Winning Space Race with Data Science

Arian Larios
7/22/2022

Outline

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

Executive Summary

- Summary of methodologies
  - Using APIs to collect data
  - Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis
  - Interactive Dashboard
  - Predictive Analytics
- Summary of all results
  - Analysis Results
  - Machine Learning Results

Introduction

- Project background and context

  The purpose of this project is to determine if we can perform rocket launches at a lower cost than Space X can. We can accomplish this by analyzing data to determine whether the first stage of our launch will be successful. The result of this project will be a machine learning pipeline that can predict if these launches will be successful.

- Problems you want to find answers
  - Which variables affect the result of our launches?
  - How can we increase the chances of a successful launch?
  - Are there any risks or opportunities?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX's API and web scraping.
- Perform data wrangling
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models

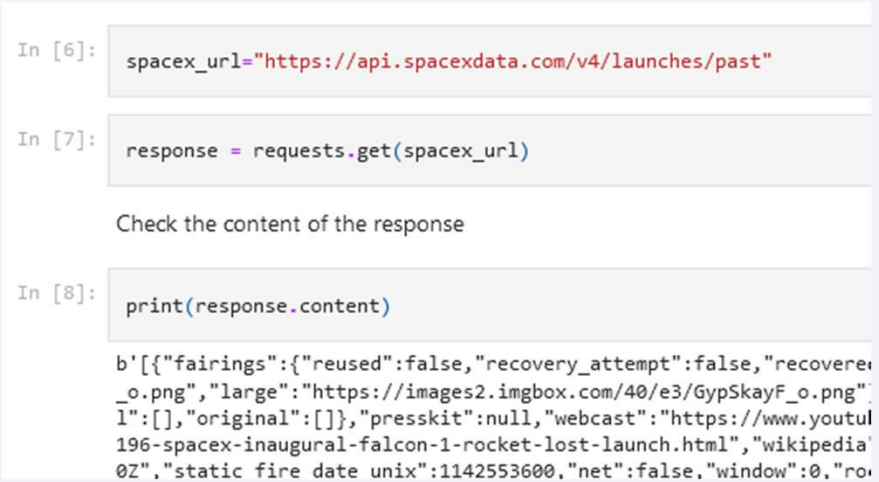
Data Collection

- Describe how data sets were collected.
  - Some of the data was collect via SpaceX's API.
  - Then I converted the JSON response content into a Pandas data frame for future analysis.
  - I cleaned the data and checked for any missing values.
  - I used the Beautiful Soup library to web scrape data for Falcon 9 launches from Wikipedia.
  - Beautiful Soup is used to extract the HTML table and convert it to a Pandas data frame as well.

Data Collection – SpaceX API

- I used the get request function to collect data from the Space X API

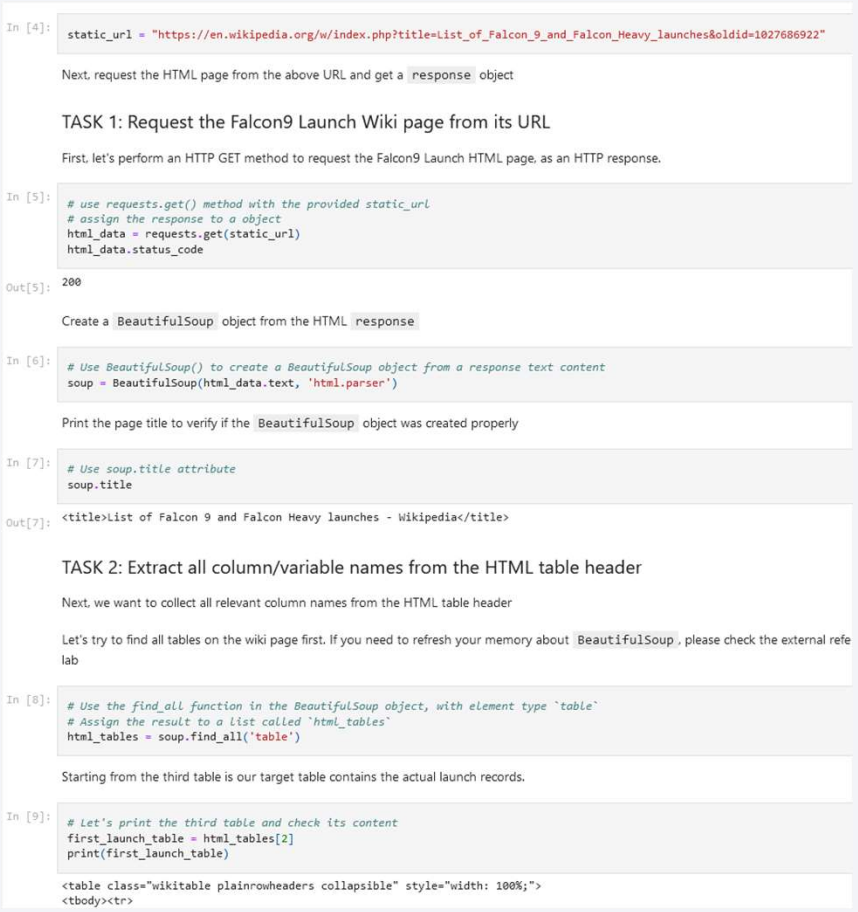
- https://github.com/Arianlg/Applied-Data-Capstone-Project/blob/main/Data%20Collection%20API.ipynb



Data Collection - Scraping

- I used beautiful soup to web scrape the Falcon 9 Launch data.

- https://github.com/Arianlg/Applied-Data-Capstone-Project/blob/main/Data%20Collection%20with%20Web%20Scraping.ipynb



In [4]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

Next, request the HTML page from the above URL and get a `response` object

TASK 1: Request the Falcon9 Launch Wiki page from its URL

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

In [5]:
```
# 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
```

Out[5]:
```
200
```

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

In [6]:
```
# 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

In [7]:
```
# Use soup.title attribute
soup.title
```

Out[7]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external refe lab

In [8]:
```
# 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.

In [9]:
```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

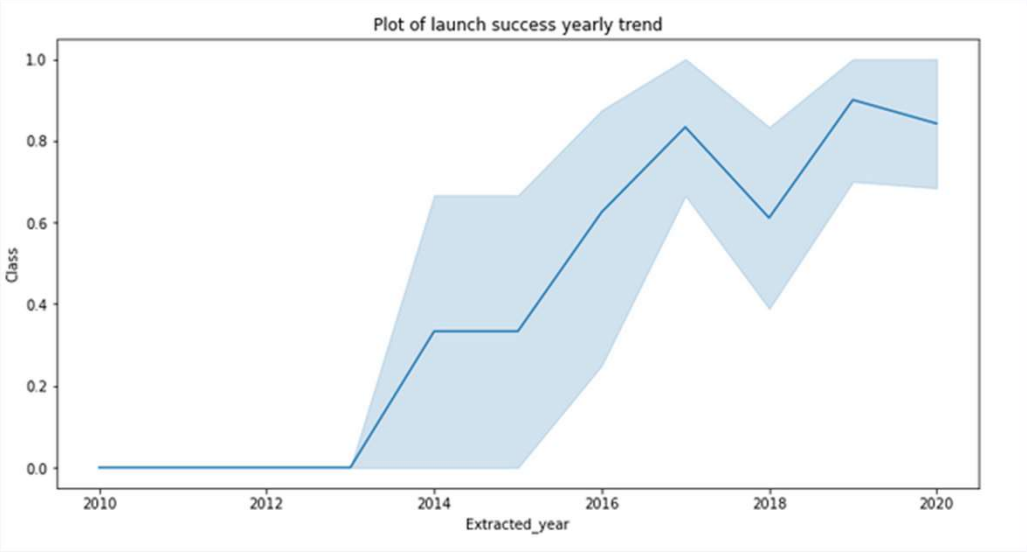
```
<table class="wikitable plainrowheaders collapsible" style="width: 100%;">
<tbody><tr>
```

Data Wrangling

- I performed exploratory data analysis to better understand the data.
- I calculated the total number of launches at each site and the number of orbits.
- https://github.com/Arianlg/Applied-Data-Capstone-Project/blob/main/Data%20Wrangling.ipynb

EDA with Data Visualization

- I used bar and line graphs to understand the relationship between variables and understand trends over time.
- https://github.com/Arianlg/Applied-Data-Capstone-Project/blob/main/Exploratory%20Data%20Analysis.ipynb



EDA with SQL

- I use SQL to query the names of launch sites in each mission, total payload mass carried by boosters, and the total number of successful for failed missions.
- https://github.com/Arianlg/Applied-Data-Capstone-Project/blob/main/Exploratory%20Data%20Analysis%20with%20SQL.ipynb

Build an Interactive Map with Folium

- I marked the launch sites on a map and noted which ones were successful and which ones failed.
- It was also important to mark the distance from each launch site.
- https://github.com/Arianlg/Applied-Data-Capstone-Project/blob/main/Folium%20Dashboard.ipynb

Build a Dashboard with Plotly Dash

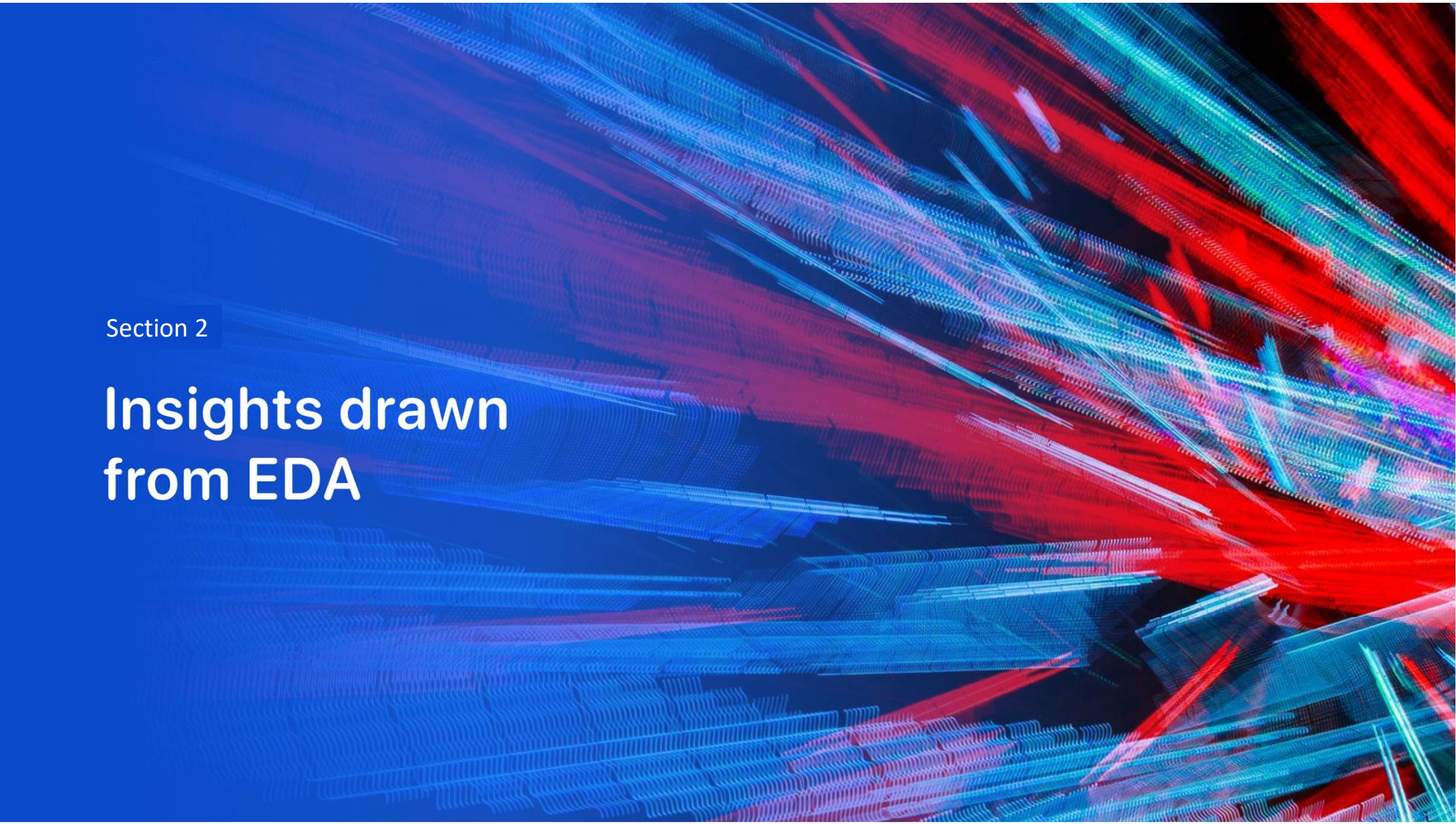
- I created an interactive dashboard using Plotly
- I used visualizations such as pie charts to derive insight from the data.
- https://github.com/Arianlg/Applied-Data-Capstone-Project/blob/main/Dashboard.py

Predictive Analysis (Classification)

- I used numpy and pandas to transform the data and split the data into training and testing data.
- It is important to use a few machine learning algorithms and then evaluate to determine which one is the most accurate.
- https://github.com/Arianlg/Applied-Data-Capstone-Project/blob/main/Machine%20Learning%20Prediction.ipynb

Results

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

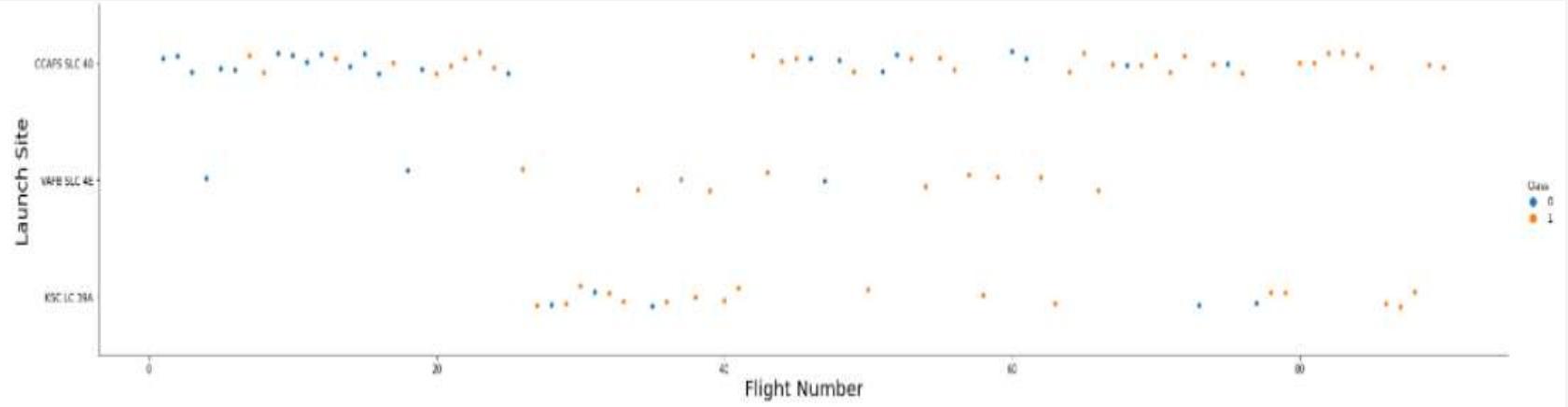


Section 2

Insights drawn from EDA

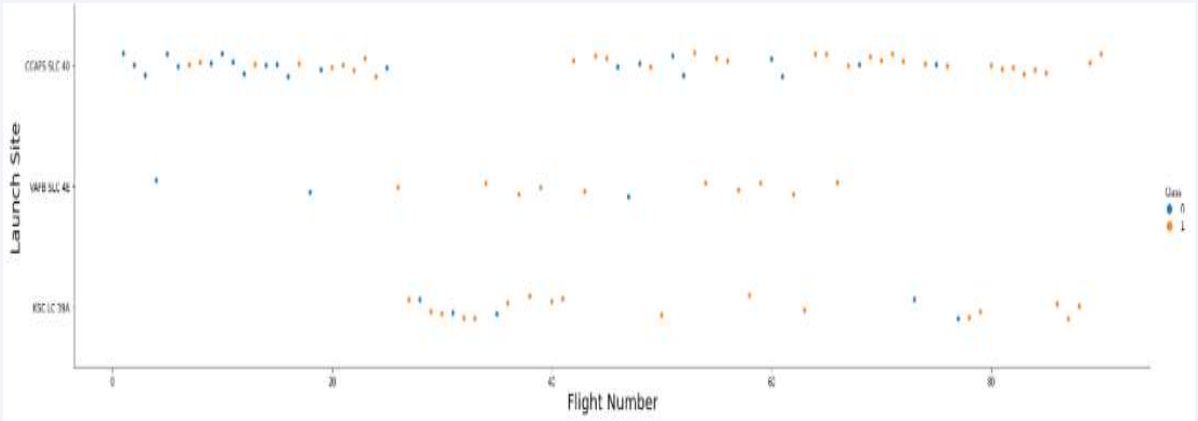
Flight Number vs. Launch Site

- Based on the plot, we can see that the larger the flight number, the greater the success rate.



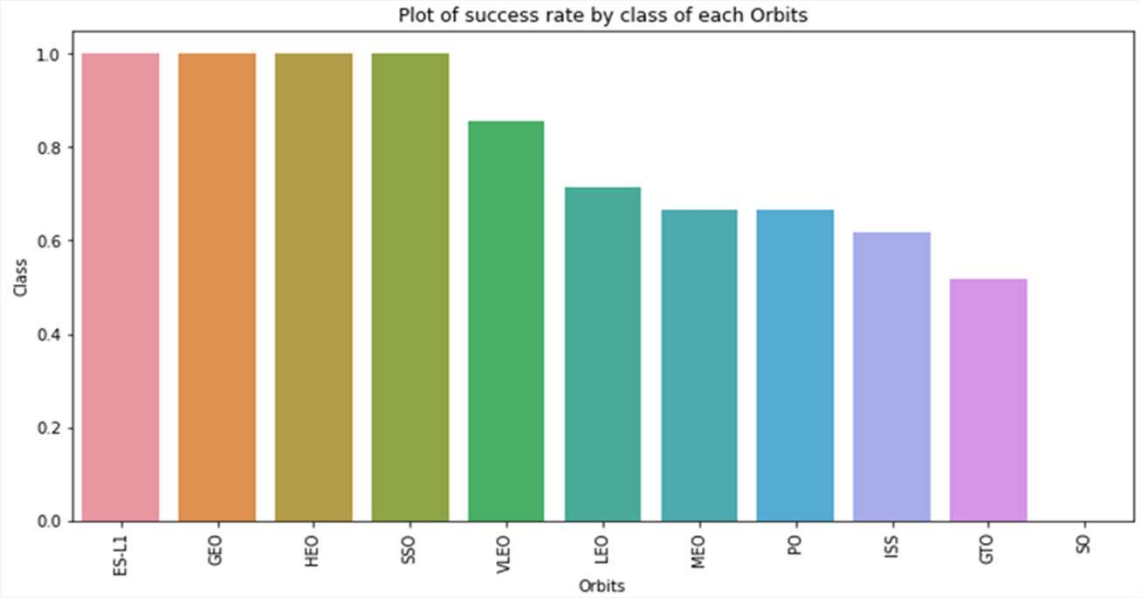
Payload vs. Launch Site

- The greater the payload mass, the higher the success rate for the rocket.



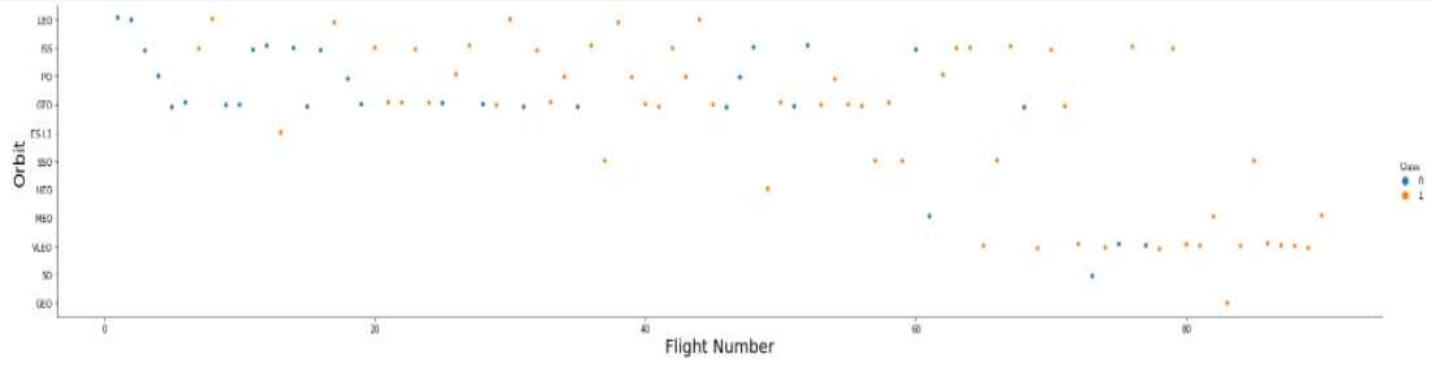
Success Rate vs. Orbit Type

- The bar chart shows that ES-L1, GEO, HEO, SSO are the most successful.



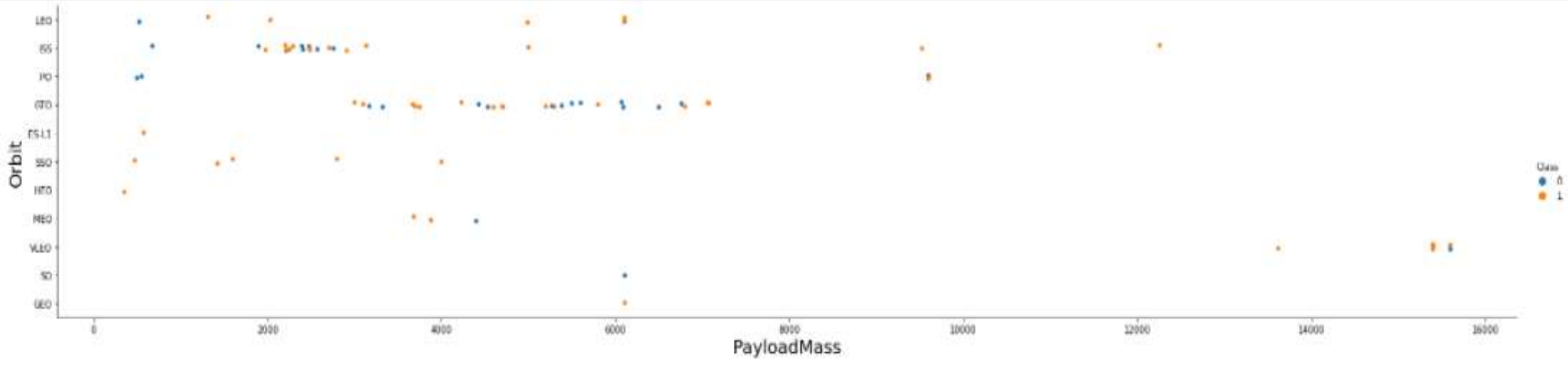
Flight Number vs. Orbit Type

- The plot below shows that there is a relationship between the success of a mission and the number of flights regarding the LEO orbit.



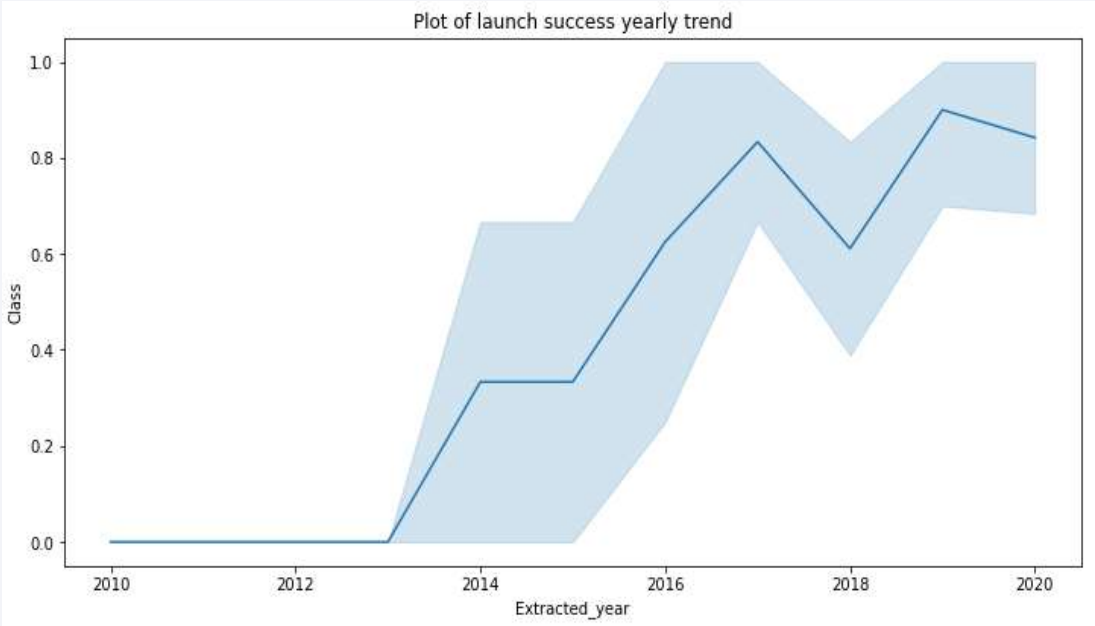
Payload vs. Orbit Type

- The heavier the payload, the more chance of success.



Launch Success Yearly Trend

- The success rate has constantly increased since 2013, with a slight dip in 2018.



All Launch Site Names

- I used the Distinct function in SQL to query the names of the launch sites.

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

In [10]:
```
task_1 = '''
        SELECT DISTINCT LaunchSite
        FROM SpaceX
'''
create_pandas_df(task_1, database=conn)
```

Out[10]:

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

Launch Site Names Begin with 'CCA'

- I used a Where clause to find launch sites that contained “CCA”

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

In [11]:
```
task_2 = '''
        SELECT *
        FROM SpaceX
        WHERE LaunchSite LIKE 'CCA%'
        LIMIT 5
        '''
create_pandas_df(task_2, database=conn)
```

Out[11]:

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

Total Payload Mass

- I used the Sum function to calculate the total payload mass.

In [12]:
```
task_3 = '''
        SELECT SUM(PayloadMassKG) AS Total_PayloadMass
        FROM SpaceX
        WHERE Customer LIKE 'NASA (CRS)'
        '''
create_pandas_df(task_3, database=conn)
```

Out[12]:

	total_payloadmass
0	45596

Average Payload Mass by F9 v1.1

- I used the Average function and Where clause to calculate the average payload mass for FP v1.1's.

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

In [13]:
```
task_4 = '''
        SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
        FROM SpaceX
        WHERE BoosterVersion = 'F9 v1.1'
        '''
create_pandas_df(task_4, database=conn)
```

Out[13]:

	avg_payloadmass
0	2928.4

First Successful Ground Landing Date

- I used the Minimum function to find the date of the first successful ground landing date.

In [14]:
```
task_5 = '''
        SELECT MIN(Date) AS FirstSuccessfull_landing_date
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Success (ground pad)'
        '''
create_pandas_df(task_5, database=conn)
```

Out[14]:

	firstsuccessfull_landing_date
0	2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- I used multiple Where clauses to filter the data to just successful landings with a payload between 4000 and 6000.

In [15]:
```
task_6 = '''
        SELECT BoosterVersion
        FROM SpaceX
        WHERE LandingOutcome = 'Success (drone ship)'
            AND PayloadMassKG > 4000
            AND PayloadMassKG < 6000
        '''
create_pandas_df(task_6, database=conn)
```

Out[15]:

	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

- Two separate queries are used to sum the total successes and failures.

List the total number of successful and failure mission outcomes

In [16]:
```
task_7a = '''
        SELECT COUNT(MissionOutcome) AS SuccessOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Success%'
        '''

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

Out[16]:

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

- I used a subquery to determine the booster that could carry the max payload.

In [17]:
```
task_8 = '''
        SELECT BoosterVersion, PayloadMassKG
        FROM SpaceX
        WHERE PayloadMassKG = (
                                SELECT MAX(PayloadMassKG)
                                FROM SpaceX
                                )
        ORDER BY BoosterVersion
        '''
create_pandas_df(task_8, database=conn)
```

Out[17]:

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

2015 Launch Records

- A combination of Where, LIKE, and BETWEEN are used to pull the 2015 lauch records.

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

In [18]:
```
task_9 = '''
        SELECT BoosterVersion, LaunchSite, LandingOutcome
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Failure (drone ship)'
            AND Date BETWEEN '2015-01-01' AND '2015-12-31'
        '''
create_pandas_df(task_9, database=conn)
```

Out[18]:

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

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

- The Group By and Order By functions are used to filter the data.

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

In [19]:
```
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)
```

Out[19]:

	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

Launch Sites

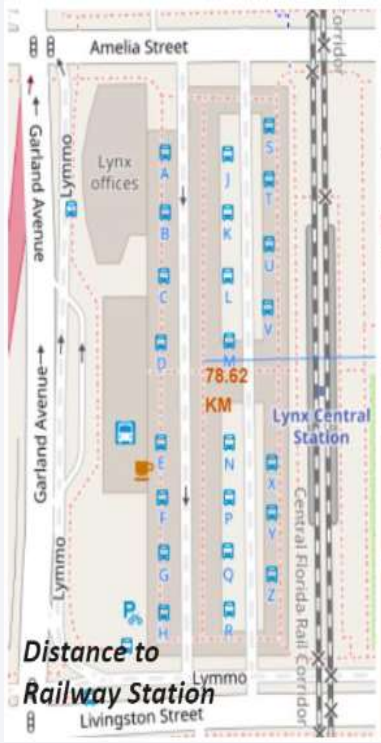
- The screenshot below shows that all launch sites are within the United States.

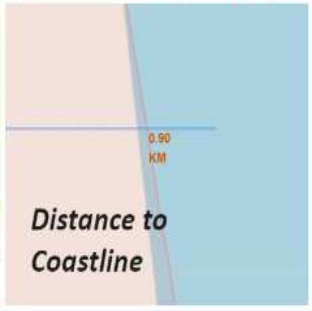
Map with Markers.

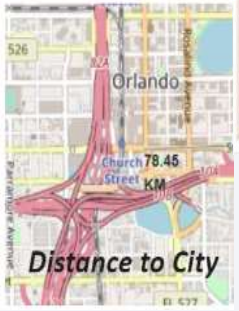
- Green markers are used to signify successful launches, and red markers are the failures.

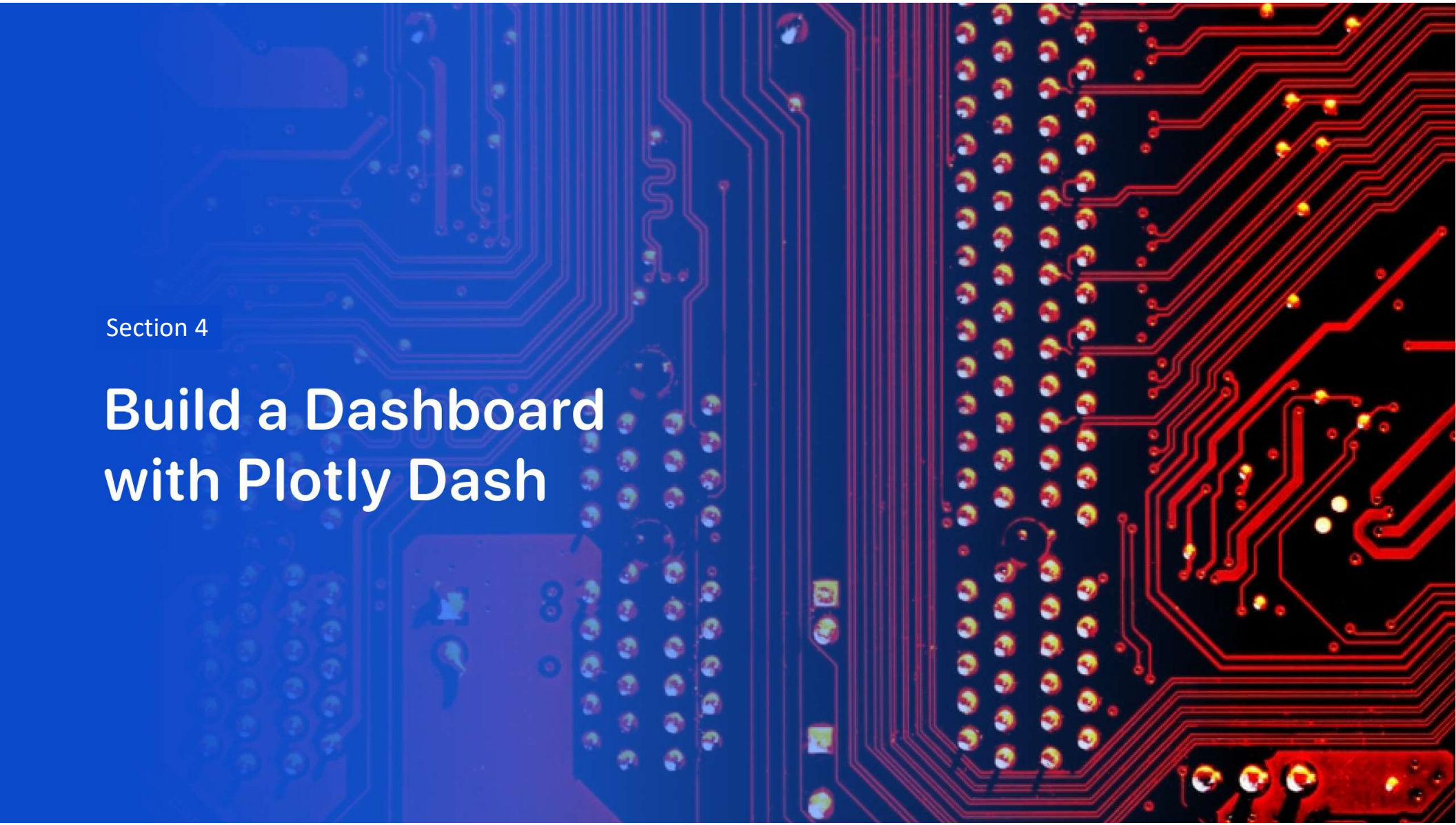
Distance from Launch Sites to Landmarks

- The maps show that launch sites are not close to railways nor highways, but are close to coastlines and far enough away from cities.







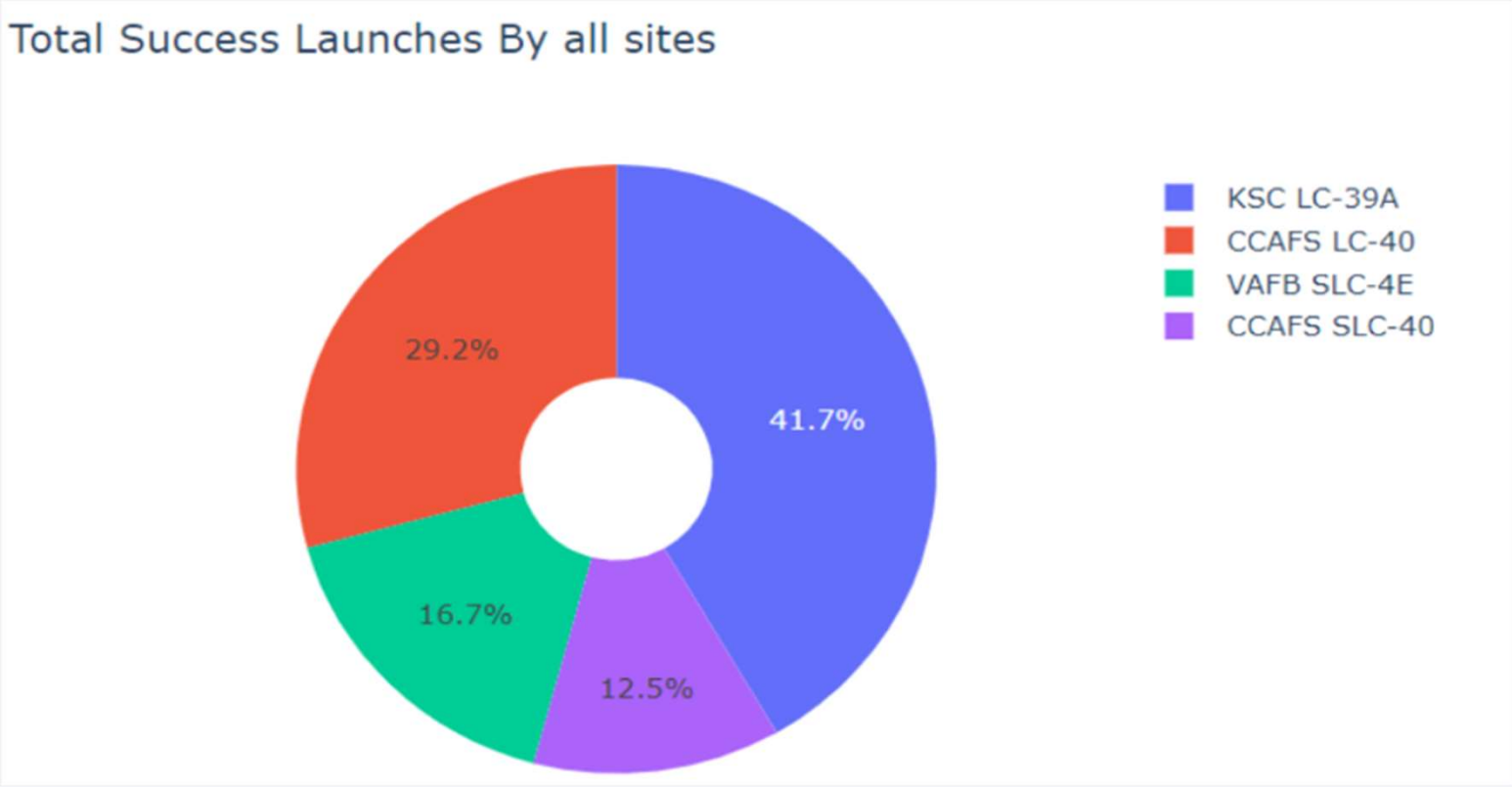


Section 4

Build a Dashboard with Plotly Dash

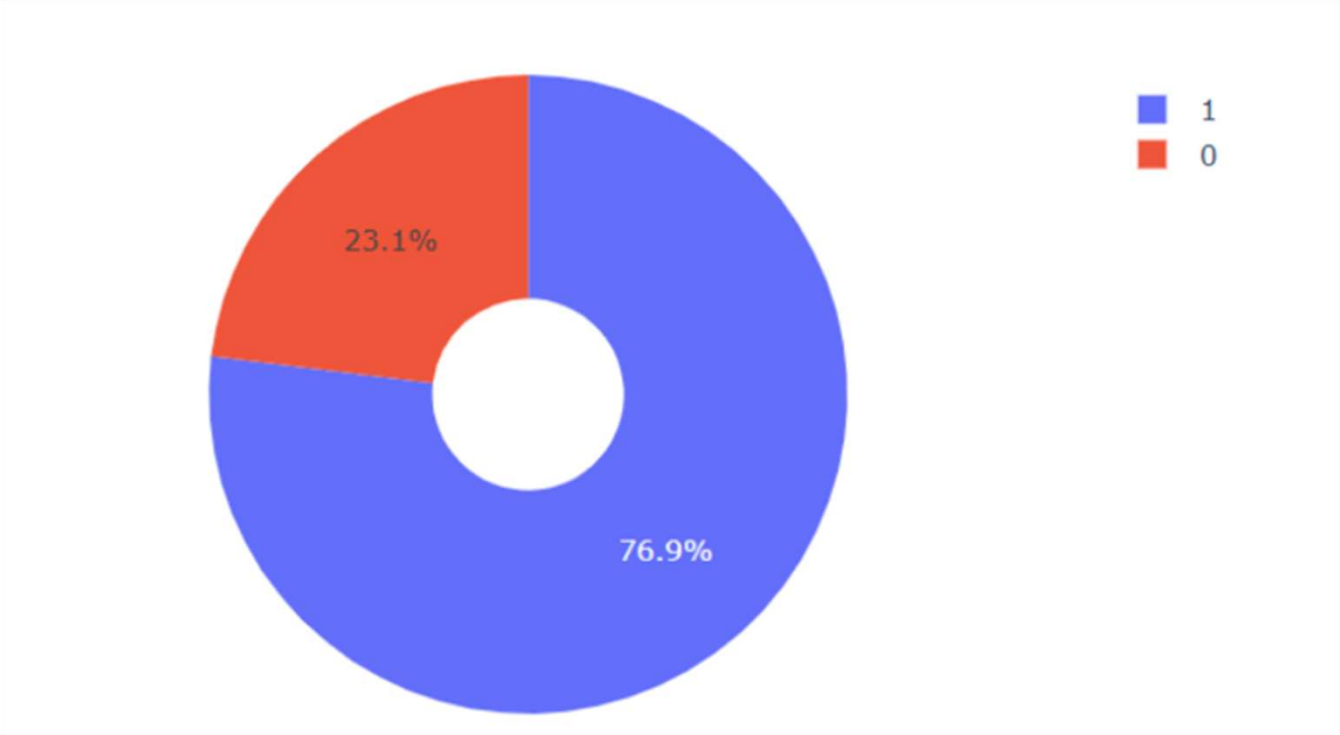
Pie Chart Success Rate

- The pie chart shows that the KSC LC-39A



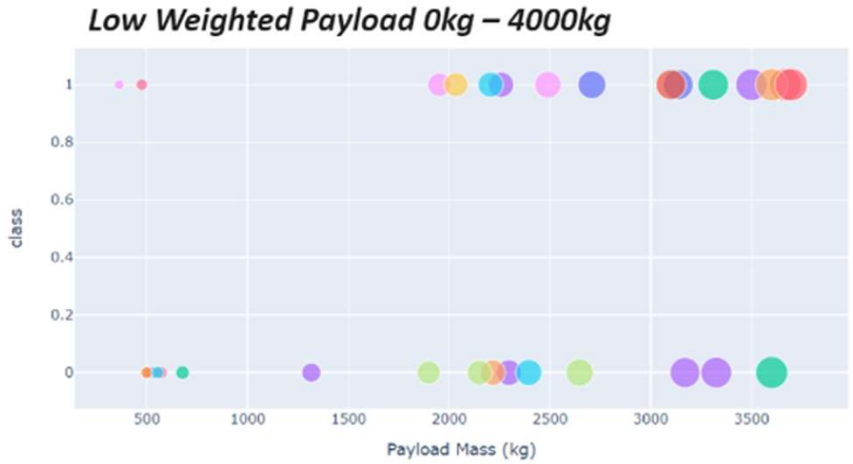
Pie Chart Success Ratio

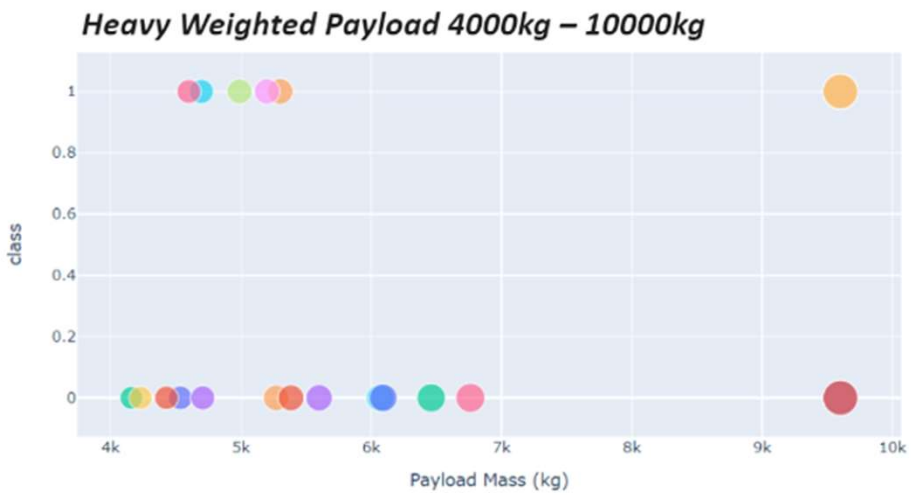
- The KSC LC-39A achieved a higher success rate.



Scatter Plot – Payload vs Launch Outcome

- The success rate for lower weights is higher than higher weighted rockets.





Section 5

Predictive Analysis (Classification)

Classification Accuracy

- The decision tree classifier is the best model.

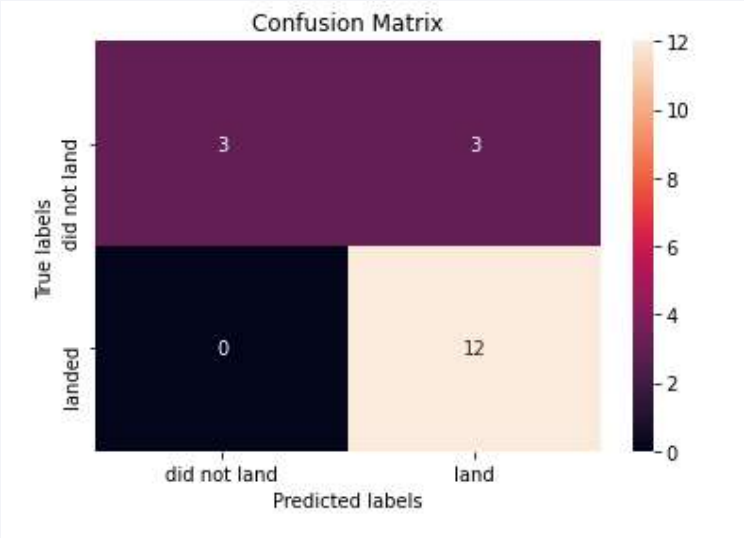
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

Confusion Matrix

- This classifier can distinguish between different classes.



Conclusions

- The larger the flight number, the higher chance of success.
- Our rockets did not become successful until 2013.
- The ES-L1, GEO, HEO, SSO, and VLEO were the most successful.
- The decision tree classifier is the most accurate model.

Thank you!