IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Christopher Redden
October 18, 2022

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection thru API
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization
  - Interactive Visual Analysis with Folium
  - Machine Learning Predictions
  - Data Wrangling
  - Data Collection with Web Scraping
- Summary of all results
  - Interactive Analytics
  - Predictive Analytics
  - Exploratory Data Analytics Results

Introduction

- Project background and context
  - SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upwards 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. Training a machine learning model and using public information to predict if SpaceX will reuse the first stage will help to determine if a new company, SpaceY, can compete with SpaceX based on the price of each launch.
- Problems you want to find answers
  - What factors determine the success rate of launches and landings.
  - What conditions must exist to guarantee a successful landing.

Section 1

Methodology

Methodology

Executive Summary

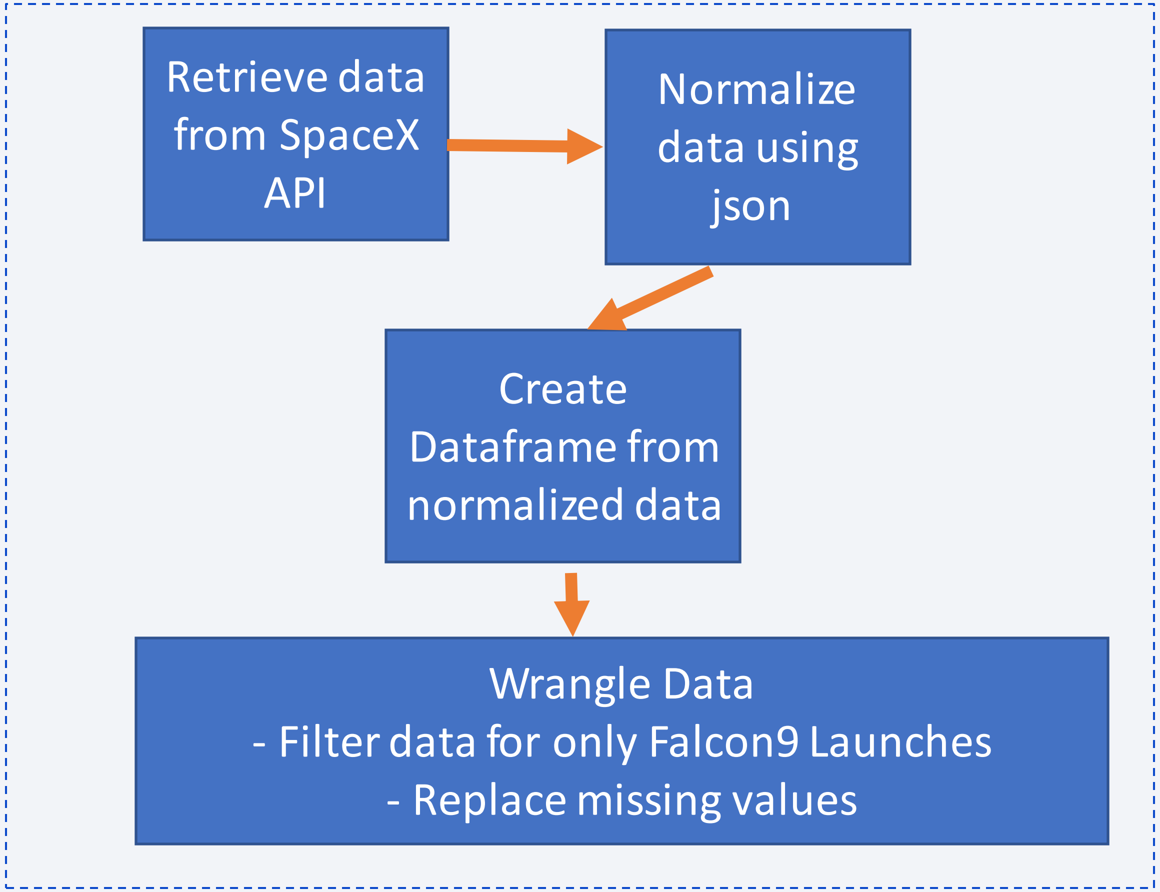
- Data collection methodology:
    - Data was collected using web scraping from Wiki and the SpaceX API
- Perform data wrangling
    - Data was wrangled using one-hot encoding applied to categories.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
    - How to build, tune, evaluate classification models

Data Collection

- Data was collected via:
  - Using "get" requests to the SpaceX API
  - Decoding responses using json functions and turning that into a normalized pandas dataframe
  - Web scraping launch records using BeautifulSoup from the Falcon9 Launch Records found on Wikipedia
  - Extracting the data as an HTML table allowed for the ability to parse the data and convert it to a pandas dataframe.

Data Collection – SpaceX API

- Using get requests to the SpaceX API, I normalized the data using json, created a datafram from the normalized data, and then did some basic data wrangling to filter down to the data required and replace missing values.
- The GitHub URL of the completed SpaceX API calls notebook can be found here: https://github.com/credden76/applied_data_science_capstone/blob/d560238445af2467bb5181917f378e485052c706/Machine%20Learning%20Prediction%20lab.ipynb



Data Collection - Scraping

- Applying web scraping against the Falcon9 Launch Wiki and using BeautifulSoup we were able to parse the data into a pandas dataframe.
- The completed web scraping notebook can be found here: https://github.com/credden76/applied_data_science_capstone/blob/9f0e2e71817a58fdd856655fe5375bfc5de0d856/Data%20Collection%20With%20Web%20Scraping.ipynb

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url).text
```

Create a `BeautifulSoup` object from the HTML `response`

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

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

```
# Use soup.title attribute
print(soup.title)
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

```
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 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
```

Data Wrangling

- Performed some Exploratory Data Analysis (EDA) to find some patterns in the data and determine what would be the label for training supervised models.
- Using Pandas: read from a CSV file, calculated the number of launches on each site, the number of occurrences of each orbit, number of occurences of mission outcome per obit type, and create a landing outcomes label from the outcome column to derive the success rate.
- The Completed data wrangling related notebooks can be found here: https://github.com/credden76/applied_data_science_capstone/blob/9f0e2e71817a58fdd856655fe5375bfc5de0d856/DataWrangling.ipynb

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

```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

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

```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
Name: Orbit, dtype: int64
```

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

```
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
Name: Outcome, dtype: int64
```

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = df['Outcome'].replace({'False Ocean': 0, 'False ASDS': 0, 'None None': 0, 'None ASDS': 0, 'False RTLS': 0, 'True ASDS': 1, 'True RTLS'
df['Outcome'] = df['Outcome'].astype(int)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 90 entries, 0 to 89
Data columns (total 17 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   FlightNumber    90 non-null     int64
 1   Date            90 non-null     object
 2   BoosterVersion  90 non-null     object
 3   PayloadMass     90 non-null     float64
 4   Orbit           90 non-null     object
 5   LaunchSite      90 non-null     object
 6   Outcome         90 non-null     int64
 7   Flights         90 non-null     int64
 8   GridFins        90 non-null     bool
 9   Reused          90 non-null     bool
 10  Legs            90 non-null     bool
 11  LandingPad      64 non-null     object
 12  Block           90 non-null     float64
 13  ReusedCount     90 non-null     int64
 14  Serial          90 non-null     object
 15  Longitude       90 non-null     float64
 16  Latitude        90 non-null     float64
dtypes: bool(3), float64(4), int64(4), object(6)
memory usage: 10.2+ KB
```

EDA with Data Visualization

- Used catplots bar charts and line charts to see the what different launch site have different success rates.
  - Catplot showing relationship between Flight Number and Launch Site
  - Catplot showing relationship between Payload and Launch Site
  - Bar Chart showing relationship between success rate and each orbit type
  - Catplot showing relationship between Flight Number and Orbit type
  - Catplot showing relationship between Payload and Orbit type
  - Line Chart showing the launch success yearly trend
- Completed EDA with data visualization notebook can be found here: https://github.com/credden76/applied_data_science_capstone/blob/9f0e2e71817a58fdd856655fe5375bfc5de0d856/EDA%20with%20Visualization%20lab.ipynb

EDA with SQL

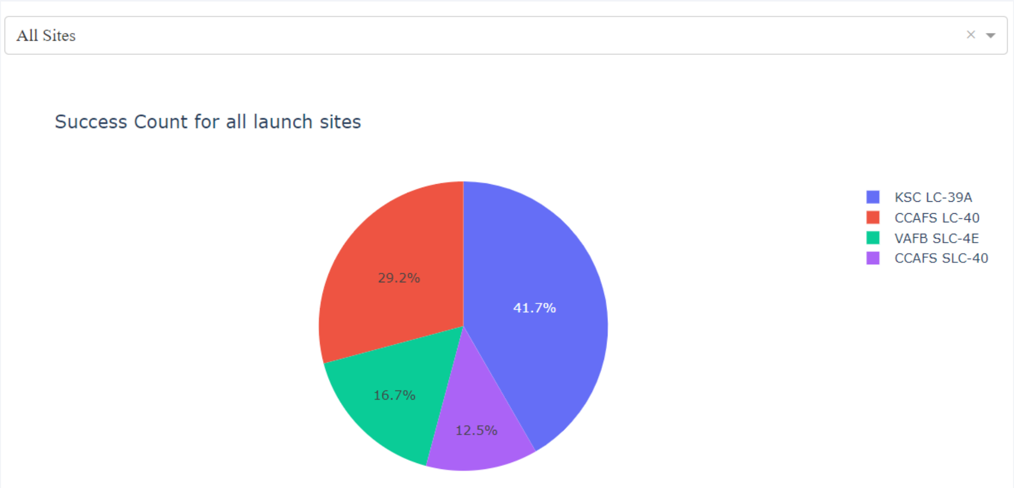
- The following SQL queries were performed to gain more insights into the data:
  - Unique Values
  - Limiting the data results and only show 5 results
  - Summing a field to show the total value.
  - Finding the average of a field
  - Finding the min of a date
  - Finding a values based on a range
  - Counting the number of times a value occurs
  - Performing a query based on a sub-query
  - Limiting data based on year and month values
  - Performing a ranking based on an order by condition.
- The completed EDA with SQL notebook can be found here: https://github.com/credden76/applied_data_science_capstone/blob/9f0e2e71817a58fdd856655fe5375bfc5de0d856/EDA%20with%20SQL%20Lab.ipynb

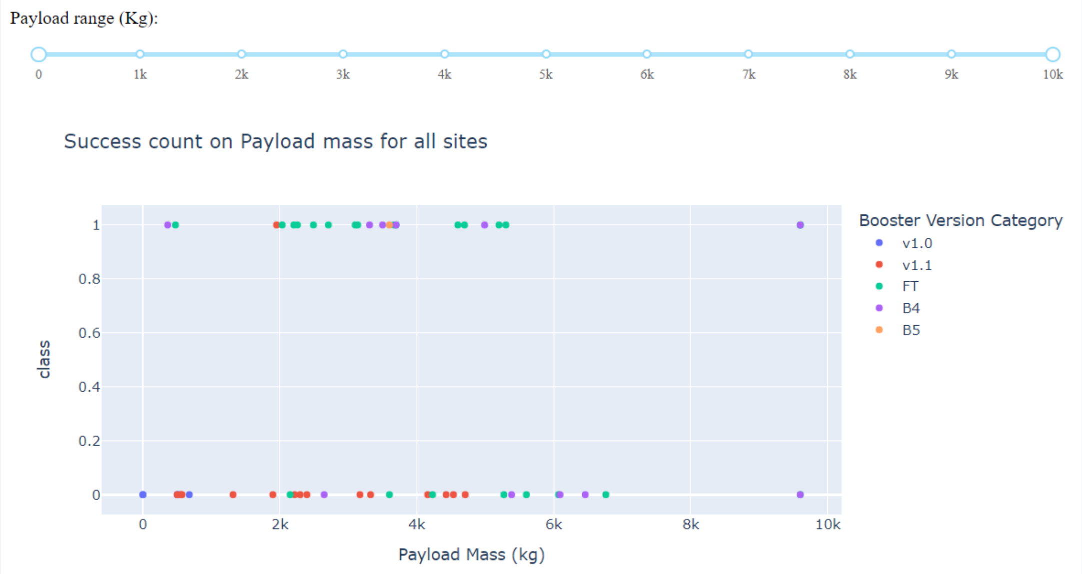
Build an Interactive Map with Folium

- I added map markers, circles, and lines to show the success or failure rates of launches for each site plotted on a folium map.
- Using color-coded labeled marker clusters, I showed which launch sites have high success rates compared to others.
- Able to show and calculate the distance between launch sites and provide additional details about those sites (i.e. proximity to railways, coastlines, distance to nearest city).
- The completed interactive map with Folium map can be found here: https://github.com/credden76/applied_data_science_capstone/blob/5db14e7a8c2de544bab1a8ca1af843e1d7176ea9/Interactive%20Visual%20Analytics%20with%20Folium%20lab.ipynb

Build a Dashboard with Plotly Dash

- Built an interactive dashboard with Plotly dash
- A Pie chart was added to show the total launches by sites
- A Scatter Plot graph was added to show the relationship between outcome and payload for the different boosters.
- The completed Plotly Dash lab can be found here: https://github.com/credden76/applied_data_science_capstone/blob/98696d2b1374b68a85d6e670778ffa475c409020/firstpython.py





Predictive Analysis (Classification)

- After loading the data, I transformed, split, trained and tested the data using numpy and pandas.
- Used an accuracy metric to model, and improve the model. Used feature engineering and algorithm tuning.
- Built machine learning models and fine-tuned the model using GridSearchCV.
- Found the most performing classification model once analysis was complete.
- The completed predictive analysis lab can be found here: https://github.com/credden76/applied_data_science_capstone/blob/98696d2b1374b68a85d6e670778ffa475c409020/Machine%20Learning%20Prediction%20lab.ipynb

Results

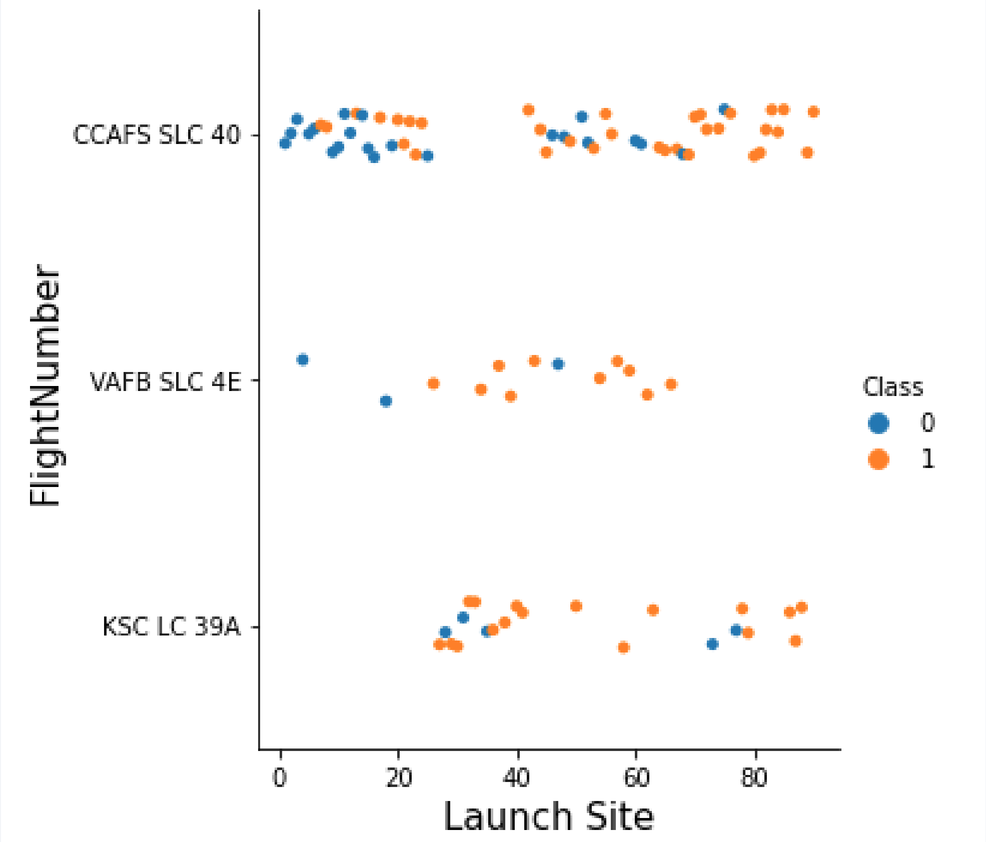
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

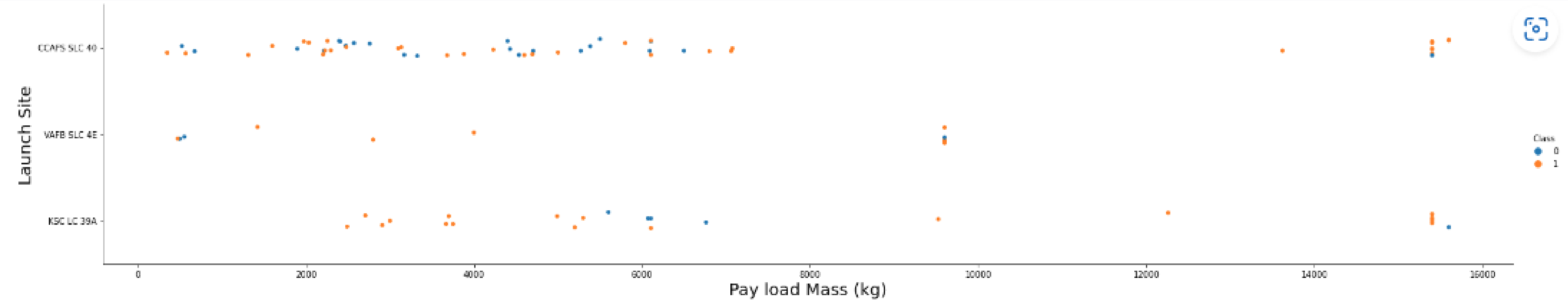
Flight Number vs. Launch Site

- From the graph shown here, the larger the amount of flights at a launch site, the greater the success rate at a launch site.



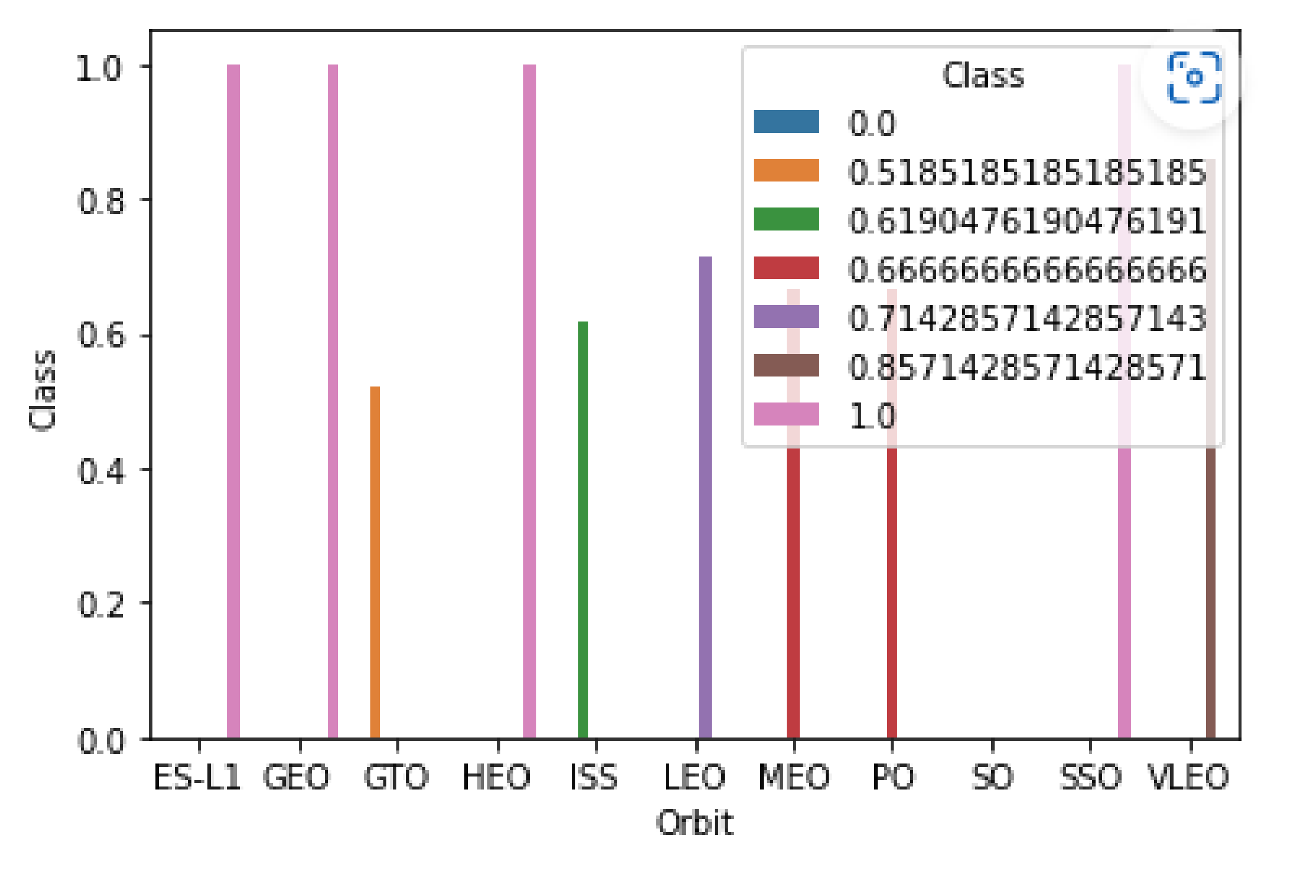
Payload vs. Launch Site

- The scatter point chart shows that the larger the payload mass, the higher success rate of the rocket.
- In the below scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).



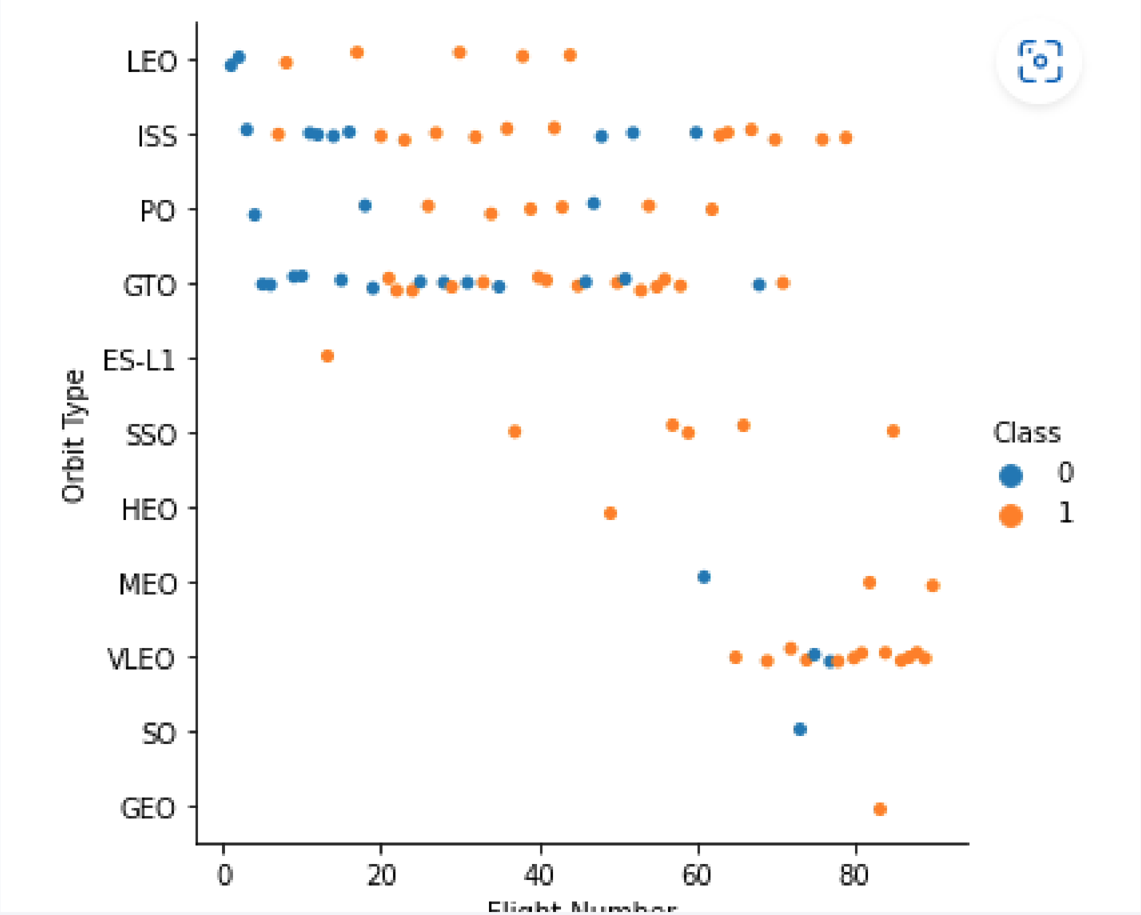
Success Rate vs. Orbit Type

- From the bar chart, we can see that ES-L1, GEO, HEO, SSO, and VLEO had the most success of orbit based on class.



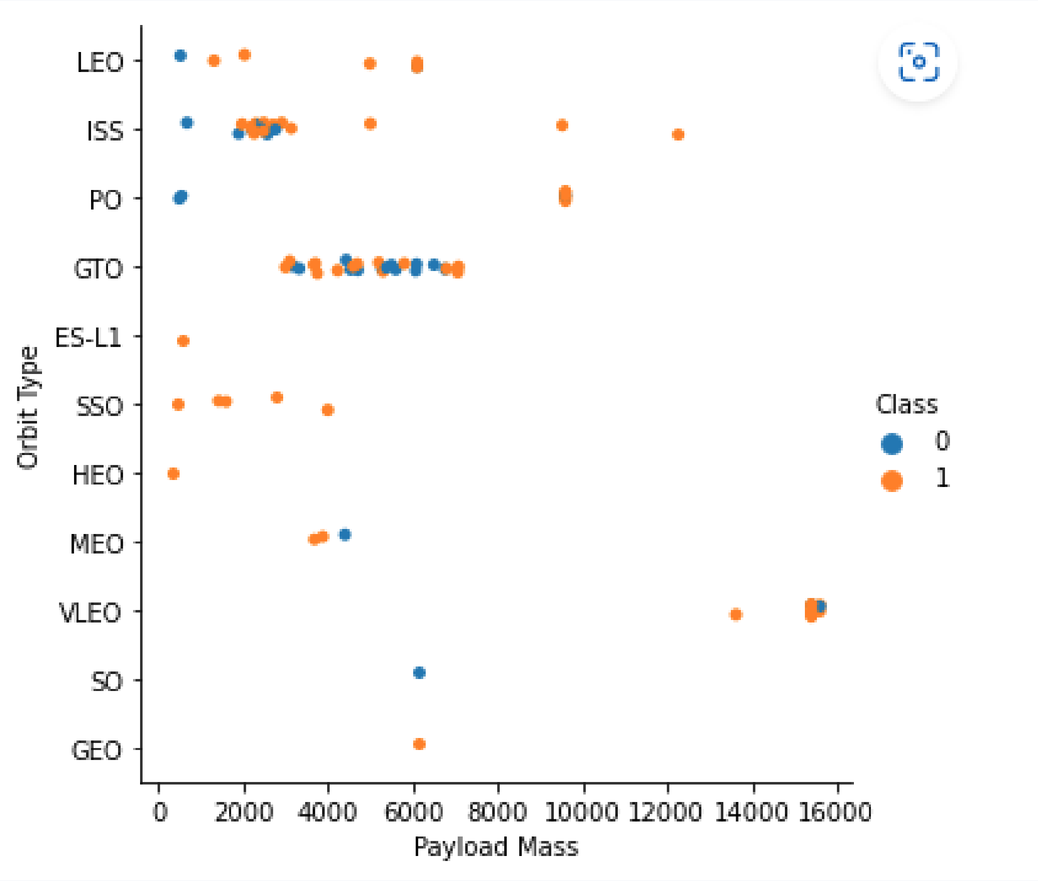
Flight Number vs. Orbit Type

- The scatter plot shown here provides the variances between Flight Number and Orbit Type. The LEO orbit shows a there is a relation between number of flights and successful orbit, wherein the GTO orbit does not.



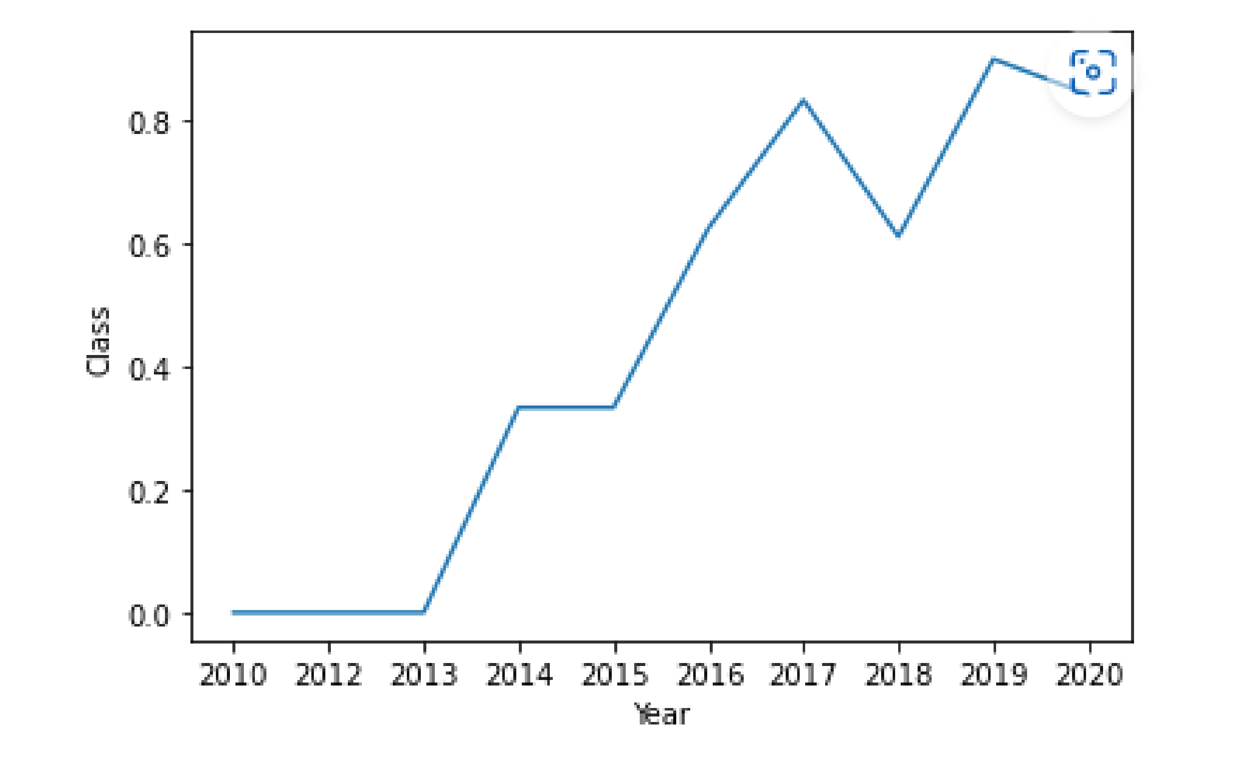
Payload vs. Orbit Type

- In this scatter plot, we can see that successful landings were more common for PO, LEO, and ISS orbits when they have heavy payloads.



Launch Success Yearly Trend

- This line graph tells us that the success rate has been increasing since 2013 with a small dip in 2018.



All Launch Site Names

- Using SQL, we can easily find the distinct launch sites from the SpaceX data.

```
%sql select Unique(LAUNCH_SITE) from SPACEXTBL;
```

```
 * ibm_db_sa://cxv62372:***@b0aebb68-94fa-46ec-a1
Done.
```

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

Launch Site Names Begin with 'CCA'

- Using SQL again against the SpaceX data, we can easily find all the Launch Sites that begin with "CCA". Here we have limited our results to the first 5 occurrences.

```
%sql SELECT * from SPACEXTBL where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5;
```

```
 * ibm_db_sa://cxv62372:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.
```

9]:

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

Total Payload Mass

- Here you can see that we can use SQL to calculate the total payload carried by boosters from NASA (CRS)

```
]: %sql SELECT SUM(CAST (PAYLOAD_MASS__KG_ AS INT)) as payloadmass from SPACEXTBL WHERE CUSTOMER = 'NASA (CRS)';

 * ibm_db_sa://cxv62372:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.c
Done.

[11]:
```

payloadmass
45596

Average Payload Mass by F9 v1.1

- Below we can see how to use SQL to calculate the average PayloadMass carried by the booster version F9 v1.1 and determine that the value is an average of 2928.

```
%sql select avg(PAYLOAD_MASS__KG_) as avg_payloadmass from SPACEXTBL WHERE Booster_Version = 'F9 v1.1';
```

```
 * ibm_db_sa://cxv62372:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.
```

14]:

avg_payloadmass
2928

First Successful Ground Landing Date

- Then again, using SQL, we can limit the data to where the Landing Outcome is "Success (ground pad)" and find the first landing date.

```
%sql SELECT MIN(DATE) as firstlanding FROM SPACEXTBL WHERE Landing__Outcome= 'Success (ground pad)';
```

```
 * ibm_db_sa://cxv62372:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdo
Done.
```

]:

firstlanding
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- Another SQL query where we can further restrict the data to a range and a landing outcome of our choosing, in this case the "Success (drone ship)".

```
: %sql select BOOSTER_VERSION from SPACEXTBL where LANDING__OUTCOME='Success (drone ship)' and PAYLOAD_MASS__KG_ BETWEEN 4000 and 6000;

 * ibm_db_sa://cxv62372:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.
```

9]:

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

Total Number of Successful and Failure Mission Outcomes

- Using SQL, we can group the mission outcomes, and count the number of times those outcomes occur to summarize failed and successful missions based on outcome.

```
%sql select MISSION_OUTCOME, count(MISSION_OUTCOME) as countmissionoutcomes from SPACEXTBL GROUP BY MISSION_OUTCOME;
```

```
 * ibm_db_sa://cxv62372:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31
Done.
```

mission_outcome	countmissionoutcomes
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

Boosters Carried Maximum Payload

- In SQL, we also used a sub-query to find the booster names that carried the maximum payloads.

```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL where PAYLOAD_MASS__KG_=(select MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
```

```
 * ibm_db_sa://cxv62372:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud
Done.
```

2]:

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

2015 Launch Records

- Here we are using SQL to list the 2015 failed landing outcomes in drone ship and the booster versions that failed.

```
: %sql SELECT MISSION_OUTCOME,BOOSTER_VERSION,LAUNCH_SITE FROM SPACEXTBL where Landing__Outcome = 'Failure (drone ship)' and EXTRACT(YEAR FROM DATE)='2015';

   * ibm_db_sa://cxv62372:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
  Done.
```

27]:

mission_outcome	booster_version	launch_site
Success	F9 v1.1 B1012	CCAFS LC-40
Success	F9 v1.1 B1015	CCAFS LC-40

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

- Using SQL, we ranked the landing outcomes between 2010-06-04 and 2017-03-20 in descending order of the count of the landing outcomes.

```
%sql SELECT LANDING__OUTCOME, COUNT(Landing__Outcome) as CountLandingOUtcome FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY Landing__Outcome ORDER BY COUNT(Landing__Outcome) DESC;
```

```
 * ibm_db_sa://cxv62372:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.
```

1]:

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

Section 3

Launch Sites Proximities Analysis

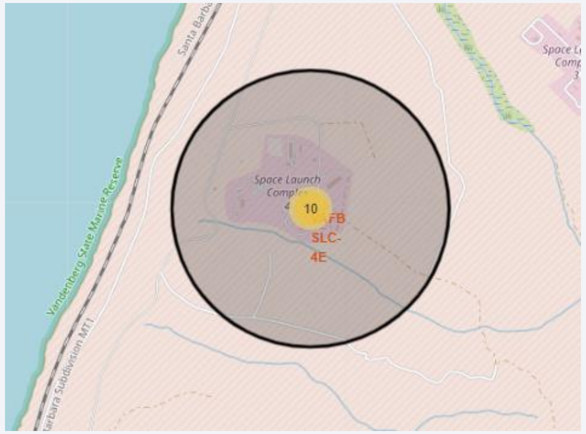
All launch sites global map markers.

All SpaceX launch sites are located in the United States of America and reside in California and Florida.

Markers showing launch sites with color lables

- The circles represent the total launches by launch sites.

California



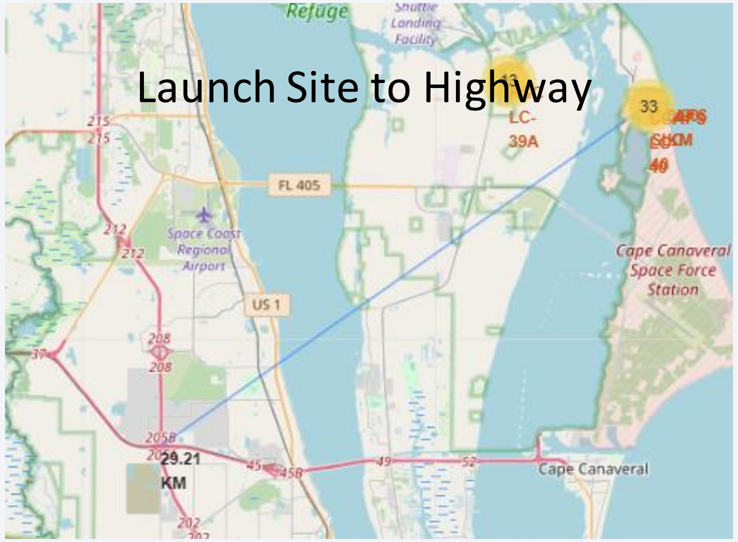
Florida

Launch Site Distance to landmarks

Significant Findings
- Launch sites are not in close proximity to highways

- Launch sites are in close proximity to coastlines

- Launch sites are not close to cities

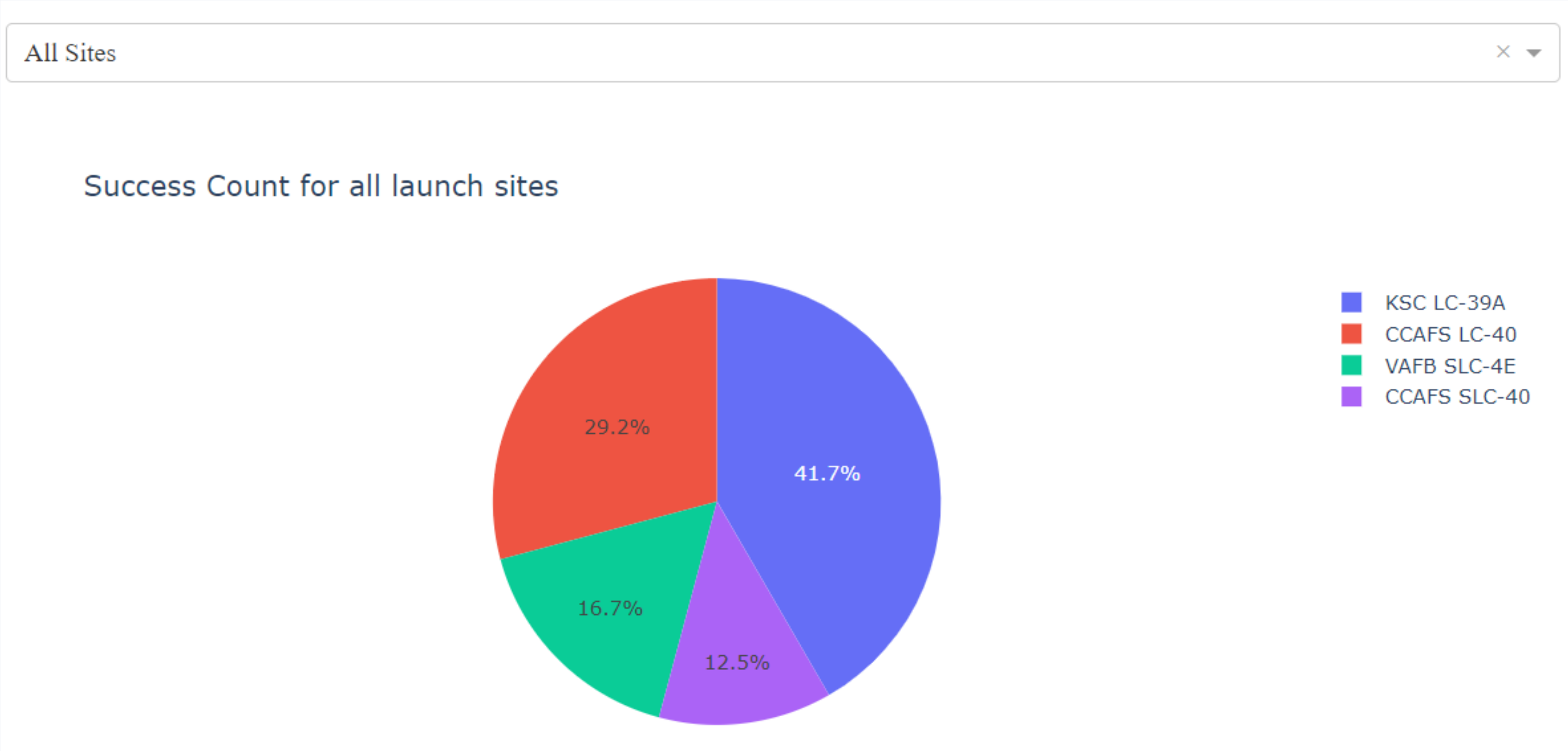


Section 4

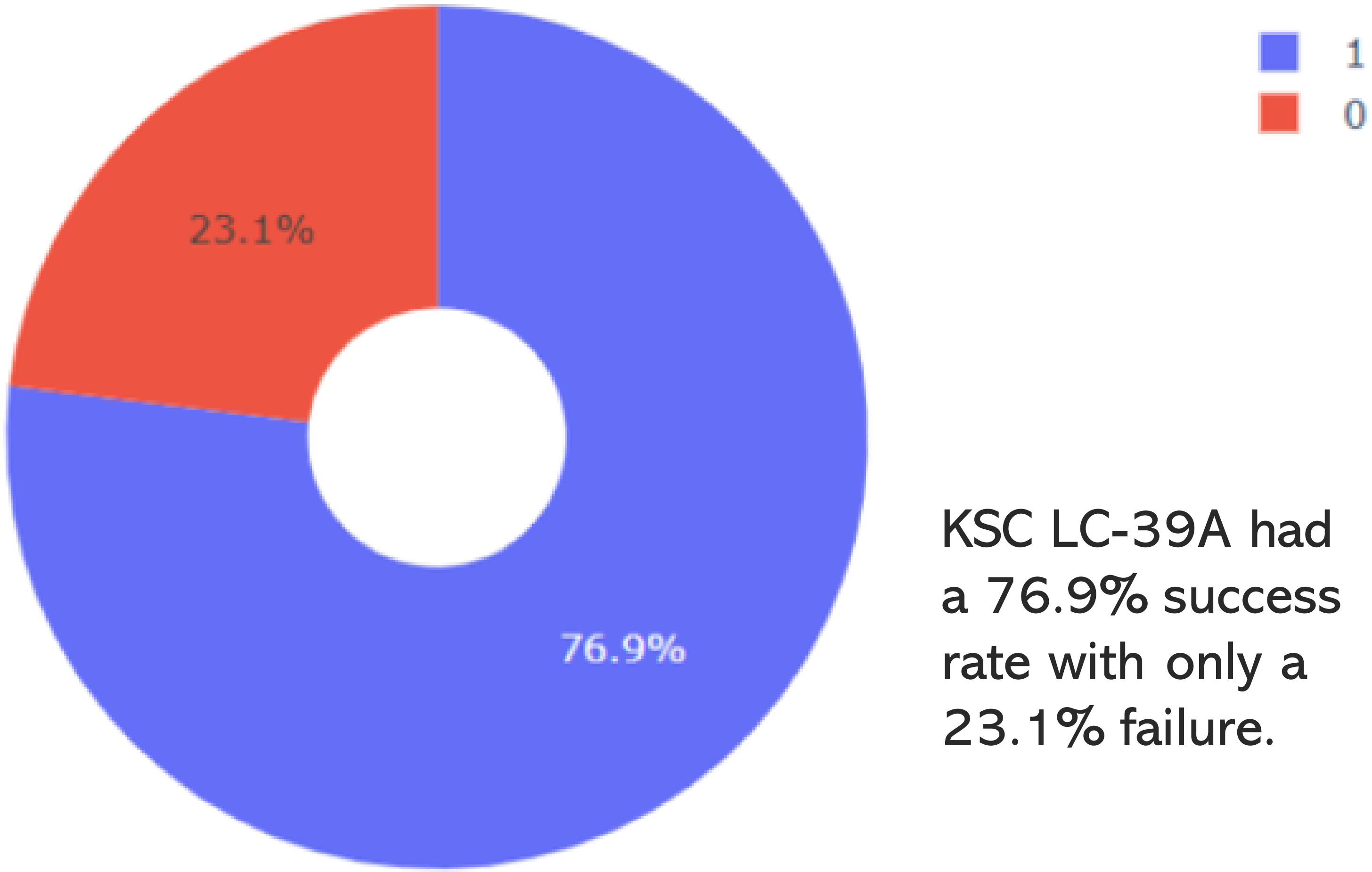
Build a Dashboard with Plotly Dash

Pie Chart Showing the Success Percentage Achieved by Each Launch Site

- The KSC LC-39A Launch Site had the most successful launches in all sites.
- The CCAFS SLC-40 Launch Site had the least successful launches in all sites.



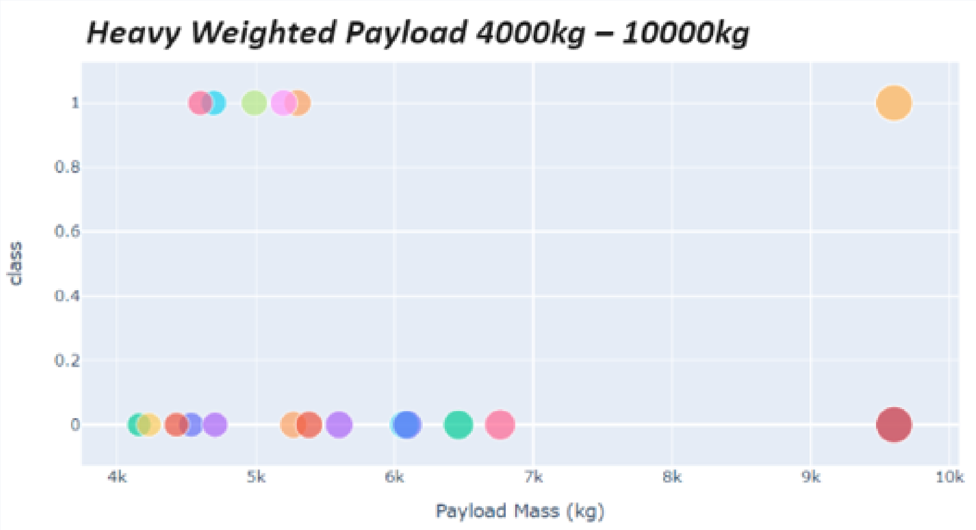
Pie Chart Showing the Launch Site With the Highest Launch Success Ratio

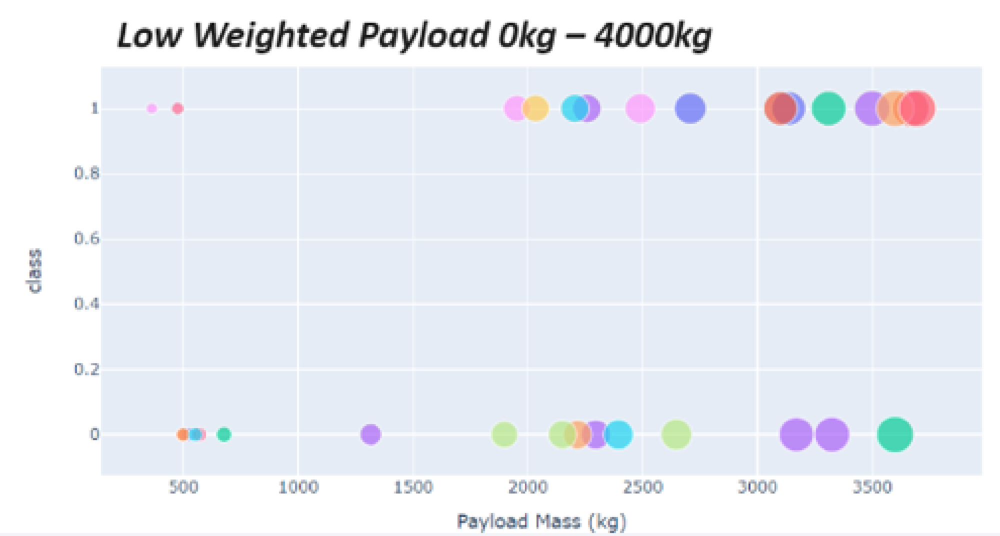


KSC LC-39A had a 76.9% success rate with only a 23.1% failure.

Scatter Plat of Payload vs. Launch Outcome for all sites.

Using different ranges:





The success rates for Heavy Payloads is lower than the Low Weighted Payloads

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- Decision Tree is the model with the best accuracy score.

Find the method performs best:

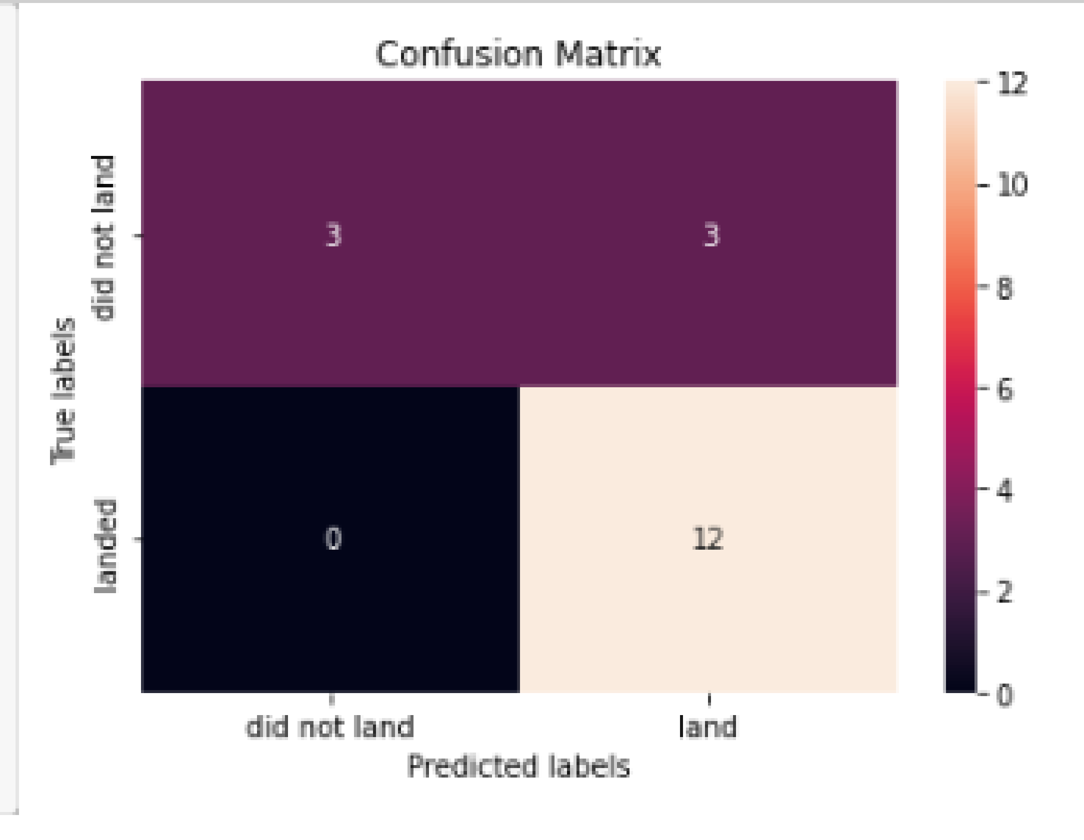
```
models = {'KNeighbors':knn_cv.best_score_,
          'DecisionTree':tree_cv.best_score_,
          'LogisticRegression':logreg_cv.best_score_,
          'SupportVector': svm_cv.best_score_}

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'}
```

Confusion Matrix

- The confusion matrix of the best performing model is the decision tree classifier. This shows that the classifier can distinguish different classes. False positives are still a factor.



Conclusions

- Launch rate success starts to increase in 2013 thru 2020.
- KSC LC-39A has had the most successful launches of any sites.
- ES-L1, GEO, SSO, HEO, VLEO have had the best success rate.
- The bigger the flight amount at the launch site, the better the rate of success at the launch site.
- The decision tree classifier performs the best when completing machine learning algorithms for this task.

Thank you!