SPACEX

IBM

IBM Developer SKILLS NETWORK

IBM Data Science Professional Certificate Capstone Project

Ingrid Lindstrom

February 2022

Outline

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

Executive Summary

- **Methodologies Utilized**
  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - EDA with Data Visualization
  - Interactive Data Analytics with Folium
  - Machine Learning Prediction
- **Results Summary**
  - Exploratory Data Analysis results
  - Interactive Analytics demonstration with screenshots
  - Predictive Analytics results

Introduction

- **Project Background and Context**

The purpose of this project is to 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. The majority of these savings is due to SpaceX's ability to reuse the first stage of the rocket. If I can determine if the first stage of a Falcon 9 rocket will land, I can determine the cost of the launch. This financial information can be used if a rival competitor wants to bid against SpaceX for rocket launches and other aerospace engineering endeavors.

- **Pertinent Questions that can be answered with Data Science techniques**
  - What **factors** determine if the rocket will land successfully?
  - Which parameters determine the **success rate** of a successful landing?
  - What **operating conditions** need to be in place to ensure a successful landing?

Section 1

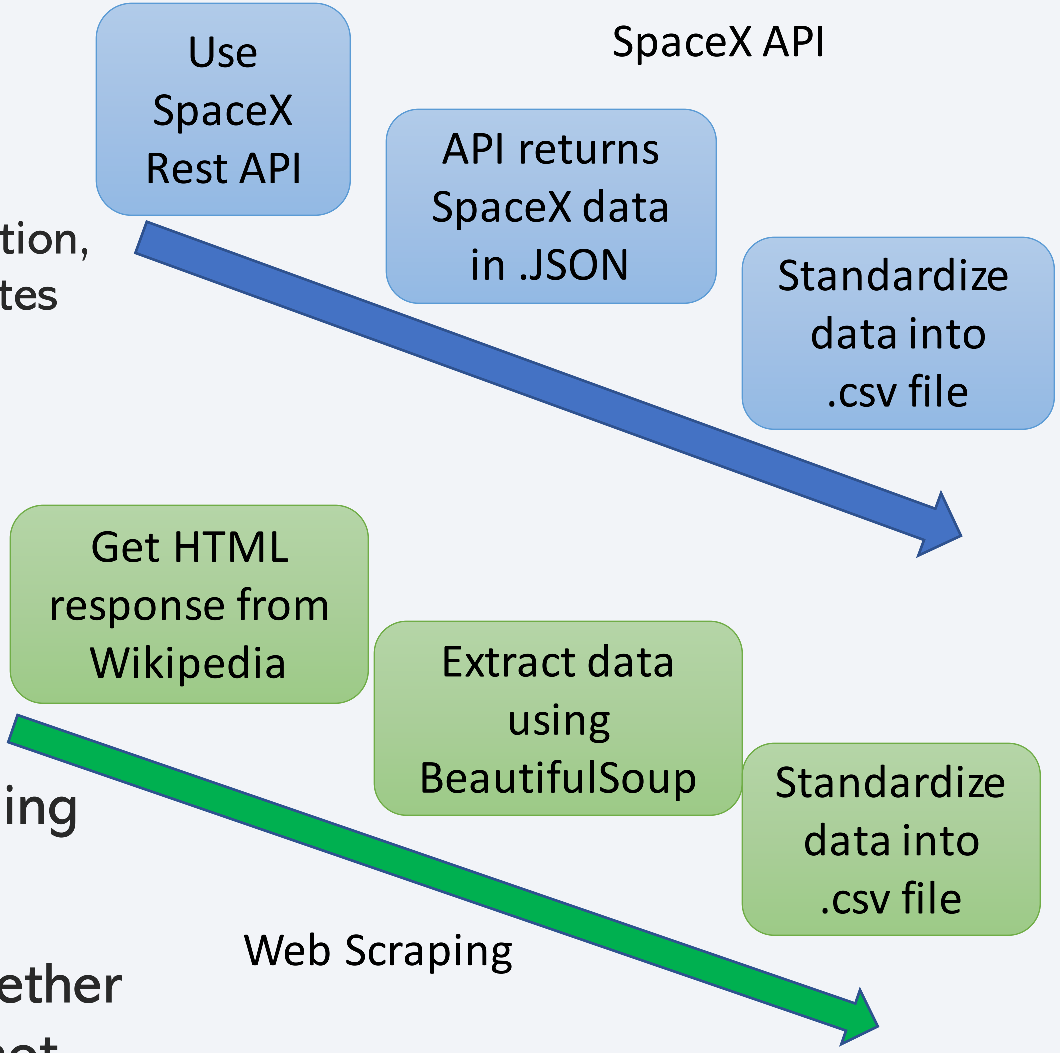
Methodology

Methodology Overview

- Data collection methodology
  - SpaceX Rest API
  - Web Scraping from Wikipedia – using Python library *BeautifulSoup*
- Perform data wrangling
  - One Hot Encoding was applied to categorical features in the dataframe
  - OHE prepares fields for Machine Learning, eliminating redundant or irrelevant columns
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash – dynamic results in real time!
- Perform predictive analysis using classification models
  - Building, tuning, and evaluating the highest accuracy of various classification models

Data Collection

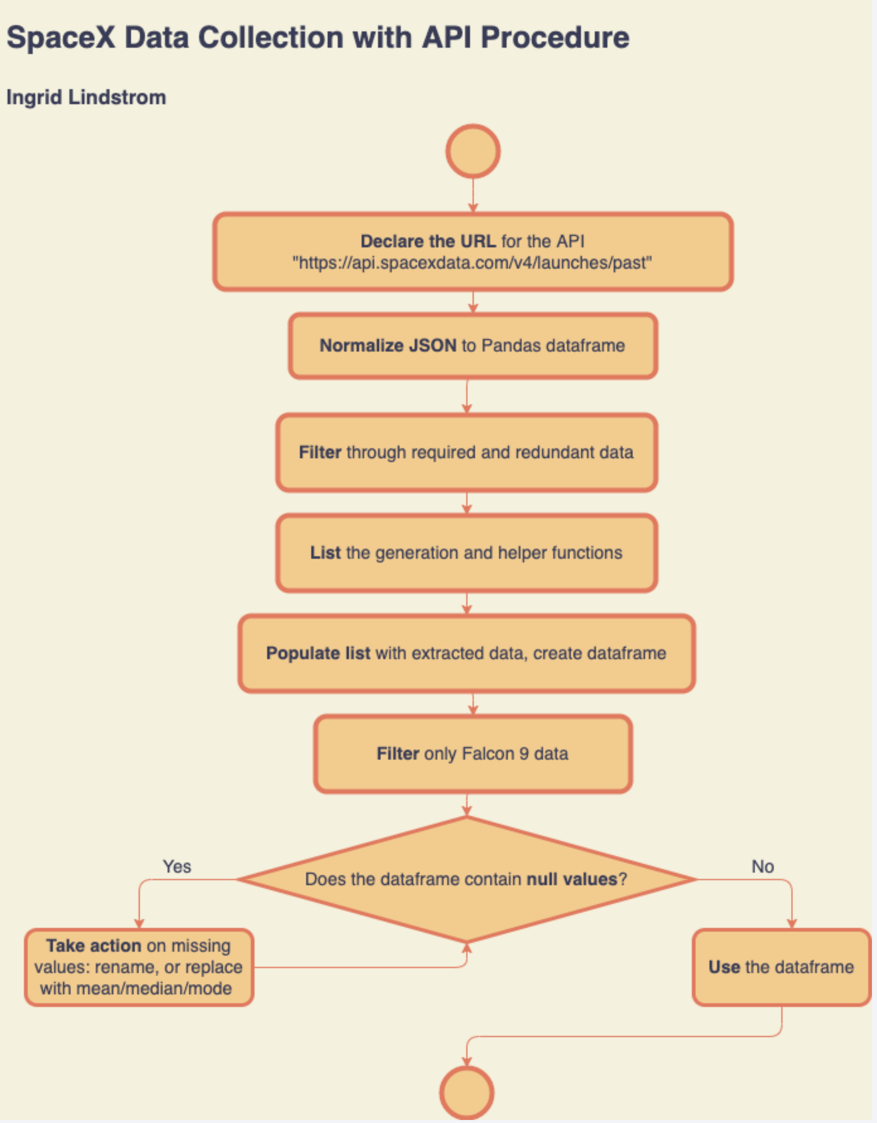
- The primary data set used in this project was gathered from the SpaceX Rest API.
  - This API gives us the relevant data about launch information, including the model number of the rocket, geographic sites utilized, the payload delivered, launch and landing specifications, and landing outcome.
  - The SpaceX Rest endpoints, or URL, is obtained from api.spacexdata.com/v4/
- An additional data source for obtaining Falcon 9 launch data is web scraping a Wikipedia page using *BeautifulSoup*, a Python library used for pulling data out of XML and HTML files.
- The ultimate goal is to use this data to predict whether SpaceX will attempt to land a Falcon 9 rocket or not.



Data Collection – SpaceX API

Core Procedure: Declare URL, Normalize JSON, Filter, List, Populate, Specific Filter, Take Action on Null Values

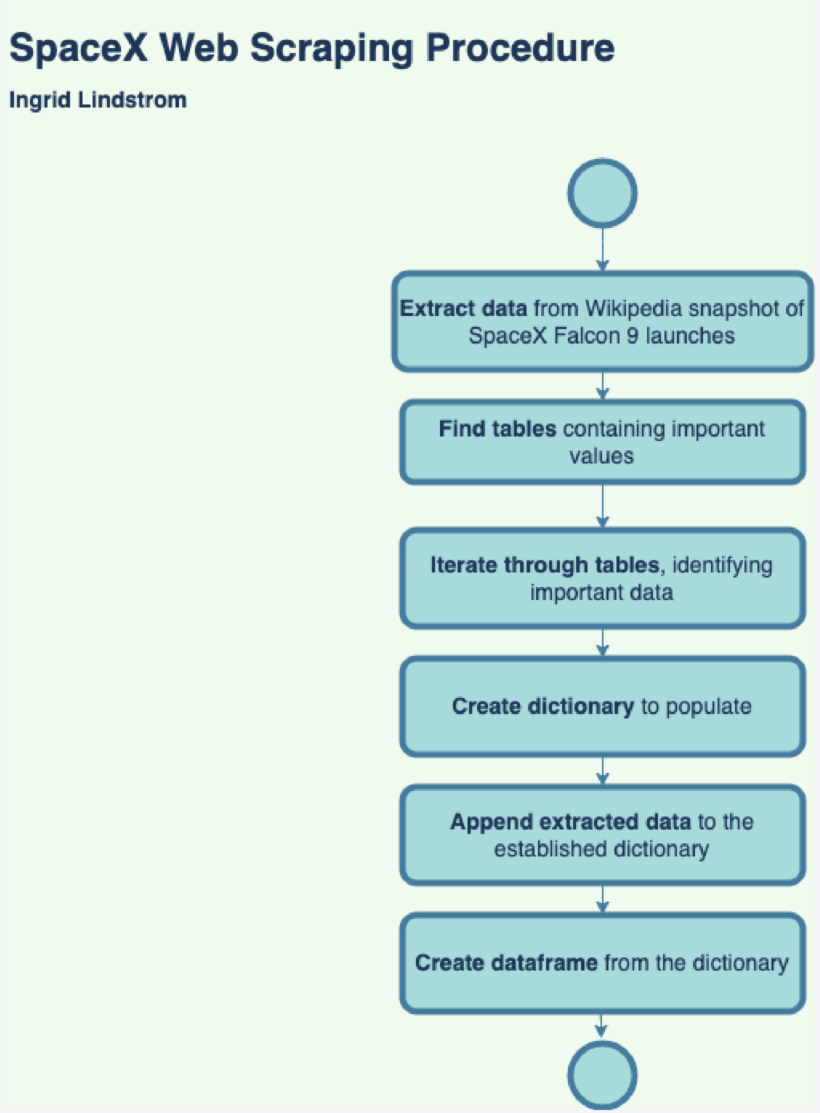
GitHub URL to [API Jupyter Notebook]()



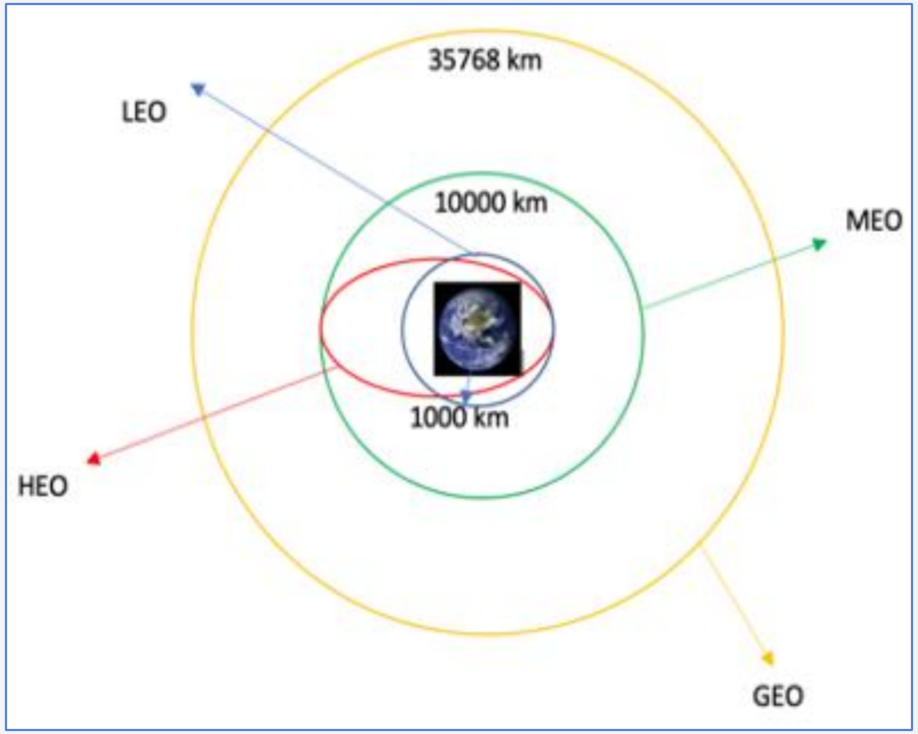
Data Collection – Web Scraping

Core Procedure: Extract Data, Find Tables, Iterate, Create Dictionary, Append Extracted Data, Create Dataframe

GitHub URL link to [Web Scraping Jupyter Notebook]



Data Wrangling



Each SpaceX launch is aimed at a dedicated orbit. This figure depicts some common orbit types.

- In the data set, there are several possible outcomes for booster landings; these are labeled appropriately for sorting and analysis.
- I converted these outcomes to binary training labels: 1 for successful or 0 for unsuccessful
- True Ocean: successful landing to a specific ocean region; False Ocean: unsuccessful landing
- True RTLS: successful landing to on a ground pad; False RTLS: unsuccessful landing
- True ASDS: successful landing on a drone ship; False ASDS: unsuccessful landing
- GitHub URL link to Data Wrangling Notebook

Data Wrangling, continued

This is the procedure utilized for converting outcomes into training labels to determine 1 for a successful booster landing, or 0 for an unsuccessful booster landing.

Perform Exploratory Data Analysis (EDA) on dataset

Calculate the number of **launches** at each site

Calculate the **number** and **occurrence** of each orbit

Calculate the number and occurrence of **mission outcome per orbit type**

Work out the success rate for every landing in the data set

Create a "Landing Outcome" label from the Outcome column

Export dataset as .csv file

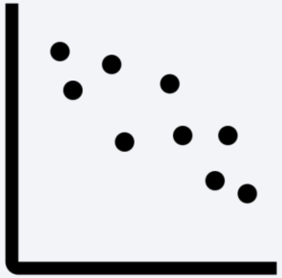
EDA with Data Visualization

GitHub Link to EDA Notebook

Scatterplot Graphs

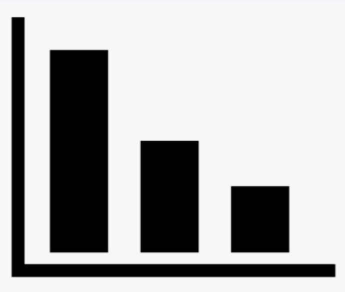
Used to show the relationship between two variables; best for large datasets.

- Flight Number vs. Payload Mass
- Flight Number vs. Launch Site
- Payload vs. Launch Site
- Orbit vs. Flight Number
- Payload vs. Orbit Type
- Orbit vs. Payload Mass



Bar Graphs

Convenient to compare categories on one axis, and discrete values on the other. Used to see clear differences between groups.



- Mean Value vs. Orbit

Line Graphs

These depict variables and trends over time; extrapolations are used to form predictions.

- Success Rate vs. Year

EDA with SQL

[GitHub Link to SQL Lite Notebook]

DISPLAY QUERIES

- ✓ Display all unique names of distinct launch sites for each rocket launch mission
- ✓ Display 5 records where the launch sites begin with the string "KSC"
- ✓ Display the **total** payload mass (kg) carried by boosters launched by NASA (CRS)
- ✓ Display the average payload mass (kg) carried by Falcon 9 version 1.1

LISTING QUERIES

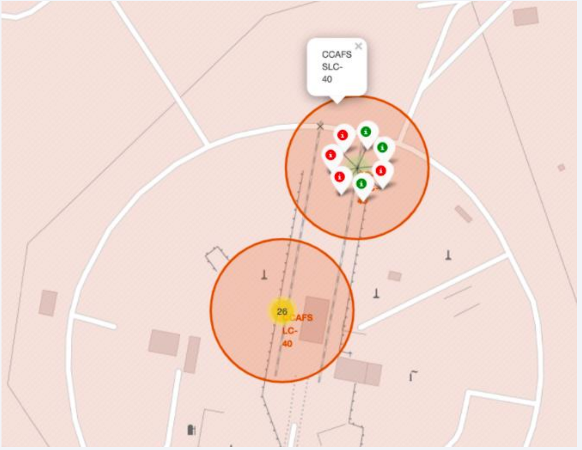
- ✓ List the date of the successful landing outcome [drone ship]
- ✓ List the names of the boosters of successful landing outcomes [ground pad] with subquery of calculating payload mass greater than 4000 but less than 6000 kg
- ✓ List the **total number** of successful and failure outcomes
- ✓ List the names of [Booster_Versions] which carried the maximum payload mass
- ✓ List the records of month names, successful landing outcomes for [ground pad], booster versions, launch sites in 2017

RANKING QUERY

- ✓ Rank the count of successful landing outcomes between June 2010 and March 2017 in descending order

Build an Interactive Map with Folium

- Folium allows us to illustrate data with movement, and easily identifies successful or failed launches
  - Take Latitude & Longitude coordinates of each launch site
  - Add a circle marker around each site, labeled clear with the site name
- Assign dataframe launch_outcomes(failures, successes) to class 0 and 1 with red and green markers, respectively, using MarkerCluster()
- Calculate the distance between Launch Sites and various landmarks (railways, highways, coastline) to identify trends. Lines are drawn on the map to indicate distance



Launch Site Trends

Close in proximity to railways? NO

Close in proximity to highways? NO

Close in proximity to coastlines? YES

Certain distance away from cities? YES

GitHub URL Link to [Interactive Visual Analytics with Folium Jupyter Notebook]()

Build a Dashboard with Plotly Dash

A live website dashboard built with Dash shows real-time data updates with graphs and charts.

CHARTS	GRAPHS
• A **pie chart** shows the total launches at a specific site or ALL sites • The size of the circle (pie) can be scaled proportionally to the total quantity • Plotly Dash displays relative proportions of multiple classes of data	• A **scatterplot** shows the relationship between Launch Outcome and Payload Mass (kg) for various Booster Versions • The relationship between two variables are depicted • Most reliable method to display a non-linear pattern • Minimum and maximum values are shown in a range

GitHub URL Link to [Plotly Dash source code]

Predictive Analysis (Classification)

GitHub Link to [Machine Learning Prediction Jupyter Notebook]

1. BUILD
- Load the dataset into NumPy & Pandas
- Transform the data
- Split the data into training and testing sets
- Consider which algorithms should be used
- Set the parameters & algorithms to GridSearchCV
- Fit the datasets into GridSearchCV objects and train the models

2. EVALUATE
- Check the accuracy of each model
- Tune the hyperparameters for each type of algorithm
- Plot the Confusion Matrix, make sure colors are consistent

3. IMPROVE
- Proceed to feature engineering
- Algorithm tuning
- Re-run the accurate models, noting any discrepancies

4. DECIDE
- Look at the hard numbers
- The model with the best accuracy score = **best performance**

Results

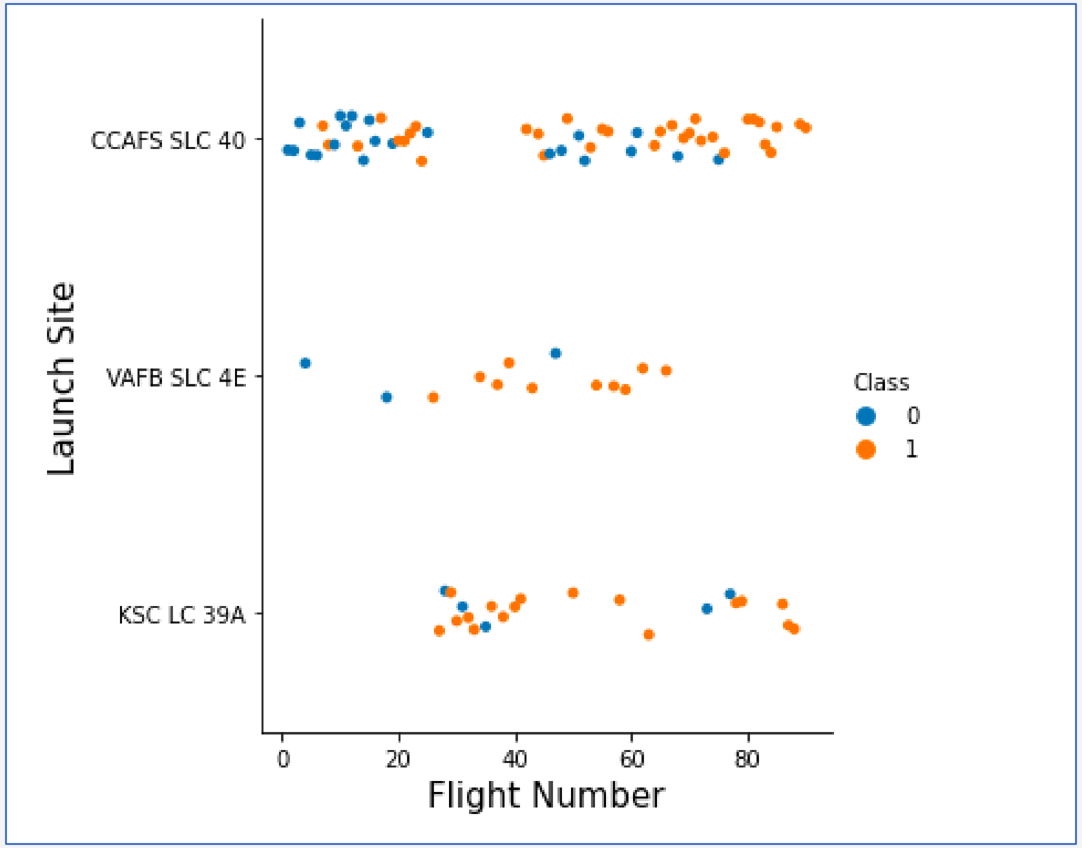
- Exploratory Data Analysis results
  - SQL comparisons, contrasts, lists, rankings, and selections
- Interactive Analytics demo in screenshots
  - Folium, Dash charts and scatterplots
- Predictive Analysis results
  - Examining the most effective machine learning models to

    turn data into decisions!

Section 2

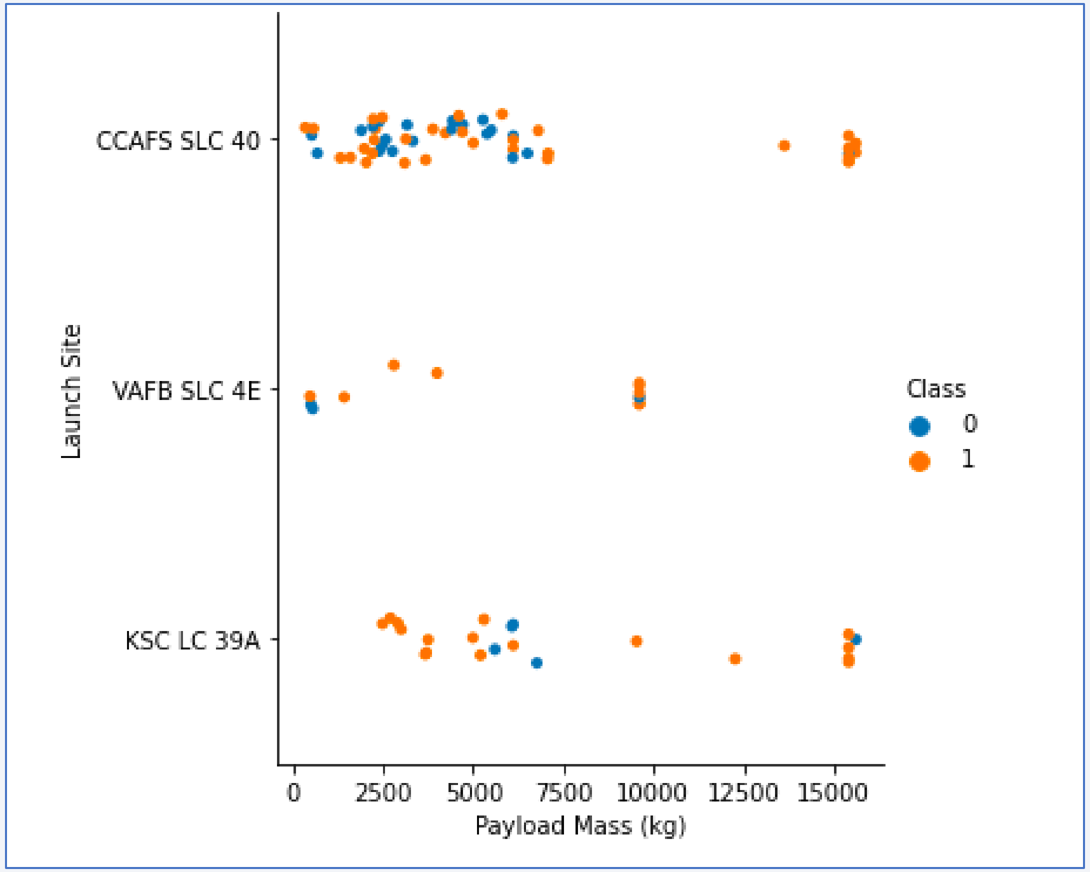
Insights Drawn from
Exploratory Data Analysis (EDA)

Flight Number vs. Launch Site



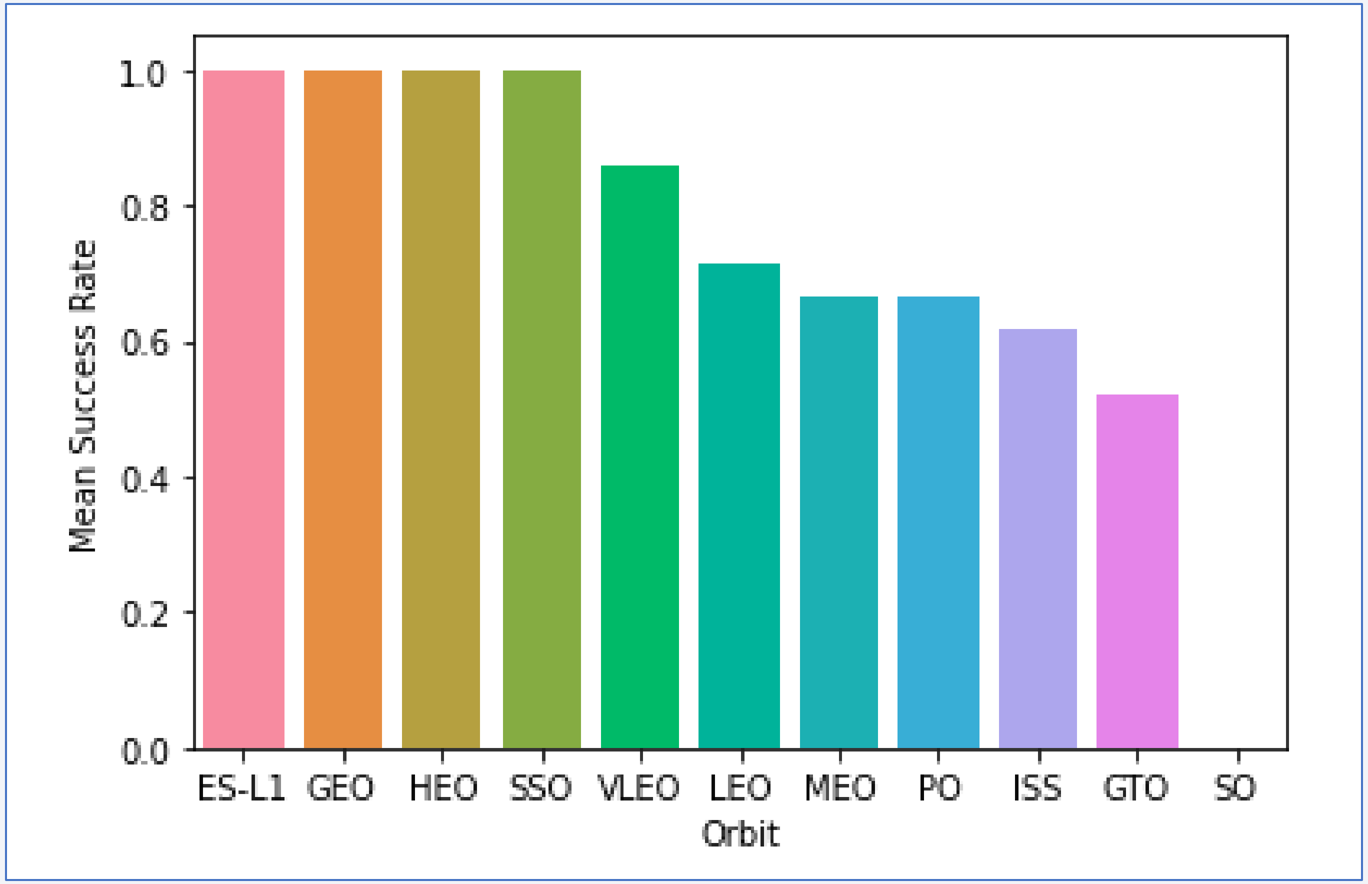
If the number of flights is greater than 30, the higher the success rate of the launch at that site.

Payload vs. Launch Site



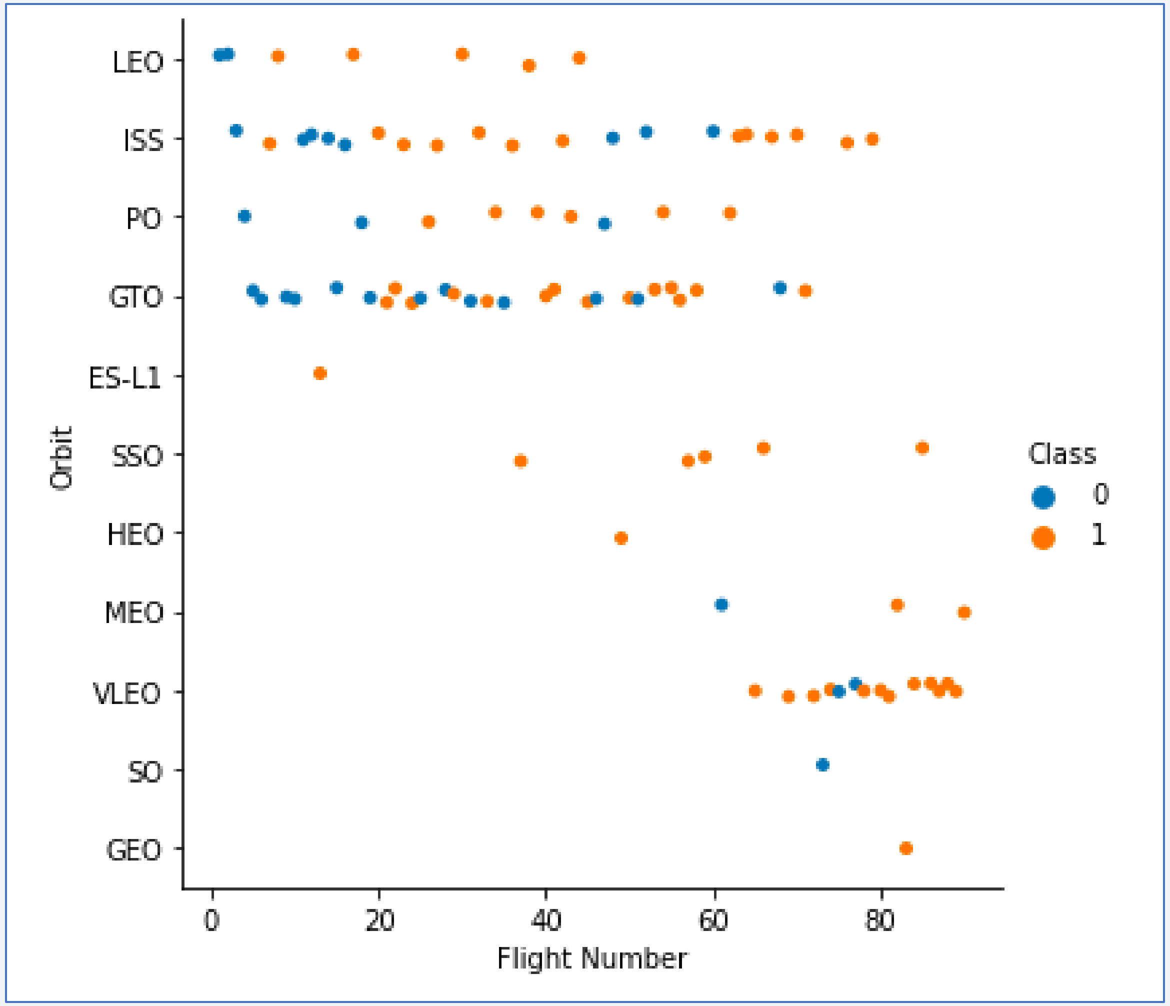
At launch site CCAFS SLC 40, the greater the payload mass, the higher the success rate for the rocket launch.

Success Rate vs. Orbit Type



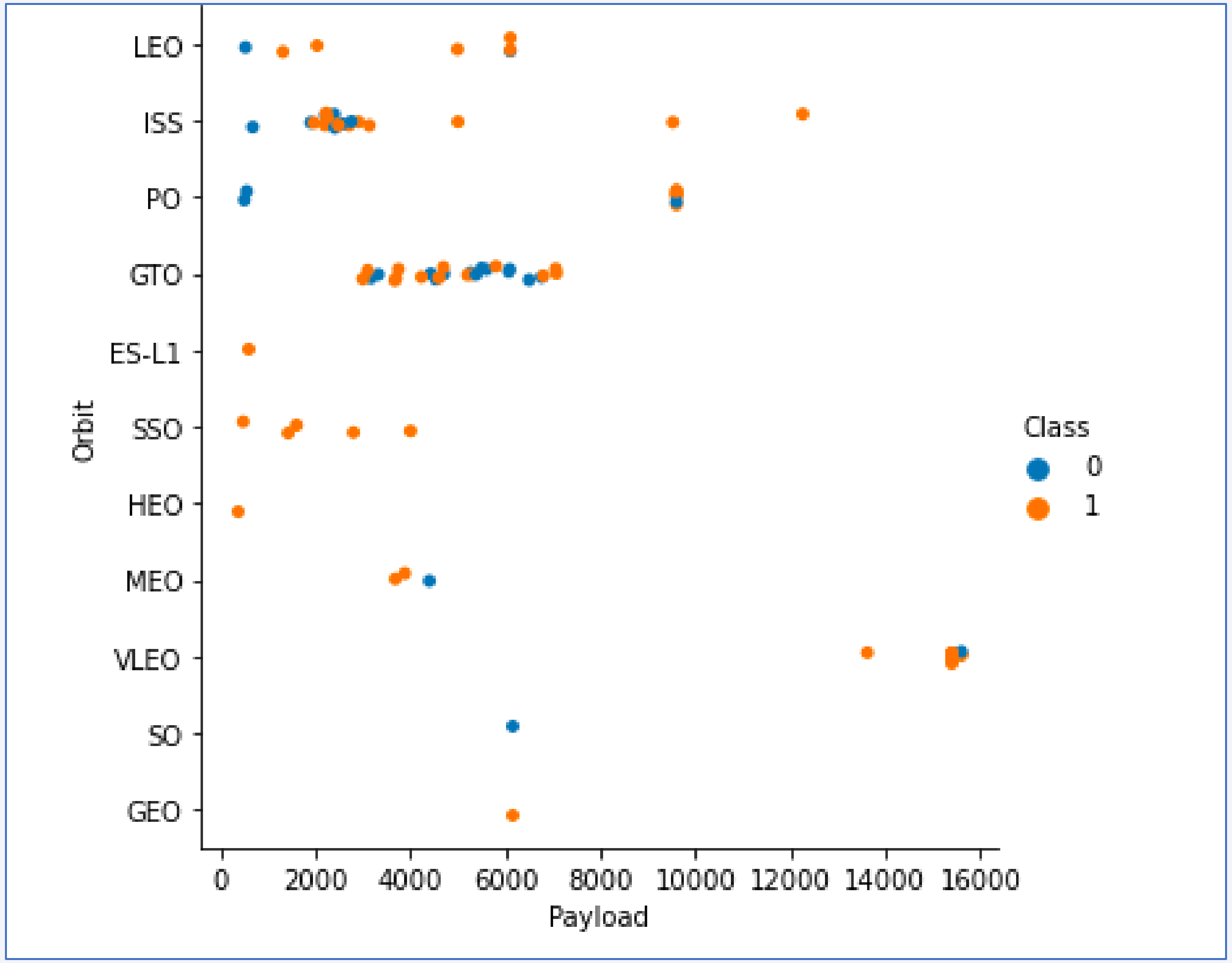
Four orbits: ES-L1, GEO, HEO, and SSO, had the highest mean success rates. These are the most stable and dependable orbits for the SpaceX rockets.

Flight Number vs. Orbit Type



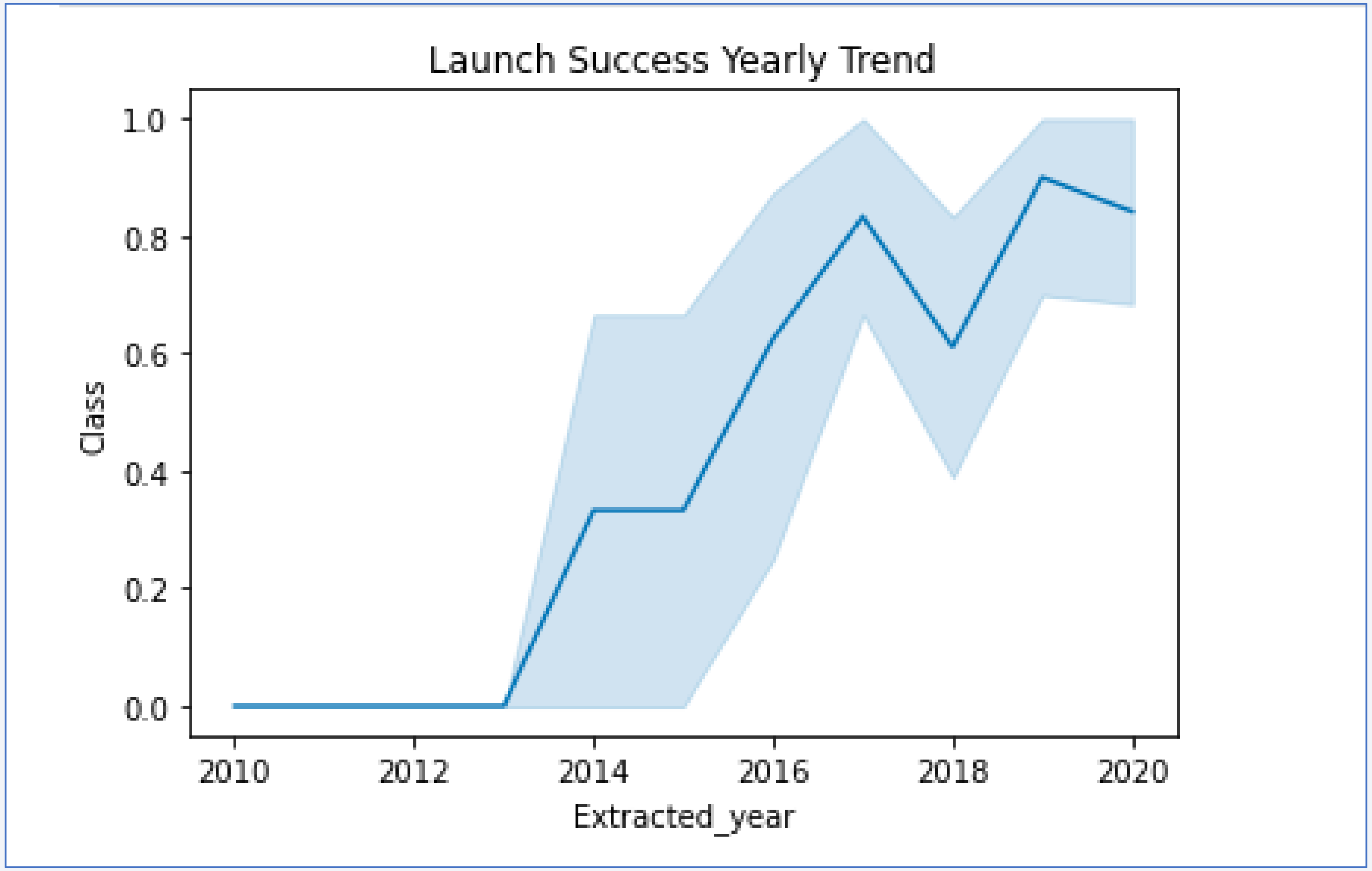
- In LEO orbits, successful outcomes are correlated with the number of flights
- In GTO orbits, there is no relationship between number of flights and successful outcomes
- Similarly, SSO orbits are consistently successful, but does not appear to be dependent on number of flights

Payload vs. Orbit Type



In launches with heavier payloads, successful landings are higher for PO, LEO and ISS orbits.

Launch Success Yearly Trend



Except for a dip in 2018, the launch success rate has continuously increased each year from 2013 through 2020.

All Launch Site Names

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

```
In [7]: df["Launch_Site"].unique()

Out[7]: array(['CCAFS LC-40', 'VAFB SLC-4E', 'KSC LC-39A', 'CCAFS SLC-40'],
              dtype=object)
```

- SQLite Query Word: UNIQUE
- This command shows only unique values in the **Launch_Site** column from **tblSpaceX**

Launch Site Names Begin with "KSC"

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

In [8]:
```
df["Launch_Site"].str.startswith("KSC")
b = df["Launch_Site"].str.startswith("KSC")
df[b].head()
```

Out[8]:

	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
29	19-02-2017	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
30	16-03-2017	06:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	5600	GTO	EchoStar	Success	No attempt
31	30-03-2017	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
32	01-05-2017	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300	LEO	NRO	Success	Success (ground pad)
33	15-05-2017	23:21:00	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070	GTO	Inmarsat	Success	No attempt

- SQLite Query Words: STARTSWITH
- Using **HEAD** will show only five records from the dataframe; **STARTSWITH** will yield only **Launch_Site** records beginning with the string "KSC"

Total Payload Mass

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

In [9]:
```
df["Customer"] == "NASA (CRS)"
n = df["Customer"] == "NASA (CRS)"
df[n]["PAYLOAD_MASS__KG_"].sum()
```

Out[9]:
```
45596
```

- SQLite Query Word: SUM
- The total payload mass carried by NASA CRS boosters was 45,496kg.

Average Payload Mass by F9 v1.1

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

In [12]:
```
df["Booster_Version"] == "F9 v1.1"
F9 = df["Booster_Version"] == "F9 v1.1"
df[F9]["PAYLOAD_MASS__KG_"].mean()
```

Out[12]:
```
2928.4
```

- SQLite Query Word: MEAN
- The average payload mass carried by booster version F9 v1.1 is 2,928.4kg

First Successful Ground Landing Date

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

Hint:Use min function

In [25]:
```
df["Landing _Outcome"] == "Success (drone ship)"
Achieve = df["Landing _Outcome"] == "Success (drone ship)"
df[Achieve]
df[Achieve]["Date"]
```

Out[25]:
```
22    08-04-2016
23    06-05-2016
24    27-05-2016
27    14-08-2016
28    14-01-2017
31    30-03-2017
35    23-06-2017
36    25-06-2017
39    24-08-2017
41    09-10-2017
42    11-10-2017
43    30-10-2017
52    18-04-2018
53    11-05-2018
Name: Date, dtype: object
```

- SQLite Query Words: DATE, MIN
- All the dates in the list correspond to a successful [drone ship] landing outcome. Using MIN would reveal that the earliest date is April 8, 2016.

Successful Drone Ship Landing with Payload between 4000 and 6000

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

In [30]:
```
df["Landing _Outcome"].unique()
```

Out[30]:
```
array(['Failure (parachute)', 'No attempt', 'Uncontrolled (ocean)',
       'Controlled (ocean)', 'Failure (drone ship)',
       'Precluded (drone ship)', 'Success (ground pad)',
       'Success (drone ship)', 'Success', 'Failure', 'No attempt '],
      dtype=object)
```

In [39]:
```
(df["PAYLOAD_MASS__KG_"] < 4000 )& (df["PAYLOAD_MASS__KG_"] > 6000)
```

Out[39]:
```
0      False
1      False
2      False
3      False
4      False
       ...
96     False
97     False
98     False
99     False
100    False
Name: PAYLOAD_MASS__KG_, Length: 101, dtype: bool
```

In [52]:
```
Boostername = (df["Landing _Outcome"] == "Success (ground pad)") & (df["PAYLOAD_MASS__KG_"] > 4000) & (df["P
df[Boostername]
```

Out[52]:

	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
32	01-05-2017	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300	LEO	NRO	Success	Success (ground pad)
40	07-09-2017	14:00:00	F9 B4 B1040.1	KSC LC-39A	Boeing X-37B OTV-5	4990	LEO	U.S. Air Force	Success	Success (ground pad)
46	08-01-2018	01:00:00	F9 B4 B1043.1	CCAFS SLC-40	Zuma	5000	LEO	Northrop Grumman	Success (payload status unclear)	Success (ground pad)

In [53]:
```
df[Boostername]["Booster_Version"]
```

Out[53]:
```
32    F9 FT B1032.1
40    F9 B4 B1040.1
46    F9 B4 B1043.1
Name: Booster_Version, dtype: object
```

- SQLite Query Word: **AND (&)**
- The clause == filters for boosters which successfully landed on a [drone ship]
- The & condition determines successful landings with certain payload mass in the desired range
- Three F9 rockets met this condition

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

In [57]:
```
df["Mission_Outcome"].count()

#Total Mission Outcomes = 101
```

Out[57]:
```
101
```

In [66]:
```
#Successful Mission Outcomes
Suc = (df["Mission_Outcome"] == "Success") | (df["Mission_Outcome"] == 'Success (payload status unclear)') |
df[Suc]
df[Suc].count()
```

Out[66]:
```
Date                100
Time (UTC)          100
Booster_Version     100
Launch_Site         100
Payload             100
PAYLOAD_MASS__KG_   100
Orbit               100
Customer            100
Mission_Outcome     100
Landing _Outcome    100
dtype: int64
```

In [65]:
```
df["Mission_Outcome"] == "Failure (in flight)"
Fai = (df["Mission_Outcome"] == "Failure (in flight)")
df[Fai]
df[Fai].count()
```

Out[65]:
```
Date                1
Time (UTC)          1
Booster_Version     1
Launch_Site         1
Payload             1
PAYLOAD_MASS__KG_   1
Orbit               1
Customer            1
Mission_Outcome     1
Landing _Outcome    1
dtype: int64
```

In [64]:
```
df["Mission_Outcome"].unique()
```

Out[64]:
```
array(['Success', 'Failure (in flight)',
       'Success (payload status unclear)', 'Success '], dtype=object)
```

- SQLite Query Words: COUNT, UNIQUE
- 100 successful outcomes, 1 failed outcome
- This query identified specific parameters for successful or failed mission outcomes, including one qualified category (success, but payload status was unclear).

Boosters Carried Maximum Payload

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

In [75]:

```
MaxVal = df["PAYLOAD_MASS__KG_"].max()
df["PAYLOAD_MASS__KG_"] == MaxVal
bol = (df["PAYLOAD_MASS__KG_"] == MaxVal)
df[bol]["Booster_Version"]
```

Out[75]:

```
74      F9 B5 B1048.4
77      F9 B5 B1049.4
79      F9 B5 B1051.3
80      F9 B5 B1056.4
82      F9 B5 B1048.5
83      F9 B5 B1051.4
85      F9 B5 B1049.5
92     F9 B5 B1060.2
93     F9 B5 B1058.3
94      F9 B5 B1051.6
95      F9 B5 B1060.3
99     F9 B5 B1049.7
Name: Booster_Version, dtype: object
```

- SQLite Query Words: == clause, and the MAX() function
- The booster which carries the maximum payload is the F9 B5 version

2017 Launch Records

In [177…

```
df2 = df
df2['Year'] = df['Date'].apply(lambda x: int(x.split('-')[2]))
df2['Month'] = df['Date'].apply(lambda x: int(x.split('-')[1]))

b = (df2['Year']==2017) & (df2['Landing _Outcome']=='Success (ground pad)')
df2[b].loc[:, ['Month', 'Booster_Version', 'Launch_Site', 'Landing _Outcome']]
```

Out[177…

	Month	Booster_Version	Launch_Site	Landing _Outcome
29	2	F9 FT B1031.1	KSC LC-39A	Success (ground pad)
32	5	F9 FT B1032.1	KSC LC-39A	Success (ground pad)
34	6	F9 FT B1035.1	KSC LC-39A	Success (ground pad)
38	8	F9 B4 B1039.1	KSC LC-39A	Success (ground pad)
40	9	F9 B4 B1040.1	KSC LC-39A	Success (ground pad)
44	12	F9 FT B1035.2	CCAFS SLC-40	Success (ground pad)

- Due to utilizing SQLite, which does not display months, I had to separate and reclassify certain columns using Lambda as a window function; months are displayed numerically

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

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

In [201…

```
df['Landing _Outcome'].unique()
```

Out[201…

```
array(['Failure (parachute)', 'No attempt', 'Uncontrolled (ocean)',
       'Controlled (ocean)', 'Failure (drone ship)',
       'Precluded (drone ship)', 'Success (ground pad)',
       'Success (drone ship)', 'Success', 'Failure', 'No attempt '],
      dtype=object)
```

In [213…

```
%sql select "landing _outcome", count(*) from (select * from SPACEXTBL where (date between '04-06-2010' and
```

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

Out[213…

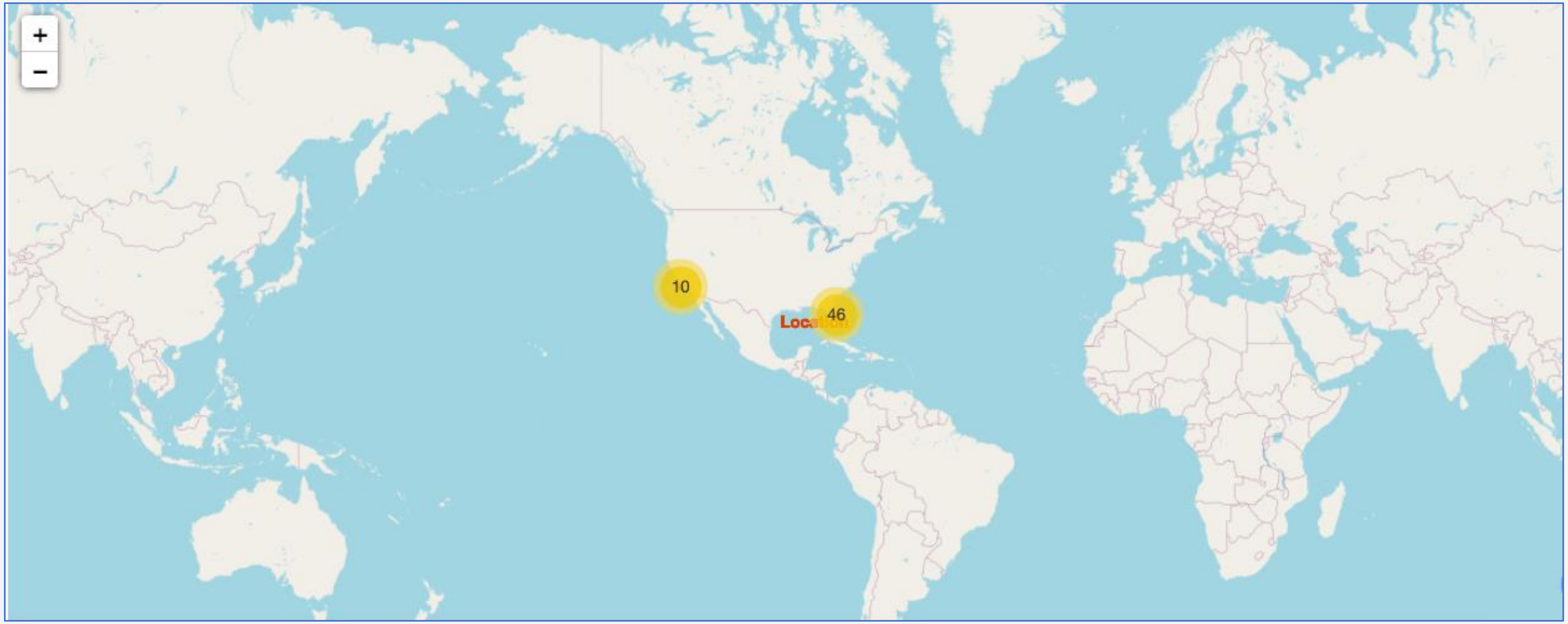
Landing _Outcome	count(*)
Success	20
Success (drone ship)	8
Success (ground pad)	6

- SQ Lite Query Words: COUNT, WHERE, BETWEEN – used to select the correct of landing outcomes in the desired range.
- Additionally, the outcomes were SELECTED from **SpaceXTBL** (failure, success), to show descending order

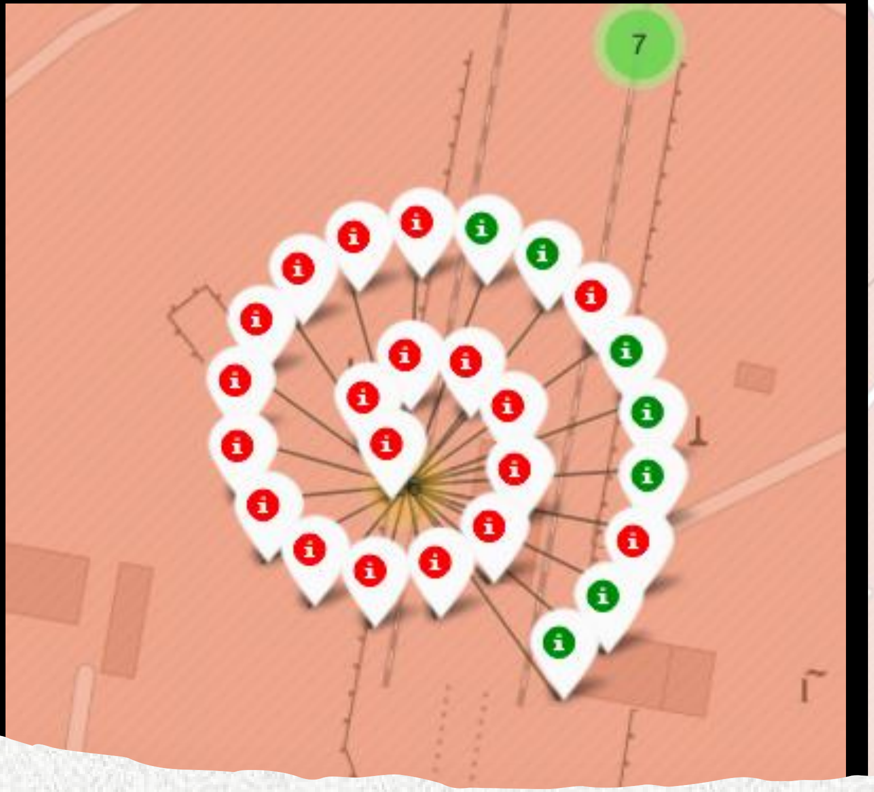
Section 3

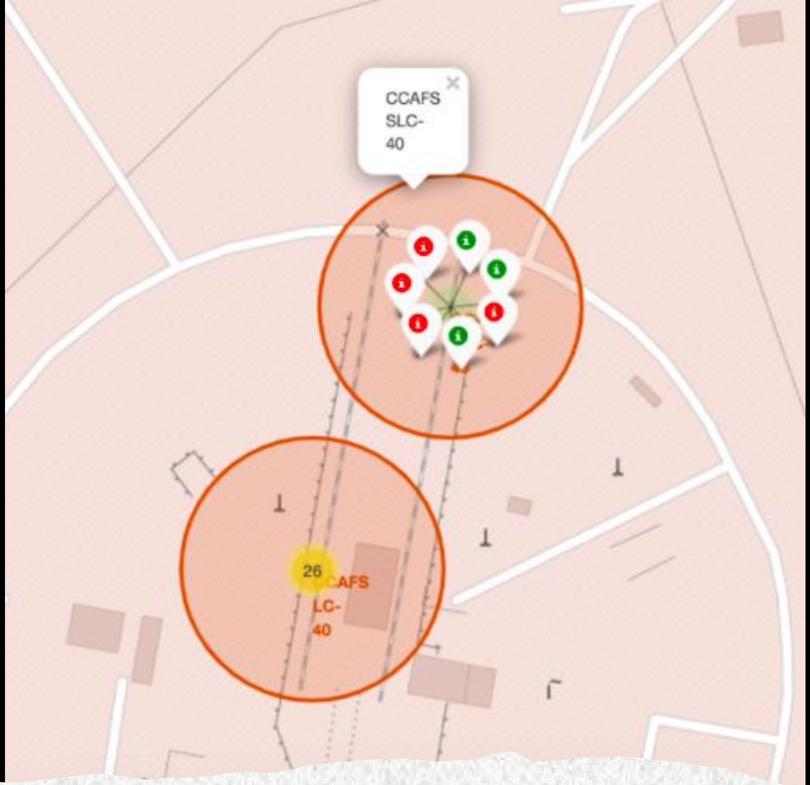
Launch Sites Proximities Analysis

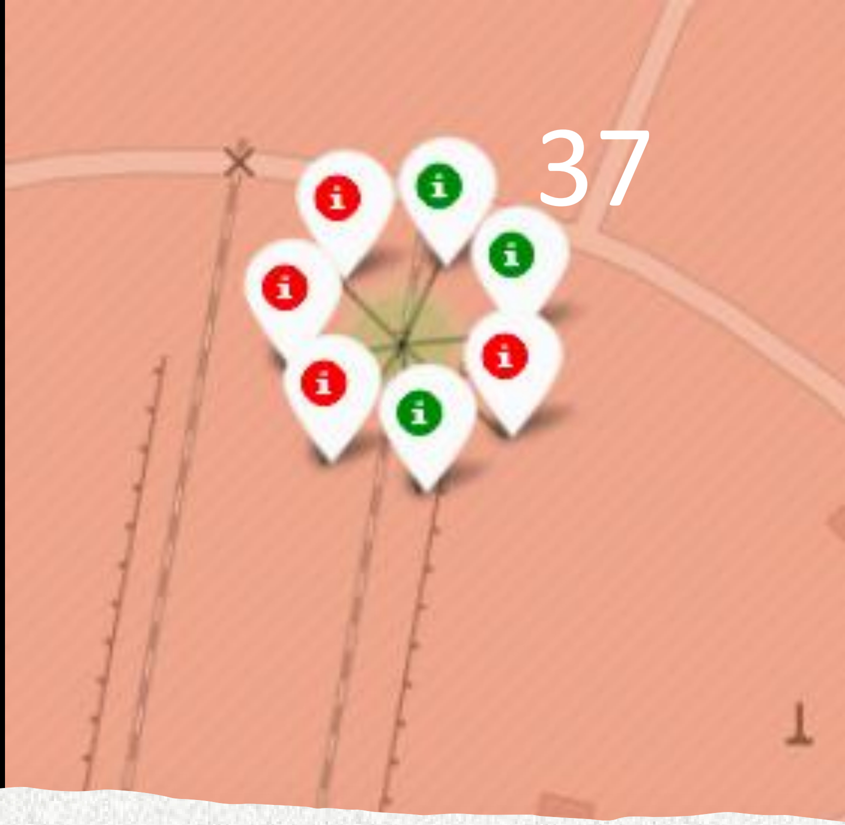
All Launch Sites – Global Map Markers



Launch Sites for SpaceX Falcon 9 rockets are in California, on the **west coast** of the United States (Vandenberg AFB), and in Florida, on the **east coast** of the United States (Cape Canaveral).







Markers Showing Launch Sites with Color Labels

A **GREEN** marker indicates a **successful** launch; whereas a **RED** marker indicates a **failed** launch

Launch Site Distances to Landmarks

Distance from the Florida coastline

Distance to the closest Florida highway

Distance to the closet metropolitan city center

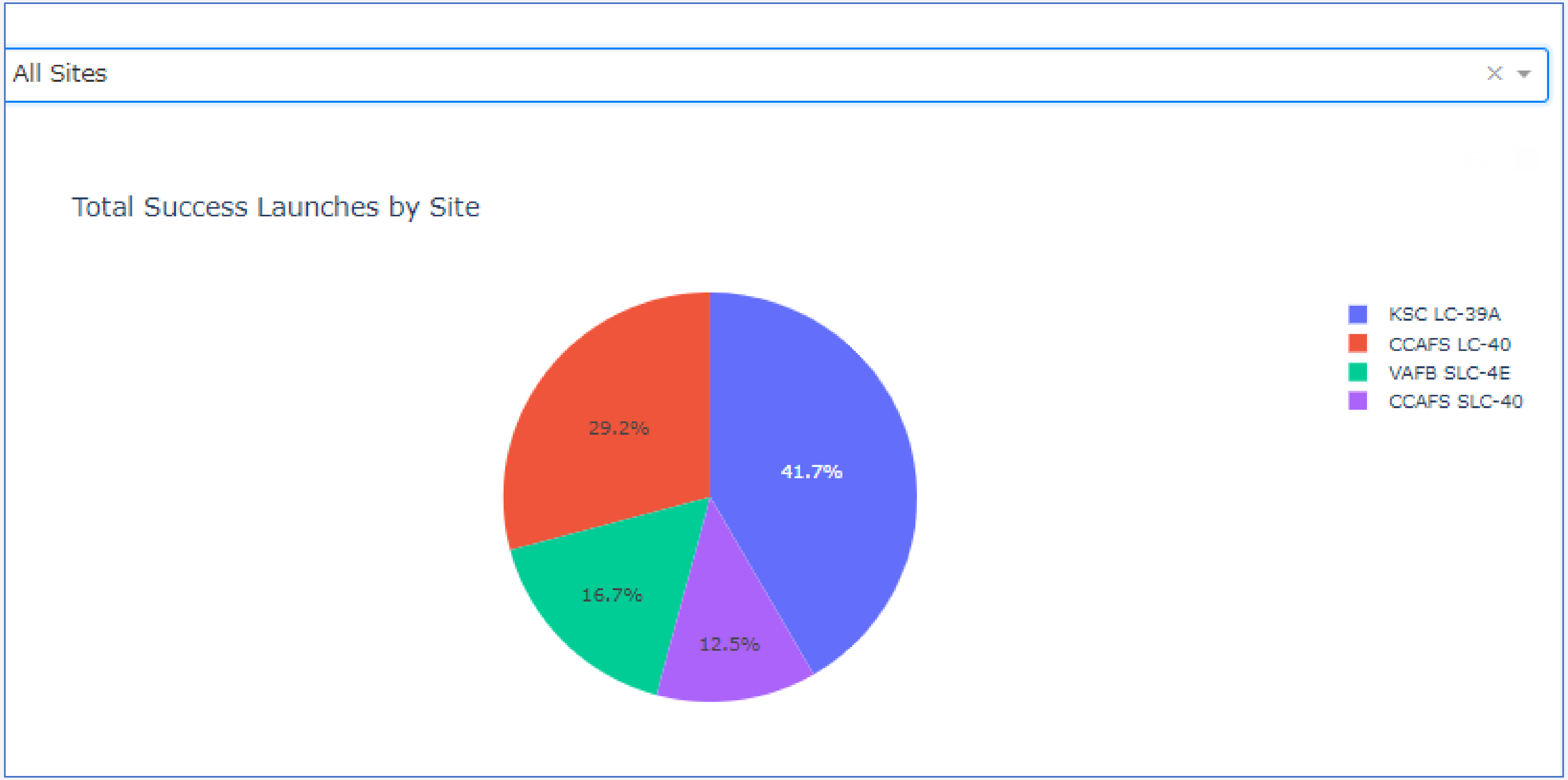
The Interactive Folium maps showed:

- Launch sites are NOT close to highways or metro city centers
- Railway proximities depend on private use (Space Force); typically commercial railways are avoided
- SpaceX certainly launch rockets close to coastlines

Section 4

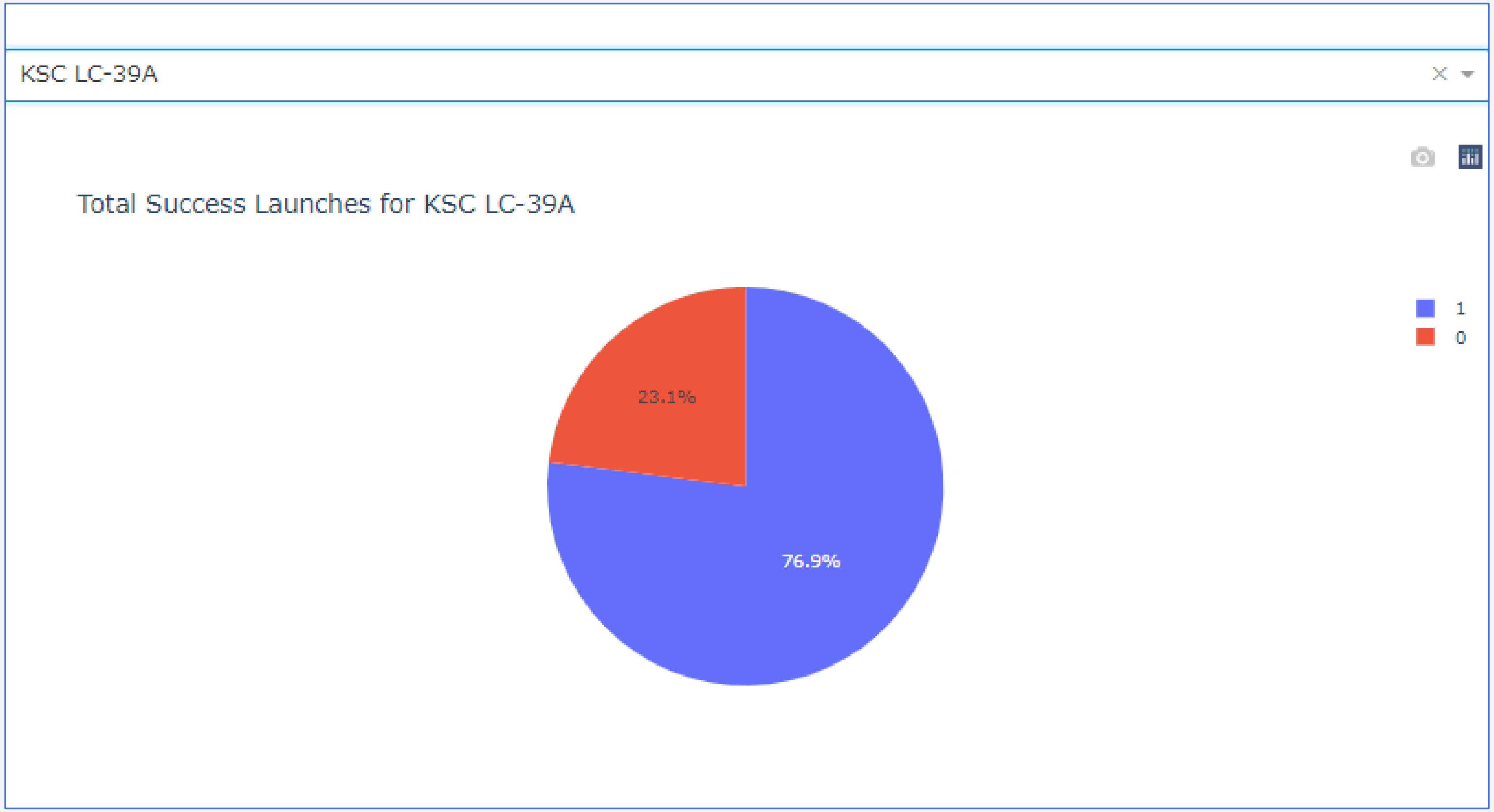
Building a Dashboard with Plotly Dash

Dashboard: Pie Chart of All Successful Launch Sites



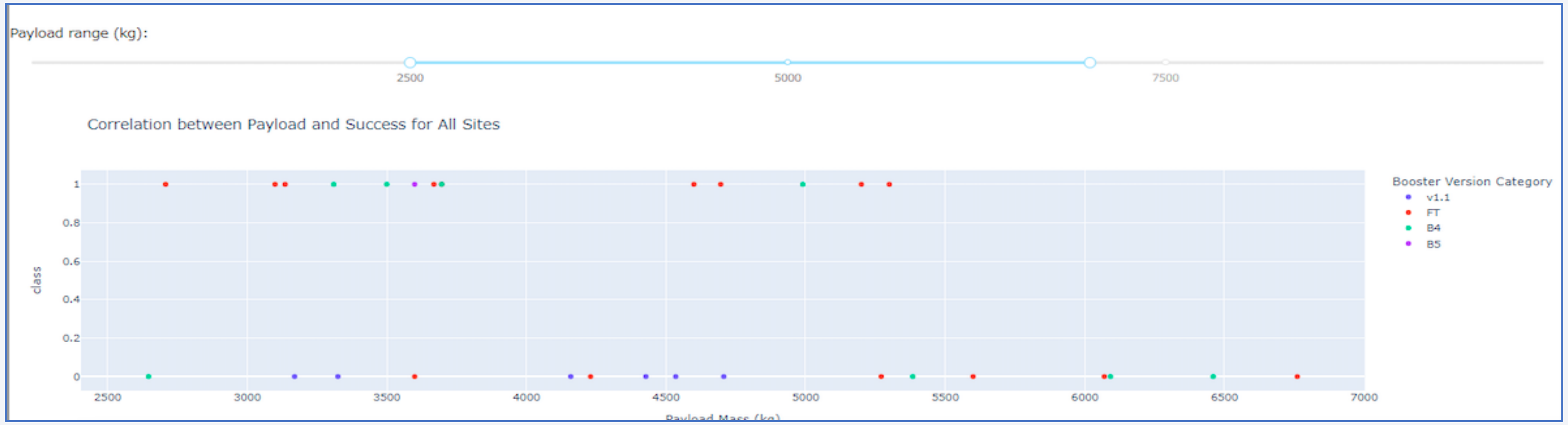
KSC LC-39A was the most successful launch site, with a rate of over 40%.

Dashboard: Pie Chart for the Launch Site with the Highest Launch Success Ratio

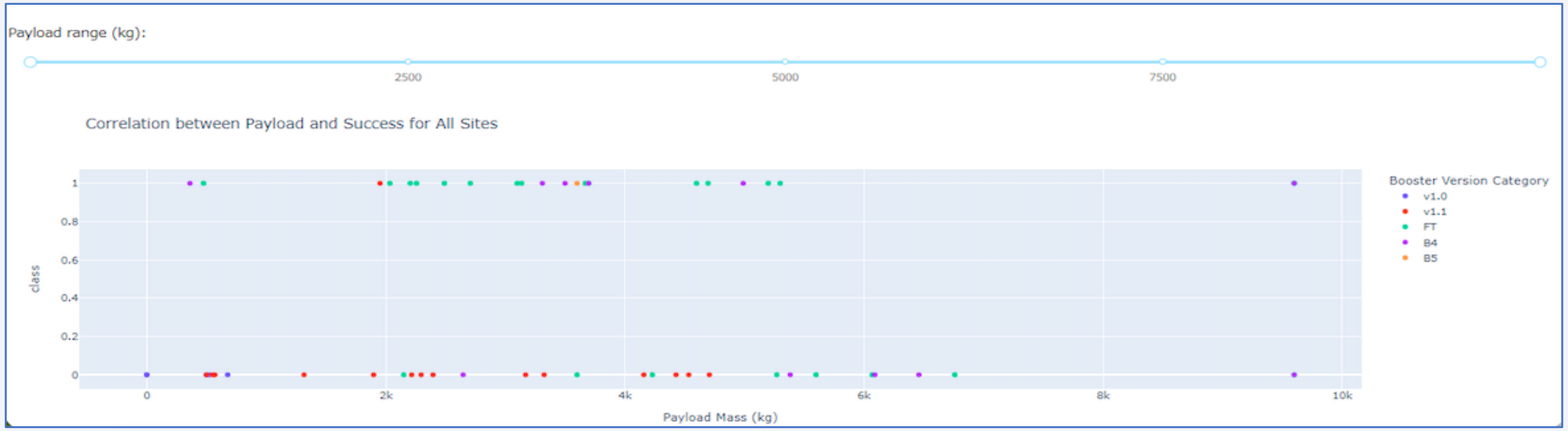


KSC LC-39A achieved a 76.9% success rate for rocket launches. Thus, the failure rate stands at 23.1% at this launch site.

Dashboard: Payload vs. Launch Scatterplot Chart for All Sites, with Slider Ranges



Lighter Payload:
Slider Scale 2500 – 7000kg



Heavy Payload:
Slider Scale 0 – 10,000kg

Success rates for lighter payloads are **higher** than heavyweight payloads.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

KNN, Decision Tree, and Logistic Regression all performed strongly – but numerically, the model with the highest numerical accuracy is the **Decision Tree.**

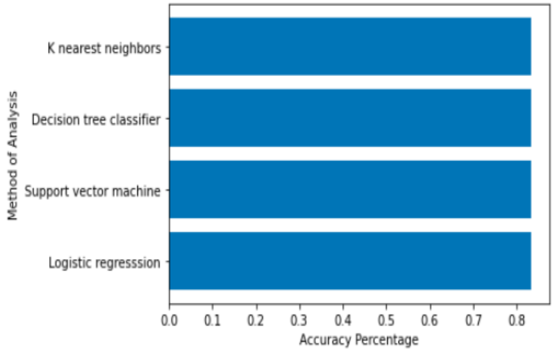
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

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

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

```
In [37]: import numpy as np
         import matplotlib.pyplot as plt

         plt.barh(methods, accuracy)
         plt.xlabel("Accuracy Percentage")
         plt.ylabel("Method of Analysis")
         plt.show()
```

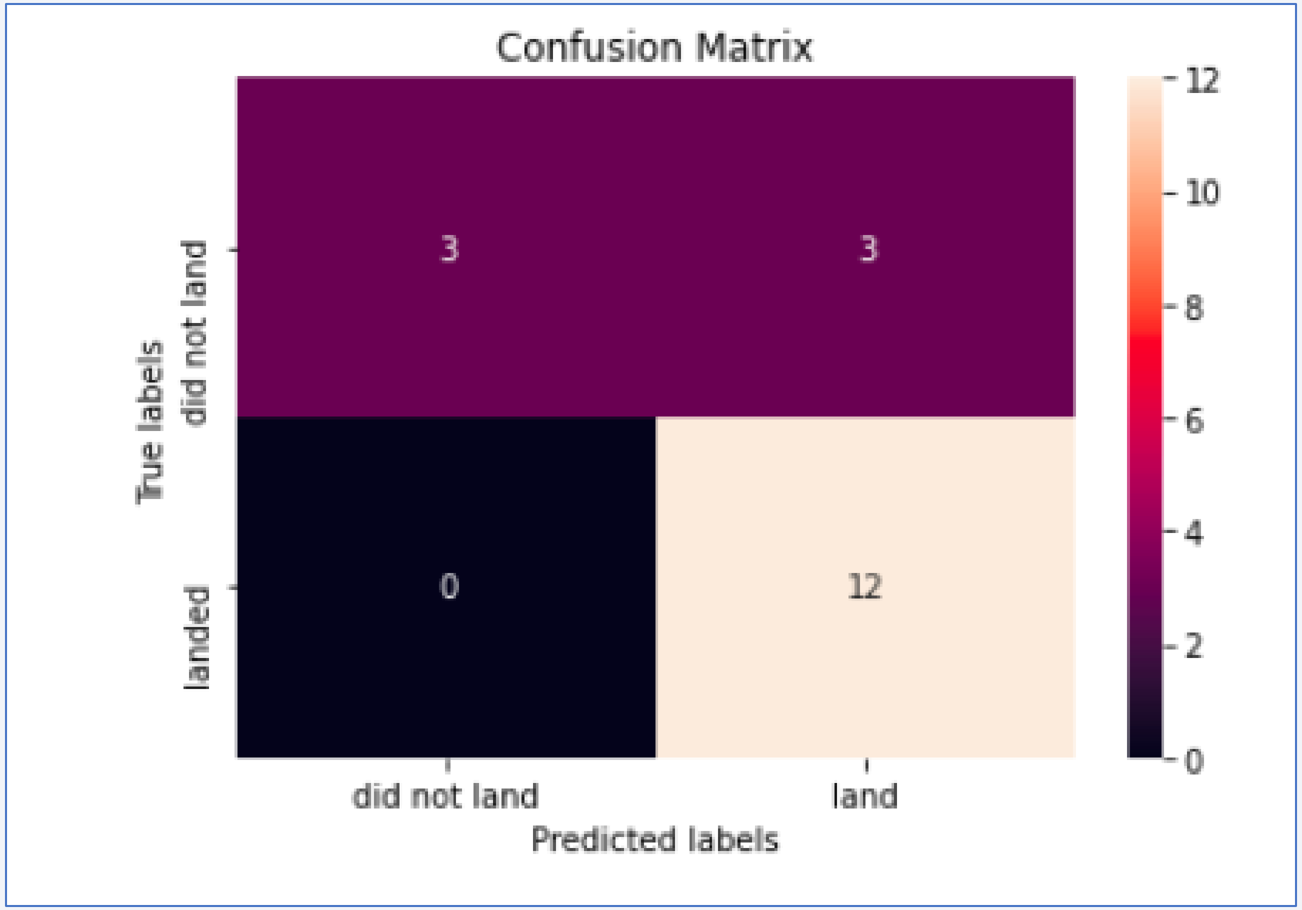


After selecting the best hyperparameters for the Decision Tree classifier using the validation data, the model achieved 83.33% accuracy on the test data.

Confusion Matrix

The Decision Tree machine learning model is able to distinguish between the different classes.

The major issue is the number of **false positives**, where failed launches are marked as successful. This information has a direct impact on deciding whether SpaceX launches can be accurately predicted.



Case Study: SpaceX Rocket Launch
Data Driven Insights & Conclusions

- GEO, HEO, SSO, ES-L1 orbits have the best success rate
- Low weight payloads perform better than heavier ones
- SpaceX launches are proportionally successful; better performance is commensurate with time since the first launch
- KSC LC-39A was the most successful launch site
- The Decision Tree classifier is the best Machine Learning algorithm

Appendix

Photograph credits for section headings are all sourced from commercial-free, royalty-free, open-source wallpaper websites:

- wallpapercave.com
- unsplash.com
- pexels.com


"Data will talk if you're willing to listen."
- Jim Bergeson