
```
yhat_tree = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test, yhat_tree)
```



In [16]:
```
yhat_svm = svm_cv.predict(X_test)
plot_confusion_matrix(Y_test, yhat_svm)
```



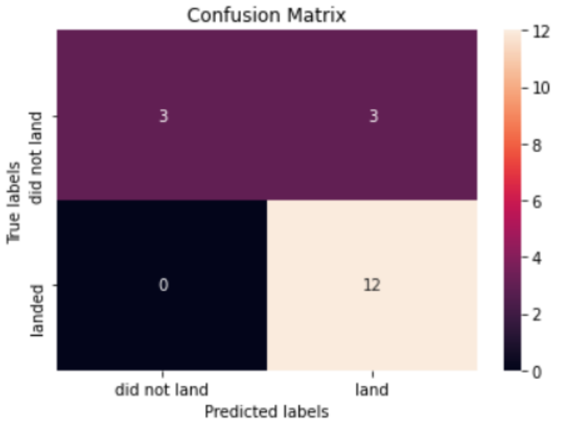
SVM

Accuracy: 0.833

K-Nearest Neighbors

Accuracy: 0.873

In [24]:
```
yhat_knn = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test, yhat_knn)
```



BEST PERFORMING MODEL: DECISION TREE (0.873)

In [25]:
```
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'}
```

LINK TO MACHINE LEARNING CODE

https://github.com/GraysonKeever/SpaceX-Project/blob/main/SpaceX%20-%20Machine%20Learning%20Data%20Prediction.ipynb

CONCLUSION

Starting in 2013, a steady rise in launch success rates was observed. Higher success rates corresponded with shorter distances to orbit. Notably, orbits ES-L1, GEO, HEO, SSO, and VLEO exhibited exceptional success rates. The KSC LC-39A reigns the dominant rocket with the highest recorded number of successful launches among all sites. The Decision Tree classifier emerged as the most effective machine learning algorithm for analyzing predictive models.

APPENDIX

- Grayson Keever GitHub Repository: SpaceX Project
- https://github.com/GraysonKeever/SpaceX-Project

BEYOND THE STRATOSPHERE

Presentation by Grayson Keever

10 December 2023