IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Rick Mozolic
April 4th, 2022

Outline

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

Executive Summary

- This report will aid SpaceY to be competitive with a company like SpaceX.
- We will determine what features of space launch correlate to first stage rocket launch and landing successes/failures.
  - Train machine learning models using SpaceX public available data to predict whether the first stage rocket lands successfully or not
  - Reuse of first stage rockets is key to reducing overall cost of launches
- The data does show that certain features have a correlation with the outcome of the launches.
- The overall best performing method for prediction of launch success was analyzed to be a Decision Tree

Introduction

- SpaceX is an established company that performs rocket launches that are relatively inexpensive.
- SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upwards of 165 million dollar
- Savings exist because SpaceX can reuse the first stage rocket.
- Problems as want to find answers
  - What correlations or features can we attribute to a successful first stage landing?
    - Payload Mass?
    - Orbit Type?
    - Launch Site?
  - What is the best algorithm that can be used to build the most accuracte prediction model?

Section 1

Methodology

Methodology

- Data collection methodology:
  - Descibe how data was collected
- Perform data wrangling
  - Describe how data was processed
- 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-SpaceX API/Wrangling

- A GET request was executed on the API https://api.spacexdata.com/v4/launches/past
- Utilized json_normalize method to convert json result and turned it into a Pandas dataframe.
- Developed another API to get information on rocket, payloads, launchpads, and cores storing the information into a list and creating another dataframe.

Import Libraries/Define Auxiliary Functions

- Requests
- Pandas
- Numpy
- datetime

Request/Parse Data

- GET request
- Normalize data: JSON
- Convert into Panda data frame

Filter/Missing Data

- Focus on Falcon 9 only data
- Replace missing data with mean PayloadMass

Data Collection-SpaceX API/Wrangling (continued)

- Filtered new dataframe to only include Falcon 9 launches
- Calculated mean value of PayloadMass and replaced all null values with the mean.

Out[40]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad
4	1	2010-06-04	Falcon 9	6123.547647	LEO	CCSFS SLC 40	None None	1	False	False	False	None
5	2	2012-05-22	Falcon 9	525.000000	LEO	CCSFS SLC 40	None None	1	False	False	False	None
6	3	2013-03-01	Falcon 9	677.000000	ISS	CCSFS SLC 40	None None	1	False	False	False	None
7	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None
8	5	2013-12-03	Falcon 9	3170.000000	GTO	CCSFS SLC 40	None None	1	False	False	False	None

GitHub URL: https://github.com/Mozo752/Applied-Data-Science-Capstone-IBM-Coursera/blob/master/API_Collection

Data Collection - Scraping

- A GET request was executed on the wiki page:
  https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922
- Used BeautifulSoup() to create a BeautifulSoup object
- Extracted all column and variable names from the HTML table header

Import Libraries/Define Auxiliary Functions

- Sys
- Requests
- BeautifulSoup
- Re,unicodedata
- Pandas

Request Wiki Page URL

- GET request
- BeautifulSoup object creation
- Extract all column/variable names from HTML table header

Parse Data

- Create empty dictionary and fill with data
- Convert into panda dataframe

GitHub URL: https://github.com/Mozo752/Applied-Data-Science-Capstone-IBM-Coursera/blob/master/Web%20Scraping.ipynb

Data Collection – Scraping (continued)

- Created a dataframe utilizing dictionary and parsed the launch HTML tables and converted into a Panda data frame.

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(BSS.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            #print(flight_number)
            launch_dict['Flight No.'].append(flight_number)
            print(flight_number)
            datatimelist=date_time(row[0])

            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            date = datatimelist[0].strip(',')
            print(date)

            # Time value
            # TODO: Append the time into launch_dict with key `Time`
            time = datatimelist[1]
            print(time)

            # Booster version
            # TODO: Append the bv into launch_dict with key `Version Booster`
            bv=booster_version(row[1])
            if not(bv):
                bv=row[1].a.string
            print(bv)
```

GitHub URL: https://github.com/Mozo752/Applied-Data-Science-Capstone-IBM-Coursera/blob/master/Web%20Scraping.ipynb

Data Wrangling

- NOTE: I BELIEVE I WAS ABLE TO DESCRIBE THIS PROCESS IN THE PREVIOUS SECTIONS

EDA with Data Visualization

- Charts to understand/discover relationships between several features
  - Matloblib and Seaborn: Scatterplots, Bar Charts, Line Charts
  - Features and associated Charts
    - **Flight # vs Launch Site: Scatterplot-easily shows relationship between the two variables**
    - **Payload Mass vs. Launch Site: Scatterplot-Same reason as above**
    - **Success Rate vs. Orbit Type: Bar Chart- easily compares data (side by side) amongst different categories; in this case the different orbit types**
- Add the GitHub URL of your completed EDA with data visualization notebook, as an external reference and peer-review purpose

GitHub URL: https://github.com/Mozo752/Applied-Data-Science-Capstone-IBM-Coursera/blob/master/EDA%20with%20Visualization.ipynb

EDA with SQL

- Following the Data Collection process, we were able to easily query the data frame with SQL to answer several questions:

  - SELECT FROM to display unique launch sites in the space mission
  - SELECT FROM WHERE LIKE to display specific launch sites that began with a specific string
  - SELECT SUM() FROM WHERE to display total payload mass carried by boosters by a specific customer
  - SELECT AVERAGE() FROM WHERE to calculate average payload
  - SELECT min() FROM WHERE to find first successful landing outcome
  - SELECT FROM WHERE BETWEEN to list specific booster version in a specific range
  - SELECT FROM WHERE utilizing a subquery, SELECT MAX() FROM to list booster versions with a max payload
  - SELECT FROM WHERE to list multiple columns that met a criteria
  - SELECT, COUNT as COUNT FROM WHERE BETWEEN GROUP BY ORDER BY to rank outcomes between a certain date and list them in descending order

GitHub URL: https://github.com/Mozo752/Applied-Data-Science-Capstone-IBM-Coursera/blob/master/EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

- Folium Map was created utilizing various objects to visualize data on an interactive map.
- The following objects were used:
  - Launch Site Maps (Circle and Marker) – To label and show geographic location of all the launches
  - Launch Success/Failures (MarkerCluster) – To visually label a launch as successful (green) or a failure (red) at each site.
  - Proximity of Launch Site to nearest Coastline (Marker, Polyline) – Measured and displayed launch site distance to a feature on the map.

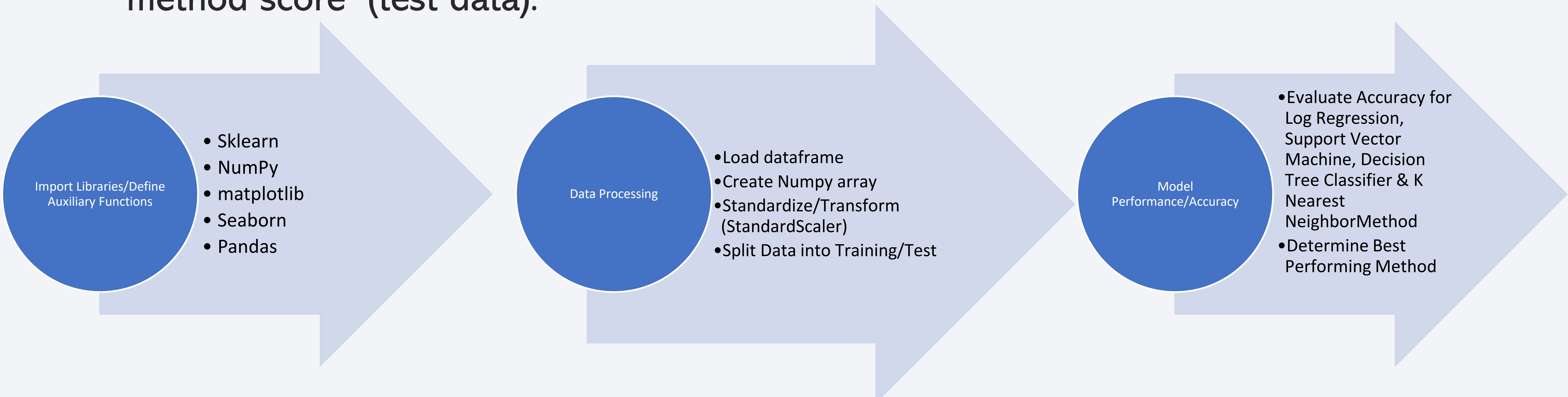
GitHub URL: https://github.com/Mozo752/Applied-Data-Science-Capstone-IBM-Coursera/blob/master/Folium

Build a Dashboard with Plotly Dash

- Summarize what plots/graphs and interactions you have added to a dashboard
- Explain why you added those plots and interactions
- Add the GitHub URL of your completed Plotly Dash lab, as an external reference and peer-review purpose
- CURRENTLY TROUBLESHOOTING LAB TO OBTAIN SCREEN SHOTS

Predictive Analysis (Classification)

- Overall, we took the data set and split it into training and test data. From there we created GridSearchCV objects from each of the methods for Log Regression, Support Vector Machine, Decision Tree, and K Nearest Neighbor (training data)
- Once we found the best parameters, we calculated the accuracy using the method score (test data).



Predictive Analysis (Classification) (continued)

- We then plotted a confusion matrix to show how the prediction of the test data of each method compared with the actual test data.
- To find the best overall method analyze the accuracy numbers and created Jaccard, F1, and LogLoss Scores for the methods

	Jaccard	F1-score	LogLoss
LogReg	0.800000	0.814815	0.478667
SVM	0.800000	0.814815	NaN
Decision Tree	0.846154	0.888889	NaN
KNN	0.800000	0.814815	NaN

GitHub URL: https://github.com/Mozo752/Applied-Data-Science-Capstone-IBM-Coursera/blob/master/Machine%20Learn

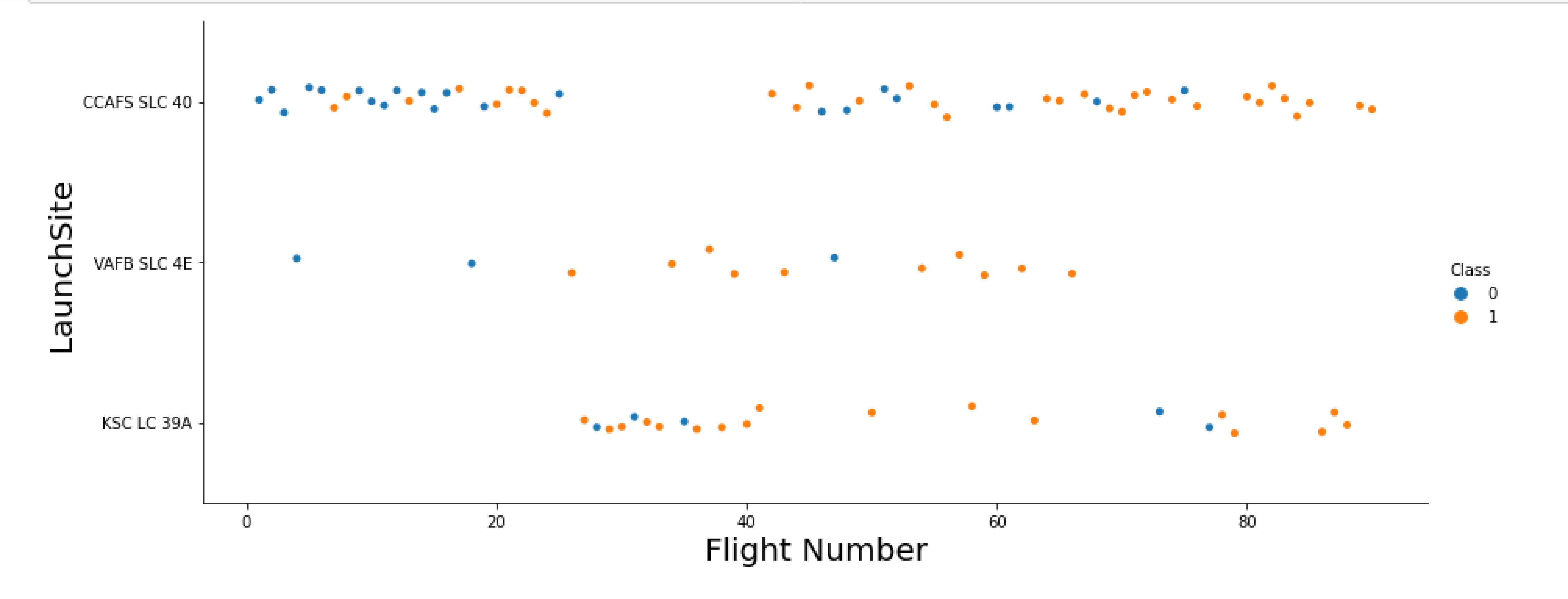
Results

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

Section 2

Insights drawn from EDA

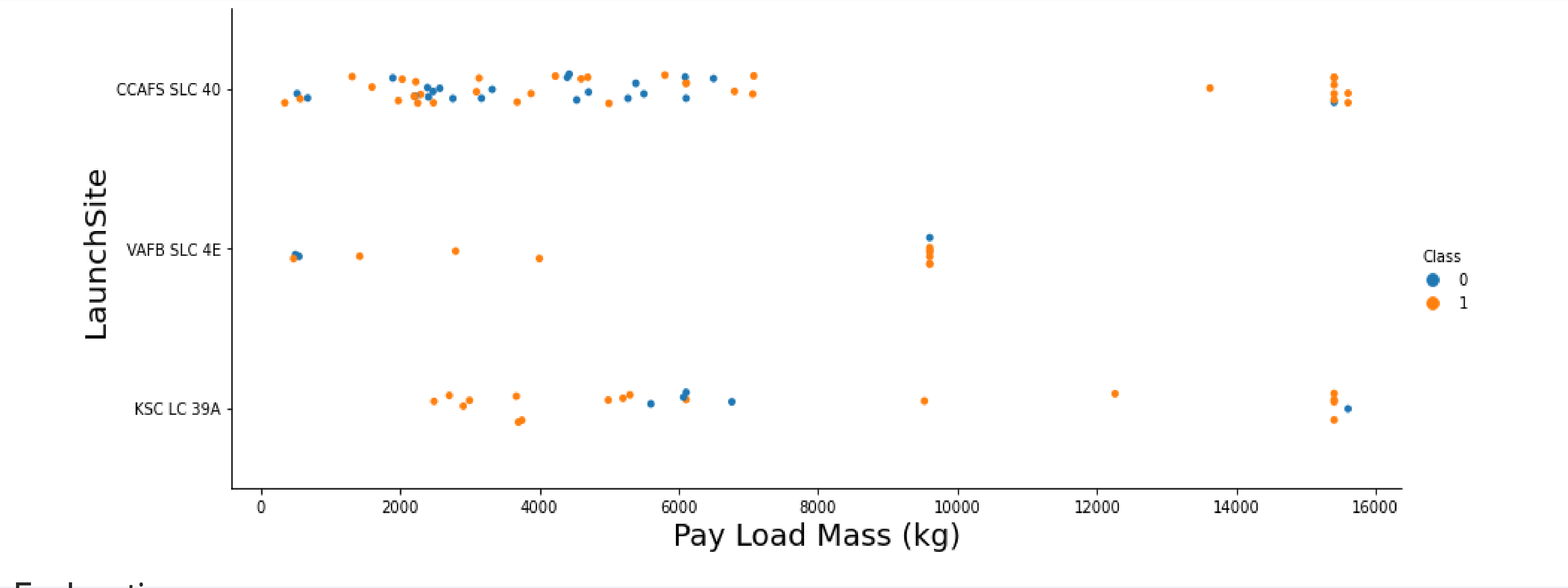
Flight Number vs. Launch Site



Explanation:

- Sites with Greatest Success Rate: KSC LC-39A and VAFB SLC-4E 77%
- Majority of Flights are launched from CCAFS LC-40 and KSC LC-39A

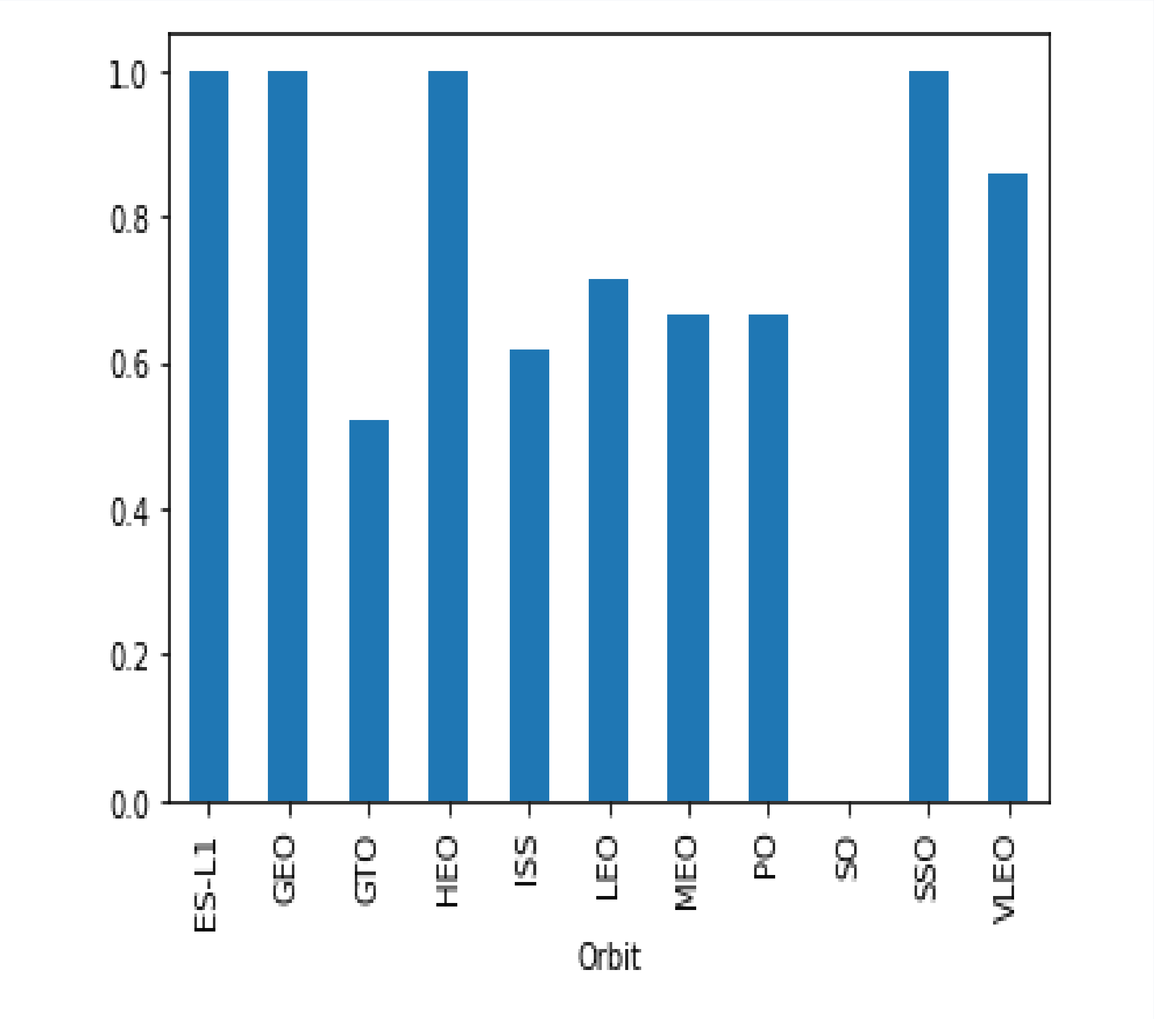
Payload vs. Launch Site



Explanation:

- VAFB SLC-4E has no rockets launched for a heavypayload mass greater than 10,000
- Heavier payloads (greater than 14,000) appeared to have a higher success rate at CCAFS LC-40 and KSC LC-39A

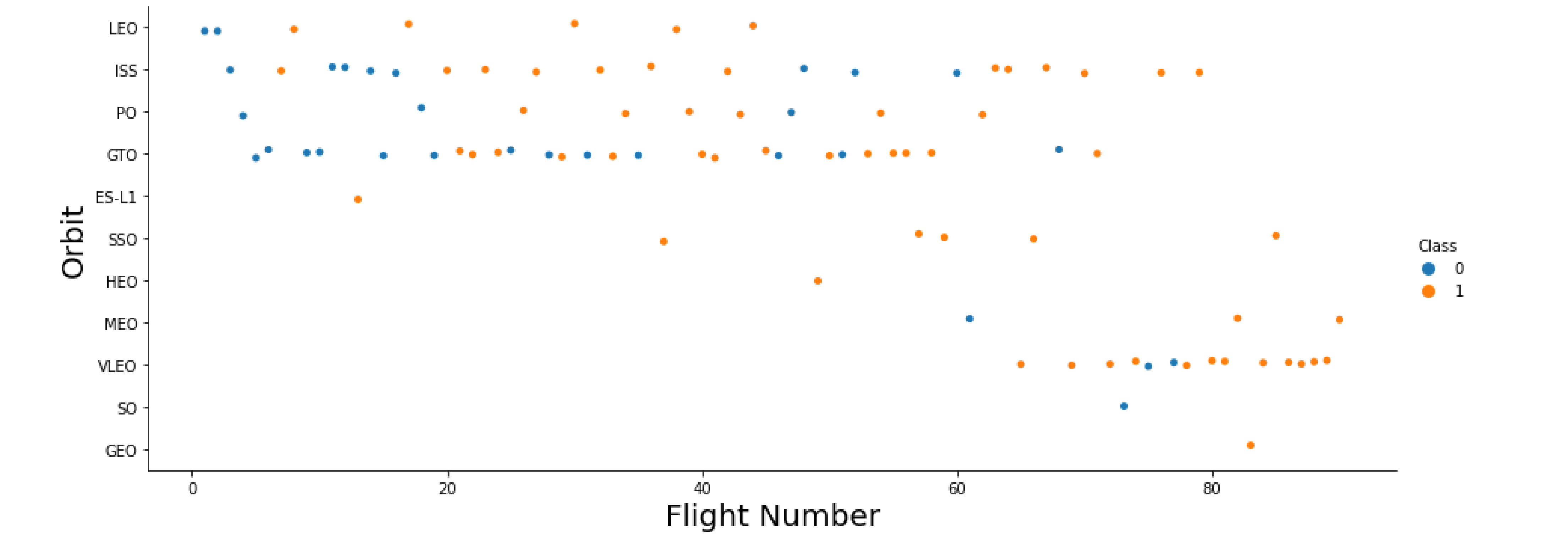
Success Rate vs. Orbit Type



Explanation:

- High Success Rates occur at these Orbits:
    - ES-L1,GEO, HEO, SSO
- Risk of failure exists at remaining Orbits

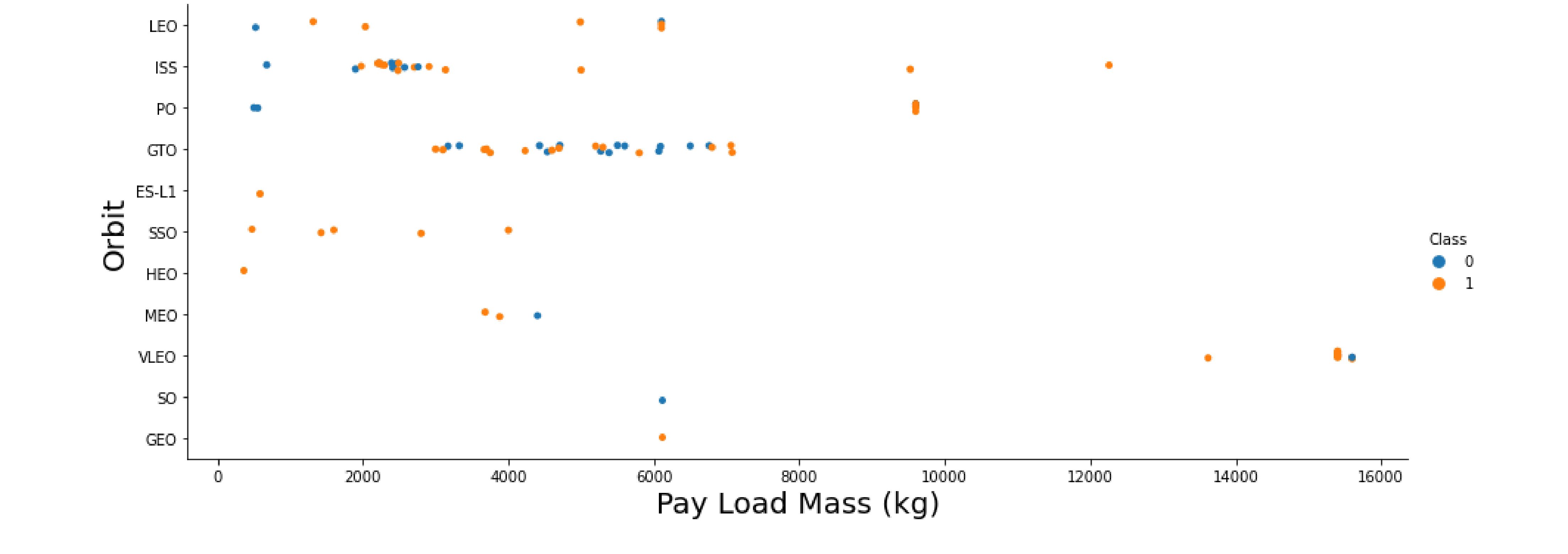
Flight Number vs. Orbit Type



Explanation:

- LEO Orbit success appears to be related to the number of flights
- For GTO, there appears to be no relationship between the flight number

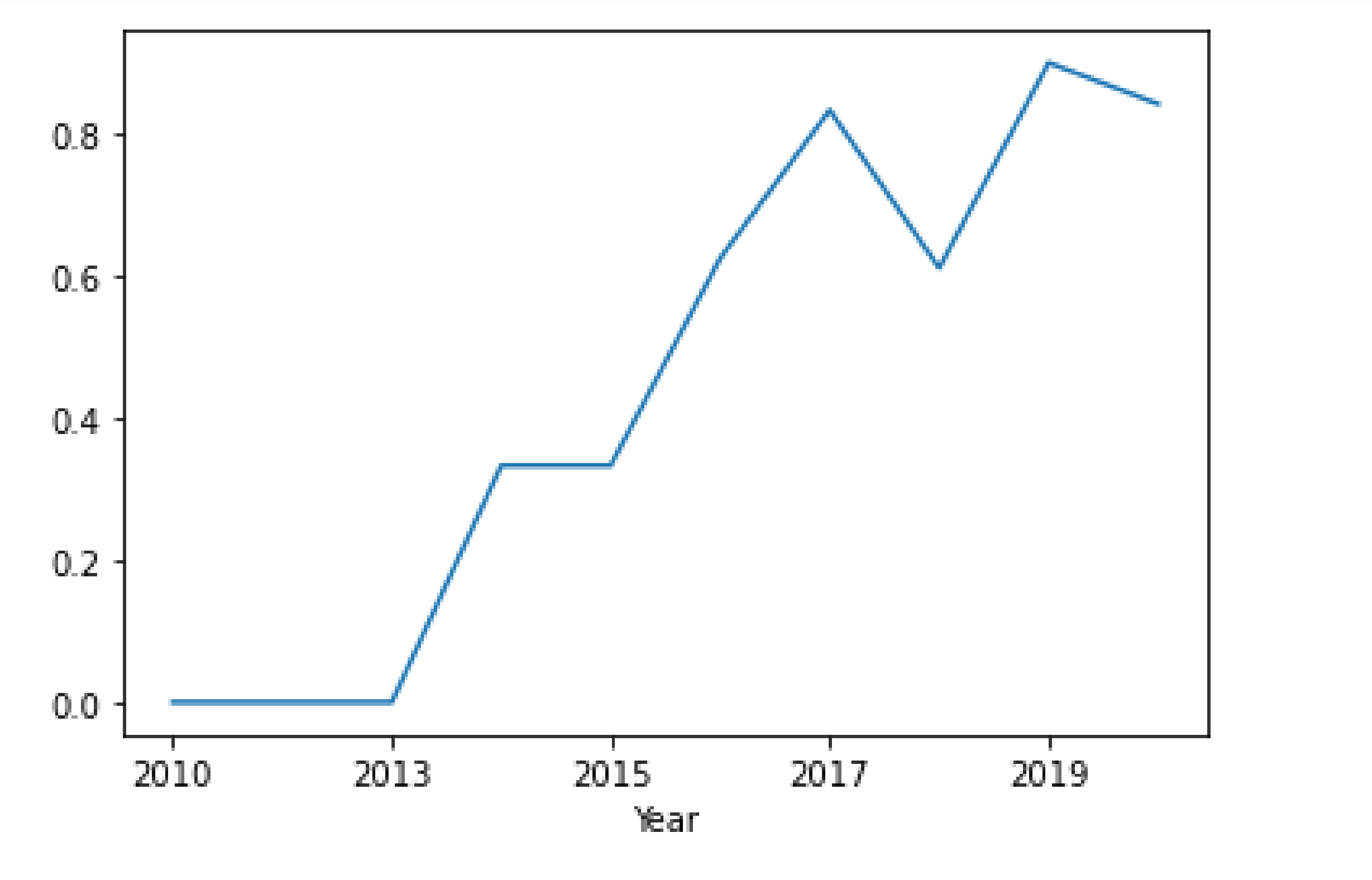
Payload vs. Orbit Type



Explanation:

- Heavy payloads the successful landing or positive landing rate are more for PO, VLEO, ISS
- GTO is hard to distinguish this relationship since both positive and landing rates exist.

Launch Success Yearly Trend



Explanation:

- Overall success rate since 2013 kept rising until 2020.

All Launch Site Names

- Executed a SQL query of the data to find distinct names of Launch Sites
- Results: Output Below

In [34]:
```
%sql SELECT distinct Launch_Site FROM SPACEXTABLE
```

```
 * ibm_db_sa://tcp26192:***@b70af05b-76e4-4bca-a1f5-23dbb4c6a74e.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32716/bludb
Done.
```

Out[34]:

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

Launch Site Names Begin with 'CCA'

- Executed a SQL query of the data to display specific launch sites that began with the specific string ‘CCA’
- Results: Output Below

In [35]:
```
%sql SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 5
```

```
 * ibm_db_sa://tcp26192:***@b70af05b-76e4-4bca-a1f5-23dbb4c6a74e.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32716/bludb
Done.
```

Out[35]:

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	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	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

Task 3

Total Payload Mass

- Executed a SQL query of the data to display the total payload mass by a specifc customer ‘NASA (CRS)’
- Results: Output Below

In [38]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE customer='NASA (CRS)'
```

```
 * ibm_db_sa://tcp26192:***@b70af05b-76e4-4bca-a1f5-23dbb4c6a74e.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32716/bludb
Done.
```

Out[38]:

1
67603

Average Payload Mass by F9 v1.1

- Executed a SQL query of the data to calculate the average payload mass of a specific booster version.
- Results: Output Below

```
In [41]: %sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE booster_version = 'F9 v1.1'

 * ibm_db_sa://tcp26192:***@b70af05b-76e4-4bca-a1f5-23dbb4c6a74e.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32716/bludb
Done.

Out[41]:
```

1
3209

First Successful Ground Landing Date

- Executed a SQL query of the data to identify the date of the first successful landing outcome
- Results: Output Below

In [42]:
```
%sql SELECT min(Date) FROM SPACEXTABLE WHERE Landing__outcome='Success'
```

```
 * ibm_db_sa://tcp26192:***@b70af05b-76e4-4bca-a1f5-23dbb4c6a74e.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32716/bludb
Done.
```

Out[42]:

1
2018-03-12

Successful Drone Ship Landing with Payload between 4000 and 6000

- Executed a SQL query of the data to display the list of a specific booster version within a specific payload range
- Results: Output Below

In [45]:
```
%sql SELECT booster_version FROM SPACEXTABLE where payload_mass__kg_ between 4000 and 6000 AND landing__outcome='Success (drone ship)'
```

```
 * ibm_db_sa://tcp26192:***@b70af05b-76e4-4bca-a1f5-23dbb4c6a74e.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32716/bludb
Done.
```

Out[45]:

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

Total Number of Successful and Failure Mission Outcomes

- Executed a SQL query of the data to count total number of success and failed missions
- Results: Output Below

```
In [73]: %sql SELECT COUNT(*) FROM SPACEXTABLE WHERE mission_outcome like 'Success%' or mission_outcome like 'Failure%'

 * ibm_db_sa://tcp26192:***@b70af05b-76e4-4bca-a1f5-23dbb4c6a74e.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32716/bludb
Done.
```

Out[73]:

1
146

Boosters Carried Maximum Payload

- Executed a SQL query and subquery of the data to list all the boosters versions that carried the maximum payload.
- Results: Output Below

In [74]:
```
%sql SELECT booster_version FROM SPACEXTABLE WHERE Payload_Mass__kg_ = (SELECT MAX(Payload_Mass__kg_) FROM SPACEXTABLE)
```

```
 * ibm_db_sa://tcp26192:***@b70af05b-76e4-4bca-a1f5-23dbb4c6a74e.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32716/bludb
Done.
```

Out[74]:

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

2015 Launch Records

- Executed a SQL query of the data to display all the failed landing outcomes, booster versions, and the launch sites in 2015
- Results: Output Below

In [100]:
```
%sql SELECT Landing__outcome, booster_version, launch_site, Date  FROM SPACEXTABLE where Landing__outcome ='Failure (drone shi
p)' and Date  like '%2015%'
```

```
 * ibm_db_sa://tcp26192:***@b70af05b-76e4-4bca-a1f5-23dbb4c6a74e.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32716/bludb
Done.
```

Out[100]:

landing__outcome	booster_version	launch_site	DATE
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40	2015-01-10
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40	2015-04-14
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40	2015-10-01

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

- Executed a SQL query of the data to rank the landing outcomes between a date range and listing them in descending order
- Results: Output Below

```
In [161]: %sql SELECT Date, COUNT(Landing__outcome) as COUNT FROM SPACEXTABLE WHERE Date BETWEEN '2010-06-04' and '2017-03-20' GROUP BY Da
te ORDER BY COUNT(Landing__outcome) DESC

 * ibm_db_sa://tcp26192:***@b70af05b-76e4-4bca-a1f5-23dbb4c6a74e.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32716/bludb
Done.
```

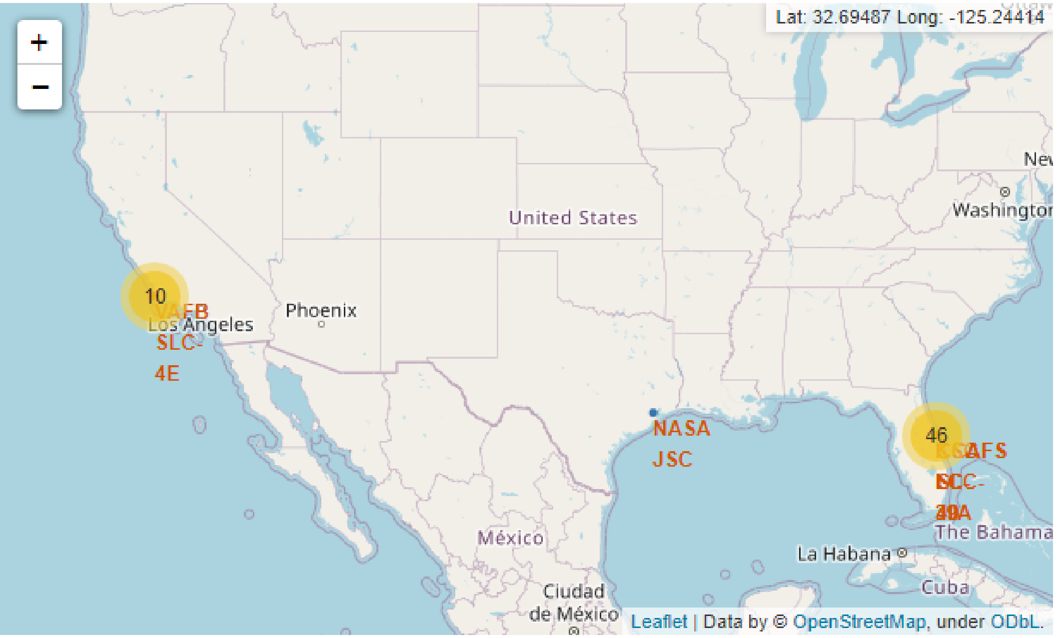
Out[161]:

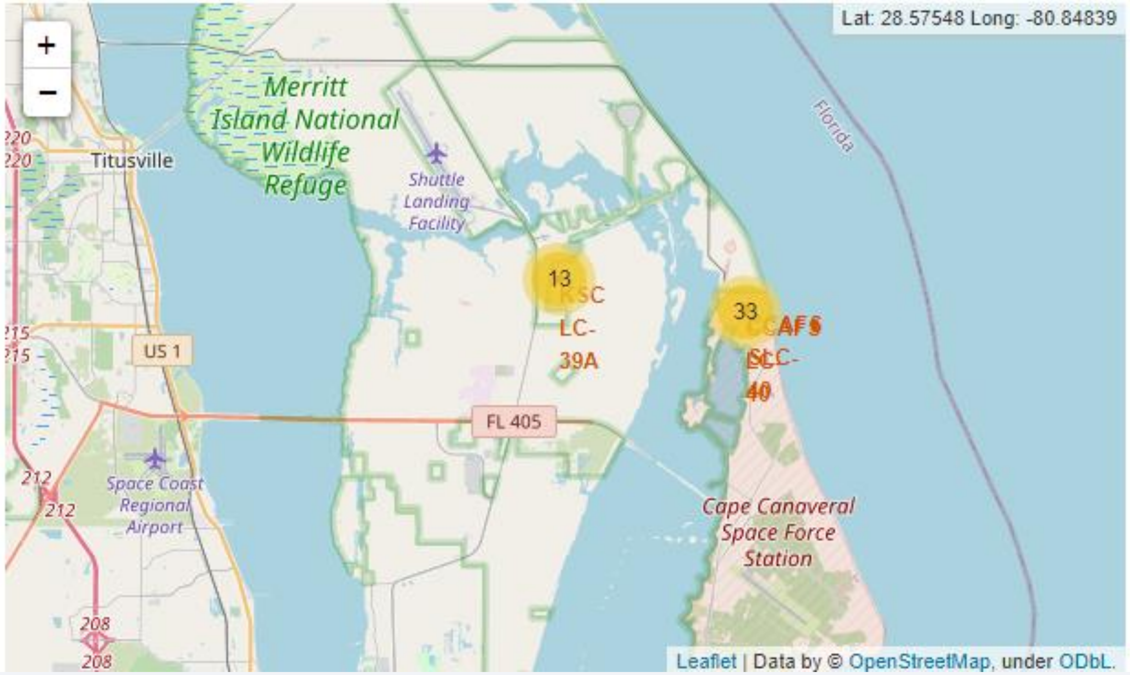
DATE	COUNT
2010-06-04	1
2010-08-12	1
2010-12-08	1
2012-05-22	1
2012-08-10	1
2012-10-08	1
2013-01-03	1
2013-03-01	1
2013-03-12	1
2013-09-29	1
2013-12-03	1
2014-01-06	1
2014-04-18	1
2014-05-08	1
2014-06-01	1
2014-07-09	1
2014-07-14	1
2014-08-05	1
2014-09-07	1
2014-09-21	1
2015-01-10	1
2015-02-03	1
2015-02-11	1
2015-03-02	1
2015-04-14	1
2015-04-27	1
2015-06-28	1
2015-10-01	1
2015-11-02	1
2015-12-22	1
2016-01-17	1
2016-03-04	1
2016-04-03	1
2016-04-08	1
2016-05-06	1
2016-05-27	1
2016-06-05	1
2016-06-15	1
2016-07-18	1
2016-08-04	1
2016-08-14	1
2017-01-05	1
2017-01-14	1
2017-02-19	1
2017-03-06	1
2017-03-16	1

Section 4

Launch Sites Proximities Analysis

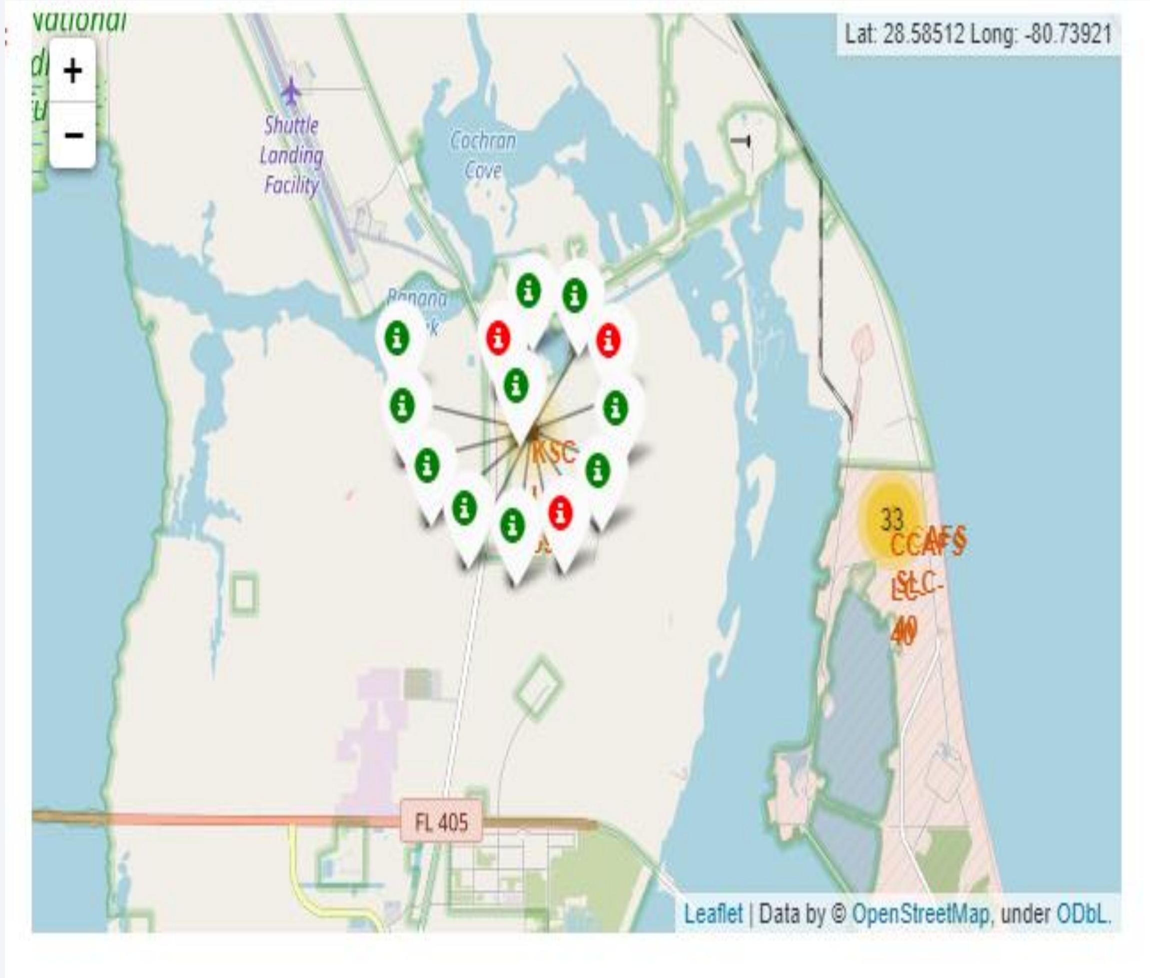
Folium Map: All Launch Sites





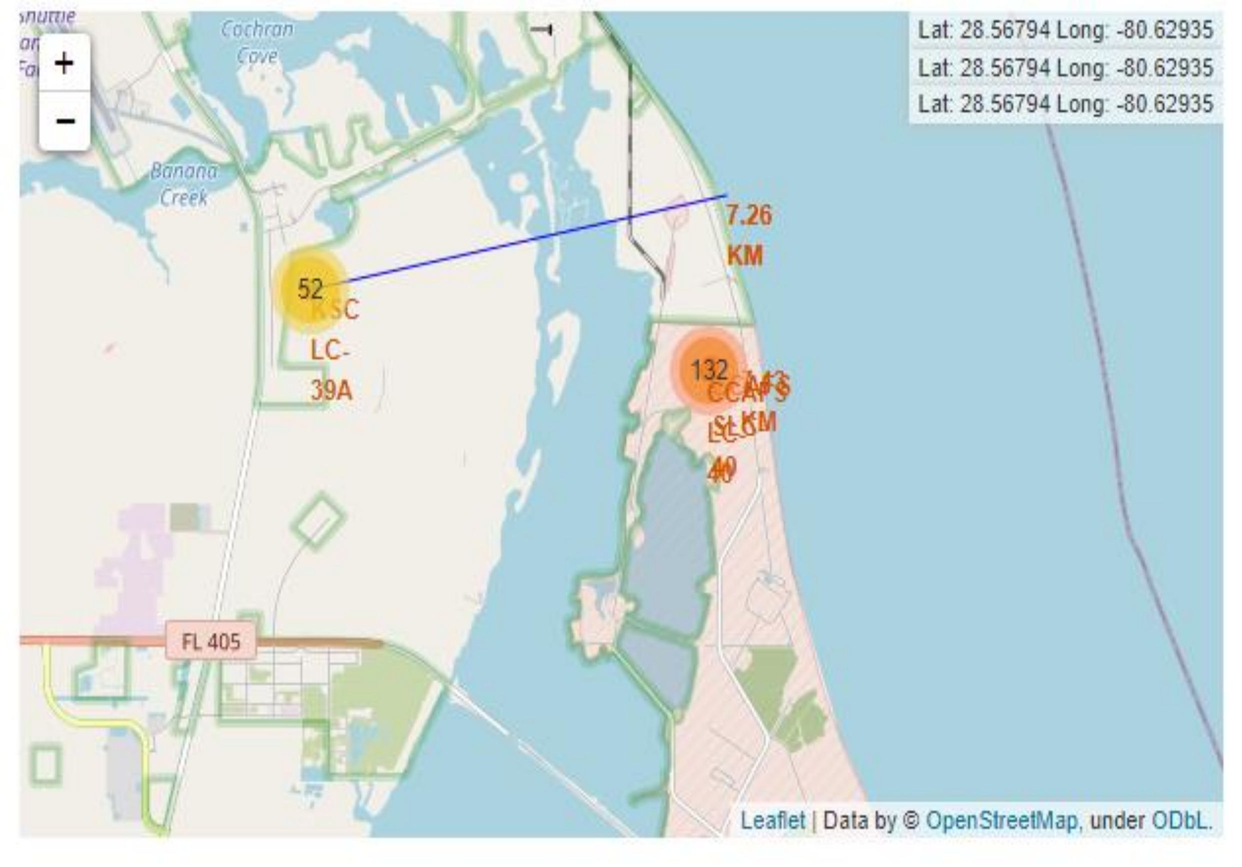
- There were 10 launched in California at the VAFB SLC-4E
- There were 46 launches in Florida, 13 at KSC LC-39A & 33 CCAFS LC-40
- They are geographically marked and labled utilizing circle and marker cluster

Folium Map: Success/Failed Launches



- Each launch was proper labled and marked utilizing a marker cluster.
- Each marker was then assigned a color, red for failure, green for success.
- In this example, you can see out of the 13 launches, 3 were failed outcomes and 10 were successful

Folium Map: Launch Site Proximity to Coastline



- We were able to utilize math function to calculate the distance between the longitude/latitude of the site KSC LC-39A and the coastline
- We then created a Polyline object using the coordinates and added the calculation of the distance between those coordinates.

Section 5

Build a Dashboard with Plotly Dash

Launch Success Count

- Issues with completing Week 3 Plotly Lab…Awaiting remedy to complete lab and take screen shots
- Show the screenshot of launch success count for all sites, in a piechart
- Explain the important elements and findings on the screenshot

Highest Launch Success Ratio

- **Issues with completing Week 3 Plotly Lab…Awaiting remedy to complete lab and take screen shots**
- Show the screenshot of the piechart for the launch site with highest launch success ratio
- Explain the important elements and findings on the screenshot

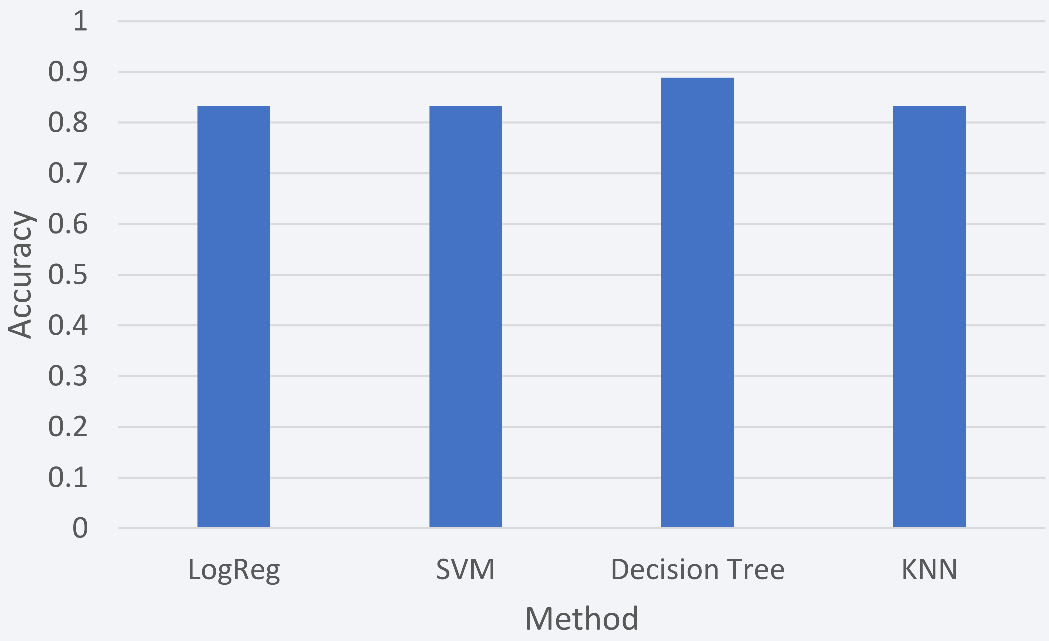
Payload vs. Launch Outcome

- **Issues with completing Week 3 Plotly Lab…Awaiting remedy to complete lab and take screen shots**
- Show screenshots of Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider
- Explain the important elements and findings on the screenshot, such as which payload range or booster version have the largest success rate, etc.

Section 6

Predictive Analysis (Classification)

Classification Accuracy

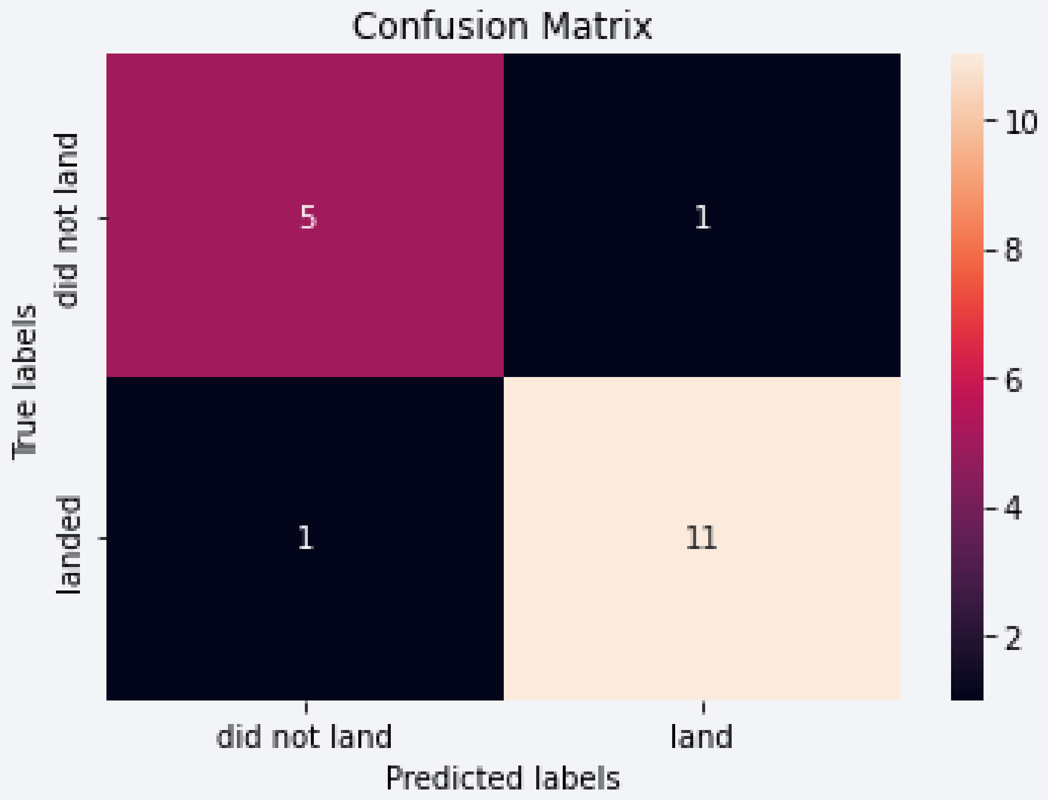


- The Decision Tree has the highest classification Accuracy
- See Bar Chart (left) and Scores (below)

	Jaccard	F1-score	LogLoss
LogReg	0.800000	0.814815	0.478667
SVM	0.800000	0.814815	NaN
Decision Tree	0.846154	0.888889	NaN
KNN	0.800000	0.814815	NaN

Confusion Matrix

- Decision Tree Confusion Matrix
  - True Positive (Landed: 11)
  - True Negative (Did Not Land: 5)
  - Out of the 18 test data, 16 was correctly predicted



Conclusions

- The Decision Tree Model is the best algorithm for this particular dataset
  - Accuracy scores generated from the Train and Test data were the highest.
  - Confusion matrix generated the most True Positives and True Negatives
- Features such as Payload Mass and Orbit Type should be considered toward determining the cost of a launch since it appears that those have a correlation to the success of landings resulting in the likelihood of reusing rockets
- 2 of 3 sites have data on heavier payloads. Consider those when launching anything greater than 10,000kg.
- Launch Success Rate has been continuing to rise since 2013. With the current data provided from SpaceX, we can accurately predict future launch success.

Appendix

- Model Scores Calculation

```
In [62]: from sklearn.metrics import jaccard_score
         from sklearn.metrics import f1_score
         from sklearn.metrics import log_loss

In [54]: #KNN Score

         y_pred_knn = knn_cv.predict(X_test)

         knn_jaccard = jaccard_score(Y_test, y_pred_knn)
         knn_f1 = f1_score (Y_test, y_pred_knn, average = "weighted")

In [55]: #Decision Tree Score
         y_pred_tree = tree_cv.predict(X_test)

         tree_jaccard = jaccard_score(Y_test, y_pred_tree)
         tree_f1 = f1_score (Y_test, y_pred_tree, average = "weighted")

In [56]: #SVM Score
         y_pred_svm = svm_cv.predict(X_test)

         svm_jaccard = jaccard_score(Y_test, y_pred_svm)
         svm_f1 = f1_score (Y_test, y_pred_svm, average = "weighted")

In [67]: # LogReg Score

         Y_pred_logreg = logreg_cv.predict(X_test)

         LogReg_jaccard = jaccard_score(Y_test, Y_pred_logreg)
         LogReg_f1 = f1_score(Y_test, Y_pred_logreg, average = 'weighted')

         Y_pred_logreg_prob = logreg_cv.predict_proba(X_test)

         LogReg_logloss = log_loss(Y_test, Y_pred_logreg_prob)

In [68]: Jaccard = np.full(4, np.nan)
         F1_score = np.full(4, np.nan)
         LogLoss = np.full(4, np.nan)
         Algorithm = np.array(4)
         Algorithm = ["LogReg", "SVM", "Decision Tree", "KNN"]

In [69]: Jaccard[0] = LogReg_jaccard
         Jaccard[1] = svm_jaccard
         Jaccard[2] = tree_jaccard
         Jaccard[3] = knn_jaccard
```

```
In [70]: F1_score[0] = LogReg_f1
         F1_score[1] = svm_f1
         F1_score[2] = tree_f1
         F1_score[3] = knn_f1

In [71]: LogLoss[0] = LogReg_logloss

In [73]: Report = pd.DataFrame({"Jaccard":Jaccard, "F1-score":F1_score, "LogLoss":LogLoss}, index=Algorithm)
         Report
```

Out[73]:

	Jaccard	F1-score	LogLoss
LogReg	0.800000	0.814815	0.478667
SVM	0.800000	0.814815	NaN
Decision Tree	0.846154	0.888889	NaN
KNN	0.800000	0.814815	NaN

Thank you!