Winning Space Race with Data Science

Kitty Zhao
Dec 8, 2022

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result

Introduction

- Project background and context
  - 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. The goal here is to predict if the Falcon 9 first stage will land successfully
- Problems you want to find answers
  - What factors determine if the rocket will land successfully?
  - Various parameters that determine the success rate of a successful landing.
  - What operating conditions needs to be in place to ensure a successful landing.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe 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

Here's the flow of data collection

- Firstly we requested the data to the SpaceX API.
- Then we decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().
- Next we cleaned the data, checked for missing values and fill in missing values where necessary.
- We also performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
- The objective was to extract the launch records as HTML table and convert it to a pandas dataframe for future analysis.

Data Collection – SpaceX API

- https://github.com/zyim333/Chap10_repo/blob/main/Complete%20the%20Data%20Collection%20API%20Lab.ipynb

1. Request data using API →

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

```
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 [8]: print(response.content)
```

2. Convert to data frame →

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

```
# Use json_normalize meethod to convert the json result into a dataframe
response.json()
data=pd.json_normalize(response.json())
```

3. Data cleaning →

```
In [ ]: # Calculate the mean value of PayloadMass column
PayloadMass = pd.DataFrame(data_falcon9['PayloadMass'].values.tolist()).mean(1)
print(PayloadMass)

# Replace the np.nan values with its mean value

rows = data_falcon9['PayloadMass'].values.tolist()[0]

df_rows = pd.DataFrame(rows)
df_rows = df_rows.replace(np.nan, PayloadMass)

data_falcon9['PayloadMass'][0] = df_rows.values
data_falcon9
```

Data Collection - Scraping

- https://github.com/zyim333/Chap10_repo/blob/main/Data%20Collection%20with%20Web%20Scraping%20lab.ipynb

1. Perform HTTP Get method to request Falcon 9 rocket launch data

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

```
# use requests.get() method with the provided static_url
# assign the response to a object
html_data = requests.get(static_url)
html_data.status_code
```

```
200
```

2. Create a BeautifulSoup object from the HTML response

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

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html_data.text, 'html.parser')
```

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

```
# Use soup.title attribute
soup.title
```

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

3. Extract all column names from the HTML table header

```
# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names


column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names

element = soup.find_all('th')
for row in range(len(element)):
    try:
        name = extract_column_from_header(element[row])
        if (name is not None and len(name) > 0):
            column_names.append(name)
    except:
        pass
```

4. Create a data frame by parsing the launch HTML tables
5. Export it to a CSV

Data Wrangling

- https://github.com/zyim333/Chap10_repo/blob/main/EDA%20-%20Data%20Wrangling.ipynb

1. Import Libraries and Define Auxiliary Functions

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

```
df.isnull().sum()/df.count()*100
```

Identify which columns are numerical and categorical:

```
df.dtypes
```

2. Calculate the number of launches on each site

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

3. Calculate the number and occurrence of each orbit

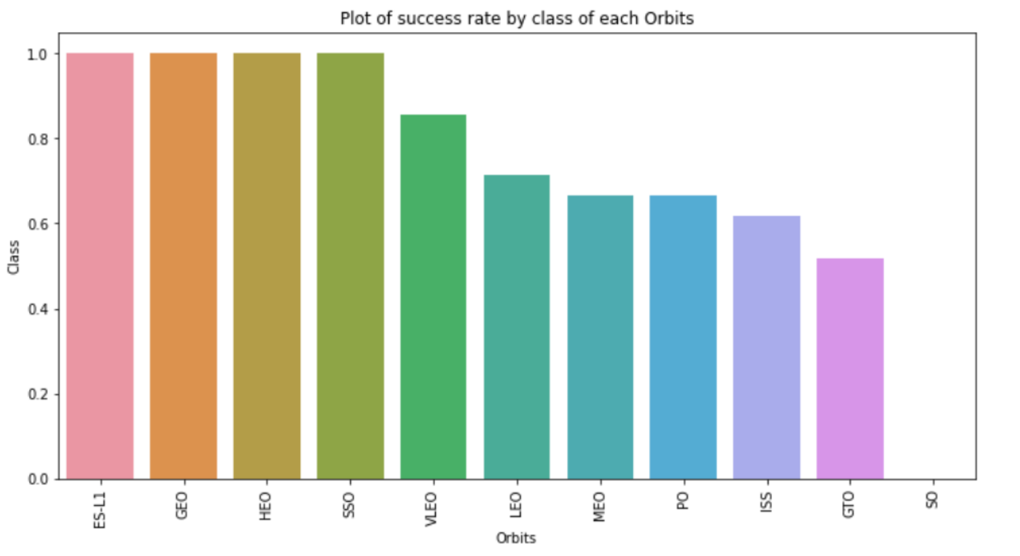
```
# Apply value_counts on Orbit column
df['Orbit'].value_counts()
```

4. Create a landing outcome label from Outcome column

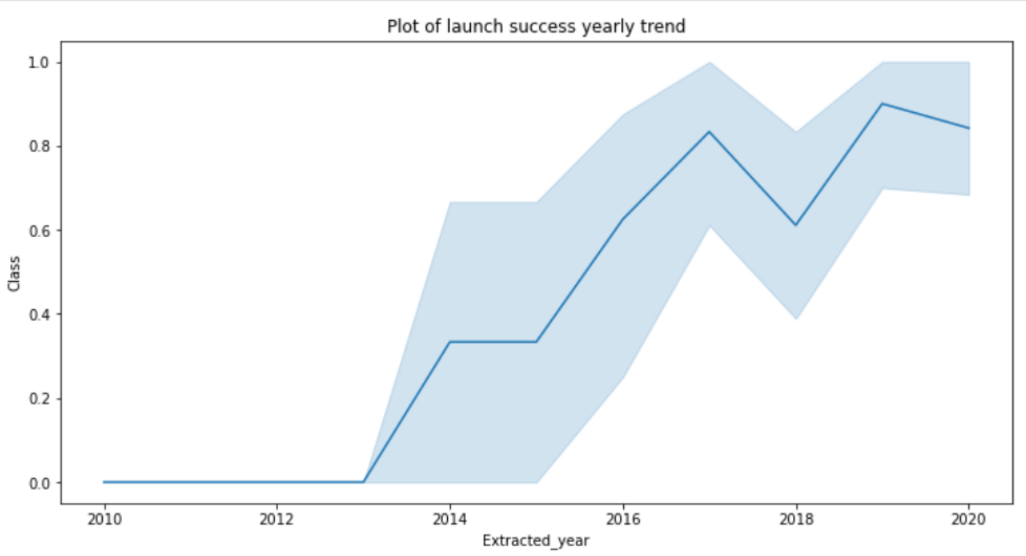
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
#landing_class = [x for x in bad_outcomes if df['Outcome'][x] ]

landing_class = []

for key, value in df['Outcome'].items():
    if value in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

5. Export it to a CSV

EDA with Data Visualization



A bar chart to tell the success rate of each orbit



A line chart to tell sucess rate since 2013 kept increasing till 2020

- https://github.com/zyim333/Chap10_repo/blob/main/EDA%20with%20Visualization%20(week2-part2).ipynb

EDA with SQL

- We loaded the SpaceX dataset into a PostgreSQL database without leaving the jupyter notebook.
- We applied EDA with SQL to get insight from the data. We wrote queries to find out data

- https://github.com/zyim333/Chap10_repo/blob/main/EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

- Marked all launch sites on a map.
- Added map objects such as markers, circles, lines to mark the success/failed launches for each site on the map
- Calculated the distances between a launch site to its proximities

- https://github.com/zyim333/Chap10_repo/blob/main/Interactive%20Visual%20Analytics%20with%20Folium%20lab%20(wk3).ipynb

Build a Dashboard with Plotly Dash

- Summarize what plots/graphs and interactions you have added to a dashboard
- Explain why you added those plots and interactions
- Add the GitHub URL of your completed Plotly Dash lab, as an external reference and peer-review purpose

Predictive Analysis (Classification)

- We first created a column for the class, standardized the data, splited into training data and test data

```
# target data as numpy array
Y = data['Class'].to_numpy()
Y
```

```
# students get this
transform = preprocessing.StandardScaler()
```

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

- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- Calculated accuracy of each model, improved the model using feature engineering and algorithm tuning.
- By finding best Hyperparameter for SVM, Classification Trees and Logistic Regression, we found the best performing classification model
- https://github.com/zyim333/Chap10_repo/blob/main/Machine%20Learning%20Prediction%20(week4).ipynb

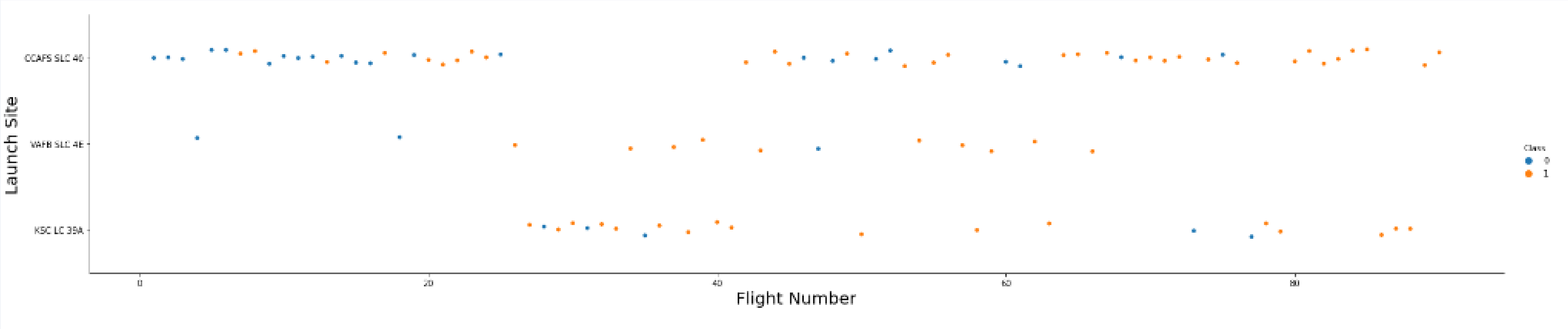
Results

- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

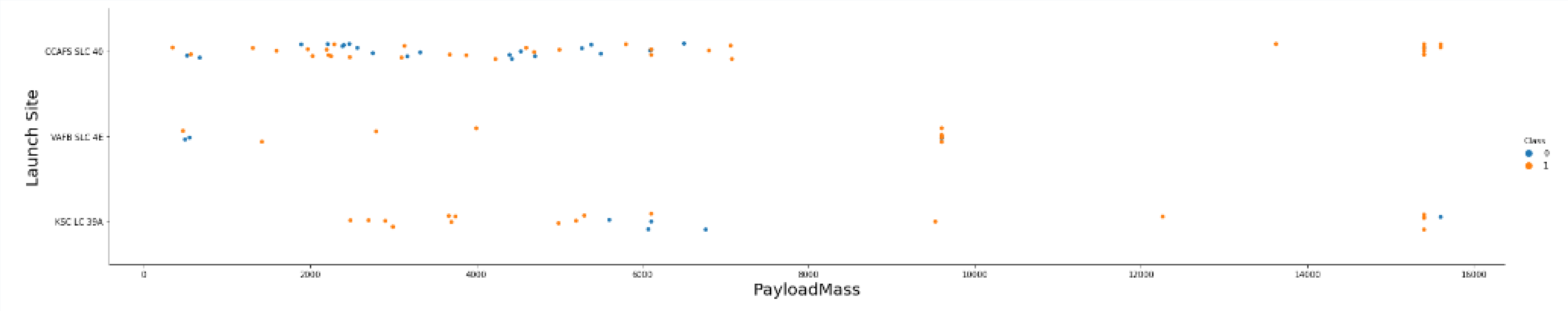
Insights drawn from EDA

Flight Number vs. Launch Site



- The larger the flight amount at a launch site, the greater the success rate at a launch site.

Payload vs. Launch Site

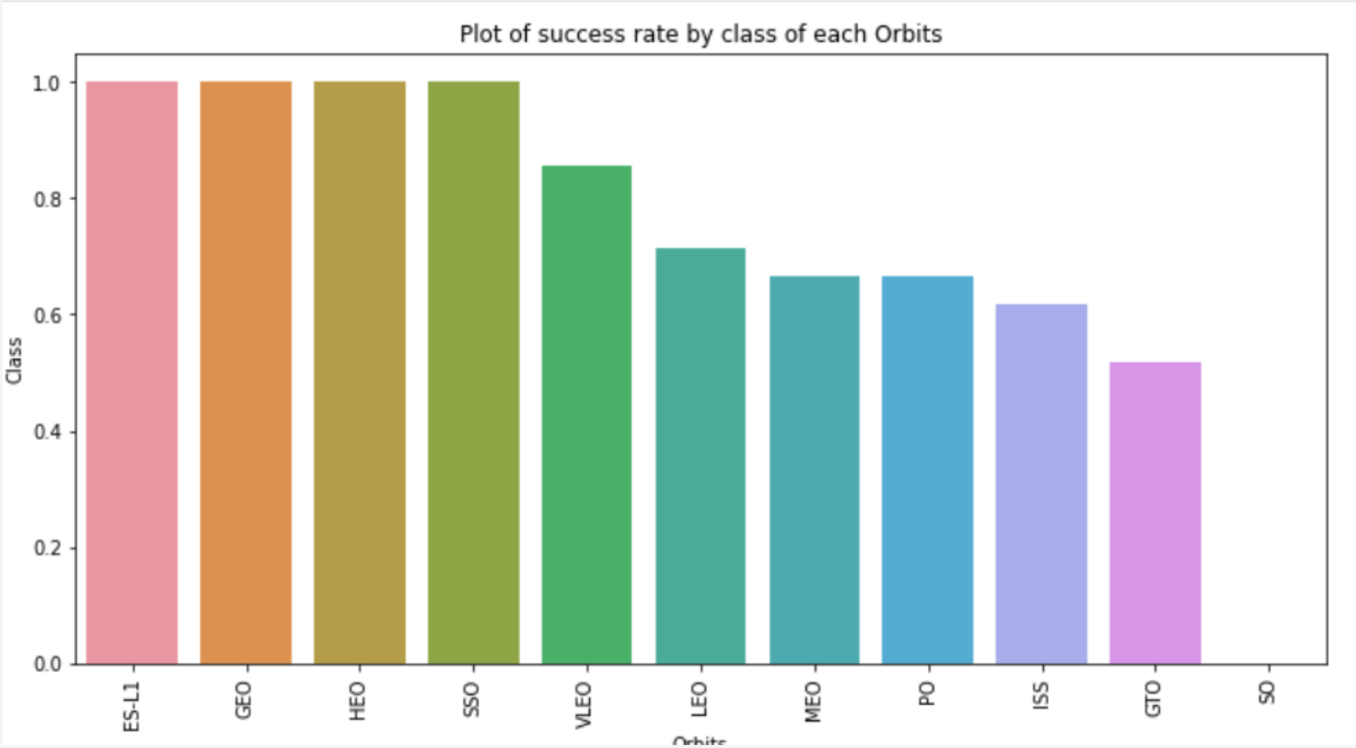


- VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

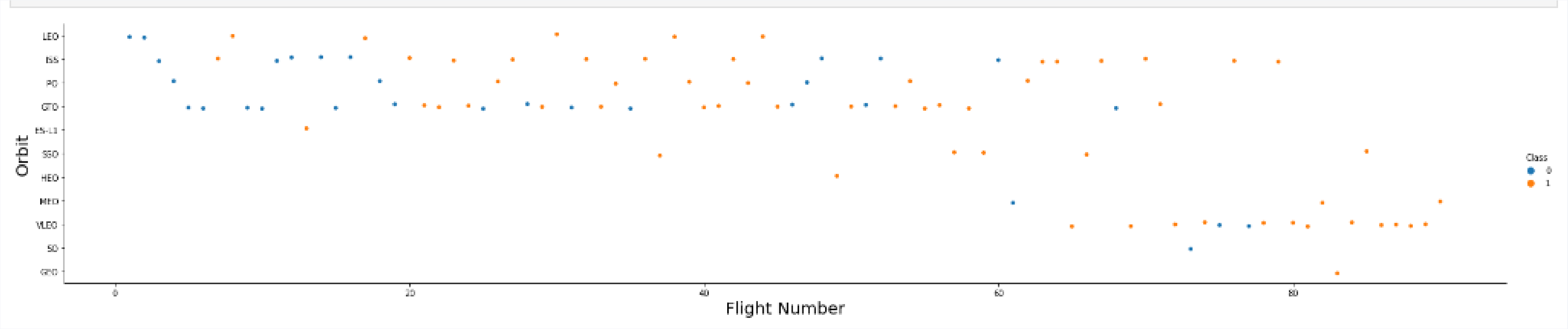
- The greater the payload mass for launch site CCAFS SLC 40, the higher success rate for launch

Success Rate vs. Orbit Type

- ES-L1, GEO, HEO, SSO, VLEO had the most success rate.

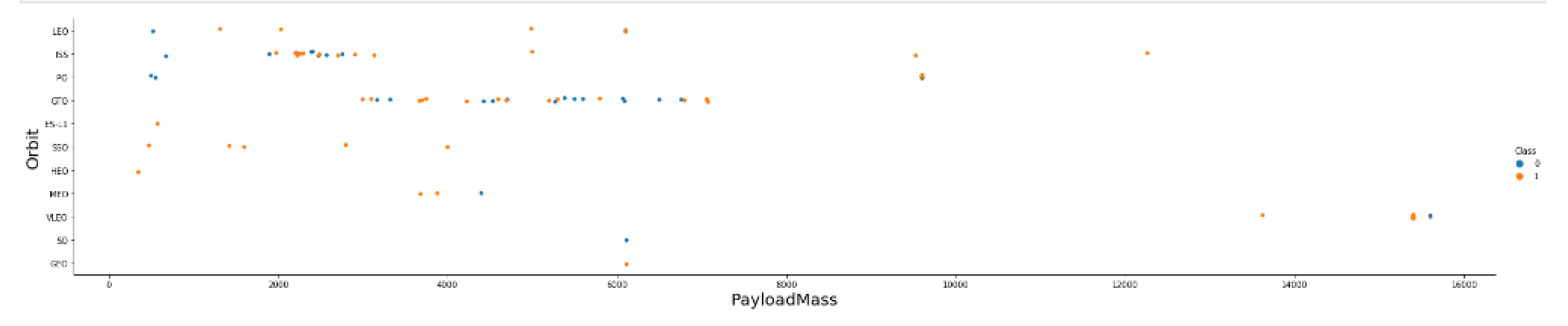


Flight Number vs. Orbit Type



- 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.

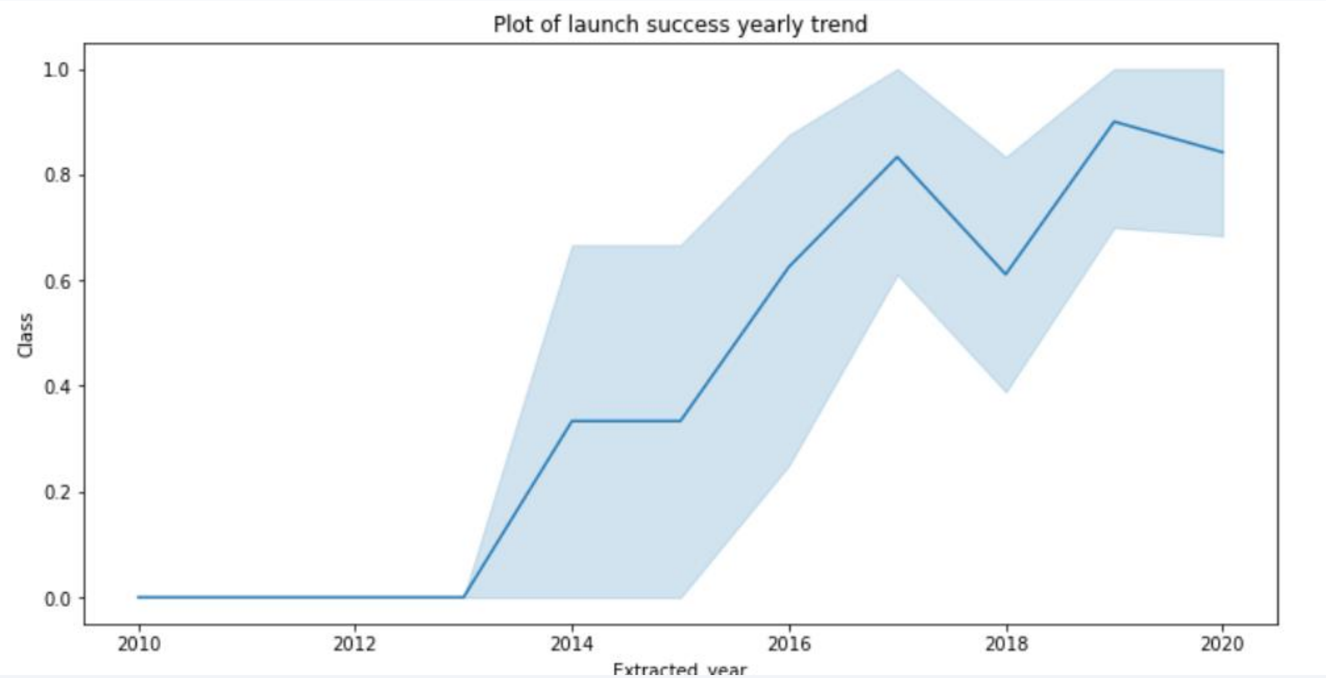
Payload vs. Orbit Type



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

Launch Success Yearly Trend

- The success rate since 2013 kept increasing till 2020



All Launch Site Names

- Used Select DISTINCT to show unique values

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

```
task_1 = '''
        SELECT DISTINCT LaunchSite
        FROM SpaceX
'''
create_pandas_df(task_1, database=conn)
```

	launchsite
0	KSC LC-39A
1	CCAFS LC-40
2	CCAFS SLC-40
3	VAFB SLC-4E

Launch Site Names Begin with 'CCA'

- Use' Select… where' query to select site names begin with 'CCA'

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

```
task_2 = '''
        SELECT *
        FROM SpaceX
        WHERE LaunchSite LIKE 'CCA%'
        LIMIT 5
        '''
create_pandas_df(task_2, database=conn)
```

	date	time	boosterversion	launchsite	payload	payloadmasskg	orbit	customer	missionoutcome	landingoutcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
3	2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- Used ‘SELECT SUM()… AS.. Where…’ to calculate the total payload mass

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

```
task_3 = '''
        SELECT SUM(PayloadMassKG) AS Total_PayloadMass
        FROM SpaceX
        WHERE Customer LIKE 'NASA (CRS)'
        '''
create_pandas_df(task_3, database=conn)
```

	total_payloadmass
0	45596

Average Payload Mass by F9 v1.1

- Used 'SELECT AVG()… AS…where… = …' to calculate the average payload mass by F9 v1 1

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

```
task_4 = '''
        SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
        FROM SpaceX
        WHERE BoosterVersion = 'F9 v1.1'
        '''
create_pandas_df(task_4, database=conn)
```

	avg_payloadmass
0	2928.4

First Successful Ground Landing Date

- Used 'SELECT MIN(Date) as …. Where… like..' to find the first success landing date

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

Hint:Use min function

```
task_5 = '''
        SELECT MIN(Date) AS FirstSuccessfull_landing_date
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Success (ground pad)'
        '''
create_pandas_df(task_5, database=conn)
```

	firstsuccessfull_landing_date
0	2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- Used ‘Select…where… = and >, <’ to select success landing with a payload range

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

```
task_6 = '''
        SELECT BoosterVersion
        FROM SpaceX
        WHERE LandingOutcome = 'Success (drone ship)'
            AND PayloadMassKG > 4000
            AND PayloadMassKG < 6000
        '''
create_pandas_df(task_6, database=conn)
```

	boosterversion
0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

- Used ‘select count… where… like ‘success%’/’failure%’’ to calculate the total outcomes

List the total number of successful and failure mission outcomes

```
task_7a = '''
        SELECT COUNT(MissionOutcome) AS SuccessOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Success%'
        '''

task_7b = '''
        SELECT COUNT(MissionOutcome) AS FailureOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Failure%'
        '''
print('The total number of successful mission outcome is:')
display(create_pandas_df(task_7a, database=conn))
print()
print('The total number of failed mission outcome is:')
create_pandas_df(task_7b, database=conn)
```

```
The total number of successful mission outcome is:
```

	successoutcome
0	100

```
The total number of failed mission outcome is:
```

	failureoutcome
0	1

Boosters Carried Maximum Payload

- Used “select…where… = (SELECT MAX(…))” to list the boosters carried maximum payload

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

```
task_8 = '''
        SELECT BoosterVersion, PayloadMassKG
        FROM SpaceX
        WHERE PayloadMassKG = (
                                SELECT MAX(PayloadMassKG)
                                FROM SpaceX
                                )
        ORDER BY BoosterVersion
        '''
create_pandas_df(task_8, database=conn)
```

	boosterversion	payloadmasskg
0	F9 B5 B1048.4	15600
1	F9 B5 B1048.5	15600
2	F9 B5 B1049.4	15600
3	F9 B5 B1049.5	15600
4	F9 B5 B1049.7	15600
5	F9 B5 B1051.3	15600
6	F9 B5 B1051.4	15600
7	F9 B5 B1051.6	15600
8	F9 B5 B1056.4	15600
9	F9 B5 B1058.3	15600
10	F9 B5 B1060.2	15600
11	F9 B5 B1060.3	15600

2015 Launch Records

- Used “select… where … like… and DATE BETWEEN…. And….” to list the failure landing in year 2015

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

```
task_9 = '''
        SELECT BoosterVersion, LaunchSite, LandingOutcome
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Failure (drone ship)'
            AND Date BETWEEN '2015-01-01' AND '2015-12-31'
        '''
create_pandas_df(task_9, database=conn)
```

	boosterversion	launchsite	landingoutcome
0	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
1	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

- Used ‘select… where date between… and… Group by …. Group by COUNT () DESC’ to rand outcome between the dates

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

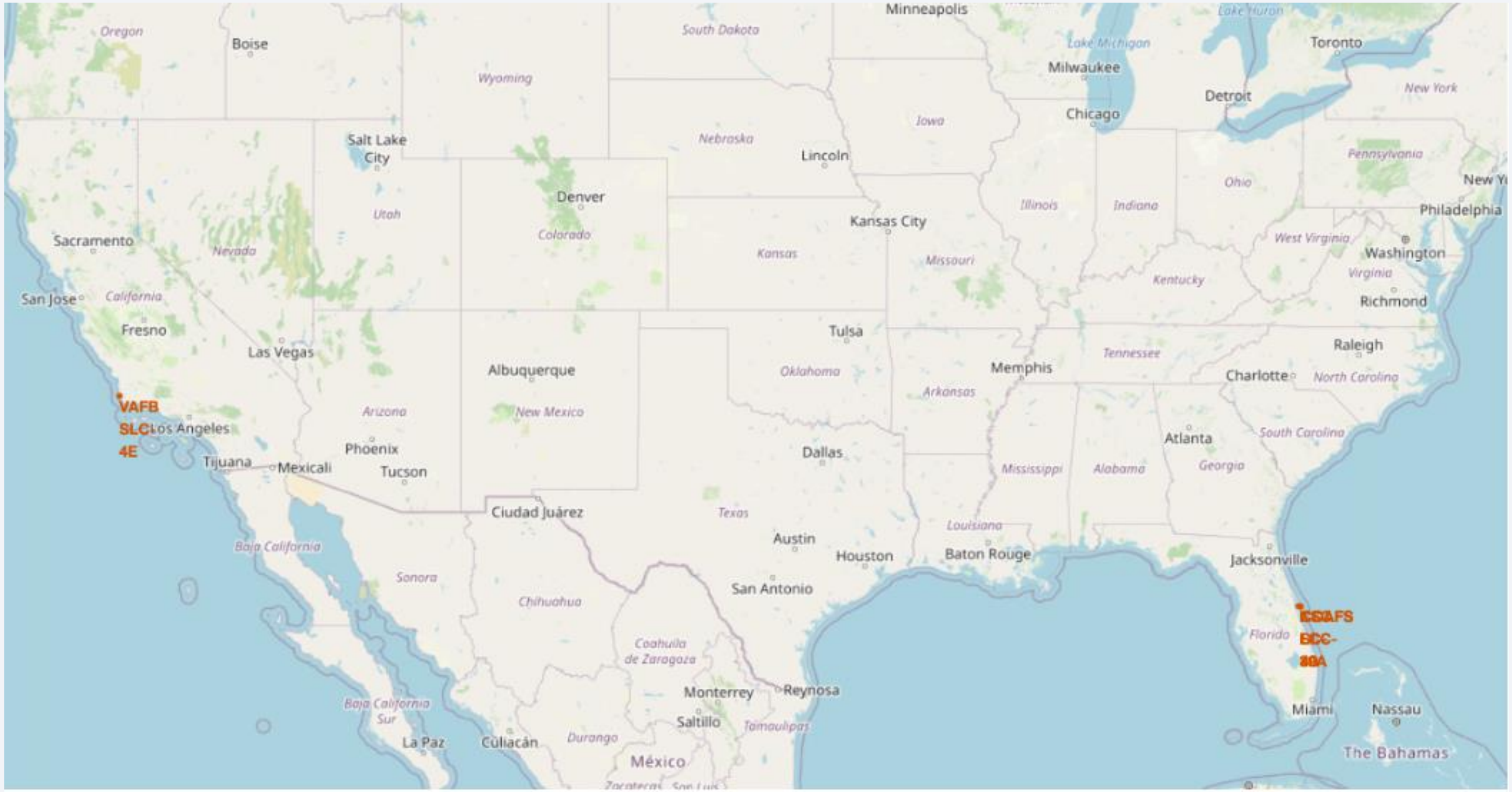
```
task_10 = '''
        SELECT LandingOutcome, COUNT(LandingOutcome)
        FROM SpaceX
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
        GROUP BY LandingOutcome
        ORDER BY COUNT(LandingOutcome) DESC
        '''
create_pandas_df(task_10, database=conn)
```

	landingoutcome	count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1
7	Failure (parachute)	1

Section 3

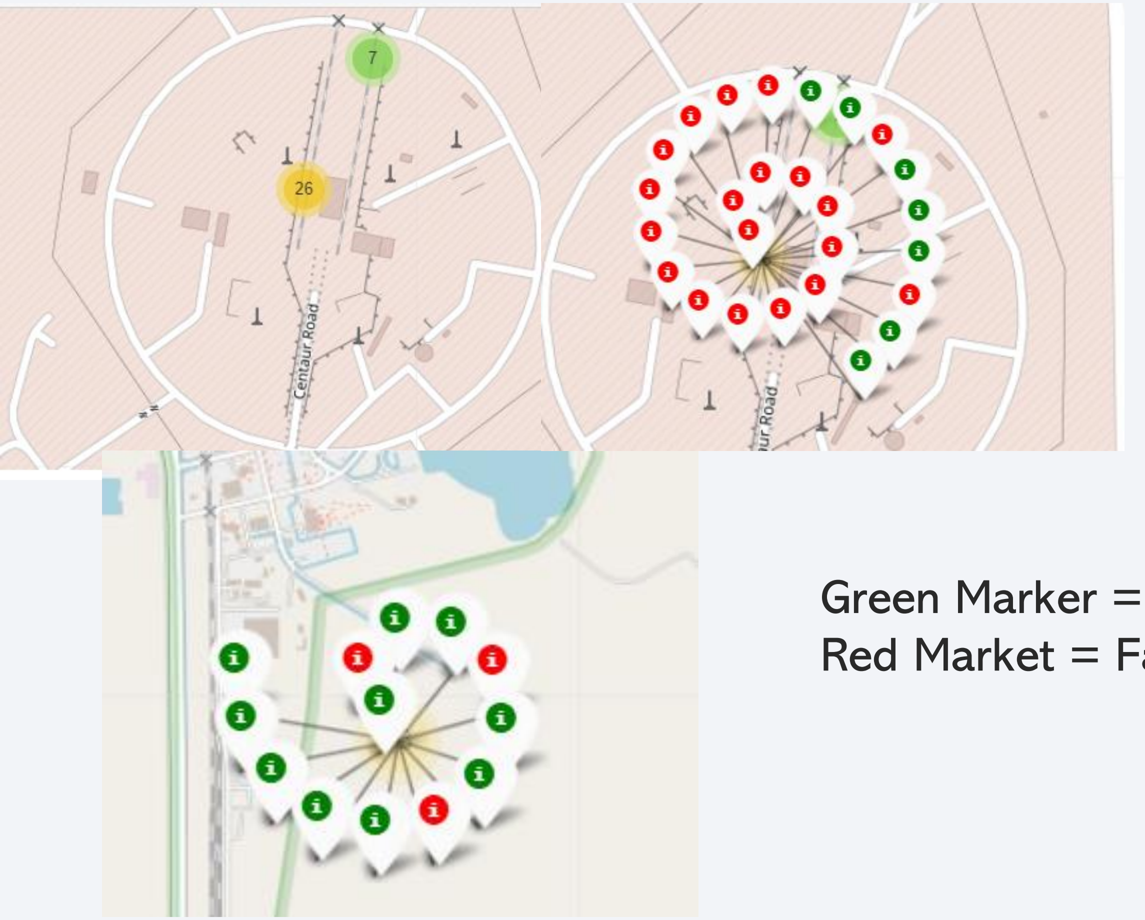
Launch Sites Proximities Analysis

All Launch Sites Locations



- All launch sites are in the US coasts – east and west.

Launch outcomes on map



Green Marker = Success
Red Market = Failer

Florida Launch sites

California Launch sites

Launch site distance to Coast

- Re
w

- Ex
sh
si
hi
ar

- Ex
fir

Section 4

Build a Dashboard with Plotly Dash

<Dashboard Screenshot 1>

- Replace <Dashboard screenshot 1> title with an appropriate title
- Show the screenshot of launch success count for all sites, in a piechart
- Explain the important elements and findings on the screenshot

<Dashboard Screenshot 2>

- Replace <Dashboard screenshot 2> title with an appropriate title

- Show the screenshot of the piechart for the launch site with highest launch success ratio

- Explain the important elements and findings on the screenshot

<Dashboard Screenshot 3>

- Replace <Dashboard screenshot 3> title with an appropriate title

- Show screenshots of Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider

- Explain the important elements and findings on the screenshot, such as which payload range or booster version have the largest success rate, etc.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- Best model is decisionTree

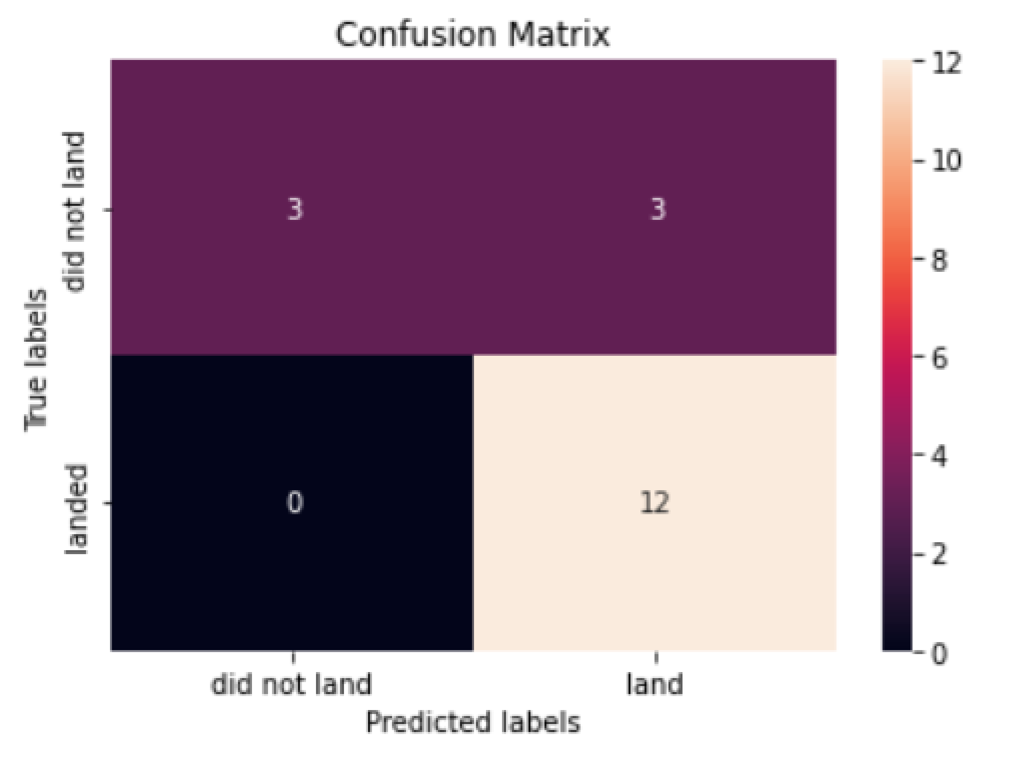
Find the method performs best:

```
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'}
```

Confusion Matrix

- The confusion matrix of decision tree classifier shows it can differentiate different class. The error is around false positive.



Conclusions

- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- The greater the payload mass for launch site CCAFS SLC 40, the higher success rate for launch
- ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- 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.
- With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.
- The decision tree is the best machine learning classifier model.

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!