IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Cooper Heaton
12/20/2023

Outline

Executive Summary

- Summary of methodologies
  - Data Collection with 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 with Screenshots
  - Predictive Analytics Result

Introduction

- Project background and context
- Problems you want to find answers
  - What factors will determine if the rocket can land successfully?

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

- A get request to the SpaceX API was used to collect the data.
- Next, decode the response using .json() function and turn it into a pandas dataframe with .json_normalize().
- Then, clean the data and fill in missing values when necessary.
- To get the Falcon 9 launch records we must perform web scraping with BeautifulSoup.
- Once we have the launch records we need to parse the data and convert it to a pandas dataframe for future analysis.

Data Collection – SpaceX API

- We called a get request to the SpaceX API to collect data and then clean the data

- https://github.com/cheaton622/IBMDataScienceCapstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Get request for rocket launch data with API

```
[ ]: spacex_url="https://api.spacexdata.com/v4/launches/past"
     response = requests.get(spacex_url)
```

convert json to pandas dataframe

```
[ ]: static_json_df = res.json()
     data = pd.json_normalize(static_json_df)
```

Data cleanse and fill in missing values

```
[ ]: 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

- Use BeautifulSoup to webscrape Falcon 9 launch records then parse the data and convert it to a pandas dataframe
- https://github.com/cheaton622/IBMDataScienceCapstone/blob/main/jupyter-labs-webscraping.ipynb

Apply HTTP Get method to request the Falcon 9 rocket launch page

```
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
html_data = requests.get(static_url)
html_data.status_code
```

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')
```

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

Extract all column names from the HTML table header

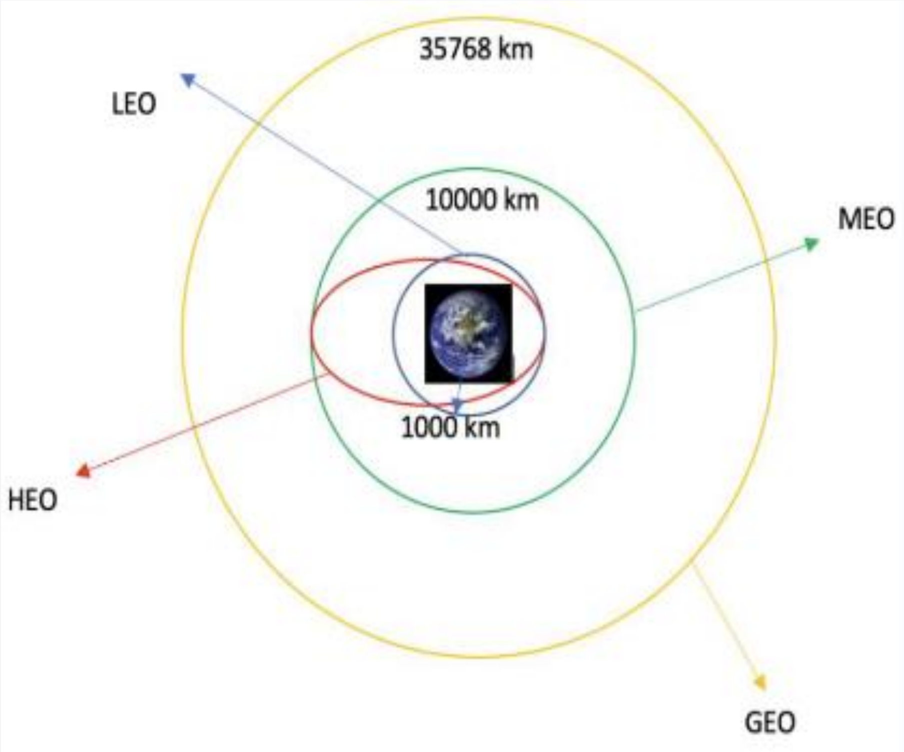
```
column_names = []

# 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
```

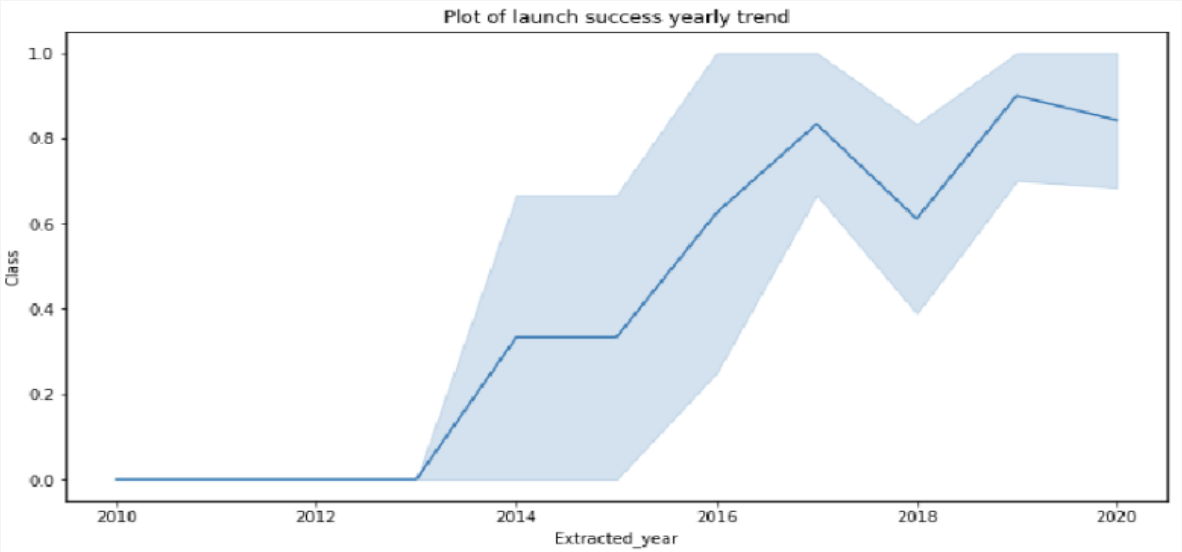
Data Wrangling

- I explored the data to determine training labels
- Then calculated the number of launches at each site and the count of occurrences for each orbit
- Lastly, I used the outcome column to create a landing outcome column and exported the results to a csv file.
- https://github.com/cheaton622/IBMDataScienceCapstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb



EDA with Data Visualization

- I explored the data by visualizing the relationship between flight number/launch site, payload/launch site, success of each orbit type, flight number/orbit type and a yearly trend of launch success



- https://github.com/cheaton622/IBMDataScienceCapstone/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

- I used a sqlite to store the SpaceX dataset
- I used queries to EDA within the sqlite database such as:
  - Display the unique launch sites
  - Total payload mass carried by boosters launched by NASA
  - Average payload mass carried by booster version F9 v1.1
  - List boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- https://github.com/cheaton622/IBMDataScienceCapstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- First, I marked all launch sites and used map objects to specify the success/failure of launches for each site on the folium map
- Assign the launch outcomes to 0 for failure and 1 for success
- Identify the launch sites with a high success rate using color-labeled n=marker clusters
- Calculate the distances between launch sites to see if they are near major structures such as cities, railways, highways, and oceans
- https://github.com/cheaton622/IBMDataScienceCapstone/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- On the dashboard there is a pie char that show the total launches by certain launch sites, as well as a scatter plot showing the relationship between outcome/payload mass for different booster versions
- https://github.com/cheaton622/IBMDataScienceCapstone/blob/main/app.py

Predictive Analysis (Classification)

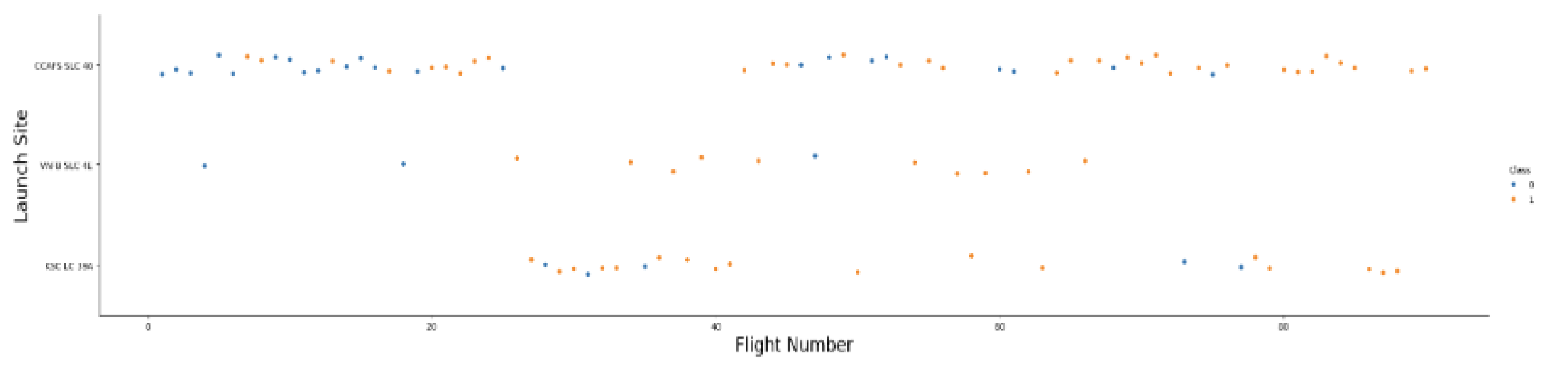
- Load, transform, and split the data into training/testing sets with numpy and pandas
- Build machine learning models and tune different hyperparameters with GridSearchCV
- Improve the accuracy of the model with feature engineering and tuning the algorithm to find the best performing classification model
- https://github.com/cheaton622/IBMDataScienceCapstone/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Section 2

Insights drawn from EDA

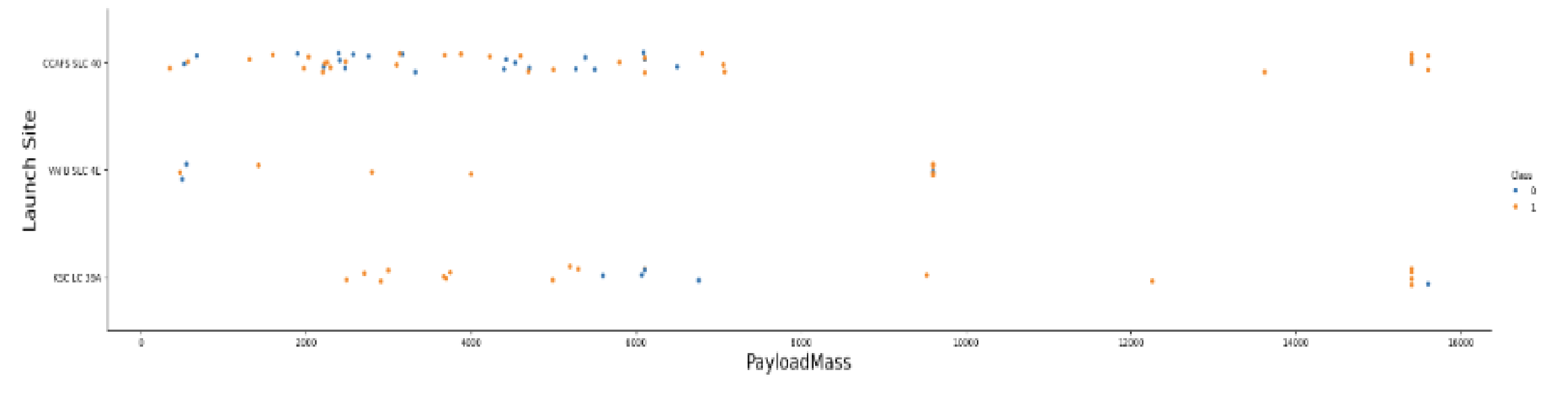
Flight Number vs. Launch Site

- The findings of this comparison is that the larger the flight amount at a launch site equals a greater success rate



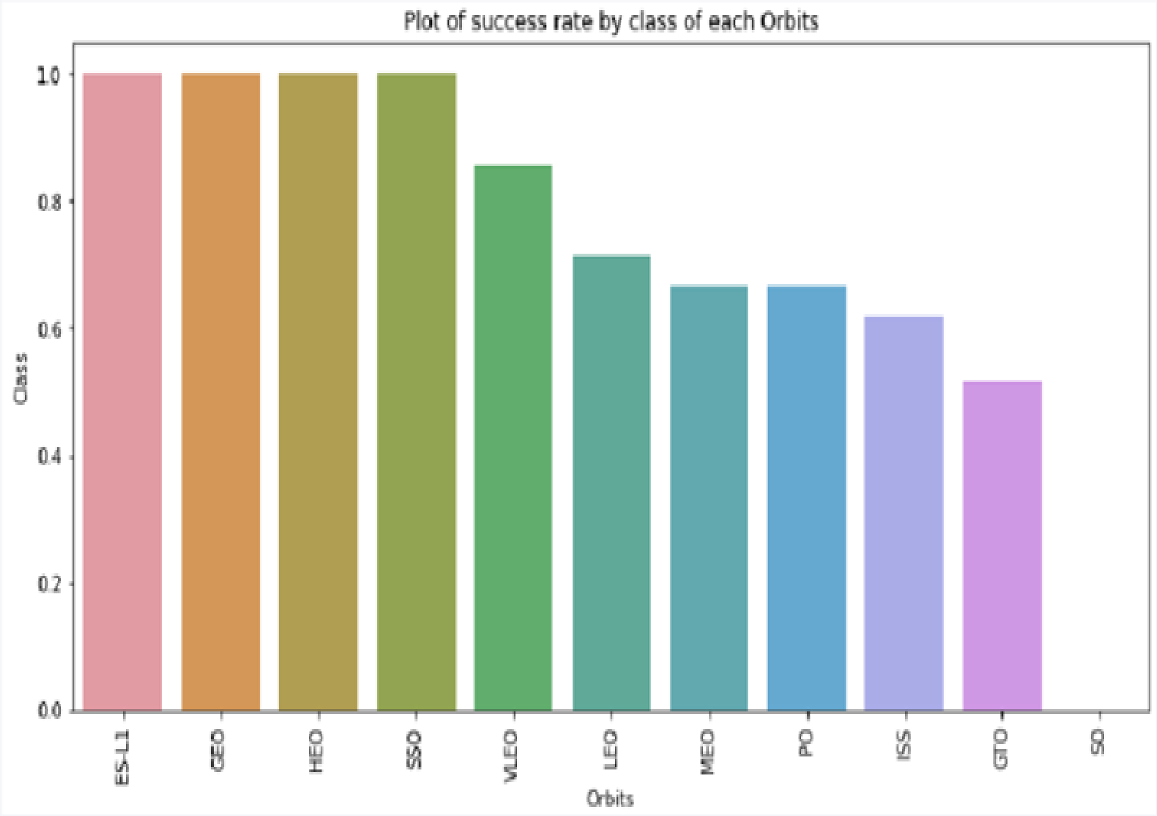
Payload vs. Launch Site

- The greater the payload mass typically means there will be a higher success rate for that launch site



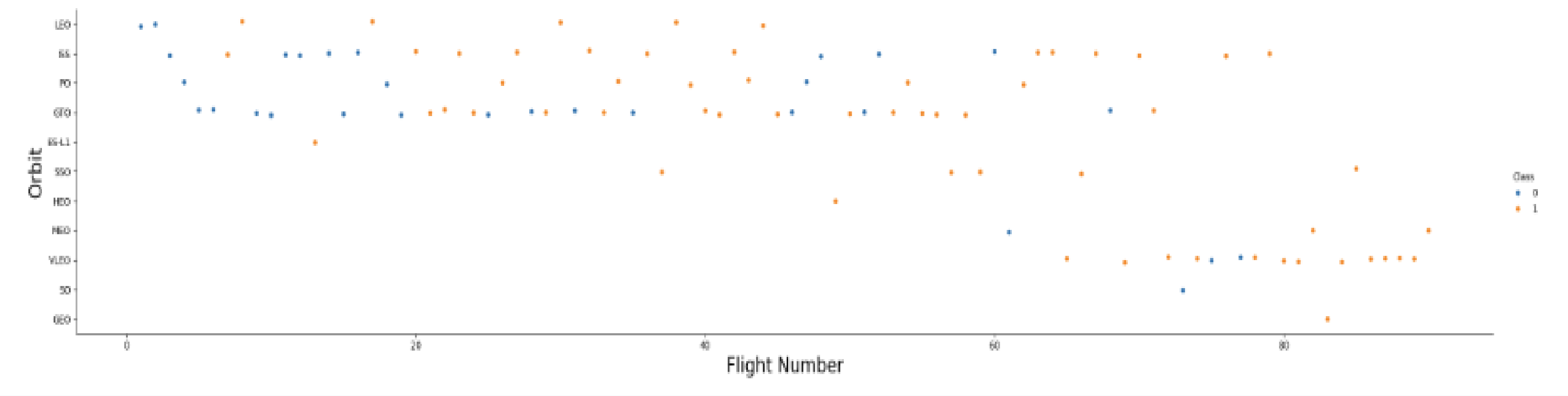
Success Rate vs. Orbit Type

- From the graph, we can see that ESL-1, GEO, HEO, and SSO had the highest success rate of each orbit



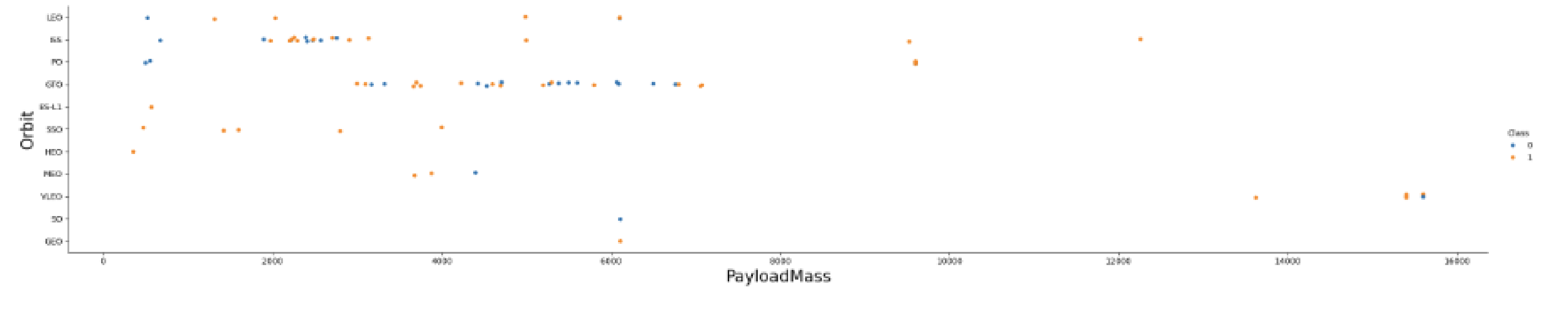
Flight Number vs. Orbit Type

- This plot shows that the more flights will yield a higher success rate for each orbit



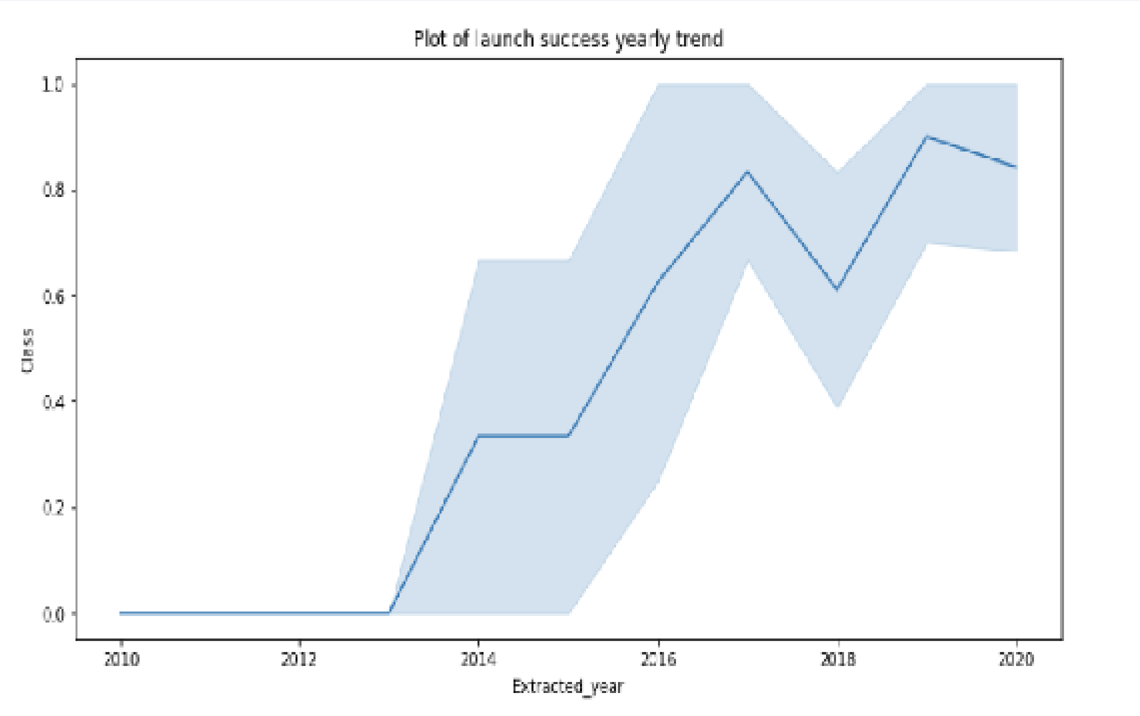
Payload vs. Orbit Type

- Most orbit sites use a smaller payload to yield a high success rate except for PO, YLEO, and ISS



Launch Success Yearly Trend

- The line chart shows that launch success has increased significantly since 2013



All Launch Site Names

- In this query ‘DISTINCT’ is used to show only unique launch site names

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

```
task_1 = '''
        SELECT DISTINCT Launch_Site
        FROM SPACEXTBL
'''

df = pd.read_sql(task_1, con)
df
```

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

Launch Site Names Begin with 'CCA'

- LIMIT 5 is used to select only the first 5 rows

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

```
task_2 = '''
        SELECT *
        FROM SPACEXTBL
        WHERE Launch_Site LIKE 'CCA%'
        LIMIT 5
        '''
df = pd.read_sql(task_2, con)
df
```

	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcor
0	2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachu
1	2010-12-08	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 (parachu
2	2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No atterr
3	2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No atterr
4	2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No atterr

Total Payload Mass

- Display the sum of payload mass by using the SUM() function
- Specify the customer using a WHERE statement

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

```
task_3 = '''
        SELECT SUM(PAYLOAD_MASS__KG_) AS Total_PayloadMass
        FROM SPACEXTBL
        WHERE Customer LIKE 'NASA (CRS)'
        '''
df = pd.read_sql(task_3, con)
df
```

	Total_PayloadMass
0	45596

Average Payload Mass by F9 v1.1

- The average payload mass carried by booster version F9 v1.1 is 2,928.4 kg

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

```
task_4 = '''
        SELECT AVG(PAYLOAD_MASS__KG_) AS Avg_PayloadMass
        FROM SPACEXTBL
        WHERE Booster_Version = 'F9 v1.1'
        '''
df = pd.read_sql(task_4, con)
df
```

	Avg_PayloadMass
0	2928.4

First Successful Ground Landing Date

- The date of the first successful ground landing date is December 22, 2015

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

Hint:Use min function

```
task_5 = '''
        SELECT MIN(Date) AS FirstSuccessfull_landing_date
        FROM SPACEXTBL
        WHERE Landing_Outcome LIKE 'Success (ground pad)'
        '''
df = pd.read_sql(task_5, con)
df
```

	FirstSuccessfull_landing_date
0	2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- There have been 4 different booster versions to land successfully with a payload between 4,000 and 6,000

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 Booster_Version
        FROM SPACEXTBL
        WHERE Landing_Outcome = 'Success (drone ship)'
            AND PAYLOAD_MASS__KG_ > 4000
            AND PAYLOAD_MASS__KG_ < 6000
        '''
df = pd.read_sql(task_6, con)
df
```

	Booster_Version
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 the wildcard feature "%" to find that there were 100 successful outcomes and 1 failure

List the total number of successful and failure mission outcomes

```
task_7a = '''
        SELECT COUNT(Mission_Outcome) AS SuccessOutcome
        FROM SPACEXTBL
        WHERE Mission_Outcome LIKE 'Success%'
        '''

task_7b = '''
        SELECT COUNT(Mission_Outcome) AS FailureOutcome
        FROM SPACEXTBL
        WHERE Mission_Outcome LIKE 'Failure%'
        '''
print('The total number of successful mission outcome is:')
df = pd.read_sql(task_7a, con)
print(df)
print()
print('The total number of failed mission outcome is:')
df1 = pd.read_sql(task_7b, con)
print(df1)
```

```
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

- In this query there is a subquery that has a WHERE clause using the MAX() function to determine which booster versions have used the maximum payload mass

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

```
task_8 = '''
        SELECT Booster_Version, PAYLOAD_MASS__KG_
        FROM SPACEXTBL
        WHERE PAYLOAD_MASS__KG_ = (
                                    SELECT MAX(PAYLOAD_MASS__KG_)
                                    FROM SPACEXTBL
                                    )
        ORDER BY Booster_Version
        '''
df = pd.read_sql(task_8, con)
df
```

	Booster_Version	PAYLOAD_MASS__KG_
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

- This query uses a WHERE clause to specify that landing outcome equals failure and it is in the year 2015

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.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

```
task_9 = '''
        SELECT Booster_Version, Launch_Site, Landing_Outcome
        FROM SPACEXTBL
        WHERE Landing_Outcome LIKE 'Failure (drone ship)'
            AND Date BETWEEN '2015-01-01' AND '2015-12-31'
        '''
df = pd.read_sql(task_9, con)
df
```

	Booster_Version	Launch_Site	Landing_Outcome
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

- This query shows the landing outcomes in the dates between 2010-06-04 and 2017-03-20.
- The outcomes are sorted by count in descending order using the ORDER BY clause

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad))

```
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)
```

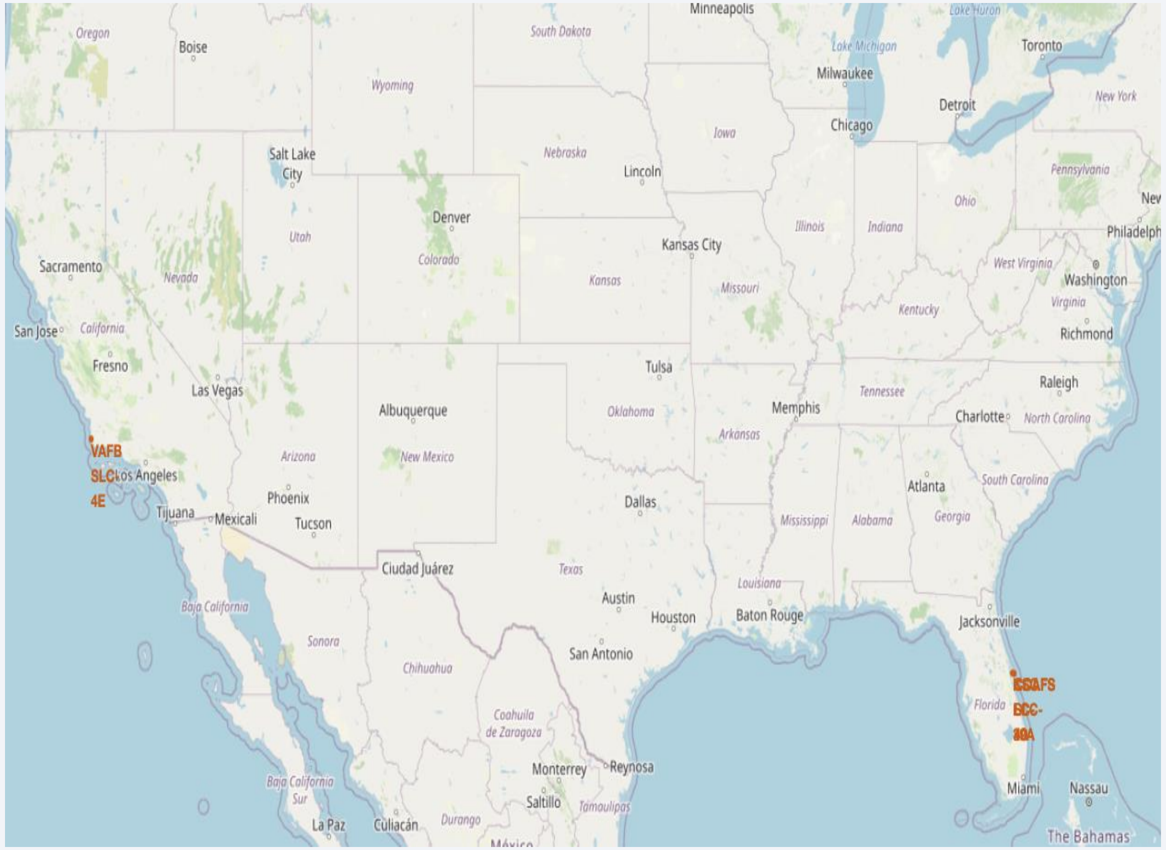
	landingoutcome	count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1
7	Failure (parachute)	1

Section 3

Launch Sites Proximities Analysis

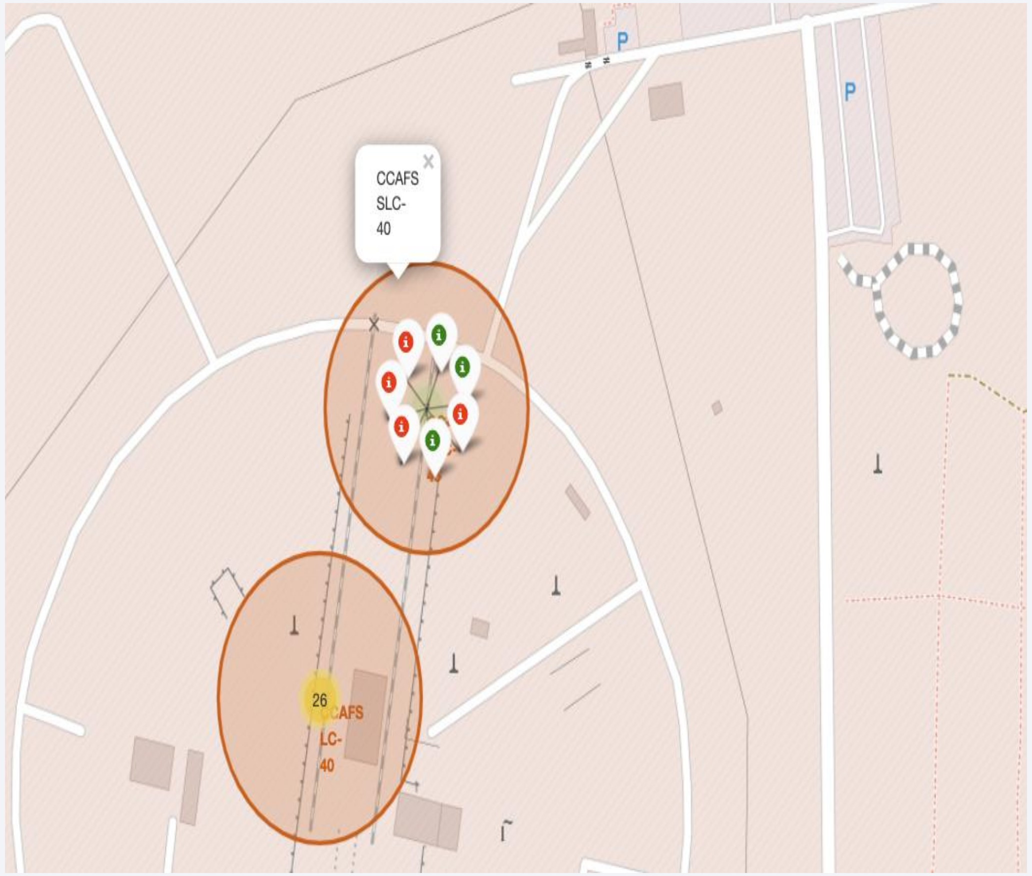
SpaceX Launch Sites

- SpaceX launch sites are on the coasts of Florida and California



Launch Sites with Color Labels

- These are the Florida launch sites
- The green indicate successful launches and red are unsuccessful



Launch Sites Proximity to Major Landmarks

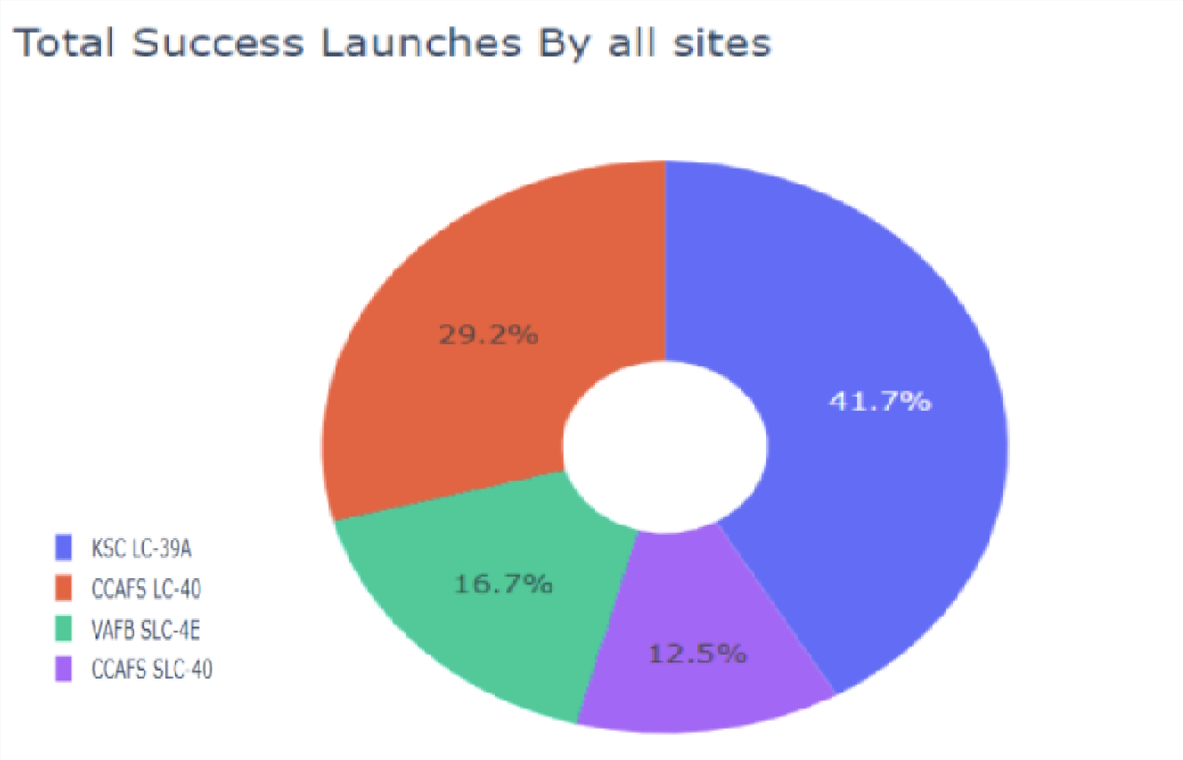
- Launch sites are not close to railways, highways, or cities
- Launch sites are close to coastlines

Section 4

Build a Dashboard with Plotly Dash

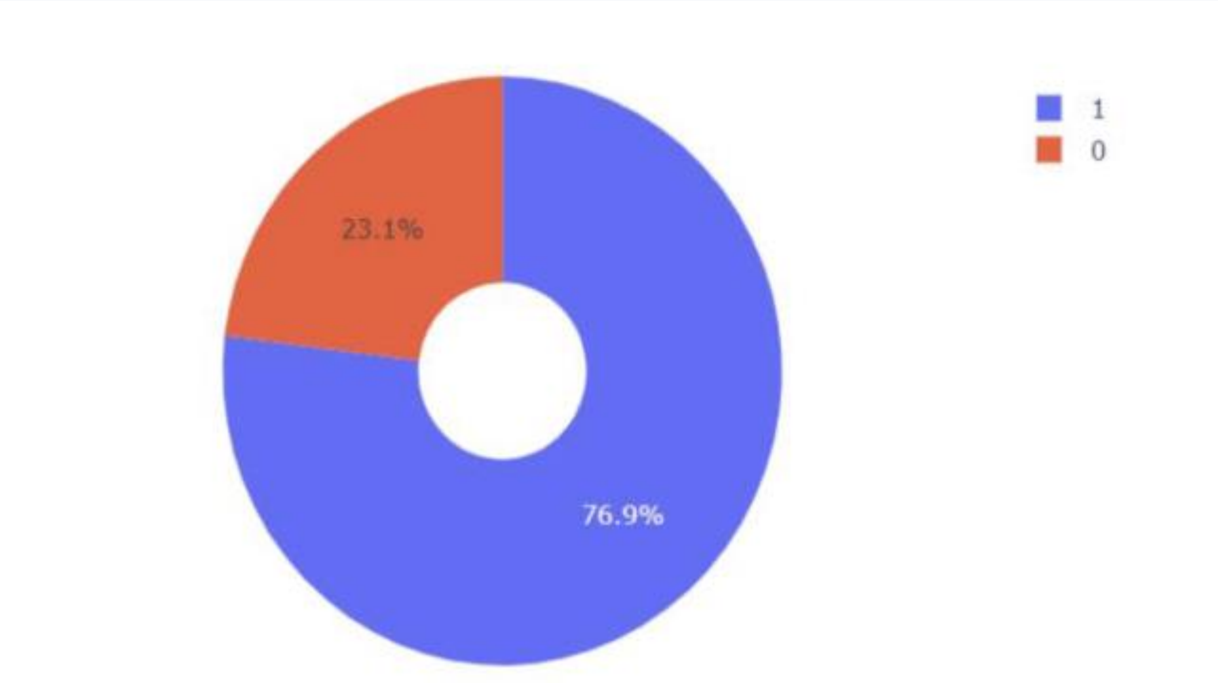
Success Percentage by Launch Site

- KSC LC-39A has the highest success rate



Launch Site with the Highest Success Ratio

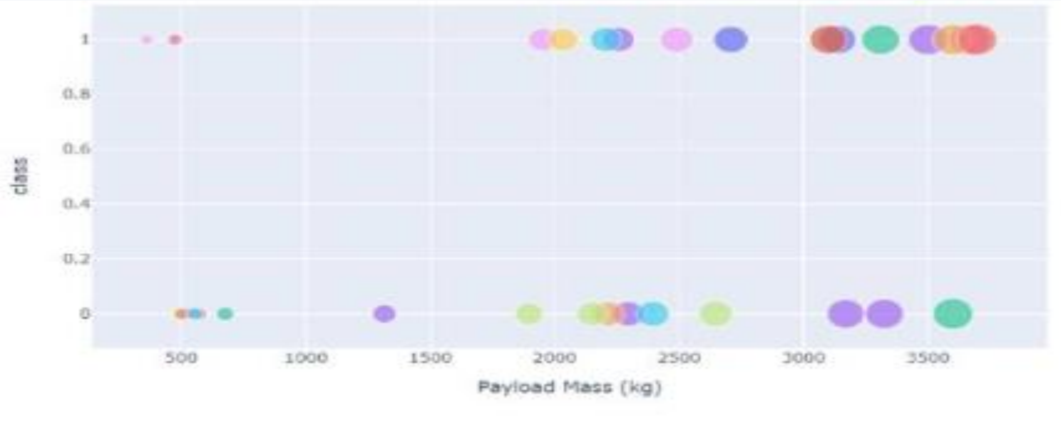
- KSC LC-39A achieved a 76.9% success percentage



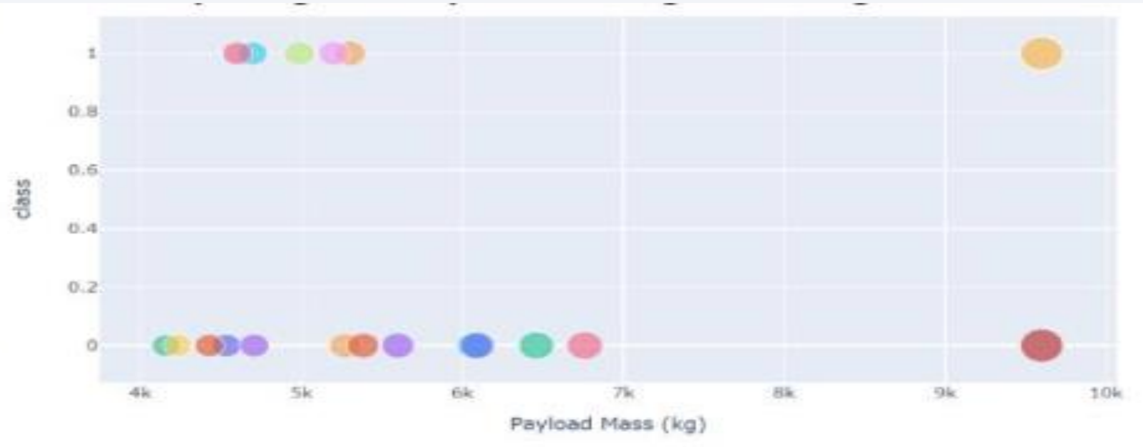
Payload vs Launch Outcome for Each Site

- The success rate for lower weighted payloads is higher than the heavy weighted payloads

Payload 0-4000kg



Payload 4000-10000kg

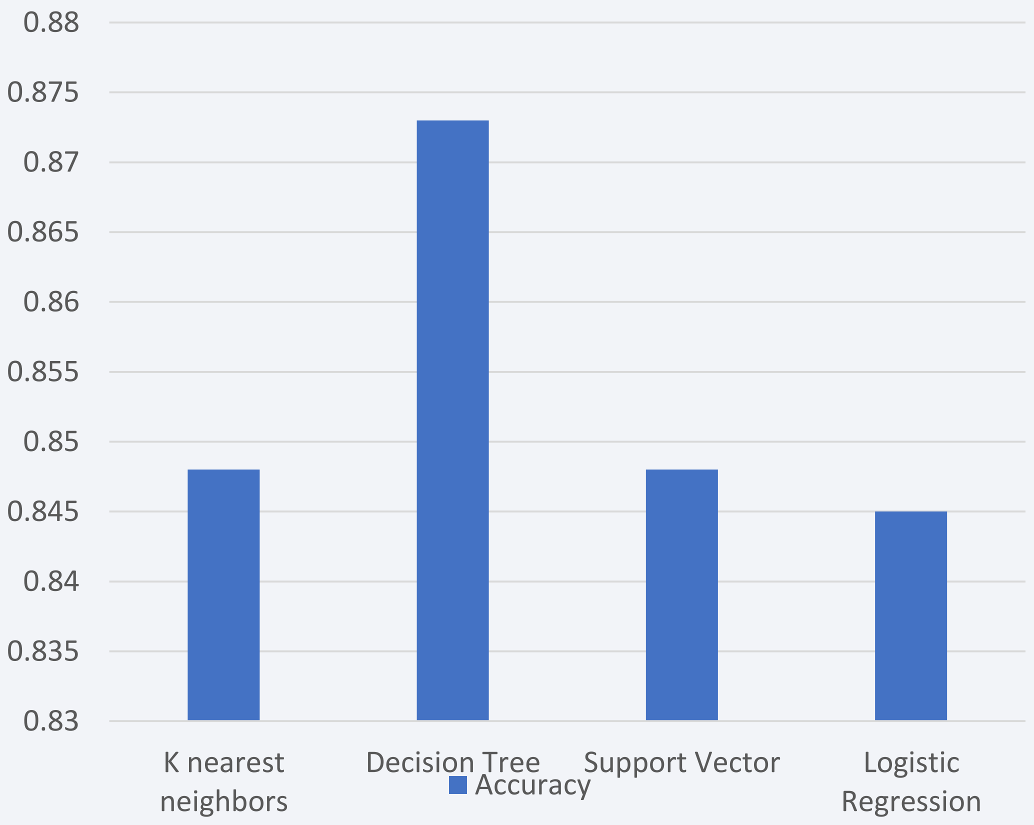


Section 5

Predictive Analysis (Classification)

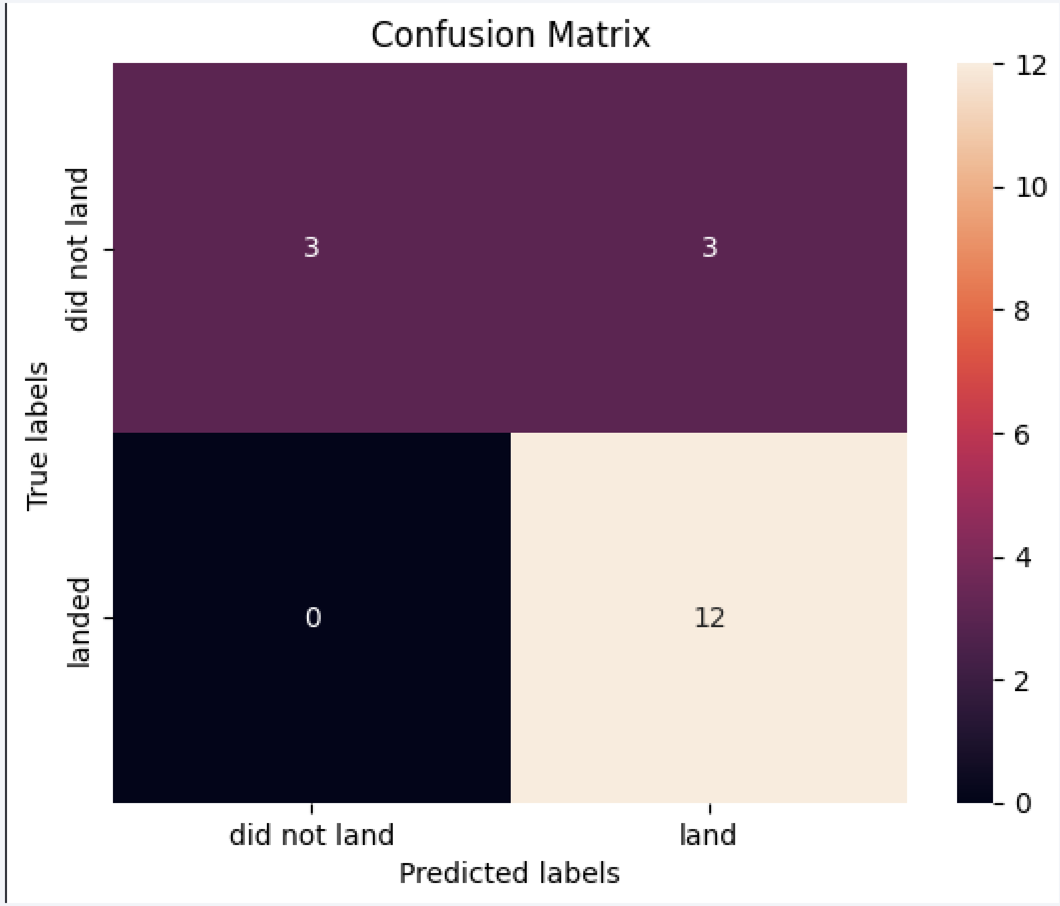
Classification Accuracy

- The decision tree classifier model has the highest accuracy



Confusion Matrix

- The confusion matrix for the decision tree classification model shows how well the model can distinguish between the different classes.
- It also shows that the classifier could have been more accurate if it did not mark unsuccessful landings as successful



Conclusions

- Launch sites had greater success rates when they had a higher number of flights.
- Launch success rate has increased dramatically over time.
- The orbits with the highest success rates were ES-L 1, GEO, HEO, SSO, and VLEO.
- The launch site with the highest success rate was KSC LC-39A.
- The decision tree classifier was the most successful algorithm to predict if the first stage would land.

Appendix

- All of my relevant assets are on my GitHub repository:

https://github.com/cheaton622/IBMDataScienceCapstone/tree/main

Thank you!