IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Name>
<Date>

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection via API
  - Data Collection - Web Scraping
  - Data Wrangling
  - SQL: Analysis
  - Explore & Prep Data
  - Folium Visual Analytics
  - Machine Learning
- Summary of all results
  - EDA via visualizations
  - Machine Learning Results

Introduction

- Background:

  Commercial space age has arrived. With growing competitors, SpaceX is a key competitor who wants to make space travel affordable for everyone. While competitor rocket launches err on the side of $165M, SpaceX offers rocket launches, specifically via the Falcon 9, as low as $62M. The savings are driven by SpaceX's innovation to reuse the first stage of the launch for a future mission.

  Space Y is interested in determining the price of each SpaceX launch to help build Space Y's strategy. Space Y has partnered with data scientists to model and predict the landing outcomes for the reuse of the first stage.

- Problems you want to find answers:
  - Understand independent variables that influence SpaceX landing outcome
  - Probability of reuse for the first stage
  - Price of launch

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was sourced via web scrapping from a public Wikipedia source using the SpaceX API.
- Perform data wrangling
  - To process, data received one-hot encoding – impacting 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

A dataset was identified that captured information of interest for Space Y. Once identified, it was ingested via an API and web scrapping.

- The API used a get request, where it was transitioned into Json. In order to turn the data into a dataframe, json_normalize() was used. This was essential to be able to address missing values & clean the data.
- Web scrapping was completed using BeautifulSoup and an HTML response to gather the information available via the Wikipedia page and convert into a dataframe.

REST API

- Get request
- json_normalize to establish dataframe
- Data cleansing by addressing missing values

WEB SCRAPPING

- Beautiful Soup via HTML response
- Extracting data from HTML

Data Collection – SpaceX API

Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_ap
```

We should see that the request was successfull with the 200 status response code

```
response.status_code
```

```
200
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

```
# Use json_normalize meethod to convert the json result into a dataframe
# Use json_normalize() to convert JSON to DataFrame
data = pd.json_normalize(response.json())
```

```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that have multiple payloa
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the feature.
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

SpaceX Data API Request via GET

Normalize Json to create dataframe

Clean data to select features & fill missing value(s)

Data Collection API GitHub

Data Collection - Scraping

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

```
# use requests.get() method with the provided static_url
# assign the response to a object
data  = 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(data, 'html5lib')
```

Snippet of extract code:

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.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.`
            launch_dict['Flight No.'].append(flight_number)
            #print(flight_number)
            datatimelist=date_time(row[0])
```

Get request for Falcon 9 Launch data

Use BeautifulSoup to create BeautifulSoup object from response

Extract each column/variable name from HTML table

Web Scraping GitHub

Data Wrangling

In order to analyze, the data needed to undergo data wrangling to clean up the data set for future analysis / Exploratory Data Analysis (EDA).

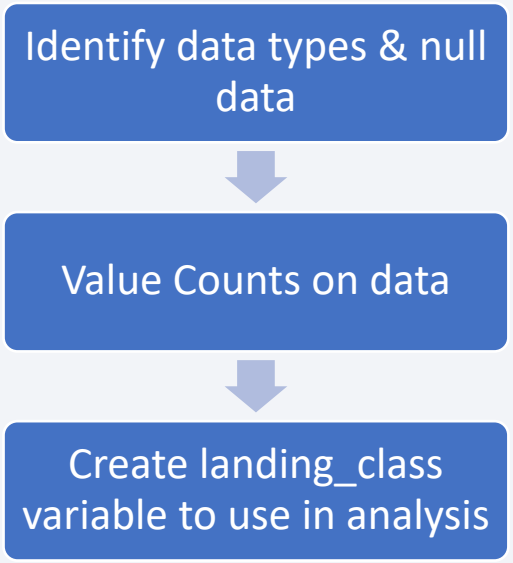
Pandas and numpy were imported to explore the SpaceX dataset. Null data was identified, along with numerical and categorical data types. Value counts was used to identify counts of specific values.

A variable was created called 'landing_outcomes' that uses the .value_counts() on 'Outcome' to determine the number of landing outcomes. Once complete, the result was exported to a CSV for EDA.

Data Wrangling GitHub

Code snippet:

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for outcome in df['Outcome']:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

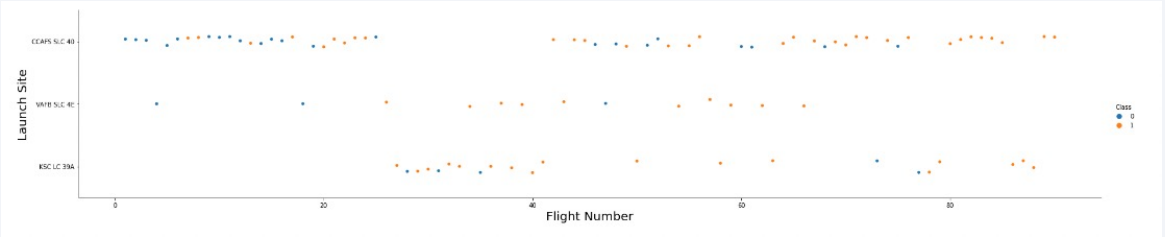


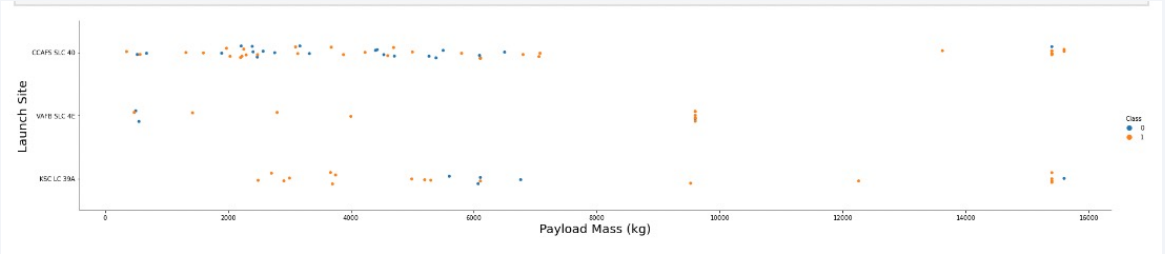
EDA with Data Visualization

To explore data relationships, the following graphs were used:

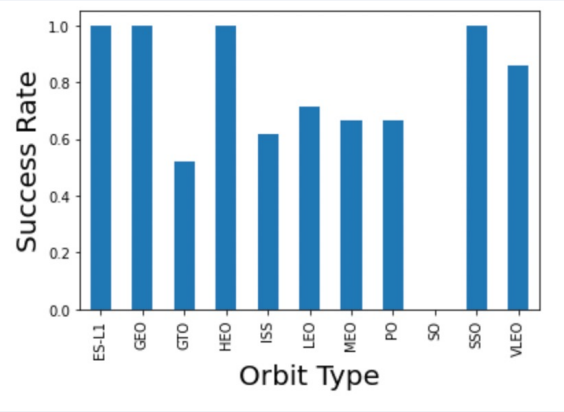
EDA Data Viz GitHub

- Scatter Plots → helpful to gain better understanding of relationships
  - Payload & Flight Number
  - Flight Number & Launch Site
  - Payload & Launch Site
  - Flight Number & Orbit Type
  - Payload & Orbit Type





- Bar Graph → useful to understand relationship between attributes
  - Success Rate by Orbit Type



- Line Plots → identifies attribute trends over time
  - SpaceX Rocket Success Rate by Year

At the end of the lab, feature engineering was used to predict success. Created by using the dummy variable in categorical columns.

EDA with SQL

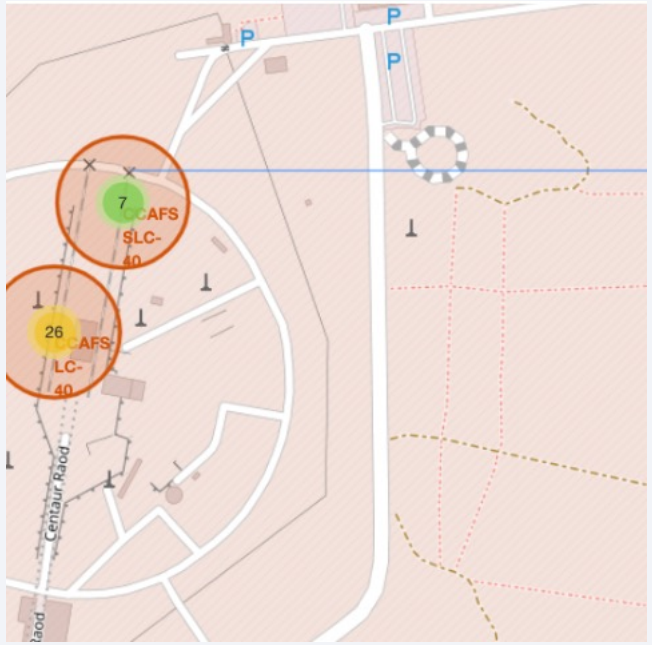
- SQL queries performed:
  - Launch site names
  - 5 records where launch site name begins with ”CCA”
  - Total payload mass carried by NASA (CRS) booster launch
  - Average payload mass carried by F9 1.1v booster
  - Date when first successful landing outcome in ground pad was achieved
  - Boosters with successful drone ship, with payload mass > 4,000 but < 6,000
  - Total of successful & failure mission outcomes
  - Booster versions names that carried max payload mass
  - 2015 failed drone ship landing outcomes, with their booster versions and launch site names
  - Rank of the count of landing outcomes between 6/4/2010 and 3/20/2017 in descending order.

EDA with SQL GitHub

Build an Interactive Map with Folium

In order to visualize the geographical component of the data, the launch data was mapped via a folium map to plot the markers via latitude and longitude coordinates. Circle markers were added that highlight the circle area with a text label (Launch Site name) of a specific coordinate.

This helped to visualize the launch outcomes by launch site. Red/green color markers were created to highlight failure and success.



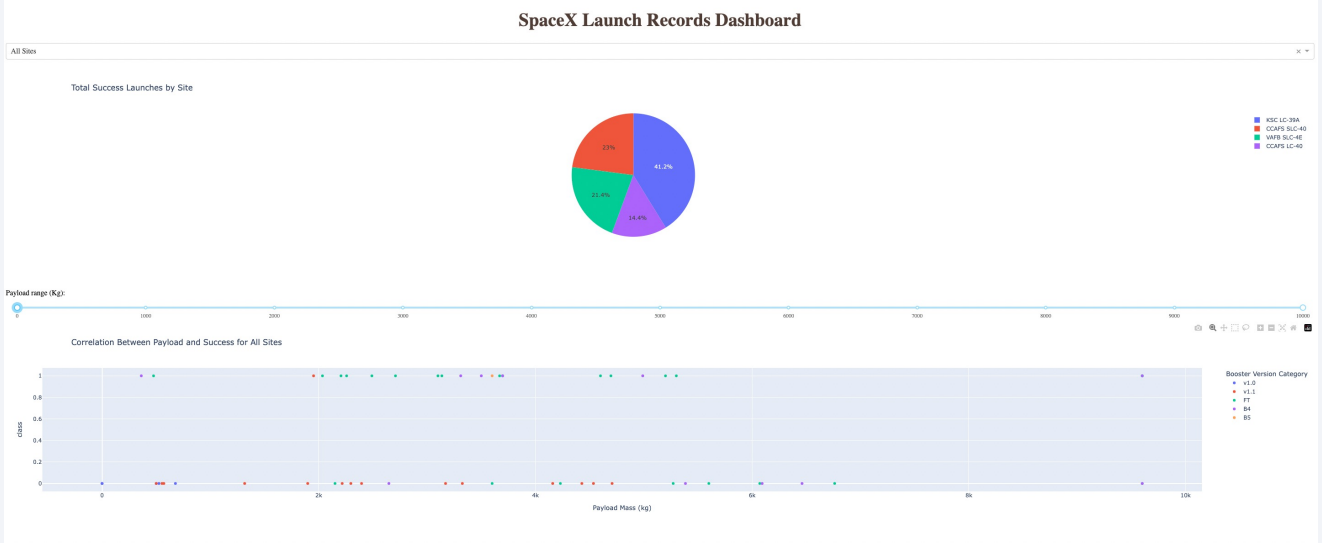
Folium Map GitHub

Build a Dashboard with Plotly Dash

A Plotly Dashboard was created to visualize plots/graphs by users. The visualizations are filterable by launch site.

Visualizations present in the dashboard include:

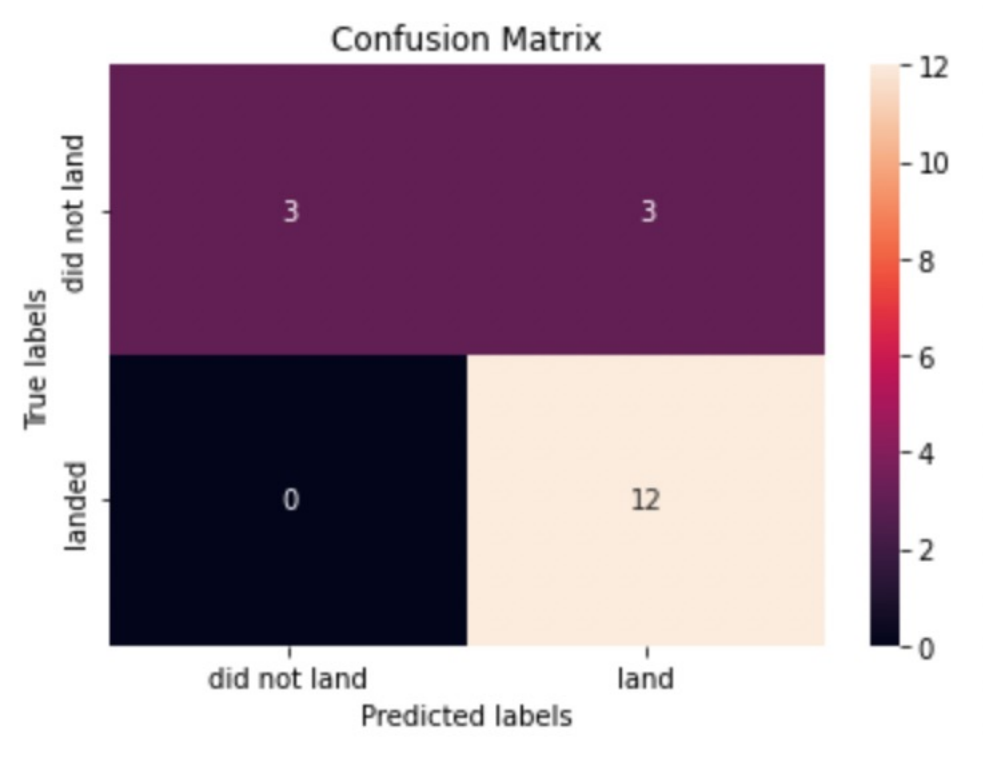
- Pie chart → success and count of launch sites
- Scatter plot graph → identify relationship of Payload Mass and Outcome for different boosters



[Plotly Dashboard GitHub]

Predictive Analysis (Classification)

To identify the best classification model, the following approach was used to build, evaluate, and improve.



Build Model
- Import Pandas & numpy
- Transform data
- Split into training & test datasets
- Identify best ML approach to use
- Set parameters, algorithms & fit to dataset

Evaluate Model
- Identify accuracy of model
- Adjust parameters as needed
- Utilize confusion matrix to visualize result(s)

Improve Model
- Algorithm Tuning
- Feature Engineering

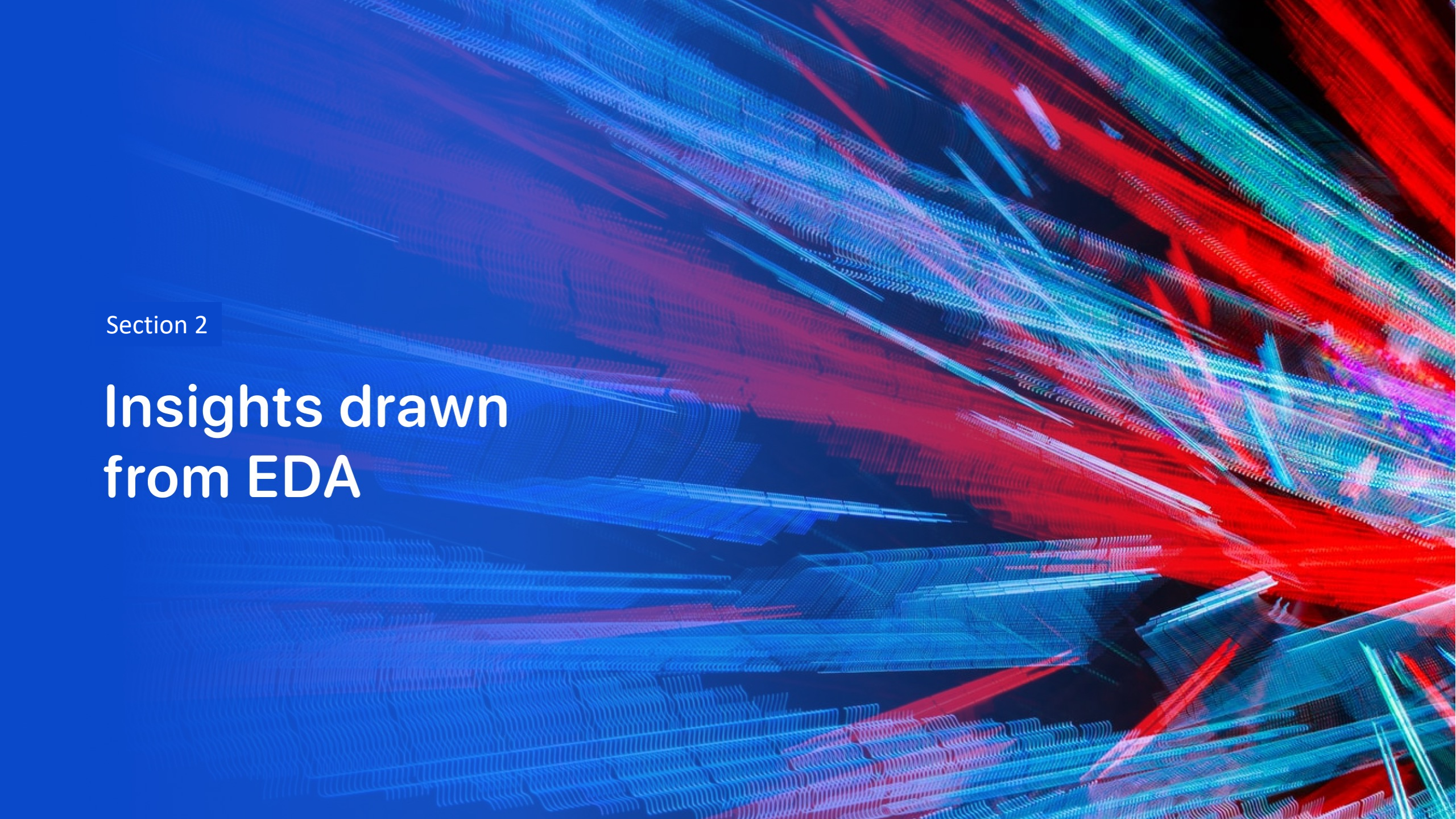
Identify Best Model
- Select model with best accuracy score

Machine Learning GitHub

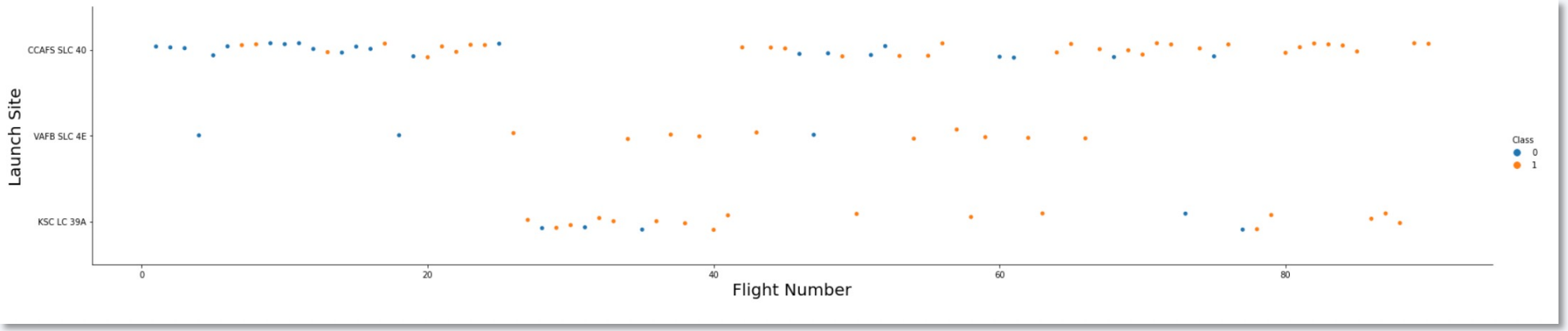
Results

Please view the following sections to view detailed results from the following analyses:

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


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

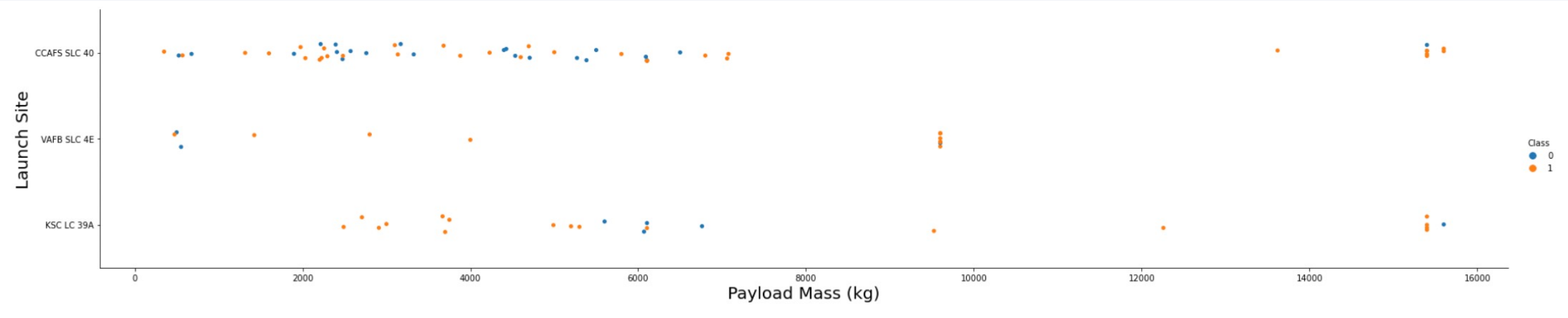


This scatter plot aids in understanding the relationship between Flight Number & Launch Site.

As visualized, the larger the flight amounts, the greater the success rate.

CCAFS SLC40 calls attention as despite the increase in flight numbers, there are still some failures that occur. While this visualization is great to identify possible tests, it should not be used to form conclusions without proper testing.

Payload vs. Launch Site

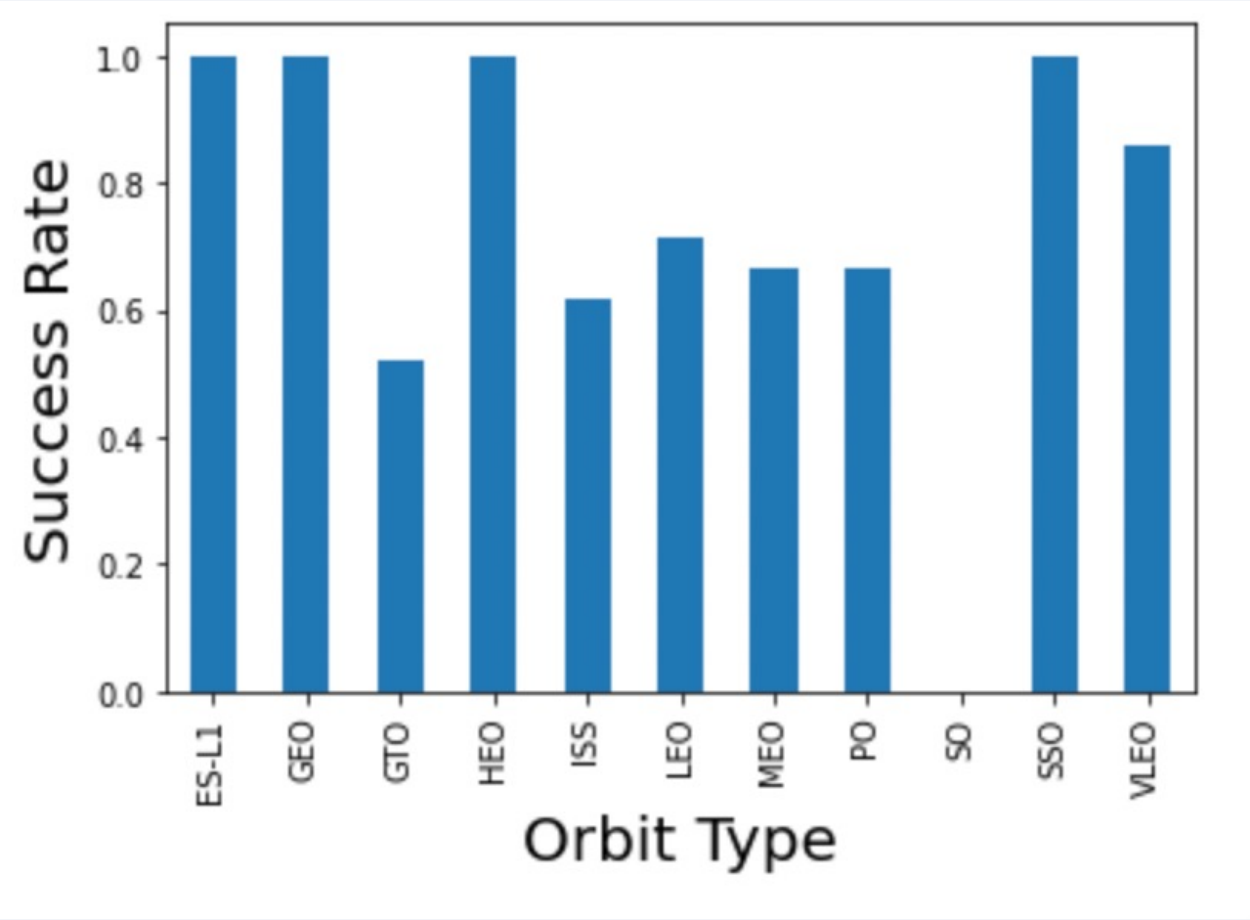


Visualized above is the relationship between Payload Mass (kg) and Launch Site.

As displayed, the higher the payload mass, the higher the probability of success. Specifically, failures decrease after payload increases higher than 6500-7000 kg.

Disclaimer that there are still failures for each launch site above a payload of 7,000 kg, which helps inform that the payload is not the only independent variable that can impact a launch site's success outcome.

Success Rate vs. Orbit Type

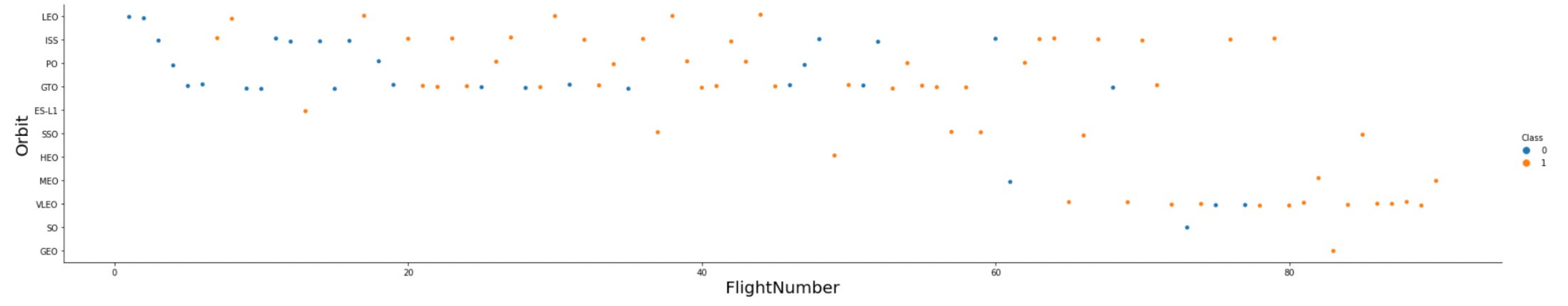


The bar chart is to visualize the success rate by orbit type to understand if the certain orbit paths have higher landing success rates.

- ES-L1, GEO, HEO, SSO all have 100% success rate.
- SO has a 0% success rate.

That said, not all orbit types have had the same number of launches, which means we cannot assume that a 0% success rate is due to the orbit. But could be rather that the specific orbit has not had the same # of tests as other orbits. Similarly, a 100% success rate could mean there is only 1 success, vs an orbit like GTO which could have more launches.

Flight Number vs. Orbit Type

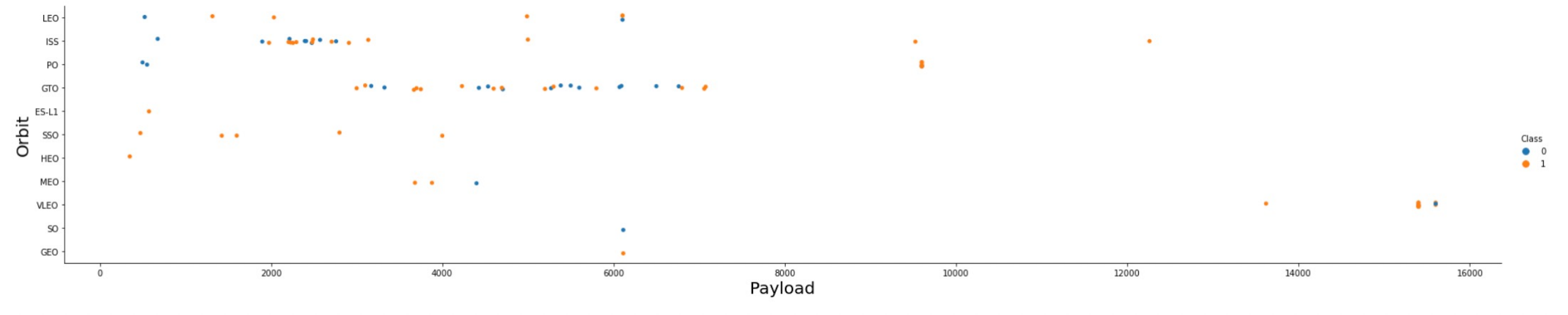


This scatter plot helps visualize the number of flights per each orbit.

The larger the flight number of the orbit, the higher the success rate – especially as shown on the lower right of the graph (for LEO).

That said, GTO does not have a conclusive pattern to infer the relationship between orbit and flight number.

Payload vs. Orbit Type

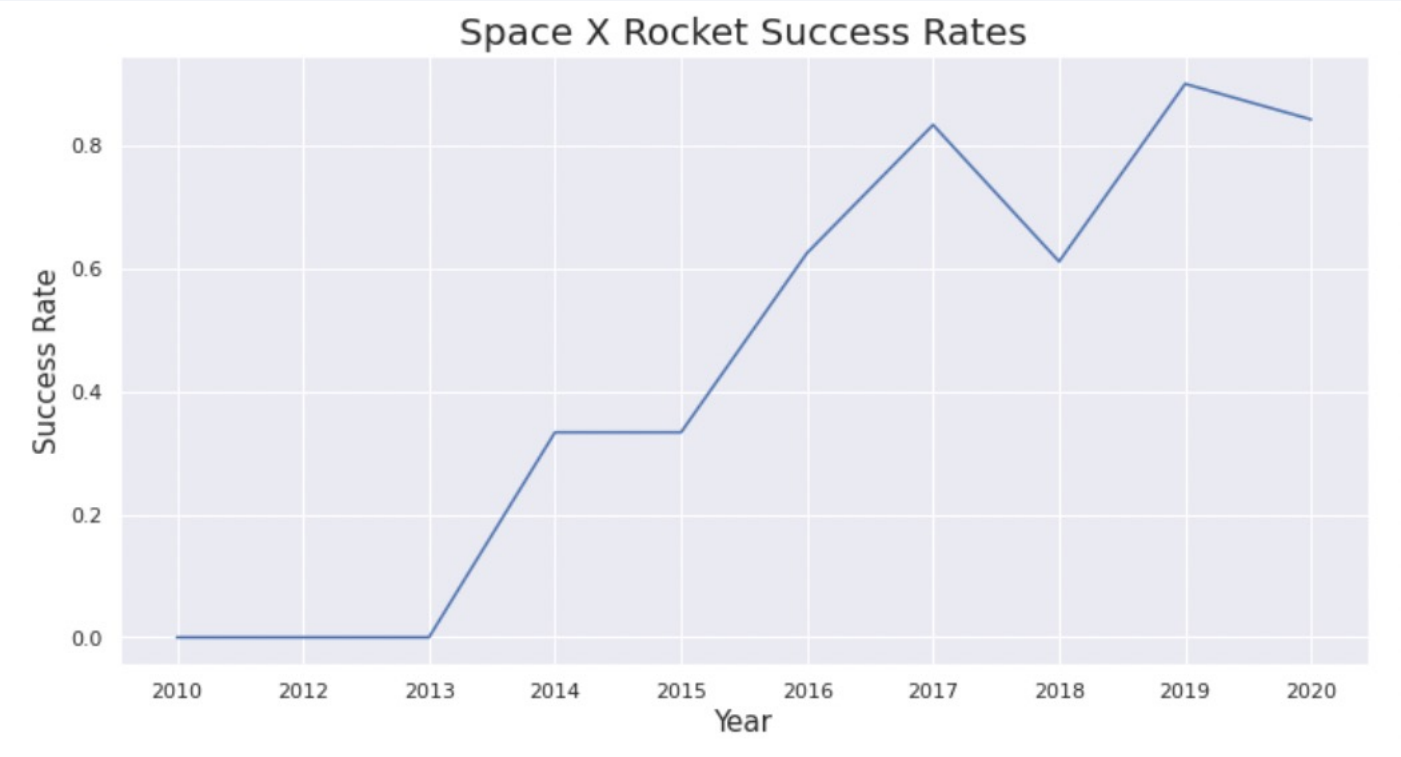


The scatter plot visualizes the relationship between Payload and Orbit Type.

The heavier the payload, the better the success for some orbits. However, there are a couple orbits like MEO and VLEO that have opposite impact.

For this visual, it is tough to infer a conclusion, which means the variables are not the only impact on success for orbits.

Launch Success Yearly Trend



The line chart helps visualize the trend of success rates by time (year).

As depicted, success rates have increased throughout time. Specifically, between 2013-2017. There was an interesting decrease in success rate (20% drop) in 2018.

All Launch Site Names

The SQL query selected all distinct launch site names from the dataset.

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

```
%sql select distinct launch_site from SPACEXDATASET;
```

```
 * ibm_db_sa://wzf08322:***@0c77d6f2-5da9-48a9-81f8-86b520b8751
8.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

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

Launch Site Names Begin with 'CCA'

The SQL query selected all information from the dataset where launch site name started with the letters ‘CCA’. The SQL also limited the query to return 5 records that contained the above requirements.

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

```
%sql select * from SPACEXDATASET where launch_site like 'CCA%' limit 5;
```

```
 * ibm_db_sa://wzf08322:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

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

SQL query requested the total sum of all payload mass (kg) that was launched by NASA (CRS). The query also requested that the total sum of all payload mass (kg) be labeled as 'total_payload_mass'.

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

```
%sql select sum(payload_mass__kg_) as total_payload_mass from SPACEXDATASET where customer = 'NASA (CRS)';
```

```
 * ibm_db_sa://wzf08322:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

total_payload_mass
45596

Average Payload Mass by F9 v1.1

The SQL query requested the average payload mass from the dataset where it carried by booster version F9 v1.1, to be labeled under the name ‘average_payload_mass’.

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

```
%sql select avg(payload_mass__kg_) as average_payload_mass from SPACEXDATASET where booster_version like '%F9 v1.1%';
```

```
 * ibm_db_sa://wzf08322:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

average_payload_mass
2534

First Successful Ground Landing Date

SQL query requested the dates of the first successful landing outcome on ground pad by using the min function. The SQL requested first successful landing date be labeled as 'first_successful_landing'

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

```
%sql select min(date) as first_successful_landing from SPACEXDATASET where landing__outcome = 'Success (ground pad)';
```

```
 * ibm_db_sa://wzf08322:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

first_successful_landing
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- SQL query requests the list the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000. This query filters the data to identify what the user/data scientist is looking to identify.

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

```
%sql select booster_version from SPACEXDATASET where landing__outcome = 'Success (drone ship)' and payload_mass__kg_ between 4000 and 600(
```

```
 * ibm_db_sa://wzf08322:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

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

Total Number of Successful and Failure Mission Outcomes

SQL query counts the total number of successful and failure mission outcome and groups by mission outcome.

List the total number of successful and failure mission outcomes

```
%sql select mission_outcome, count(*) as total_number from SPACEXDATASET group by mission_outcome;
```

```
 * ibm_db_sa://wzf08322:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

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

Boosters Carried Maximum Payload

SQL query lists the names of the booster which have carried the maximum payload mass by using the max subquery.

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

```
%sql select booster_version from SPACEXDATASET where payload_mass__kg_ = (select max(payload_mass__kg_) from SPACEXDATASET);
```

```
 * ibm_db_sa://wzf08322:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

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

To digest easier, the month SQL query feature was used to parse out the months of 2015 where there were drone ship failures. The date, booster version, and launch site were pulled in for the two drone ship failures in 2015.

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for the in year 2015

```
%%sql select monthname(date) as month, date, booster_version, launch_site, landing__outcome from SPACEXDATASET
      where landing__outcome = 'Failure (drone ship)' and year(date)=2015;
```

```
 * ibm_db_sa://wzf08322:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

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

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

- The SQL query counted the number of landing outcomes between 6/4/2010 and 3/20/20117, sorting by highest outcome count (descending order). The count of outcomes was given the name “count_outcomes.”

Task 10

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

```
%%sql select landing__outcome, count(*) as count_outcomes from SPACEXDATASET
      where date between '2010-06-04' and '2017-03-20'
      group by landing__outcome
      order by count_outcomes desc;
```

```
 * ibm_db_sa://wzf08322:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

landing__outcome	count_outcomes
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

Launch Site Locations

Identifies the geographical locations of the SpaceX launch sites.

Launch Site Locations: Successful & Failure Results

The screenshot shows the geographical mapping of CCAFS LC-40.

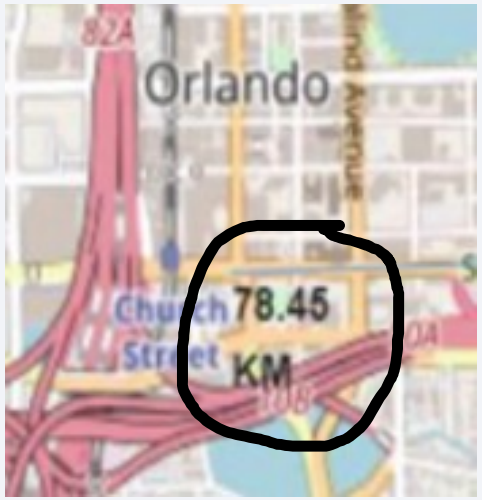
The circle marker is also labeled with red and green labels to represent the Successful and Failure launches.

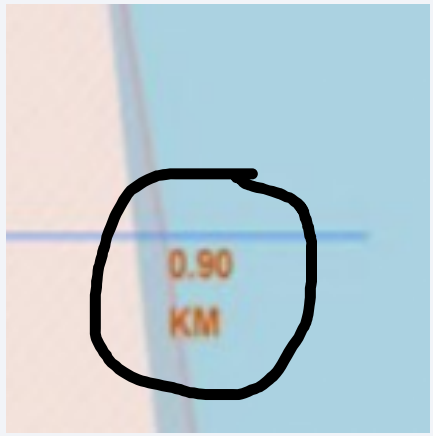
Launch Sites: Landmark Distance

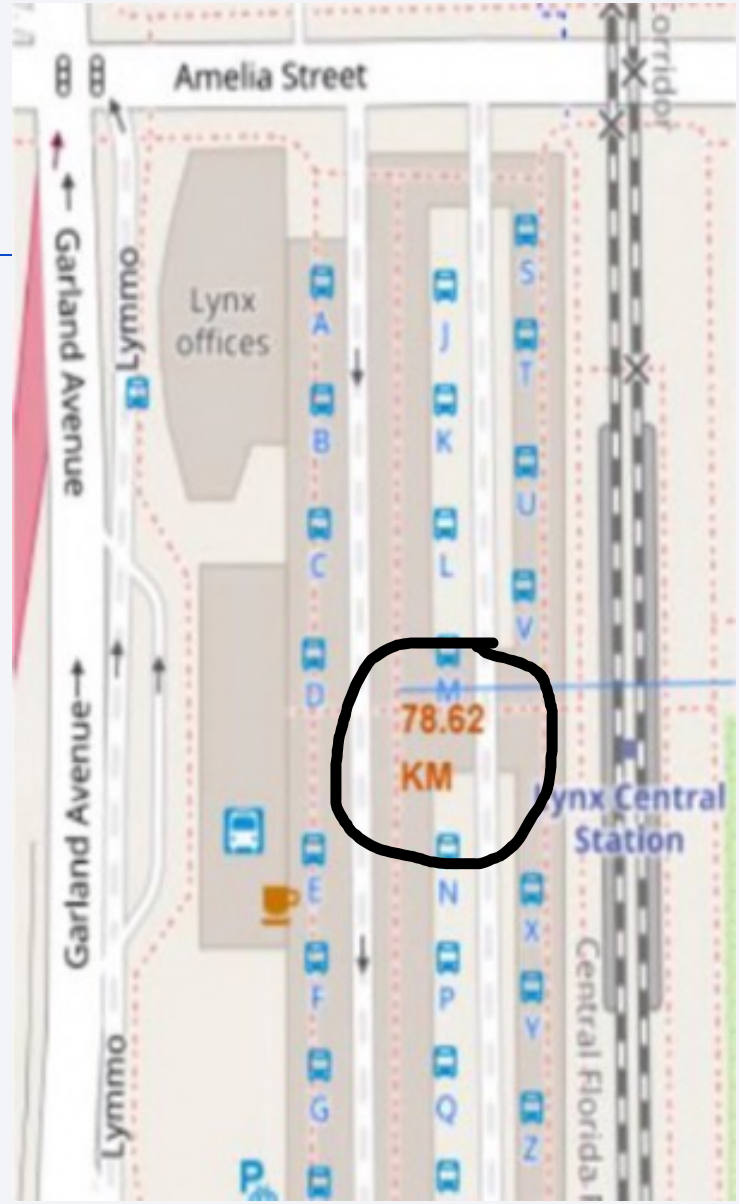
Launch sites are not within close proximity to the following:

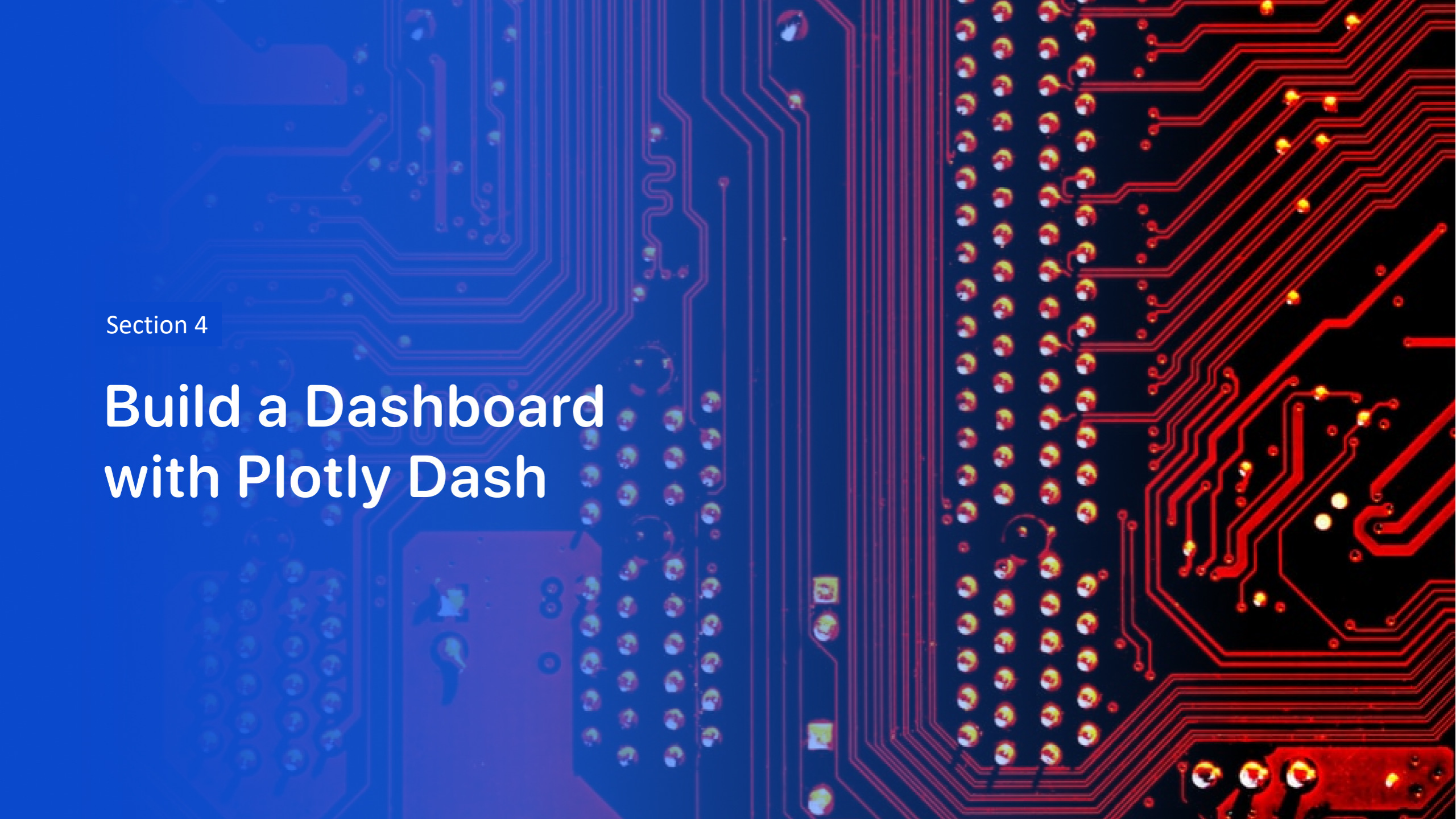
- Railways
- Highways
- Coastline

Launch sites also remain a certain distance away from cities.





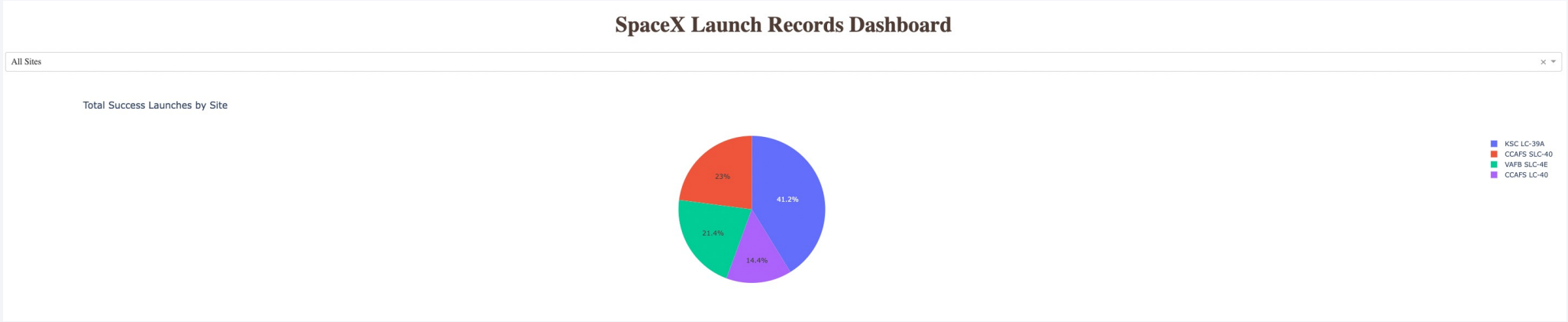



Section 4
Build a Dashboard with Plotly Dash

SpaceX Launch Records Dashboard: Launch Success

In the screenshot below, we can see the success of launches by launch site. When hovering over the pie chart, the count of launches is viewable.

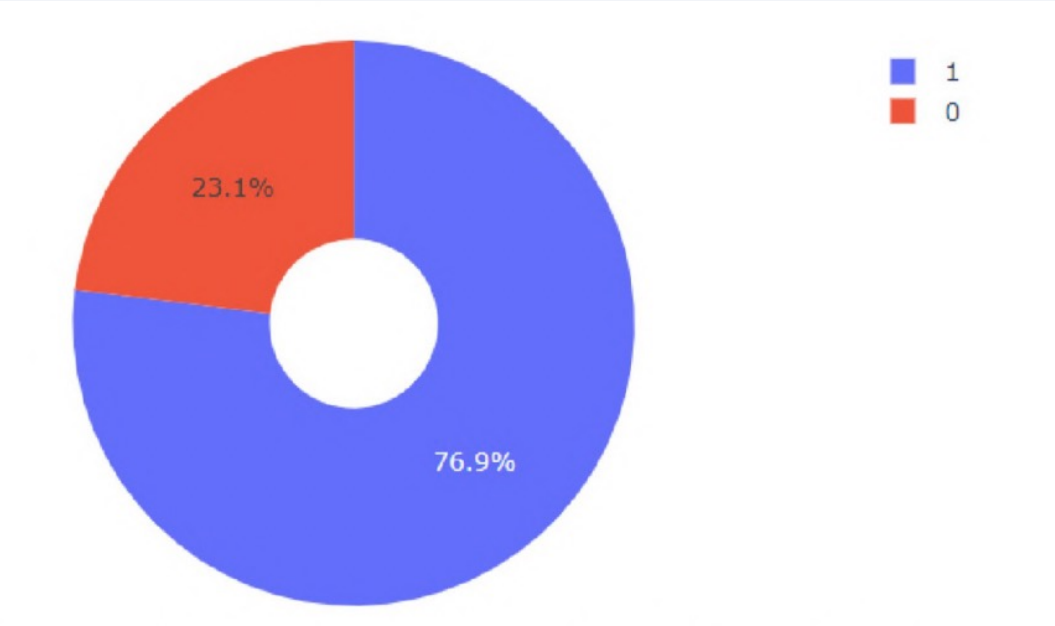
- KSC LC-39A has the highest successful launches vs. the other launch sites.



SpaceX Launch Records Dashboard: Highest Launch Success Ratio

KSC LC-39A has the highest success ratio for launches.

- 23.1% failure
- 76.9% success



SpaceX Launch Records Dashboard: Payload vs Launch Outcome

Payload Mass 0 kg – 5,000kg

Payload Mass 5,000 kg – 10,000kg

- The payload mass between 2000 – 5000kg has higher success rates than the payloads between 5000kg and 10000kg.
- F9 B4 B1031.2 has a high level of success and with a range of payload masses.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

Identified the decision tree model has the highest classification accuracy

```
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'}
```

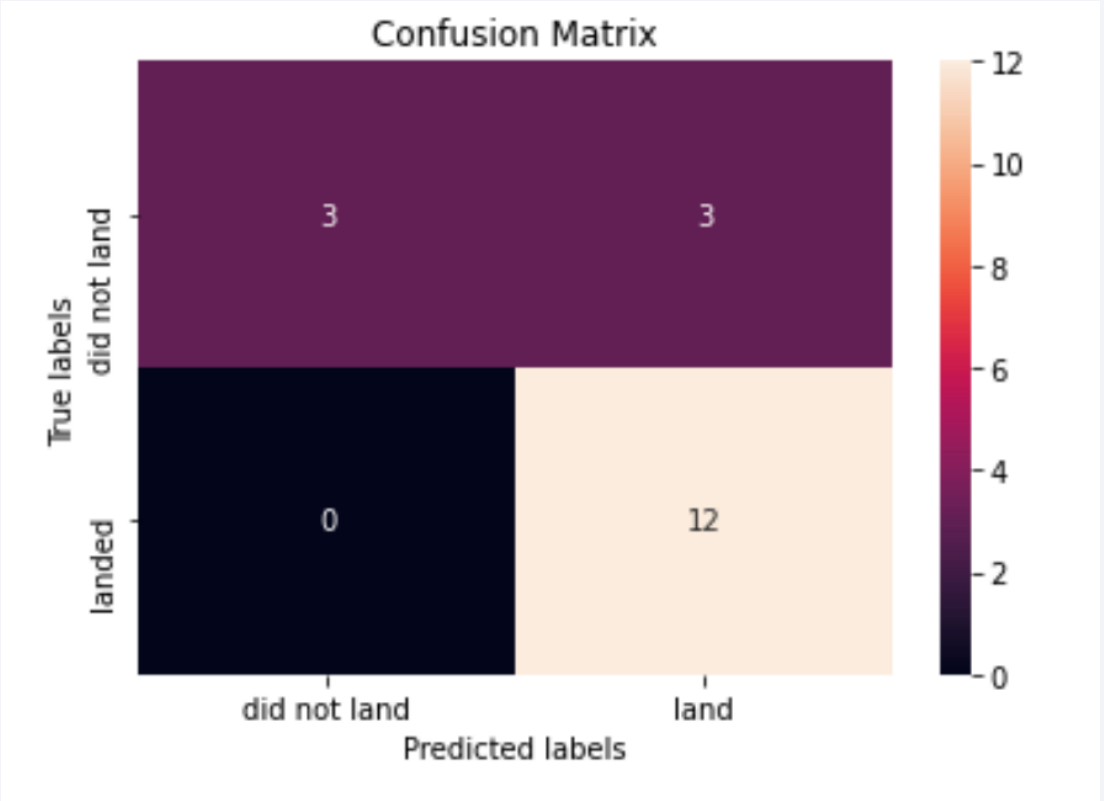
	LogReg	SVM	Tree	KNN
Jaccard_Score	0.833333	0.845070	0.808219	0.819444
F1_Score	0.909091	0.916031	0.893939	0.900763
Accuracy	0.866667	0.877778	0.844444	0.855556

Confusion Matrix

Displayed is the confusion matrix for the decision tree.

Highlights the classifier between different classes.

- 3 false positives
- 3 true negatives
- 0 false negatives
- 12 true positives



Conclusions

- The decision tree classifier is the best machine learning approach for the given dataset.
- The higher flight numbers, the greater the success rate.
- KSC LC-39A has the highest number of successful launches, and highest success ratio of all the launch sites.
- Orbits ES-L1, GEO, HEO, SSO, VLEO had 100% success rate.
- Launch sites are not built within close proximity to railroads, cities, coast, or major highways.
- Via the dashboard, the heavier the payload mass, the lower the success rate.

Appendix

Please view the GitHub repository for further detail.

Thank you!