IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

John Lukasavage
4/17/2023

Outline

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

Executive Summary

METHODOLOGIES USED

- Data Collection
- Data Wrangling
- Data Analysis with Visualisation
- Data Analysis with SQL
- Folium Interactive Map
- Interactive Dashboard
- Predictive Analysis

RESULTS

- EDA Results
- Interactive Analysis Dashboard
- Predictive Analysis

Introduction

- Project background and context

In this capstone, we will predict if the Falcon 9 first stage will land successfully. SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage.

- Problems you want to find answers

if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

Section 1

Methodology

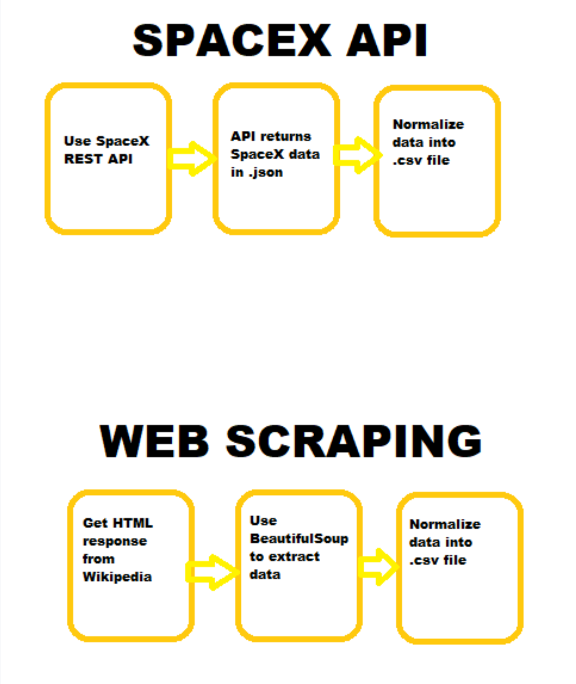
Methodology

Executive Summary

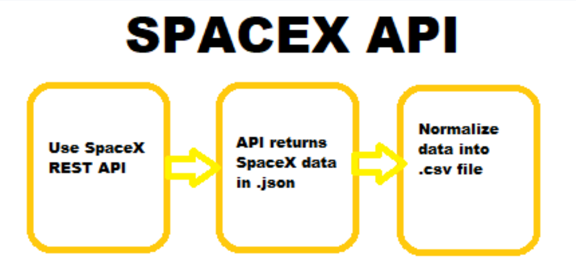
- Data collection methodology:
  - Webscraping and obtaining data through SpaceX rest API
- Perform data wrangling
  - Data was cleaned from irrelevent information and to fill null values
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Logistic Regression, SVM, Decision Tree and K-Nearest Neighbor models were created, trained, tested, tuned and scored for accuracy singularly as well as against each other.

Data Collection

- Data sets were collected through the SpaceX rest API and through webscraping SpaceX launch information from Wikipedia



Data Collection – SpaceX API



- https://github.com/prostheticears/IBMcapstone/blob/c6b7f928d2251d39c2cfdbccc8a74ea7202c5442/Capstone%20Collecting%20Data.ipynb

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

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

Check the content of the response

[8]:
```
print(response.content)
```

b'[{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},"links":{"patch":{"small":"https://images2.imgbox.com/94/f2/NN6F
_o.png","large":"https://images2.imgbox.com/5b/02/QcxHUb5V_o.png"},"reddit":{"campaign":null,"launch":null,"media":null,"recovery":null},"flickr":{"sm
l":[],"original":[]},"presskit":null,"webcast":"https://www.youtube.com/watch?v=0a_00nJ_Y88","youtube_id":"0a_00nJ_Y88","article":"https://www.space.c
196-spacex-inaugural-falcon-1-rocket-lost-launch.html","wikipedia":"https://en.wikipedia.org/wiki/DemoSat"},"static_fire_date_utc":"2006-03-17T00:00:0
0Z","static_fire_date_unix":1142553600,"net":false,"window":0,"rocket":"5e9d0d95eda69955f709d1eb","success":false,"failures":[{"time":33,"altitude":nu
reason":"merlin engine failure"}],"details":"Engine failure at 33 seconds and loss of vehicle","crew":[],"ships":[],"capsules":[],"payloads":["5eb0e4b

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

[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

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

[10]: 200

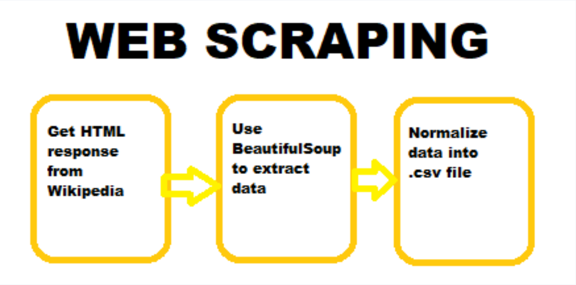
Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

[14]:
```
# 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

[15]:
```
# Get the head of the dataframe
data.head()
```

Data Collection - Scraping



- https://github.com/prostheticears/IBMcapstone/blob/9af82f2af5d97a5f8bcb1927dea2db96f464abc9/jupyter-labs-webscraping.ipynb

In [4]:
```python
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]:
```python
# 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]:
```python
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html_data.text, 'html.parser')
```

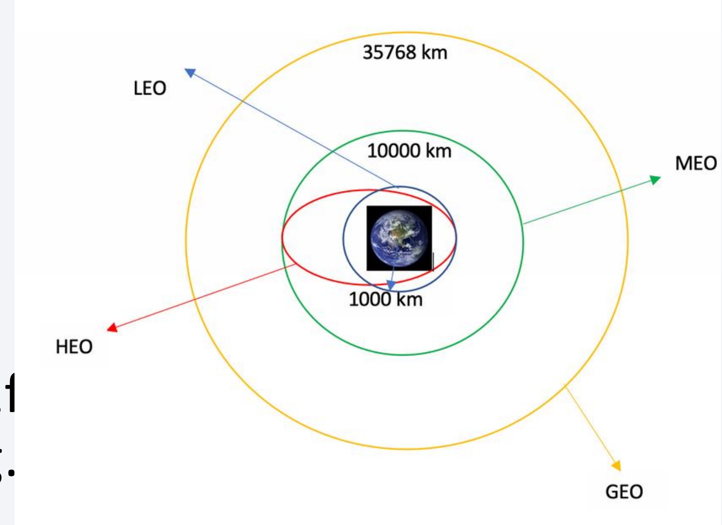
Print the page title to verify if the `BeautifulSoup` object was created properly

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

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

Data Wrangling

- Exploratory data analysis determined training labels
- Calculate number of launches and orbits
- Determine landing outcome and convert to a .csv file
- https://github.com/prostheticears/IBMcapstone/blob/9af82f cb1927dea2db96f464abc9/Capstone%20Data%20Wrangling.



EDA with Data Visualization

- Scatter plots were created to easily visualize successful landing compared to orbit and payload values
- https://github.com/prostheticears/IBMcapstone/blob/10515e9d897b25c0086ffa386e30d1804ce4c7da/IBM-DS0321EN-SkillsNetwork_labs_module_2_jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

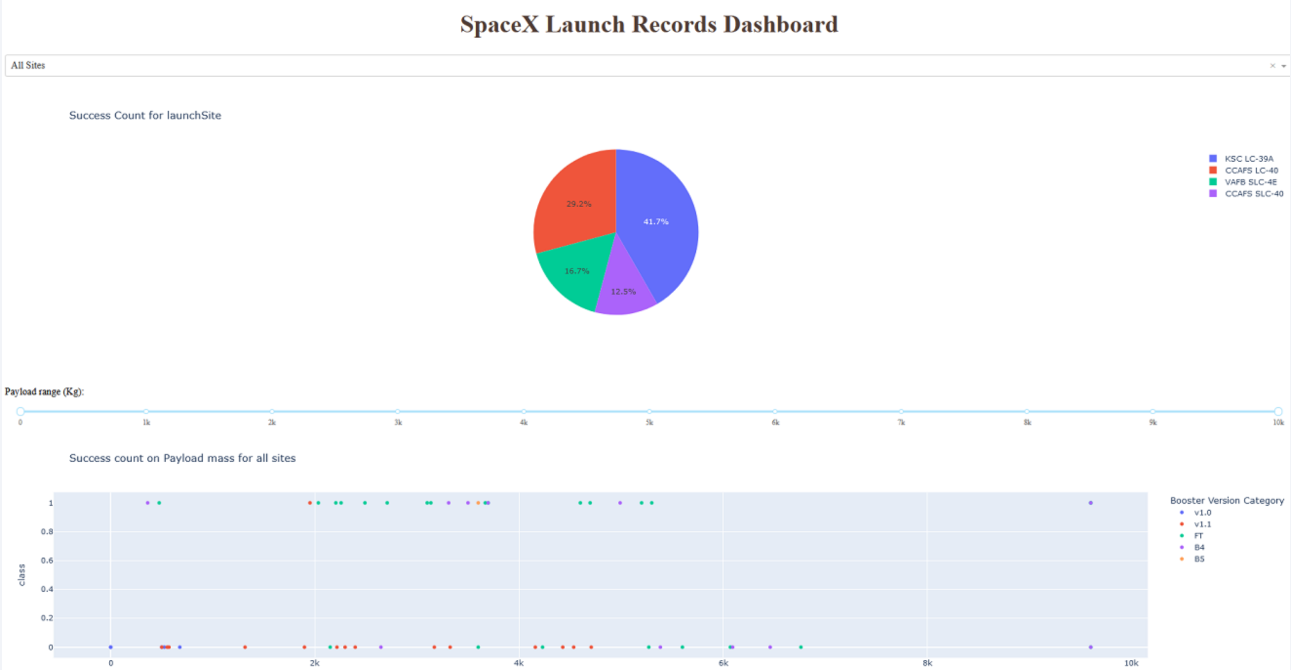
- SQL queries included
  - Unique launch sites
  - Payload mass
  - Successful vs failed recoveries
- https://github.com/prostheticears/IBMcapstone/blob/10515e9d897b25c0086ffa386e30d1804ce4c7da/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Circles, lines and map markers were created with Folium
- These objects were used to better break down information based on launch location in a visually appealing manner.
- https://github.com/prostheticears/IBMcapstone/blob/b257c8ce9607c67a51447e94563c622e952a5208/IBM-DS0321EN-SkillsNetwork_labs_module_3_lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- INteractive pie chart and scatter plot with slider were added to dashboard
- Plots and interactions can further break down information simply.
- https://github.com/prostheticears/IBMcapstone/blob/b257c8ce9607c67a51447e94563c622e952a5208/spacex_dash_app.py



Predictive Analysis (Classification)

- Loaded dataset with numpy and pandas, transformed and split into training and test sets.
- Built different machine learning models, fit paramenters and tuned using GridSearchCV
- Checked for accuracy standalone and against each other mathematically and visually with confusion matrices.
- https://github.com/prostheticears/IBMcapstone/blob/b257c8ce9607c67a51447e94563c622e952a5208/machine%20learning%20predict.ipynb

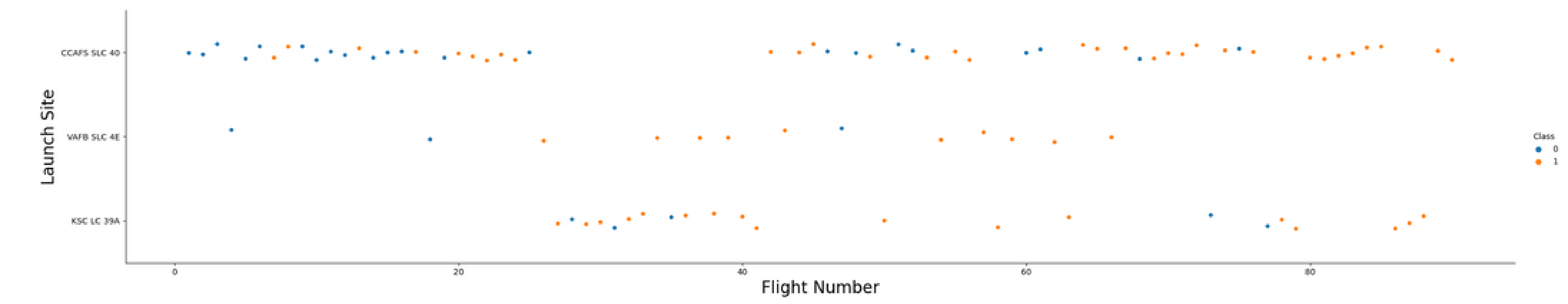
Results

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

Section 2

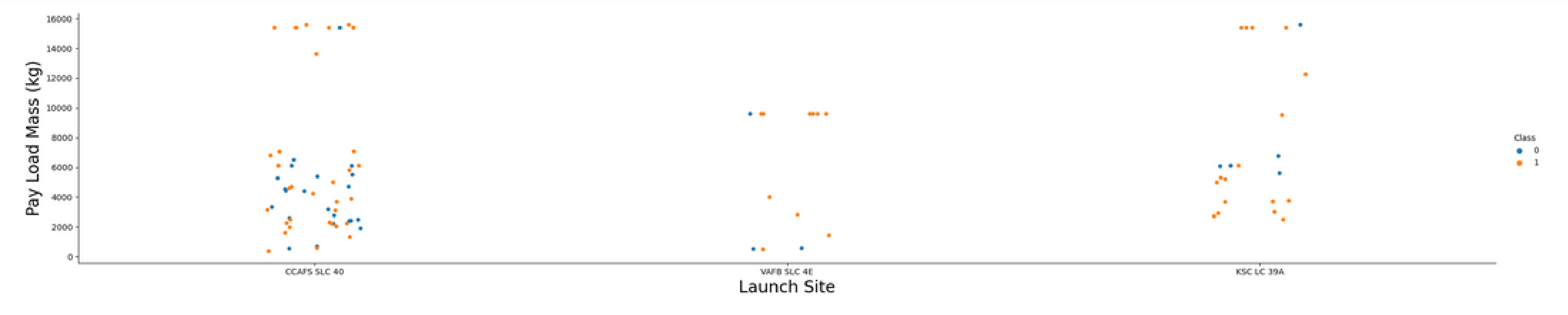
Insights drawn from EDA

Flight Number vs. Launch Site



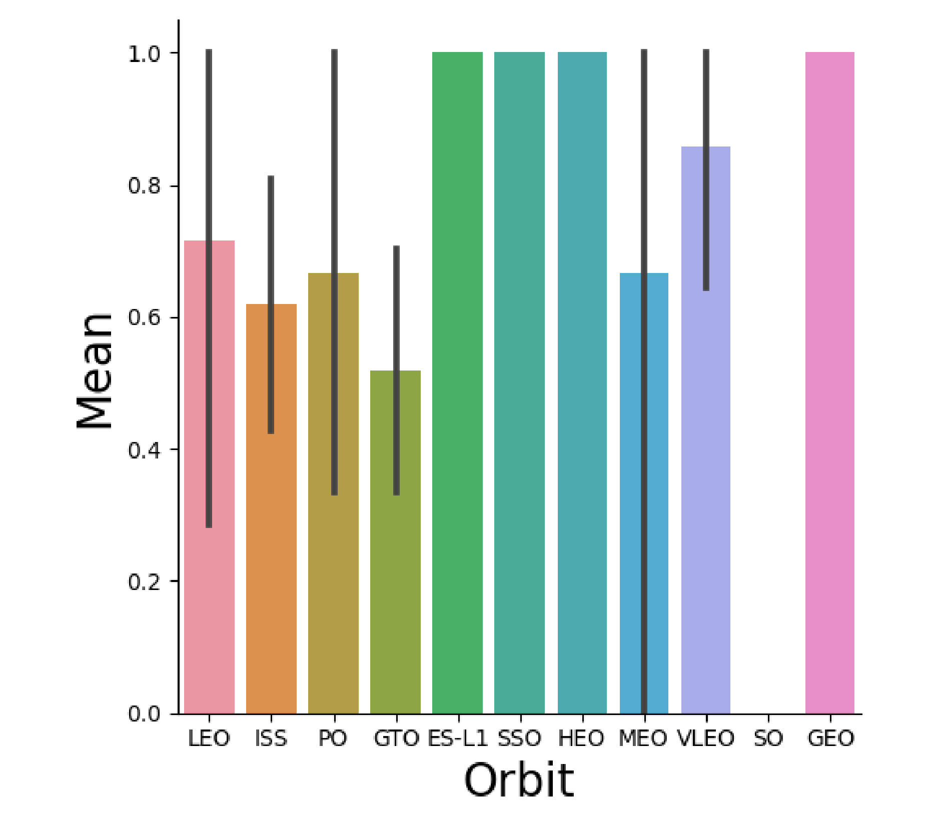
- Plot of successful missions based on Launch Site and Flight Number

Payload vs. Launch Site

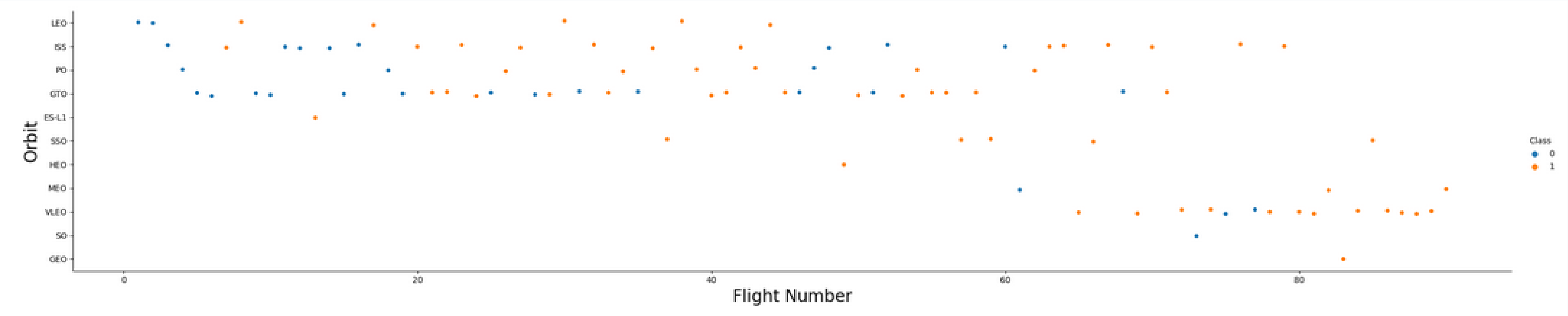


- Scatter plot of Payload vs Launch Site

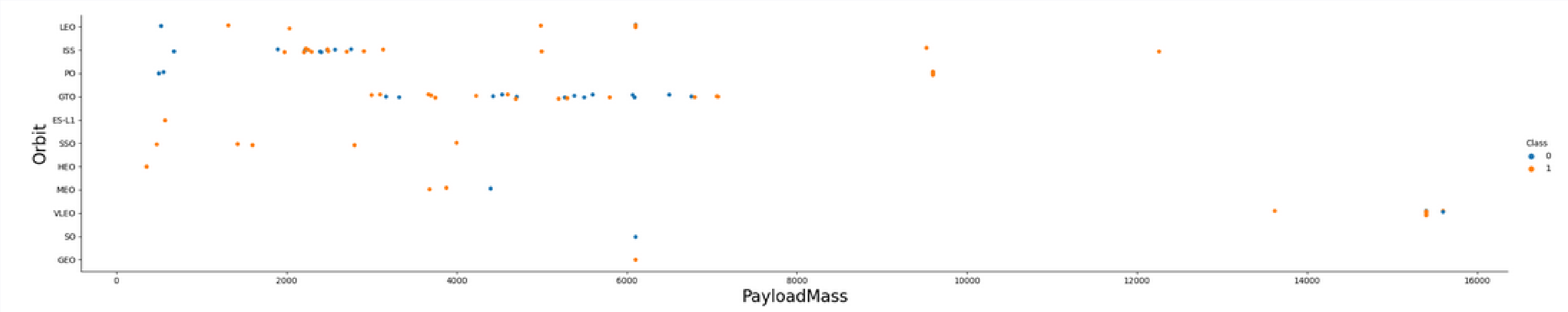
Success Rate vs. Orbit Type



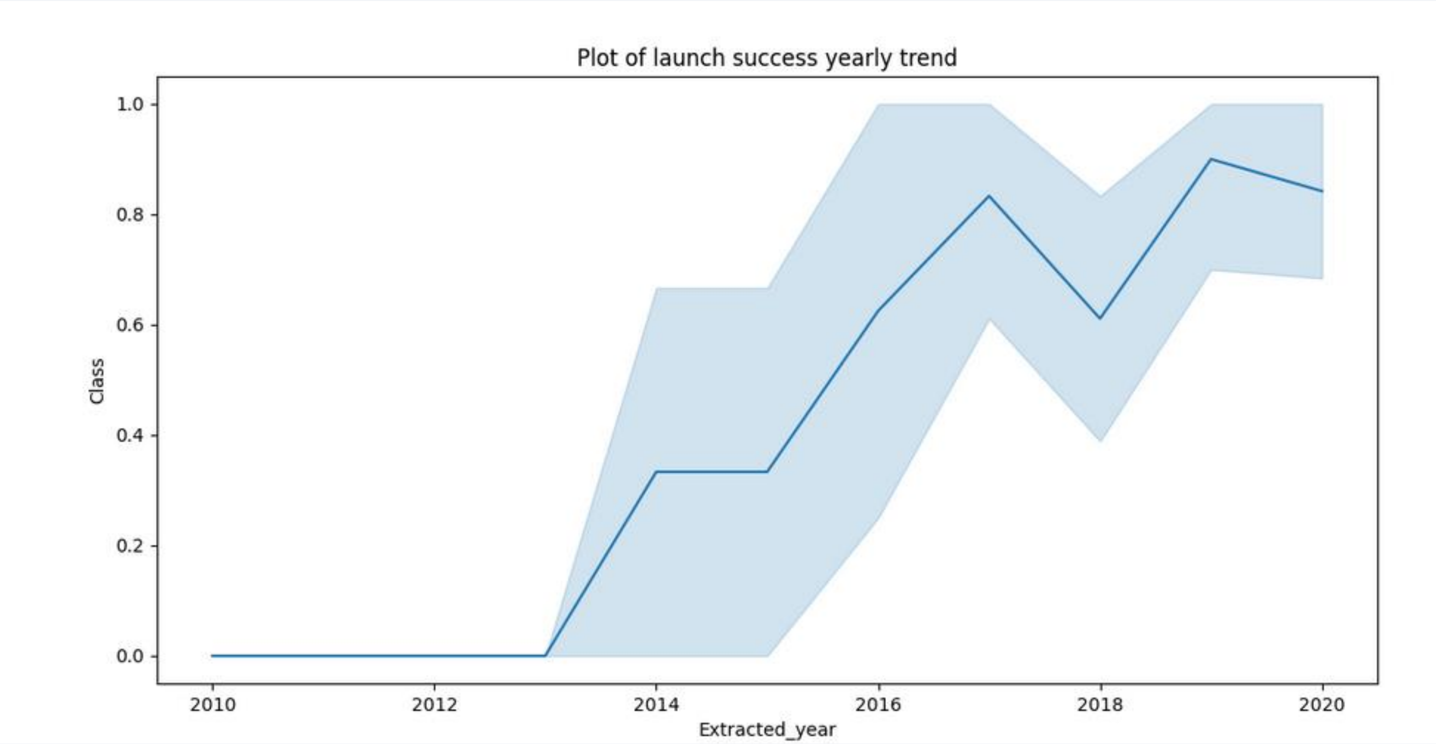
Flight Number vs. Orbit Type



Payload vs. Orbit Type



Launch Success Yearly Trend



All Launch Site Names

- We find all launch site names with the command Distinct

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'

- Find 5 records where launch sites begin with `CCA`
- We launch a query looking only for values where Launch_Site is LIKE '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

- Calculate the total payload carried by boosters from NASA
- Use the SUM function to calculate total of 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

- Calculate the average payload mass carried by booster version F9 v1.1 by querying F9 launch payloads and using AVG function

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

- The first successful ground pad landing was 12/22/2015

```
        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

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000 using the WHERE clause in query

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

- Calculate the total number of successful and failure mission outcomes using WHERE statement in query

```
        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)
```

```
The total number of successful mission outcome is:
```

	successoutcome
0	100

```
The total number of failed mission outcome is:
```

Out[16]:

	failureoutcome
0	1

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass

```
        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

2015 Launch Records

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015 using WHERE clause using BETWEEN, AND, LIKE

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

- 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 [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]:

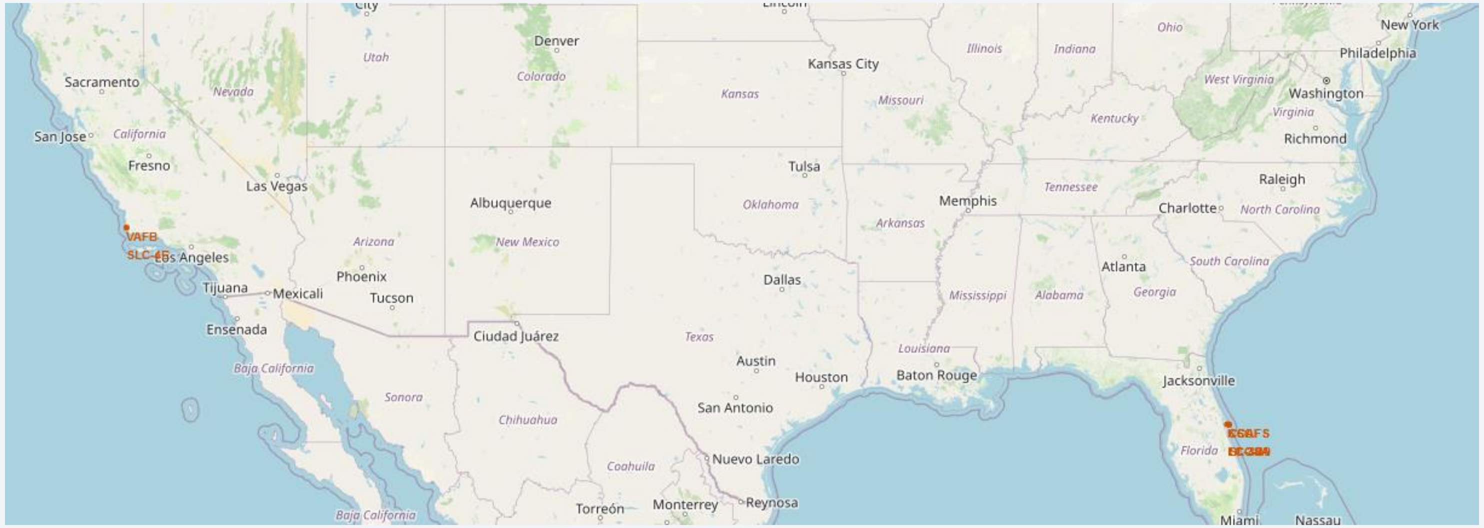
	landingoutcome	count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3

Section 3

Launch Sites Proximities Analysis

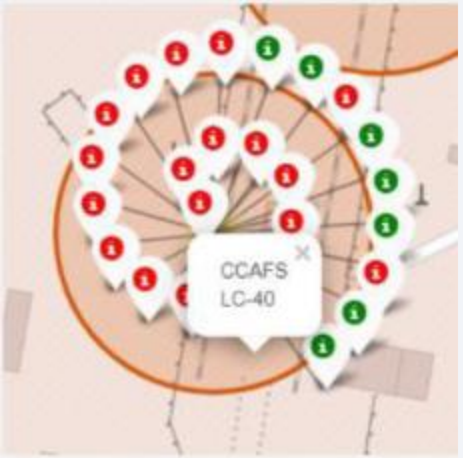
Launch Site Map

- Screenshot shows all launch locations in the United States



Launch Outcomes

- Further zooming will display successful vs failed launch outcomes at each launch site



<Folium Map Screenshot 3>

- Replace <Folium map screenshot 3> title with an appropriate title

- Explore the generated folium map and show the screenshot of a selected launch site to its proximities such as railway, highway, coastline, with distance calculated and displayed

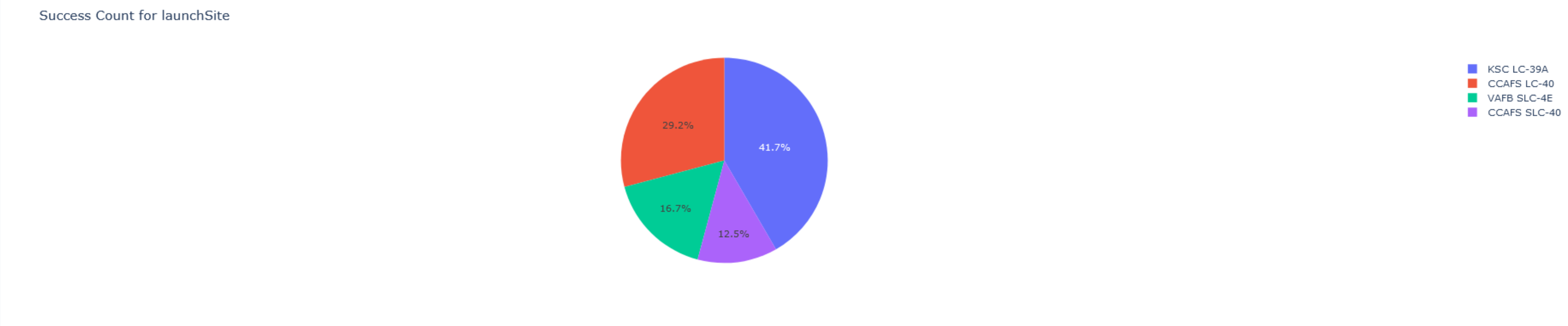
- Explain the important elements and findings on the screenshot

Section 4

Build a Dashboard with Plotly Dash

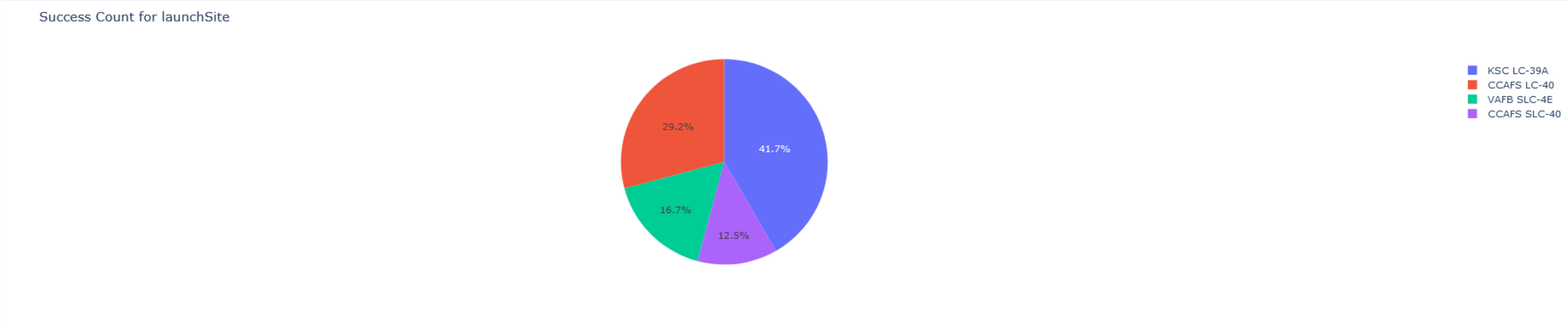
Success pie chart

- Pie chart shows launch success for all launch sites



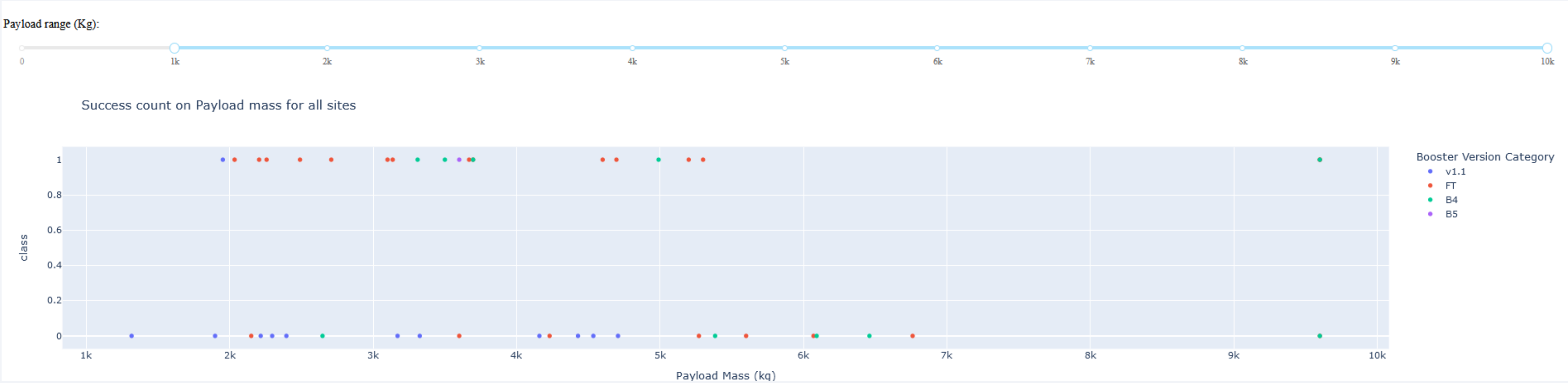
Highest success ratio

- Pie chart shows KSC LC-39A has the highest success percentage



Scatter plot with slider

Success rate is higher for lighter payloads



Section 5

Predictive Analysis (Classification)

Classification Accuracy

- The decision tree has the highest accuracy of prediction compared to other models

Find the method performs best:

In [35]:

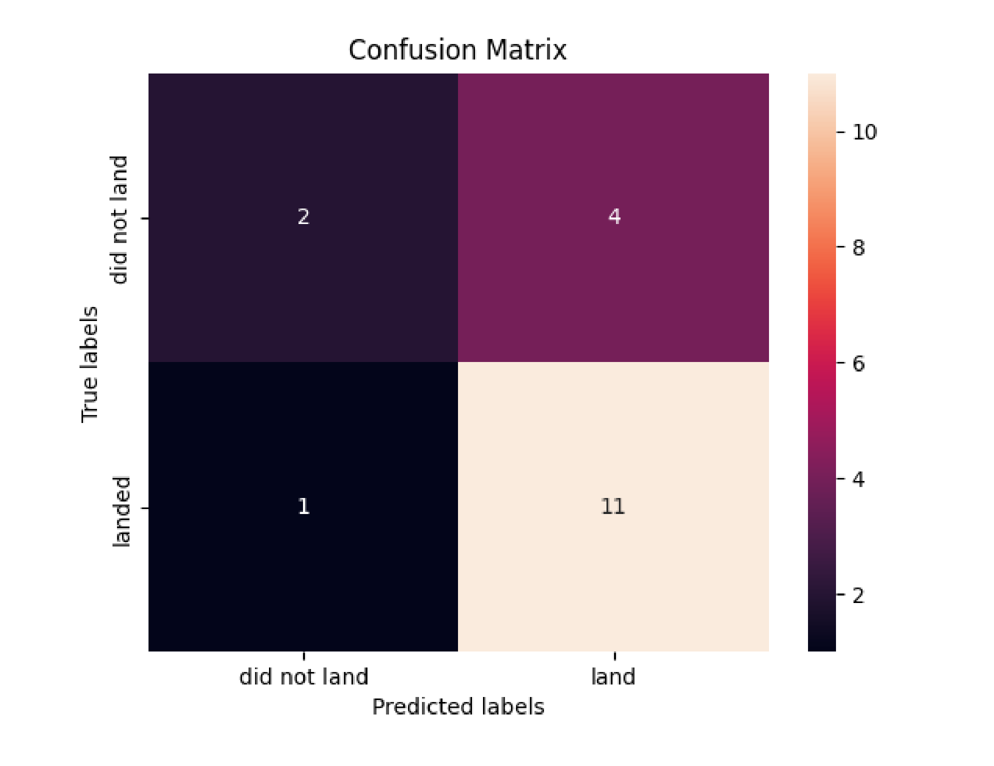
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
```

```
Best model is DecisionTree with a score of 0.8875
```

Confusion Matrix

- The confusion matrix shows the accuracy of models based on false positives and false negatives vs true positives and true negatives



Conclusions

- The Decision Tree is the best model to predict outcome
- Lighter payloads have a higher success rate than heavy payloads
- ES-L1, GEO, HEO, SSO, VLEO had the highest success rate
- As years increase, so does success rate
- ...

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!