Winning Space Race with Data Science

Dylan Kruger
05/02/22

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection through API
  - Data Collection through Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis (EDA) with SQL
  - EDA with Data Visualization
  - Interactive Visual Analytics through Folium
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis
  - Interactive Analytics Screenshots
  - Predictive Analytics Results

Introduction

- Project background and context

With the advent of commercial space travel SpaceX has proven to be the most successful of the early startups. This is due to the relative cheapness of the trip with the low low price of 62 million dollars per trip while the competitors can cost upwards of 165 million dollars. The disparity is caused because SpaceX can reuse the first stage of the rocket.

This gives us the goal of accurately predicting the ability to have the first stage of the rocket land safely, this allows us to predict the cost of the launch. This information can be used by alternate companies in their bid against SpaceX.

- Problems you want to find answers

What factors correlate between rocket variables and successful landings.

What conditions need to be in place to ensure a successful landing of the the first stage of the rocket.

Section 1

Methodology

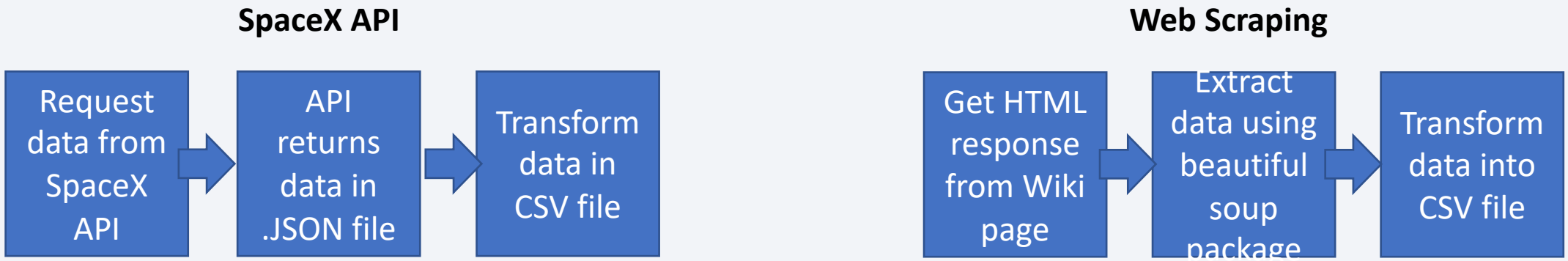
Methodology

Executive Summary

- Data collection methodology:
  - Data was collected with a combination of SpaceX API & web scraping from Wikipedia.
- Perform data wrangling
  - One-hot encoding was applied to the categorical features.
- 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 Collection types:
- API requests from the SpaceX API
- Web scraping from the SpaceX page
- Web scraping from the Falcon 9 page
- Web scraping from the Falcon Heavy Launches Records page



Data Collection – SpaceX API

- Requesting rocket data from SpaceX API
- Convert Response to JSON file
- Functions to clean data
- Filtering dataframe and exporting to CSV

```
In [6]:  spacex_url="https://api.spacexdata.com/v4/launches/past"

In [7]:  response = requests.get(spacex_url)
```

```
In [15]: # Use json_normalize meethod to convert the json result into a da
         data = pd.json_normalize(response.json())
```

```
In [22]: # Call getLaunchSite
         getLaunchSite(data)

In [23]: # Call getPayloadData
         getPayloadData(data)

In [24]: # Call getCoreData
         getCoreData(data)
```

```
data_falcon9 = df[df['BoosterVersion']!='Falcon 1']
```

```
data_falcon9.to_csv('dataset_part\_1.csv', index=False)
```

- Github URL: https://github.com/dkruger77/Capstone_project.md/blob/main/Data%20Collection.ipynb

Data Collection - Scraping

- Using BeautifulSoup we applied web scrapping to the Falcon 9 records
- Parsed the table and converted to a pandas dataframe

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

```
In [4]: static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

```
In [5]: # use requests.get() method with the provided static_url
        # assign the response to a object
        html_data = requests.get(static_url)
        html_data.status_code
Out[5]: 200
```

2. Create a BeautifulSoup object from the HTML response

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

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

```
In [7]: # Use soup.title attribute
        soup.title
Out[7]: <title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

3. Extract all column names from the HTML table header

```
In [10]: column_names = []

         # Apply find_all() function with `th` element on first_launch_table
         # Iterate each th element and apply the provided extract_column_from_header() to get a column name
         # Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names

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

4. Create a dataframe by parsing the launch HTML tables
5. Export data to csv

Data Wrangling

- Data was processed based on the number of launches at each site

```
df["LaunchSite"].value_counts()
```

- Calculating the value of each orbit

```
df["Orbit"].value_counts()
```

- Calculating the numbers of outcome based on orbit type

```
landing_outcomes = df["Outcome"].value_counts()
landing_outcomes
```

- Calculating success rate for landing

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

```
0.6666666666666666
```

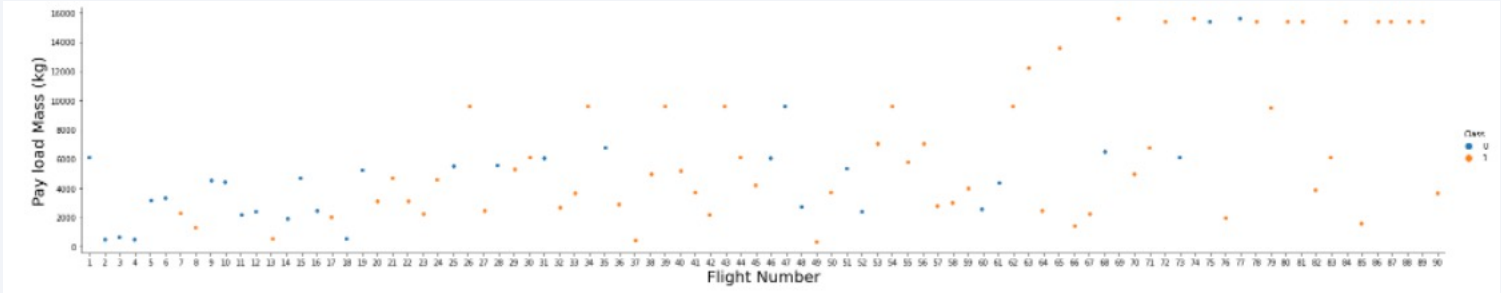
- Exporting to CSV

```
df.to_csv("dataset_part\_2.csv", index=False)
```

https://github.com/dkruger77/Capstone_project.md/blob/main/Data%20Wrangling.ipynb

EDA with Data Visualization

- Scatter Chart
  - *Flight number vs. Launch site*
  - *Payload vs. Launch site*
  - *Flight number vs. Orbit type*
  - *Payload vs. Orbit type*



  - The scatter plot shows the relationship between variables and helps to identify if correlations exist.
- Bar Chart
  - *Orbit Type vs. Success Rate*
  - Bar chart was used to quickly compart multiple data sets immediately. The pro of the bar chart is to quickly compare the relationship of the two points expressed by the axes.
- Line Chart
  - *Year vs. Success Rate*
  - The line chart was integrated to show trends in the data clearly while predicting future data.

https://github.com/dkruger77/Capstone_project.md/blob/main/EDA%20with%20Visualization%20.ipynb

EDA with SQL

- The dataset was loaded using the Db2 database, and SQL was applied to find the answers to:
  - The names of unique launch sites in the missions
  - The total payload measured in mass carried by the boosters
  - The date of the first successful landing outcome on a ground pad
  - The average payload measured in mass carried by the F9 booster
  - The number of mission outcomes based on success and failure
  - Listing the failed outcomes of drone ships/booster versions/launch site

https://github.com/dkruger77/Capstone_project.md/blob/main/EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

- We marked all launch sites, added map objects (markers, circles, lines that mark success/failure)
- We assigned features to launch outcomes 0 for failure 1 for success.
- We used color-labeled marker clusters to identify launch sites with the highest success rate
- We calculated the distance between launch sites to railways, highways, and coastlines.

https://github.com/dkruger77/Capstone_project.md/blob/main/Visual%20with%20Folium%20.ipynb

Build a Dashboard with Plotly Dash

- We created a dashboard application that is expressed using a pie chart and scatter point chart
  - Pie chart
    - Shows total success launches organized by sites
    - Can show either the success rate across all sites or by individual site
  - Scatter Point
    - Organized by booster this graph shows the relationship between outcomes and payload mass
    - Graded between the scale of 0-10000 kg
    - Shows the relationship between launch point, payload mass, and booster type

https://github.com/dkruger77/Capstone_project.md/blob/main/Plotly%20dash

Predictive Analysis (Classification)

- The data was loaded, transformed, and split into training and testing sets using the NumPy and pandas libraries.
- The machine learning models were built and tuned using the GridSearchCV
- The model was built to use accuracy as the main metric, as we attempted to improve the model using algorithmic tuning and feature engineering
- The ideal method was selected based on the performance based on the logistic regression and classification trees

https://github.com/dkruger77/Capstone_project.md/blob/main/Machine%20Learning%20Prediction.ipynb

Results

- Exploratory data analysis results
  - The screenshot to the right expresses the highest accuracy off all the training methodologies tested with a yield of .889

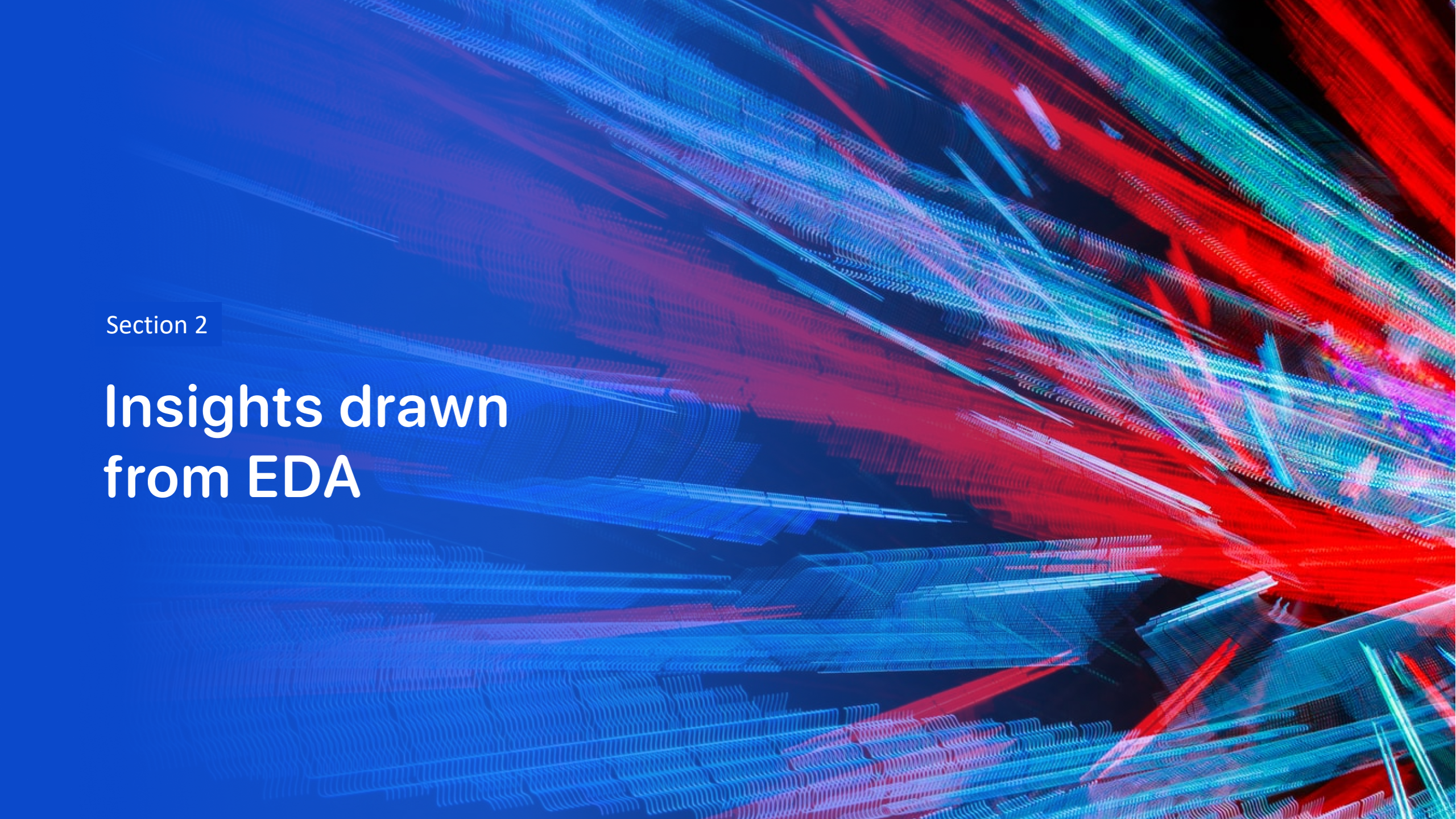
```
parameters = {'criterion': ['gini', 'entropy'],
     'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],
     'max_features': ['auto', 'sqrt'],
     'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10]}

tree = DecisionTreeClassifier()
```

```
tree_cv = GridSearchCV(tree,parameters,cv=10)
tree_cv.fit(X_train, Y_train)
```

```
GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 1
8],
                         'max_features': ['auto', 'sqrt'],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10],
                         'splitter': ['best', 'random']})
```

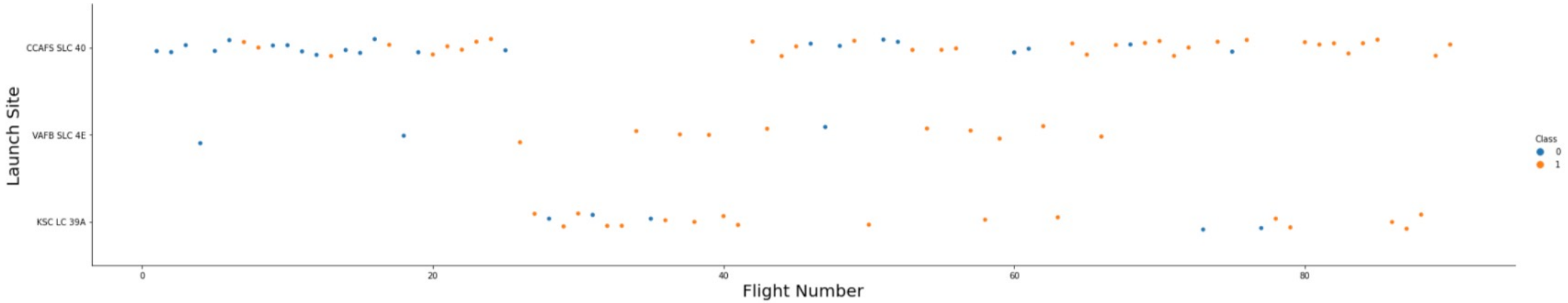
```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params
print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_
depth': 12, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_sampl
es_split': 5, 'splitter': 'random'}
accuracy : 0.8892857142857145
```


Section 2
Insights drawn from EDA

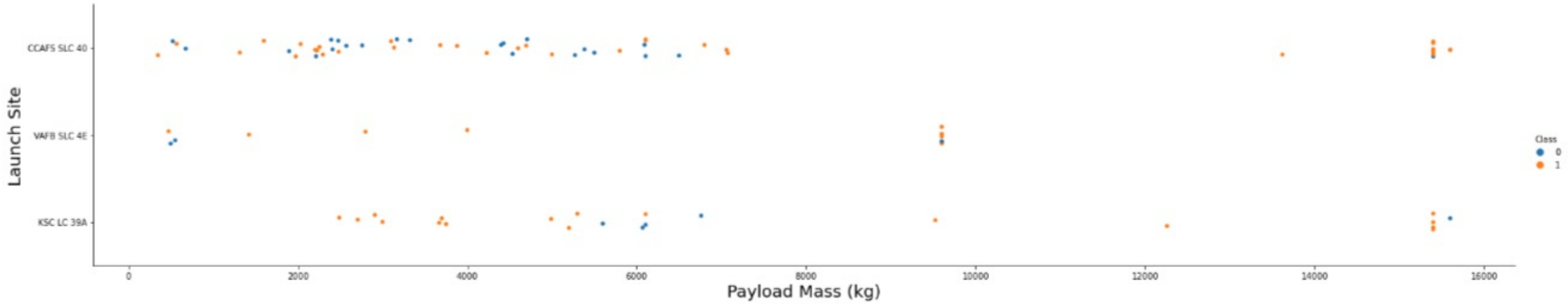
Flight Number vs. Launch Site

- The blue dots of the graph represent launch failures, while orange dots show successful launches.
- Success rate per site seems to be going up as the number of flights increases.



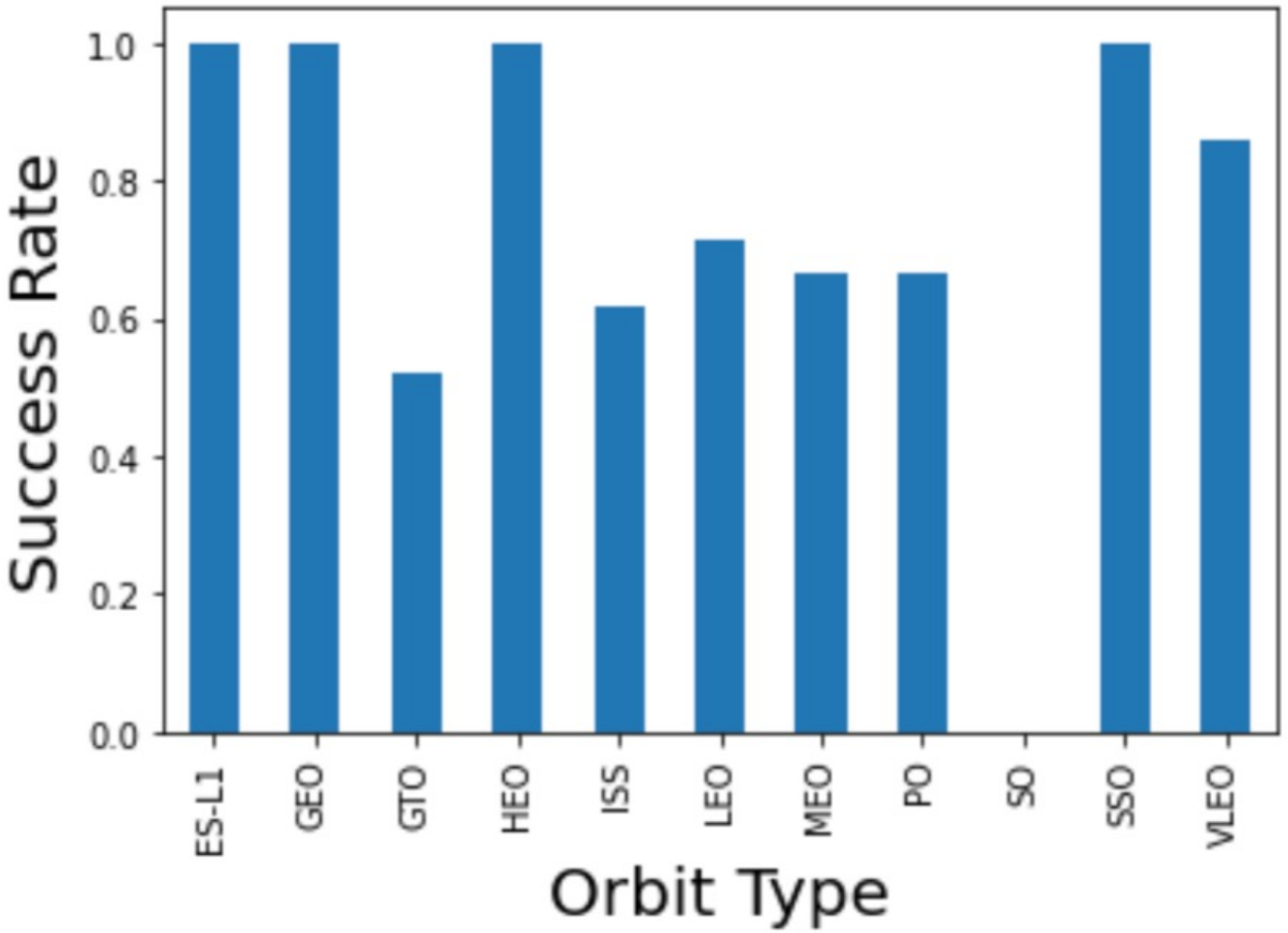
Payload vs. Launch Site

- Again, blue represents failed launches, while orange represent successful launches.
- The greater the payload mass the higher the success rate seems to follow for the rocket.



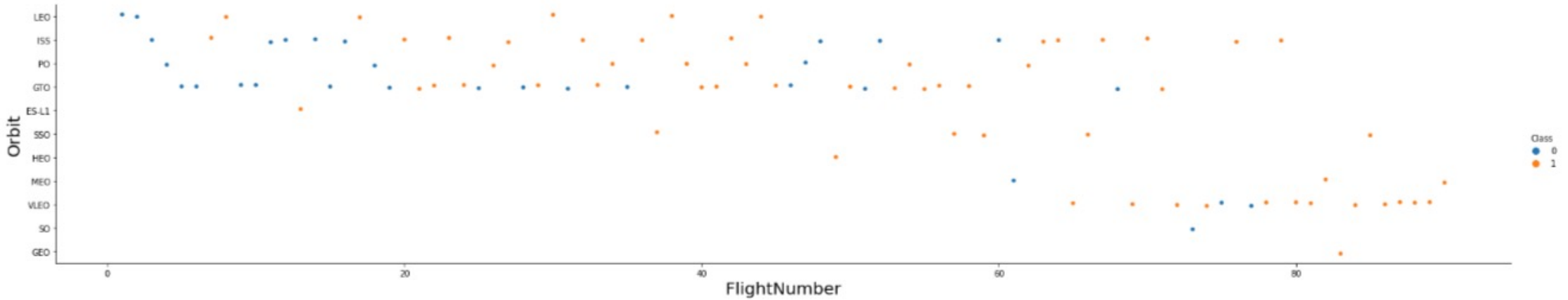
Success Rate vs. Orbit Type

- The highest success rates are related to the SSO, HEO, GEO, and ES-L1 orbit types. (100%)
- Conversely, the GTO orbit type only caries about a 50% success rate.



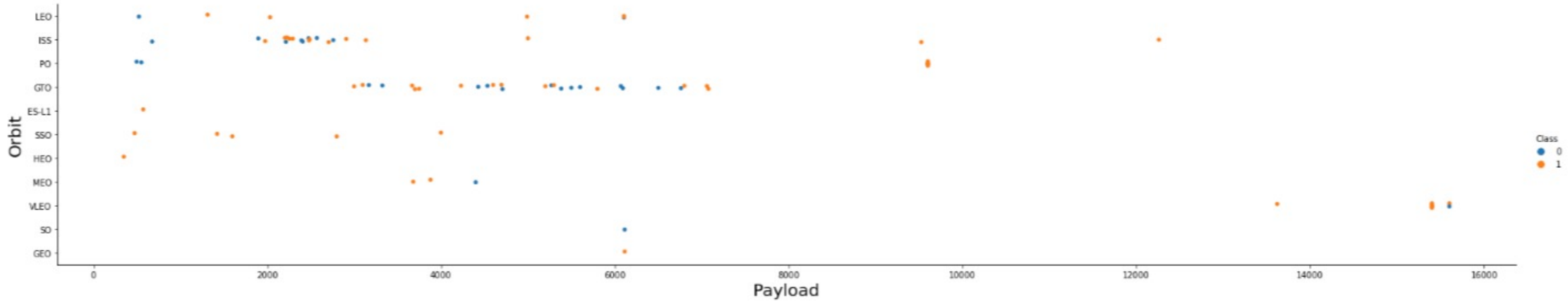
Flight Number vs. Orbit Type

- Blue is still represented by unsuccessful launches, orange is success
- Seems to be no relationship in the GTO orbits
- VLEO on the other hand seems to have a high success rate that is associated with the recent launch dates



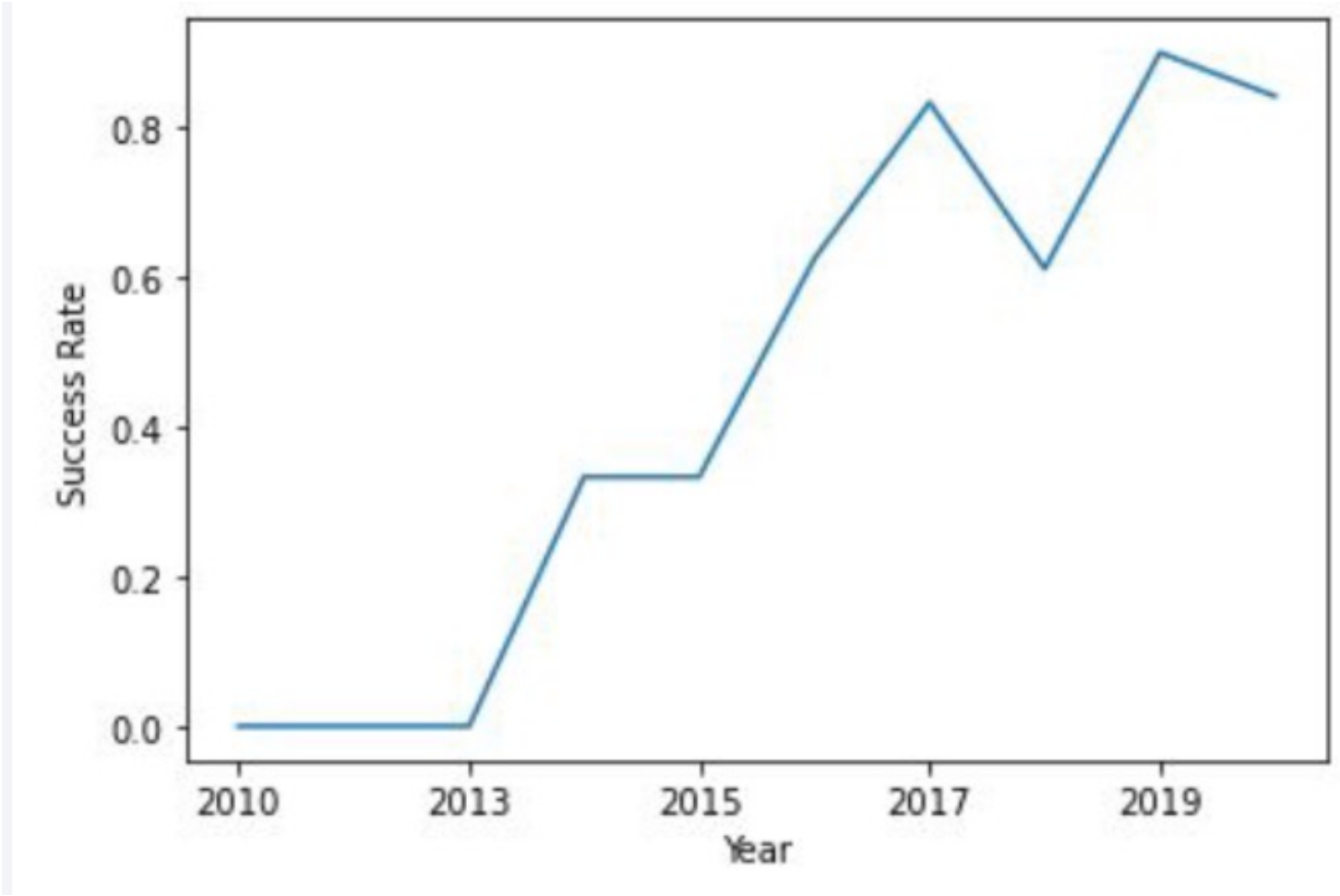
Payload vs. Orbit Type

- Blue is still represented by the unsuccessful launches, orange is success
- Overall, mass seems to have a positive correlation in terms of launch success
- However, it is not possible to distinguish in the case of GTO orbit as they as so closely graphed



Launch Success Yearly Trend

- Despite a dip in success rate in 2018 the rate of success has steadily increased from the year 2013
- Peak success is graphed around 80%



All Launch Site Names

- The 'Select 'Distinct' function was used to only show distinct terms from the database
- The four unique launch sites are:
  - CCAFS LC-40
  - CCAFS SLC-40
  - KSC LC-39A
  - VAFD SLC-4E

```
%%sql
SELECT DISTINCT LAUNCH_SITE
FROM SPACEXTBL;
```

```
 * ibm_db_sa://rqv68474:***@2f
0nqnrk39u98g.databases.appdoma
Done.
```

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

Launch Site Names Begin with 'CCA'

- Unlike the previous slide this query was used to find all launch sites that began with ‘CCA’
- This was accomplished using the ‘LIKE’ operator and (%) following the term

```
%%sql
SELECT LAUNCH_SITE
FROM SPACEXTBL
WHERE LAUNCH_SITE LIKE 'CCA%'
LIMIT 5;
```

```
 * ibm_db_sa://rqv68474:***@2f3
0nqnrk39u98g.databases.appdomai
Done.
```

launch_site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Total Payload Mass

- Using the following query, we calculated the total payload mass of the boosters from NASA as specified using the ‘Where’ clause to match customer types
- The result was found to be 45596 KG using the Sum() function to find the total

```
%%sql
SELECT SUM(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE Customer = 'NASA (CRS)';
```

```
 * ibm_db_sa://rqv68474:***@2f32
0nqnrk39u98g.databases.appdomain
Done.
```

1
45596

Average Payload Mass by F9 v1.1

- The following query calculates the average payload mass carried by the F9 v1.1 type booster
- This was matched using the ‘Where’ and ‘Like’ clause

```
%%sql
SELECT AVG(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE Booster_Version LIKE 'F9 v1.0%';
```

```
 * ibm_db_sa://rqv68474:***@2f3279a5-73c
0nqnrk39u98g.databases.appdomain.cloud:3
Done.
```

1
340

First Successful Ground Landing Date

- This query shows the First Successful ground landing
- Utilizing the Min() operation will locate the earliest available date in this type of dataset

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

```
 * ibm_db_sa://rqv68474:***@2f3279a5-73d1-4859-88f
0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

1
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- The following query was filtered using the 'Where' clause to narrow the field to only the successful drone landings with payload variables that fall within our selected parameters.
- The 'AND' operator is used to make sure the value is between the parameters set

```
%%sql
SELECT BOOSTER_VERSION
FROM SPACEXTBL
WHERE LANDING__OUTCOME = 'Success (drone ship)'
    AND 4000 < PAYLOAD_MASS__KG_ < 6000;
```

```
 * ibm_db_sa://rqv68474:***@2f3279a5-73d1-4859-88
0nqnrk39u98g.databases.appdomain.cloud:30756/bluc
Done.
```

booster_version
F9 FT B1021.1
F9 FT B1023.1
F9 FT B1029.2
F9 FT B1038.1
F9 B4 B1042.1
F9 B4 B1045.1
F9 B5 B1046.1

Total Number of Successful and Failure Mission Outcomes

- Grouping the data by mission outcome is a quick way of separating the data, this is done using the 'Group By' statement
- As shown by the data nearly 99% of all missions have a successful outcome

```
%%sql
SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER
FROM SPACEXTBL
GROUP BY MISSION_OUTCOME;
```

```
 * ibm_db_sa://rqv68474:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907
0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

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

Boosters Carried Maximum Payload

- Using the ‘Select Distinct’ statement we can select all the unique booster versions that has carried the maximum payload mass (shown by the ’Max()’ query)

```
%%sql
SELECT DISTINCT BOOSTER_VERSION
FROM SPACEXTBL
WHERE PAYLOAD_MASS__KG_ = (
    SELECT MAX(PAYLOAD_MASS__KG_)
    FROM SPACEXTBL);
```

```
 * ibm_db_sa://rqv68474:***@2f3279a
0nqnrk39u98g.databases.appdomain.cl
Done.
```

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

2015 Launch Records

- The following query shows all failed drone ship missions categorized by outcome, booster type, and launch site
- Using the ‘Where’ clause limits the search to only the failures by drone type ships

```
%%sql
SELECT LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE
FROM SPACEXTBL
WHERE Landing__Outcome = 'Failure (drone ship)'
    AND YEAR(DATE) = 2015;
```

```
 * ibm_db_sa://rqv68474:***@2f3279a5-73d1-4859-88f0-a6c
0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

- With the ‘Where’ clause we can sort the result using the data parameters that we want
- This is organized using the ‘Order By’ keyword that is showing the data with a descending order using the ‘DESC’ keyword

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

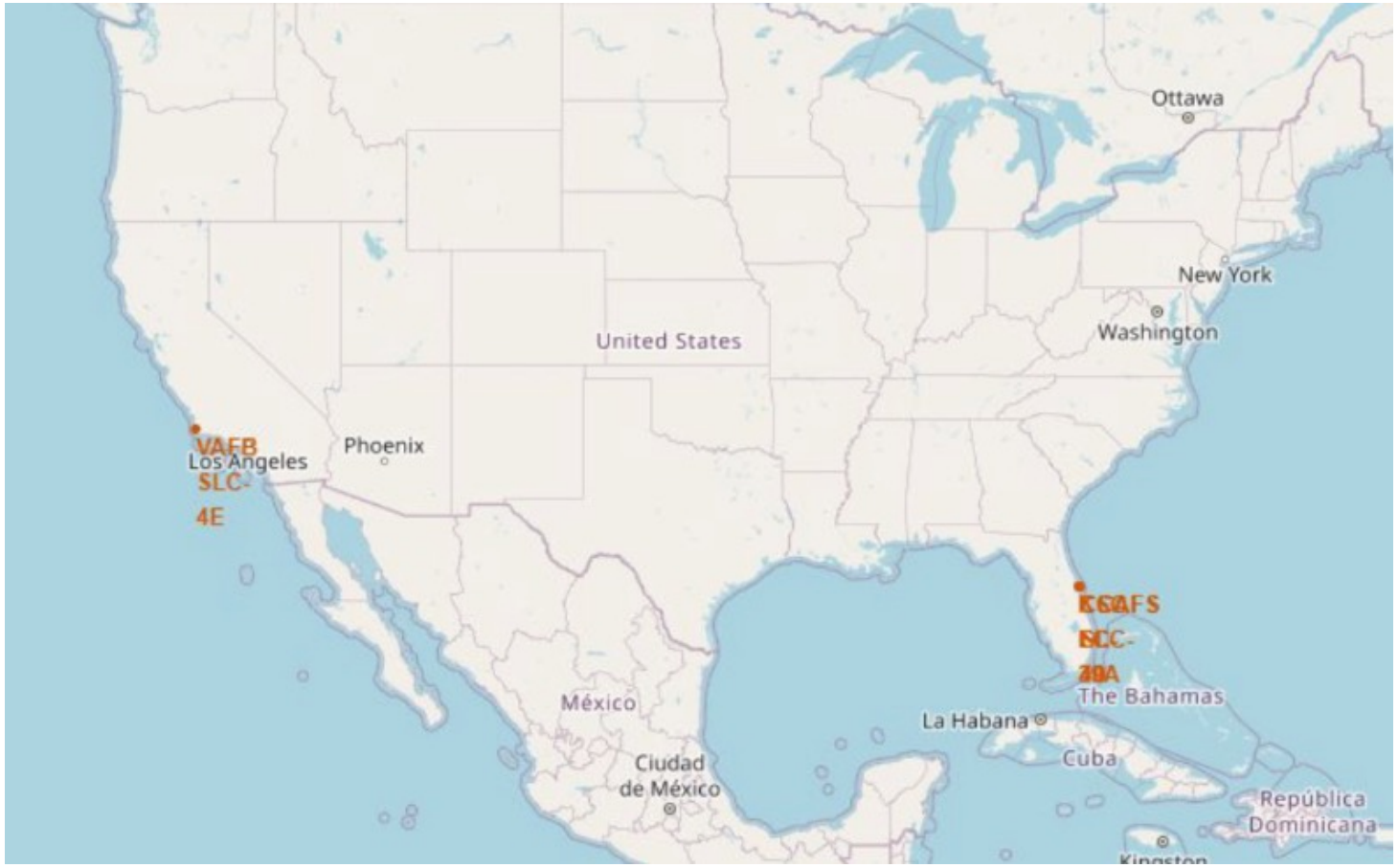
```
 * ibm_db_sa://rqv68474:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c
0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

landing__outcome	total_number
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

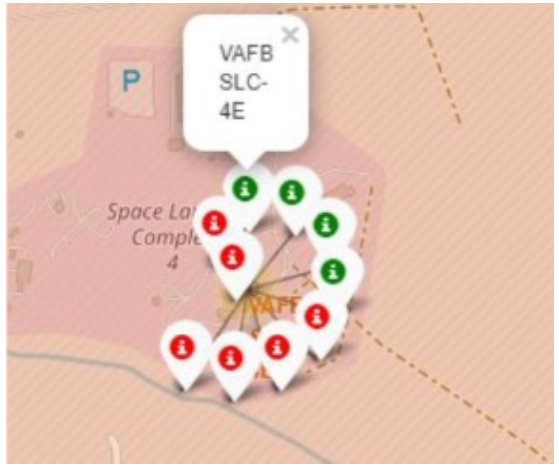
All Launch Site Locations

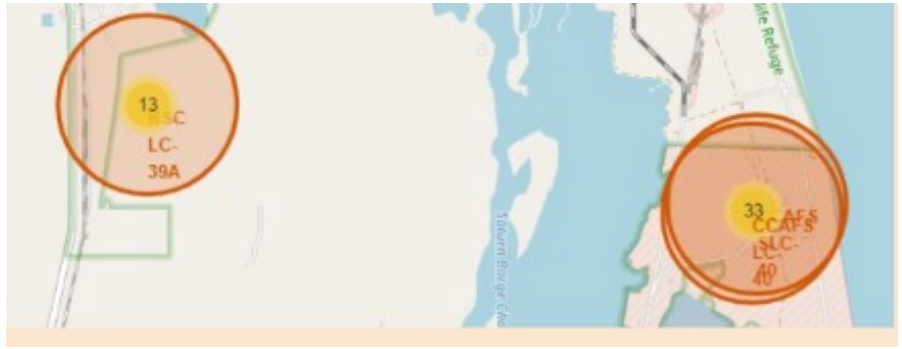
- This map shows that all SpaceX launch sites are in the United States
- All sites are located near the coastline (California and Florida)

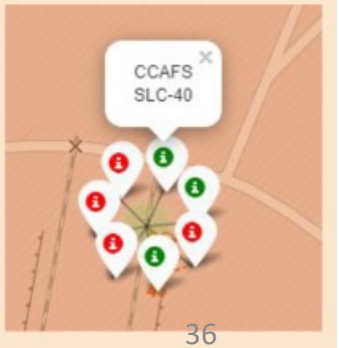


Launch Outcomes – Color Coded

- The topmost graph shows the California Launch site
- The other graphs show the Florida launch sites
- All successful landings are shown with a green marker while all failed landing are represented with a red marker

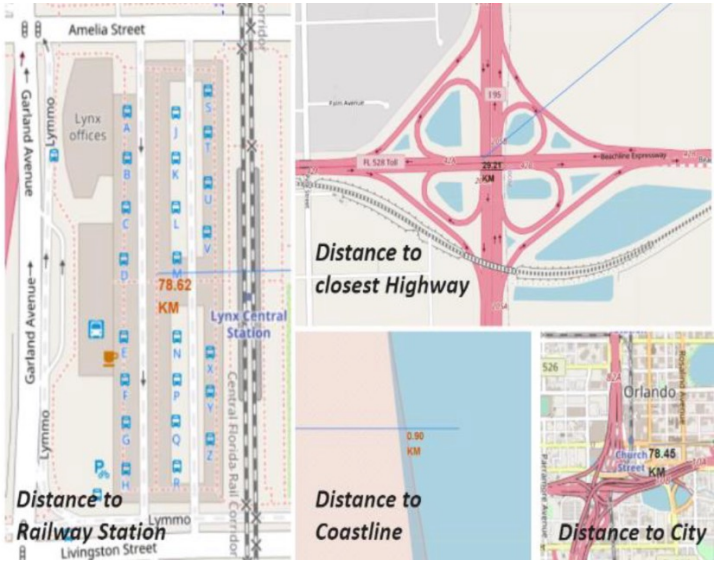


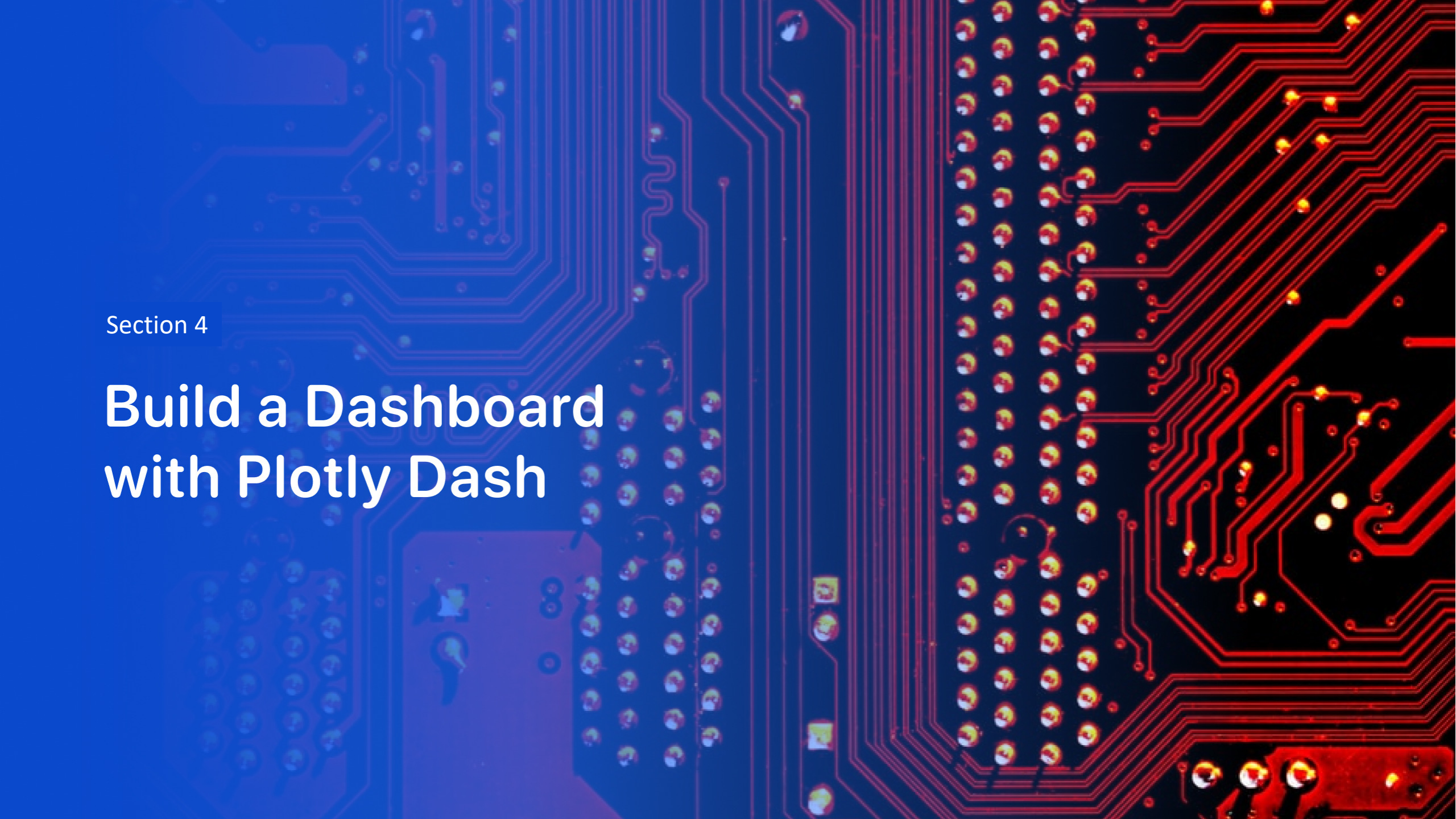




Launch Site Proximity to Landmarks

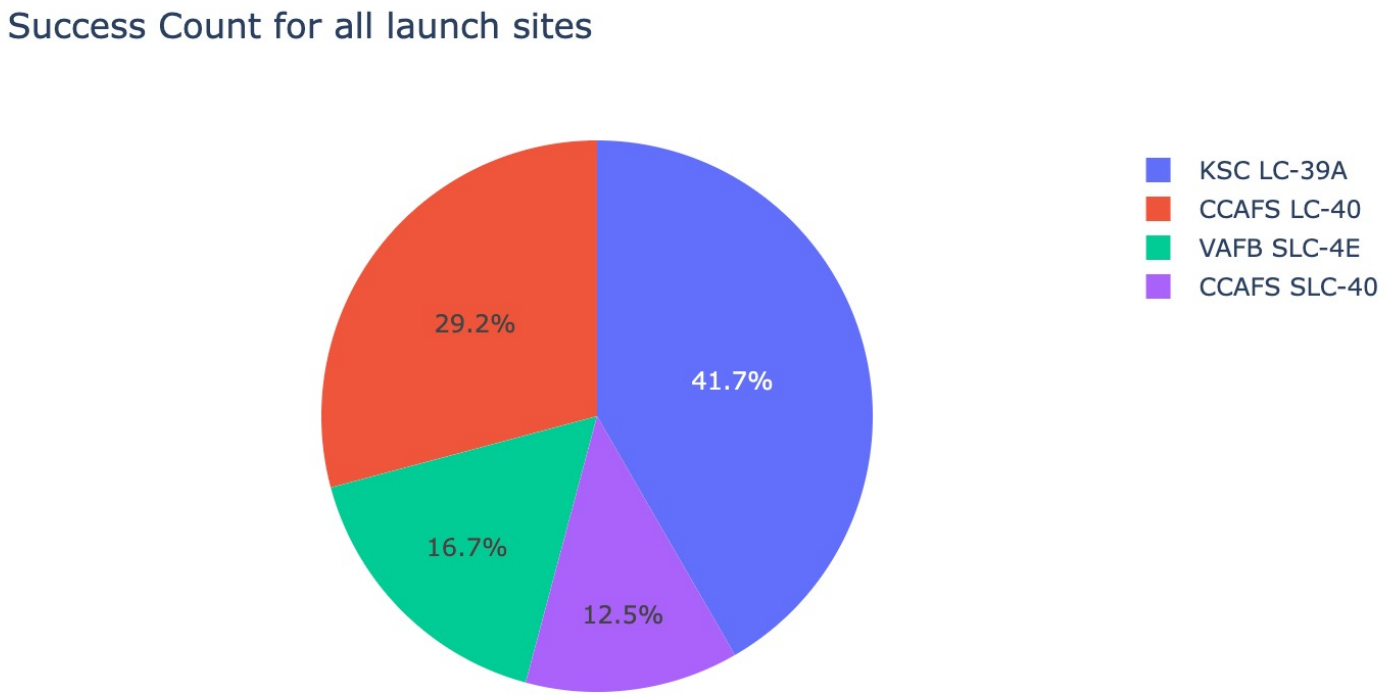
- The ideal proximity for launch sites seems to be close to railways and highways to optimize transportation and equipment deliveries.
- Close to the coastline and decently located from major cities to minimize the threat levels that a launch could pose.




Section 4
Build a Dashboard with Plotly Dash

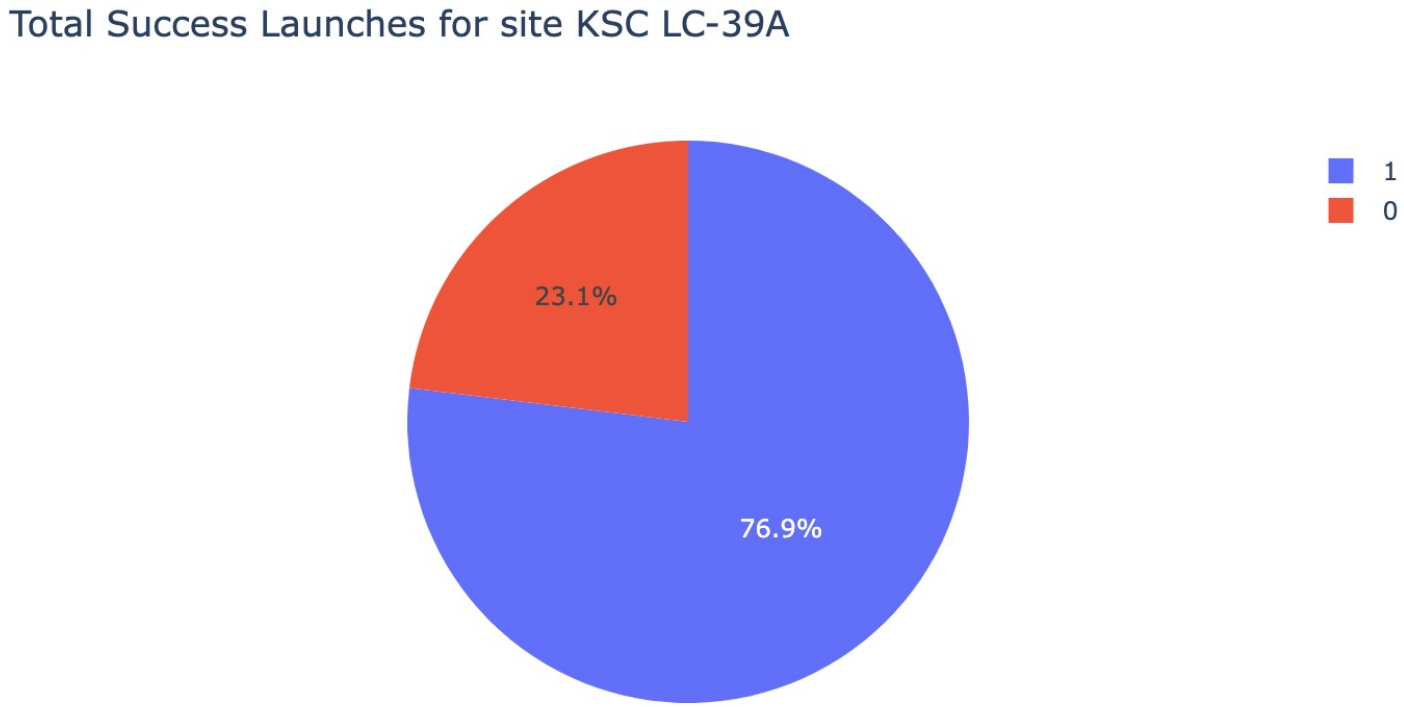
Launch Success Rate By All Sites

- The following pie chart shows the success rate of all launch sites
- The KSC LC-39A has the highest success rate of all the sites



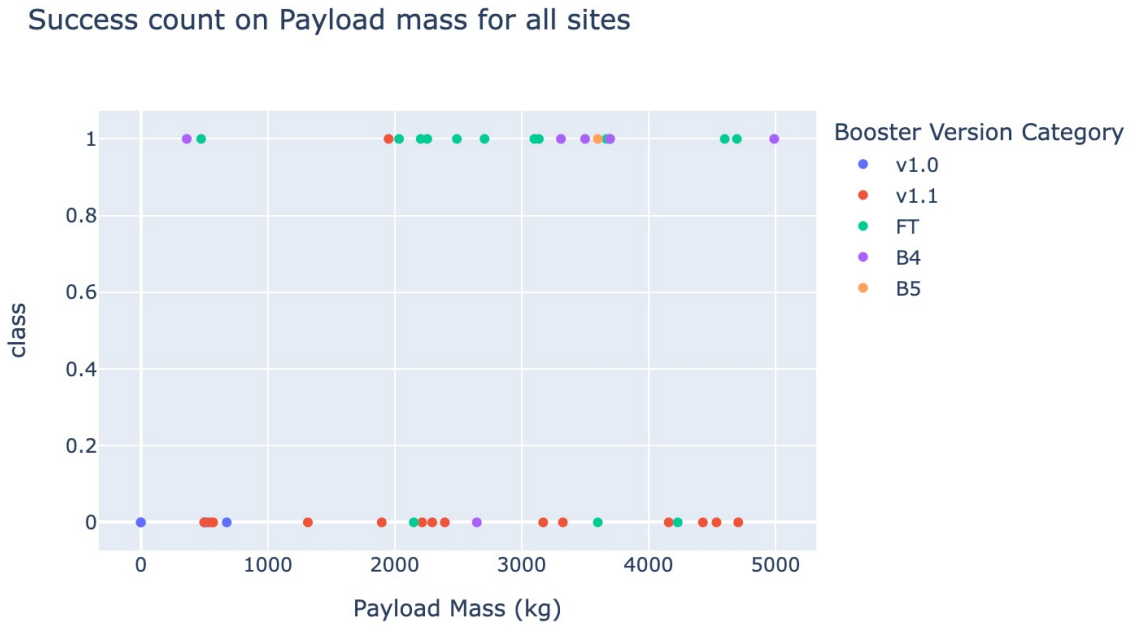
Launch Site With The Highest Success Ratio

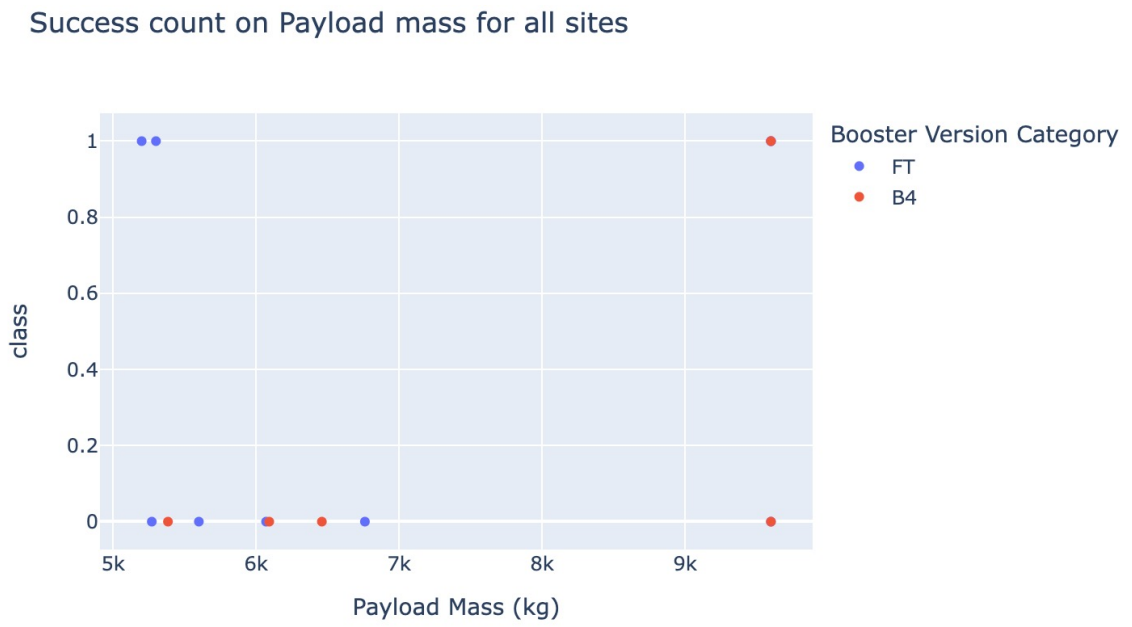
- The KSC LC-39A has the highest success rate out of all the sites
- This is shown with 10 successful landings (76.9%) and 3 failed landings (23.1%)



Payload vs Launch Outcome For All Sites

- The Figures below show that the success rate of the launch for lower weighted payloads (left graph) is higher than that of the heavier payloads (right graph)





Section 5

Predictive Analysis (Classification)

Classification Accuracy

- The following section built on the decision tree classifier had the highest accuracy score for me, with a score of .889 accuracy

```
parameters = {'criterion': ['gini', 'entropy'],
     'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],
     'max_features': ['auto', 'sqrt'],
     'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10]}

tree = DecisionTreeClassifier()
```

```
tree_cv = GridSearchCV(tree,parameters,cv=10)
tree_cv.fit(X_train, Y_train)
```

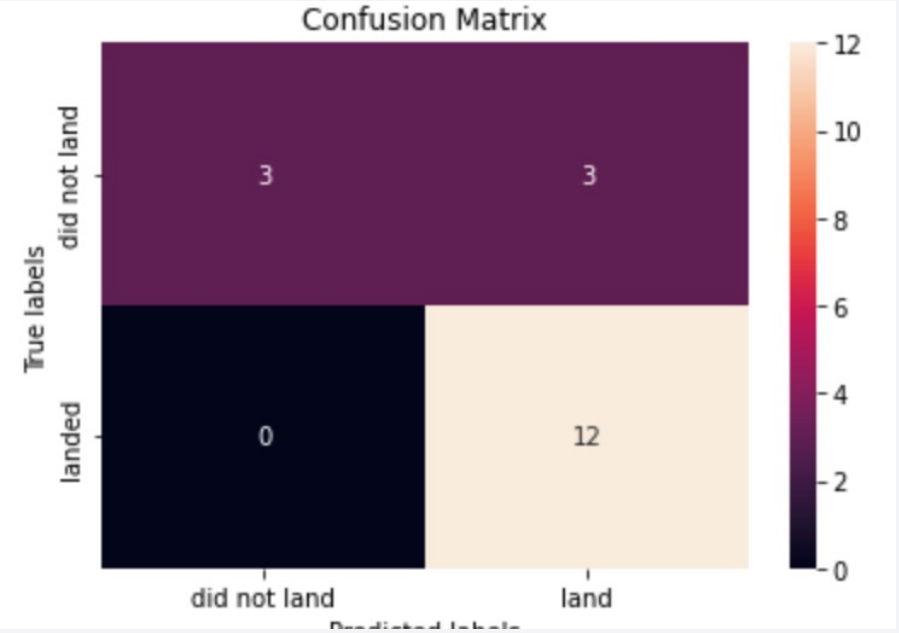
```
GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18],
                         'max_features': ['auto', 'sqrt'],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10],
                         'splitter': ['best', 'random']})
```

```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_dept
h': 12, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_spli
t': 5, 'splitter': 'random'}
accuracy : 0.8892857142857145
```

Confusion Matrix

- The confusion matrix for the decision tree shows that the classifier distinguished between different classes with the major issue staying with false positives
- As these models show successful landing, they were the same for all the tests in the dataset



Conclusions

- As the number of flights increase at a given site the success rate grows as well, exceeding 80% in recent tests
- The orbital types SSO, GEO, HEO, ES-L1 all have 100% success rates
- Launch sites are located near highways, railways, coastlines, and far from major cities
- The KSC LC-39A has the most successful recorded launches compared to other sites
- The Decision Tree Classifier is proven to be the best machine learning algorithm for this dataset

Appendix

- https://github.com/dkruger77/Capstone_project.md
- https://ibm-applied-data-science-capst.herokuapp.com/
- https://www.coursera.org/professional-certificates/ibm-data-science

Thank you!