IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Aastha Singh
27 March 2024

Outline

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

Executive Summary

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

Introduction

- Project background and context

Space X 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 Space X 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 space X for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

- Problems you want to find answers

  - Factors responsible for a successful landing of the rocket.
  - Inter-Relationship between different features responsible for a successful landing of the rocket.
  - Conditions to met to ensure a successful landing program.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX API and web scraping from Wikipedia.
- 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 models
  - How to build, tune, evaluate classification models

Data Collection

SpaceX API

API Request
- Request launch data from SPACEX API

Read API response
- Read .json response and convert to a data frame using json_normalize method

Prelim Data Wrangling
- Extract relevant data fields to match the requirements

Convert to CSV format
- Convert new Dataframe to CSV format for the next phase

Web scraping data from Wiki

Perform HTTP GET
- Request Falcon9 html page via HTTP GET method

Create BeautifulSoup
- Create BeautifulSoup object from HTML response

Extract data to df
- Collect all relevant headers and extract rows to dataframe.

Convert to CSV format
- Convert Dataframe to CSV for next phase.

Data Collection – SpaceX API

- We used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.
- https://github.com/aastapasta/DATA-Science-project-for-analysing-SpaceX/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Step 1

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
response = requests.get(spacex_url)
```

Check the content of the response

```
print(response.content)
```

Step 2

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_
```

We should see that the request was successfull with the 200 status response code

```
response.status_code
```

```
200
```

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())
```

Using the dataframe `data` print the first 5 rows

```
# Get the head of the dataframe
data.head()
```

Step 3

```
# Create a data from launch_dict
data = pd.DataFrame(launch_dict)
```

Show the summary of the dataframe

```
# Show the head of the dataframe
data.head()
```

Step 4

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = data[data['BoosterVersion'] != 'Falcon 1']
data_falcon9
```

Step 5

```
# Calculate the mean value of PayloadMass column
PayloadMass = pd.DataFrame(data_falcon9['PayloadMass'].values.tolist()).mean()
print(PayloadMass)
# Replace the np.nan values with its mean value
data_falcon9["PayloadMass"].replace(np.nan, data_falcon9["PayloadMass"].mean(), inplace=True)
data_falcon9
```

Data Collection - Scraping

- We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup
- We parsed the table and converted it into a pandas dataframe.
- https://github.com/aastapasta/DATA-Science-project-for-analysing-SpaceX/blob/main/jupyter-labs-webscraping.ipynb

Step 1

```
# 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
```

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
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

Step 2

```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

Starting from the third table is our target table contains the actual launch records.

```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

Step 3

```
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
```

Check the extracted column names

```
print(column_names)
```

Data Wrangling

- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.
- https://github.com/aastapasta/DATA-Science-project-for-analysing-SpaceX/blob/main/labs-jupyter-spacex-Data%20wrangling%20(1).ipynb

Step 1

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

Step 2

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

Step 3

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

Step 4

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []

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

EDA with Data Visualization

As part of the Exploratory Data Analysis (EDA), following charts were plotted to gain further insights into the dataset:

1. Scatter plot to visualize :

- Relationship between Flight Number and Launch Site
- Relationship between Payload and Launch Site
- Relationship between Flight Number and Orbit Type
- Relationship between Payload and Orbit Type

2. Bar chart to visualize: Relationship between success rate of each orbit type

3. Line chart to observe: Average launch success yearly trend

- https://github.com/aastapasta/DATA-Science-project-for-analysing-SpaceX/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.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 for instance:
  - The names of unique launch sites in the space mission.
  - The total payload mass carried by boosters launched by NASA (CRS)
  - The average payload mass carried by booster version F9 v1.1
  - The total number of successful and failure mission outcomes
  - The failed landing outcomes in drone ship, their booster version and launch site names.
- https://github.com/aastapasta/DATA-Science-project-for-analysing-SpaceX/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the distances between a launch site to its proximities. We answered some question for instance:
  - Are launch sites near railways, highways and coastlines.
  - Do launch sites keep certain distance away from cities.
- https://github.com/aastapasta/DATA-Science-project-for-analysing-SpaceX/blob/main/lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- https://github.com/aastapasta/DATA-Science-project-for-analysing-SpaceX/blob/main/dash_interactivity.py

Predictive Analysis (Classification)

- We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.
- https://github.com/aastapasta/DATA-Science-project-for-analysing-SpaceX/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Step 1

```
Y = data['Class'].to_numpy()
Y
```

Step 2

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

```
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.

Step 3

```
Y_test.shape
```

Step 4

```
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()
grid_search_lr = GridSearchCV(
    estimator = lr,
    param_grid = parameters,
    scoring = 'accuracy',
    cv = 10
)
# execute search
logreg_cv = grid_search_lr.fit(X_train,Y_train)
```

We output the `GridSearchCV` object for logistic regression. We display the best parameters using th
the accuracy on the validation data using the data attribute `best_score_`.

```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("accuracy :",logreg_cv.best_score_)
```

Results

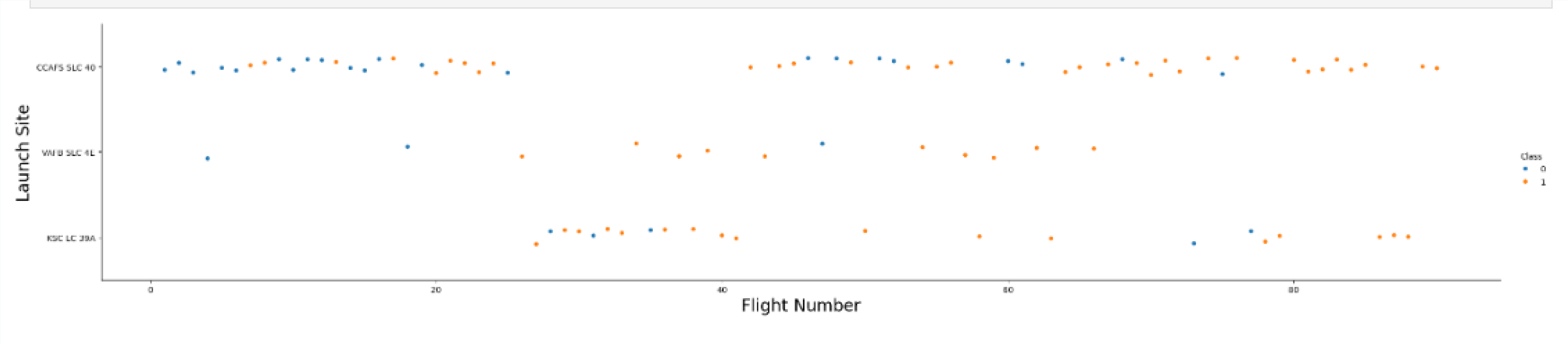
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

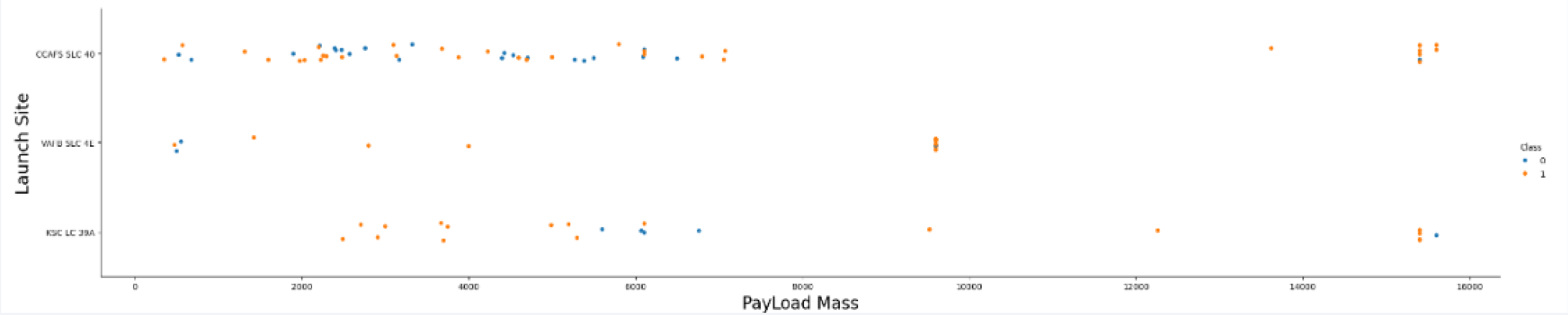
Flight Number vs. Launch Site

- Success rates (Class=1) increases as the number of flights increase
- For launch site ‘KSC LC 39A’, it takes at least around 25 launches before a first successful launch



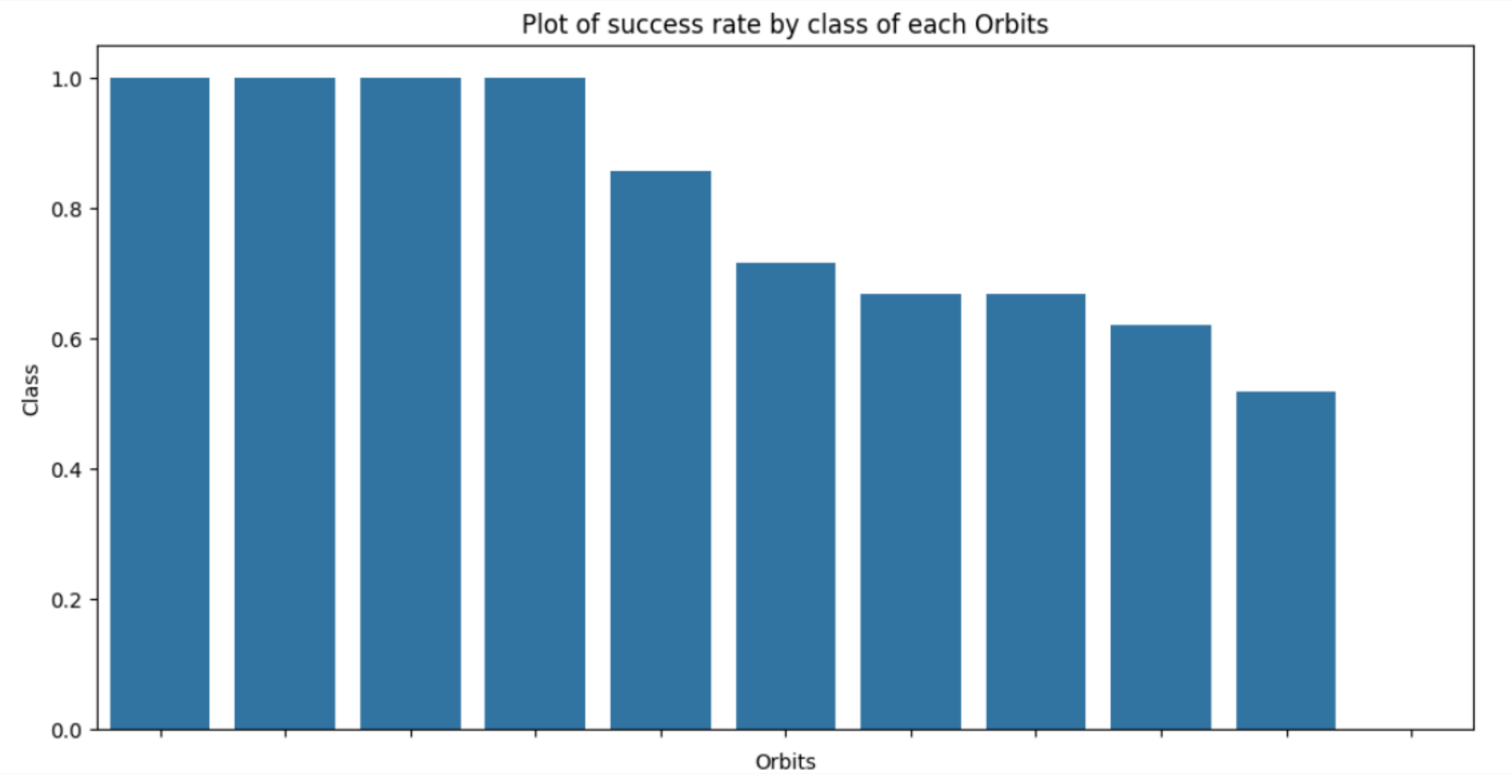
Payload vs. Launch Site

- For launch site ‘VAFB SLC 4E’, there are no rockets launched for payload greater than 10,000 kg
- Percentage of successful launch (Class=1) increases for launch site ‘VAFB SLC 4E’ as the payload mass increases
- There is no clear correlation or pattern between launch site and payload mass



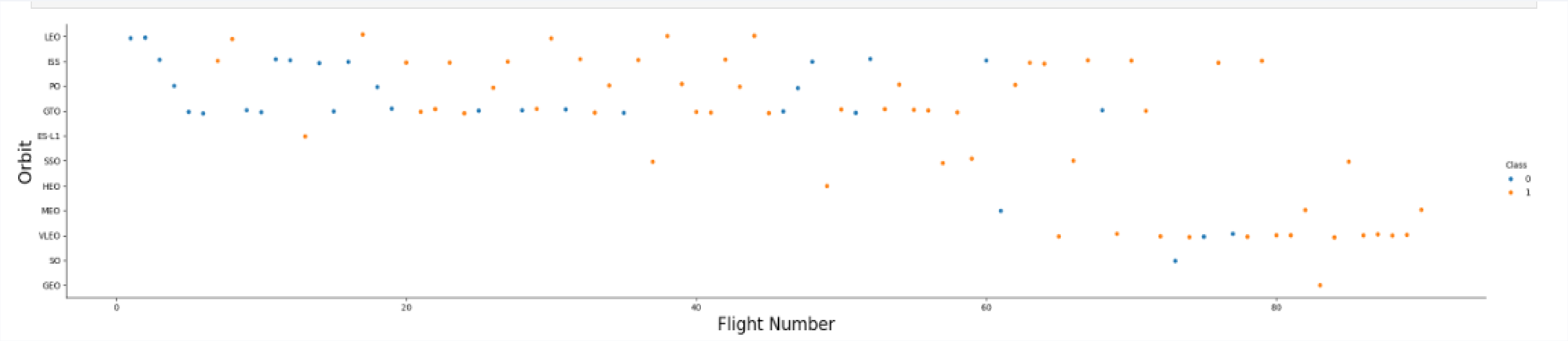
Success Rate vs. Orbit Type

- Orbits ES-LI, GEO, HEO, and SSO have the highest success rates
- GTO orbit has the lowest success rate



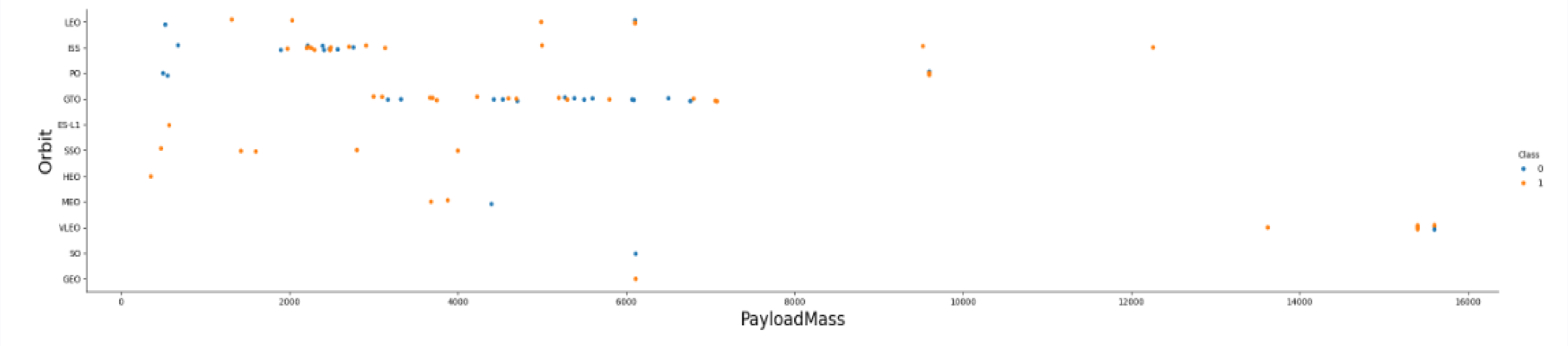
Flight Number vs. Orbit Type

- For orbit VLEO, first successful landing (class=1) doesn't occur until 60+ number of flights
- For most orbits (LEO, ISS, PO, SSO, MEO, VLEO) successful landing rates appear to increase with flight numbers 22
- There is no relationship between flight number and orbit for GTO



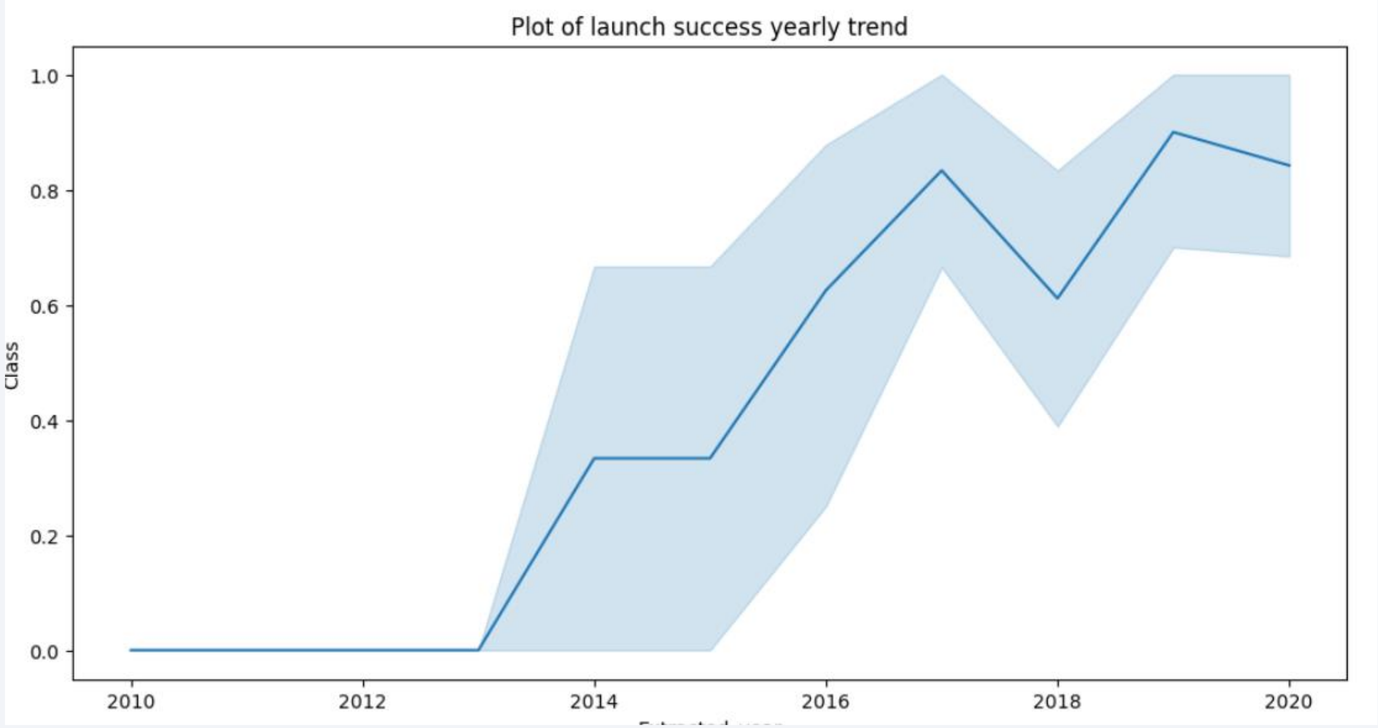
Payload vs. Orbit Type

- Successful landing rates (Class=1) appear to increase with pay load for orbits LEO, ISS, PO, and SSO
- For GEO orbit, there is not clear pattern between payload and orbit for successful or unsuccessful landing



Launch Success Yearly Trend

- Success rate (Class=1) increased by about 80% between 2013 and 2020
- Success rates remained the same between 2010 and 2013 and between 2014 and 2015
- Success rates decreased between 2017 and 2018 and between 2019 and 24 2020



All Launch Site Names

- We used the key word **DISTINCT** to show only unique launch sites from the SpaceX data.

Task 1

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

In [17]:

```
%%sql

select distinct Launch_Site from spacextbl
```

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

Out[17]:

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

Launch Site Names Begin with 'CCA'

We used the query above to 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.
```

Total Payload Mass

We calculated the total payload carried by boosters from NASA as 45596 using the query below

```
%%sql

select sum(PAYLOAD_MASS__KG_) from spacextbl where Customer = 'NASA (CRS)'
```

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

sum(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

We calculated the average payload mass carried by booster version F9 v1.1 as 2928.4

```
%%sql

select avg(PAYLOAD_MASS__KG_) from spacextbl where Booster_Version LIKE 'F9 v1.1';
```

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

avg(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

We observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015

```
%%sql

select min(Date) as min_date from spacextbl where Landing__Outcome = 'Success (ground pad)';
```

min_date
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

We used the **WHERE** clause to filter for boosters which have successfully landed on drone ship and applied the **AND** condition to determine successful landing with payload mass greater than 4000 but less than 6000

```
%%sql

select Booster_Version from spacextbl where (PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MASS__KG_ < 6000)
and (Landing__Outcome = 'Success (drone ship)');
```

	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

We used wildcard like ‘%’ to filter for **WHERE** MissionOutcome was a success or a failure.

```
%%sql

select Mission_Outcome, count(Mission_Outcome) as counts from spacextbl group by Mission_Outcome;
```

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

Mission_Outcome	counts
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

We determined the booster that have carried the maximum payload using a subquery in the **WHERE** clause and the **MAX()** function.

```
%%sql
select Booster_Version, PAYLOAD_MASS__KG_ from spacextbl where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from spac
```

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

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

2015 Launch Records

We used a combinations of the **WHERE** clause, **LIKE**, **AND**, and **BETWEEN** conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

```
%%sql

select Landing__Outcome, Booster_Version, Launch_Site from spacextbl where Landing__Outcome = 'Failure (drone ship)'
```

landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

- We selected Landing outcomes and the **COUNT** of landing outcomes from the data and used the **WHERE** clause to filter for landing outcomes **BETWEEN** 2010-06-04 to 2010-03-20.
- We applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome in descending order.

```
%%sql

select Landing__Outcome, count(*) as LandingCounts from spacextbl where Date between '2010-06-04' and '2017-03-20'
group by Landing__Outcome
order by count(*) desc;/
```

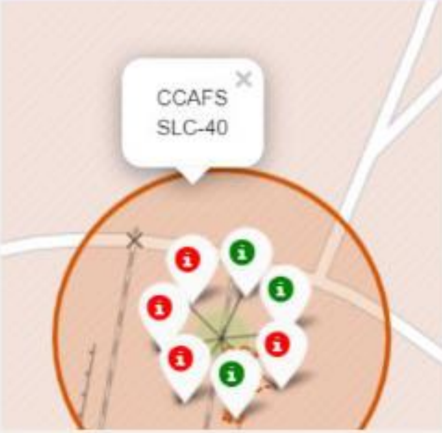
landing__outcome	landingcounts
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Success (ground pad)	5
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	1
Precluded (drone ship)	1

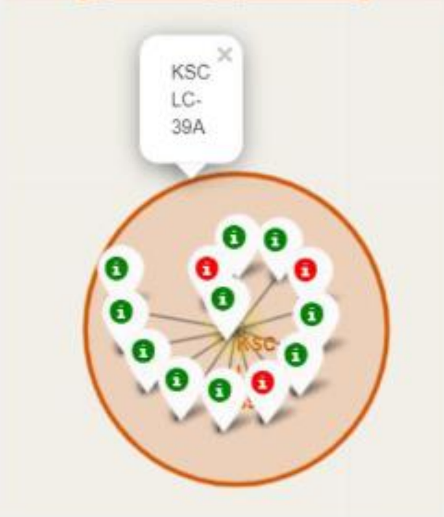
Section 3

Launch Sites Proximities Analysis

All Launch Sites

Color Labels





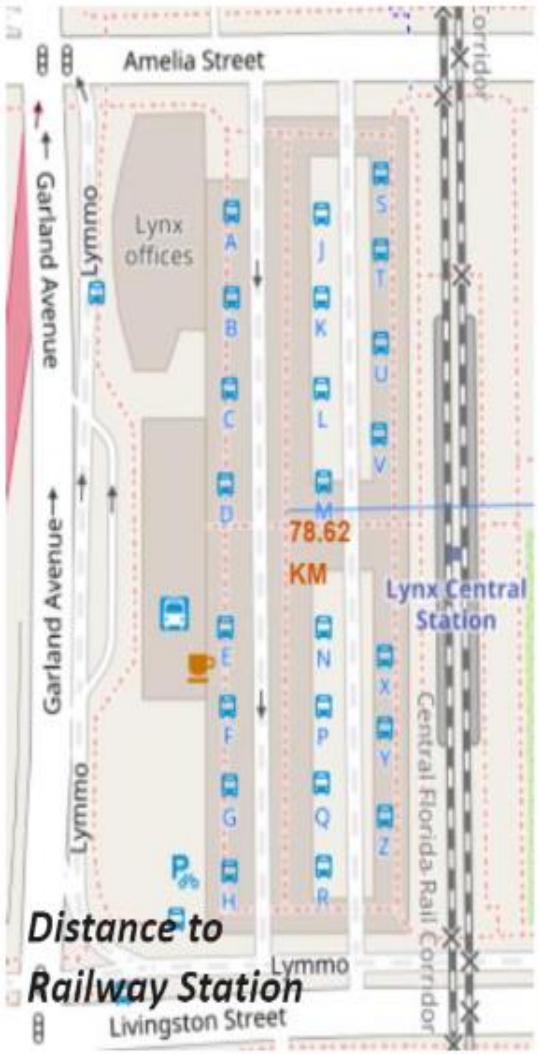
Florida Launch Sites

Green Marker shows successful Launches and Red Marker shows Failures

California Launch Site

37

Distances



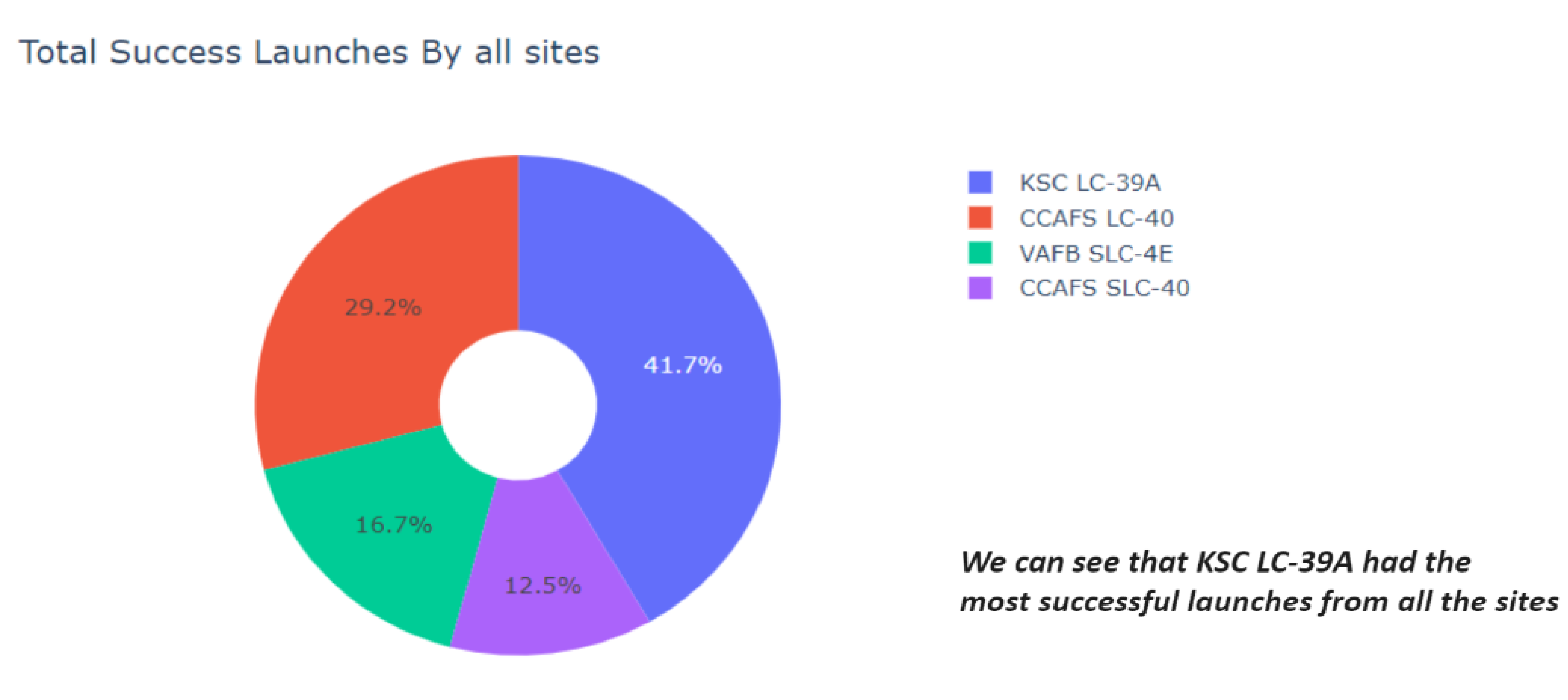
- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Section 4

Build a Dashboard with Plotly Dash

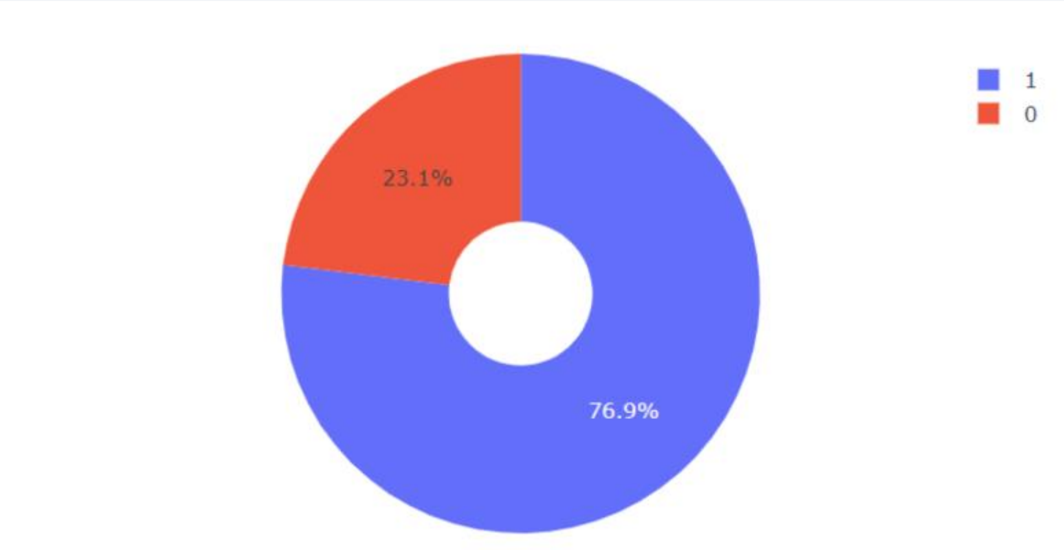
Launch Success Counts For All Sites

- Launch Site ‘KSC LC-39A’ has the highest launch success rate
- Launch Site ‘CCAFS SLC 40’ has the lowest launch success rate



Launch Site with Highest Launch Success Ratio

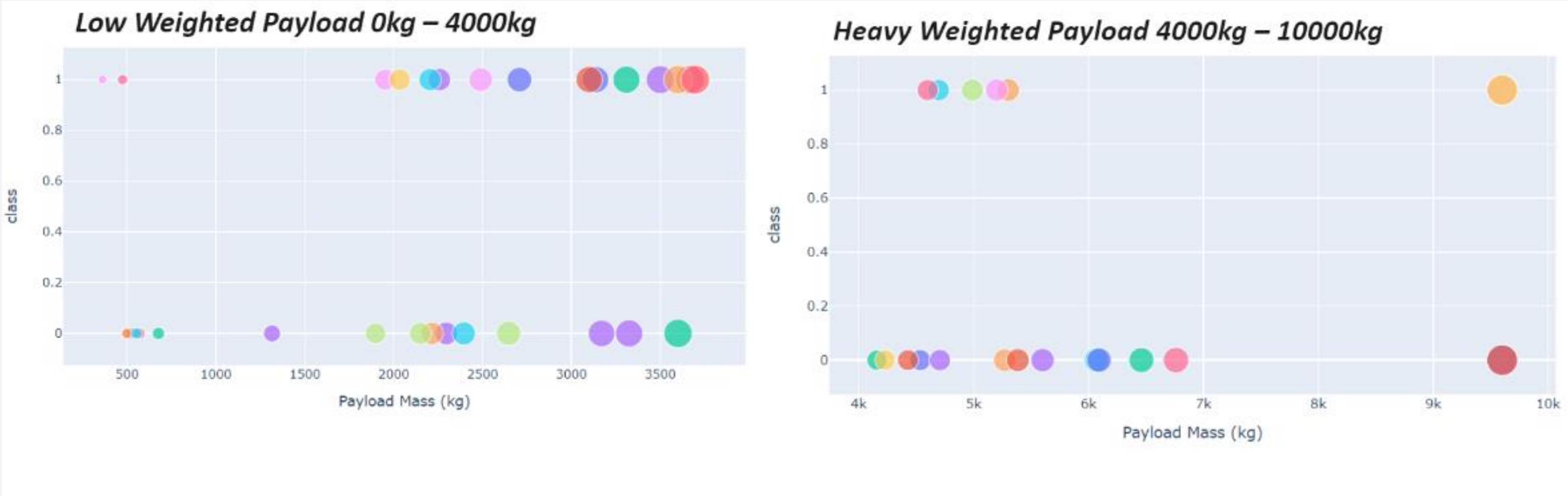
- KSC LC-39A Launch Site has the highest launch success rate and count
- Launch success rate is 76.9%
- Launch success failure rate is 23.1%



KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

Payload vs. Launch Outcome Scatter Plot for All Sites

- Most successful launches are in the payload range from 2000 to about 5500
- Booster version category ‘FT’ has the most successful launches
- Only booster with a success launch when payload is greater than 6k is ‘B4’



We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- Based on the Accuracy scores and as also evident from the bar chart, Decision Tree algorithm has the highest classification score with a value of .88928

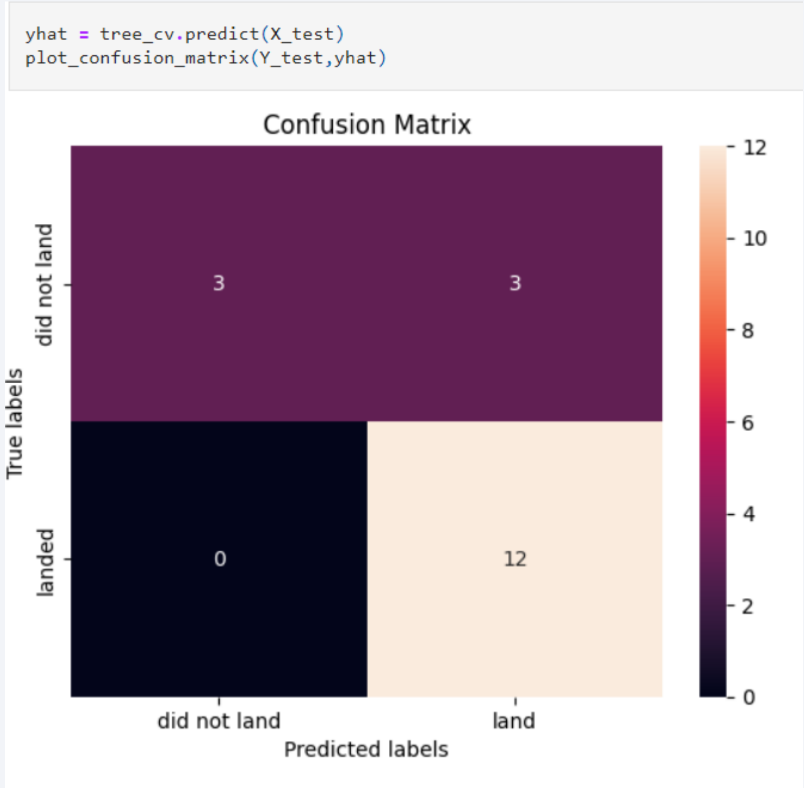
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

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 0.8892857142857145
Best params is : {'criterion': 'gini', 'max_depth': 2, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 2,
'splitter': 'random'}
```

Confusion Matrix

- The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives .i.e., unsuccessful landing marked as successful landing by the classifier.



Conclusions

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.
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- KSC LC-39A had the most successful launches of any sites.
- The Decision tree classifier is the best machine learning algorithm for this task.

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!