IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Ahmed Mirza
16/01/2022

Outline

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

Executive Summary

- Summary of methodologies: Several methods were used in this report to obtain various information on Falcon 9 in the SpaceX project
    - Data Collection through 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 in screenshots
    - Predictive Analytics result

Introduction

Project background and context:

Space X has Falcon 9 rocket launches on its Wikipedia website with a cost of 62 million dollars; other providers cost upward of 165 million USD per rocket. Savings can be realized early because Space X can reuse rockets in the first stage. Therefore, if it can be determined if the first stage will land, one can determine the cost of a launch. This information can be used by other companies wants to bid against space X for a rocket launch. This goal of the project is to create a machine learning pipeline model to predict if the first stage will land successfully.

Problems that require answers are as follows:

- What variables determine if the rocket will land successfully?
- The interactions amongst variables that will determine the success rate of a successful landing.
- What conditions needs to be in place to ensure a successful landing program.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX API and web scraping techniques from Wikipedia
- Perform data wrangling
  - Processed via one-hot encoding was applied to categorical features in data
- 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 using ML prediction

Data Collection

- The data was collected via multiple routes:
- Data collection was started using ‘get request’ function through the SpaceX API.
- Next, the response content was decoded as a Json using ‘.json() function call’ and converted into a pandas dataframe using ‘.json_normalize()’.
- The data was cleaned and checked for missing values and to fill in the missing values where needed.
- In addition, web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup was conducted.
- The aim was to extract the launch records data as HTML table, parse the table and convert it to a pandas dataframe for further analysis.

Data Collection – SpaceX API

- The ‘get request’ function was used to collect SpaceX API data, clean the data and conduct basic data wrangling and formatting.
- The link to the notebook is : https://github.com/mirzaahmed93/IBMCapstoneSpaceXProject/blob/main/Data%20Collection%20API%20Lab.ipynb

To make the requested JSON results more consistent, we will use the following static response object for this project:

In [9]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

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

In [10]:
```
response.status_code
```

Out[10]: 200

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

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

Using the dataframe `data` print the first 5 rows

In [12]:
```
# Get the head of the dataframe
data.head()
```

Out[12]:

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships	capsules	
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 33, 'altitude': None, 'reason': 'merlin engine failure'}]	Engine failure at 33 seconds and loss of vehicle	[]	[]	[]	[

Data Collection - Scraping

- Web scraping was applied to Falcon 9 launch records with the 'BeautifulSoup' function
- The data was parsed, set in a table and converted into a pandas dataframe.
- The link to the notebook is: https://github.com/mirzaahmed93/IBMCapstoneSpaceXProject/blob/main/Data%20Collection%20with%20Web%20Scraping%20lab.ipynb

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

import requests

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

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

In [6]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content

soup = BeautifulSoup(response, 'html5lib')
```

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>
```

Data Wrangling

- Data wrangling process was initiated. EDA was done to determine the training labels.
- The number of launches at each site, and the number and occurrence of each orbits was calculated.
- Landing outcome labels from outcome column were created and exported the results to.
- The link to the notebook is: https://github.com/mirzaahmed93/IBMCaps oneSpaceXProject/blob/main/Data%20Wra gling.ipynb

In [6]:
```
# Apply value_counts on Orbit column
df['Orbit'].value_counts()
```

Out[6]:
```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
SO        1
HEO       1
ES-L1     1
GEO       1
Name: Orbit, dtype: int64
```

TASK 3: Calculate the number and occurence of mission outcome per orbit type

Use the method .value_counts() on the column Outcome to determine the number of landing_outcomes.Then assign it to a variable landing_outcomes.

In [7]:
```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
landing_outcomes
```

Out[7]:
```
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
```

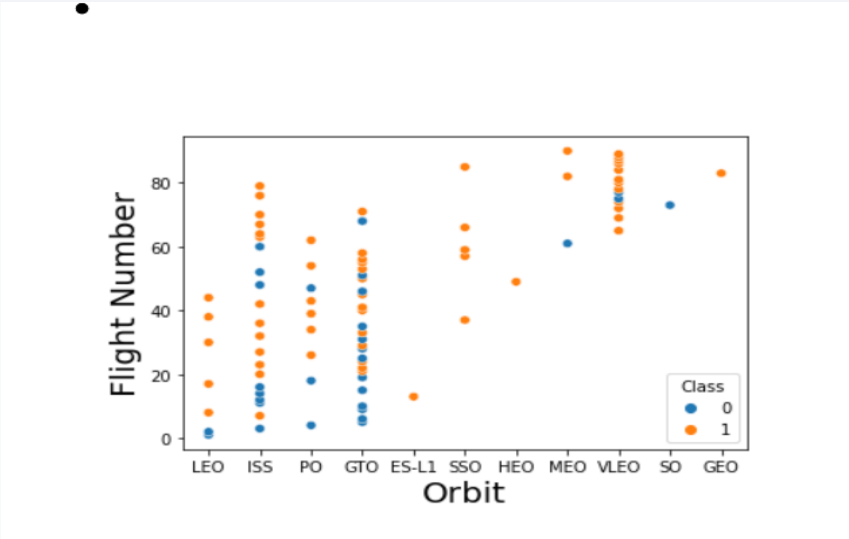
True Ocean means the mission outcome was successfully landed to a specific region of the ocean while False Ocean means the mission outcome was unsuccessfully landed to a specific region of the ocean. True RTLS means the mission outcome was successfully landed to a ground pad False RTLS means the mission outcome was unsuccessfully landed to a ground pad.True ASDS means the mission outcome was successfully landed to a drone ship False ASDS means the mission outcome was unsuccessfully landed to a drone ship. None ASDS and None None these represent a failure to land.

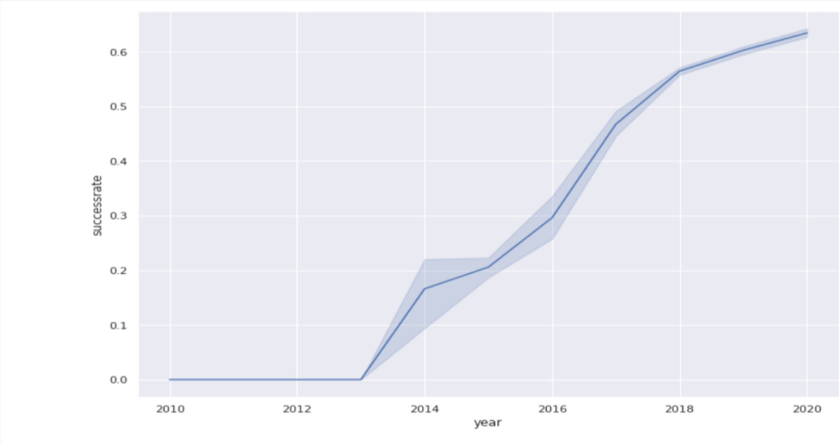
In [8]:
```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

```
0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 False Ocean
6 None ASDS
7 False RTLS
```

EDA with Data Visualization

- The data was analyzed by visualizing the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend.
- The link to the notebook is: https://github.com/mirzaahmed93/IBMCapstoneSpaceXProject/blob/main/EDA%20with%20Visualization%20lab.ipynb





EDA with SQL

- Using EDA with SQL to gain insight from the data, several queries to find out for instance:
  - The names of unique launch sites.
  - The total payload mass carried by boosters
  - The average payload mass carried by booster version F9 v1.1
  - The total number of successful and failure mission outcomes
- The link to the notebook is: https://github.com/mirzaahmed93/IBMCapstoneSpaceXProject/blob/main/EDA%20with%20SQL%20Lab.ipynb

Build an Interactive Map with Folium

- With folium, all launch sites were marked, and added map objects such as markers, circles, lines to mark the success or failure of launches.
- Launch outcomes (failure or success) were categorized to class 0 and 1 where 0 is failure, and 1 is success.
- Using the labeled marker clusters, launch sites with high success rates were identified.
- 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
- The link for the notebook is: https://github.com/mirzaahmed93/IBMCapstoneSpaceXProject/blob/main/Interactive%20Visual%20Analytics%20with%20Folium%20lab.ipynb

Build a Dashboard with Plotly Dash

- An interactive dashboard with Plotly dash was created.
- Pie charts showing the total launches by a certain sites were plotted.
- Scatter graphs showing the relationship with 'Outcome' and 'Payload Mass (Kg)' variables for the different booster versions were examined.
- The link to the notebook is: https://github.com/mirzaahmed93/IBMCapstoneSpaceXProject/blob/main/Build%20an%20Interactive%20Dashboard%20with%20Plotly%20Dash.ipynb

Predictive Analysis (Classification)

- The data was formatted using ‘numpy’ and ‘pandas’ and then was transformed. Finally, the data was split into training and testing algorithms.
- Different machine learning models were deployed and different hyperparameters for the variables using ‘GridSearchCV’ was used.
- Accuracy of the model was examined and then improved using feature engineering and algorithm tuning.
- The best performing classification model was used.
- The link to the notebook is:https://github.com/mirzaahmed93/IBMCapstoneSpaceXProject/blob/main/Machine%20Learning%20Prediction%20Lab.ipynb

Find the method performs best:

```
In [30]: algorithms = {'KNN':KNN_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',KNN_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
```

```
Best Algorithm is Tree with a score of 0.8892857142857145
Best Params is : {'criterion': 'entropy', 'max_depth': 2, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'best'}
```

Results

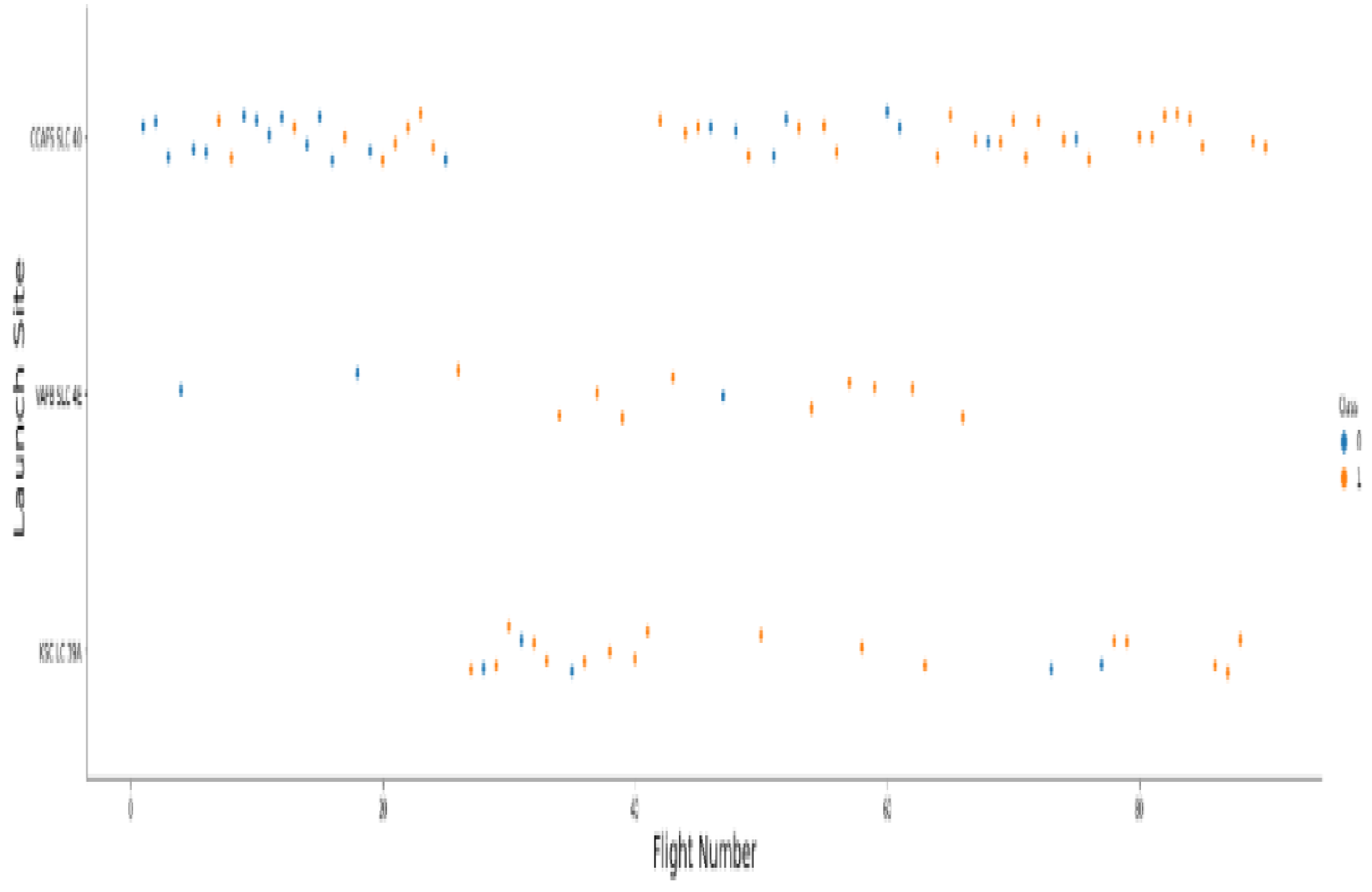
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

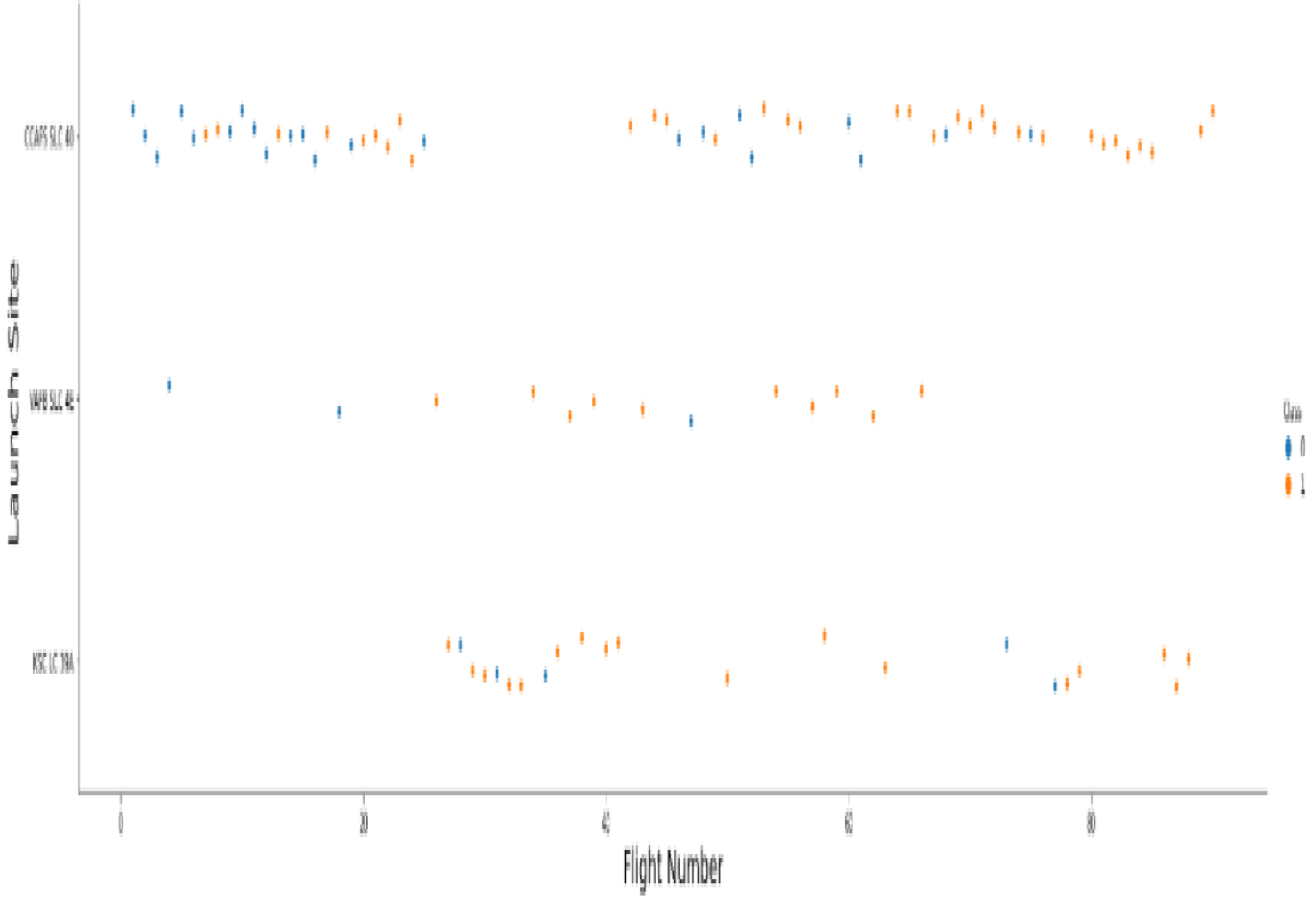
- From the plot, it was found that the larger the flight amount at a launch site, the greater the success rate at a launch site.
- This can be seen in the figure across



Payload vs. Launch Site

- Show The greater the payload mass for the launch site (CCAFS SLC 40) means the higher the success rate for the rockets.

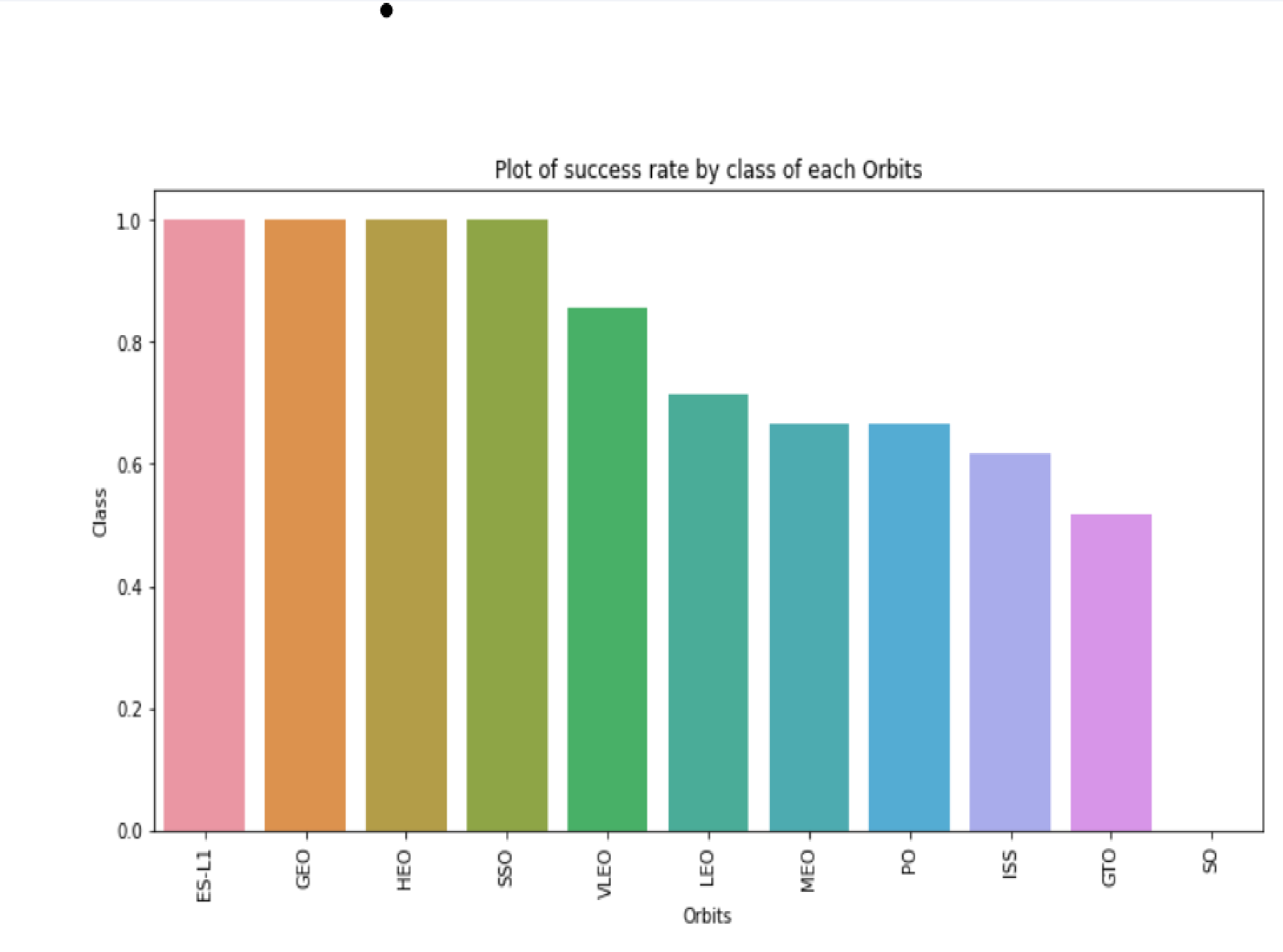
- Seen in the figure across



Success Rate vs. Orbit Type

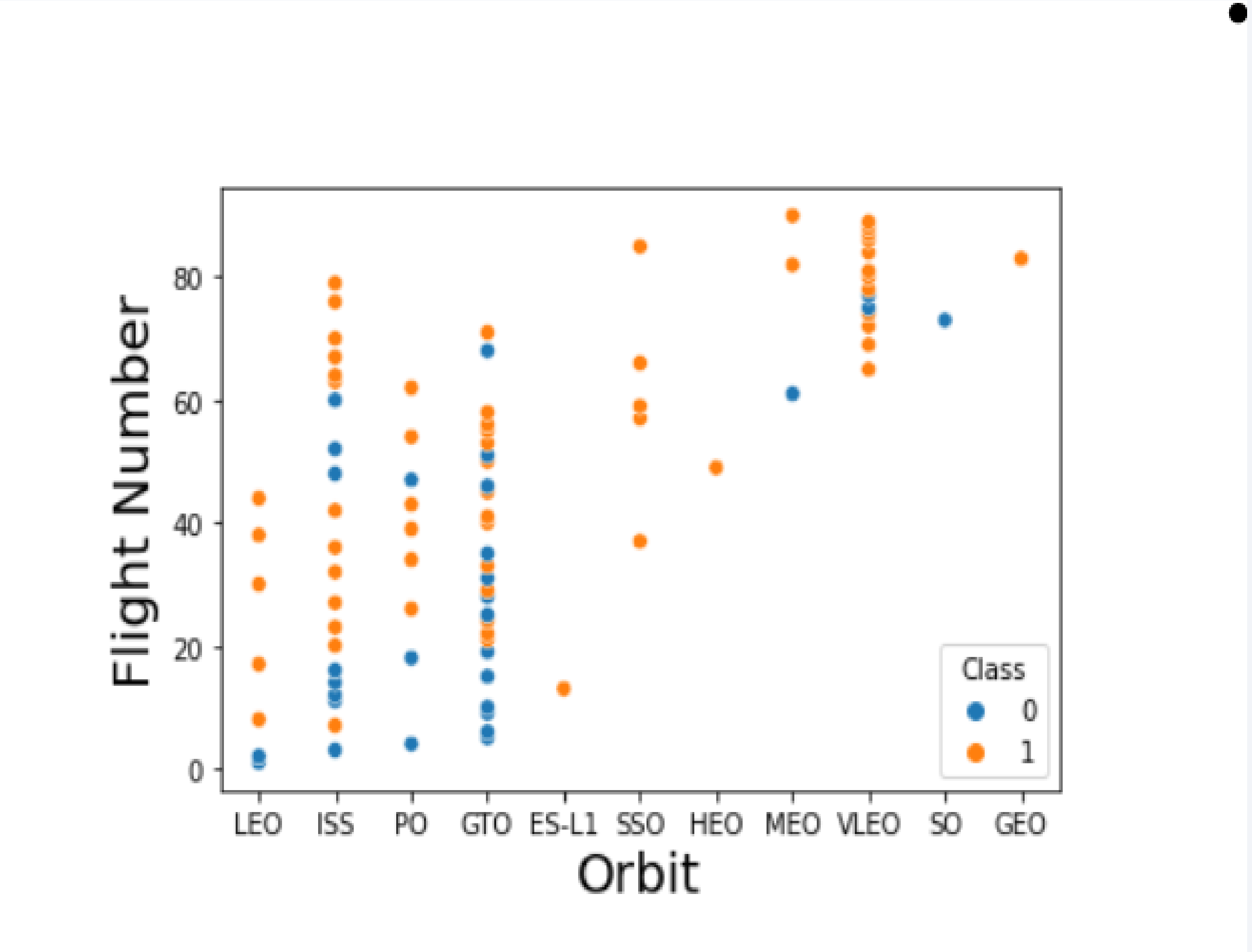
- Show a bar chart for the success rate of each orbit type

- The screenshot across displays this.



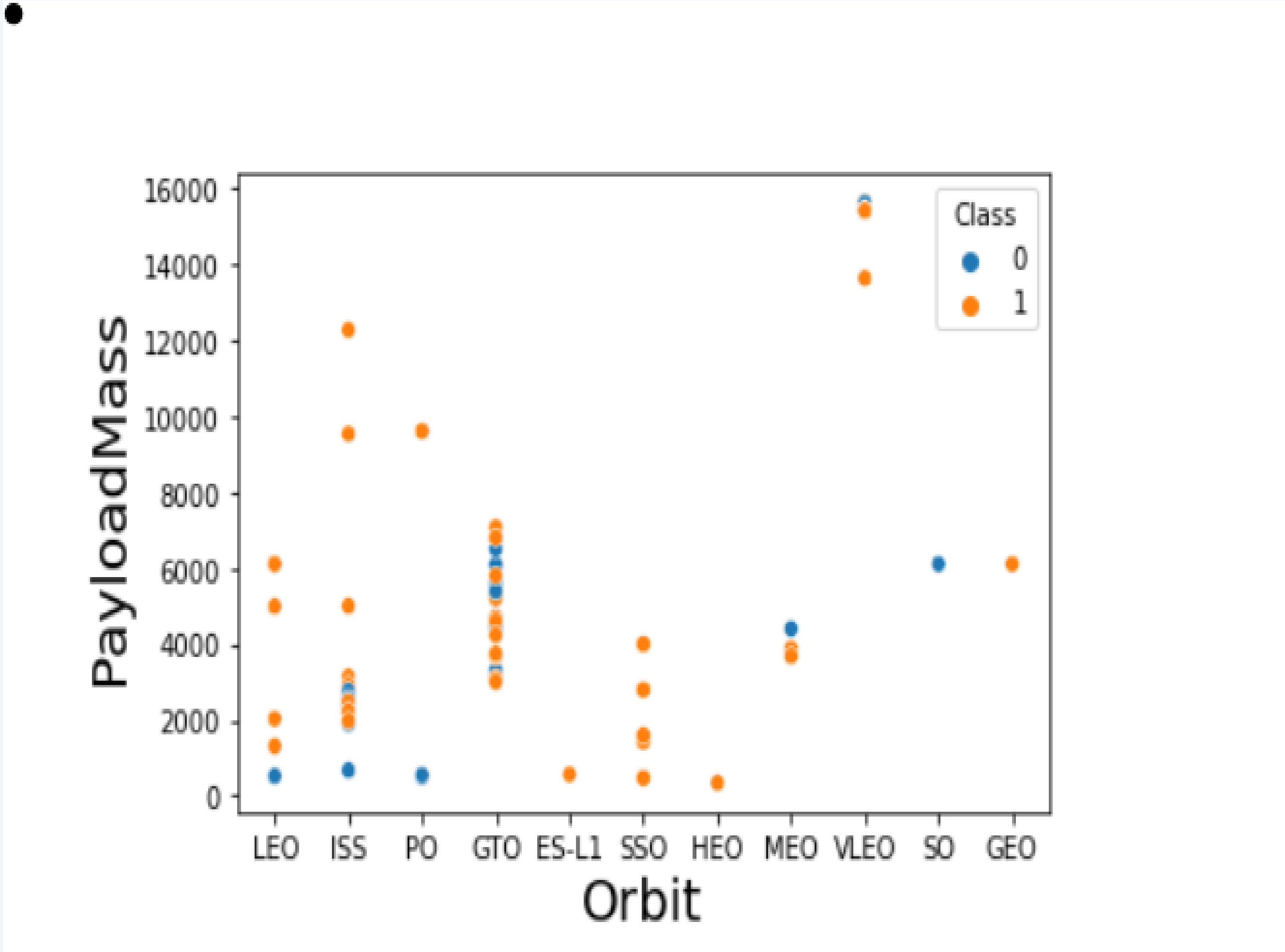
Flight Number vs. Orbit Type

- The plot across shows the 'Flight Number' vs. 'Orbit type'. In the LEO orbit category, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.



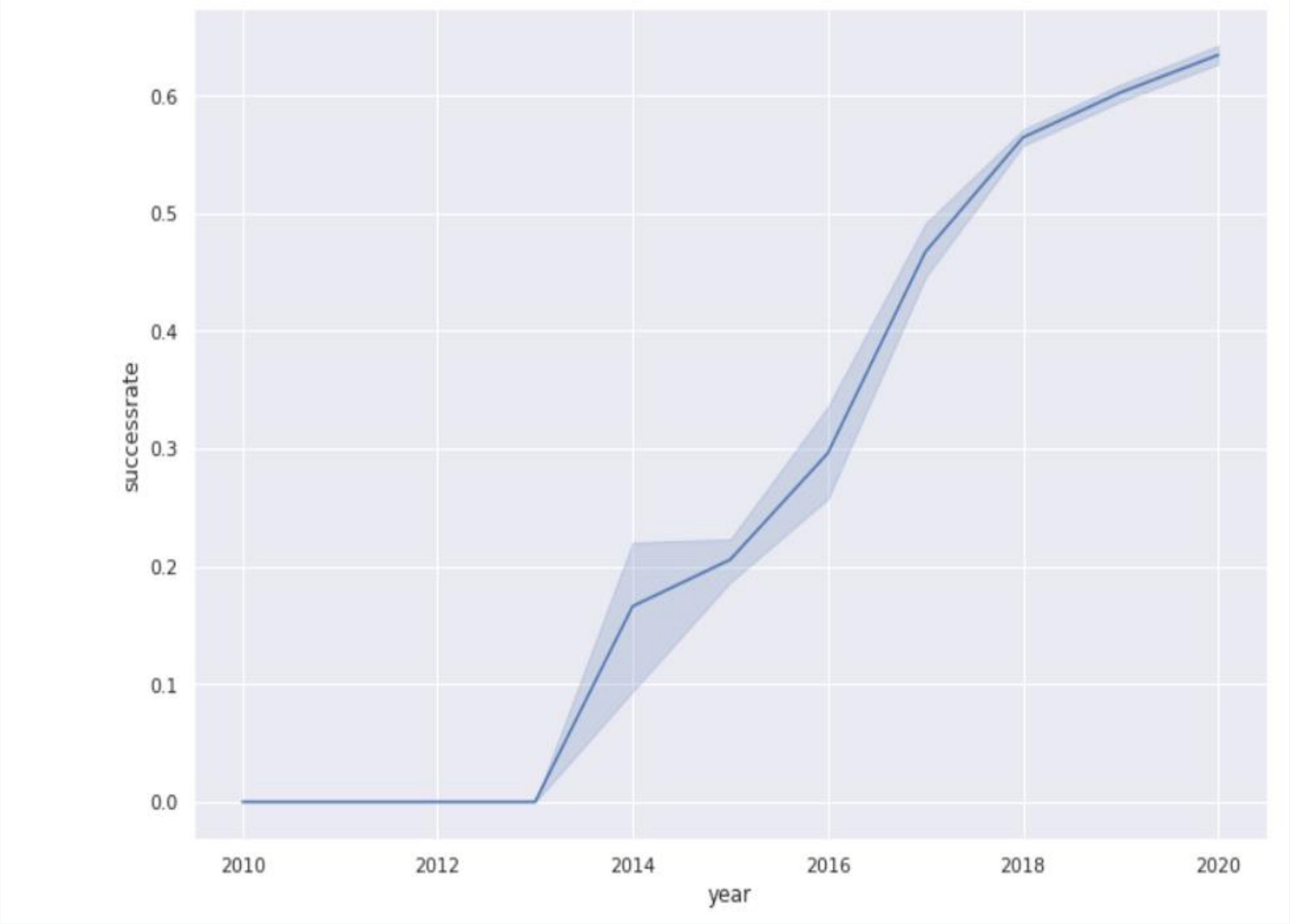
Payload vs. Orbit Type

- The scatterplot across shows ‘payloadmass’ by orbit types depending on launch outcomes.
- It can be seen that ‘GTO’ has had lots of success but at lower payload mass compared to ‘VLEO’!



Launch Success Yearly Trend

- The linegraph plot (across) demonstrates that success rates increasing from 2013 until 2020.



All Launch Site Names

- Was not able to connect ‘invalid syntax’ error ☹

Launch Site Names Begin with 'CCA'

- 5 records where launch sites begin with `CCA` were found (screen to the right)

	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

- The total payload carried by booster rockets from NASA was 45596 kg as seen with screen to the right

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

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

- Calculate the average payload mass carried by booster version F9 v1.1 is 2928.4 kg (screenshot attached)

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

	avg_payloadmass
0	2928.4

First Successful Ground Landing Date

- The first successful landing date on the ground pad was seen as 22/12/2015 (screenshot attached).

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

	firstsuccessfull_landing_date
0	2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- The '**WHERE**' function filtered for boosters which have successfully landed on drone ships and applied the '**AND**' function to determine successful landing with payload mass greater than 4000 kg but less than 6000 kg
- Screenshot attached

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

	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

- Syntax errors were generated ☹

Boosters Carried Maximum Payload

- The boosters that carried the maximum payload were found using a subquery in the '**WHERE**' function and the '**MAX()**' function. 11 boosters and their details were given.
- This can be seen with the screenshot attached.

	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 combinations of the '**WHERE**', '**LIKE**', '**AND**', and '**BETWEEN**' functions were used to filter for failed landing outcomes in drone ships and their booster versions along with the launch site names for year 2015.

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

```
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)
```

	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

- Landing outcomes and the '**COUNT**' of landing outcomes function from the data was used with the '**WHERE**' clause to filter for landing outcomes '**BETWEEN**' (functions) 2010-06-04 to 2010-03-20.
- Next, the '**GROUP BY**' function was implemented to group the landing outcomes. Then, the '**ORDER BY**' function to order the grouped landing outcomes in descending order was implemented.

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)
```

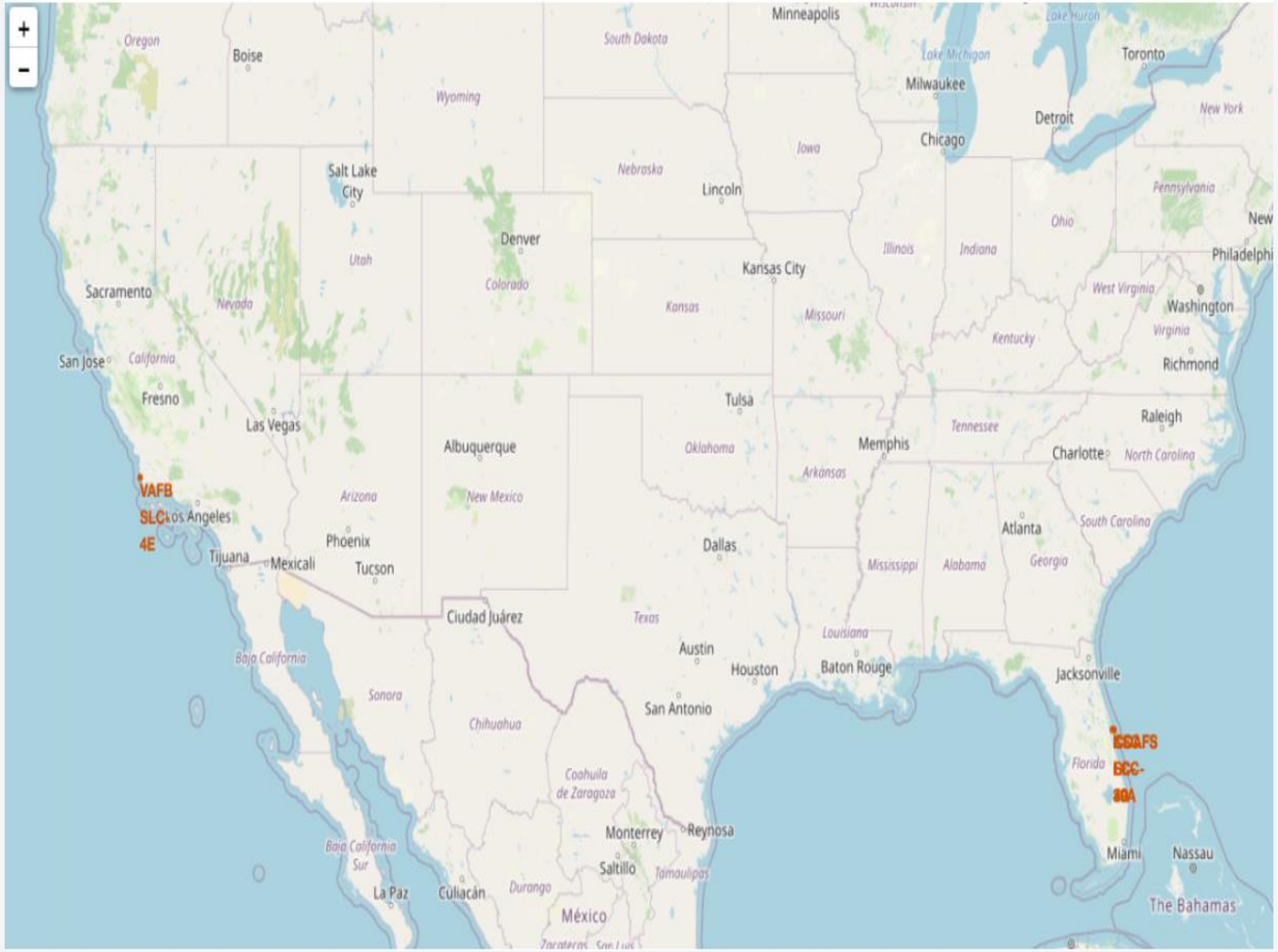
	landingoutcome	count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1
7	Failure (parachute)	1

Section 4

Launch Sites Proximities Analysis

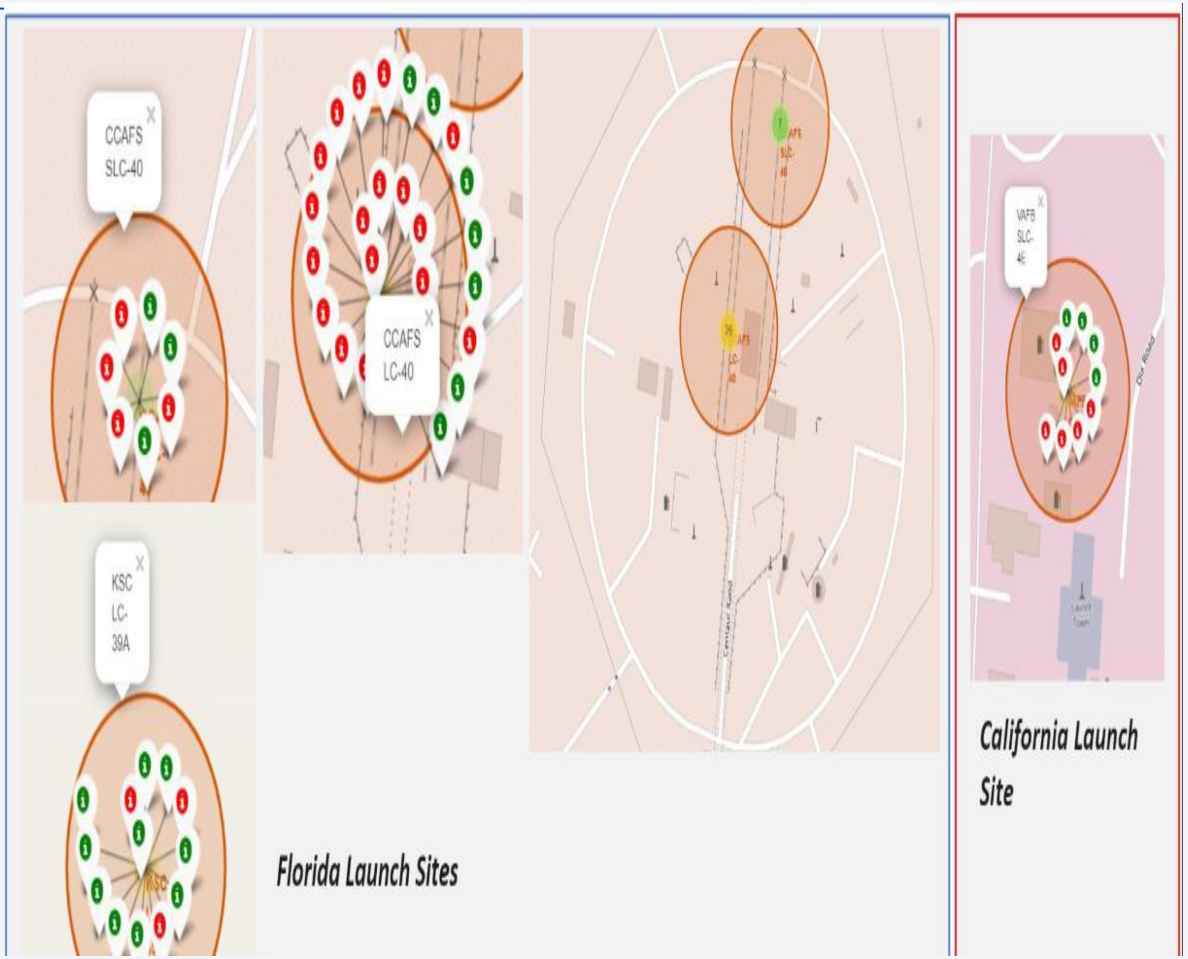
US site markers for rockets

- The folium map generated across (screenshot) depicts the launch sites for the rockets. It can be seen that the 2 main sites are in California and Florida



Launch outcomes colour scheme

- The screenshot across shows the success of the rockets by colour schemes. ‘Green’ indicates success whereas ‘red’ indicates failure



Launch site proximity to coastline

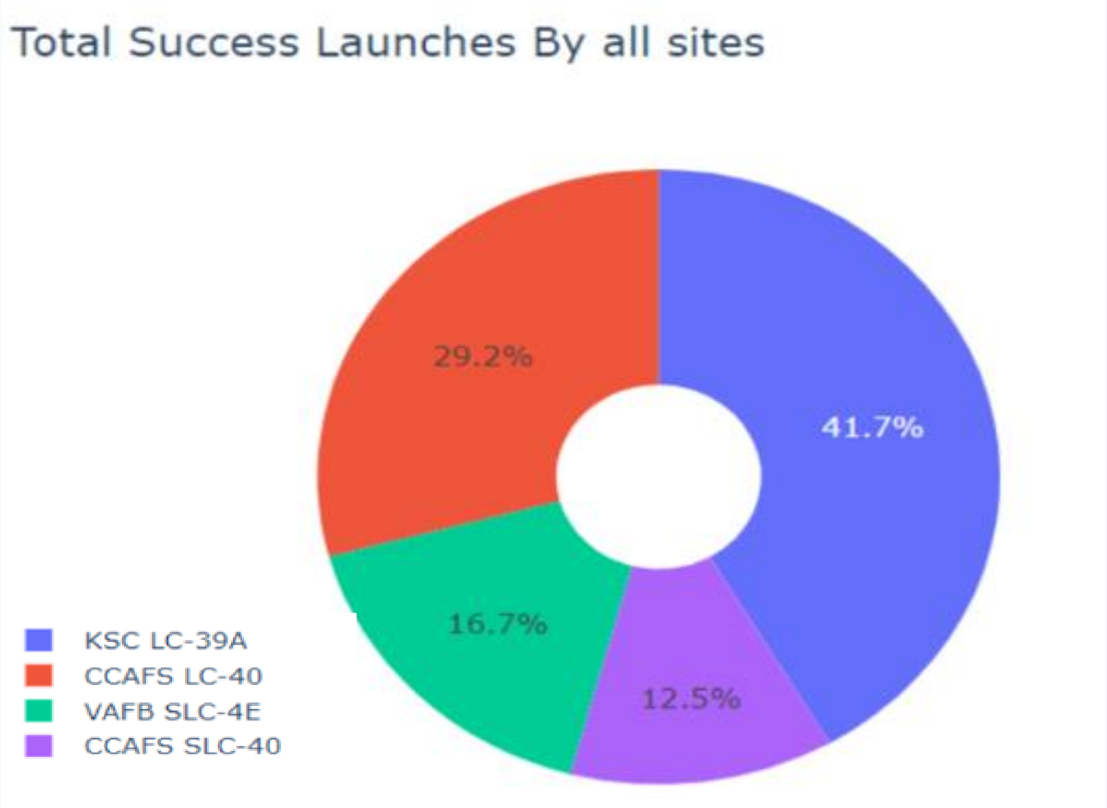
- The folium graphic shows thew distance from the launch site in Florida to the coast of being 90 km away.

Section 5

Build a Dashboard with Plotly Dash

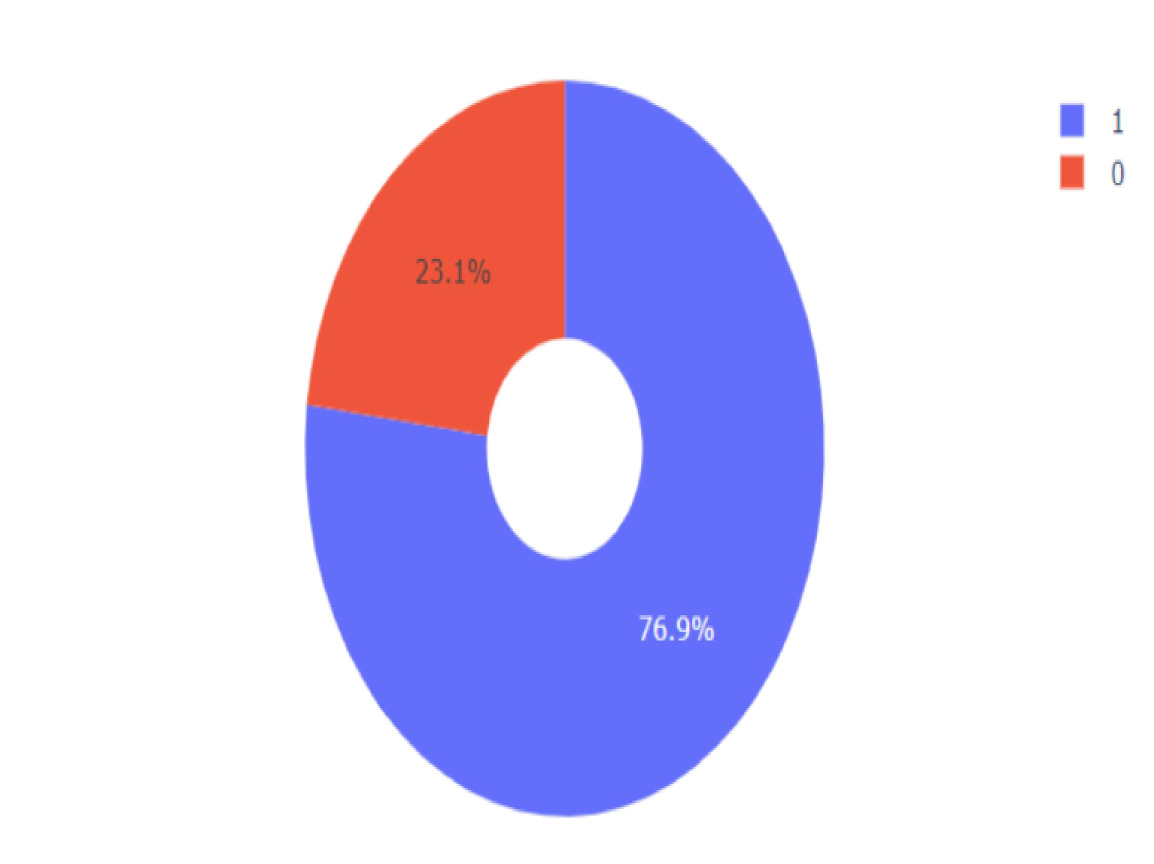
Launch success count pie chart

- The pie chart demonstrates that the 'KSC LC-39A' launch site is the most successful site with 41.7% of all successful launches coming from there.



Launch success ratio pie chart

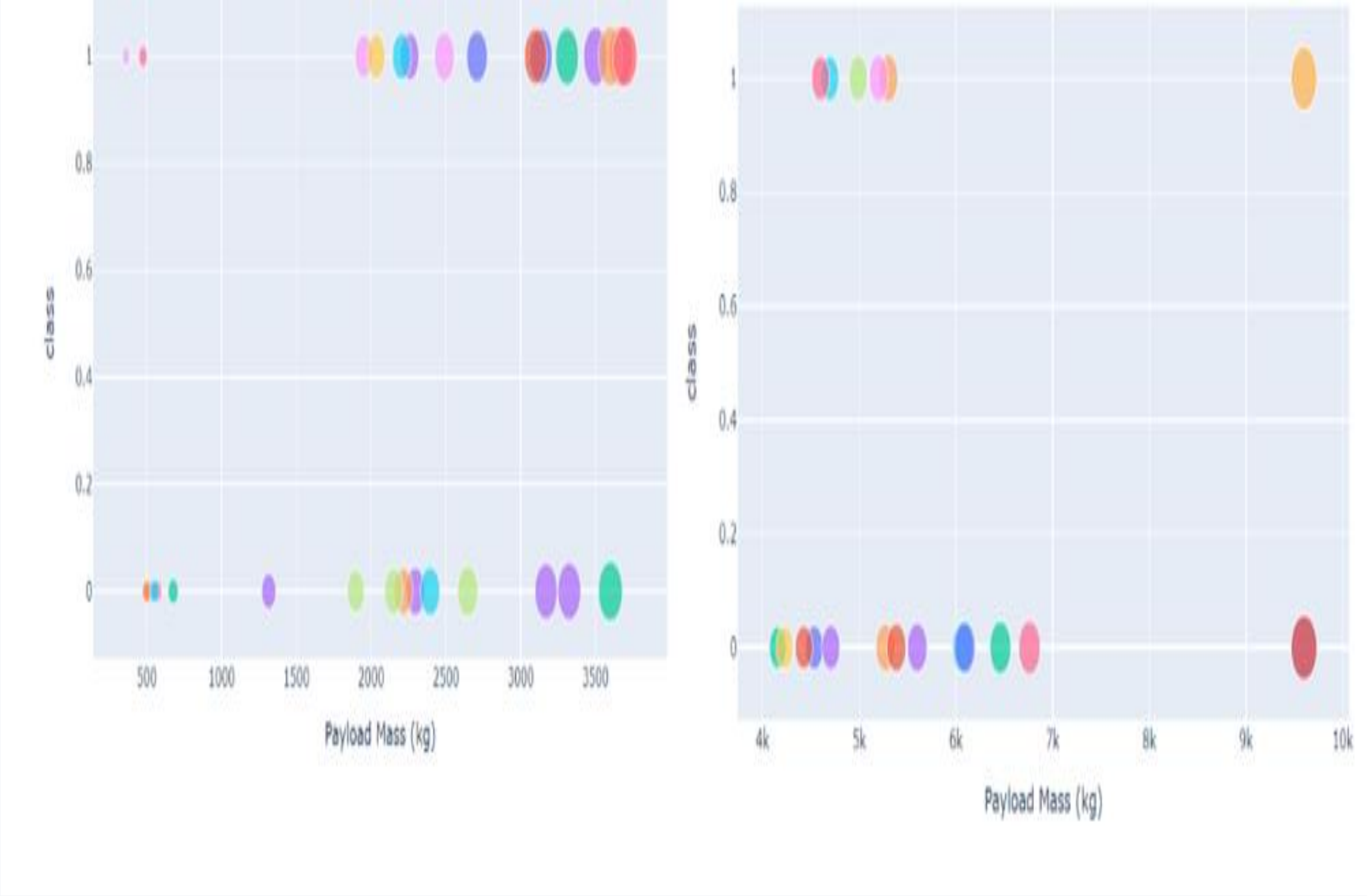
- The pie chart screenshot (across) illustrates that out of the most successful launch site (KSC LC-39A) approximately 77% of all launches were deemed as a success. Only 23% roughly were not successful.



Payload versus Launch Outcome scatterplot

- The lower payload (left diagram) has a higher success rate compared to the higher payload rate rockets.

- The y-axis depicts the 'class' of launches with the x-axis 'depicting payload mass (kg)'.



Section 6

Predictive Analysis (Classification)

Classification Accuracy

- The decision tree classifier is the model with the highest classification accuracy as the method that is the most accurate for ML.

Find the method performs best:

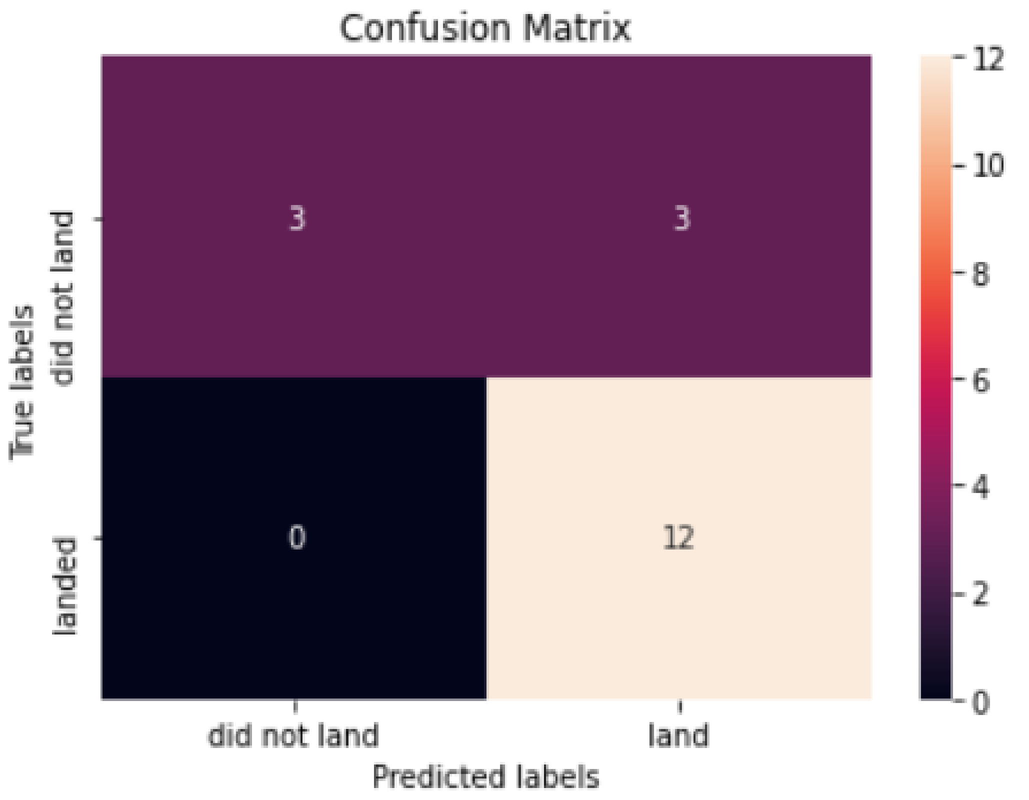
```
algorithms = {'KNN':KNN_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',KNN_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
```

```
Best Algorithm is Tree with a score of 0.8892857142857145
Best Params is : {'criterion': 'entropy', 'max_depth': 2, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'best'}
```

Confusion Matrix

- The confusion matrix for the decision tree classifier model shows that the classifier can distinguish between the different classes. The major problem is the false positives generated ('did not land' landing marked as 'land' by the classifier) was approximately 1/6th of all the labels generated.

```
yhat = KNN_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Conclusions

- The larger the flight amount at a launch site, the greater the success rate.
- Launch success rates became noticeable from 2013-2020.
- Orbits ‘ES-L1’, ‘GEO’, ‘HEO’, ‘SSO’, ‘VLEO’ had the most success rate.
- ‘KSC LC-39A’ had the most successful launches of any sites.
- The launch sites (Florida) is not that far from the coastline but is further away from other significant landmarks (e.g., highways)
- The Decision tree classifier is the best machine learning algorithm for this task.

Thank you!