IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Mikel Lazcano

19 November 2024

Outline

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

Executive Summary

Summary of methodologies

1. Data collection through API
2. Data collection with Web Scraping
3. Data Wrangling
4. Exploratory Data Analysis with SQL
5. Exploratory Data Analysis with Data Visualization
6. Interactive Visual Analytics with Folium
7. Machine Learning Prediction

Summary of all results

1. Exploratory Data Analysis result
2. Interactive analytics in screenshots
3. Predictive Analytics result

Introduction

Project background and context

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 savings is because SpaceX can reuse the first stage. Therefore, 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. In this project, we will try to predict if the Falcon 9 first stage will land successfully.

Problems you want to find answers

a. What factors determine the succesfull landing of the rocket?

b. The interaction amongst various features that determine the success rate of a landing.

c. What operation conditions needs to be in place to ensure a succesful landing program.

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 enconding was applied to categorical feautes
- 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

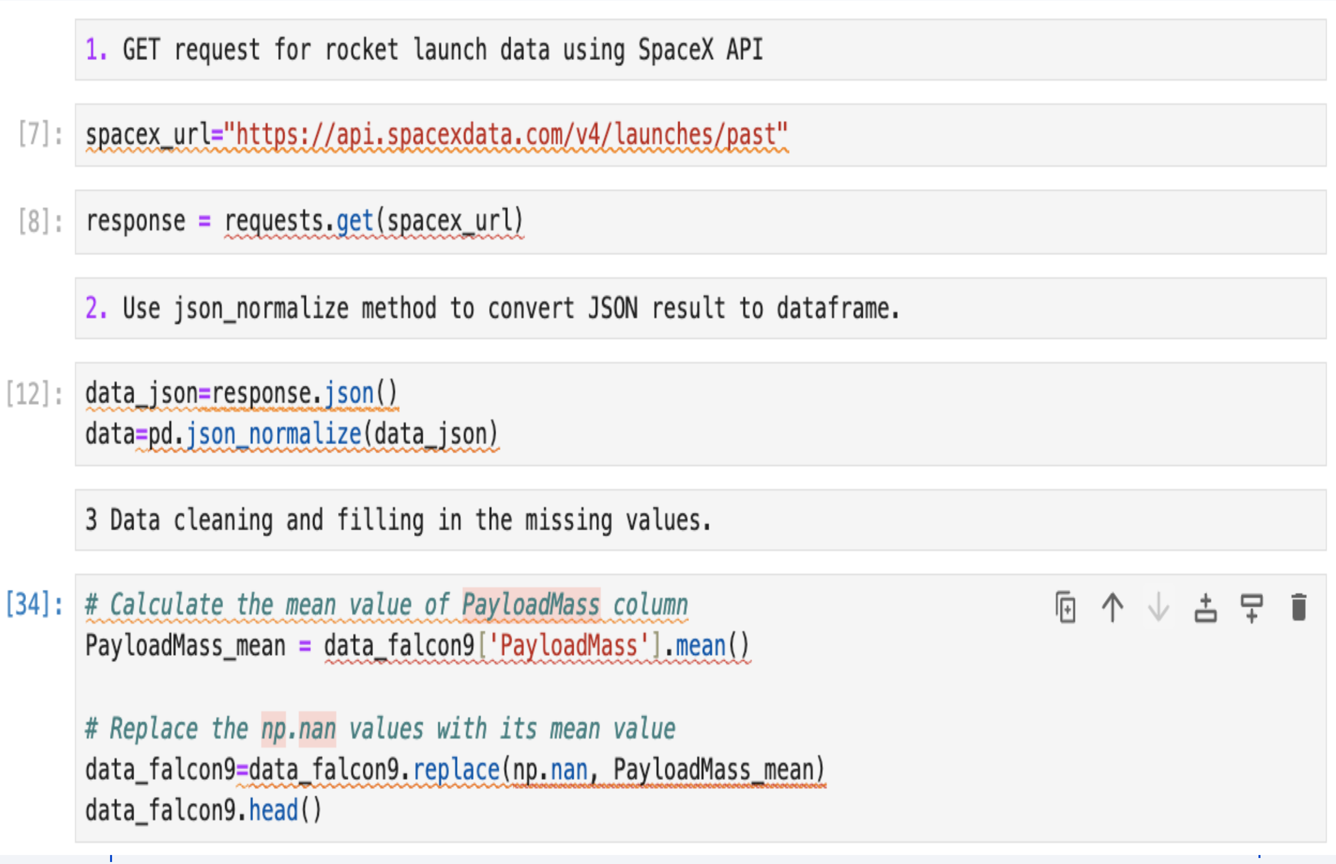
- The data was collected using various methods:

- Data collection was done using GET requests to the SpaceX API
- Next, the response content was decoded as JSON format using .json() call and turned into a pandas dataframe using .json_normalice().
- Then the data was cleaned, checked for missing values and filled in missing values where necessary.
- Additionaly web scraping was performed from Wikipedia with BeautifulSoup.
- The objetive was to extract the launch records as HTML table, parse and convert it to a pandas dataframe for future analysis.

Data Collection – SpaceX API

1. Collecting data using the SpaceX API.
2. Converting JSON result to dataframe.
3. Data cleaning and filling in the missing values.

Link to notebook



1. GET request for rocket launch data using SpaceX API

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

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

2. Use json_normalize method to convert JSON result to dataframe.

```
[12]: data_json=response.json()
      data=pd.json_normalize(data_json)
```

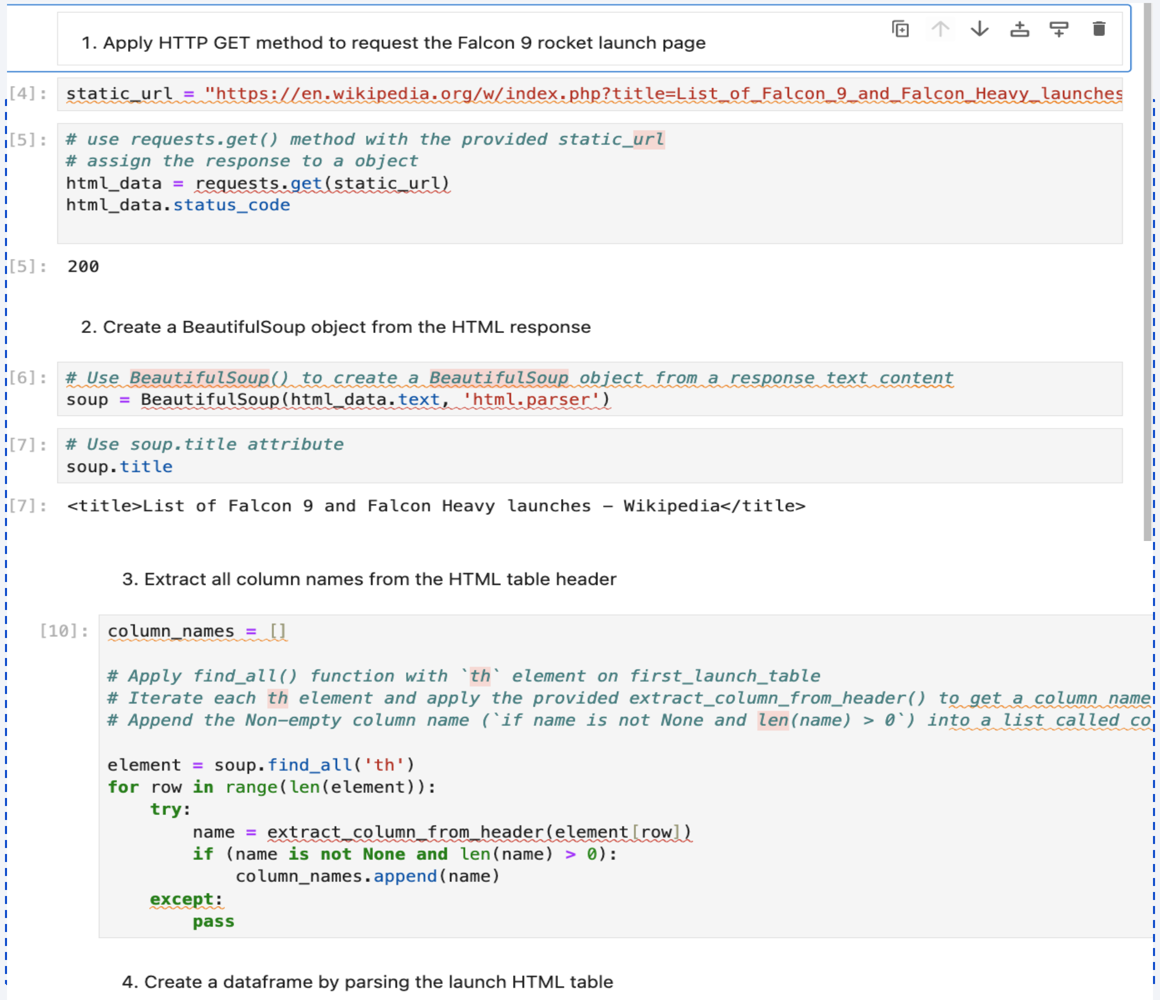
3 Data cleaning and filling in the missing values.

```
[34]: # Calculate the mean value of PayloadMass column
      PayloadMass_mean = data_falcon9['PayloadMass'].mean()

      # Replace the np.nan values with its mean value
      data_falcon9=data_falcon9.replace(np.nan, PayloadMass_mean)
      data_falcon9.head()
```

Data Collection - Scraping

1. Webscrapping Falcon 9 launch records with BeutifulSoup
2. Parsing the table and converting it into pandas dataframe

Link to notebook



1. Apply HTTP GET method to request the Falcon 9 rocket launch page

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

```
[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

[5]:  200
```

2. Create a BeautifulSoup object from the HTML response

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

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

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

3. Extract all column names from the HTML table header

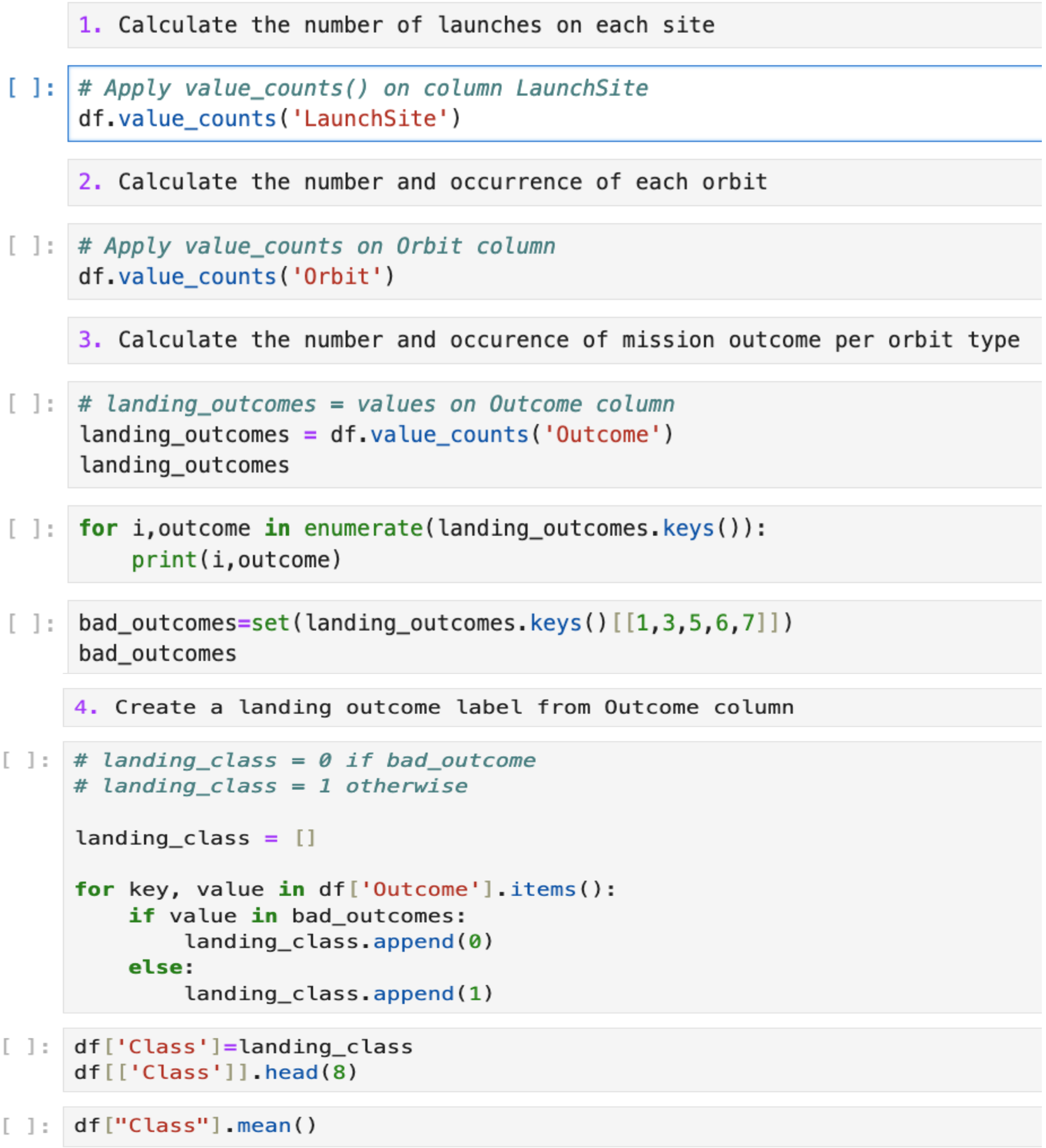
```
[10]:  column_names = []

       # 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 co

       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
```

4. Create a dataframe by parsing the launch HTML table

Data Wrangling

1. Calculating the number of launches on each site
2. Calculating the number and occurrence of each orbit
3. Calculating the number and occurence of mission outcome per orbit type
4. Creating a landing outcome label from Outcome column

Link to notebook



1. Calculate the number of launches on each site

[ ]:
```python
# Apply value_counts() on column LaunchSite
df.value_counts('LaunchSite')
```

2. Calculate the number and occurrence of each orbit

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

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

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

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

[ ]:
```python
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

4. Create a landing outcome label from Outcome column

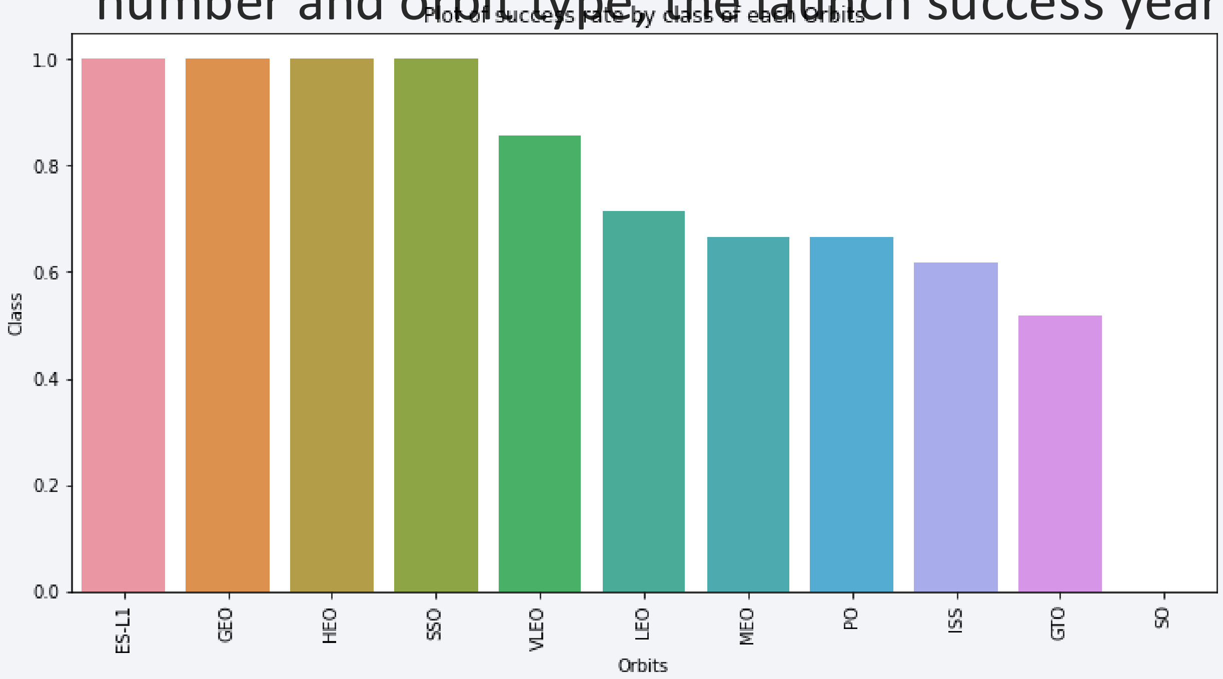
[ ]:
```python
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise

landing_class = []

for key, value in df['Outcome'].items():
    if value in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

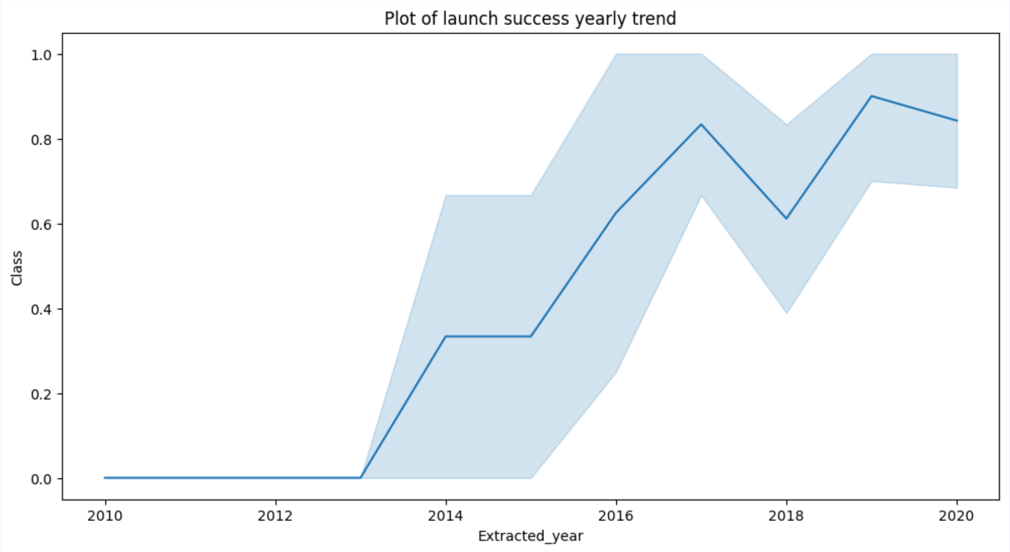
[ ]:
```python
df['Class']=landing_class
df[['Class']].head(8)
```

[ ]:
```python
df["Class"].mean()
```

EDA with Data Visualization

- Exploratory data analysis 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.





Link to notebook

EDA with SQL

- Loading the SpaceX dataset into a PostgreSQL database without leaving the jupyter notebook.
- Applying EDA with SQL to get insight from the data. Writing queries to find out for instance:

1. The names of unique launch sites in the space mission.
2. The total payload mass carried by boosters launched by NASA (CRS)
3. The average payload mass carried by booster version F9 v1.1
4. The total number of successful and failure mission outcomes
5. The failed landing outcomes in drone ship, their booster version and launch site names.

- Link to notebook

Build an Interactive Map with Folium

- Marking all launch sites and adding map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- Assigning 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, identified which launch sites have relatively high success rate.
- Calculating 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.

Link to notebook

Build a Dashboard with Plotly Dash

- Building an interactive dashboard with Plotly dash
- Plotting pie charts showing the total launches by a certain sites
- Plotting scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.

Link to notebook

Predictive Analysis (Classification)

- Loading the data using numpy and pandas, transforming the data, spliting our data into training and testing.
- Building different machine learning models and tuning different hyperparameters using GridSearchCV.
- Using accuracy as the metric for our model, improving the model using

feature engineering and algorithm tuning.

Link to notebook

Results

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

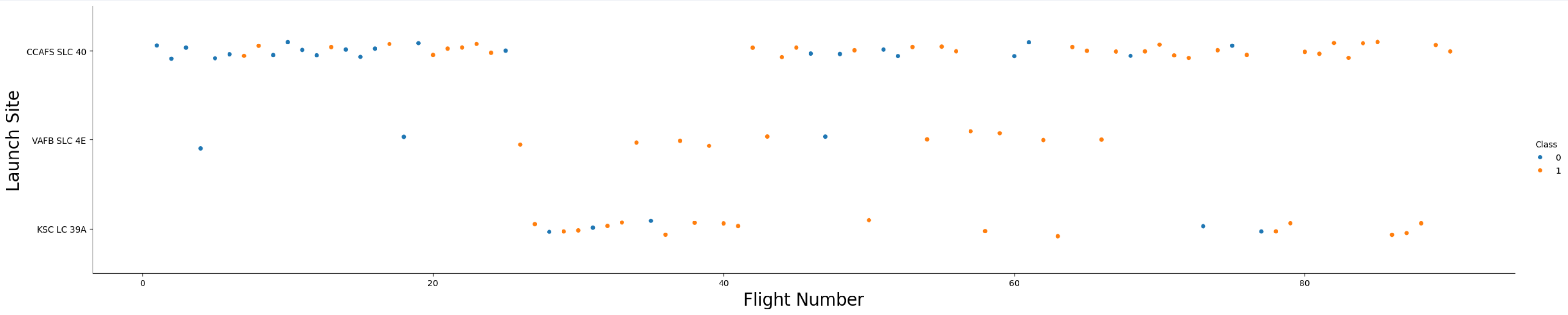
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

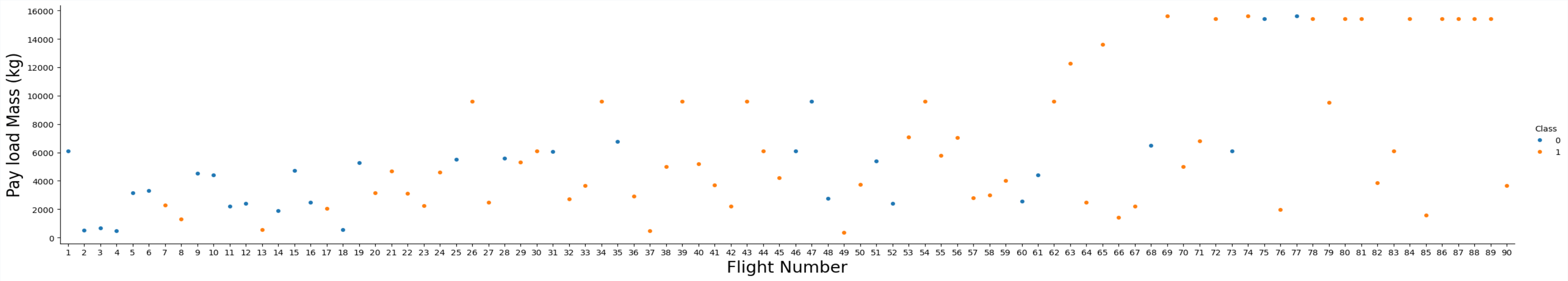
- From the plot, it can be seen the larger the flight amount at a launch site, the greater

the success rate at a launch site.



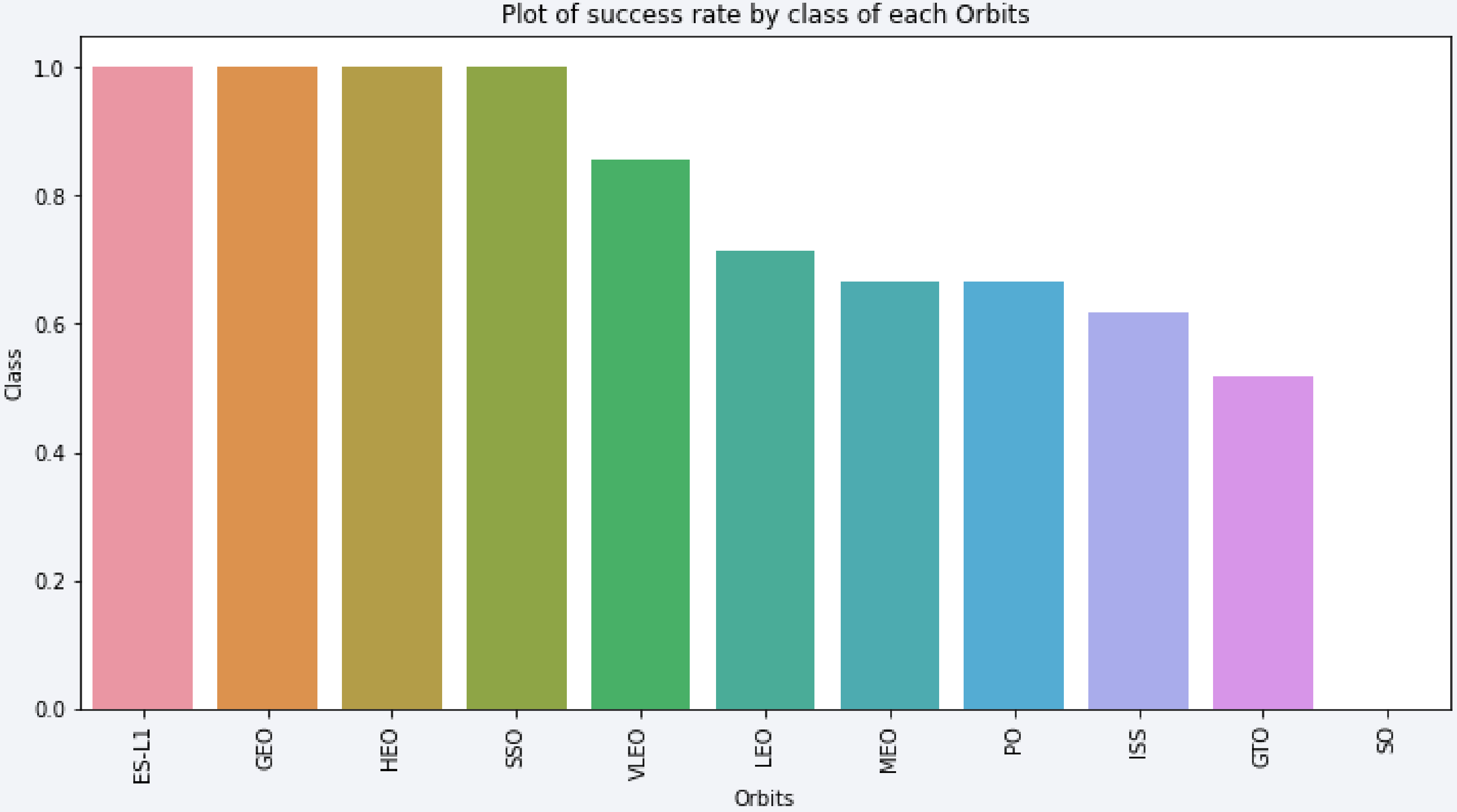
Payload vs. Launch Site

- It can be seen that as the flight number increases, the first stage is more likely to land successfully. The payload mass also appears to be a factor; even with more massive payloads, the first stage often returns successfully.



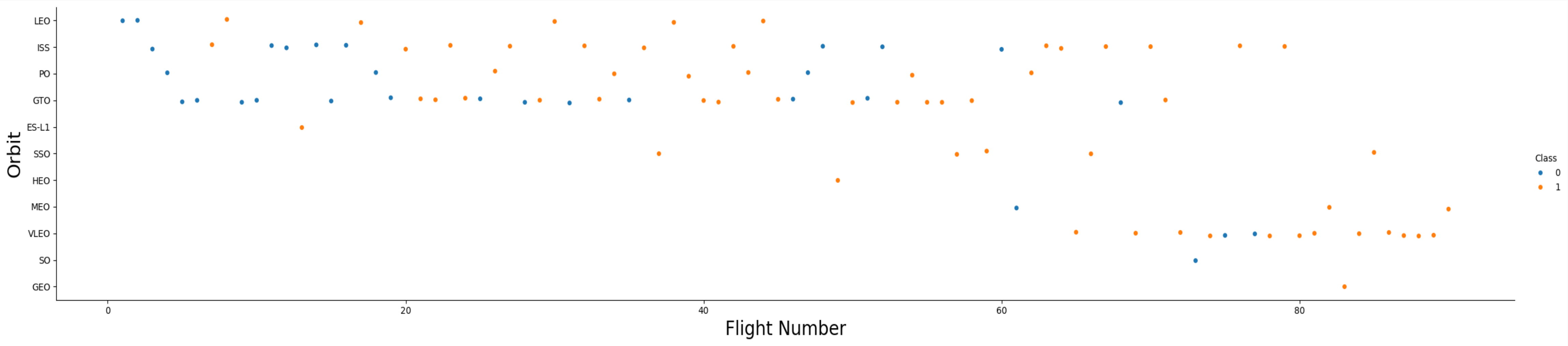
Success Rate vs. Orbit Type

- From the plot, we can see that ES-L1, GEO, HEO, SSO, VLEO had the most success rate.



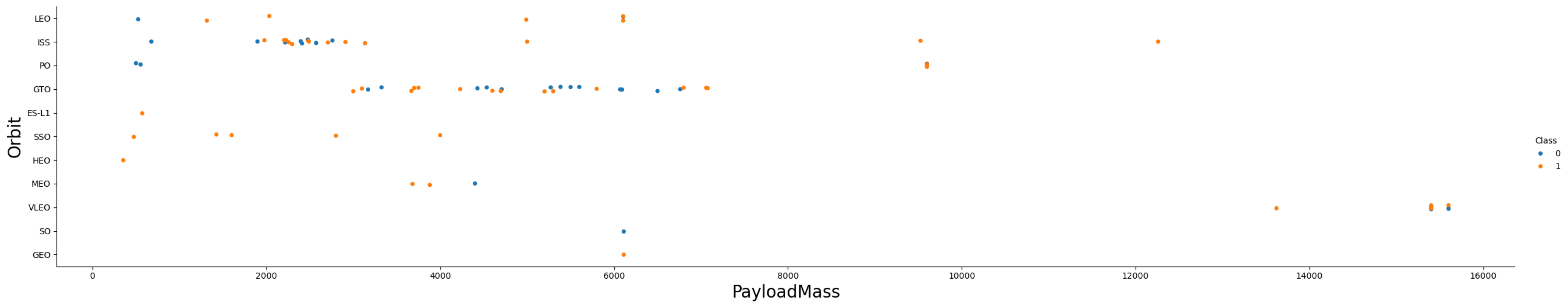
Flight Number vs. Orbit Type

- The plot below shows the Flight Number vs. Orbit type. It can be observed 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.



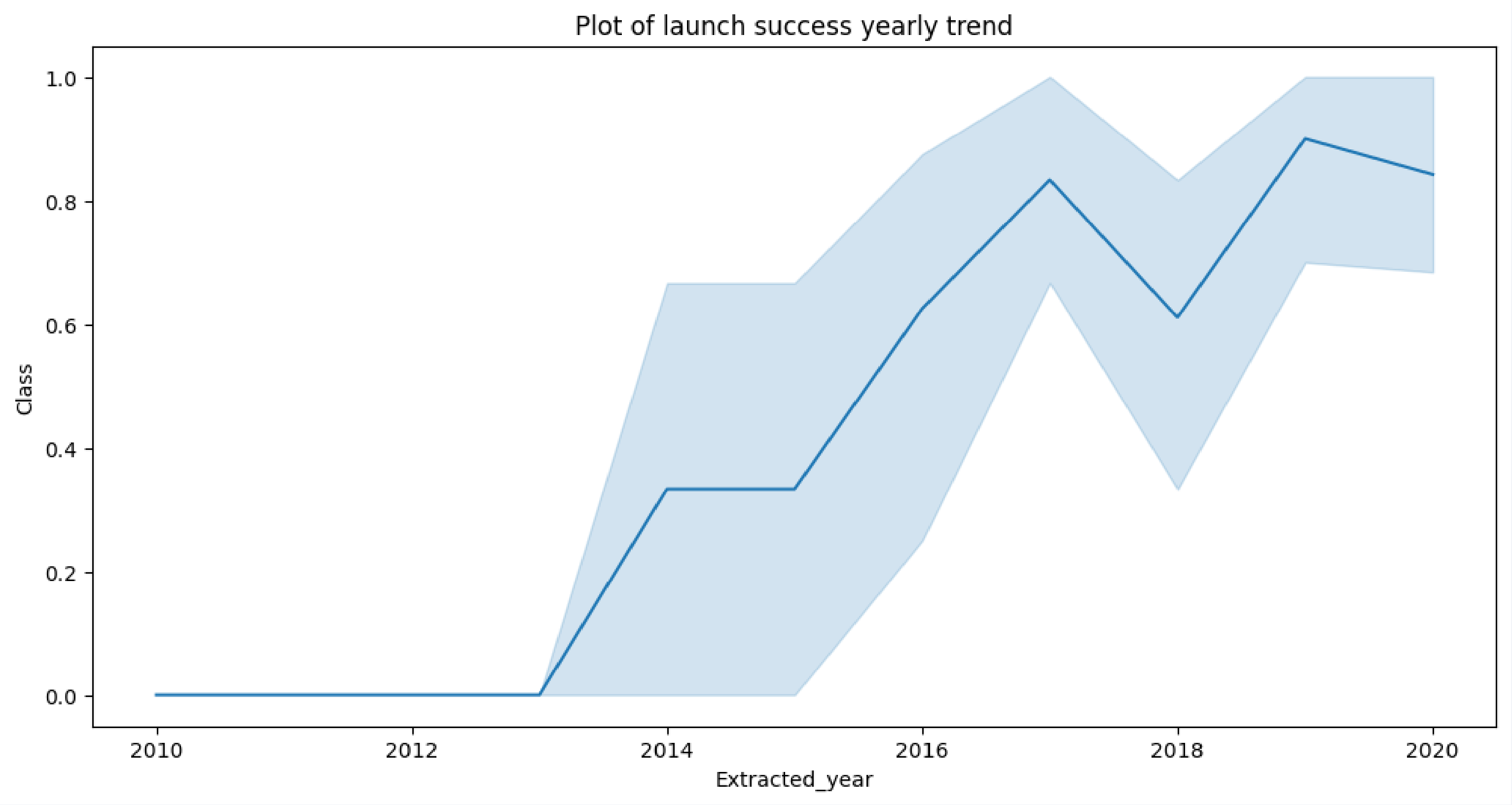
Payload vs. Orbit Type

- It can be observed that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits



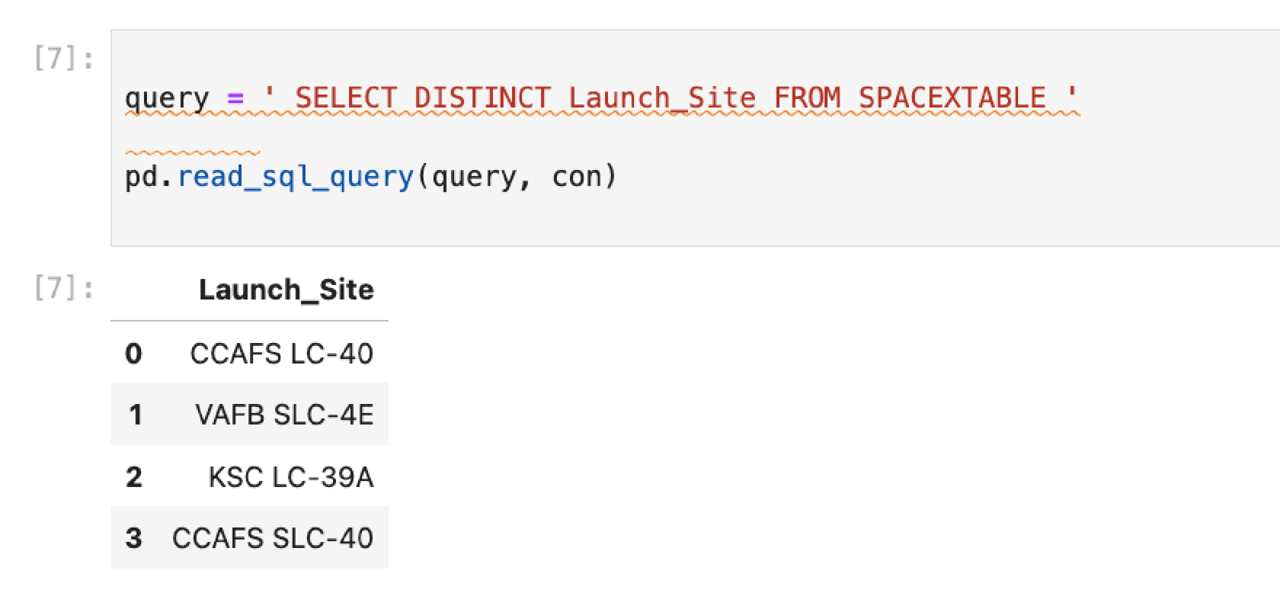
Launch Success Yearly Trend

- From the plot, it can be observed that success rate since 2013 kept on increasing till 2020.



All Launch Site Names

- Using the key word DISTINCT to show only unique launch sites from the SpaceX data.



[7]:

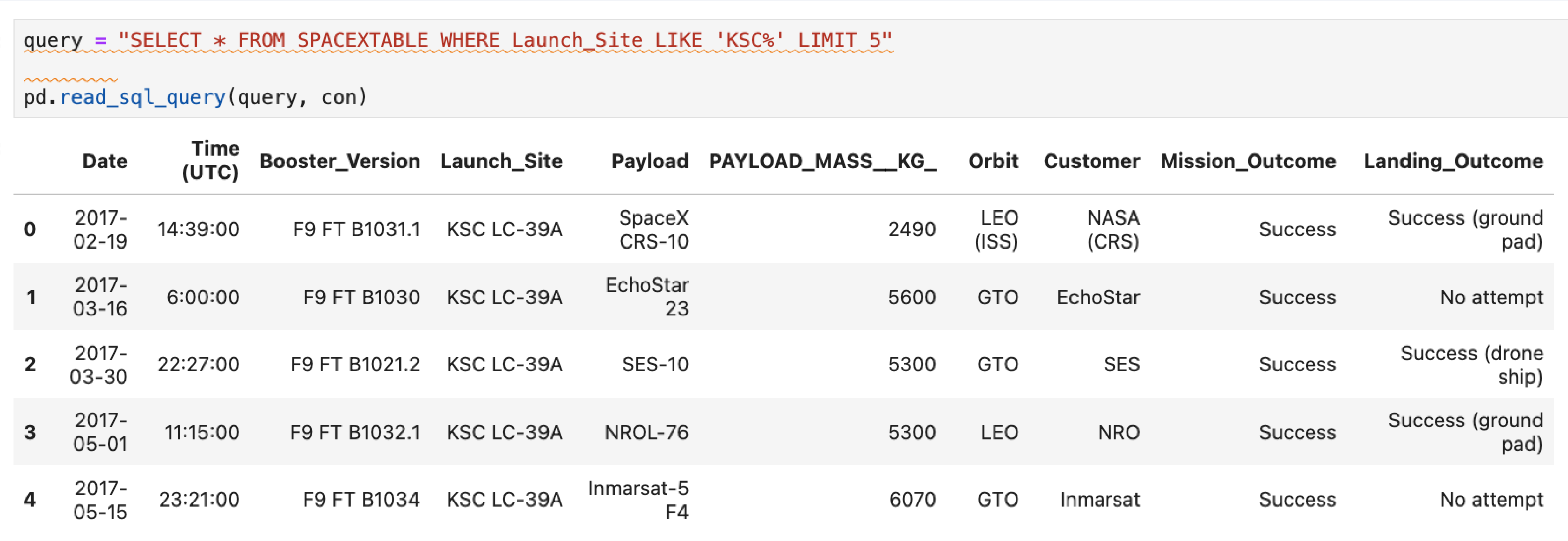
```
query = ' SELECT DISTINCT Launch_Site FROM SPACEXTABLE '

pd.read_sql_query(query, con)
```

[7]:

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

Launch Site Names Begin with 'KSC'

- Using the query below to display 5 records where launch sites begin with `KSC`

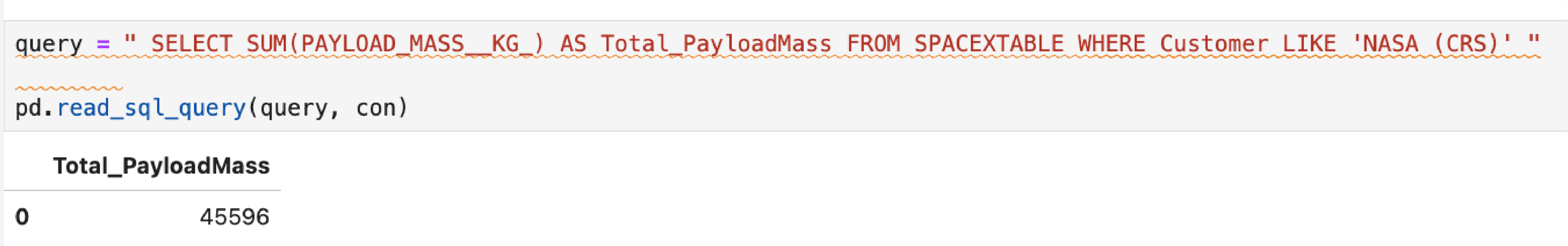


```
query = "SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'KSC%' LIMIT 5"

pd.read_sql_query(query, con)
```

	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
0	2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
1	2017-03-16	6:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	5600	GTO	EchoStar	Success	No attempt
2	2017-03-30	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
3	2017-05-01	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300	LEO	NRO	Success	Success (ground pad)
4	2017-05-15	23:21:00	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070	GTO	Inmarsat	Success	No attempt

Total Payload Mass

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

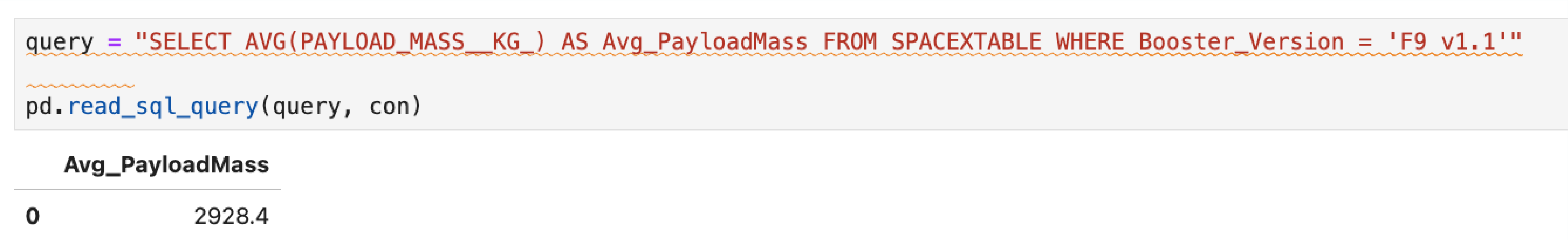


```
query = " SELECT SUM(PAYLOAD_MASS__KG_) AS Total_PayloadMass FROM SPACEXTABLE WHERE Customer LIKE 'NASA (CRS)' "

pd.read_sql_query(query, con)
```

```
   Total_PayloadMass
0              45596
```

Average Payload Mass by F9 v1.1

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

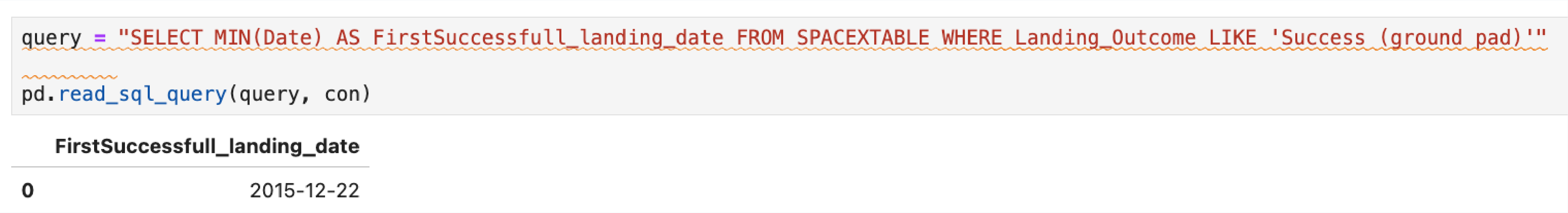


```
query = "SELECT AVG(PAYLOAD_MASS__KG_) AS Avg_PayloadMass FROM SPACEXTABLE WHERE Booster_Version = 'F9 v1.1'"

pd.read_sql_query(query, con)
```

	Avg_PayloadMass
0	2928.4

First Successful Ground Landing Date

- It can be observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015

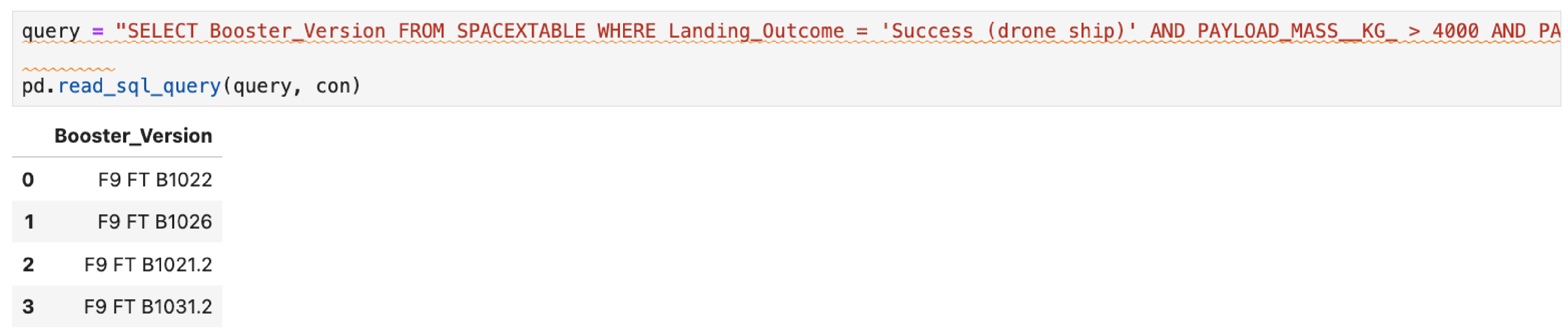


```
query = "SELECT MIN(Date) AS FirstSuccessfull_landing_date FROM SPACEXTABLE WHERE Landing_Outcome LIKE 'Success (ground pad)'"

pd.read_sql_query(query, con)
```

```
   FirstSuccessfull_landing_date
0                     2015-12-22
```

Successful Drone Ship Landing with Payload between 4000 and 6000

- Using 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

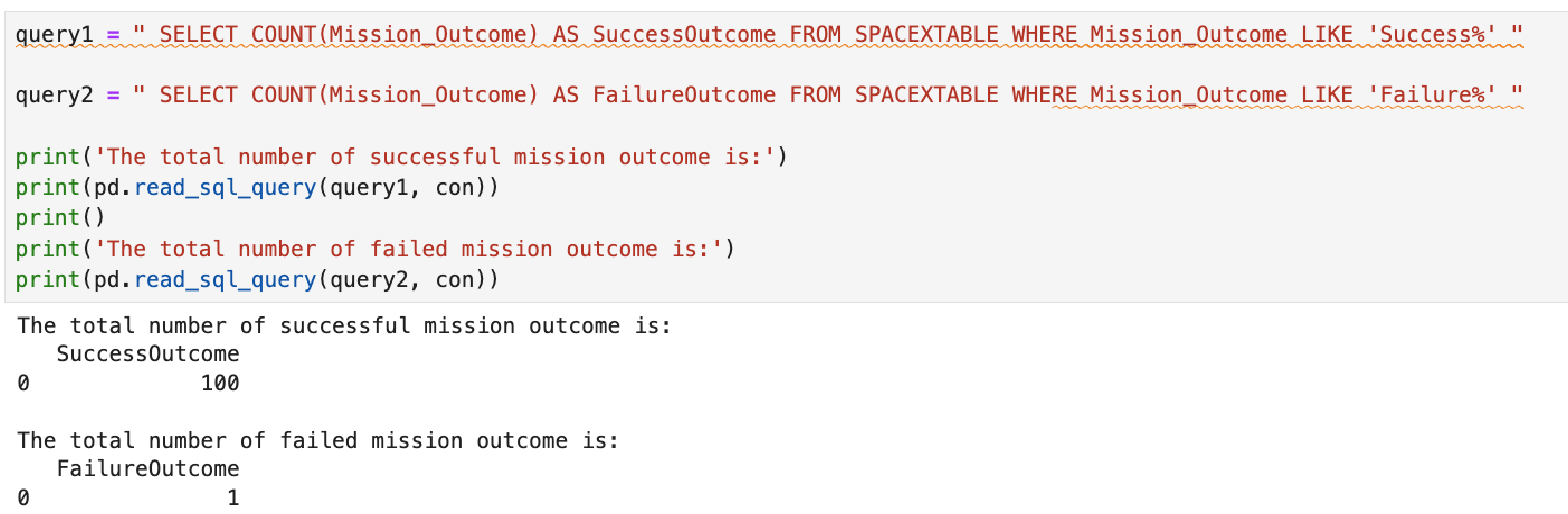


```
query = "SELECT Booster_Version FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 AND PA

pd.read_sql_query(query, con)
```

	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

- Using wildcard like ‘%’ to filter for WHERE MissionOutcome was a success or a failure.

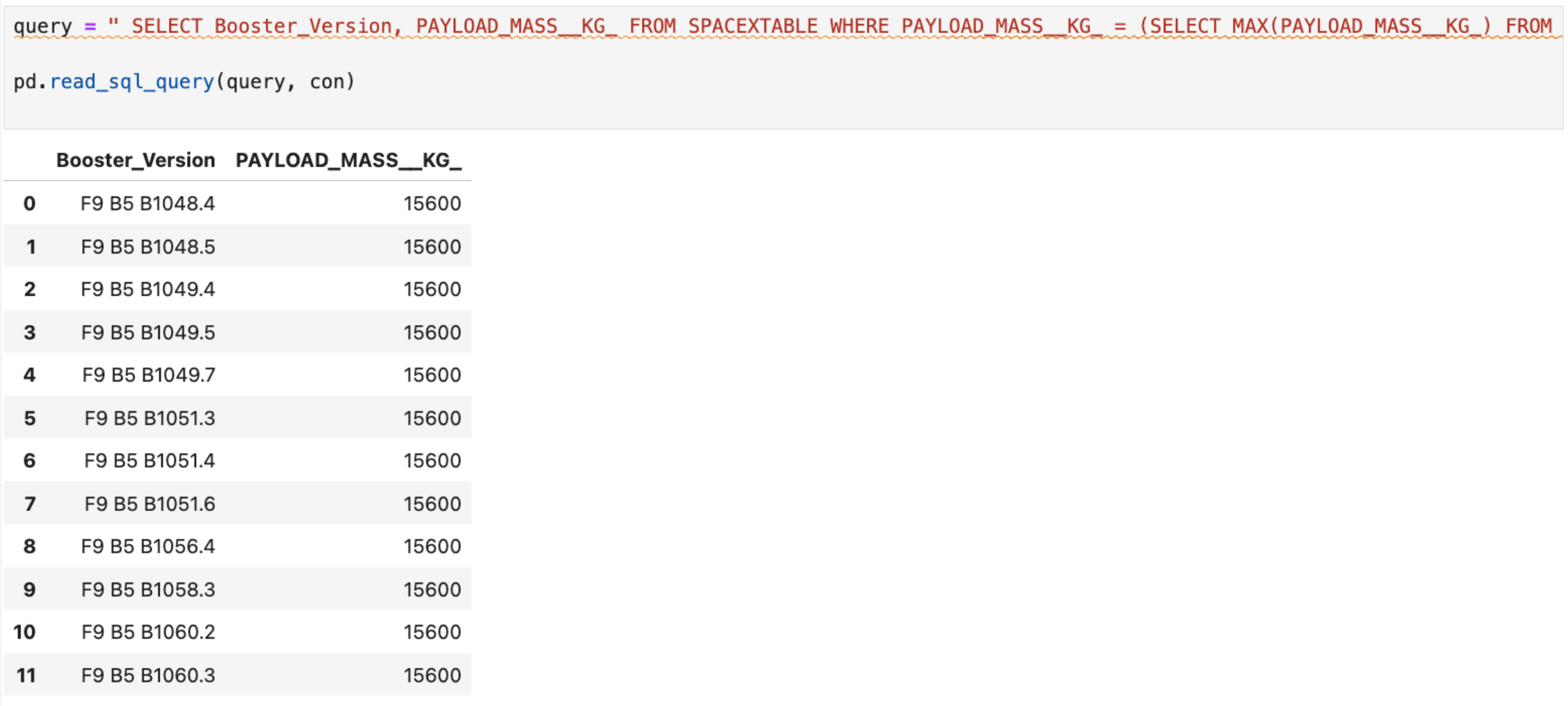


```
query1 = " SELECT COUNT(Mission_Outcome) AS SuccessOutcome FROM SPACEXTABLE WHERE Mission_Outcome LIKE 'Success%' "

query2 = " SELECT COUNT(Mission_Outcome) AS FailureOutcome FROM SPACEXTABLE WHERE Mission_Outcome LIKE 'Failure%' "

print('The total number of successful mission outcome is:')
print(pd.read_sql_query(query1, con))
print()
print('The total number of failed mission outcome is:')
print(pd.read_sql_query(query2, con))
```

```
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

- Determining the booster that have carried the maximum payload using a subquery in the WHERE clause and the MAX() function.

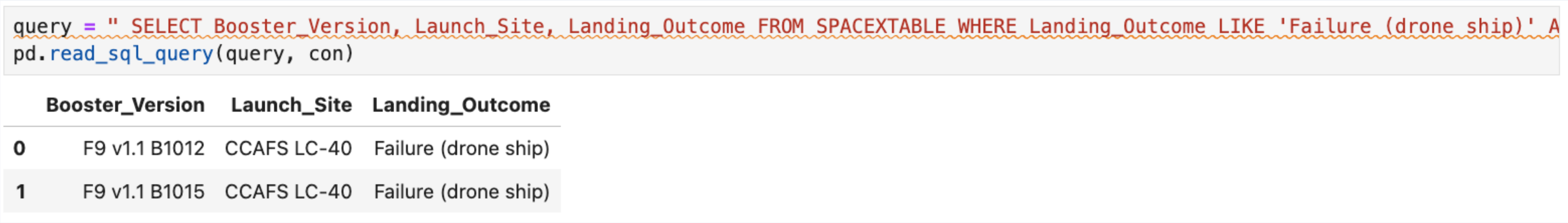


```
query = " SELECT Booster_Version, PAYLOAD_MASS__KG_ FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM

pd.read_sql_query(query, con)
```

	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

- Using a combinations of the WHERE clause, LIKE, AND, and BETWEEN conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

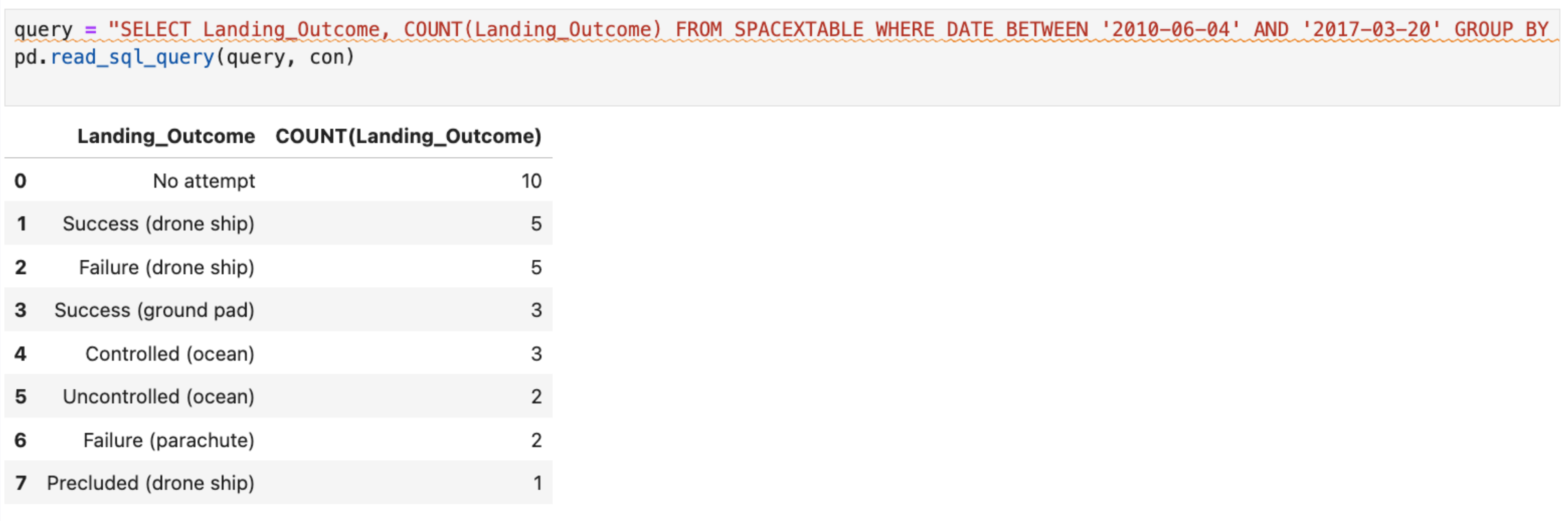


```
query = " SELECT Booster_Version, Launch_Site, Landing_Outcome FROM SPACEXTABLE WHERE Landing_Outcome LIKE 'Failure (drone ship)' A
pd.read_sql_query(query, con)
```

	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

- Selecting Landing outcomes and the COUNT of landing outcomes from the data and using the WHERE clause to filter for landing outcomes BETWEEN 2010-06-04 to 2017-03-20.
- Applying the GROUP BY clause to group the landing outcomes and the ORDER BY clause to order the grouped landing outcome in descending order.



```
query = "SELECT Landing_Outcome, COUNT(Landing_Outcome) FROM SPACEXTABLE WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY
pd.read_sql_query(query, con)
```

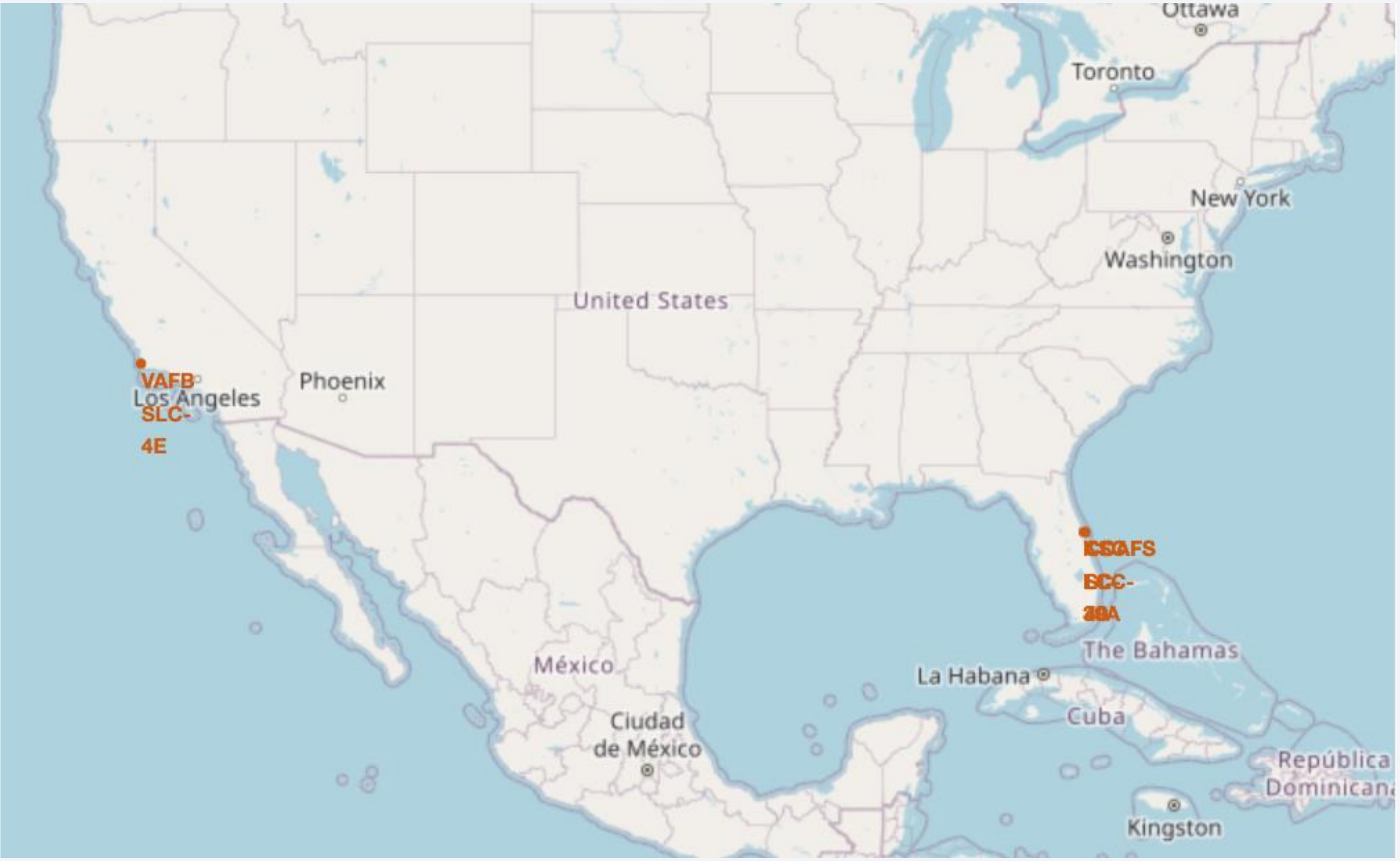
	Landing_Outcome	COUNT(Landing_Outcome)
0	No attempt	10
1	Success (drone ship)	5
2	Failure (drone ship)	5
3	Success (ground pad)	3
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Failure (parachute)	2
7	Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

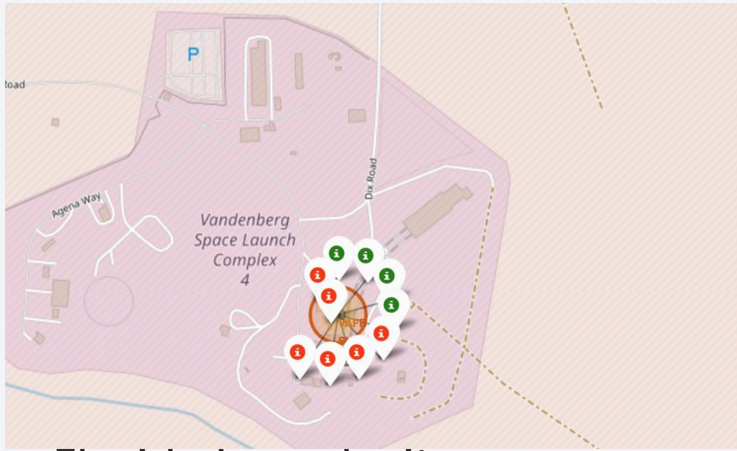
All launch sites global map markers

It can be seen that the SpaceX launch sites are in the USA coasts, Florida and California.

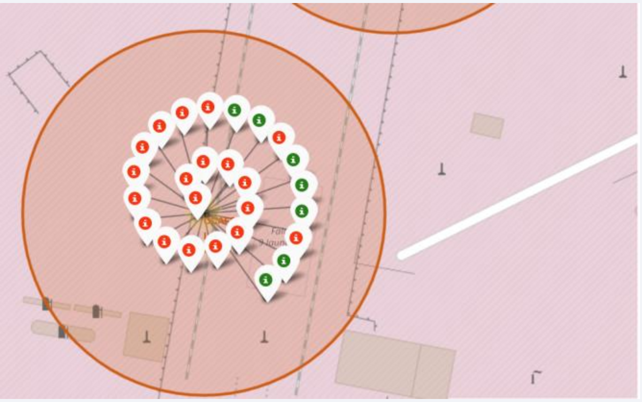


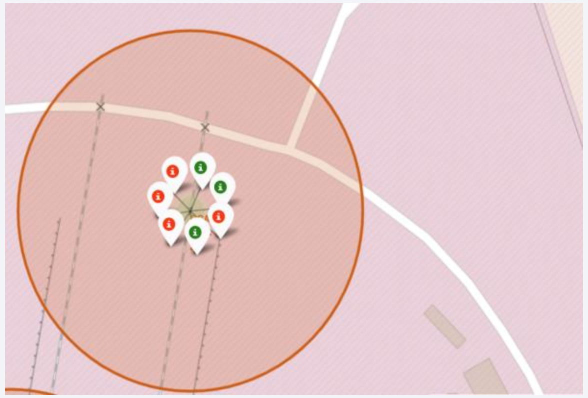
Markers showing launch sites with color labels

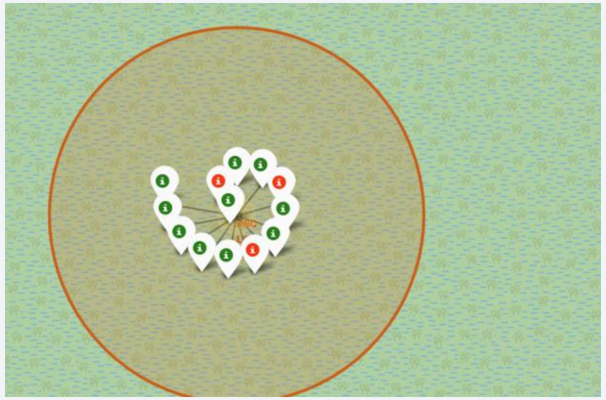
- Green markers show succesful Launches and Red markers show failures.
- California launch site:



- Florida Launch sites:

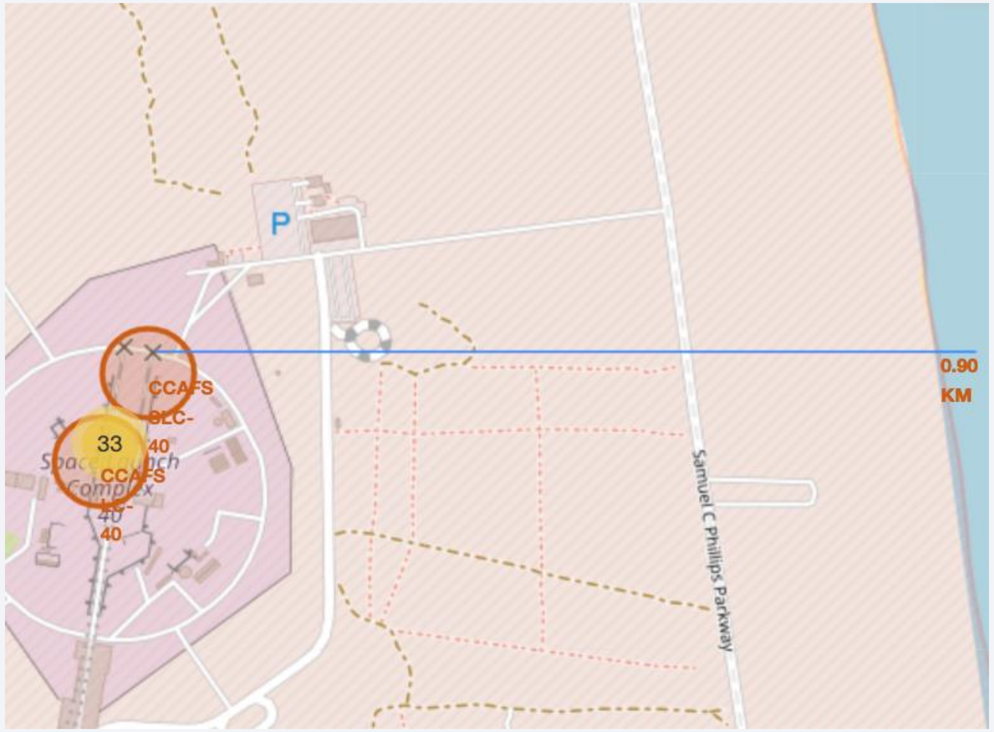




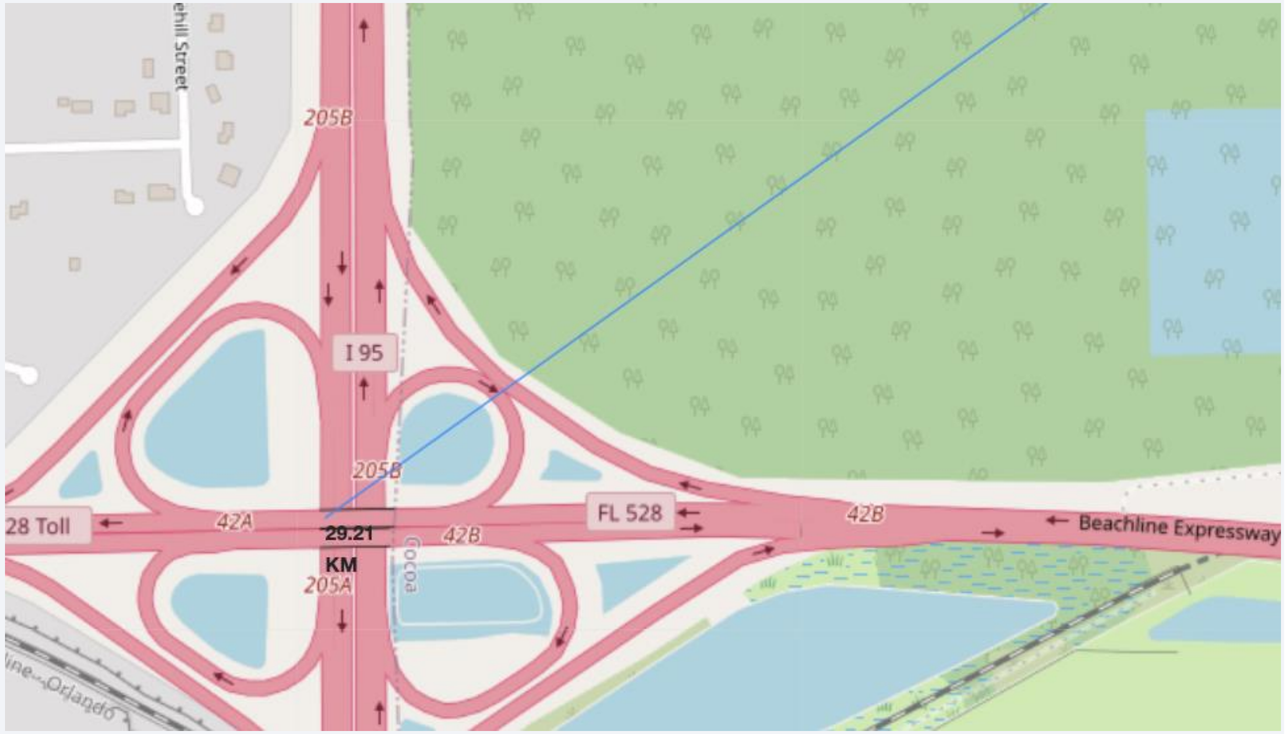


Launch Site distance to landmarks 1

- Distance to Coastline:

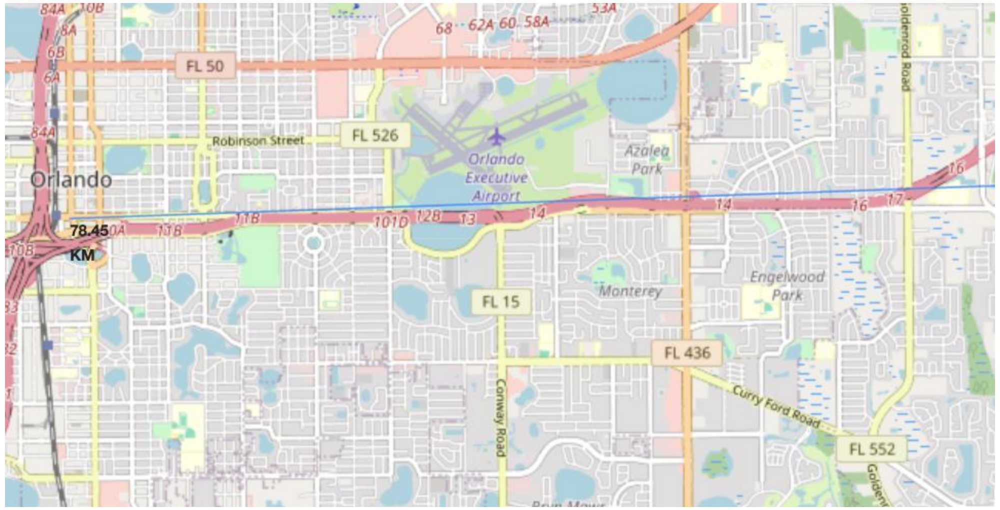


Distance ho closest Highway:

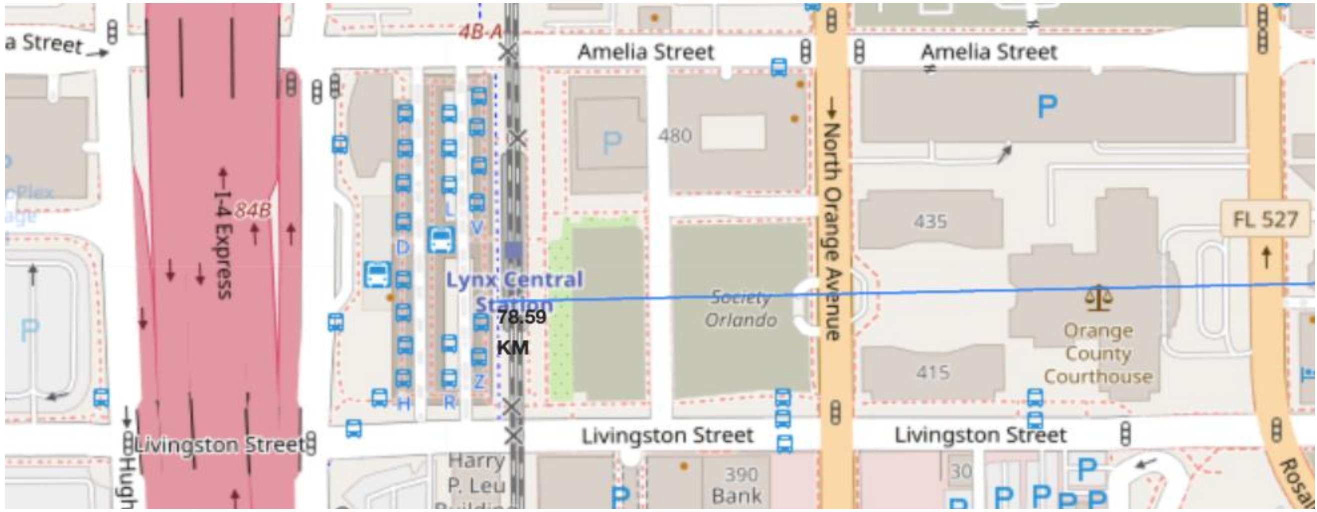


Launch Site distance to landmarks 2

- Distance to city:



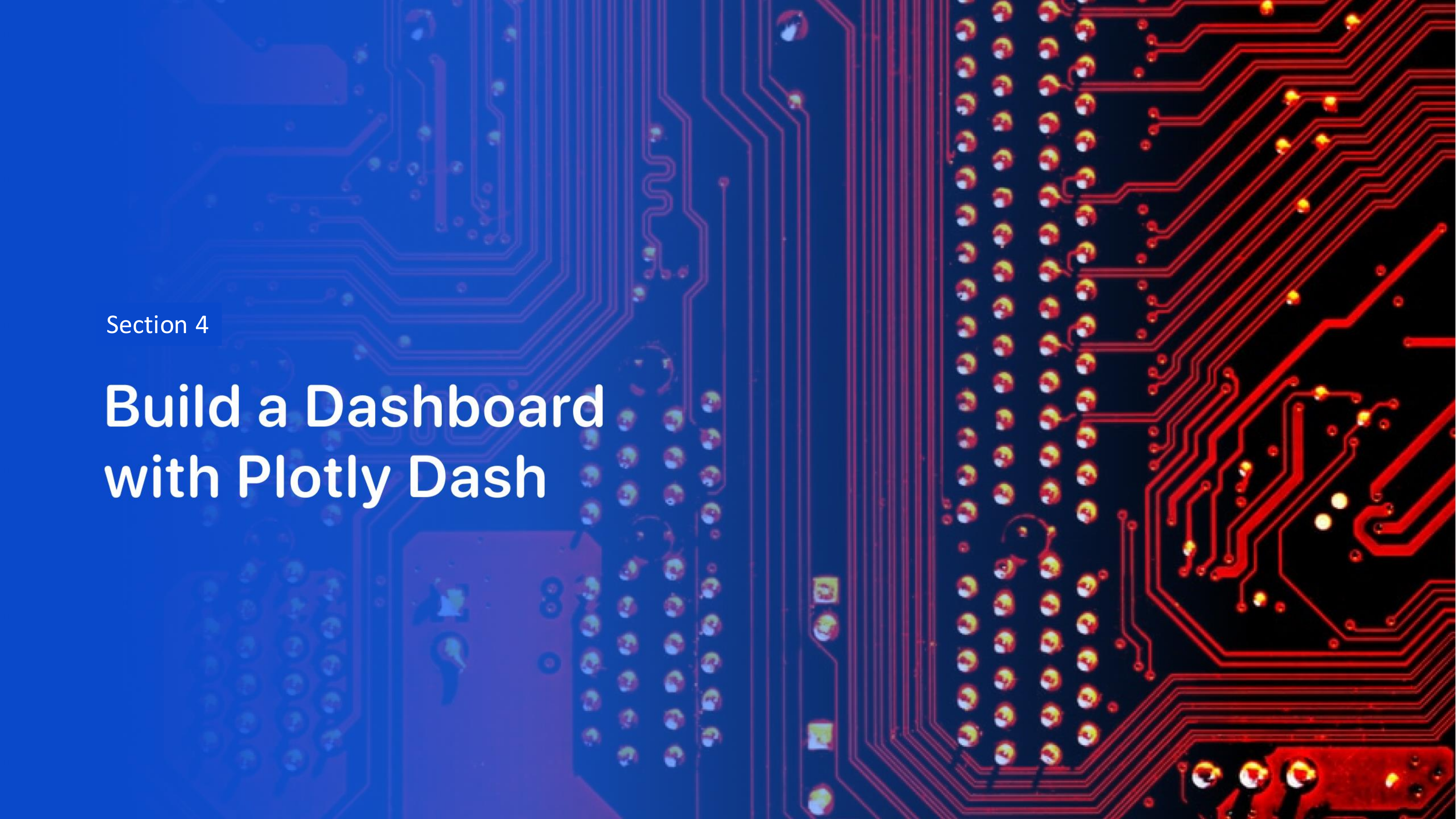
Distance to Railway station:



- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

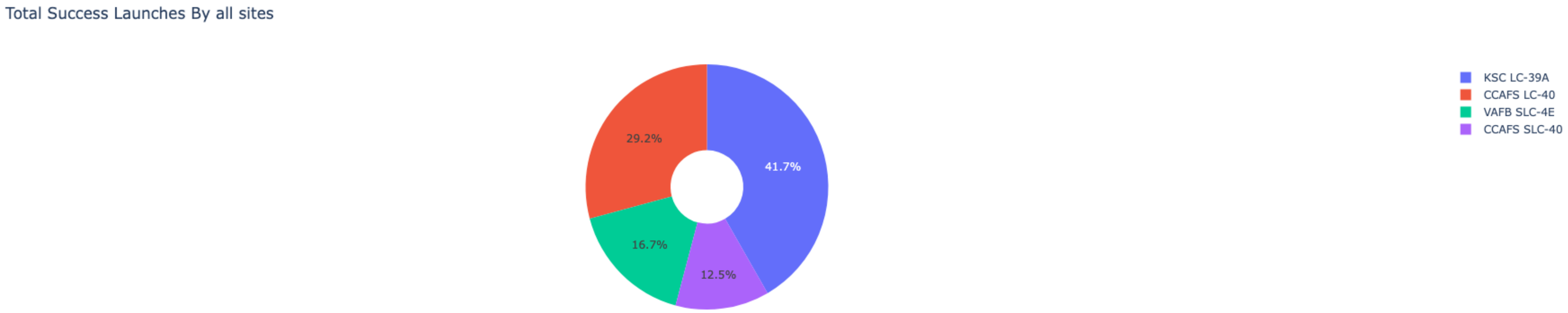
Section 4

Build a Dashboard with Plotly Dash



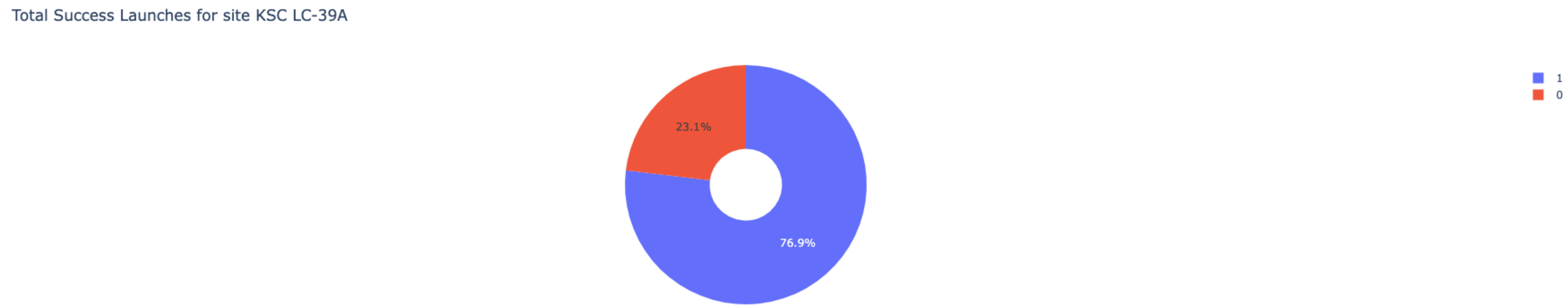
Pie chart showing the success percentage achieved by each launch site

- It can be seen that KSC LC-39A had the most succesful 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.

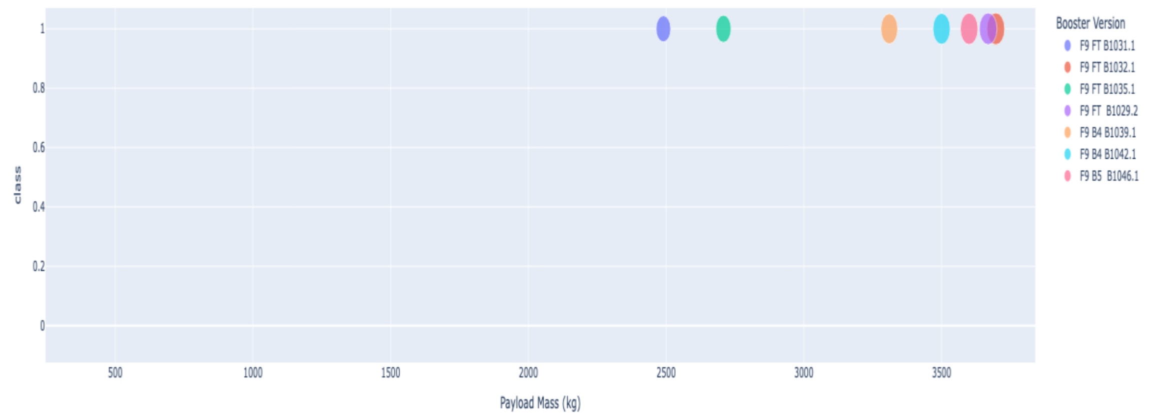


Total Success Launches for site KSC LC-39A

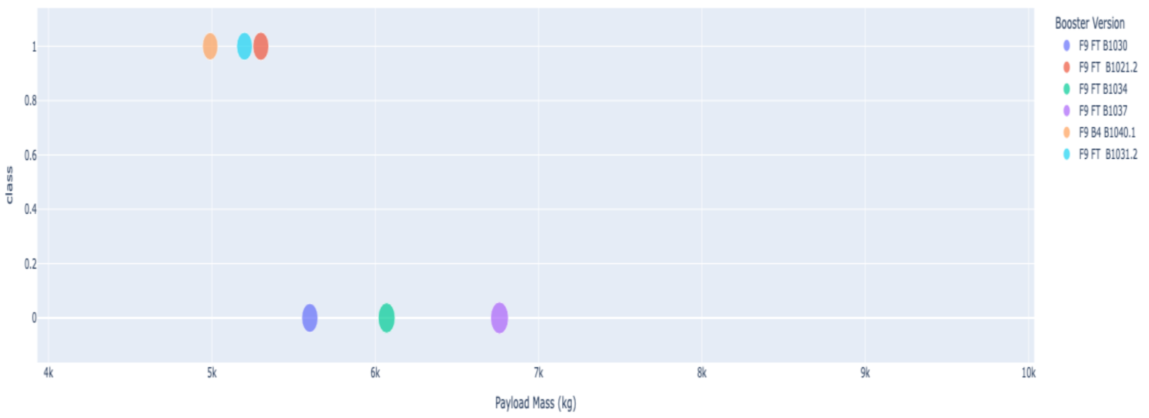
Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider

- It can be seen the success rates for low weighted payloads is higher than the heavy weighted payloads.

Low weighted Payload 0kg-4000kg



Heavy Weighted Payload 4000kg-10000kg

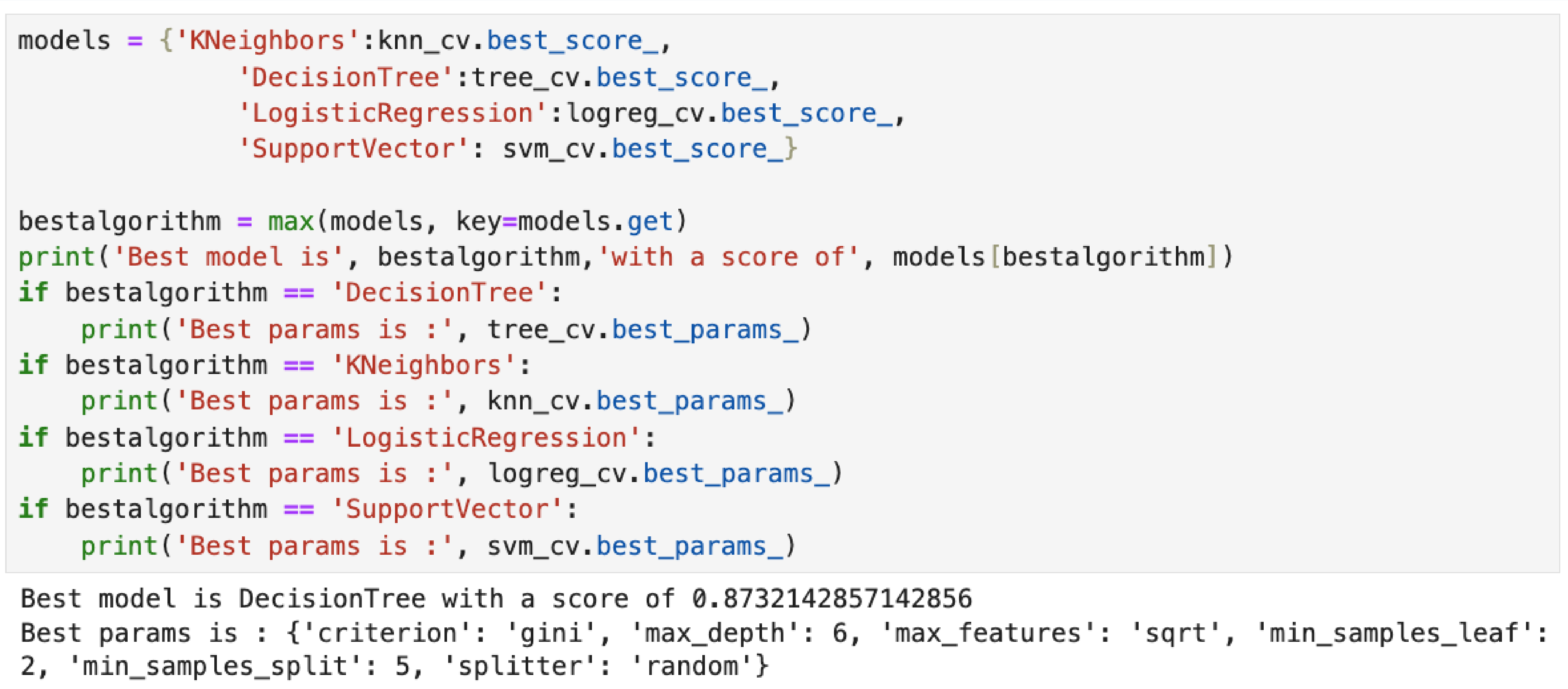


Section 5

Predictive Analysis (Classification)

Classification Accuracy

- The decision tree classifier is the model with the highest classification accuracy

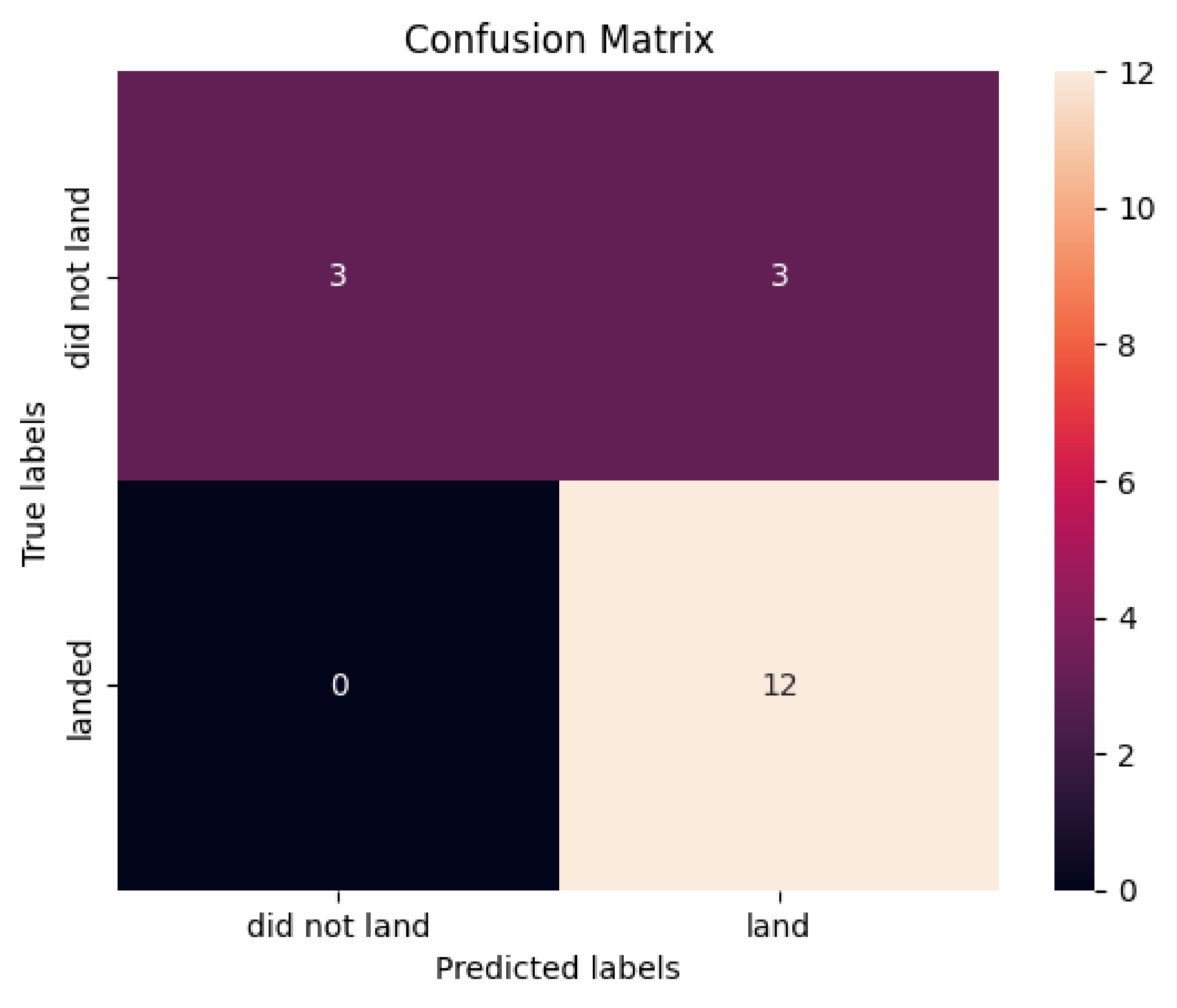


```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

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': 'sqrt', 'min_samples_leaf':
2, 'min_samples_split': 5, 'splitter': 'random'}
```

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.



Conclusions

It can be concluded 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.

Thank you!