Winning Space Race with Data Science

Nicolás Arrioja Landa Cosio
June 1st, 2022

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collecting with API
  - Data Collecting with Web Scraping
  - Data Wrangling
  - EDA with SQL
  - EDA with Visualization
  - Interactive Visual Analytics with Folium
  - Interactive Visual Analytics with Dashboard
  - Machine Learning Prediction (Classification)
- Summary of all results
  - EDA results
  - Using a dashboard for interactive analytics
  - Predictions using the ML model

Introduction

- Project background and context
  - Space exploration and launching satellites in orbit very expensive. SpaceX developed several systems like Falcon9 that allow rocket launches to be cheaper by reusing the first stage of the rocket. It is really important to determine the cost of the launch, that is the reason to create a ML model that will predict if the first stage will land or not. The goal of the project is to have a ML model that will do predictions about the landing of the first stage.
- Problems you want to find answers
  - Find out the most important factors that affect the land of the rocket
  - Determine the success rate of the landing and the features that determine it.
  - Identify the operating conditions for a successful landing

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - The data was collected from Wikipedia using web scraping. The SpaceX API is also used
- Perform data wrangling
  - The categorical features were transformed using One-hot encoding
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Several classification models were used to find the one with the best accuracy. The best hyperparameters were found using Grid Search

Data Collection

To collect the data with SpaceX API we use the get request

We use the json() method to have the reponse in json format

After it we convert it to a pandas dataframe by using the json_normalize() method

Then the data wrangling is done: check for missing values, fill missing values, check for duplicates, etc.

The other collection of data was done by using BeautifulSoup and performing web scraping on Wikipedia

The Falcon 9 launch records were scraped as an HTML table

The table was parsed and converted into a pandas dataframe

Data Collection – SpaceX API

- 1. Get request for data using API
- 2. Decode the response as json
- 3. Convert to dataframe
- 4. Fill the missing values

https://github.com/nicosiocursos/IBM_capstoneProject/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

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

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

In [11]:
```
# Use json_normalize meethod to convert the json result into a dataframe
data=pd.json_normalize(response.json())
```

In [27]:
```
# Calculate the mean value of PayloadMass column
meanPayload=data_falcon9['PayloadMass'].mean()
print(meanPayload)
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass']=data_falcon9['PayloadMass'].replace(np.nan,meanPayload)

print(data_falcon9.isnull().sum())
```

Data Collection - Scraping

- 1. Apply HTTP Get method
- 2. Create a beutifulSoup object and get the title
- 3. Extract column names
- 4. Parse the HTML table to create a dataframe

https://github.com/nicosiocursos/IBM_capstoneProject/blob/main/jupyter-labs-webscraping.ipynb

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

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

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

```
# Use soup.title attribute
print(soup.title)
```

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

```
# 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')
```

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

Continue on the next slide

Data Collection - Scraping

- 1. Apply HTTP Get method
- 2. Create a beutifulSoup object and get the title
- 3. Extract column names
- 4. Parse the HTML table to create a dataframe

https://github.com/nicosiocursos/IBM_capstoneProject/blob/mair jupyter-labs-webscraping.ipynb

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            launch_dict['Flight No.'].append(flight_number)
            #print(flight_number)


            datatimelist=date_time(row[0])

            # Date value
            # TODO: Append the date into launch_dict with key `Date`
```

Data Wrangling

1. The number of launches on each site is calculated
2. The number of occurrence of each orbit is calculated
3. The number and occurrence of mission outcome per orbit type is calculated
4. A label outcome was created from the Outcome column

https://github.com/nicosiocursos/IBM_capstoneProject/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

```
# 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
```

```
# 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_class = 0 if bad_outcome
# landing_class = 1 otherwise

landing_class=[]
for outcome in df['Outcome']:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)

#landing_class
```

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

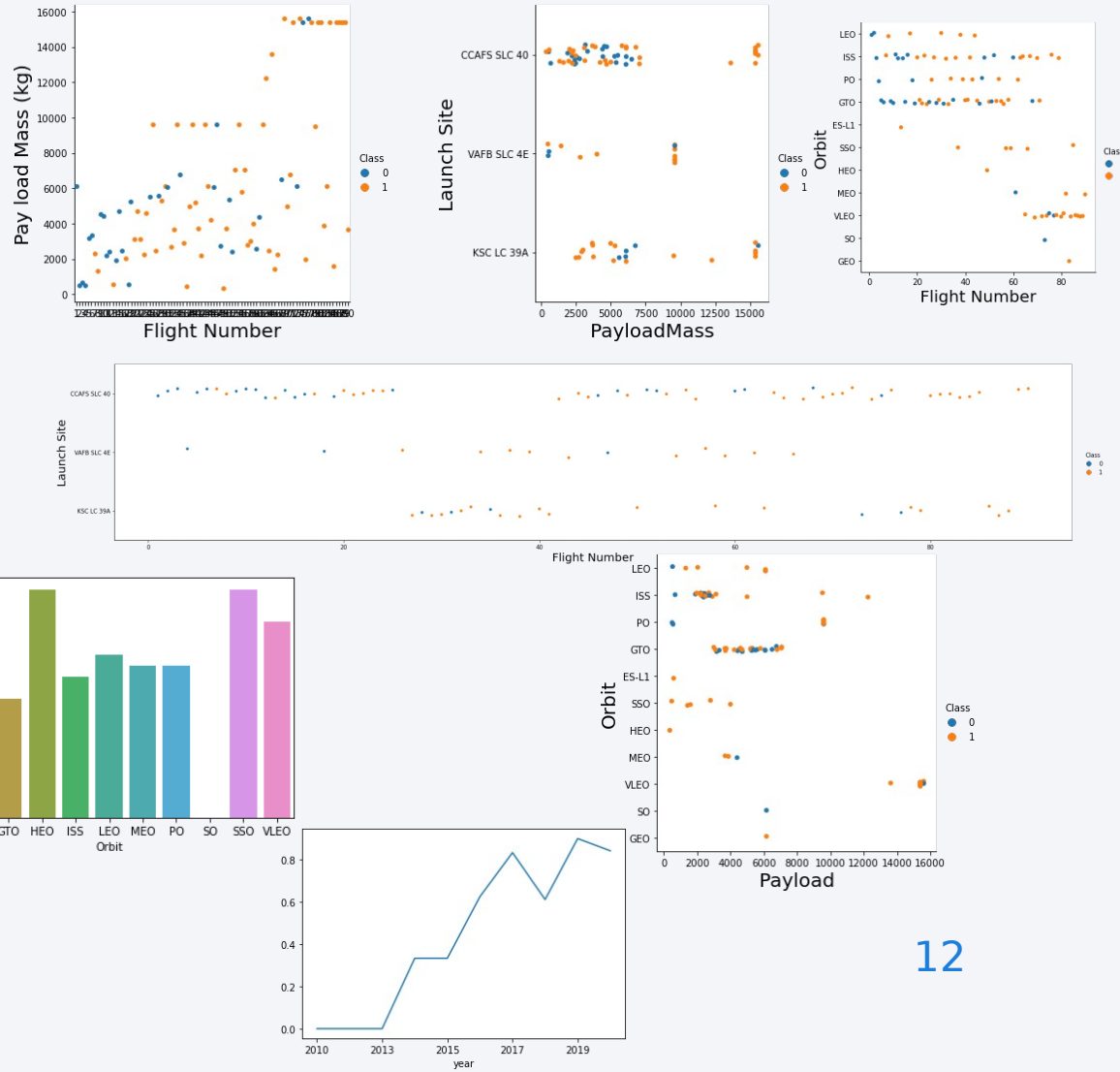
EDA with Data Visualization

Catplots were used because they show the frequencies of the categories of one or several variables and allow us to view the behavior of the classes

The bar chart was also used to understand the relationship between the success rate of each orbit type

The line plot is helpful to see the relationship between two variables , in this case the year and the success rate

https://github.com/nicosiocursos/IBM_capstoneProject/blob/main/jupyter-labs-eda-dataviz.ipynb



EDA with SQL

- The SQL queries that were used are:
  - Display the names of the unique launch sites

    ```
    SELECT DISTINCT LAUNCH_SITE
    FROM SPACEXTBL;
    ```
  - Display 5 records with launch site beginning with CCA

    ```
    SELECT *
    FROM SPACEXTBL
     WHERE LAUNCH_SITE LIKE 'CCA%'
     LIMIT 5;
    ```
  - Display the total payload launched by NASA (CRS)

    ```
    SELECT SUM(PAYLOAD_MASS__KG_)
     FROM SPACEXTBL;
    WHERE Customer LIKE 'NASA (CRS)';
    ```
  - Display the average payload by booster F9 v1.1

    ```
    SELECT AVG(PAYLOAD_MASS__KG_)
    FROM SPACEXTBL
    WHERE BoosterVersion = 'F9 v1.1';
    ```

Continue on the next slide

https://github.com/nicosiocursos/IBM_capstoneProject/blob/main/jupyter-labs-eda-sql-edx.ipynb

EDA with SQL

- The SQL queries that were used are:
  - List the date when the first successful landing was achieved

    ```
    SELECT MIN(Date)
    FROM SPACEXTBL
    WHERE Landing__Outcome LIKE 'Success (ground pad)';
    ```
  - List the names of booster that have success with payload mass between 4000 and 6000

    ```
    SELECT BoosterVersion
    FROM SPACEXTBL
    WHERE LandingOutcome = 'Success (drone ship)'
        AND PAYLOAD_MASS__KG_ > 4000
        AND PAYLOAD_MASS__KG_ < 6000;
    ```
  - List the total number of successful outocomes

    ```
    SELECT COUNT(MissionOutcome) AS SuccessOutcome
    FROM SPACEXTBL
    WHERE MissionOutcome LIKE 'Success%';
    ```
  - List the total number of failure outcomes

    ```
    SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER
    FROM SPACEXTBL
    GROUP BY MISSION_OUTCOME;
    ```

https://github.com/nicosiocursos/IBM_capstoneProject/blob/main/jupyter-labs-eda-sql-edx.ipynb

Continue on the next slide

EDA with SQL

- The SQL queries that were used are:
  - List the names of the booster version that carried maximun payload

```
SELECT DISTINCT BOOSTER_VERSION
FROM SPACEXTBL
WHERE PAYLOAD_MASS__KG_ = (
    SELECT MAX(PAYLOAD_MASS__KG_)
    FROM SPACEXTBL);
```

  - List the failed landing outcome, booster versions and launch sites for 2015

```
SELECT LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE
FROM SPACEXTBL
WHERE Landing__Outcome = 'Failure (drone ship)'
    AND YEAR(DATE) = 2015;
```

  - Rank the count of landing outcomes between 2010-06-04 and 2017-03-20

```
SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) AS TOTAL_NUMBER
FROM SPACEXTBL
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY LANDING__OUTCOME
ORDER BY TOTAL_NUMBER DESC;
```

https://github.com/nicosiocursos/IBM_capstoneProject/blob/main/jupyter-labs-eda-sql-edx.ipynb

Build an Interactive Map with Folium

The folium circle is used to highlight an area of interest on a specific coordinates. It was used for the NASA Johnson Space Center. It is possible to add a descriptive text label.

Those markers can help us identify success/failed launches, and present information about them. The value of 0 for failure and 1 for success.

It is possible to group the launches and present them in a better way. Also colors can be used to identify the success rate.

The lines can be used to show the distance between two places. Fpr example launch sites and railroads, highways, coastlines and cities.

With the mouse it is possible to find out the coordinates of a point in the map.

https://github.com/nicosiocursos/IBM_capstoneProject/blob/main/lab_jupyter_launch_site_location%20(2).ipynb

Build a Dashboard with Plotly Dash

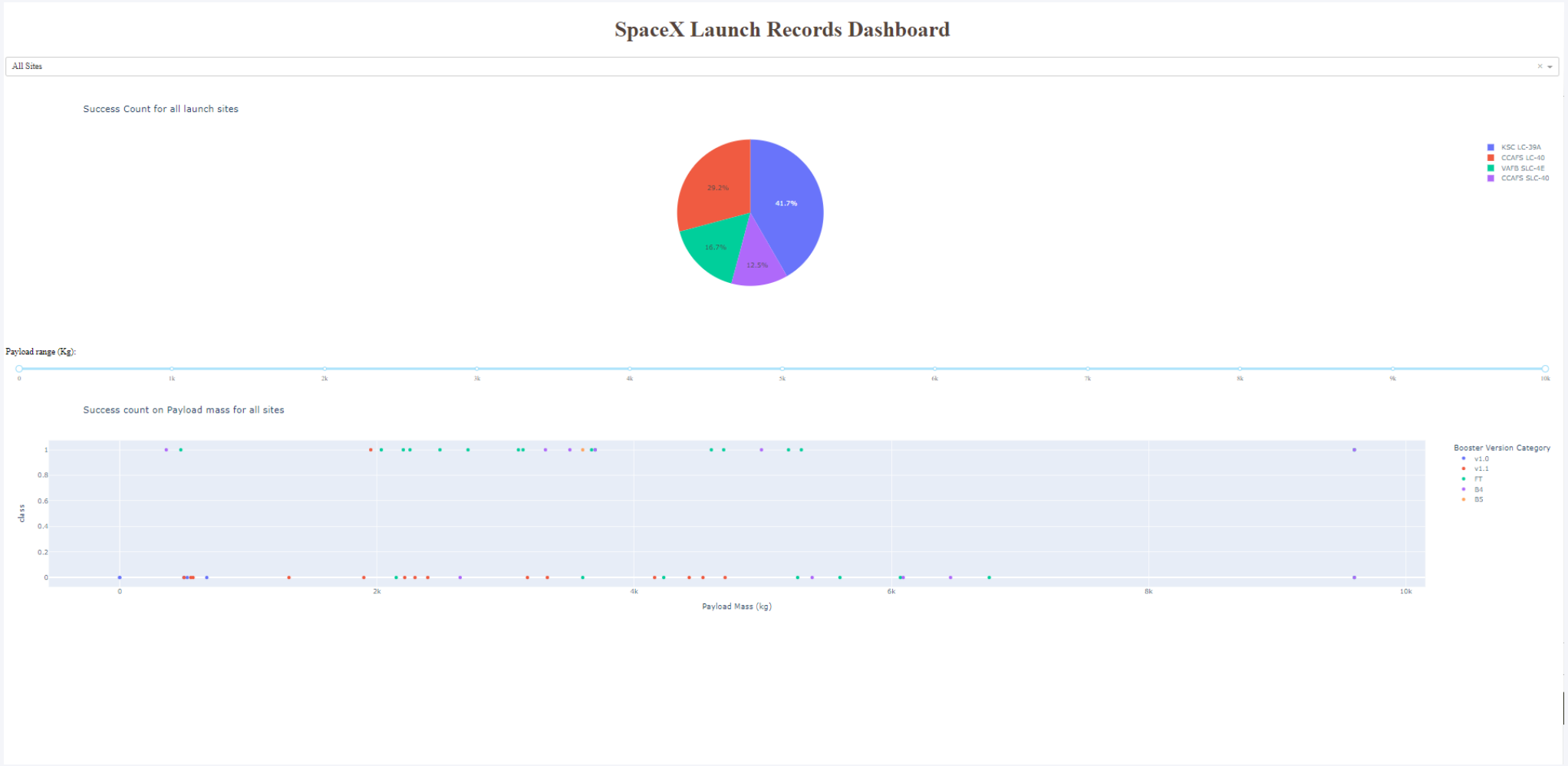
The dashboard was built with Plotly dash

A pie chart was created that shows the percentage of launches at the different launch sites

A scatter plot show the relationship between outcome and Payload Mass (KG) for the different booster versions

https://github.com/nicosiocursos/IBM_capstoneProject/blob/main/spacex_dash_app.py

Build a Dashboard with Plotly Dash



Predictive Analysis (Classification)

First we got the Y array to have the information related to the failure or success

The X dataframe is standarized using StandarScaler

The dataset is split into training and testing data

Several classifiers are created

- Logistic Regression
- Support Vector Machine
- Decision Tree Classifier
- K Nearest Neighbors

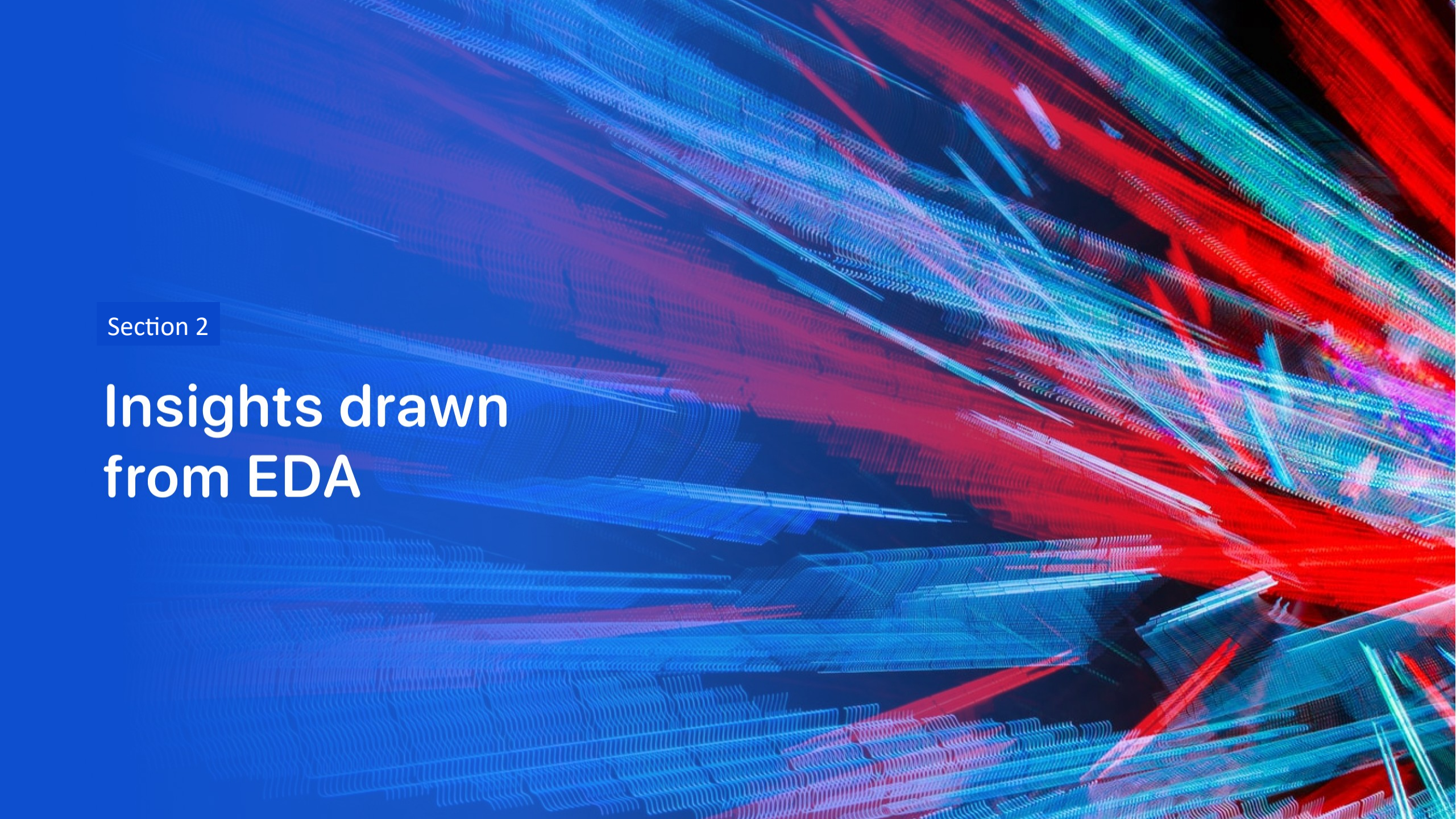
Each classifier is optimized using GridSearchCV

The best performing classifier is identified

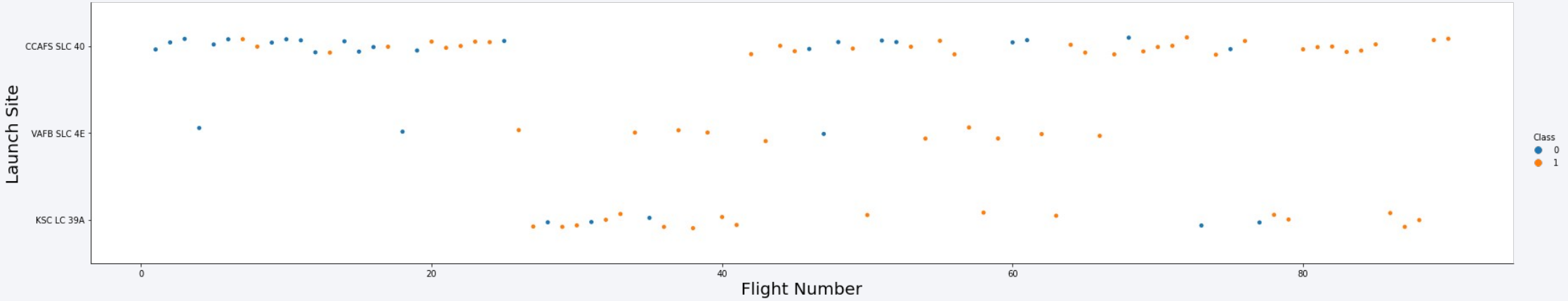
https://github.com/nicosiocursos/IBM_capstoneProject/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.ipynb

Results

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


Section 2
Insights drawn from EDA

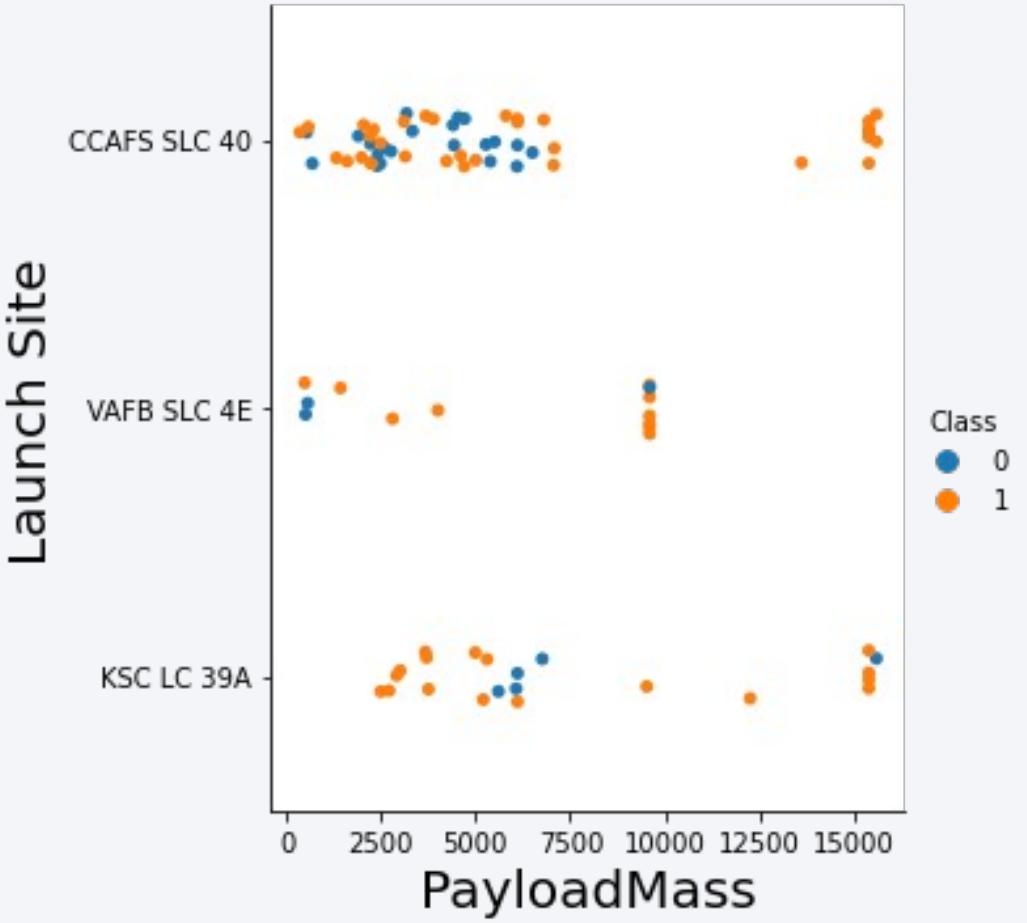
Flight Number vs. Launch Site



- We can see that as the flight number increases the success rate is higher
- As a launch site has more launches the success rate is higher

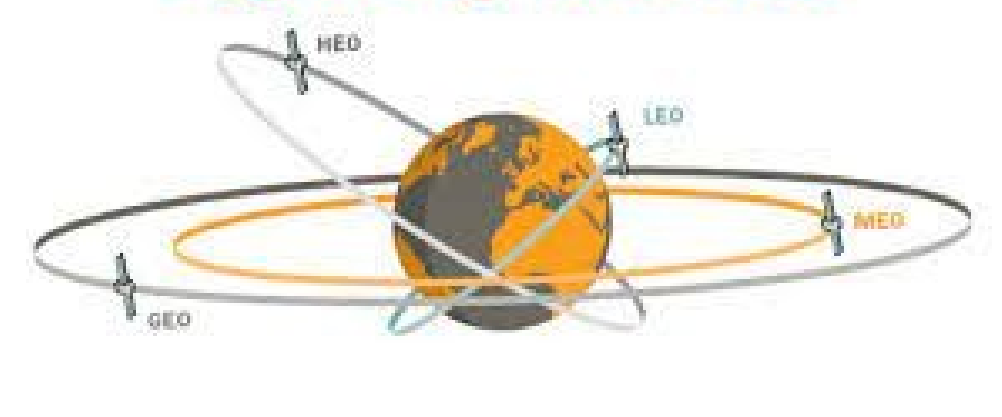
Payload vs. Launch Site

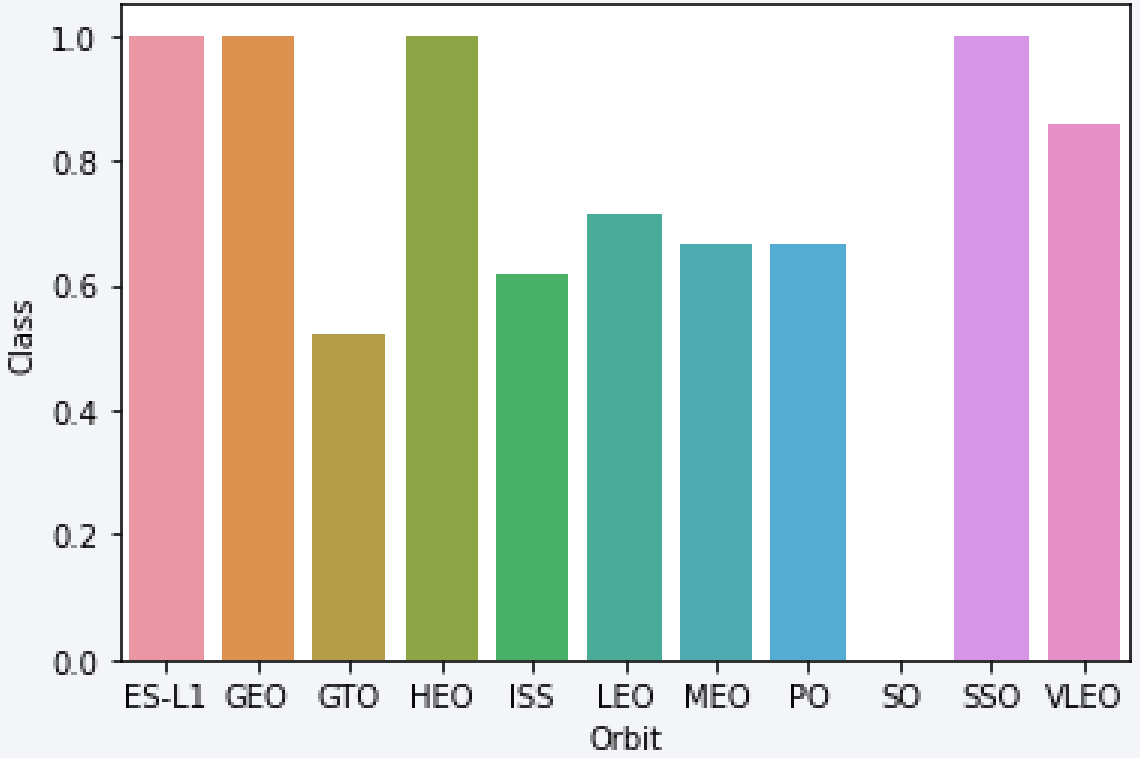
- We can see that as the playload mass increases the success rate also increases
- The launch site with most failures is CCAFS SLC 40



Success Rate vs. Orbit Type

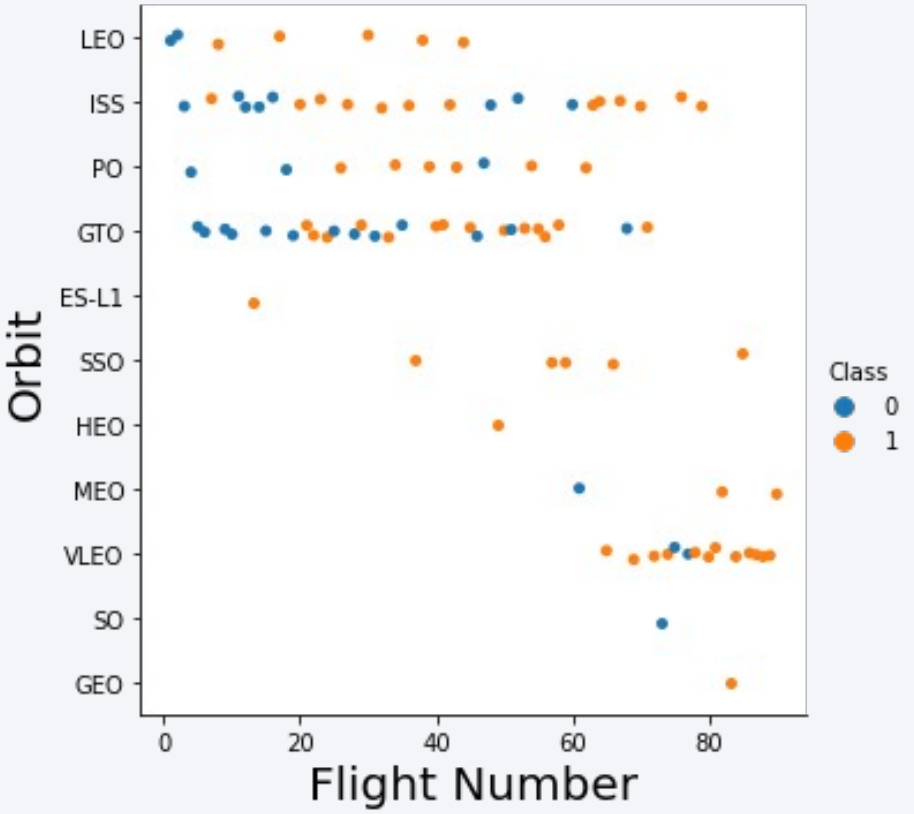
- The orbits with the higher success rate are: ES-L1, GEO, HEO, SSO,VLEO
-





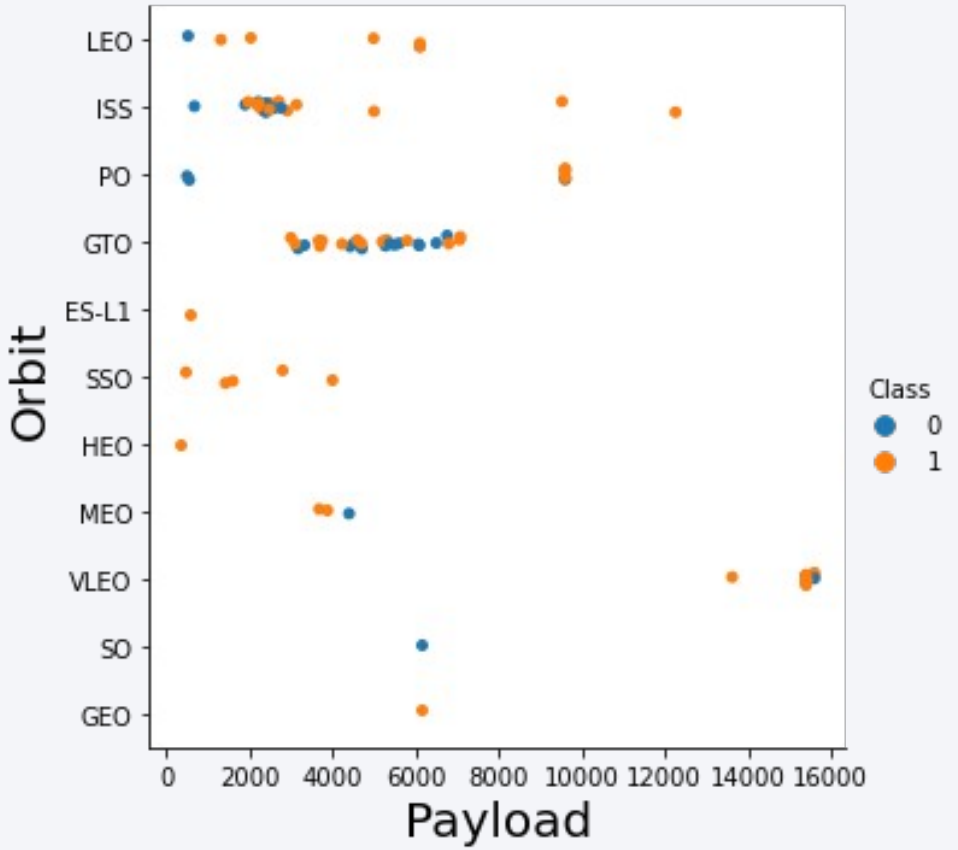
Flight Number vs. Orbit Type

- We can see that SSO orbit has a high success rate
- LEO also has a good success rate with some failure at the beginning of the launches
- As flight number increases it seems that the success rate also increases, except for GTO



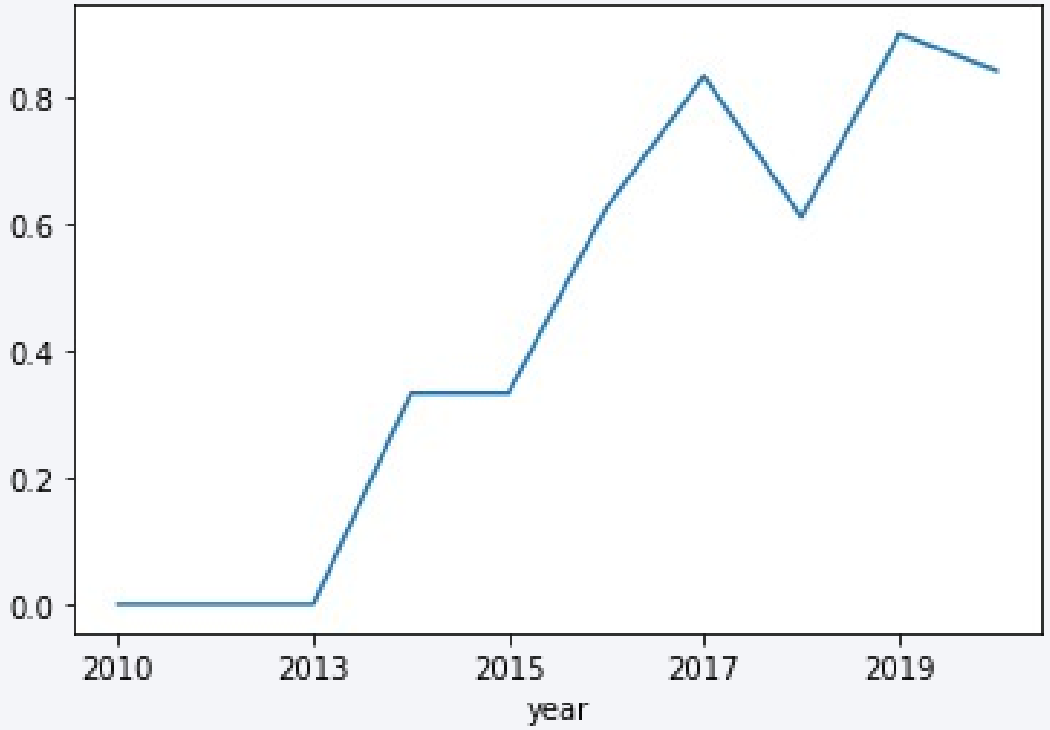
Payload vs. Orbit Type

- The higher the payload the higher the the success rate, except for GTO
- GTO have mixed results no matter the payload
- Some orbits are very successful with low payloads like ES-L1, SSO, HEO



Launch Success Yearly Trend

- From 2010 to 2013 there were no success
- Since 2013 the success rate is increasing



All Launch Site Names

To find the unique launch site names we use DISTINCT

There are four different launch sites

```
%%sql
SELECT DISTINCT LAUNCH_SITE
FROM SPACEXTBL;
```

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

Launch Site Names Begin with 'CCA

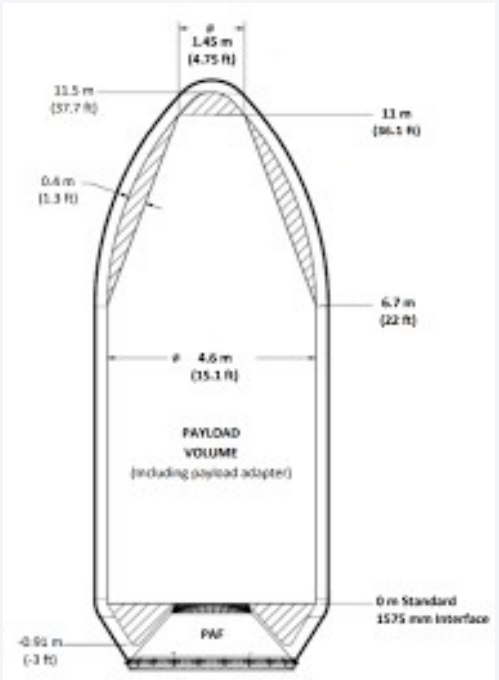
We select those that have CCA on the LAUNCH_SITE


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

launch_site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Total Payload Mass

The total payload mass calculated for NASA (CRS) is 45596




```
%%sql
SELECT SUM(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE Customer = 'NASA (CRS)';
```

```
Out[11]:  1
          45596
```

Average Payload Mass by F9 v1.1

The average payload mass that was carried by booster with version F9 v1.1 is 2928.4

```
%%sql
SELECT AVG(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE Booster_Version LIKE 'F9 v1.1';
```

Out[13]:

1
2928.4

First Successful Ground Landing Date

The date found for the first successful landing in ground pad was 2015-12-22

```
%%sql
SELECT MIN(Date)
FROM SPACEXTBL
WHERE Landing__Outcome = 'Success (ground pad)';
```

Out[15]:

1
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

There are several boosters with payloads between 4000 and 600 that successfully landed on a drone ship

```
%%sql
SELECT BOOSTER_VERSION
FROM SPACEXTBL
WHERE LANDING__OUTCOME = 'Success (drone ship)'
    AND 4000 < PAYLOAD_MASS__KG_
    AND PAYLOAD_MASS_KG_ < 6000;
```

Out[17]:

booster_version
F9 FT B1021.1
F9 FT B1023.1
F9 FT B1029.2
F9 FT B1038.1
F9 B4 B1042.1
F9 B4 B1045.1
F9 B5 B1046.1

Total Number of Successful and Failure Mission Outcomes

There is one Failure and a total of 100 Successes. 99 are Success and one is a Success (payload status unclear)

It could be possible to use the wildcard % to filter better the different kinds of successes using a WHERE… LIKE

```
%%sql
SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER
FROM SPACEXTBL
GROUP BY MISSION_OUTCOME;
```

Out[19]:

mission_outcome	total_number
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

Boosters Carried Maximum Payload

There are several boosters that have carried the maximum payload mass.

```
%%sql
SELECT DISTINCT BOOSTER_VERSION
FROM SPACEXTBL
WHERE PAYLOAD_MASS__KG_ = (
    SELECT MAX(PAYLOAD_MASS__KG_)
    FROM SPACEXTBL);
```

Out[21]:

booster_version
F9 B5 B1048.4
F9 B5 B1048.5
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1049.7
F9 B5 B1051.3
F9 B5 B1051.4
F9 B5 B1051.6
F9 B5 B1056.4
F9 B5 B1058.3
F9 B5 B1060.2
F9 B5 B1060.3

2015 Launch Records

We list the failed landing outcomes, the booster version and the launch site for the year 2015

```
%%sql
SELECT LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE
FROM SPACEXTBL
WHERE Landing__Outcome = 'Failure (drone ship)'
    AND YEAR(DATE) = 2015;
```

Out[23]:

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

It is necessary to select the landing outcomes and then use COUNT to have the total number. It is important to filter them between the requested dates from 2010-06-04 to 2017-03-20.

It is necessary to group by landing outcome and then order by the total number.

```
%%sql
SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) AS TOTAL_NUMBER
FROM SPACEXTBL
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY LANDING__OUTCOME
ORDER BY TOTAL_NUMBER DESC
```

Out[25]:

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

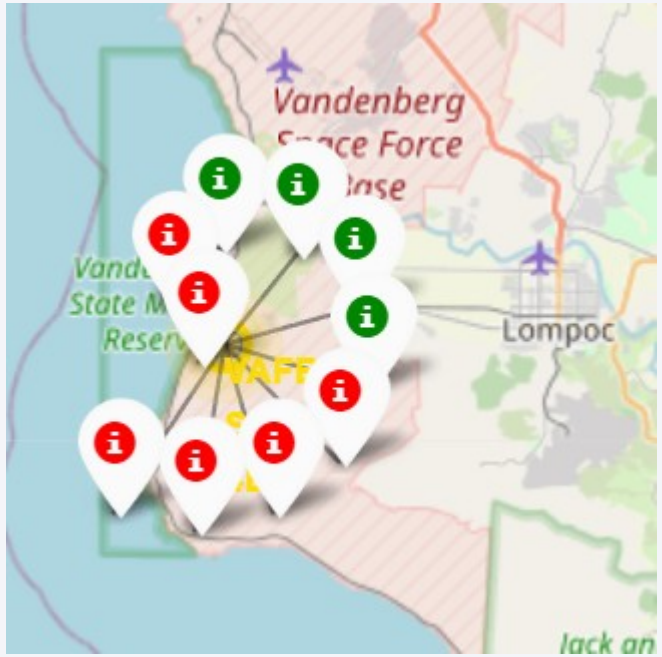

Section 3
Launch Sites
Proximities Analysis

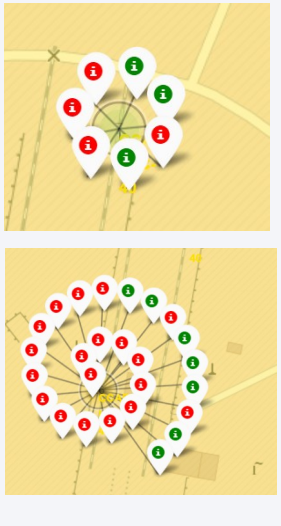
Launch sites marker on a global map

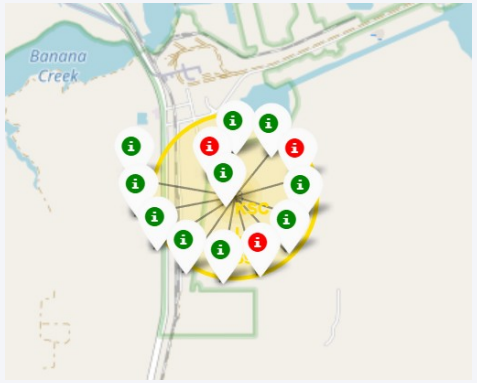
All the launch sites are in the United States near the coast. In California and Florida.

Markers at the launch site with color markers

Green means success, red means failure. At the left the site in California, at the center and right the sites in Florida

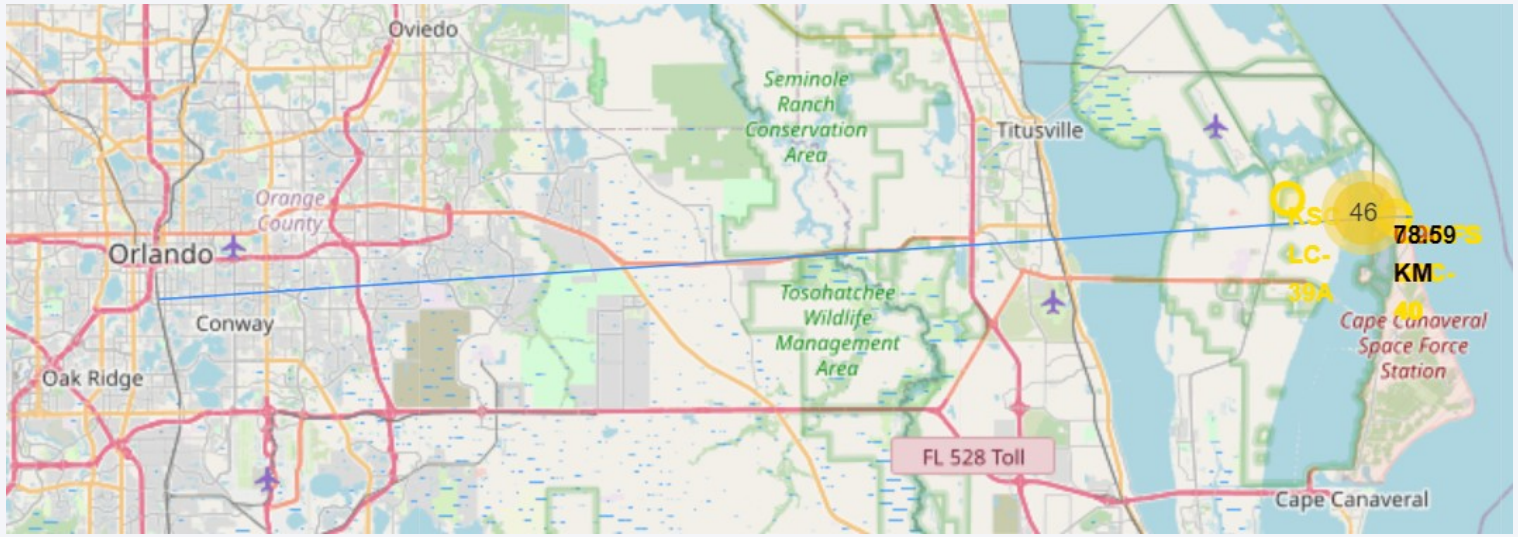


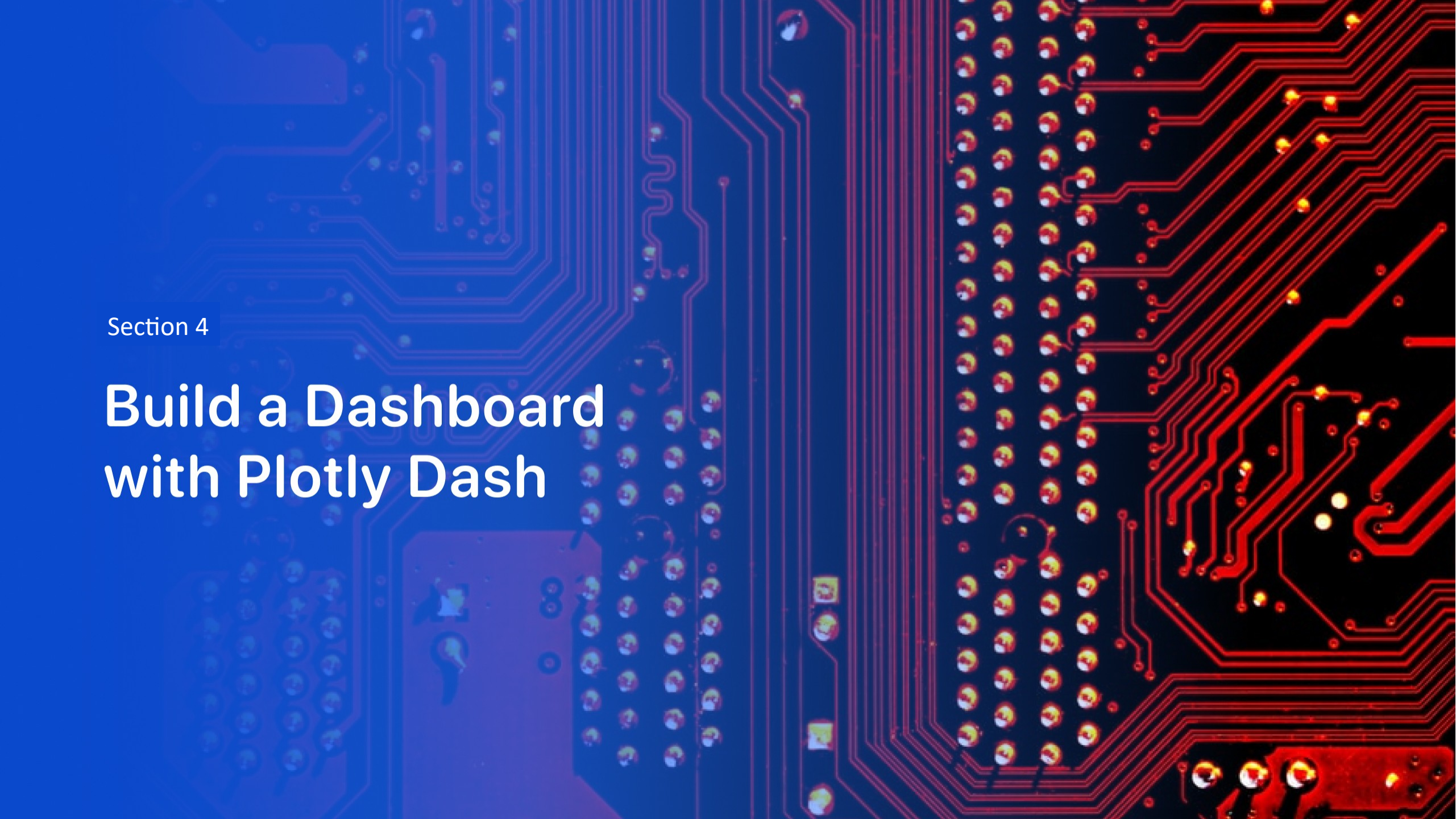




Launch site distance to landmarks

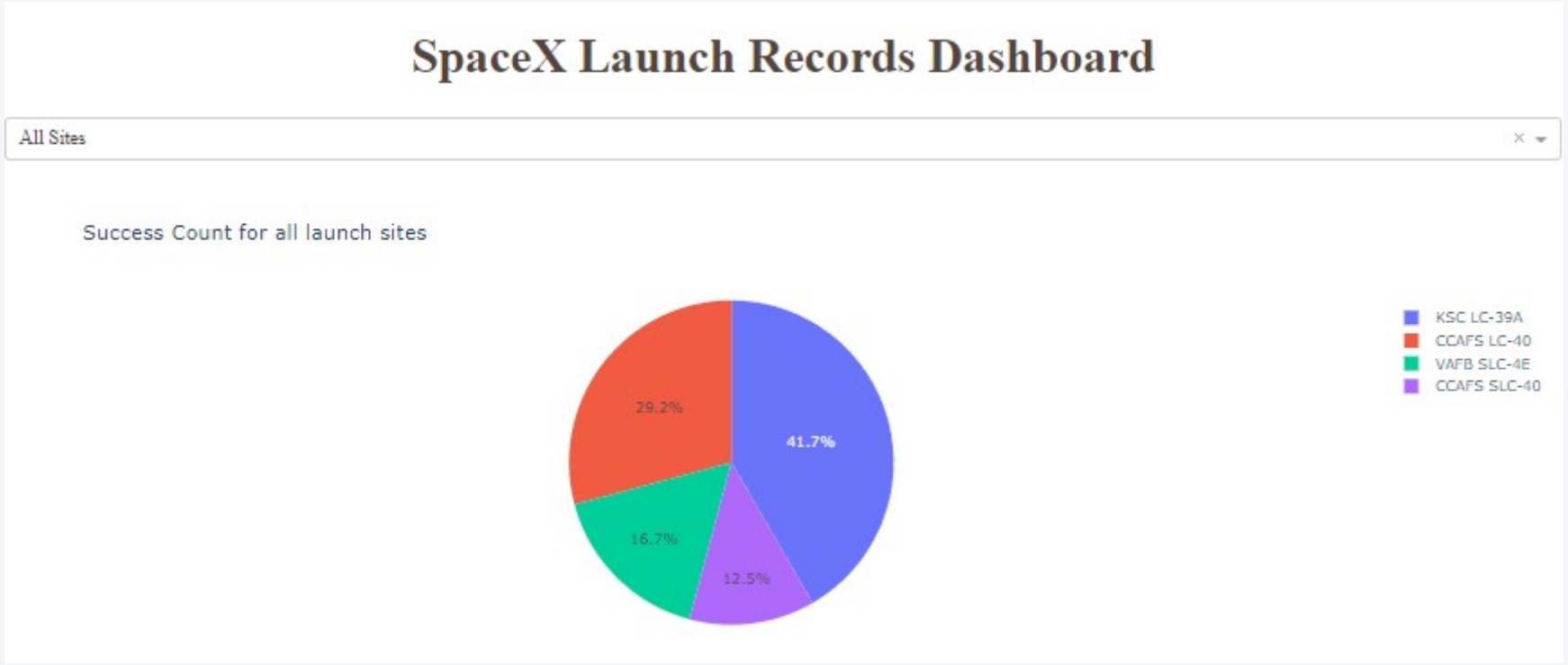
It is possible to find the distance to different landmarks, for example the coastline or the nearest city




Section 4
Build a Dashboard with Plotly Dash

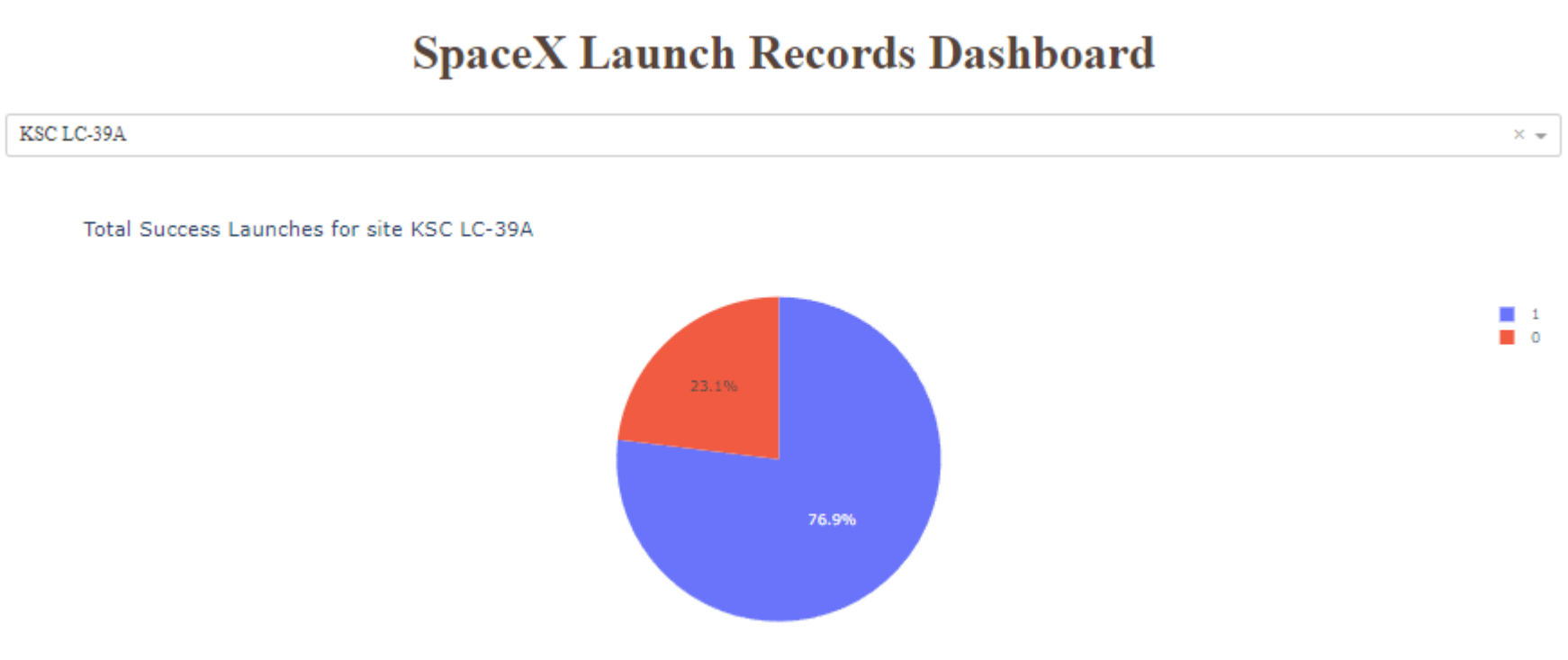
Pie chart of the success achieved by launch site

The most successful launch site is KSC LC 39-A with 41.7%



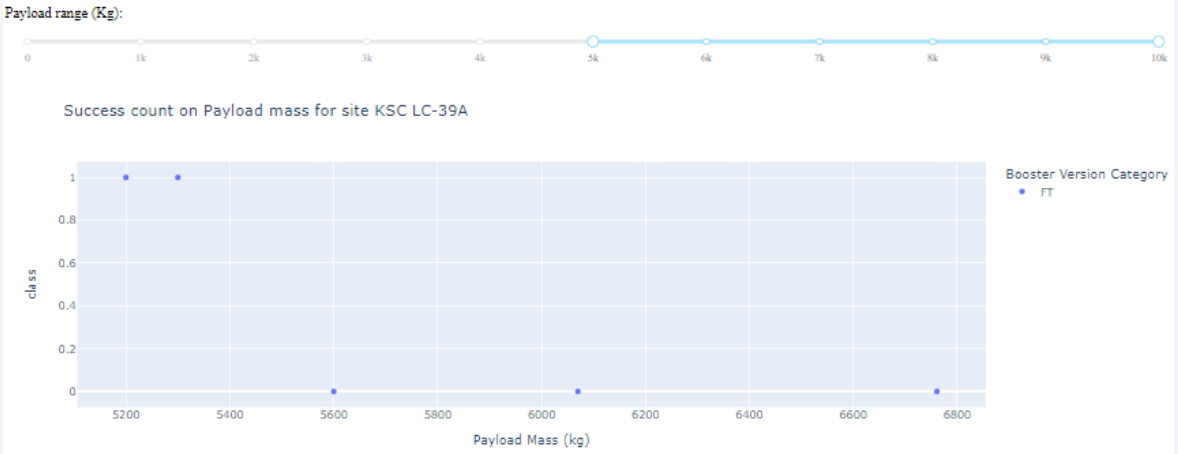
Pie chart of most successful launch site

KSC LC-39A has a 76.9% of success and a 23.1% failure rate



Scatter plots of Payload vs Launch Outcome

Launches with payloads up to 5k have a better success rate than those from 5k to 10k



Section 5

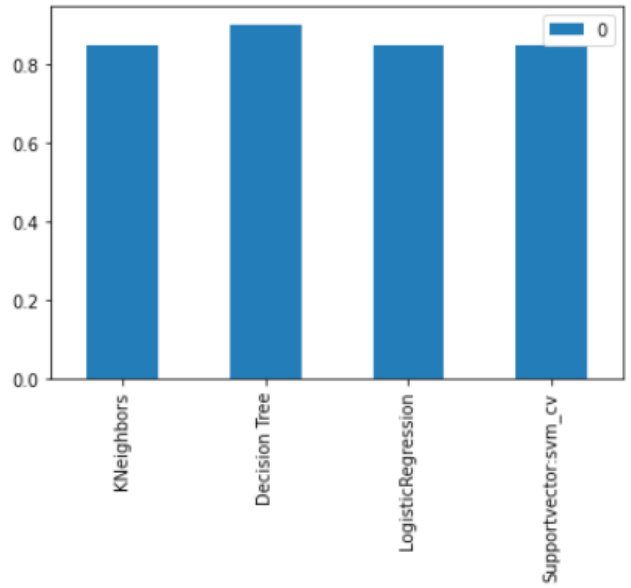
Predictive Analysis (Classification)

Classification Accuracy

- The model with the highest accuracy is DecisionTree with a value of 0.9

```
scores={'KNeighbors':knn_cv.best_score_, 'Decision Tree':tree_cv.best_score_,
       'LogisticRegression':logreg_cv.best_score_,'Supportvector:svm_cv':svm_cv.best_score_}
scoresdf=pd.DataFrame(pd.Series(scores))
print(scores)
scoresdf.head()
scoresdf.plot.bar()
```

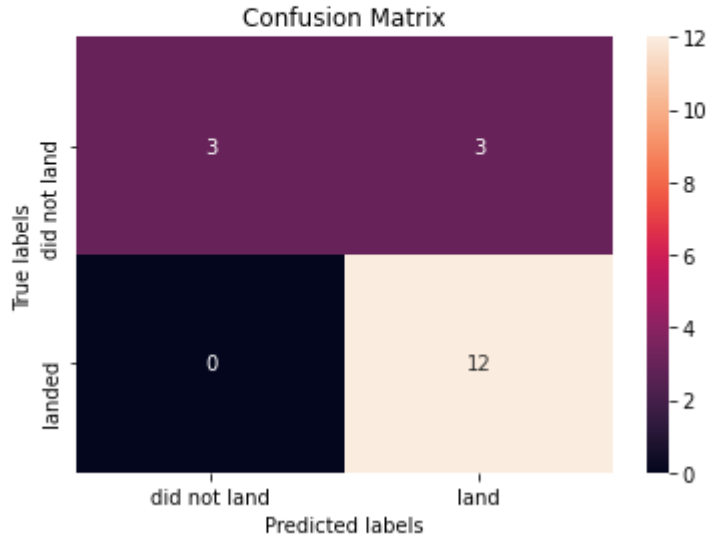
```
{'KNeighbors': 0.8482142857142858, 'Decision Tree': 0.9, 'LogisticRegression': 0.8464285714285713, 'Supportvector:svm_cv': 0.84
82142857142856}

<AxesSubplot:>
```



Confusion Matrix

- I created tree with the hyperparameters from the GridSearch results. The model only has 3 mistakes at predicting “land” that did not actually landed. All the rest were predicted correctly.

```
In [33]: tree2 = DecisionTreeClassifier(criterion='gini',max_depth=4,max_features='auto',
                                        min_samples_leaf=1, min_samples_split=10, splitter='best')
         tree2.fit(X_train, Y_train)
         yhat = tree2.predict(X_test)
         plot_confusion_matrix(Y_test,yhat)
```



Conclusions

The success rate increases as the flight amount gets larger.

The success rate increased from 2013 to 2020

The orbits with the highest success rate are ES-L1, HEO, VLEO, SSO

The site with the most success is KSC LC-39A

The best classifier is the Decision Tree Classifer

Innovative insights

It will be good to concentrate the efforts on the following orbits

ES-L1, GEO, HEO, SSP, VLEO

For heavy loads the best orbits are

LEO, PO, ISS

The success rate will improve as more launches are made, the systems will improve and the company will gain more knowledge

In the future the company can build a launching site near the coast, Florida seems to be a good place

Appendix

I added some code to the prediction notebook to show the bar plot and to create the confusion matrix of the Decision Tree classifier with the best values found by the Grid Search.

The notebook with the extra code can be found at

https://github.com/nicosiocursos/IBM_capstoneProject/blob/main/PredictionWithExtras.ipynb

All the notebooks from the Capstone Project are at

https://github.com/nicosiocursos/IBM_capstoneProject

Thank you!