IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Arjuna4b18
17-07-2023

Outline

Executive Summary

- Summary of methodologies
  - 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
- Commercial Space Age is Here
- Space X has best pricing ($62 million vs. $165 million USD)
- Largely due to ability to recover part of rocket (Stage 1)
- Space Y wants to compete with Space X

- Problems you want to find answers
- What factors determine if the rocket will land successfully?
- The interaction amongst various features that determine the success rate of a successful landing.
- What operating conditions needs to be in place to ensure a successful landing program.

SpaceX Falcon 9 Rocket – The Verge

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Combined data from SpaceX public API and SpaceX Wikipedia page
- Perform data wrangling
  - Classifying true landings as successful and unsuccessful otherwise
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Tuned models using GridSearchCV

Data Collection

- The data was collected using various methods
  - Data collection was done using get request to the SpaceX API.
  - Next, we decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().
  - We then cleaned the data, checked for missing values and fill in missing values where necessary.
  - In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
  - The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis.

Data Collection – SpaceX API

- We used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.
- The link to the notebook is

https://github.com/EgaEdx/testrepo/blob/master/jupyter-labs-spacex-data-collection-api%20(2).ipynb

Task 1: Request and parse the SpaceX launch data using the GET request

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_
```

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()
```

Data Collection - Scraping

- We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup
- We parsed the table and converted it into a pandas dataframe.
- The link to the notebook is https://github.com/EgaEdx/testrepo/blob/master/jupyter-labs-webscraping.ipynb

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 reference link towards the end of this lab

In [8]:
```python
# 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.

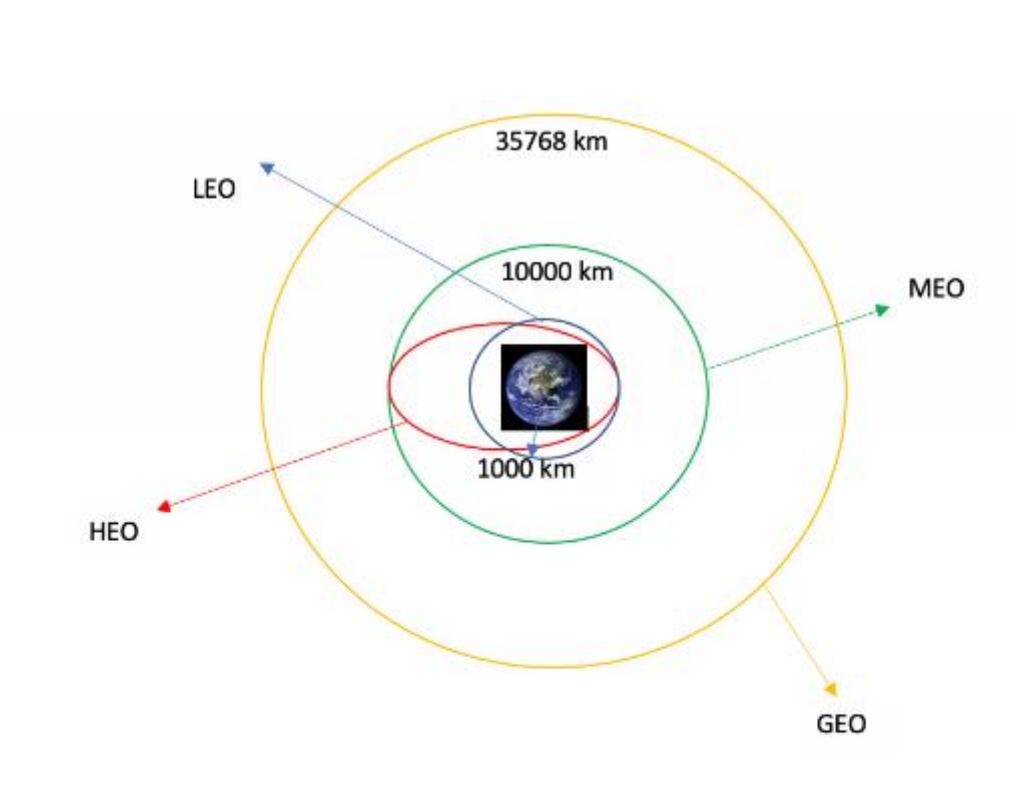
In [9]:
```python
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

```
<table class="wikitable plainrowheaders collapsible" style="width: 100%;">
<tbody><tr>
<th scope="col">Flight No.
</th>
<th scope="col">Date and<br/>time (<a href="/wiki/Coordinated_Universal_Time" title="Coordinated Universal Time">UTC</a>)
</th>
<th scope="col"><a href="/wiki/List_of_Falcon_9_first-stage_boosters" title="List of Falcon 9 first-stage boosters">Version,<b
r/>Booster</a> <sup class="reference" id="cite_ref-booster_11-0"><a href="#cite_note-booster-11">[b]</a></sup>
</th>
<th scope="col">Launch site
</th>
<th scope="col">Payload<sup class="reference" id="cite_ref-Dragon_12-0"><a href="#cite_note-Dragon-12">[c]</a></sup>
</th>
<th scope="col">Payload mass
</th>
<th scope="col">Orbit
```

Data Wrangling

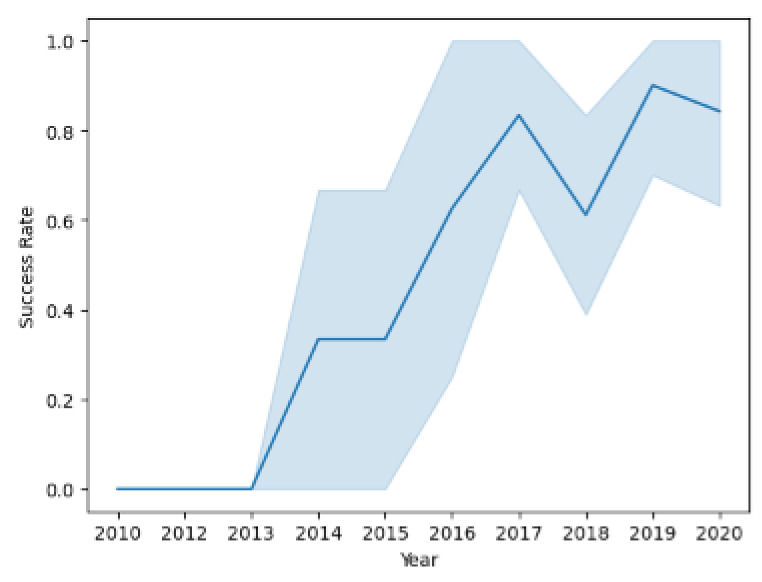
- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is

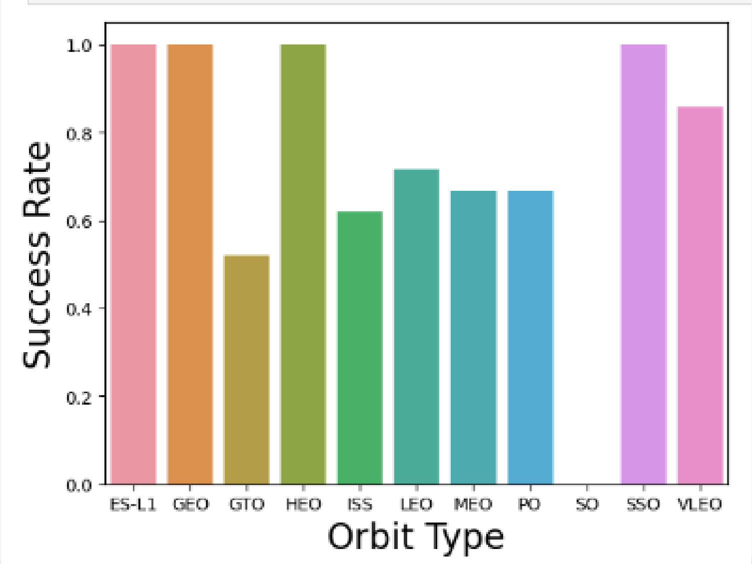
https://github.com/EgaEdx/testrepo/blob/master/labs-jupyter-spacex-data_wrangling_jupyterlite.jupyterlite.ipynb



EDA with Data Visualization

- We explored the data 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/EgaEdx/testrepo/blob/master/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

- We loaded the SpaceX dataset into a PostgreSQL database without leaving the jupyter notebook.
- We applied EDA with SQL to get insight from the data. We wrote queries to find out for instance:
  - The names of unique launch sites in the space mission.
  - The total payload mass carried by boosters launched by NASA (CRS)
  - The average payload mass carried by booster version F9 v1.1
  - The total number of successful and failure mission outcomes
  - The failed landing outcomes in drone ship, their booster version and launch site names.
- The link to the notebook is: https://github.com/EgaEdx/testrepo/blob/master/jupyter-labs-eda-sql-edx_sqllite.ipynb

Task 2

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

In [9]:
```
%sql SELECT * FROM 'SPACEXTBL' WHERE Launch_Site LIKE 'KSC%' LIMIT 5;
```

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

Out[9]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Out
19/02/2017	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490.0	LEO (ISS)	NASA (CRS)	Success	Success (g
16/03/2017	6:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	5600.0	GTO	EchoStar	Success	No at
30/03/2017	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300.0	GTO	SES	Success	Success (
05/01/2017	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300.0	LEO	NRO	Success	Success (g
15/05/2017	23:21:00	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070.0	GTO	Inmarsat	Success	No at

Task 3

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

In [10]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) as "Total Payload Mass(Kgs)", Customer FROM 'SPACEXTBL' WHERE Customer = 'NASA (CRS)'
```

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

Out[10]:

Total Payload Mass(Kgs)	Customer
45596.0	NASA (CRS)

Task 4

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

Build an Interactive Map with Folium

- We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the 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.
- https://github.com/EgaEdx/testrepo/blob/master/lab_jupyter_launch_site_location.jupyterlite.ipynb

A city map symbol may look like this:

In [18]:
```python
# Create a marker with distance to a closest city, railway, highway, etc.
# Draw a line between the marker to the launch site
closest_highway = 28.56335, -80.57085
closest_railroad = 28.57206, -80.58525
closest_city = 28.10473, -80.64531
```

In [19]:
```python
distance_highway = calculate_distance(launch_site_lat, launch_site_lon, closest_highway[0], closest_highway[1])
print('distance_highway =',distance_highway, ' km')
distance_railroad = calculate_distance(launch_site_lat, launch_site_lon, closest_railroad[0], closest_railroad[1])
print('distance_railroad =',distance_railroad, ' km')
distance_city = calculate_distance(launch_site_lat, launch_site_lon, closest_city[0], closest_city[1])
print('distance_city =',distance_city, ' km')
```

```
distance_highway = 0.584405688237401  km
distance_railroad = 1.282798438953375  km
distance_city = 51.43547596583314  km
```

In [20]:
```python
# closest highway marker
distance_marker = folium.Marker(
   closest_highway,
   icon=DivIcon(
       icon_size=(20,20),
       icon_anchor=(0,0),
       html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % "{:10.2f} KM".format(distance_highway),
       )
   )
site_map.add_child(distance_marker)
```

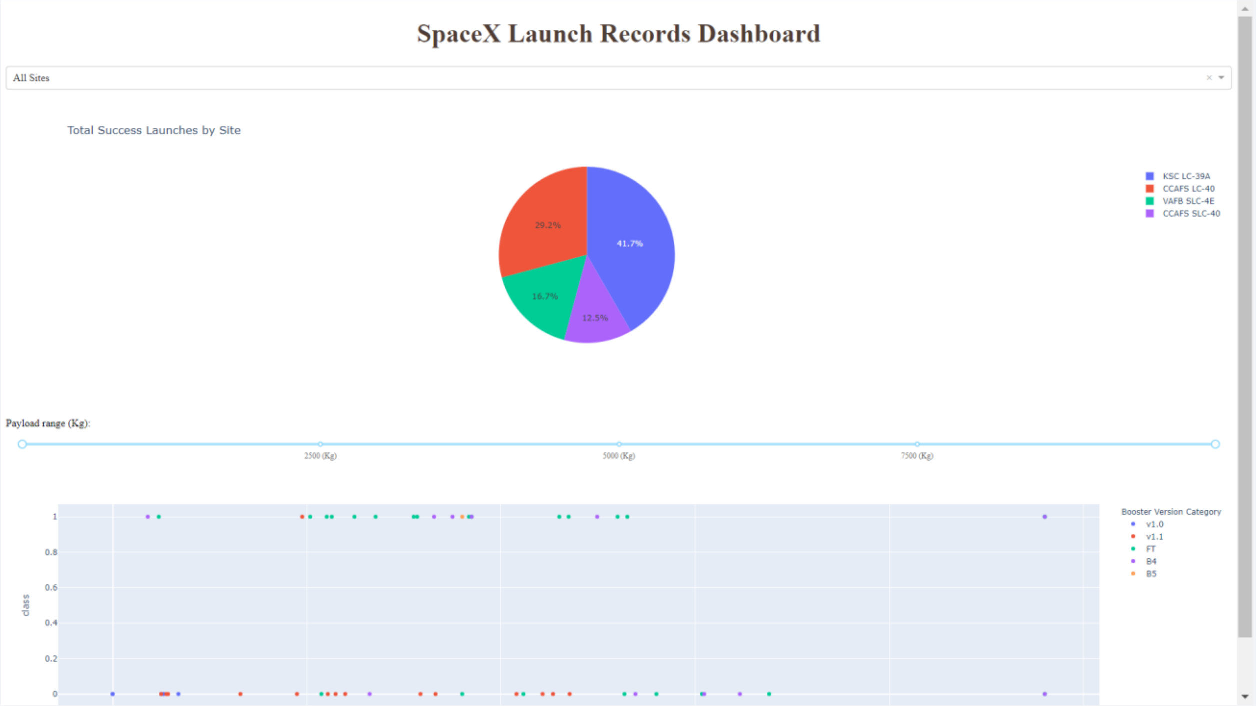
Build a Dashboard with Plotly Dash

- Dashboard includes a pie chart and a scatter plot.
- Pie chart can be selected to show distribution of successful landings across all launch sites and can be selected to show individual launch site success rates.
- Scatter plot takes two inputs: All sites or individual site and payload mass on a slider between 0 and 10000 kg.
- The pie chart is used to visualize launch site success rate.
- The scatter plot can help us see how success varies across launch sites, payload mass, and
- booster version category.

Predictive Analysis (Classification)

- We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.
- The link to the notebook is:
- https://github.com/EgaEdx/testrepo/blob/master/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results



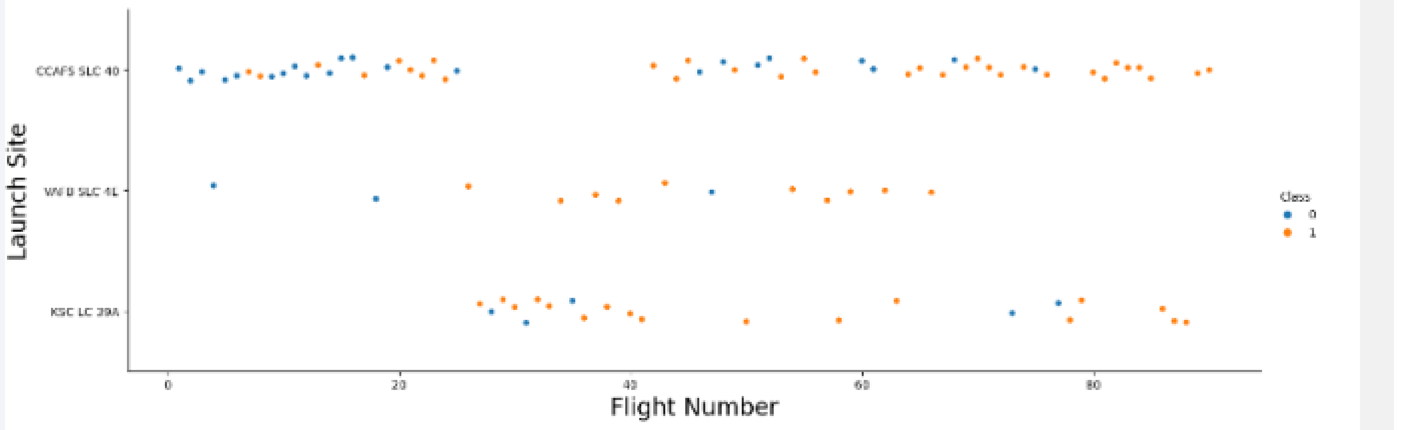
- This is a preview of the Plotly dashboard. The following sides will show the results of EDA with visualization, EDA with SQL, Interactive Map with Folium, and finally the results of our model with about 83% accuracy.

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

- Show a scatter plot of Flight Number vs. Launch Site



- Show the screenshot of the scatter plot with explanations

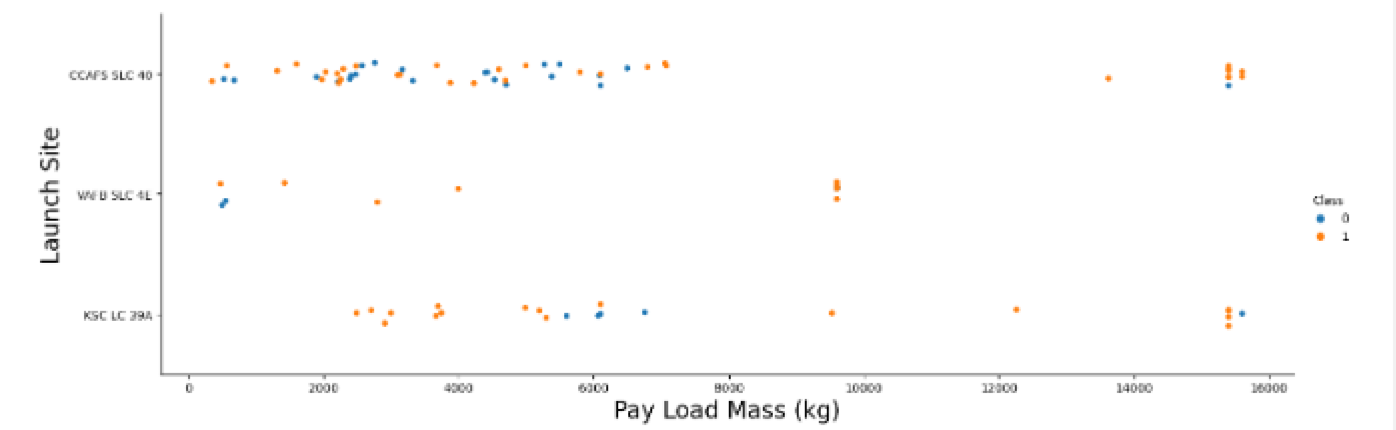
In [5]:
```
### TASK 1: Visualize the relationship between Flight Number and Launch Site
```

Use the function `catplot` to plot `FlightNumber` vs `LaunchSite`, set the parameter `x` parameter to `FlightNumber`,set the `y` to `Launch Site` and set the parameter `hue` to `'class'`

In [6]:
```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class v
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 3)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```

Payload vs. Launch Site

- Show a scatter plot of Payload vs. Launch Site



- Show the screenshot of the scatter plot with explanations

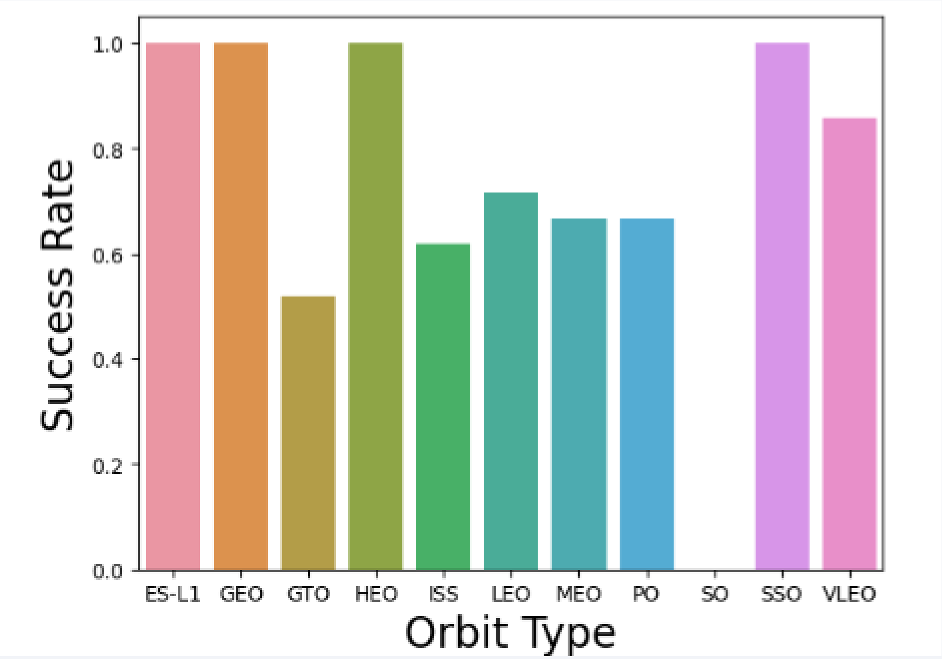
```
In [ ]: ### TASK 2: Visualize the relationship between Payload and Launch Site
```

We also want to observe if there is any relationship between launch sites and their payload mass.

```
In [7]: # Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the cl
        sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 3)
        plt.xlabel("Pay Load Mass (kg)",fontsize=20)
        plt.ylabel("Launch Site",fontsize=20)
        plt.show()
```

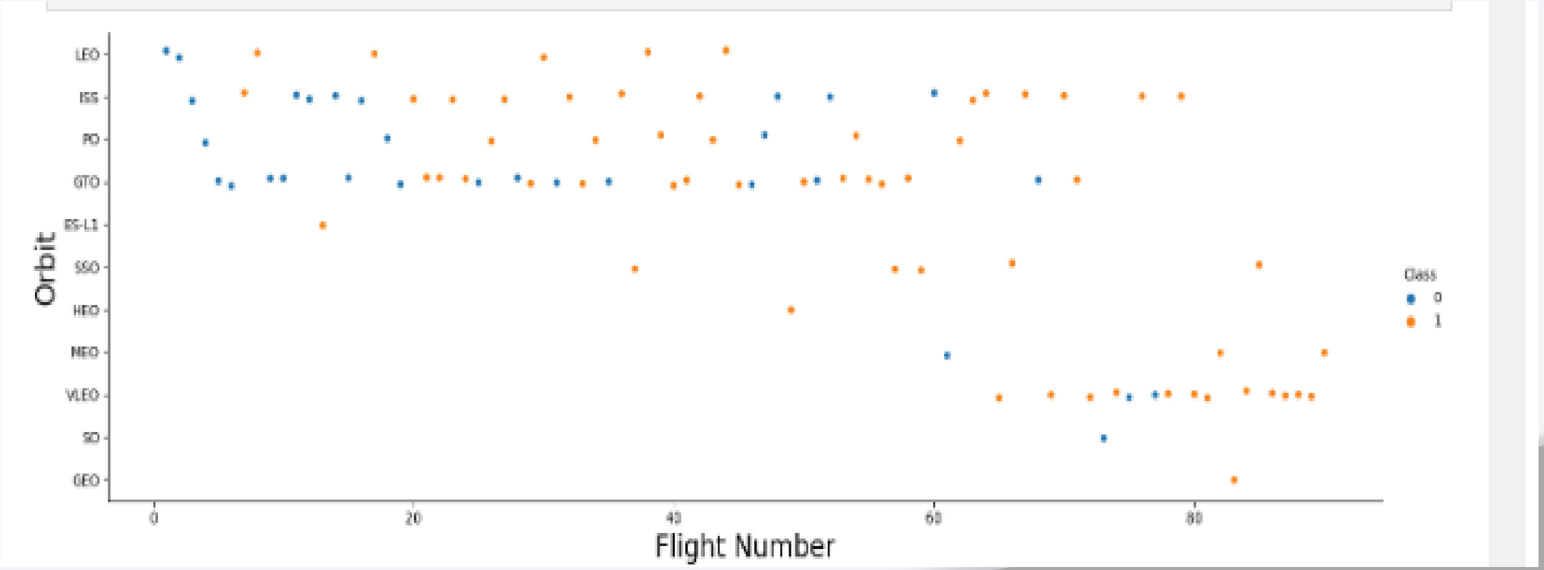
Success Rate vs. Orbit Type

- Show a bar chart for the success rate of each orbit type
- From the plot, we can see that ES-L1, GEO, HEO, SSO, VLEO had the most success rate.



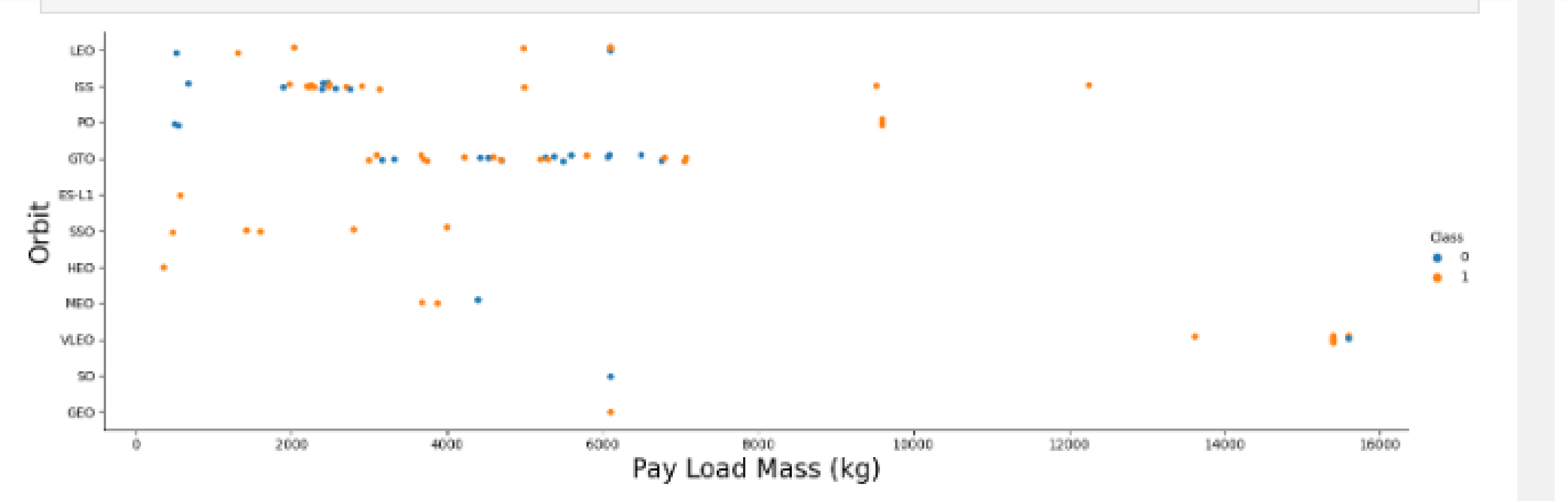
Flight Number vs. Orbit Type

- Show a scatter point of Flight number vs. Orbit type
- The plot below shows the Flight Number vs. Orbit type. We observe that in the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit



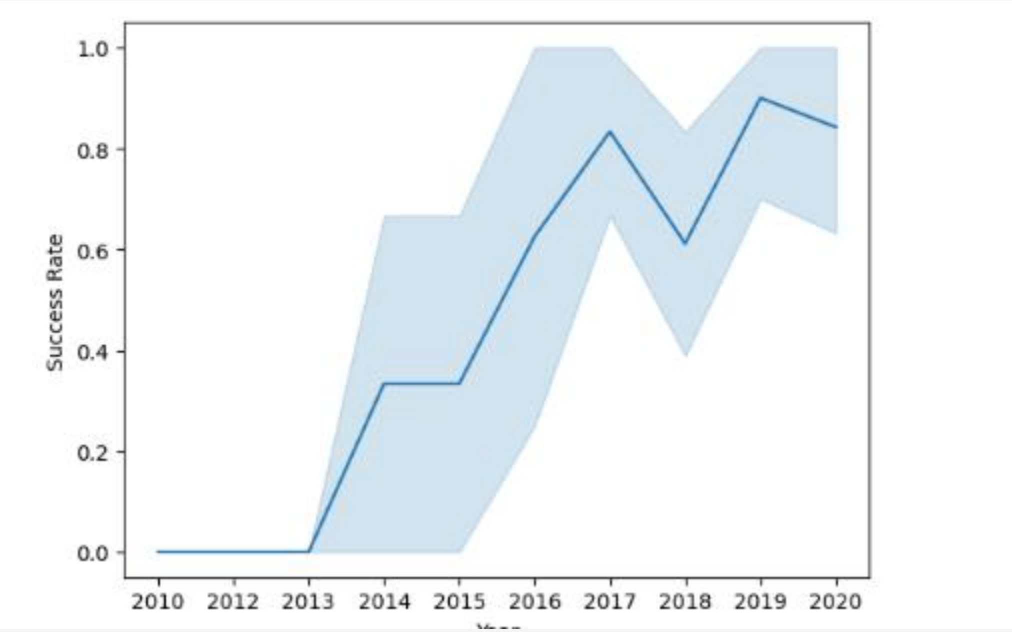
Payload vs. Orbit Type

- Show a scatter point of payload vs. orbit type
- We can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.



Launch Success Yearly Trend

- Show a line chart of yearly average success rate
- From the plot, we can observe that success rate since 2013 kept on increasing till 2020.



All Launch Site Names

- Find the names of the unique launch sites
- We used the key word DISTINCT to show only unique launch sites from the SpaceX data.
- Present your query result with a short explanation here

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 'KSC'

- We used the query above to display 5 records where launch sites begin with `KSC`

Task 2

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

In [9]:
```
%sql SELECT * FROM 'SPACEXTBL' WHERE Launch_Site LIKE 'KSC%' LIMIT 5;
```

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

Out[9]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Out
19/02/2017	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490.0	LEO (ISS)	NASA (CRS)	Success	Success (g
16/03/2017	6:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	5600.0	GTO	EchoStar	Success	No at
30/03/2017	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300.0	GTO	SES	Success	Success (
05/01/2017	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300.0	LEO	NRO	Success	Success (g
15/05/2017	23:21:00	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070.0	GTO	Inmarsat	Success	No at

Total Payload Mass

- We calculated the total payload carried by boosters from NASA as 45596 using the query below

Task 3

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

In [10]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) as "Total Payload Mass(Kgs)", Customer FROM 'SPACEXTBL' WHERE Customer = 'NASA (CRS)'
```

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

Out[10]:

Total Payload Mass(Kgs)	Customer
45596.0	NASA (CRS)

Average Payload Mass by F9 v1.1

- We calculated the average payload mass carried by booster version F9 v1.1 as 2928.4

Task 4

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

In [11]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) as "Payload Mass Kgs", Customer, Booster_Version FROM 'SPACEXTBL' WHERE BOOSTER_VERSIC
```

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

Out[11]:

Payload Mass Kgs	Customer	Booster_Version
2928.4	SES	F9 v1.1

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on drone ship.Present your query result with a short explanation here

Task 5

List the date where the succesful landing outcome in drone ship was acheived.

Hint:Use min function

In [12]:

```
%sql SELECT MIN(DATE) FROM 'SPACEXTBL' WHERE "Landing _Outcome" = "Success (drone ship)";
```

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

Out[12]:

MIN(DATE)
None

Successful Drone Ship Landing with Payload between 4000 and 6000

- We used the **WHERE** clause to filter for boosters which have successfully landed on drone ship and applied the **AND** condition to determine successful landing with payload mass greater than 4000 but less than 6000

Task 6

List the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000

In [14]:
```
%sql select BOOSTER_VERSION from SPACEXTBL where Landing_Outcome = 'Success (ground pad)' and PAYLOAD_MASS__KG_ > 4000 ar
```

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

Out[14]:

Booster_Version
F9 FT B1032.1
F9 B4 B1040.1
F9 B4 B1043.1

Total Number of Successful and Failure Mission Outcomes

- Total number of successful and failure mission outcomes below.

Task 7

List the total number of successful and failure mission outcomes

In [15]:
```
%sql SELECT "Mission_Outcome", COUNT("Mission_Outcome") as Total FROM SPACEXTBL GROUP BY "Mission_Outcome";
```

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

Out[15]:

Mission_Outcome	Total
None	0
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

Task 8

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

In [16]:
```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
```

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

Out[16]:

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

2017 Launch Records

- List the records which will display the month names, succesful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017

Task 9

List the records which will display the month names, succesful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017

Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2017' for year.

In [25]:

```
%sql SELECT substr(Date,7,4), substr(Date, 4, 2), \
    LANDING_OUTCOME AS LANDING_OUTCOME, \
    BOOSTER_VERSION AS BOOSTER_VERSION, \
    LAUNCH_SITE AS LAUNCH_SITE \
    FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (ground pad)' AND substr(Date,7,4) = '2017';
```

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

Out[25]:

substr(Date,7,4)	substr(Date, 4, 2)	LANDING_OUTCOME	BOOSTER_VERSION	LAUNCH_SITE
2017	02	Success (ground pad)	F9 FT B1031.1	KSC LC-39A
2017	01	Success (ground pad)	F9 FT B1032.1	KSC LC-39A
2017	03	Success (ground pad)	F9 FT B1035.1	KSC LC-39A
2017	08	Success (ground pad)	F9 B4 B1039.1	KSC LC-39A
2017	07	Success (ground pad)	F9 B4 B1040.1	KSC LC-39A
2017	12	Success (ground pad)	F9 FT B1035.2	CCAFS SLC-40

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

- Rank the count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order

Task 10

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

In [33]:

```
%sql SELECT "DATE", COUNT(LANDING_OUTCOME) as COUNT FROM SPACEXTBL \
    WHERE "DATE" BETWEEN '04-06-2010' and '20-03-2017';
```

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

Out[33]:

Date	COUNT
06/04/2010	56

Section 3

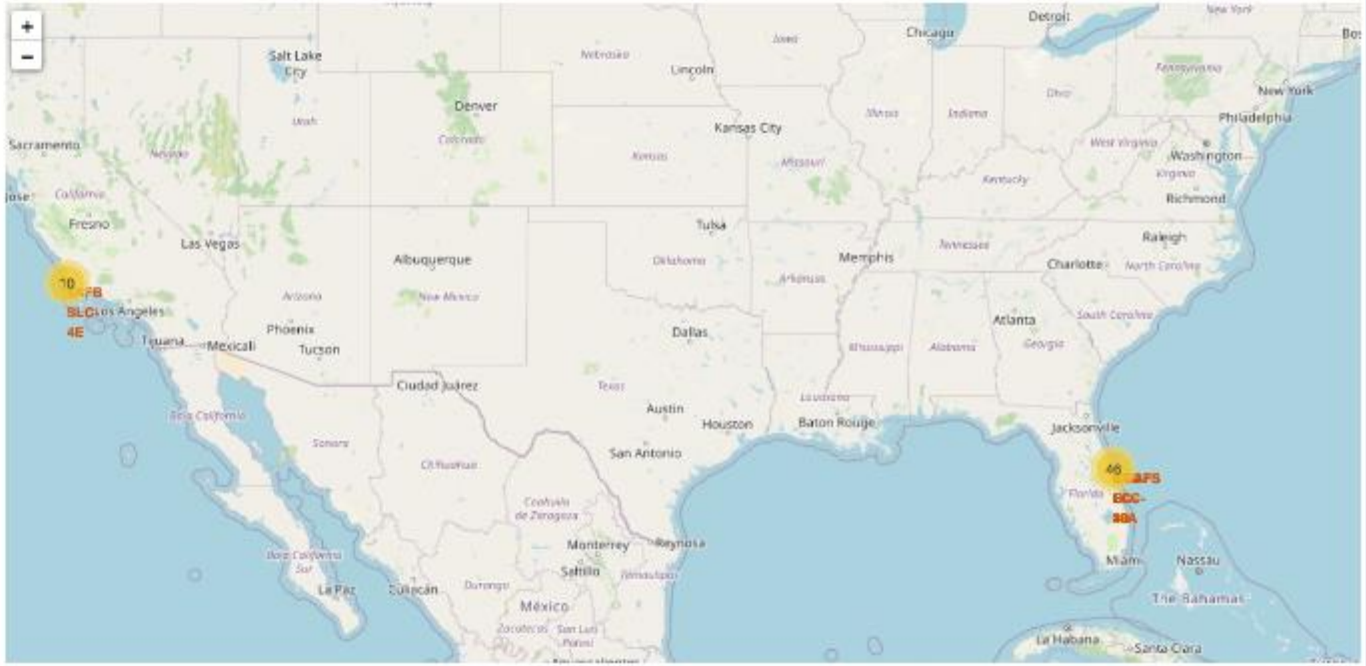
Launch Sites Proximities Analysis

Folium Map Screenshot

The generated map with marked launch sites should look similar to the following:

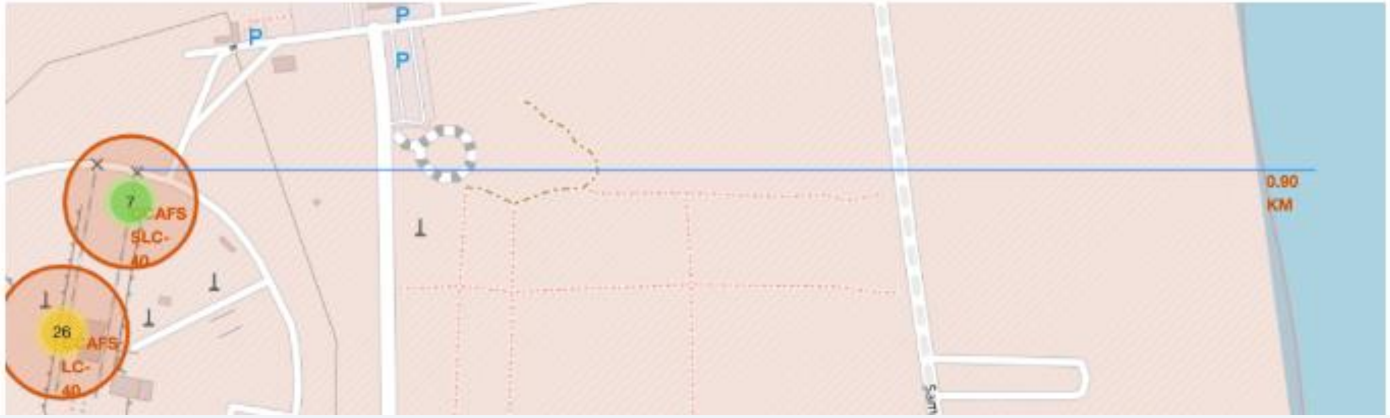
Folium Map Screenshot

Your updated map may look like the following screenshots:



Folium Map Screenshot

Your updated map with distance line should look like the following screenshot:



Also please try to explain your findings.

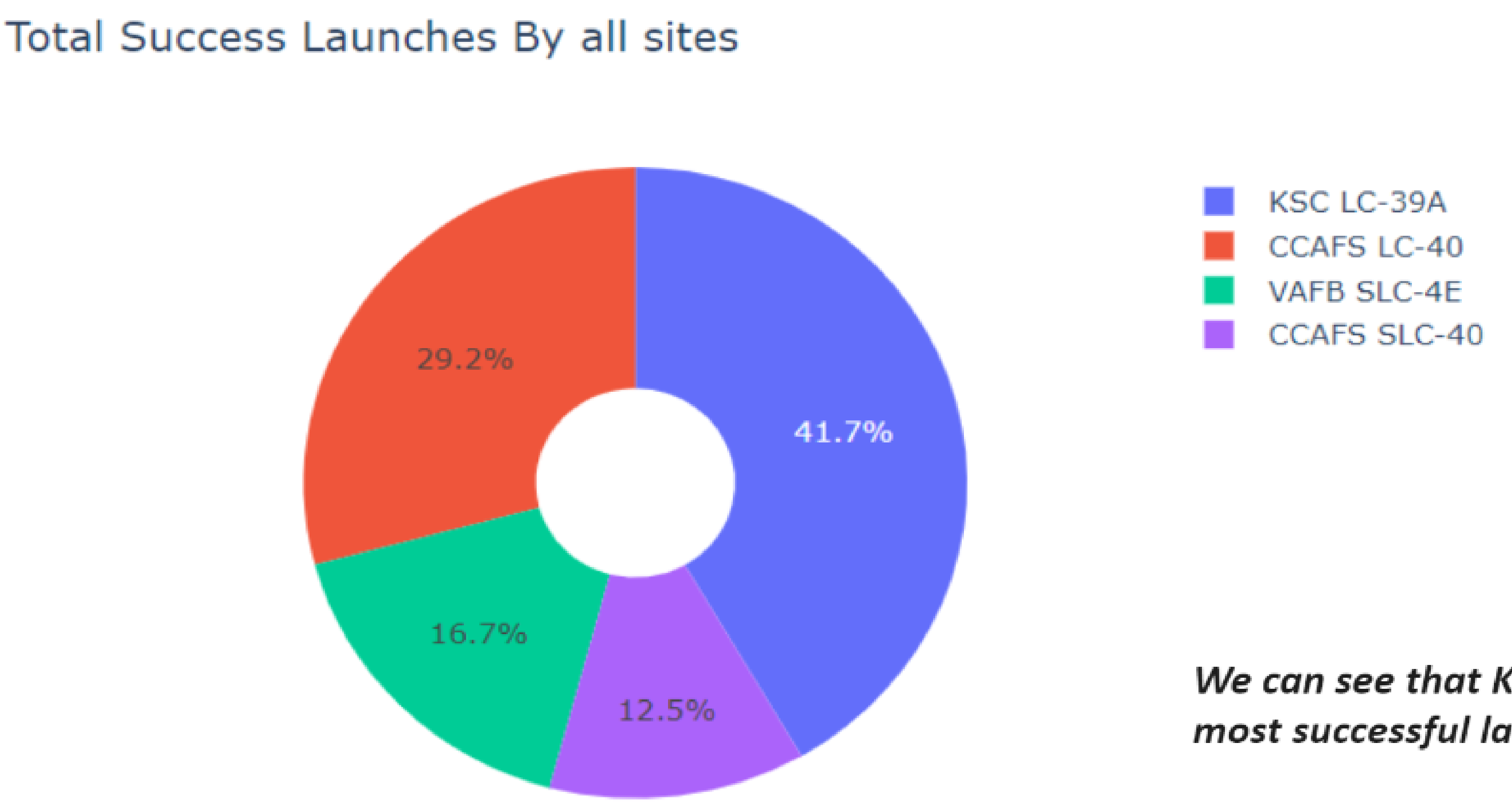
My Findings

- As mentioned before, launch sites are in close proximity to equator to minimize fuel consumption by using Earth's ~ 30km/sec eastward spin to help spaceships get into orbit.
- Launch sites are in close proximity to coastline so they can fly over the ocean during launch, for at least two safety reasons-- (1) crew has option to abort launch and attempt water landing (2) minimize people and property at risk from falling debris.
- Launch sites are in close proximity to highways, which allows for easily transport required people and property.
- Launch sites are in close proximity to railways, which allows transport for heavy cargo.
- Launch sites are not in close proximity to cities, which minimizes danger to population dense areas.As mentioned before, * launch sites are in close proximity to equator to minimize fuel consumption by using Earth's ~ 30km/sec eastward spin to help spaceships get into orbit.

Section 4

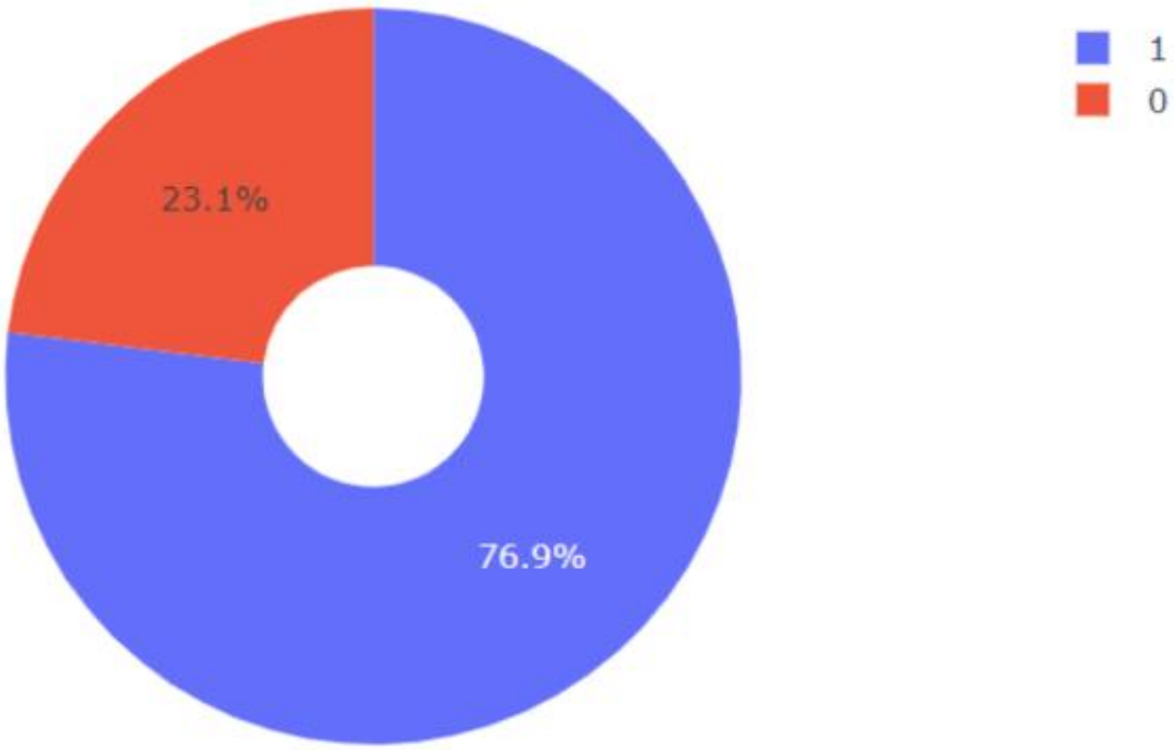
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



We can see that KSC LC-39A had the most successful launches from all the sites

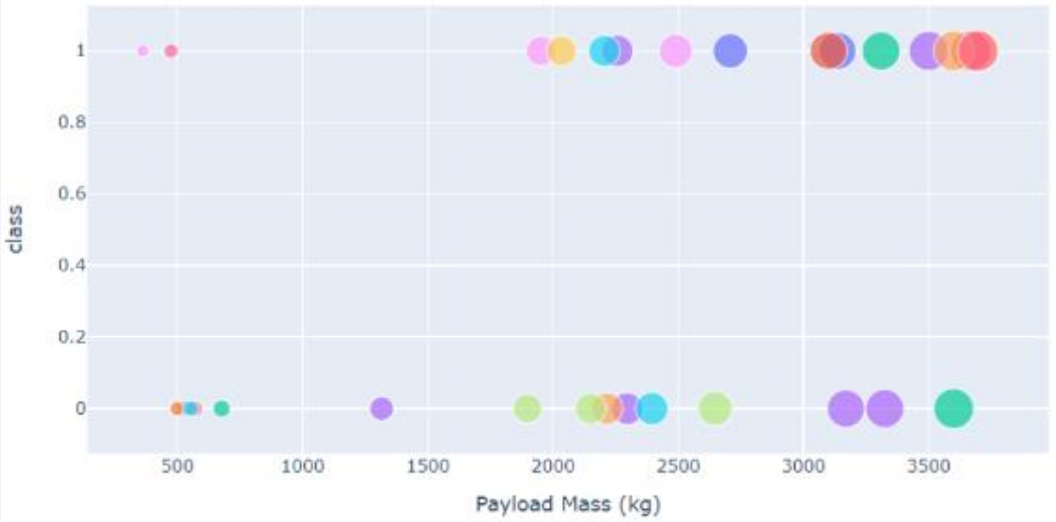
Pie chart showing the Launch site with the highest launch success ratio



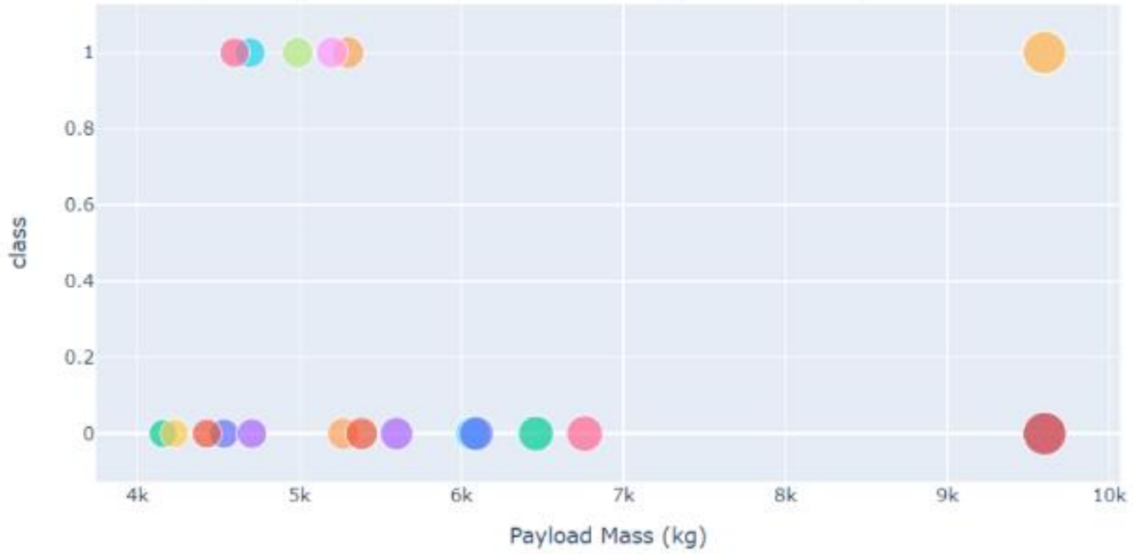
KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider

Low Weighted Payload 0kg – 4000kg



Heavy Weighted Payload 4000kg – 10000kg



We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- The accuracy is 0.8482

```
svm_cv.fit(x_train, y_train)
```

[18]:

```
GridSearchCV
GridSearchCV(cv=10, estimator=SVC(),
             param_grid={'C': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.16227766e+01,
       1.00000000e+03]),
                         'gamma': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.16227766e+01,
       1.00000000e+03]),
                         'kernel': ('linear', 'rbf', 'poly', 'rbf', 'sigmoid')})
estimator: SVC
SVC()
SVC
SVC()
```

[19]:

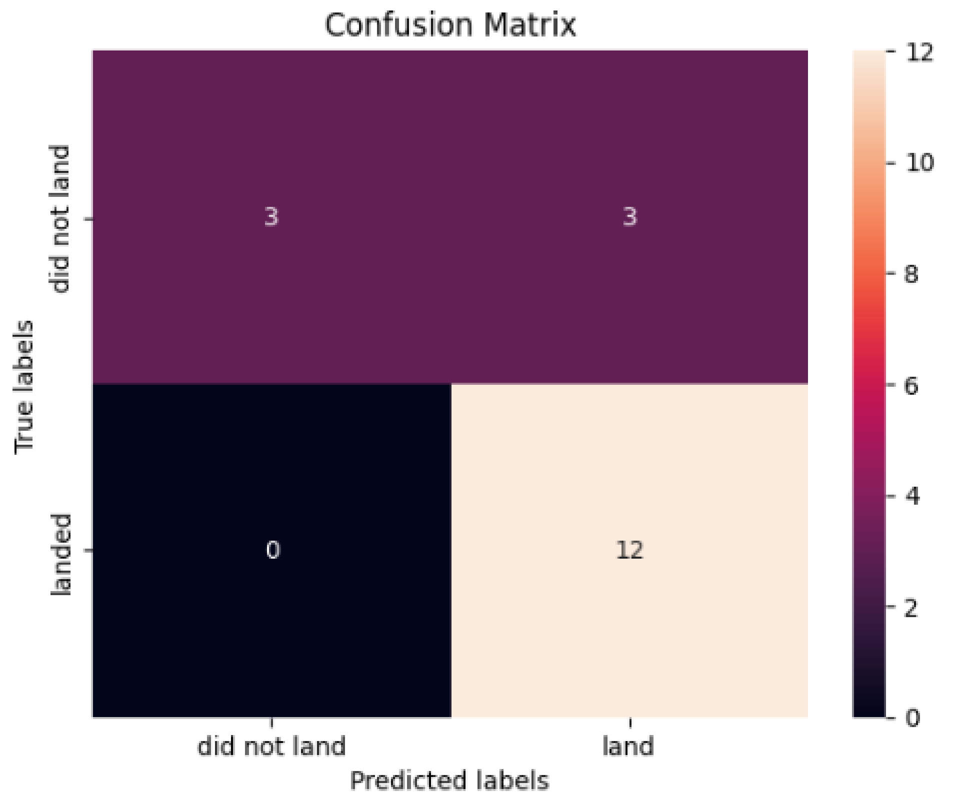
```
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)
```

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

Confusion Matrix

- The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives .i.e., unsuccessful landing marked as successful landing by the classifier.

```
[21]: yhat=svm_cv.predict(X_test)
      plot_confusion_matrix(Y_test,yhat)
      plt.show()
```



Conclusions

We can conclude that:

- 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.

Appendix

Assigning the coordinates for each site

In [4]:
```
# Download and read the `spacex_launch_geo.csv`
from js import fetch
import io

URL = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/spacex_lau
resp = await fetch(URL)
spacex_csv_file = io.BytesIO((await resp.arrayBuffer()).to_py())
spacex_df=pd.read_csv(spacex_csv_file)
```

Now, you can take a look at what are the coordinates for each site.

In [5]:
```
# Select relevant sub-columns: `Launch Site`, `Lat(Latitude)`, `Long(Longitude)`, `class`
spacex_df = spacex_df[['Launch Site', 'Lat', 'Long', 'class']]
launch_sites_df = spacex_df.groupby(['Launch Site'], as_index=False).first()
launch_sites_df = launch_sites_df[['Launch Site', 'Lat', 'Long']]
launch_sites_df
```

Out[5]:

	Launch Site	Lat	Long
0	CCAFS LC-40	28.562302	-80.577356
1	CCAFS SLC-40	28.563197	-80.576820
2	KSC LC-39A	28.573255	-80.646895
3	VAFB SLC-4E	34.632834	-120.610745

Finding distance to a closest city, railway, highway,

In [19]:
```
distance_highway = calculate_distance(launch_site_lat, launch_site_lon, closest_highway[0], closest_highway[1])
print('distance_highway =',distance_highway, ' km')
distance_railroad = calculate_distance(launch_site_lat, launch_site_lon, closest_railroad[0], closest_railroad[1])
print('distance_railroad =',distance_railroad, ' km')
distance_city = calculate_distance(launch_site_lat, launch_site_lon, closest_city[0], closest_city[1])
print('distance_city =',distance_city, ' km')
```

```
distance_highway = 0.5844095688237401  km
distance_railroad = 1.282798438953375  km
distance_city = 51.43547596583314  km
```

Thank you!