IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Kristopher Chesney
10/08/2024

Outline

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

Executive Summary

- Summary of methodologies:
  - Data collection
  - Data wrangling
  - Exploratory data analysis (EDA) with visualizations
  - Exploratory data analysis (EDA) with SQL
  - Interactive maps with Folium
  - Interactive dashboards using Plotly Dash
  - Predictive analysis using classification models
- Summary of all results
  - Exploratory data analysis results
  - Interactive analytics demo in screenshots
  - Predictive analysis results

Introduction

- Project Background and Context:

  Our goal 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, much of the 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.

- Questions We Will Answer:

  What variables determine if the first stage will land?

  Can we use these variables to categorize and predict successful first stage landings?

Section 1

Methodology

Methodology

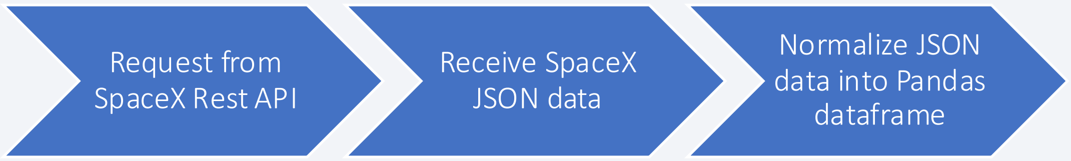
Executive Summary

- Data collection methodology:
  - Rest API from SpaceX
  - Web scraping Wikipedia
- Performed data wrangling
  - One-Hot Encoding for categorical data fields
- Performed exploratory data analysis (EDA) using visualization and SQL
- Performed interactive visual analytics using Folium and Plotly Dash
- Performed predictive analysis using classification models
  - How to build, tune, and evaluate classification models

Data Collection

Rest API from SpaceX

We used the Requests library through Python to connect to the SpaceX Rest API, which returned launch data in a JSON format. This data was normalized and imported into a Pandas dataframe so we could perform further data cleaning and analysis.



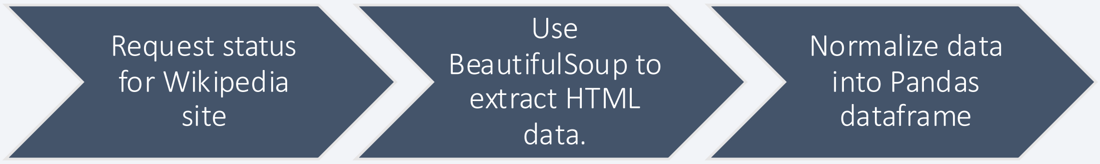
Request from SpaceX Rest API

Receive SpaceX JSON data

Normalize JSON data into Pandas dataframe

Web scraping Wikipedia

We used the Request library to verify Wikipedia site was active, then used BeautifulSoup to extract the particular table from the Wikipedia page. Then this data was normalized into a Pandas dataframe for further filtering and data cleaning.



Request status for Wikipedia site

Use BeautifulSoup to extract HTML data.

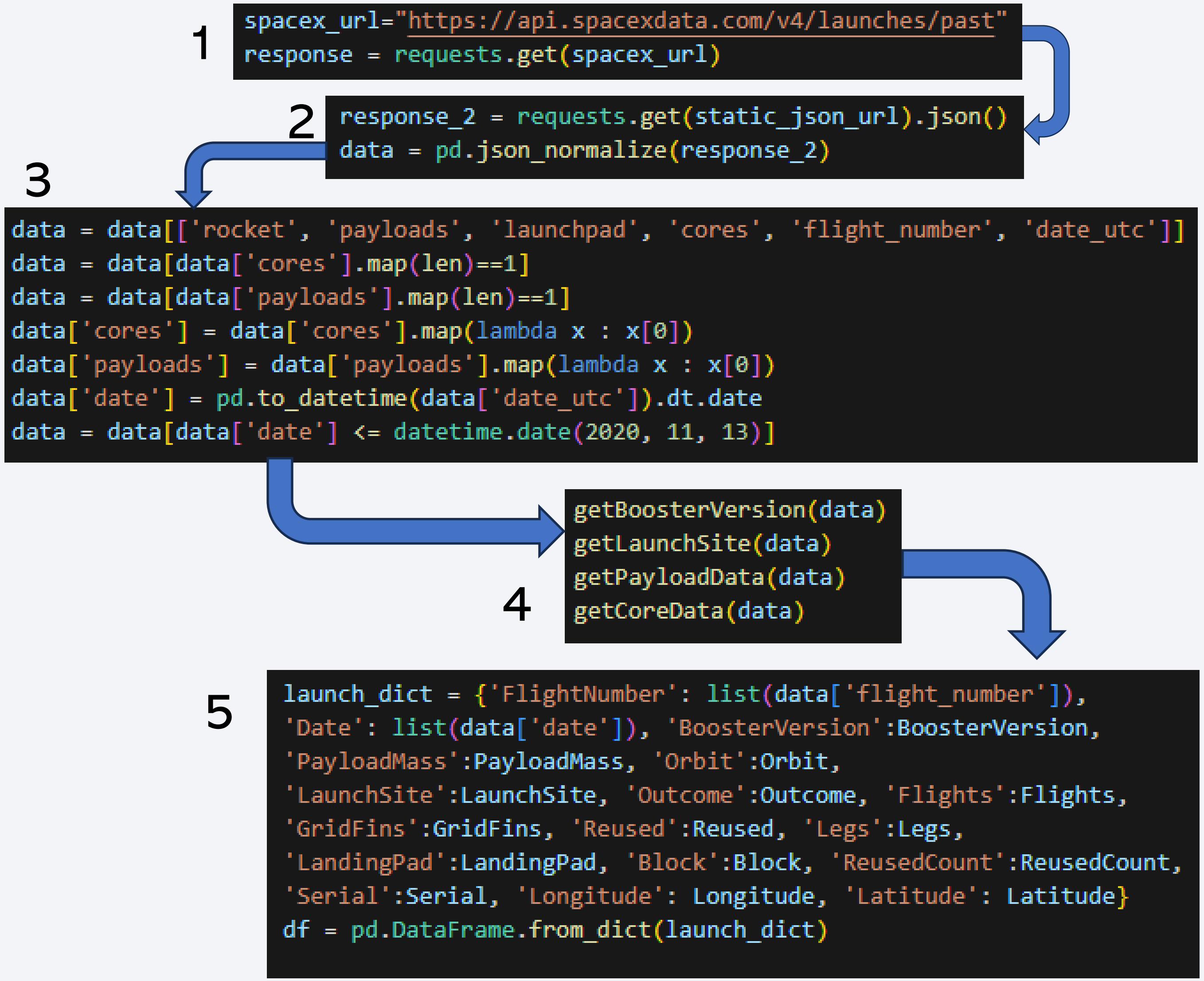
Normalize data into Pandas dataframe

Data Collection – SpaceX API

1. Use Request to call SpaceX Rest API
2. Normalize the JSON Response into a Pandas dataframe
3. Reduce dimensionality by limiting variable columns and standardize column values
4. Use helper function calls to enrich data
5. Assign enriched data into a dictionary, then finally into a new Pandas dataframe

Reference Notebook in Github

URL:https://github.com/huperniKao/ibm_ds_course/blob/main/jupyter-labs-spacex-data-collection-api-v2.ipynb



1

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url)
```

2

```
response_2 = requests.get(static_json_url).json()
data = pd.json_normalize(response_2)
```

3

```
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])
data['date'] = pd.to_datetime(data['date_utc']).dt.date
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

4

```
getBoosterVersion(data)
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

5

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']), 'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass, 'Orbit':Orbit,
'LaunchSite':LaunchSite, 'Outcome':Outcome, 'Flights':Flights,
'GridFins':GridFins, 'Reused':Reused, 'Legs':Legs,
'LandingPad':LandingPad, 'Block':Block, 'ReusedCount':ReusedCount,
'Serial':Serial, 'Longitude': Longitude, 'Latitude': Latitude}
df = pd.DataFrame.from_dict(launch_dict)
```

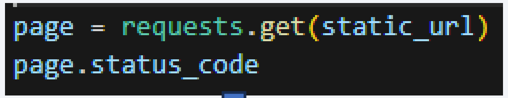
Data Collection - Scraping

1. Use Request library to verify Wikipedia URL
2. BeautifulSoup to extract HTML table
3. Extract column names from table to create a Dictionary
4. Extract data from table to append to Dictionary Keys
5. Convert Dictionary to a Pandas dataframe

[Reference Notebook in Github](https://github.com/huperniKao/ibm_ds_course/blob/main/jupyter-labs-webscraping.ipynb)

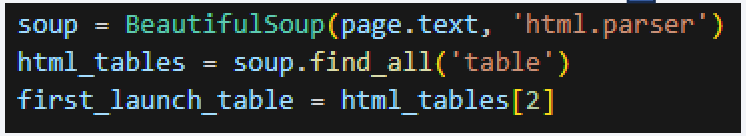
URL:https://github.com/huperniKao/ibm_ds_course/blob/main/jupyter-labs-webscraping.ipynb

1



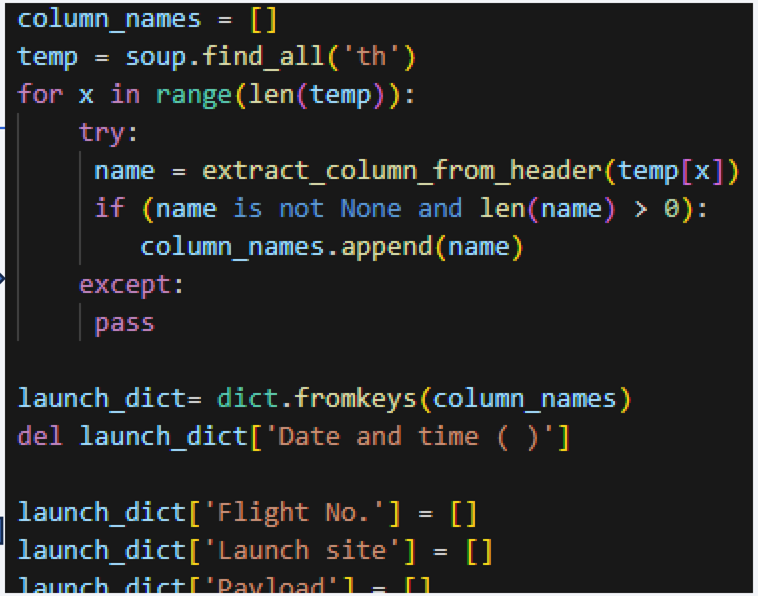
```python
page = requests.get(static_url)
page.status_code
```

2



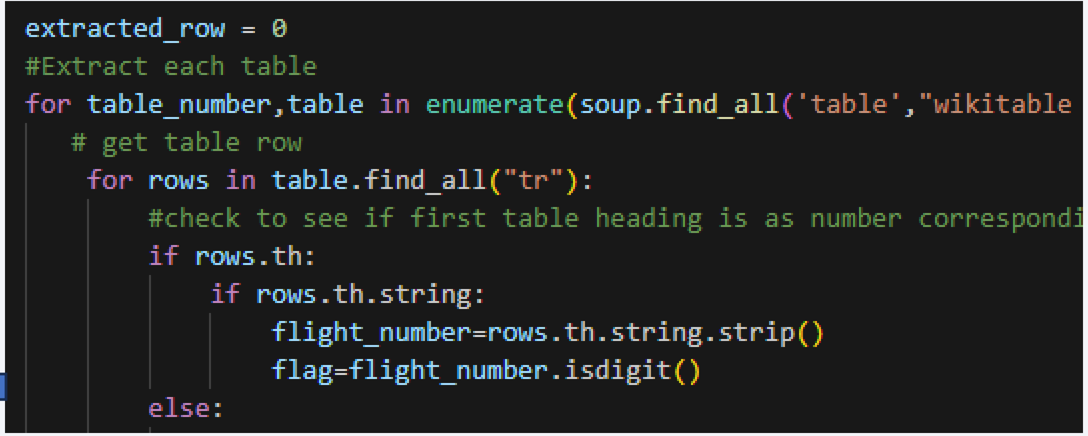
```python
soup = BeautifulSoup(page.text, 'html.parser')
html_tables = soup.find_all('table')
first_launch_table = html_tables[2]
```

3



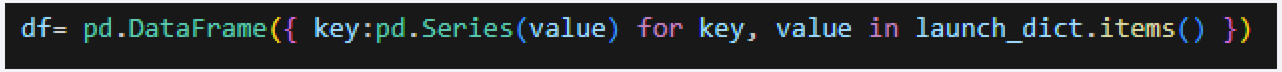
```python
column_names = []
temp = soup.find_all('th')
for x in range(len(temp)):
    try:
        name = extract_column_from_header(temp[x])
        if (name is not None and len(name) > 0):
            column_names.append(name)
    except:
        pass

launch_dict= dict.fromkeys(column_names)
del launch_dict['Date and time ( )']

launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
```

4



```python
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable
    # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number correspondi
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
```

5



```python
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

Data Wrangling

- We began to clean the data by reviewing the features and verifying which were numerical or categorical.
- We further explored the data to and researched SpaceX and rocket launches to determine which features would be most useful.
- This led to the following key features:
  - Number of unique launch sites
  - Number of launches at each site
  - Number of unique orbits and launches to those orbits
  - Different types of mission outcomes
- From the different mission outcomes, we seprated these into failures or successes as a new variable that would help with further predictive analytics.

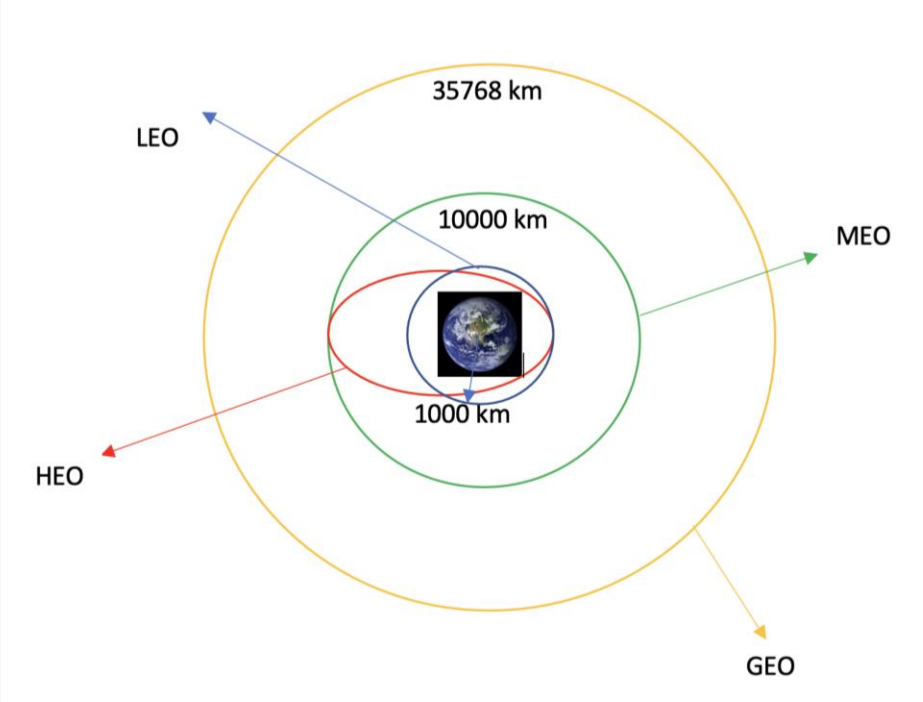


Image showing a sample of SpaceX Orbits

Reference Notebook in Github

URL:https://github.com/huperniKao/ibm_ds_course/blob/main/labs-jupyter-spacex-Data%20wrangling-v2.ipynb

EDA with Data Visualization

Scatter Plots

A scatter plot uses dots to represent values for two different numeric variables. The position of each dot on the horizontal and vertical axis indicates values for an individual data point. Scatter plots are used to observe relationships between variables.

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

Bar Chart

A bar chart shows the distribution of data using a discrete data set. It plots numeric values for categorical features as bars, where the length of each bar corresponds to the value of the category it represents.

- Success Rate per Orbit Type

Line Graph

A line graph shows how data points change over time by connecting them with lines.

- Yearly Success Rate

Reference Notebook in Github

URL:https://github.com/huperniKao/ibm_ds_course/blob
/main/jupyter-labs-eda-dataviz-v2.ipynb

EDA with SQL

Summary of SQL Queries

- List each unique Launch Site
- Show total payload mass carried by boosters launched by NASA (CRS)
- Show average payload mass carried by booster version F9 v1.1
- Show the date when the first succesful landing outcome in ground pad was acheived
- Show the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- Show the total number of successful and failure mission outcomes
- List the booster_versions which have carried the maximum payload mass
- List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015
- 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

Reference Notebook in Github

URL:https://github.com/huperniKao/ibm_ds_course/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- We developed an interactive Launch Site location map using Folium within Python.
- The Launch Sites are indicated by a Circle Marker.
- We also implemented a Cluster Market to indicate the number of Launches from each Launch Site.
- Each Launch is indicated by an Icon once the Cluster is clicked. The Icons are color coded to indicate failed or successful launch.
- We used a Haversine's formula to calculate distance from the Launch Site to other landmarks (highway, coast, city) to determine potential patterns. We used Line Markers to indicate these distances.
- Through this exploration we found that Launch Sites appear to always be built near a coast and further from cities.

Reference Notebook in Github

URL:https://github.com/huperniKao/ibm_ds_course/blob /main/lab-jupyter-launch-site-location-v2.ipynb

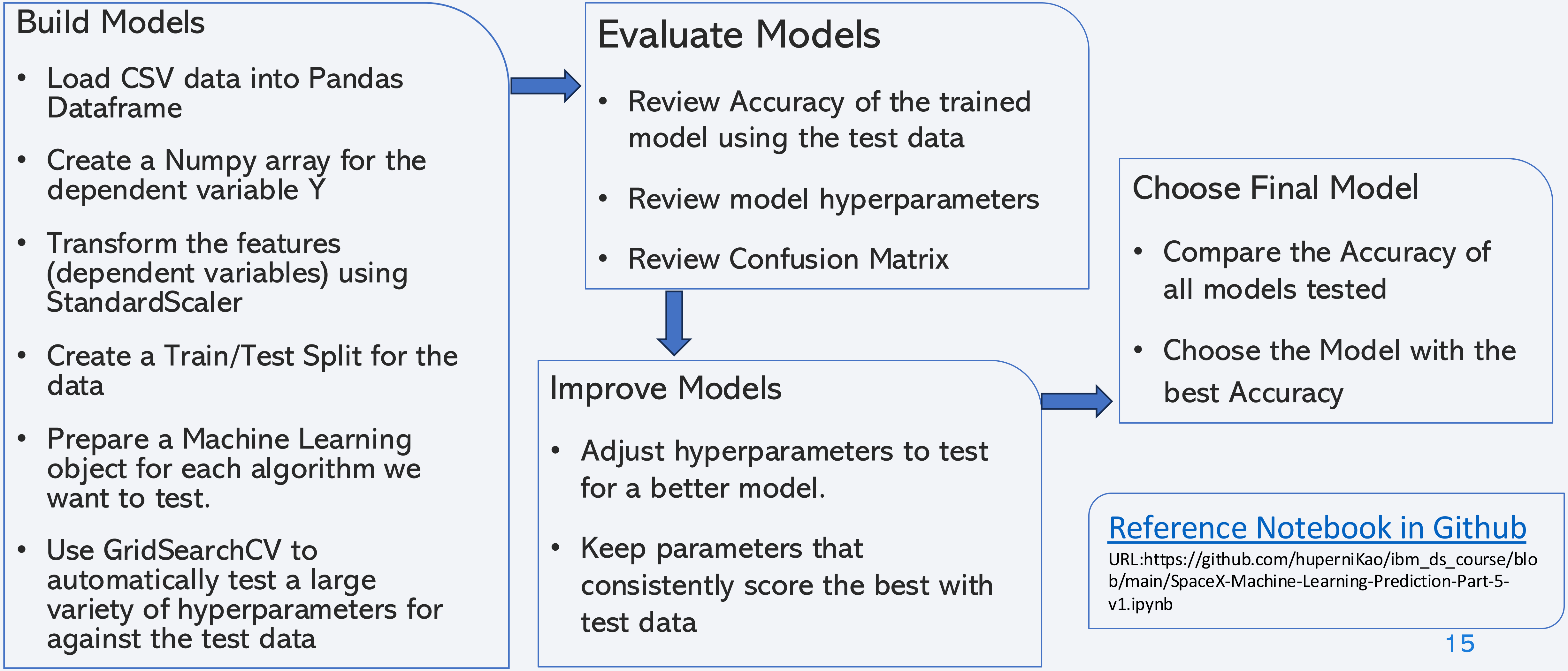
Build a Dashboard with Plotly Dash

- We created a SpaceX Launch Site Records Dashboard.
- There are two major visualizations on this dashboard:
  - Pie Chart showing Total Launches partitioned by Launch Site.
  - Scatter Plot of Payload Mass vs Launch Outcome with Hue based on different Boosters
- All visualizations can be filtered by two other interactions:
  - Drop-down list of Launch Sites
  - Range slider displaying the Payload range

Reference Notebook in Github

URL:https://github.com/huperniKao/ibm_ds_course/blob/main/launch_app.py

Predictive Analysis (Classification)



Build Models

- Load CSV data into Pandas Dataframe
- Create a Numpy array for the dependent variable Y
- Transform the features (dependent variables) using StandardScaler
- Create a Train/Test Split for the data
- Prepare a Machine Learning object for each algorithm we want to test.
- Use GridSearchCV to automatically test a large variety of hyperparameters for against the test data

Evaluate Models

- Review Accuracy of the trained model using the test data
- Review model hyperparameters
- Review Confusion Matrix

Improve Models

- Adjust hyperparameters to test for a better model.
- Keep parameters that consistently score the best with test data

Choose Final Model

- Compare the Accuracy of all models tested
- Choose the Model with the best Accuracy

Reference Notebook in Github

URL:https://github.com/huperniKao/ibm_ds_course/blob/main/SpaceX-Machine-Learning-Prediction-Part-5-v1.ipynb

Results

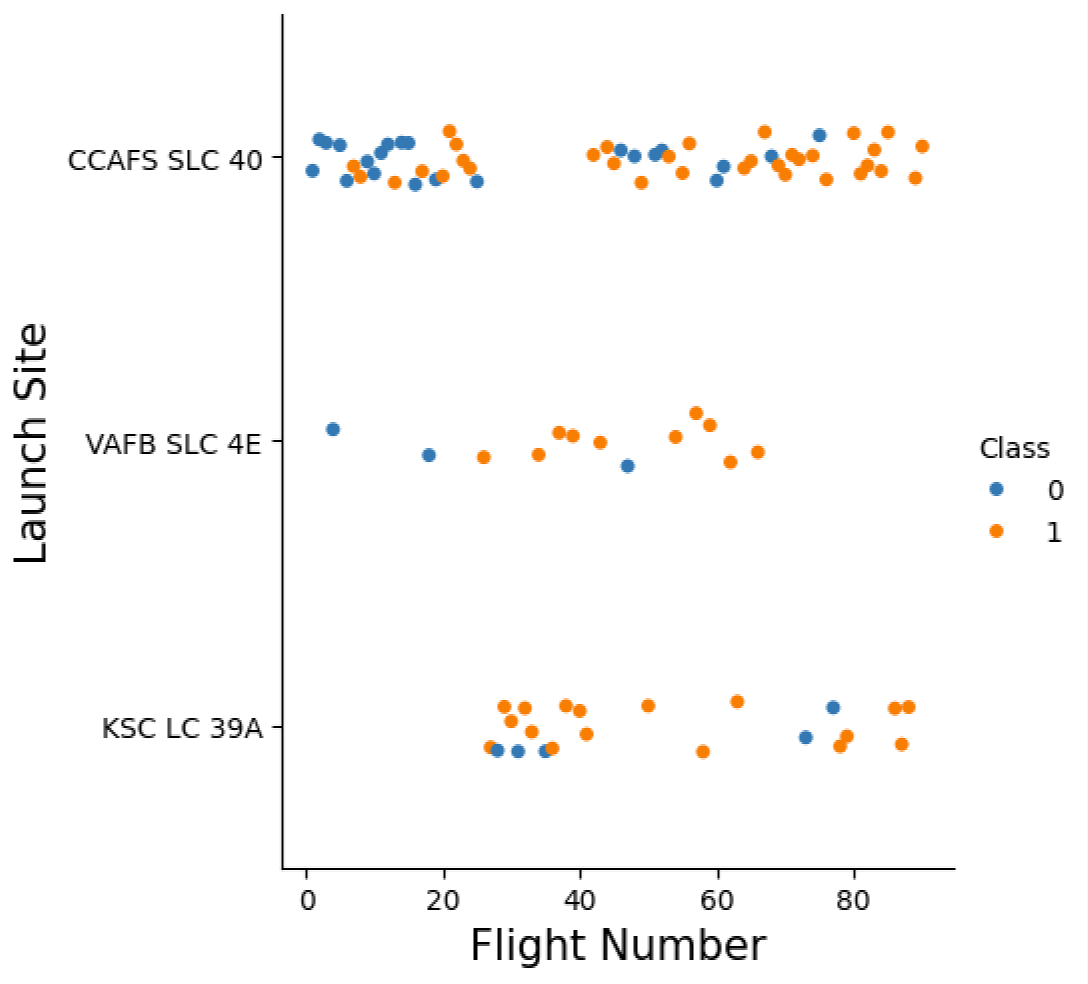
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

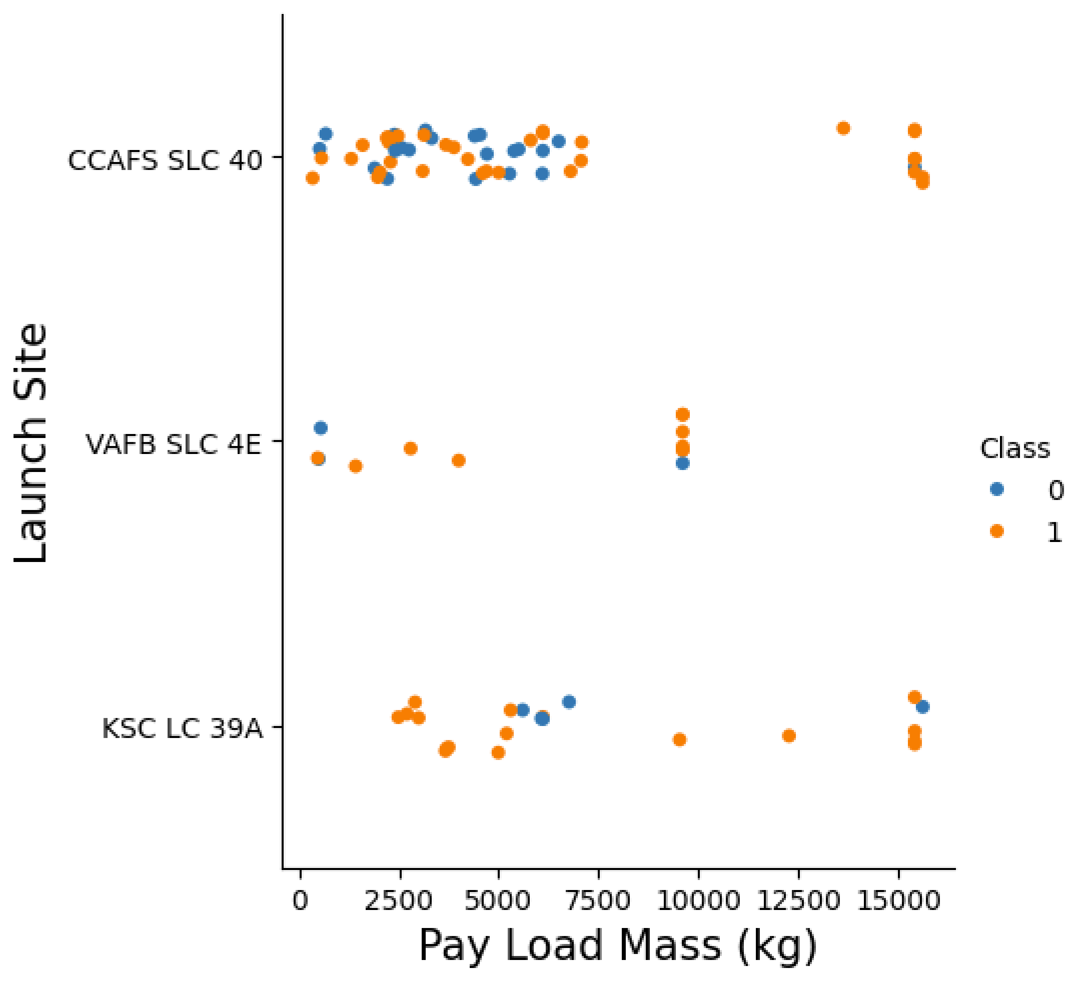
This Scatter Plot demonstrates that fewer launches that have occurred at a launch site, the more susceptible to failure that launch is.



Payload vs. Launch Site

Launch site VAFB SLC 4e has not launched any payloads heavier than 10,000 kg.

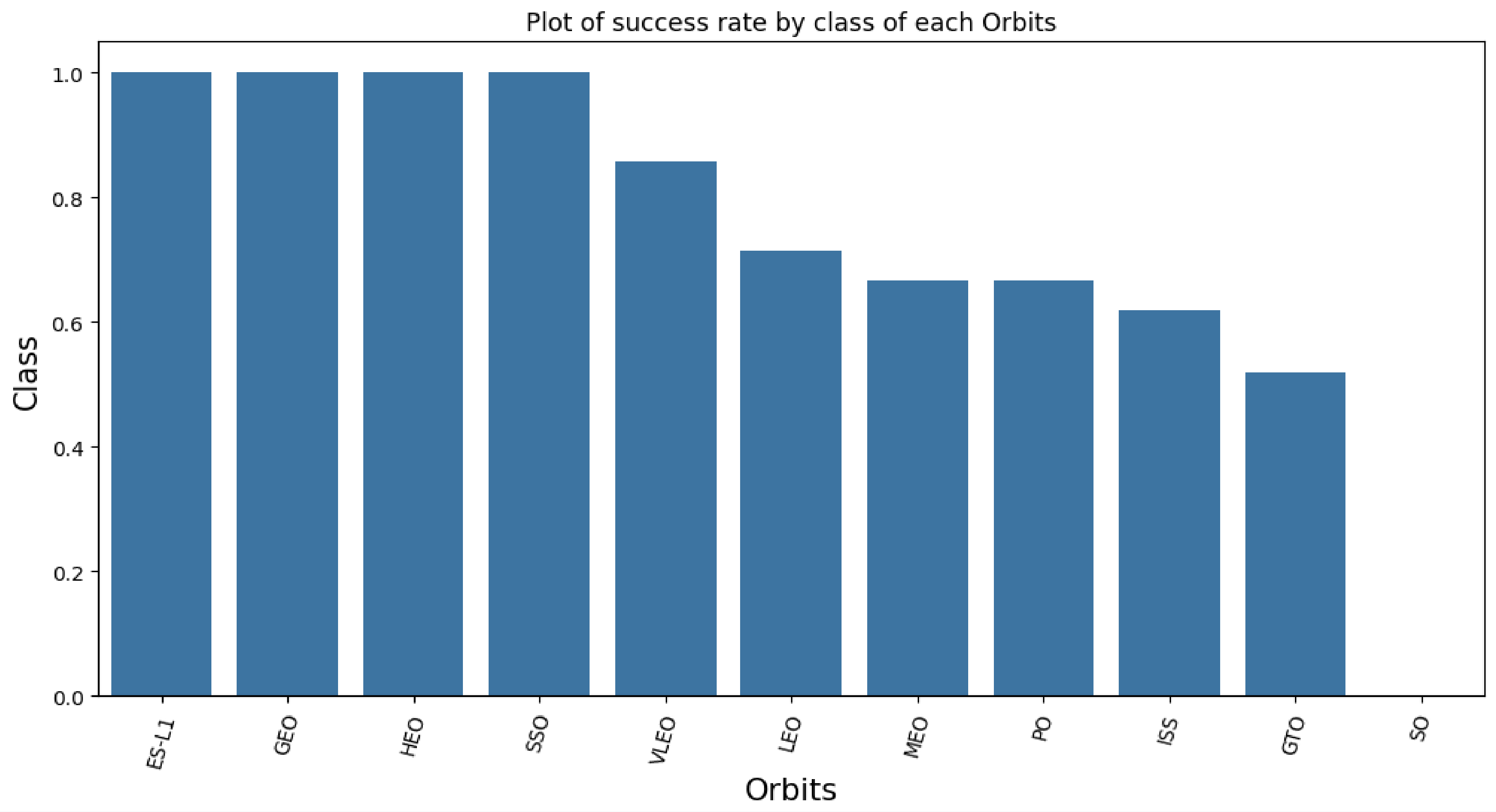
Launch sites appear to have more successful launches with the heaviest payloads than launches with mid or light weight payloads



Success Rate vs. Orbit Type

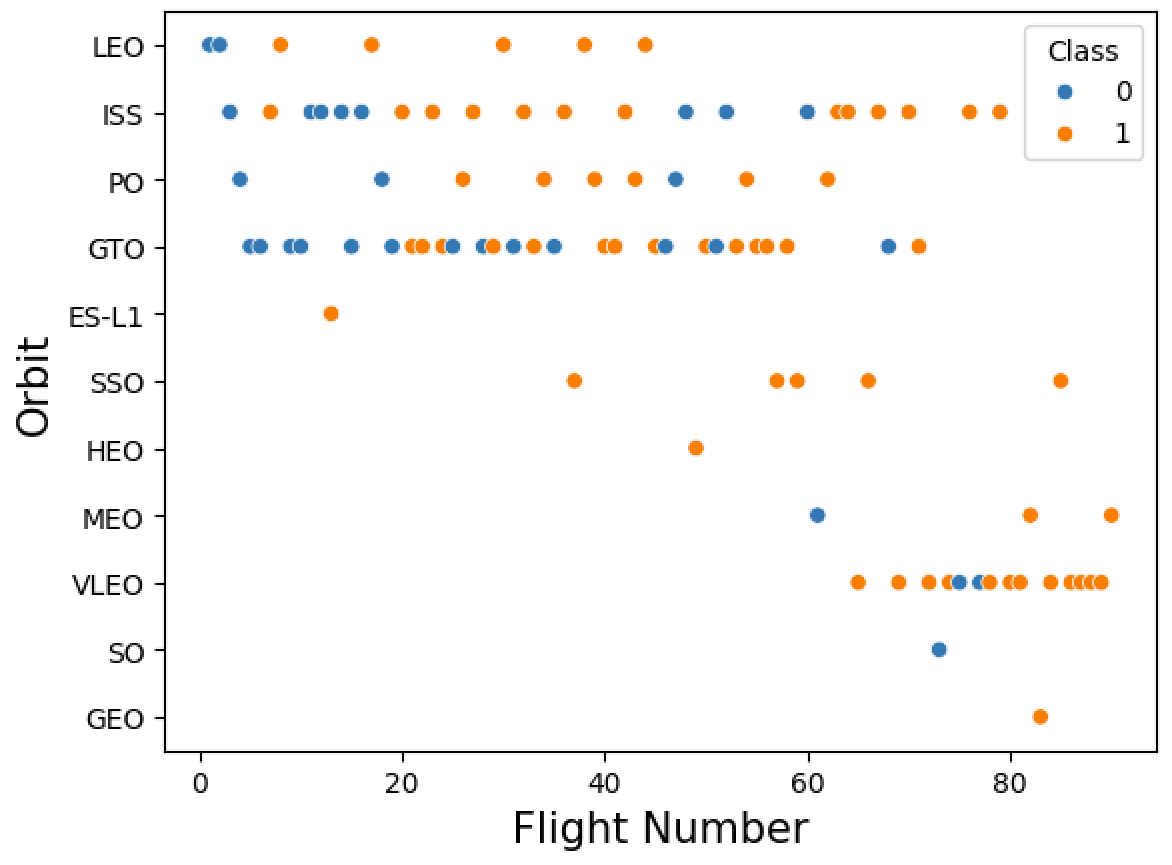
In order, the following Orbits have experienced the best Success Rate:

- ES-L1
- GEO
- HEO
- SSO
- VLEO



Flight Number vs. Orbit Type

In general, it appears that the more flights at each launch site, then the higher likelihood for success. As seen with GTO though, this will not always be guaranteed.

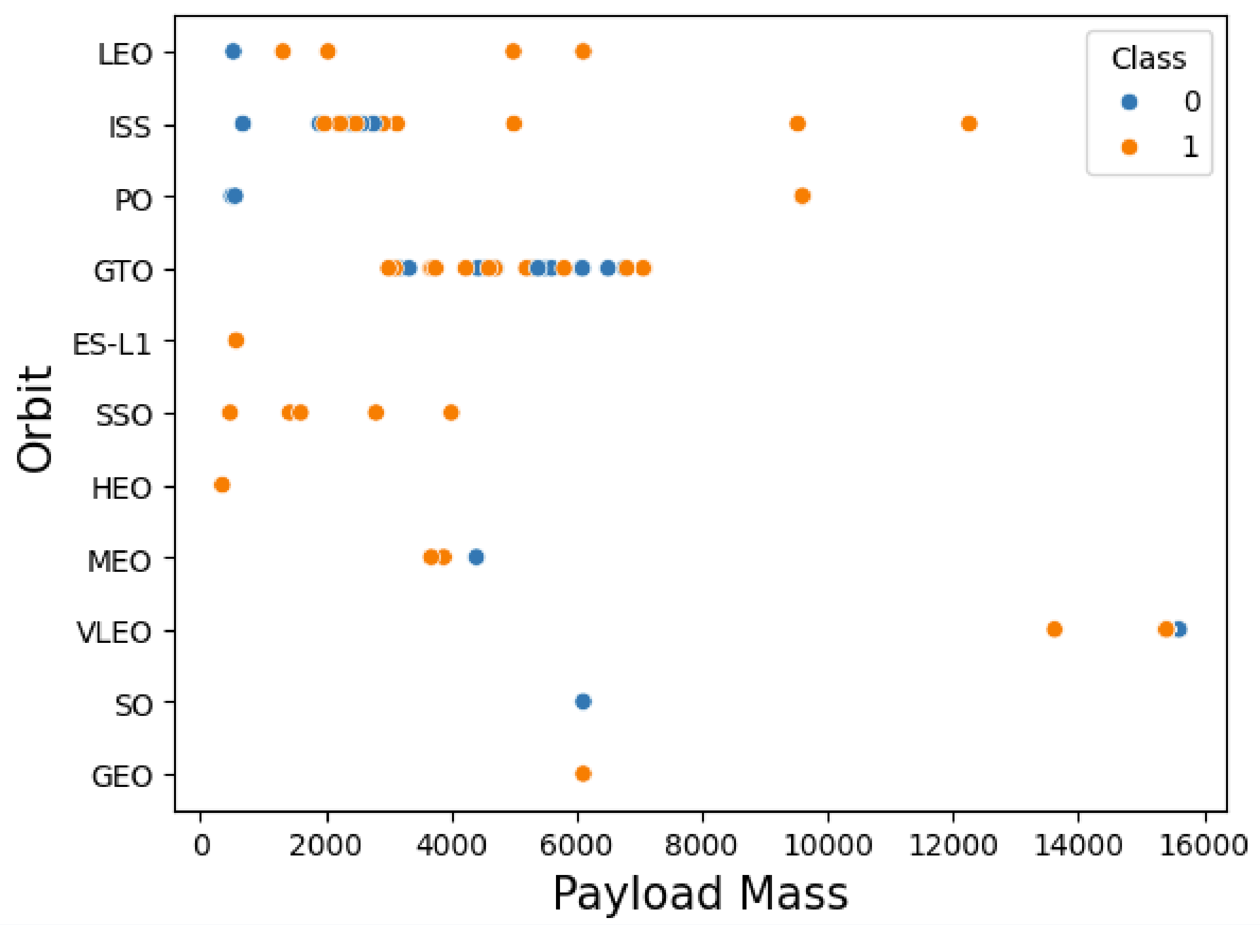


Payload vs. Orbit Type

Heavier payloads appear to have a higher success rate at LEO and ISS.

Most other launch sites do not have enough data to draw any meaningful conclusions.

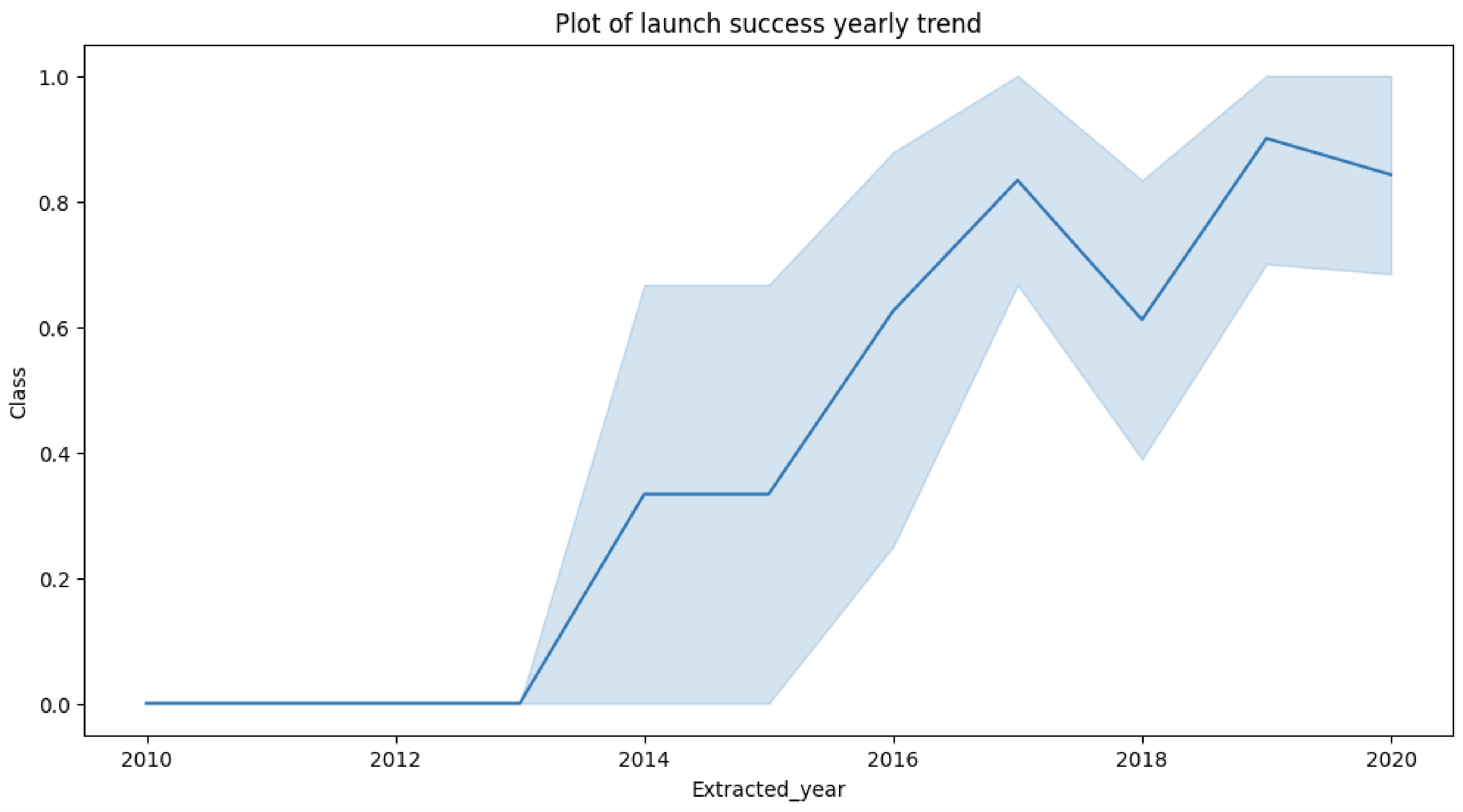
GTO has performed many launches, but payload mass does not appear to be a good indicator for success rate in this case.



Launch Success Yearly Trend

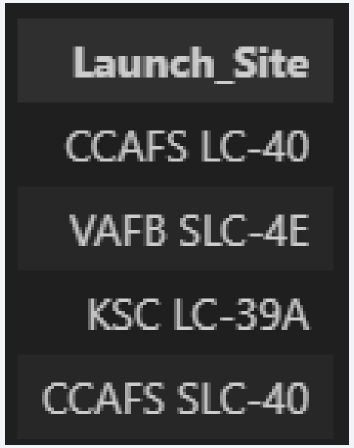
The years 2013 to 2016 shows an increasing success rate with a stable rate in 2017.

Following a drop in 2017, the rate increased again in 2018.



All Launch Site Names

```
%sql select DISTINCT "Launch_Site" from SPACEXTABLE
```

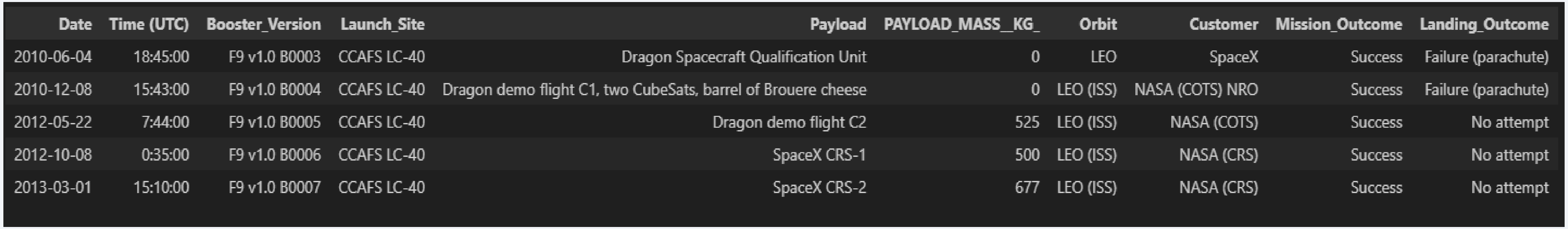


Launch_Site
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40

Using DISTINCT in this SQL Query allowed us to build a summarized list of Launch Sites

Launch Site Names Begin with 'CCA'

```
%sql select * from SPACEXTABLE where "Launch_Site" LIKE 'CCA%' limit 5
```



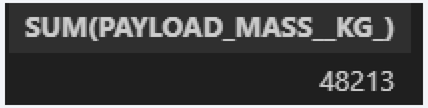
Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Using LIKE we are able to find Launch Sites with CCA in the name.

Additionally, with limit 5, we only listed the first 5 records found.

Total Payload Mass

```
%sql select SUM(PAYLOAD_MASS__KG_)
from SPACEXTABLE
where "Customer" like 'NASA (CRS)%'
```



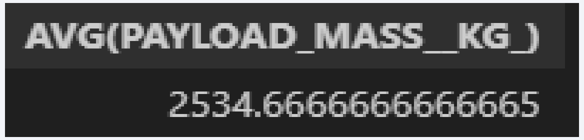
```
SUM(PAYLOAD_MASS__KG_)
                 48213
```

Using "like" we can narrow down the customer we are looking for.

SUM allows us to show the total of all Payload_Mass_KG_ values.

Average Payload Mass by F9 v1.1

```
%sql select AVG(PAYLOAD_MASS__KG_)
from SPACEXTABLE
where "Booster_Version" LIKE 'F9 v1.1%'
```

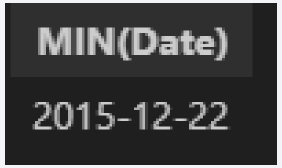


```
AVG(PAYLOAD_MASS__KG_)
    2534.6666666666665
```

Using the AVG function we are able to find the average payload mass. We refine this further by filtering for only payload mass from booster version F9 v1.1.

First Successful Ground Landing Date

```
%sql select MIN("Date")
from SPACEXTABLE
where "Landing_Outcome"='Success (ground pad)'
```

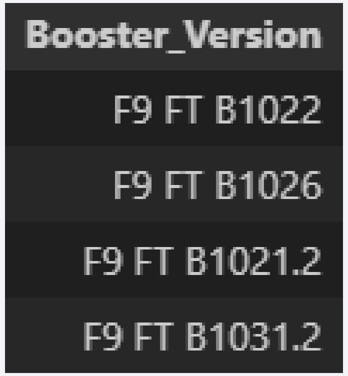


MIN(Date)
2015-12-22

Using MIN we are able to see the first landing date. We refine this further to landings with an outcome of "Success (ground pad)"

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql SELECT "Booster_Version"
FROM SPACEXTABLE
WHERE "Landing_Outcome" = 'Success (drone ship)'
AND "PAYLOAD_MASS__KG_" > 4000
AND "PAYLOAD_MASS__KG_" < 6000
```

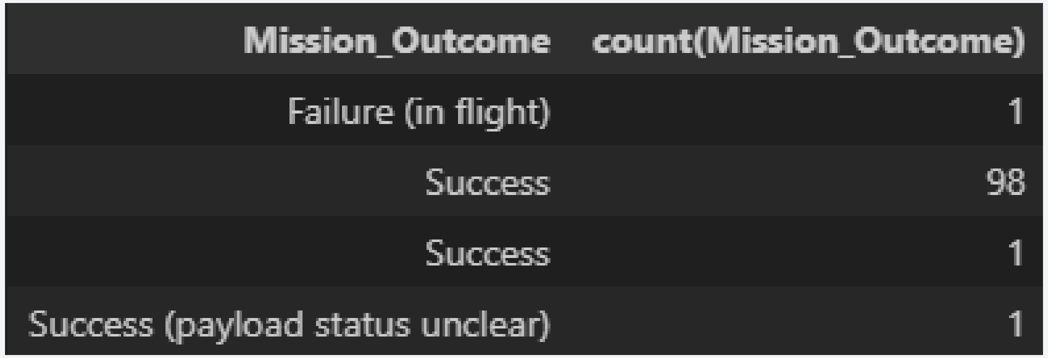


Booster_Version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

We list the Booster_Version of each launch with a payload mass greater than 4000 and less than 6000, where the landing outcome with 'Success (drone ship)'.

Total Number of Successful and Failure Mission Outcomes

```
%sql select "Mission_Outcome",count("Mission_Outcome")
from SPACEXTABLE GROUP BY "Mission_Outcome"
```



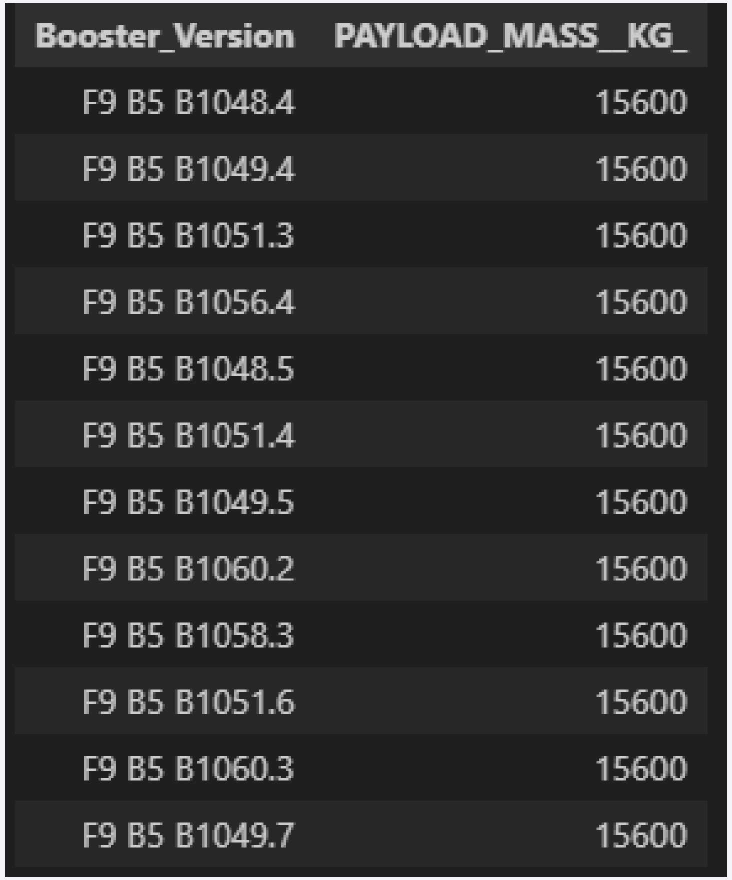
Mission_Outcome	count(Mission_Outcome)
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Using 'count' we can find the total number of launches, which we display by each "Mission Outcome" by using GROUP BY.

Boosters Carried Maximum Payload

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

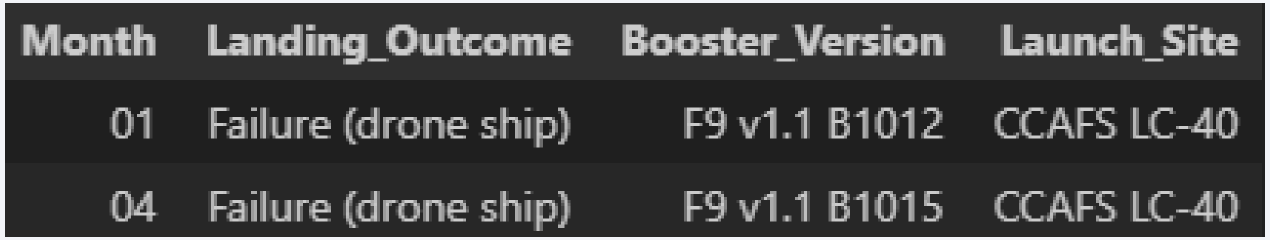
A Subquery was used to determine the MAX of the payload mass. This was passed to the original query to display all booster versions that have carried that size mass.



Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

2015 Launch Records

```
%sql SELECT substr(Date, 6, 2) as Month, "Landing_Outcome", "Booster_Version", "Launch_Site"
FROM SPACEXTABLE WHERE substr(Date, 1, 4) = '2015'
AND "Landing_Outcome" = 'Failure (drone ship)'
```



Month	Landing_Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

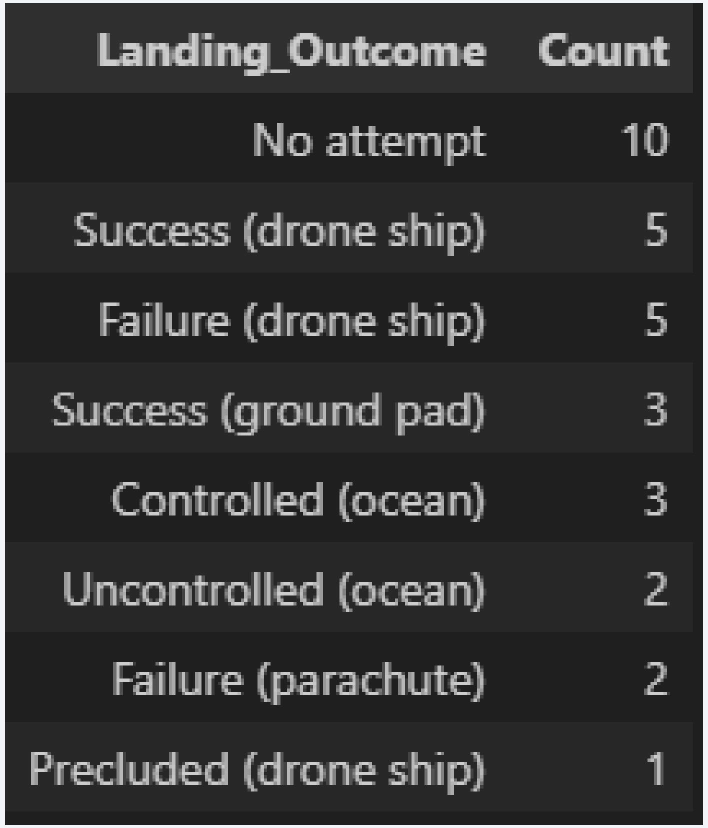
This table shows the landing outcome, booster version, and launch site for launches with the outocme of "Failure (drone ship)" in the year 2015.

Substring was used to extract the Month and Year from the Date field to display this in the table.

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

```
%sql SELECT "Landing_Outcome", COUNT(*) as Count
FROM SPACEXTABLE
WHERE Date BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY "Landing_Outcome"
ORDER BY Count DESC
```

This table lists all the launches between 2010-06-04 and 2017-03-20, grouping them by their landing outcome and listing them in descending order using DESC.

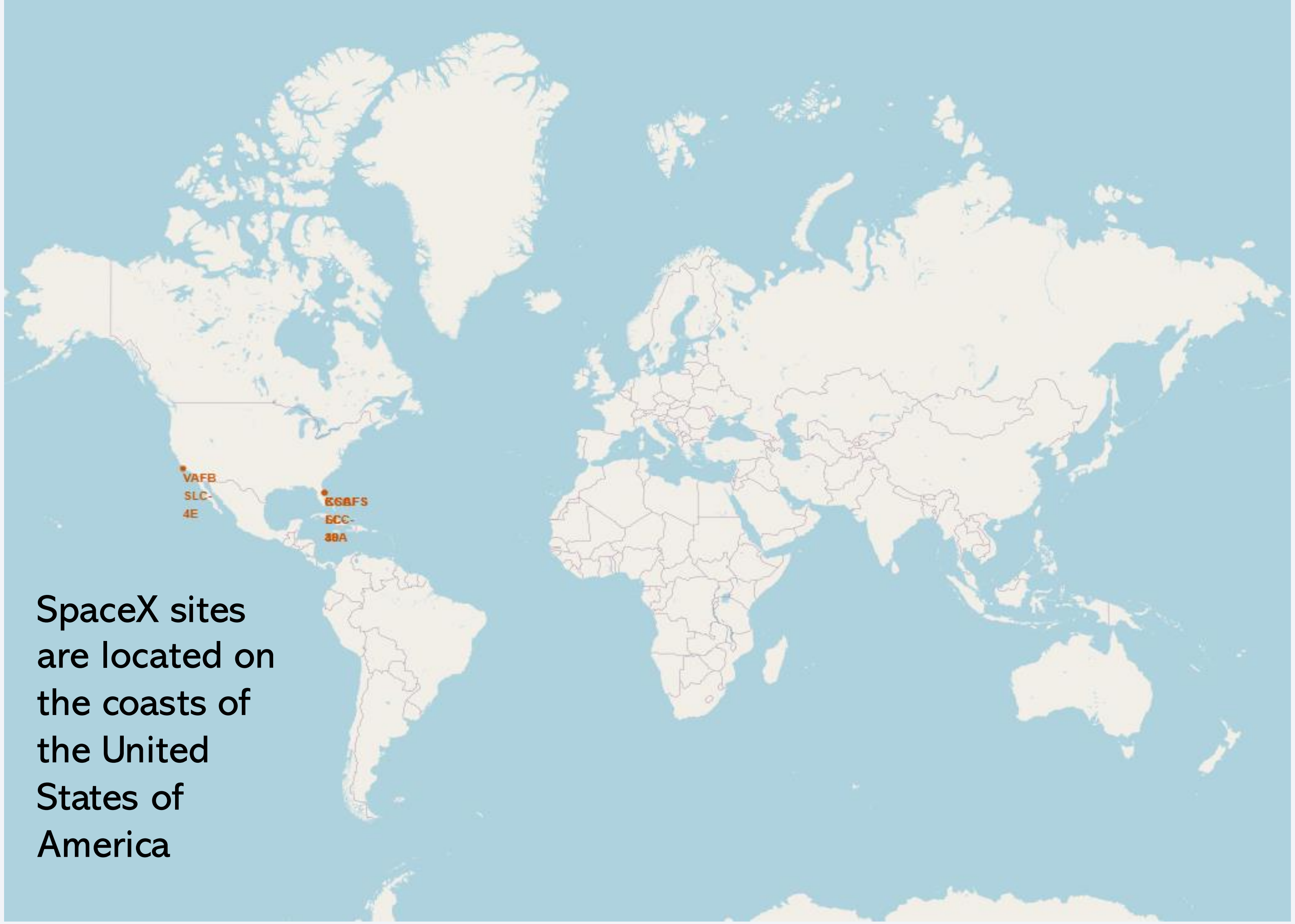


Landing_Outcome	Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

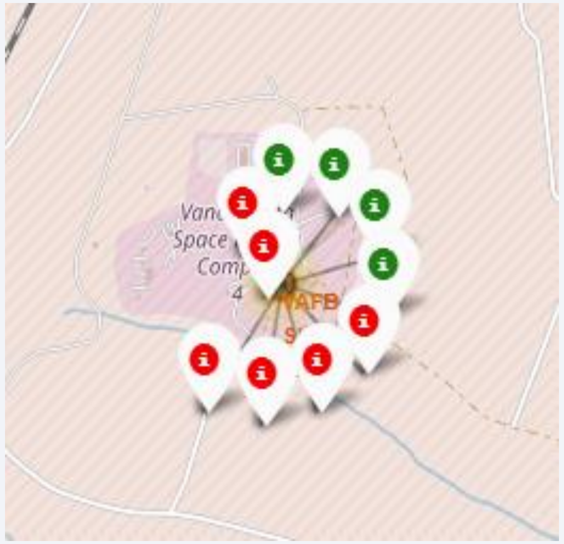
SpaceX Worldwide Launch Sites



SpaceX sites are located on the coasts of the United States of America

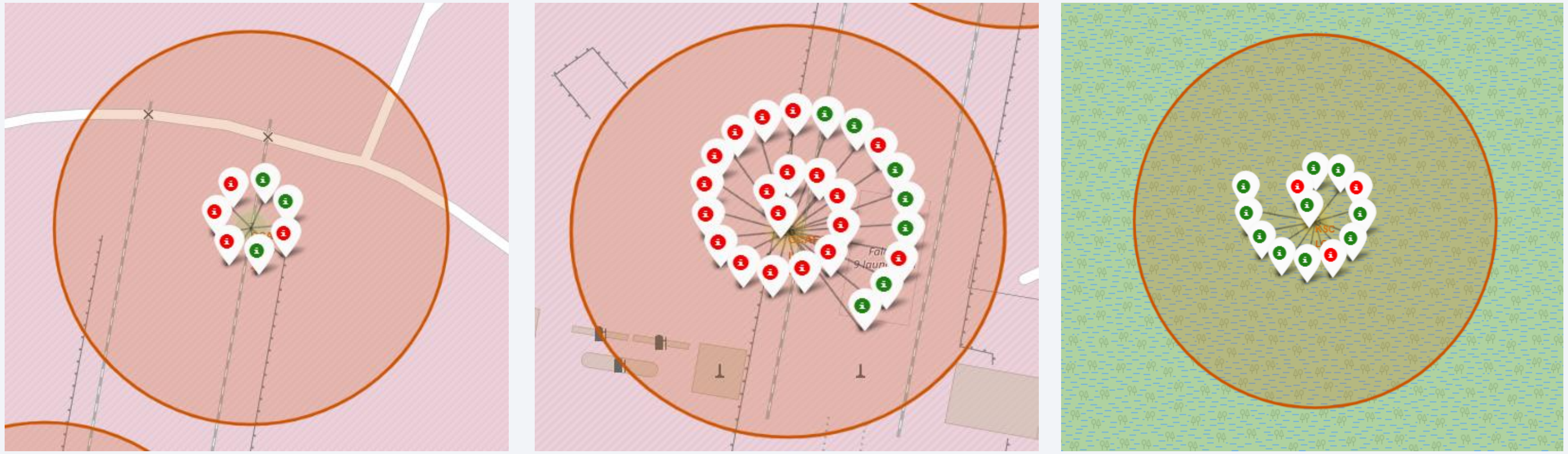
SpaceX Launch Outcomes at Launch Sites

Launch site Markers expand to show
Success and Failures of individual launches.



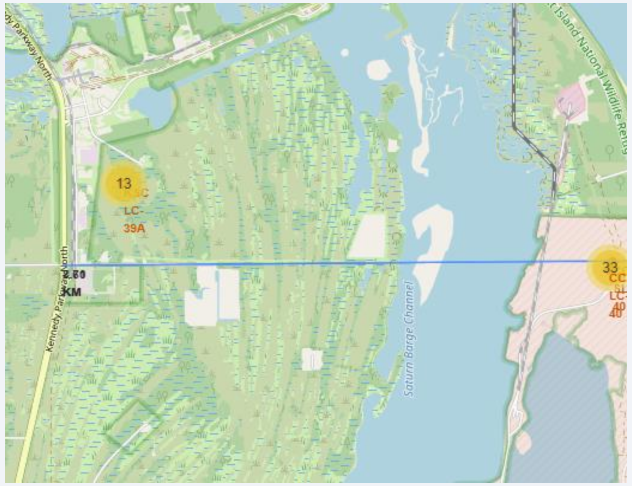
California Launch Site

Florida Launch Sites

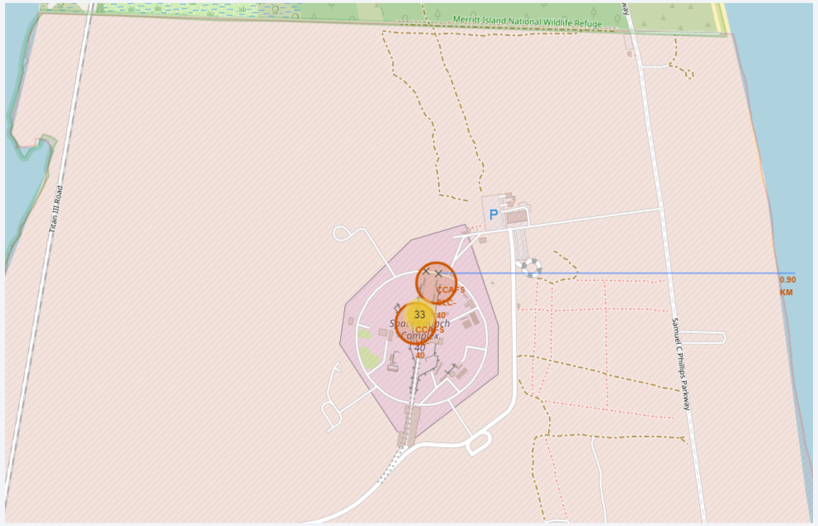


CCAFS SLC-40 Launch Site Distance to Landmarks

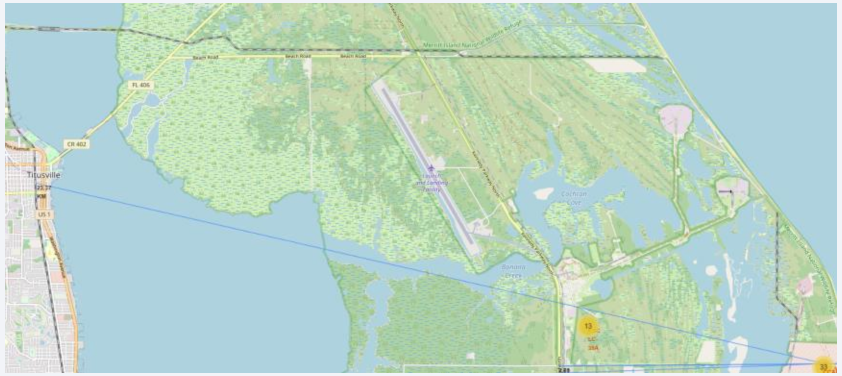
Blue lines show straight distance from Launch site to various landmarks.



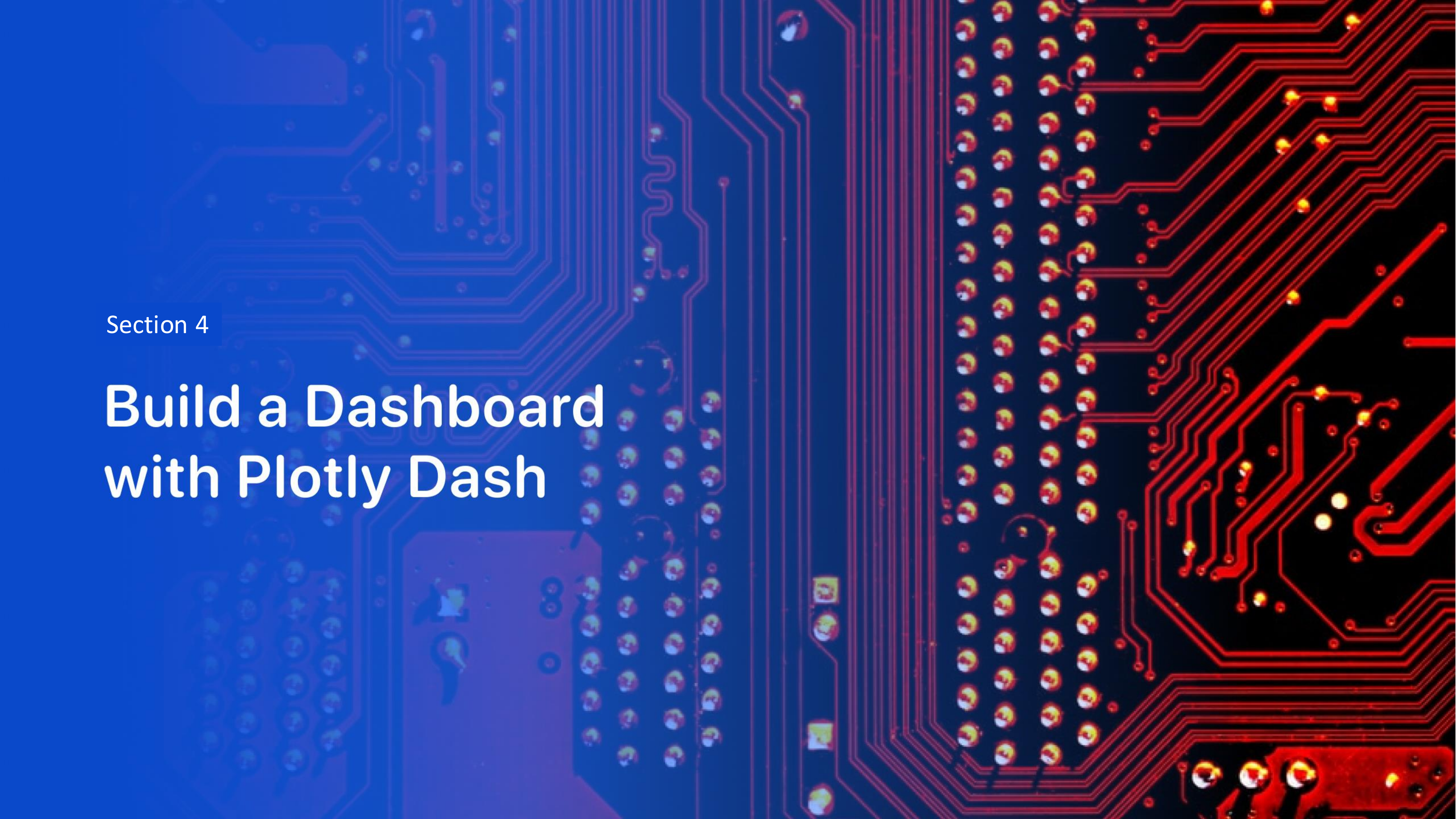
Distance to nearest highway:
8.51 KM


Merritt Island National Wildlife Refuge
Titan III Road
P
CCAFS
SLC-
40
33
Space Launch Complex
40
0.90
KM
Samuel C Phillips Parkway

Distance to nearest coast:
0.9 KM



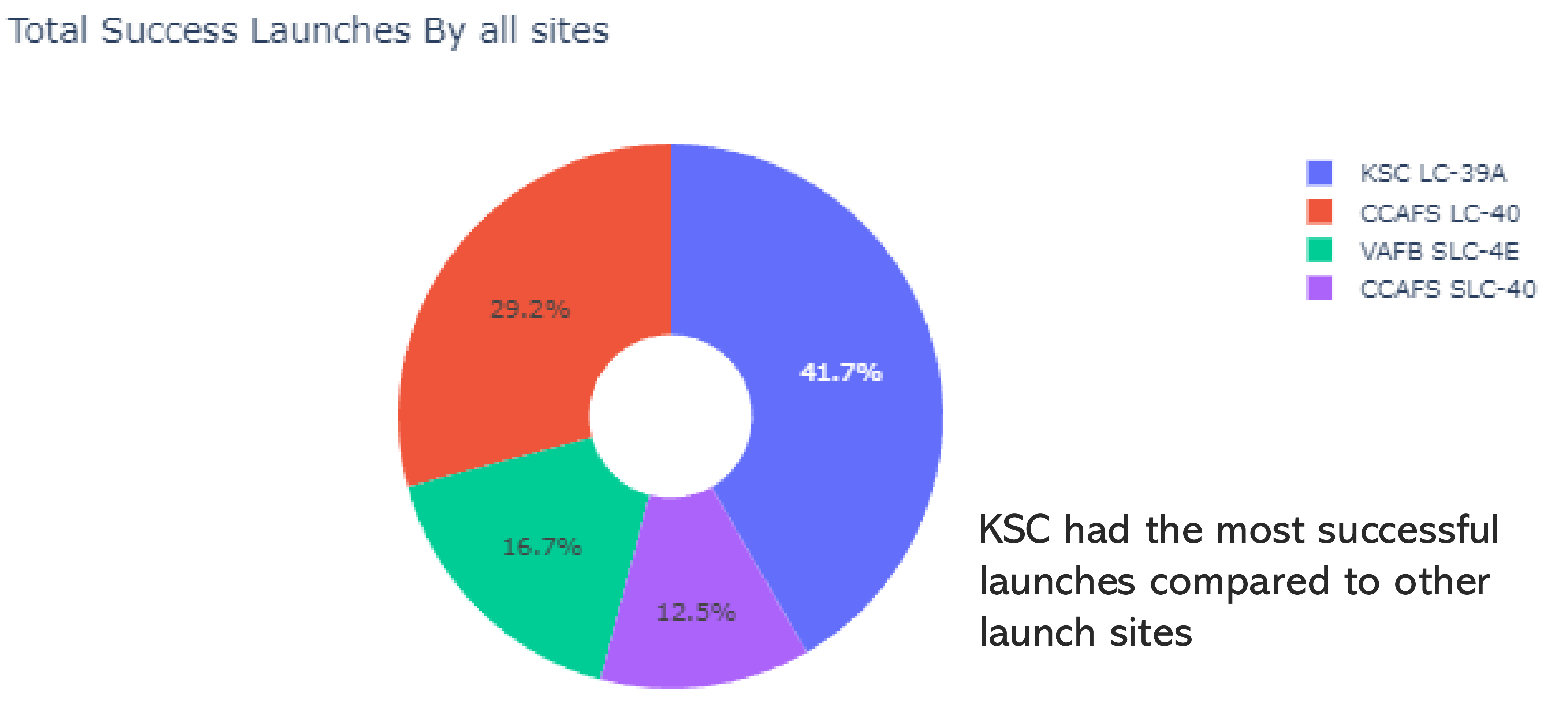
Distance to nearest city:
23.37KM



Section 4

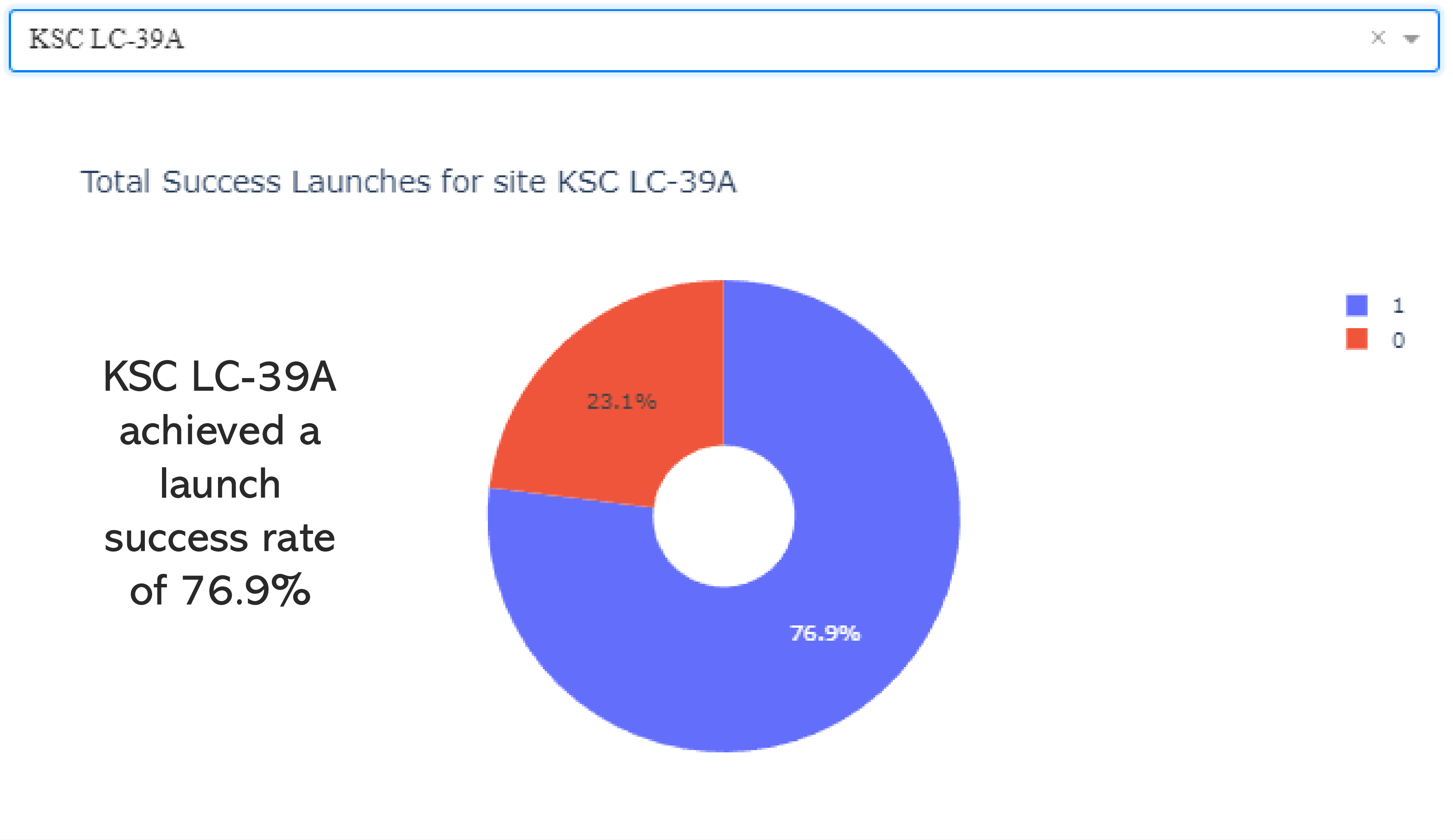
Build a Dashboard with Plotly Dash

SpaceX Successful Launches by Launch Site



KSC had the most successful launches compared to other launch sites

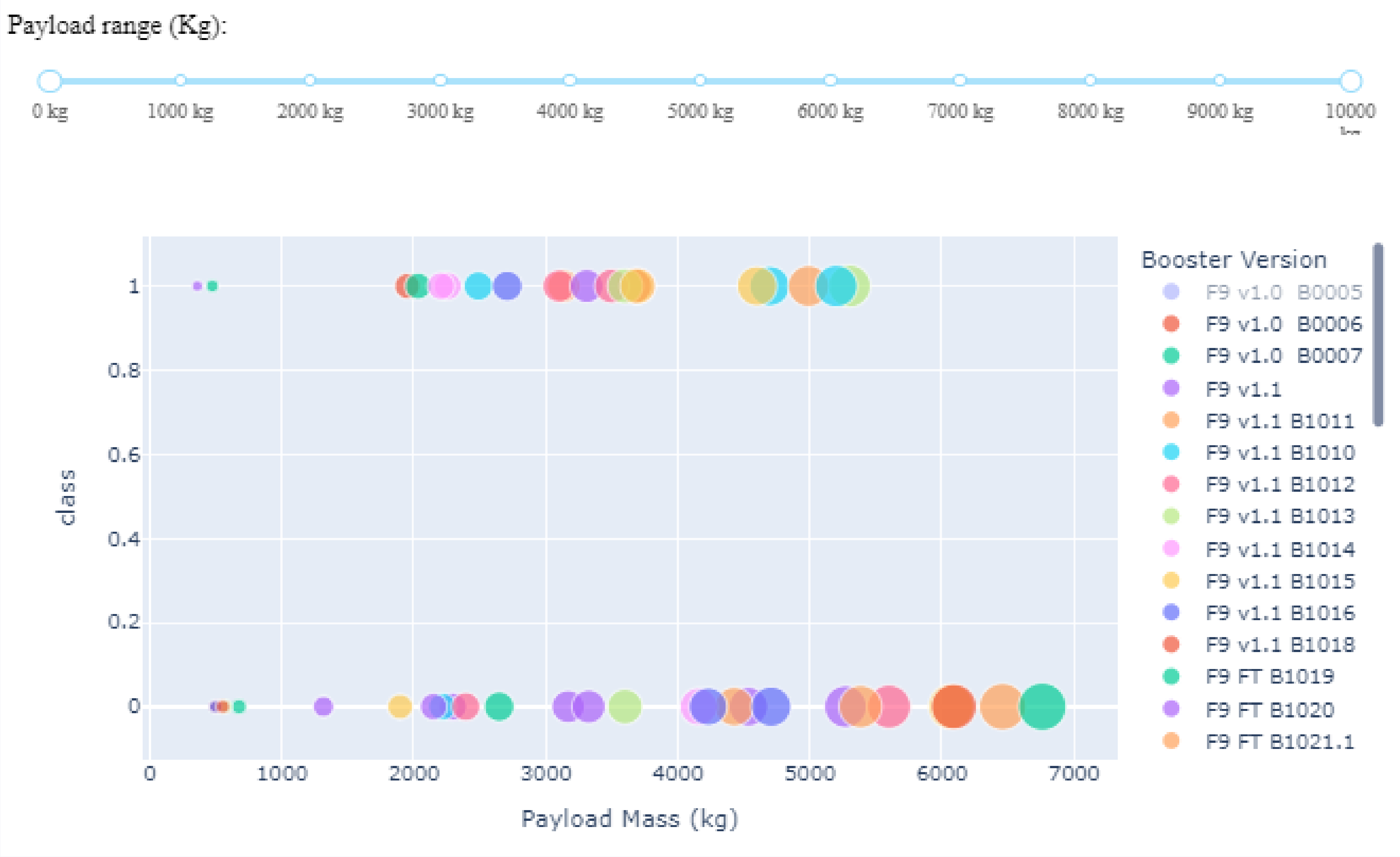
KSC LC-39A Launch Outcome Ratio



KSC LC-39A

KSC LC-39A achieved a launch success rate of 76.9%

Launch Outcome (0/1) for Payload Size



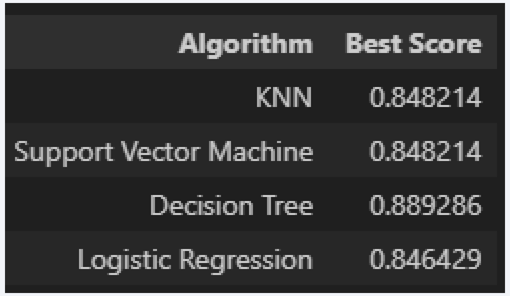
This Scatter Plot displays launch outcomes (success = 1, failure = 0) with count and booster rocket type color coded.

Section 5

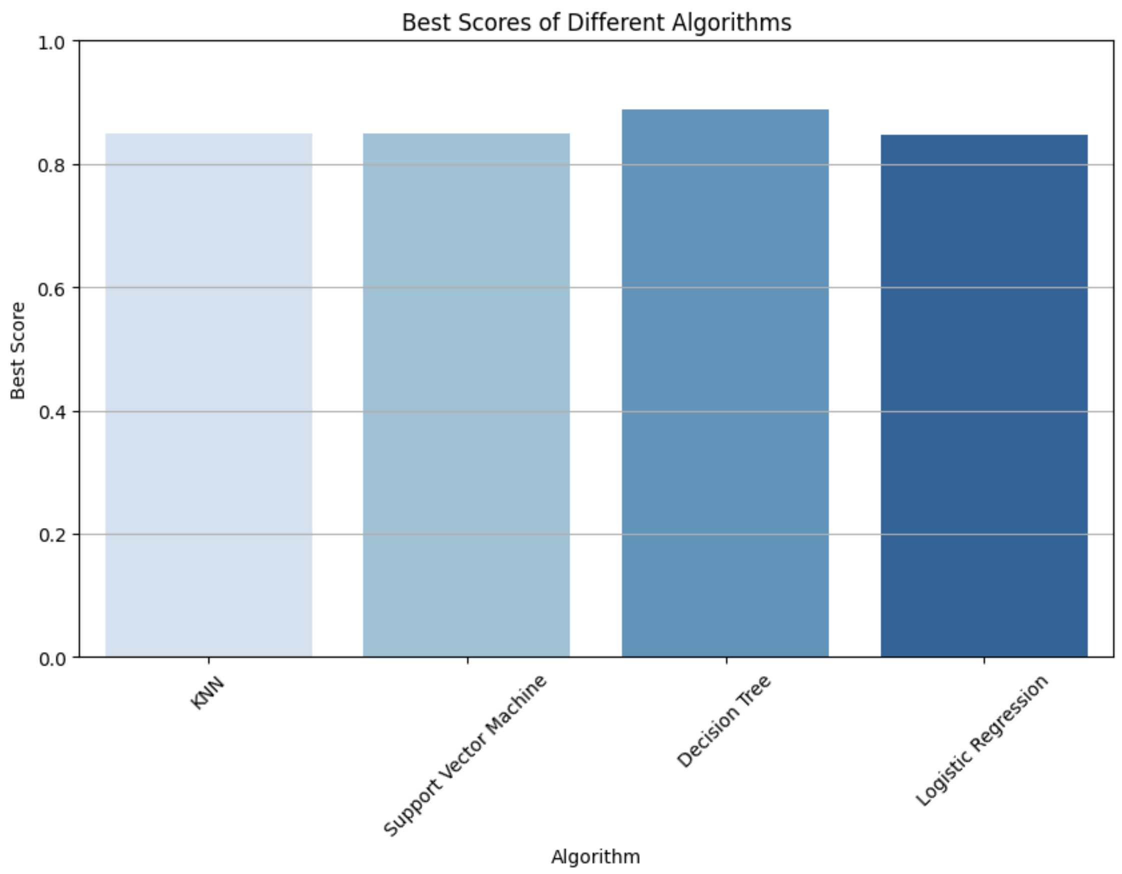
Predictive Analysis (Classification)

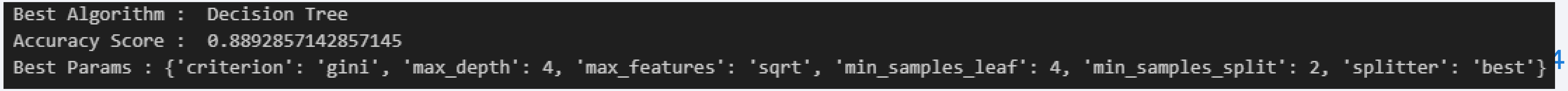
Classification Accuracy

All Models scored better than .80 accuracy, but the Decision Tree scored highest with accuracy score .8893 accuracy.



Algorithm	Best Score
KNN	0.848214
Support Vector Machine	0.848214
Decision Tree	0.889286
Logistic Regression	0.846429

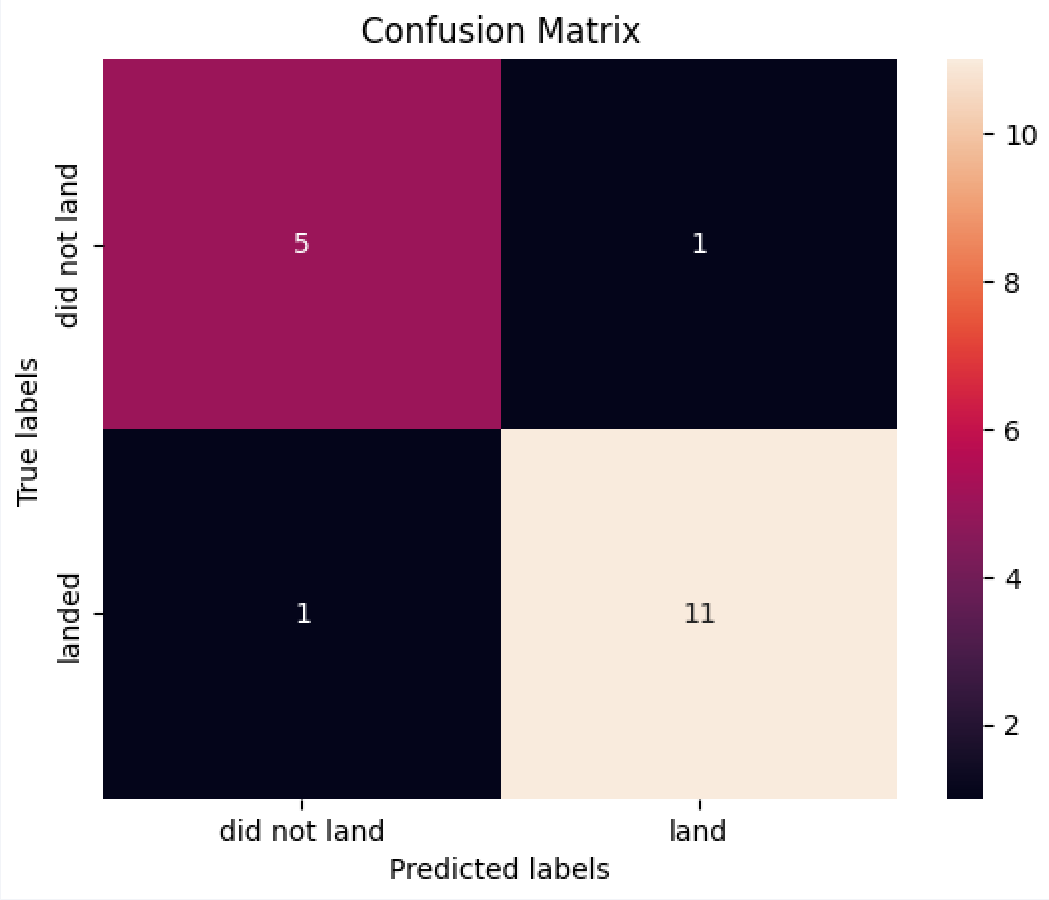




```
Best Algorithm :  Decision Tree
Accuracy Score :  0.8892857142857145
Best Params : {'criterion': 'gini', 'max_depth': 4, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 2, 'splitter': 'best'}
```

Confusion Matrix

The Confusion Matrics for the Decision Tree shows how well it accurately predicted in the top left and bottom right quadrants show the confusion matrix of the best performing model with an explanation. These are its true negatives and true positives. The bottom left and top right are false positive (Type II Error) and false negative (Type II Error).



Conclusions

- The more launch attempts by a site trend towards more successful launches
- The Orbits with the highest success rate are ES-L1, GEO, HEO, and SSO
- Larger payload mass trended towards better outcome, but this appears that it may be site and/or booster dependent
- KSC LC-39A Launch Site had the highest successful launch rate
- All Machine Learning models performed well based on achieving an accuracy score greater than 0.80
- The Decision Tree model was the overall best fit based on having the highest accuracy score across the testing data set

Appendix

All relevant code, libraries, and assests used can be found within the Github Repository here.

Thank you!