IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Zunera Bokhari

Outline

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

Executive Summary

- Summary of methodologies
  - ***Data Collection*** through API and Web Scraping
  - ***Data Wrangling*** to create successful outcome variable
  - ***Exploratory Data Analysis*** with SQL and Data Visualization focusing on payload, launch site, flight number, and yearly trend
  - ***Analyzing data*** with SQL and interactive visualizations with Folium
  - ***Visualize data*** such as launch sites and the most successful payload ranges
  - ***Build Machine Learning Models*** to predict landing outcomes using logistic regression, SVM, decision tree, and KNN
- Summary of all results
  - Exploratory Data Analysis results
  - Visual analytics using graphs and interactive visualizations
  - Predictive Analysis results from Machine Learning Models

Introduction

Background

SpaceX prominently features Falcon 9 rocket launches on its website, boasting a price tag of $62 million, a stark contrast to other providers whose costs soar beyond $165 million per launch. The primary cost-saving mechanism employed by SpaceX involves the innovative practice of re-landing and reusing the rocket's first stage for subsequent missions. Therefore, the ability to ascertain the successful landing of the first stage becomes a crucial factor in determining the overall launch cost. This information becomes invaluable for other companies seeking to compete with SpaceX in bidding for rocket launches, offering a strategic insight into cost considerations and enhancing competitiveness in the aerospace industry.

Problems Explored:

- Finding all factors that influence landing outcomes and their relationship
- Rate of landing success and best conditions needed to increase success
- Models to explore successful landings

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Using SpaceX REST API and webscraping from Wikipedia
- Perform data wrangling
  - Processing data using one-hot encoding for 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

Steps:

1. Request data from SpaceX API
2. Use .json()_normalize to convert results to a readable dataframe
3. Data cleaning by keeping only features we want and converting date go correct datatype
4. Filter data to only include Falcon 9 launches
5. Replace missing values with Payload Mass mean

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

response = requests.get(spacex_url)
```

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

```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boo
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list a
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = launch_data[launch_data['BoosterVersion']!='Falcon 1']
data_falcon9.head()
```

```
# Calculate the mean value of PayloadMass column
mean=data_falcon9['PayloadMass'].mean()

# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].replace(np.nan,mean, inplace=True)
```

https://github.com/zunerabokhari/IBM-Data-Science-Capstone-SpaceX/blob/main/1_SpaceX_Data_Collection.ipynb

Data Collection - Scraping

Steps:

1. Request Falcon 9 launch data from Wikipedia
2. Create BeautifulSoup object from the HTML response
3. Extract all column and variable names from the HTML tables and headers and create dictionary from the data
4. Create dataframe from the dictionary data

```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })

df.head()
```

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

data = requests.get(static_url).text
```

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

```
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']=[]
```

```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
temp = soup.find_all('th')

# Iterate each th element and apply the provided extract_column_from_header() to get a column name
for x in range(len(temp)):
    try:
        name = extract_column_from_header(temp[x])
        if (name is not None and len(name)>0):
            column_names.append(name)
    except:
        pass
```

https://github.com/zunerabokhari/IBM-Data-Science-Capstone-SpaceX/blob/main/2_SpaceX_Web_Scraping.ipynb

Data Wrangling

Data wrangling is the process of cleaning data and transforming it into desired formats to make it easier to consume and perform exploratory data analysis.

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

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

We calculated:

- Number of launches for each site
- Number and occurrences of each orbit
- Number and occurrence of mission outcome of orbits

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

```
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
```

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

landing_outcomes
```

```
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
```

```
df['Class']=landing_class
df[['Class']].head(8)
```

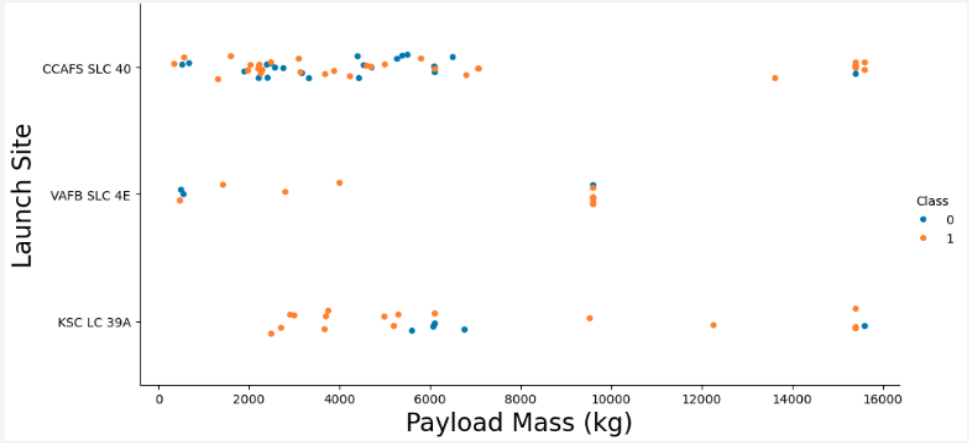
	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

We then created a binary landing outcome column which will make it easier for analysis, making visualizations, and machine learning.

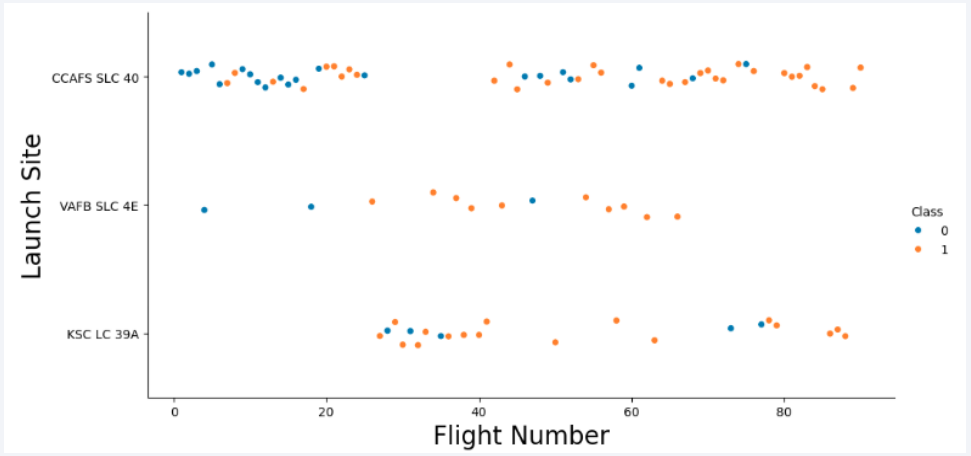
https://github.com/zunerabokhari/IBM-Data-Science-Capstone-SpaceX/blob/main/3_SpaceX_Data_Wrangling.ipynb

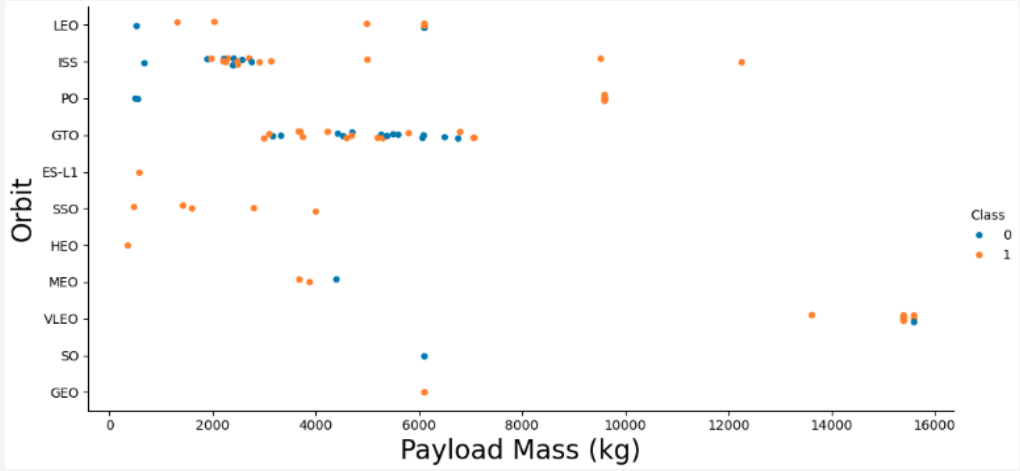
EDA with Data Visualization

Scatter plots show the relationship of attributes with each other. Once we find a pattern from graphs we're able to see which factors affect the success of landing outcomes.



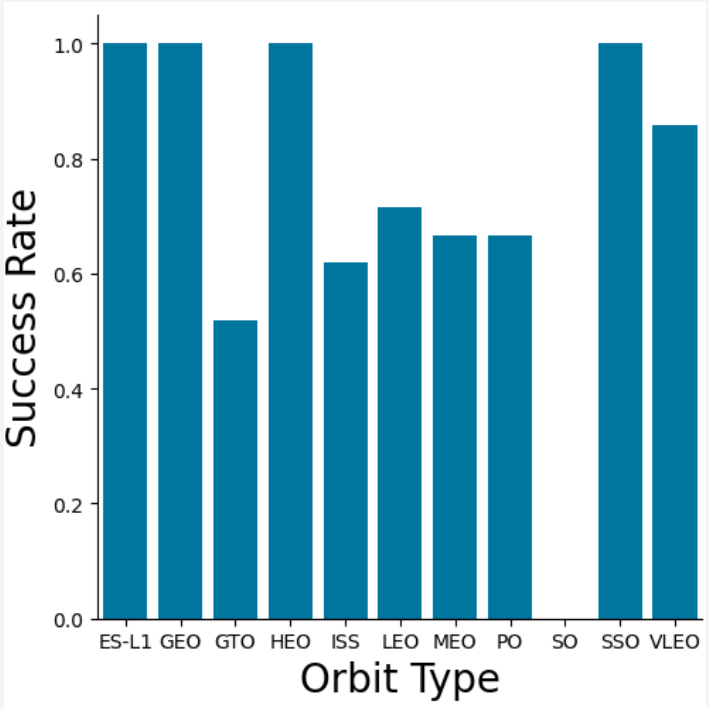
https://github.com/zunerabokhari/IBM-Data-Science-Capstone-SpaceX/blob/main/5_SpaceX_EDA_Data_Visualization.ipynb

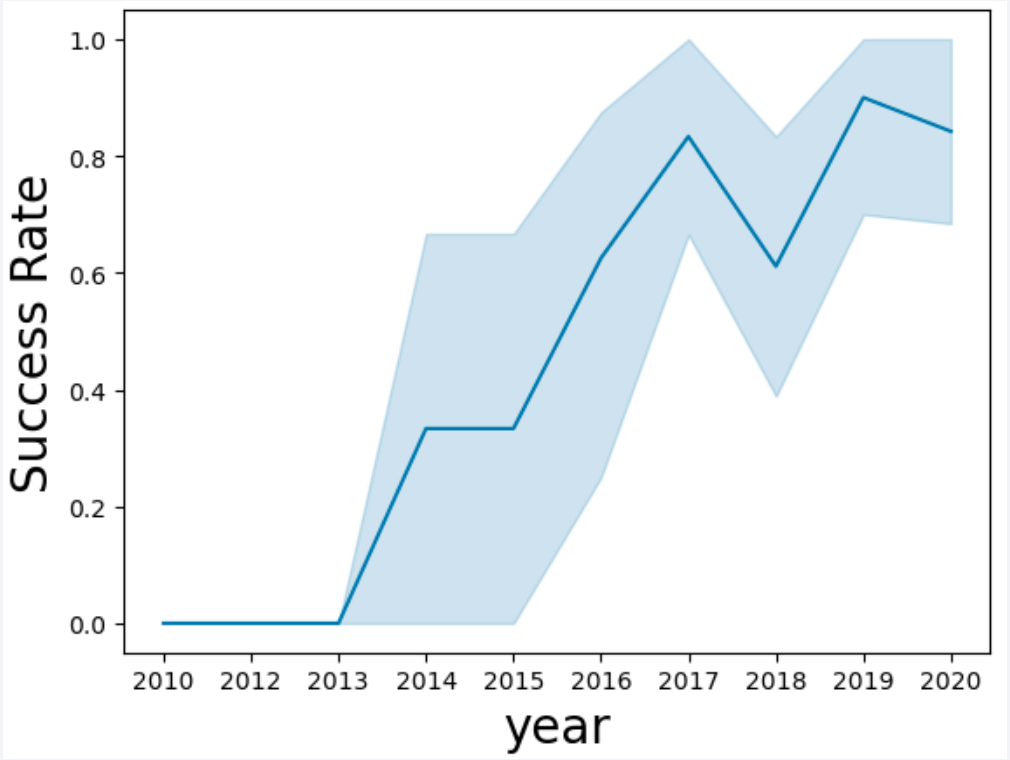




EDA with Data Visualization

Bar graphs are one of the easiest ways to interpret the relationship between attributes. They show comparisons among discrete categories. We used this bar graph to determine which orbits have the highest probability of a successful landing.





Line graphs allow us to show trends or patterns of different attributes over time. Here we used it to see the yearly trend of launch success.

https://github.com/zunerabokhari/IBM-Data-Science-Capstone-SpaceX/blob/main/5_SpaceX_EDA_Data_Visualization.ipynb

EDA with SQL

- Display launch sites
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome in ground pad was achieved
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster versions which have carried the maximum payload mass
- List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015
- Rank the count of landing outcomes between the date 2010-06-04 and 2017-03-20, in descending order

- https://github.com/zunerabokhari/IBM-Data-Science-Capstone-SpaceX/blob/main/4_SpaceX_EDA_SQL.ipynb

Build an Interactive Map with Folium

We created an interactive map to visualize launch data by taking longitude and latitude of each launch site and adding a circle marker around each launch site.

We then added colored markers of successful and unsuccessful launches are each launch site to show their success rates.

We also added colored lines to visualize the distance between launch sites. Then we used Haversine's formula to calculate the proximity to the nearest coastline, railway, highway, and city.

https://github.com/zunerabokhari/IBM-Data-Science-Capstone-SpaceX/blob/main/6_SpaceX_Interactive_Visualizations_Folium.ipynb

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly Dash which allows users to interact with data
  - a dropdown list with launch sites
    - allows users to select all or one site at a time
- Pie chart displaying successful launches
  - Shows users successful and unsuccessful launches as percentages
- Slider of Payload Mass Range
  - Allows users to select payload mass range
- Scatter chart of Payload Mass vs Outcome
  - Shows the relationship between payload mass and outcome for the different booster version

https://github.com/zunerabokhari/IBM-Data-Science-Capstone-SpaceX/blob/main/7_SpaceX_Interactive_Visualizations_Plotly.py

Predictive Analysis (Classification)

- Created a NumPy array from the Class column
- Standardized the data using StandardScaler, fit and transformed the data
- Split data using train_test_split
- Create
  - a logistic regression object
  - a GridSearchCV object and fit it to find best parameters
  - A support vector machine
  - A decision tree classifier
  - A KNN
- Calculate the accuracy on the test date using .score() for all models created
- Assess the confusion matrix for all models
- Find the method that performs best

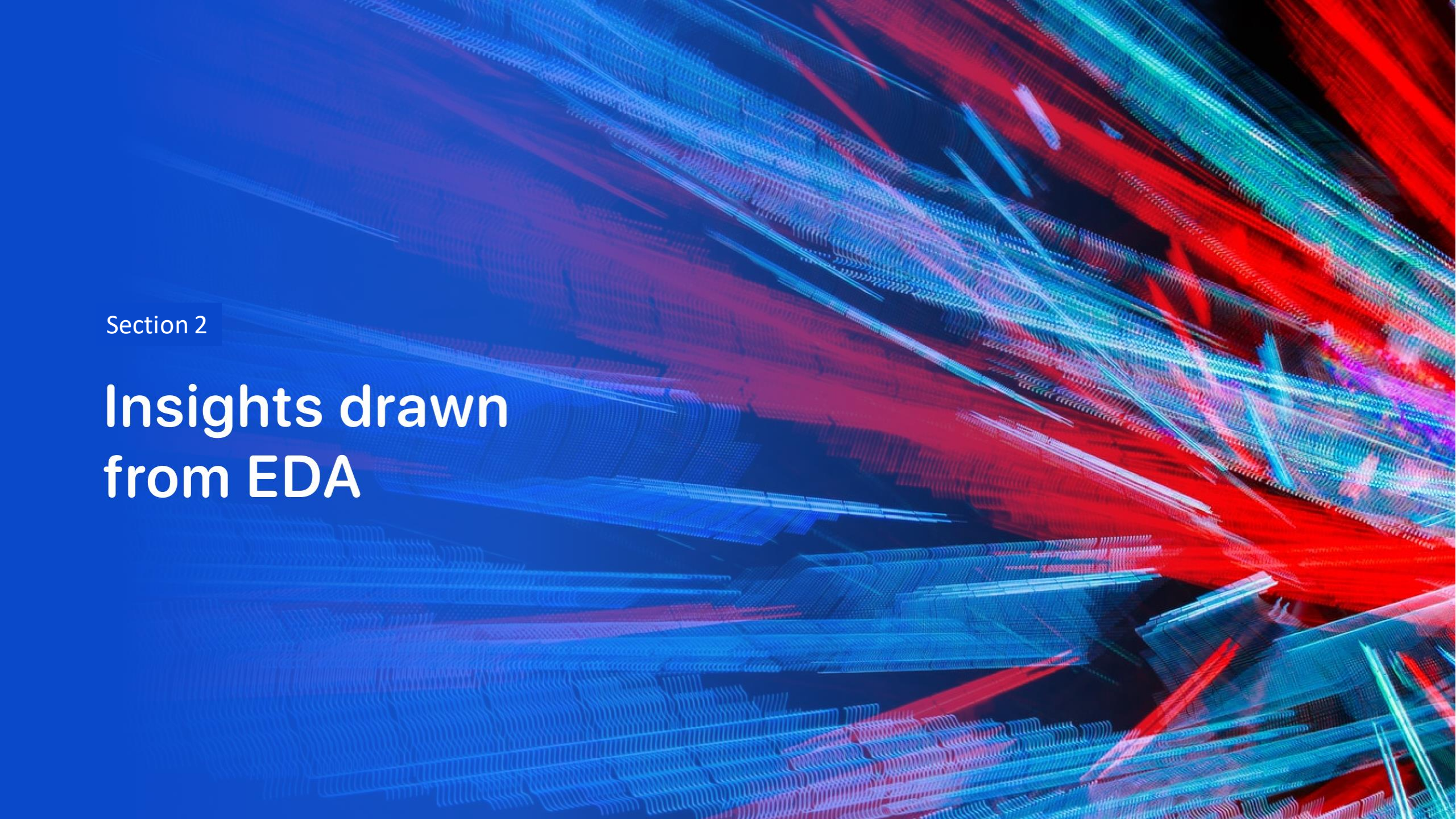
https://github.com/zunerabokhari/IBM-Data-Science-Capstone-SpaceX/blob/main/8_SpaceX_Predictive_Analysis.ipynb

Results

- Exploratory Data Analysis
  - Launch success has improved over time
- Visual Analysis
  - Most launch sites are near the equator and all are close to the coast
  - Launch sites are far from major sites so a failed launch won't damage anything such as a city, highway, or railway
- Predictive Analysis
  - Decision Tree is the best predictive model for this data

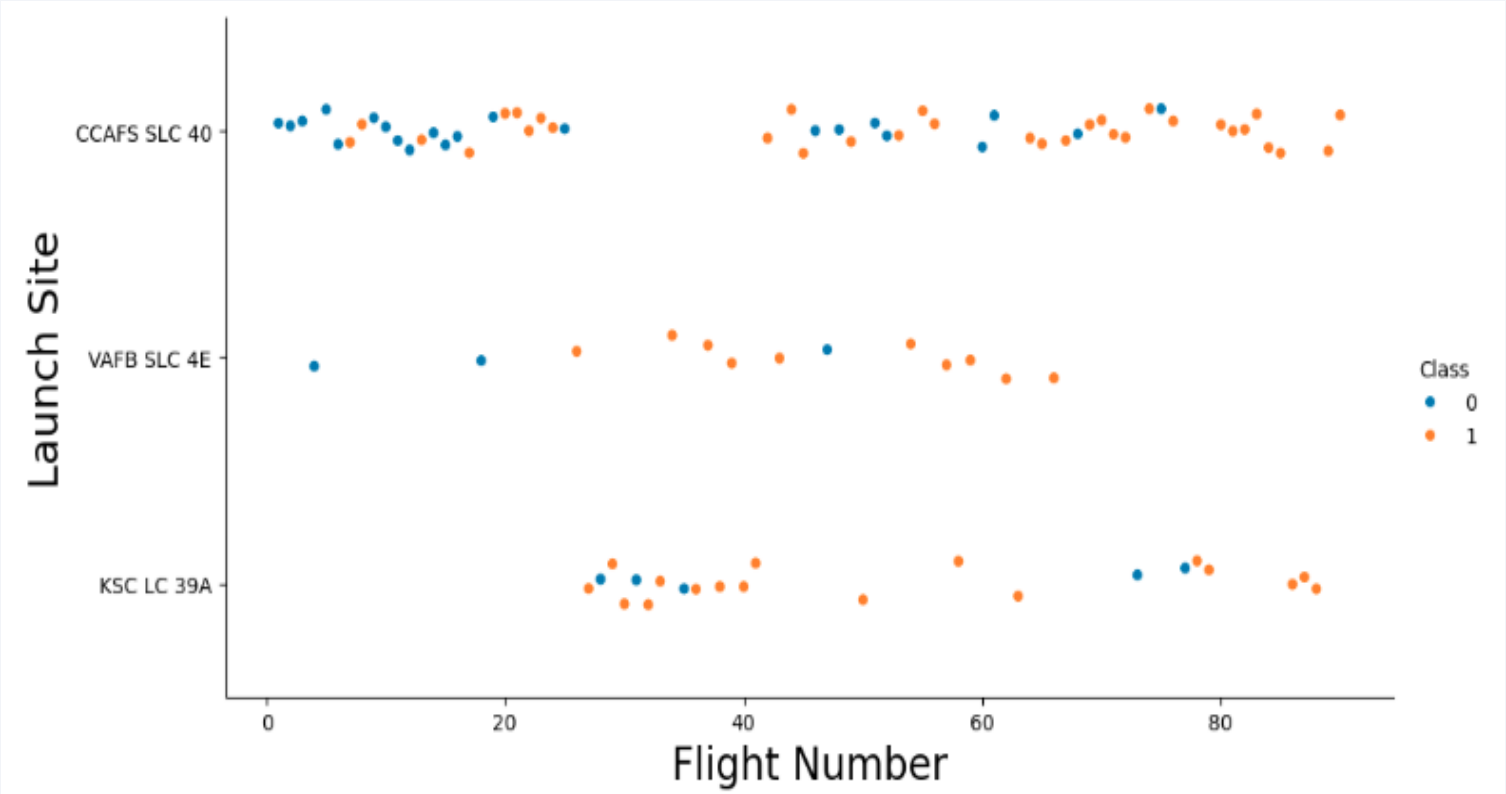
Section 2

Insights drawn from EDA



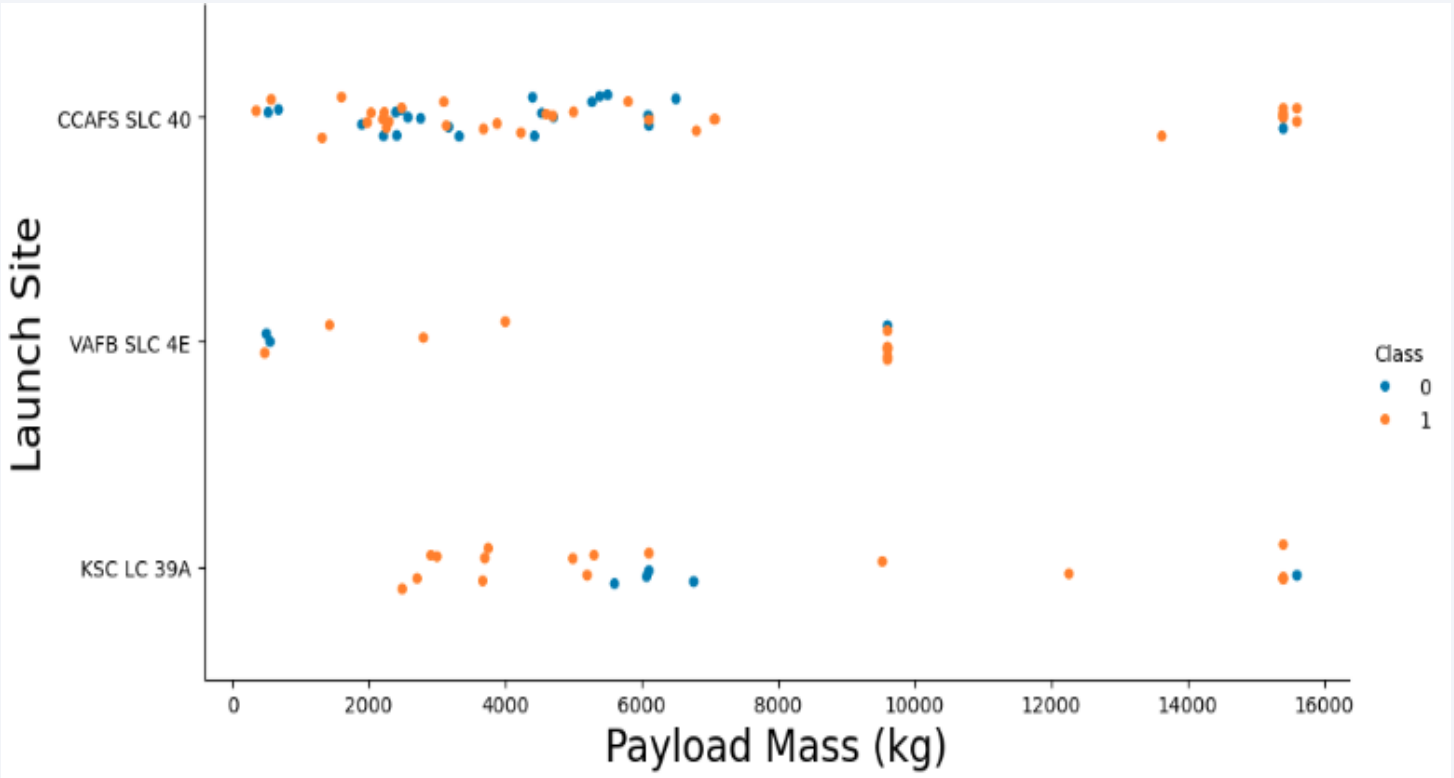
Flight Number vs. Launch Site

- It seems that earlier flights had a lower success rate and later flights had a higher success rate
- VAFB SLC 4E and KSC LC 39A have higher success rates
- Approximately half launches were from CCAFS SLC 40 launch site



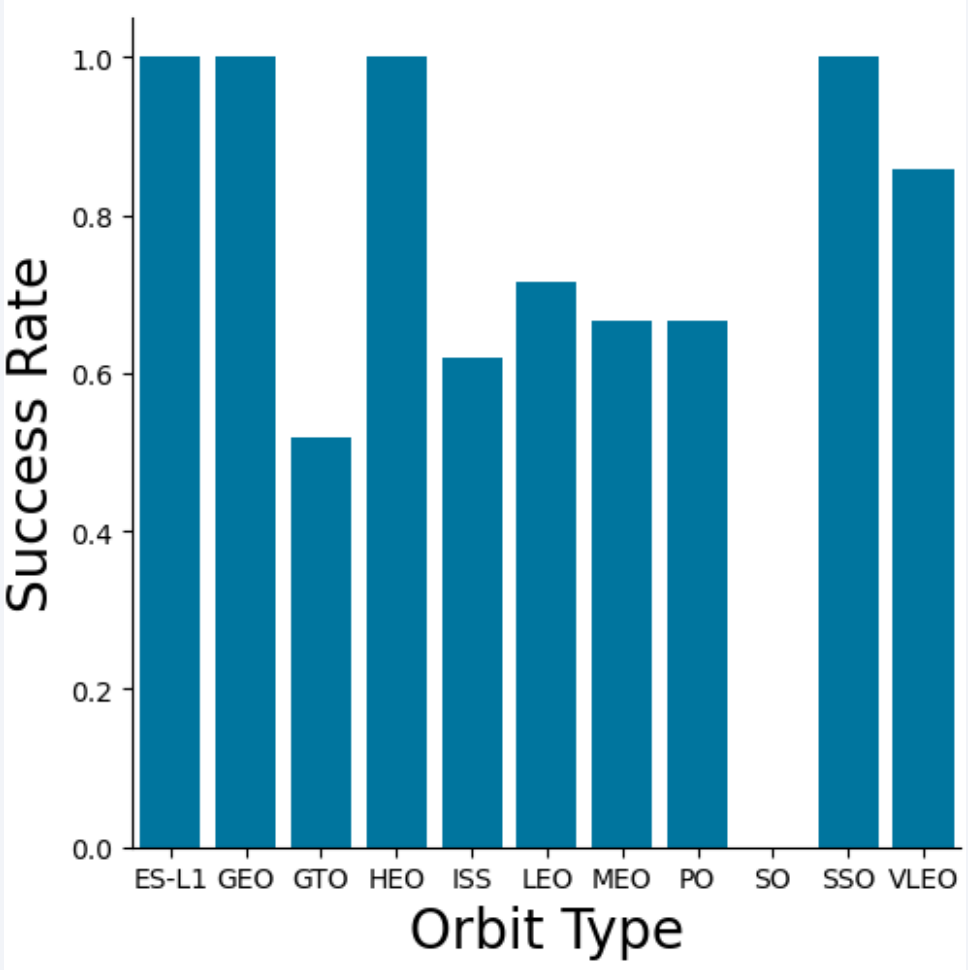
Payload vs. Launch Site

- The higher the payload mass, the higher the success rate
- KSC LC 39A has a 100% success rate for launches less than ~5000 kg
- VAFB SKC 4E has not launch anything more than 10,000kg



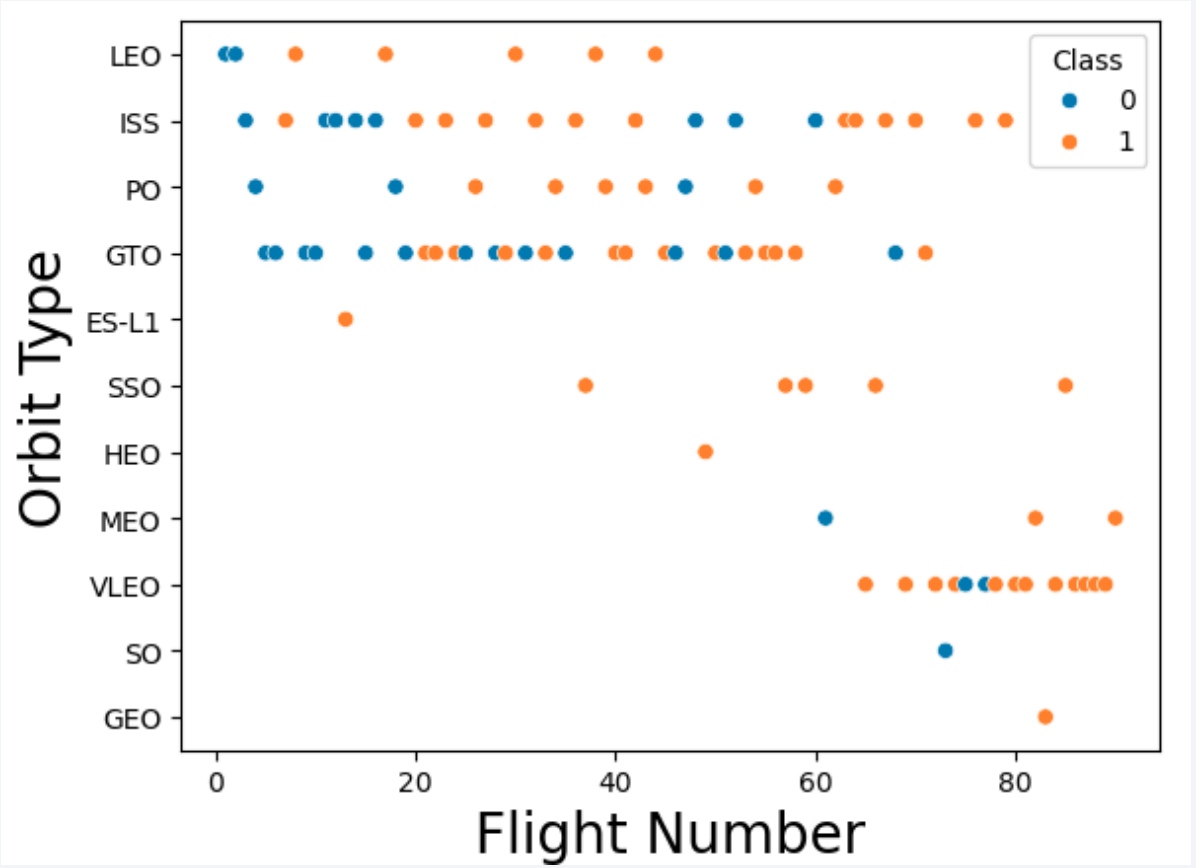
Success Rate vs. Orbit Type

- 100% Success
  - ES-L1, GEO, HEO, and SSO
- 50%-80% Success
  - GTO, ISS, LEO, MEO, PO
- 0% Success
  - SO



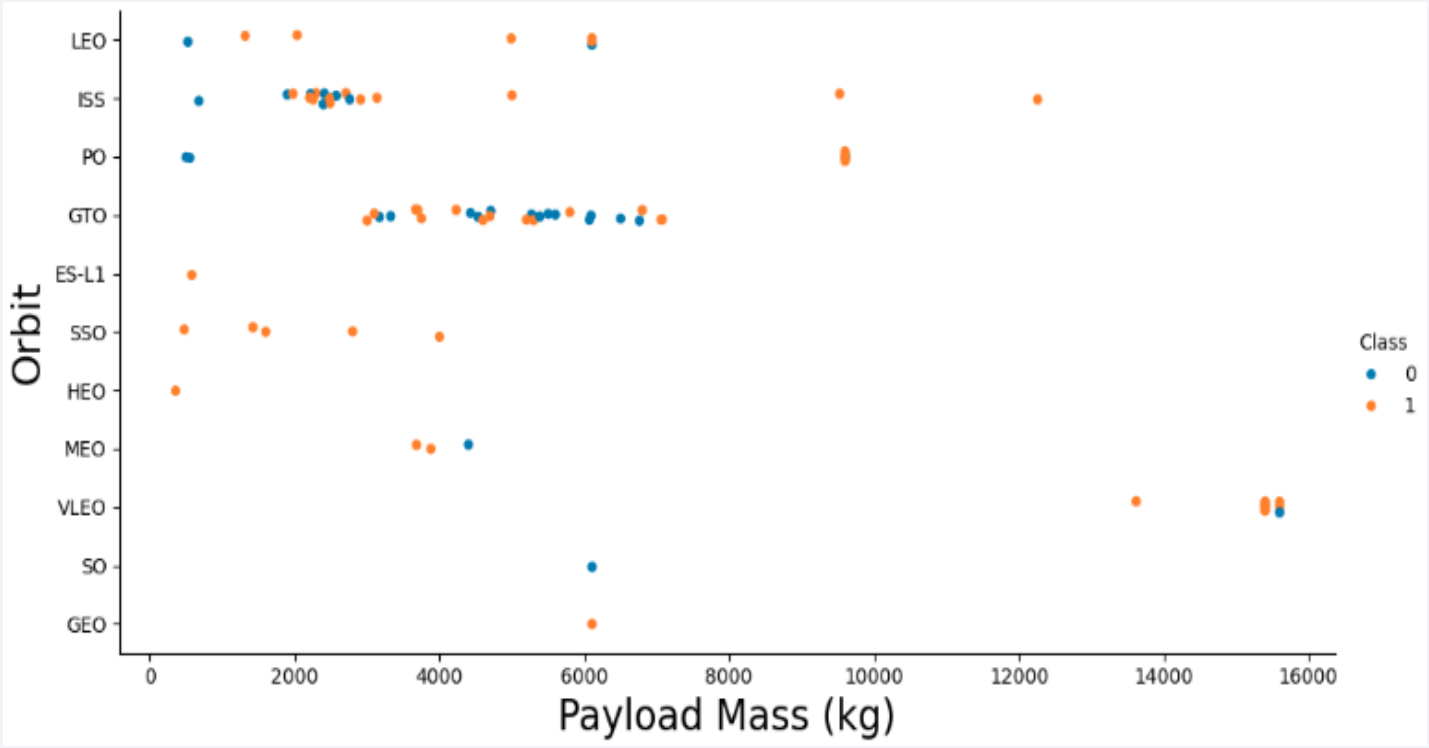
Flight Number vs. Orbit Type

- The greater the flight number on each orbit, the greater the success rate
- GTO orbit doesn't follow the pattern



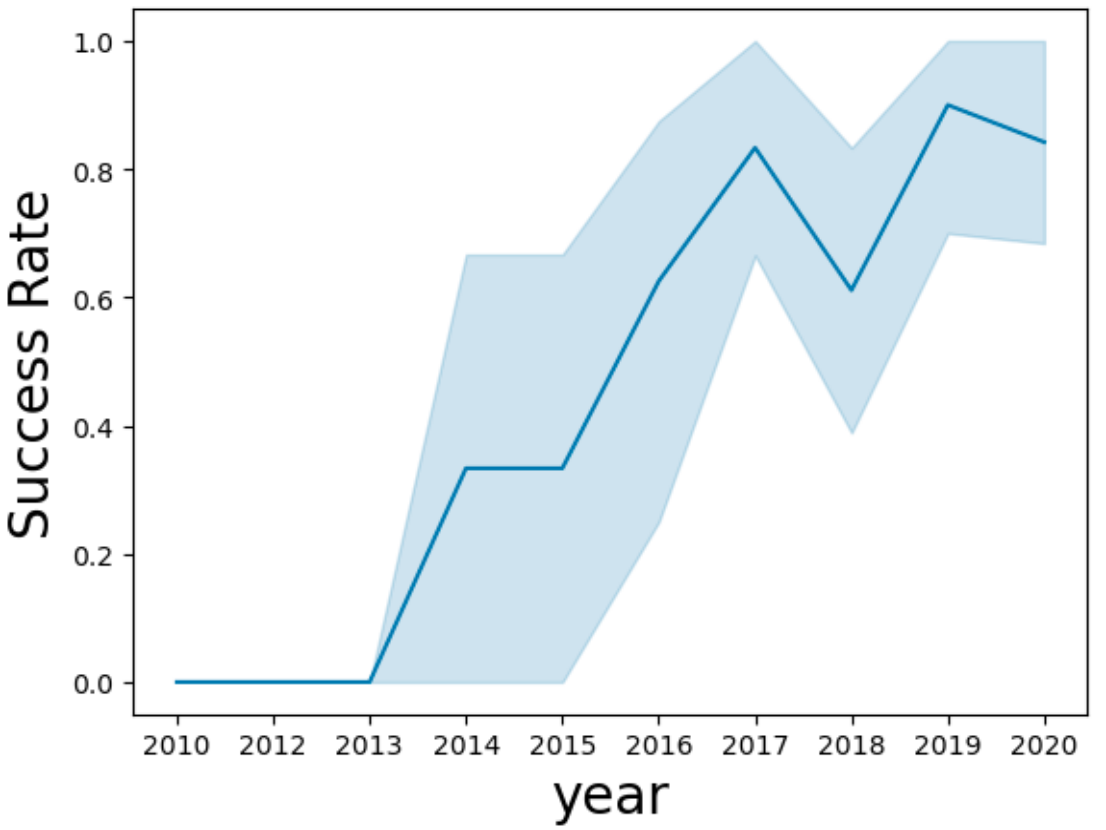
Payload vs. Orbit Type

- Heavier payload has a positive impact on LEO, ISS, and PO orbits
- The GTO orbit seems to have mixed success



Launch Success Yearly Trend

- This graph shows that success rate improved between 2013-2017 and 2018-2019 and decreased between 2017-2018 and 2019-2020
- Overall, we see that the success rate had improved since 2013.



All Launch Site Names

We used DISTINCT to display launch site names

```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
```

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

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

We were further able to filter launch sites

```
%sql SELECT LAUNCH_SITE FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

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

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

First Successful Ground Landing Date

- By using SELECT MIN(DATE) we were able to find the first successful landing

```
%sql SELECT MIN (DATE) AS "First Successful Landing" FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Succe
```

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

First Successful Landing
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- We were able to ist the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000 by filtering using 'PAYLOAD_MASS_KG between 4000 AND 6000'

```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (drone ship)' AND PAYL
```

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

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

Total Number of Successful and Failure Mission Outcomes

- By using COUNT(*) and GROUP BY we ffound the total number of successful and failed mission outcomes

```
%sql SELECT MISSION_OUTCOME, COUNT(*) as total_number FROM SPACEXTBL GROUP BY MISSION_OUTCOME;
```

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

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

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass
- Present your query result with a short explanation here

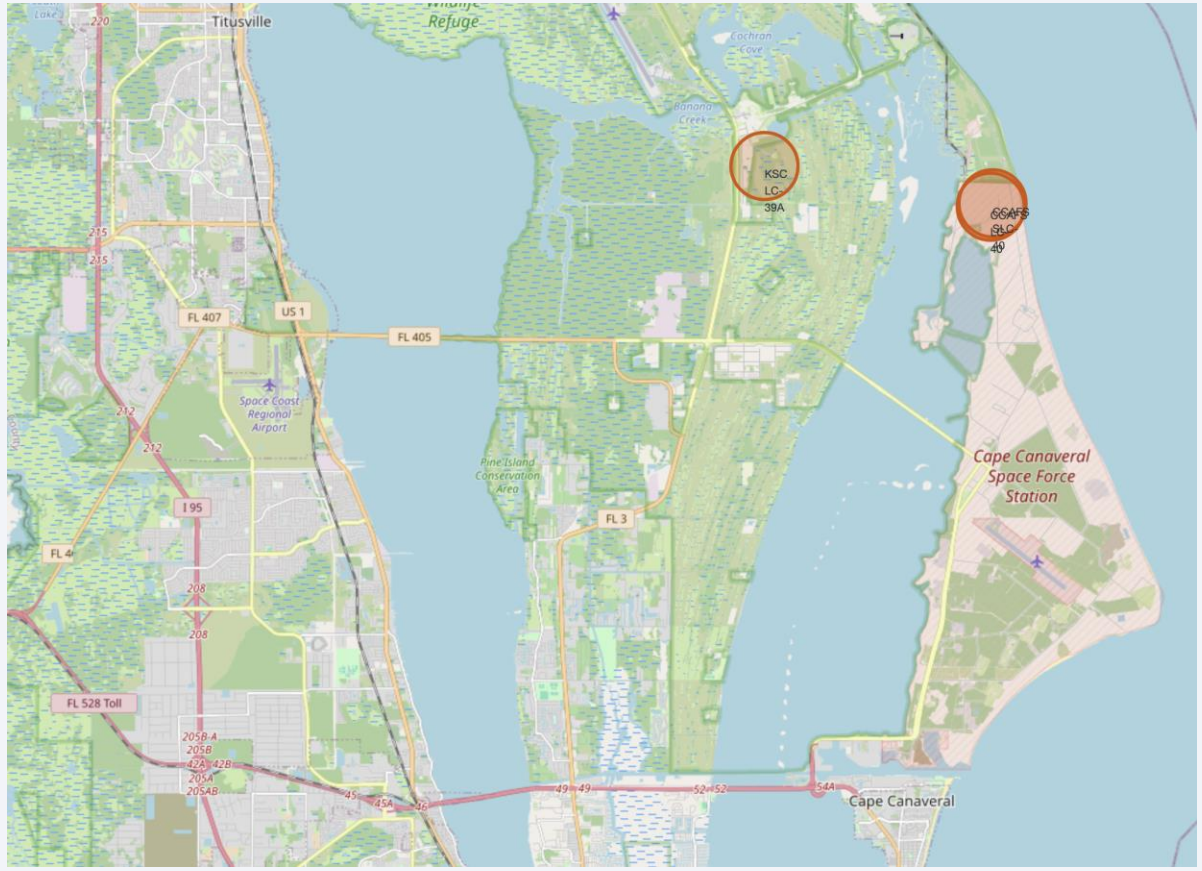

Section 3
Launch Sites
Proximities Analysis

Folium Map – Launch Sites

- This folium map shows all launch sites' location markers

Proximity Map

- This map shows the nearby railway and highway to these launch sites.



Section 5

Predictive Analysis (Classification)

Model Performance

Logistic Regression Model

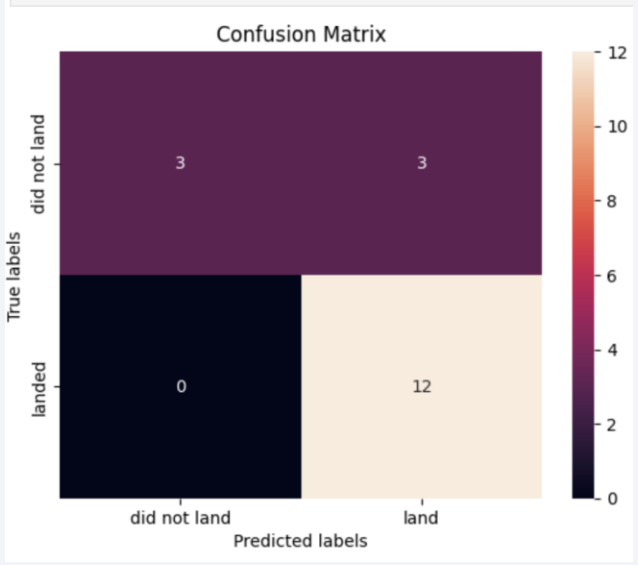
```
parameters ={'C':[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}
```

```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()

logreg_cv = GridSearchCV(estimator=lr, cv=10, param_grid=parameters).fit(X_train, Y_train)
```

We output the `GridSearchCV` object for logistic regression. We display the best parameters using the data attribute `best_params_` and 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_)
```

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



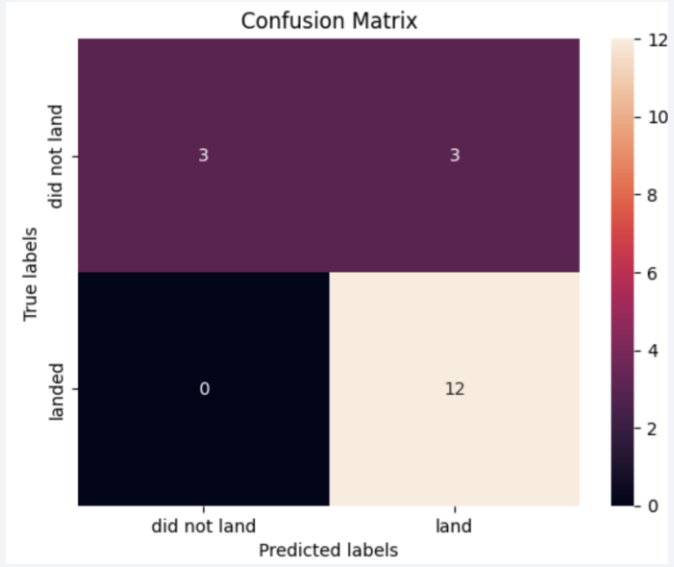
SVM

```
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
```

```
svm_cv = GridSearchCV(estimator=svm, cv=10, param_grid=parameters).fit(X_train, Y_train)
```

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

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



KNN

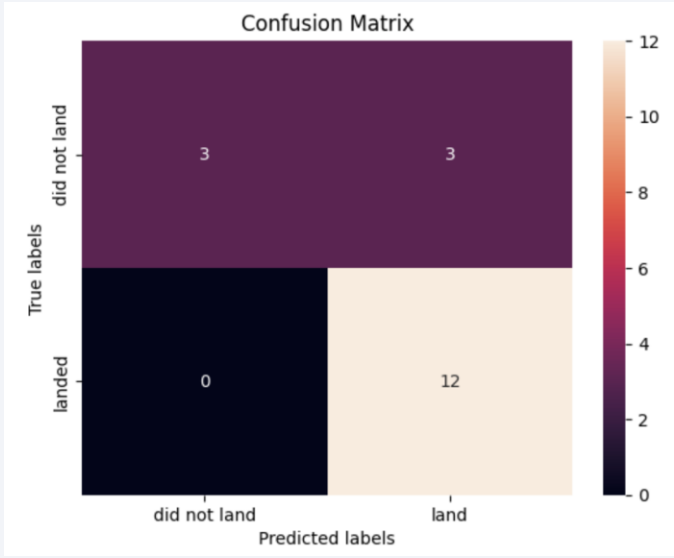
```
parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
              'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
              'p': [1,2]}

KNN = KNeighborsClassifier()
```

```
gscv = GridSearchCV(KNN,parameters,scoring='accuracy',cv=10)
knn_cv = gscv.fit(X_train,Y_train)
```

```
/lib/python3.11/site-packages/threadpoolctl.py:1019: RuntimeWarning: libc not found. The ct
11 is maybe too old for this OS.
  warnings.warn(
```

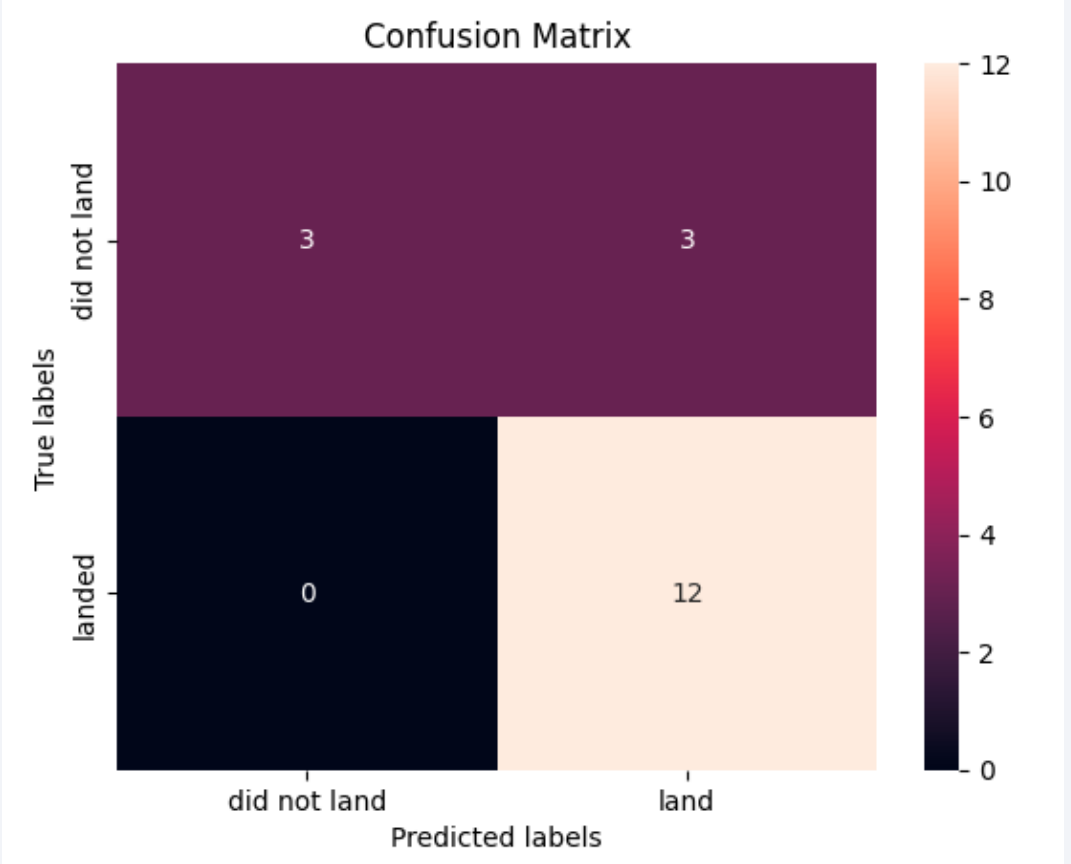
```
print("tuned hpyerparameters :(best parameters) ",knn_cv.best_params_)
print("accuracy :",knn_cv.best_score_)
```

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



Confusion Matrix

- The confusion matrix displays that the classifier can distinguish between the different classes
- One main problem is the false positive as shown by the model's incorrect prediction of the first stage booster landing 3 out of 18 samples in the test set



Classification Accuracy- Best Model

- The DecisionTree model had the highest classification accuracy with a score of 0.875.

```
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 0.875
Best params is : {'criterion': 'entropy', 'max_depth': 16, 'max_features': 'sqrt', 'min_samples_lea
f': 2, 'min_samples_split': 10, 'splitter': 'random'}
```

Conclusions

- 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

Thank you!