IBM DATA SCIENCE CAPSTONE PROJECT

Space X Falcon 9 Landing Analysis

Ethan Benavides

OUTLINE

EXECUTIVE SUMMARY

Summary of Methodologies

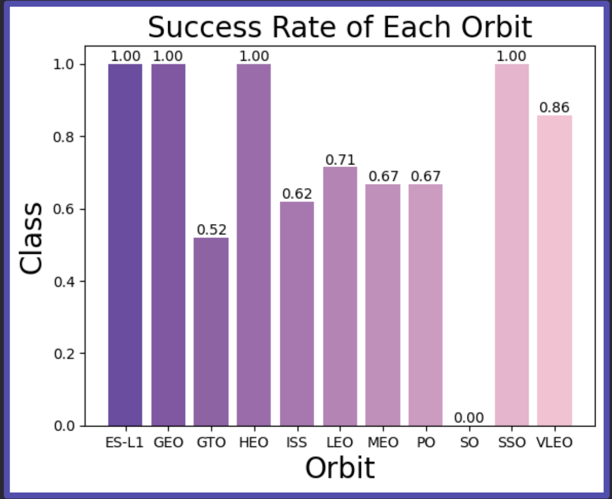
The project follows these steps:

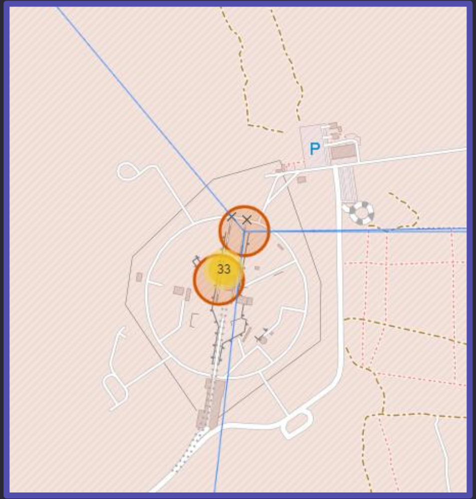
- Data Collection
- Data Wrangling
- Exploratory Data Analysis
- Interactive Visual Analytics
- Predictive Analysis (Classification)

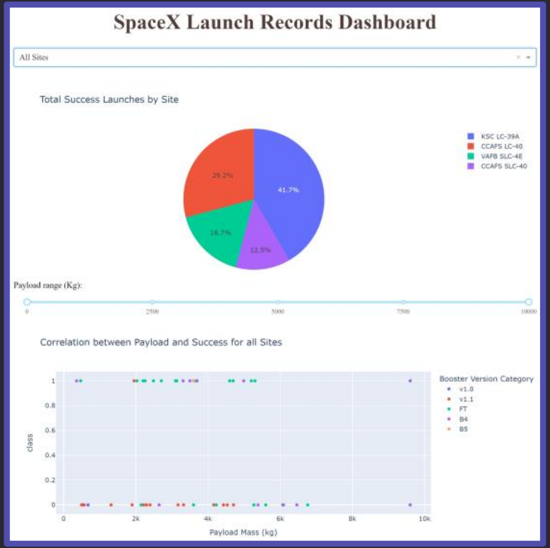
Summary of Results

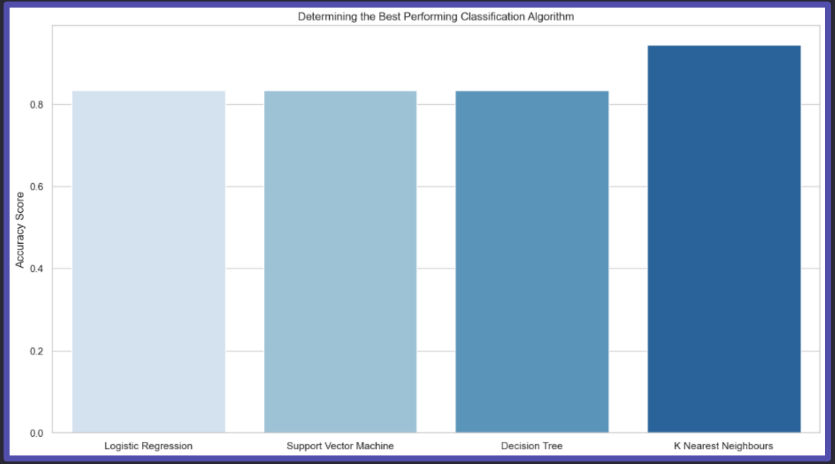
This project produced the following:

- Exploratory Data Analysis (EDA) Results
- Geospatial Analytics
- Interactive Dashboards
- Predictive Analysis of Classification Models









INTRODUCTION

- SpaceX launches Falcon 9 rockets with a lower cost per launch at about $62m as compared to other companies (typically about $165m). This can be credited to the fact that SpaceX can land and re-use the first stage of the rocket.

- If predictions can be made about whether the first stage will land or not, then we can determine the cost of a launch. We can then use this information to assess if an alternate company should bid against SpaceX for a rocket launch.

The aim of this project is to successfully predict if the SpaceX Flacon 9 will land successfully.

METHODOLOGY SUMMARY

I. Data Collection
- Using GET requests to the SpaceX REST API
- Web Scraping

II. Data Wrangling
- Using `.fillna()` method to remove NaN values
- Using `.value_counts()` method to determine:
  - Number of launches on each site
  - Number of occurrences of each orbit
  - Number and occurrence of mission outcome per orbit type
- Creating a landing outcome label that shows the following:
  - 0 when the booster did not land successfully
  - 1 when the booster landed successfully

III. Exploratory Data Analysis
- Using SQL queries to manipulate and evaluate the SpaceX dataset
- Using Pandas and Matplotlib to visualize relationships between variables and determine patterns

IV. Interactive Visual Analytics
- Geospatial analytics using Folium
- Creating an interactive dashboard using Plotly Dash

V. Data Modeling and Evaluation
- Using Scikit-Learn to:
  - Pre-process (standardize and normalize the data)
  - Split the data into training and testing data using `.train_test_split()`
  - Train different classification models
  - Find hyperparameters using `.GridSearchCV()`
- Plotting confusion matrices for each classification model
- Assessing the accuracy of each classification model

DATA COLLECTION:
SPACEX REST API

Using the SpaceX API to retrieve data about various launches, this process includes gathering information on the type of rocket used, payload delivered, launch specification, landing specification, and landing outcome for each entry.

STEP 1:

- Make a GET response to the SpaceX REST API
- Convert it to a .json file then into a Pandas DataFrame

STEP 2:

- Use custom logic to clean the data
- Define lists for the data to be stored in
- Call custom functions to retrieve data and fill the lists
- Use list values as a dictionary and construct the dataset

STEP 3:

- Create Pandas DataFrame from the constructed dictionary dataset

STEP 4:

- Filter the DataFrame to only include Falcon 9 launches
- Reset the FlightNumber column
- Replace missing values of PayloadMass with mean PayloadMass

1

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
response = requests.get(spacex_url)
```

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

2

```
#Global variables
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

```
# Call getBoosterVersion
getBoosterVersion(data)
```

```
# Call getLaunchSite
getLaunchSite(data)
```

```
# Call getPayloadData
getPayloadData(data)
```

```
# Call getCoreData
getCoreData(data)
```

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

3

```
# Create a data from launch_dict
df = pd.DataFrame(launch_dict)
```

4

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = df[df['BoosterVersion'] == 'Falcon 9']
```

```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
```

```
# Calculate the mean value of PayloadMass column
plm_mean = data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9.loc[:, 'PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, plm_mean)
```

DATA COLLECTION:
WEB SCRAPING

Web scraping done with BeautifulSoup to collect Falcon 9 historical launch records from a Wikipedia page titled Falcon 9 and Falcon Heavy launches.

STEP 1:

- Request the HTML page from the static URL

STEP 2:

- Create a BeautifulSoup object
- Find all tables within the HTML page

STEP 3:

- Collect all column header names from the tables found within the HTML page

STEP 4:

- Use the column names as keys in a dictionary
- Use custom functions and logic to parse all launch tables to fill dictionary values

STEP 5:

- Convert the dictionary to Pandas DataFrame ready for export

1

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

2

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

3

```
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
for th in first_launch_table.find_all('th'):
    name = extract_column_from_header(th)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

4

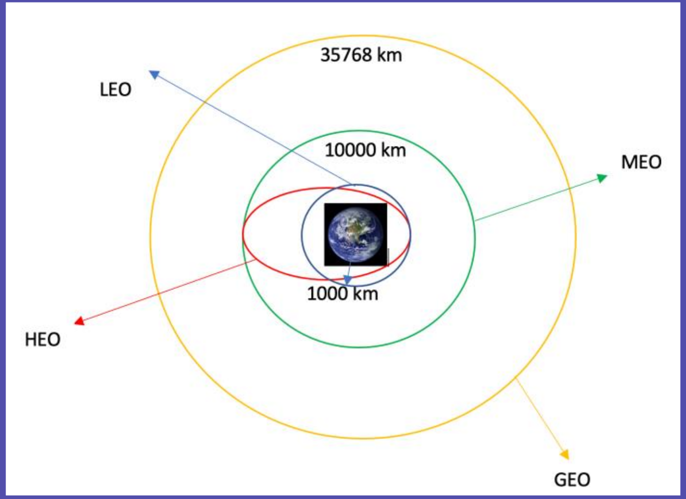
```
launch_dict= dict.fromkeys(column_names)

# 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']=[]
```

5

```
df = pd.DataFrame(launch_dict)
```

DATA WRANGLING USING PANDAS

- The SpaceX dataset contains various launch facilities. We can identify the launch facilities in the `LaunchSite` column.
- Each launch aims toward a dedicated orbit. The orbit types can be found in the `Orbit` column. The various orbit types are shown below:



- Using the `.value_counts()` method, we can determine the following information:
  - Number of launches for each site
  - Number and occurrence of each orbit
  - Number and occurrence of landing outcome per orbit type

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

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

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

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

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

```
Outcome
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: count, dtype: int64
```

DATA WRANGLING USING PANDAS

- The landing outcome is shown in the Outcome column:
  - True Ocean - the mission outcome was successfully landed to a specific region of the ocean.
  - False Ocean - the mission outcome was unsuccessfully landed to a specific region of the ocean.
  - True RTLS - the mission outcome was successfully landed to a ground pad.
  - False RTLS -the mission outcome was unsuccessfully landed to a ground pad.
  - True ASDS - the mission outcome was successfully landed to a drone ship.
  - False ASDS - the mission outcome was unsuccessfully landed to a drone ship.
  - None ASDS and None None - these represent a failure to land.

1

```
bad_outcomes = set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

2

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = [0 if outcome in bad_outcomes else 1 for outcome in df['Outcome']]
```

- To determine whether a booster will successfully land, it is best to have a binary column (i.e., where the value is 1 or 0, representing the landing status).
- The is done by:
  - Step 1: Defining a set of unsuccessful outcomes, bad_outcome.
  - Step 2: Creating a list, landing_class, where the element is 0 if the corresponding row in Outcome is in the set bad_outcome, otherwise, it's 1.
  - Step 3: Create a Class column that contains the values from the list landing_class.
  - Step 4: Export the DataFrame as a .csv file.

3

```
df['Class']=landing_class
```

4

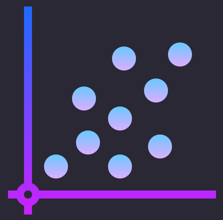
```
df.to_csv("dataset_part_2.csv", index = False)
```

EXPLORATORY DATA ANALYSIS (EDA) WITH VISUALIZATION

SCATTER PLOTS

Scatter plots were used to visualize the relationships between:

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

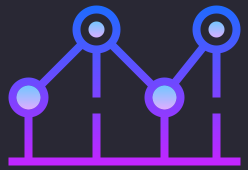


Scatter plots help visualize the relationships and correlations between two numeric variables

BAR CHARTS

A bar chart was used to visualize the relationship between:

- Success Rate x Orbit Type

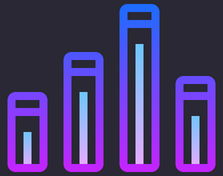


Bar charts are used to compare a numeric value to a categorical variable. They can also be used to compare multiple variables to each other.

LINE CHARTS

Line charts were used to produce and visualize the relationship between:

- Success Rate x Year



Line charts contains multiple numerical values on both axes and are generally used for showing the change in a variable measured against itself.

EXPLORATORY DATA ANALYSIS (EDA) USING SQL

To gather some information about the dataset, some SQL queries were performed.

The SQL queries performed on the data set were used to:

1. Display the names of the unique launch sites in the space mission
2. Display 5 records where launch sites begin with the string `'CCA'`
3. Display the total payload mass carried by boosters launched by `NASA (CRS)`
4. Display the average payload mass carried by booster version `F9 v1.1`
5. List the date when the first successful landing outcome on a ground pad was achieved
6. List the names of the boosters which had success on a drone ship and a payload mass between `4000` and `6000 kg`
7. List the total number of successful and failed mission outcomes
8. List the names of the booster versions which have carried the maximum payload mass
9. List the failed landing outcomes on drone ships, their booster versions, and launch site names for `2015`
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

GEOSPATIAL ANALYSIS USING FOLIUM

The following steps were taken to visualize the launch data on an interactive map:

1. Mark all launch sites on a map
   - Initialize the map using a Folium `Map` object
   - Add a `folium.Circle` and `folium.Marker` for each launch site on the launch map

2. Mark the success/failed launches for each site on a map
   - As many launches have the same coordinates, it makes sense to cluster them together.
   - Before clustering them, assign a marker color of successful (class = 1) as green, and failed (class = 0) as red. To put the launches into clusters, for each launch, add a `folium.Marker` to the `MarkerCluster()` object.
   - Create an icon as a text label, assigning the `icon_color` as the `marker_color` determined previously.

3. Calculate the distances between a launch site to its proximities
   - To explore the proximities of launch sites, calculations of distances between points can be made using the `Lat` and `Long` values.
   - After marking a point using the `Lat` and `Long` values, create a `folium.Marker` object to show the distance.
   - To display the distance line between two points, draw a `folium.PolyLine` and add this to the map.

INTERACTIVE DASHBOARD USING PLOTLY DASH

The following plots were added to a Plotly Dash dashboard to have an interactive visualization of the data:

1. Pie chart (`px.pie()`) showing the total successful launches per site
   - This makes it clear to see which sites are most successful
   - The chart could also be filtered (using a `dcc.Dropdown()` object) to see the success/failure ratio for an individual site

2. Scatter graph (`px.scatter()`) to show the correlation between outcome (success or not) and payload mass (kg)
   - This could be filtered (using a `RangeSlider()` object) by ranges of payload masses
   - It could also be filtered by booster version

PREDICTIVE ANALYSIS USING CLASSIFICATION

The following steps were taken to develop, evaluate, and find the best performing classification model:

MODEL DEVELOPMENT

- To prepare the dataset for model development:
  1. Load dataset
  2. Perform necessary data transformations (standardize and pre-process)
  3. Split data into training and test data sets, using `train_test_split()`
  4. Decide which type of machine learning algorithms are most appropriate
- For each chosen algorithm:
  1. Create a `GridSearchCV` object and a dictionary of parameters
  2. Fit the object to the parameters
  3. Use the training data set to train the model

MODEL EVALUATION

- For each chosen algorithm:
  - Using the output GridSearchCV object:
  - Check the tuned hyperparameters (`best_params_`)
  - Check the accuracy (`score` and `best_score_`)
  - Plot and examine the Confusion
  - Matrix

FINDING THE BEST MODEL

- Review the accuracy scores for each algorithm
- The model with the highest accuracy score is determined as the best performing model

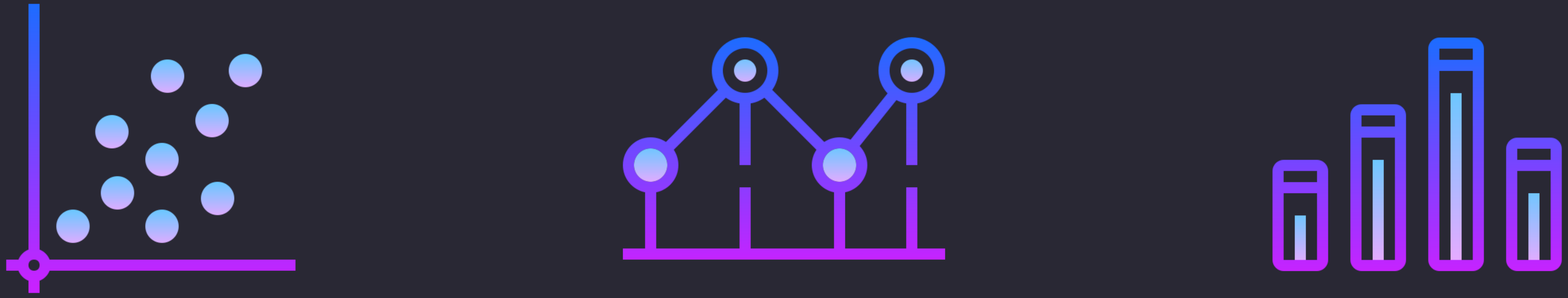
RESULTS

EXPLORATORY ANALYSIS

INTERACTIVE ANALYTICS

PREDICTIVE ANALYSIS

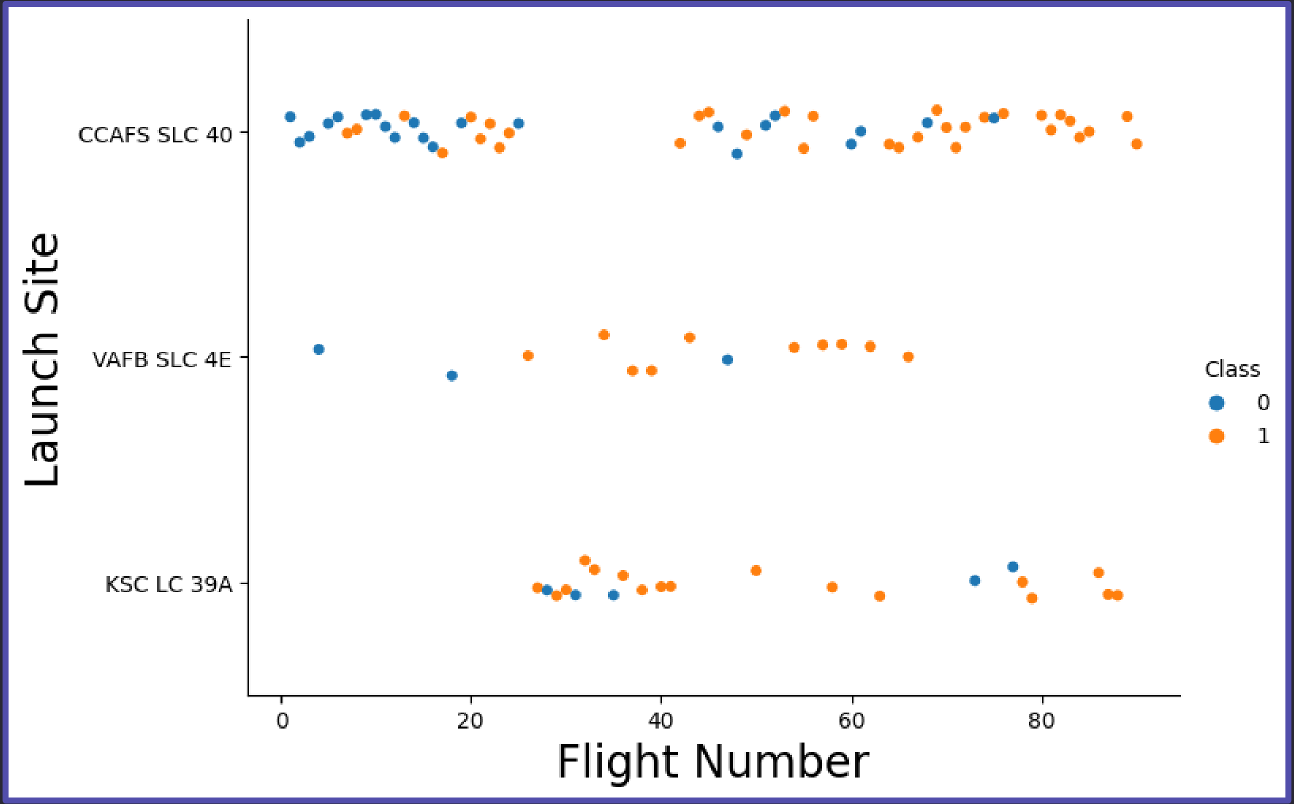
EDA WITH VISUALIZATION



LAUNCH SITE VS. FLIGHT NUMBER

The scatter plot of Launch Site vs. Flight Number shows that:

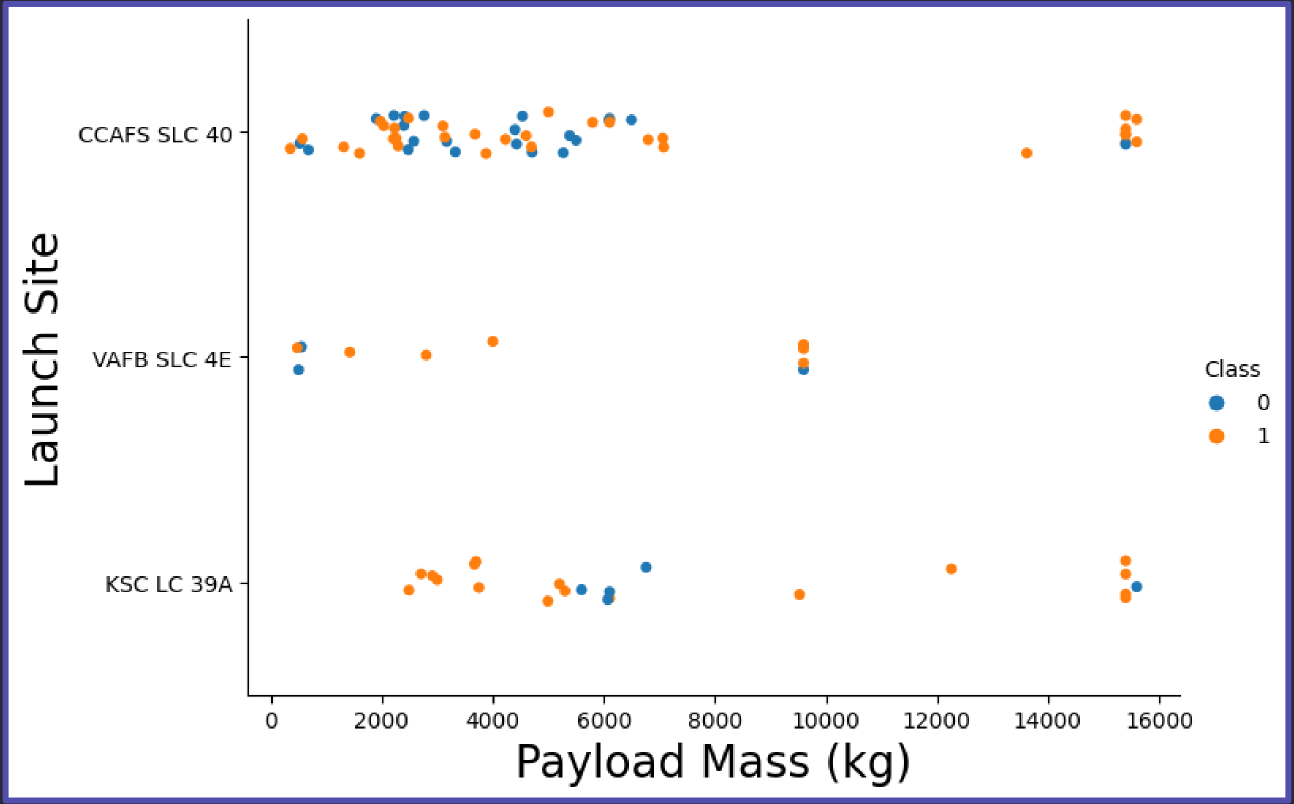
- As the number of flights increases, the rate of success at a launch site increases.
- Most of the early flights (flight numbers <30) were launched from CCAFS SLC 40, and were generally unsuccessful.
- The flights from VAFB SLC 4E also show this trend, that earlier flights were less successful.
- No early flights were launched from KSC LC 39A, so the launches from this site are more successful.
- Above a flight number of around 30, there are significantly more successful landings (Class = 1).



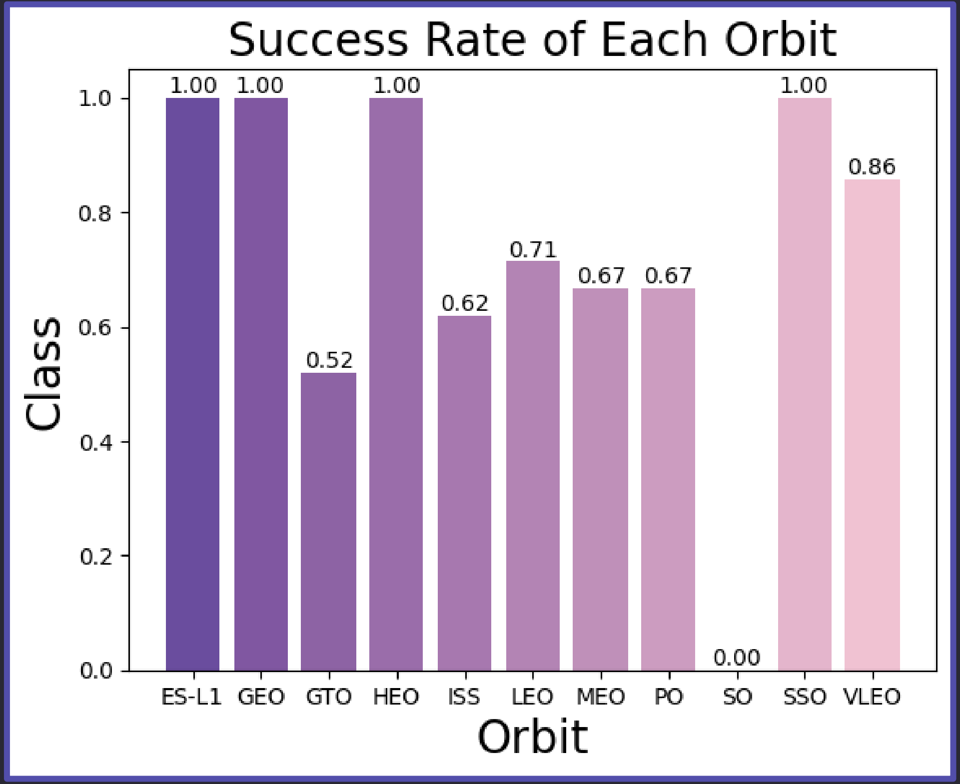
LAUNCH SITE VS. PAYLOAD MASS

The scatter plot of Launch Site vs. Payload Mass shows that:

- Above a payload mass of around 7000 kg, there are very few unsuccessful landings,
- but there is also far less data for these heavier launches.
- There is no clear correlation between payload mass and success rate for a given launch site.
- All sites launched a variety of payload masses, with most of the launches from CCAFS SLC 40 being comparatively lighter payloads (with some outliers).



SUCCESS RATE VS. ORBIT TYPE

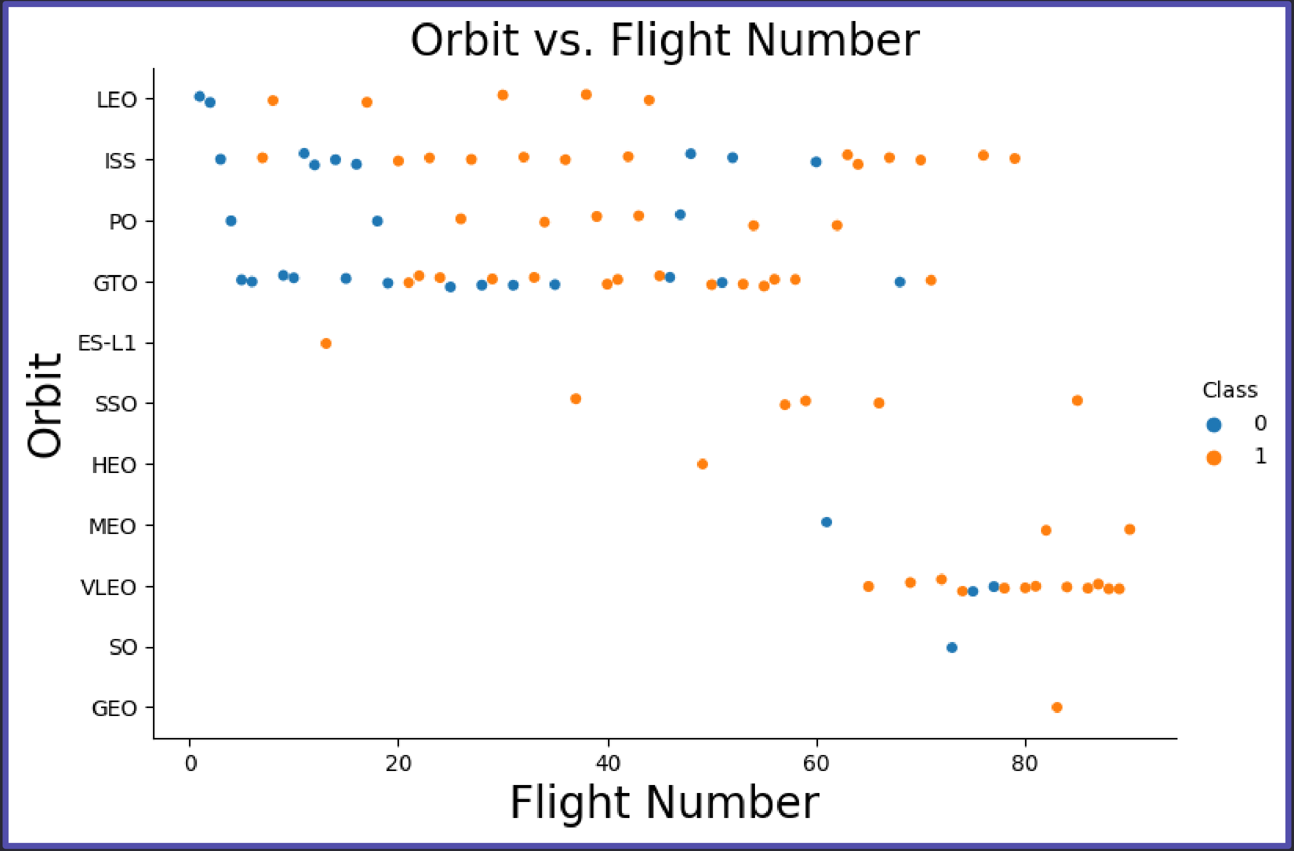


- The bar chart of Success Rate vs. Orbit Type shows that the following orbits have the highest (100%) success rate:
- ES-L1 (Earth-Sun First Lagrangian Point)
- GEO (Geostationary Orbit)
- HEO (High Earth Orbit)
- SSO (Sun-synchronous Orbit)
- The orbit with the lowest (0%) success rate is:
- SO (Heliocentric Orbit)

ORBIT TYPE VS. FLIGHT NUMBER

This scatter plot of Orbit Type vs. Flight number shows a few useful things that the previous plots did not, such as:

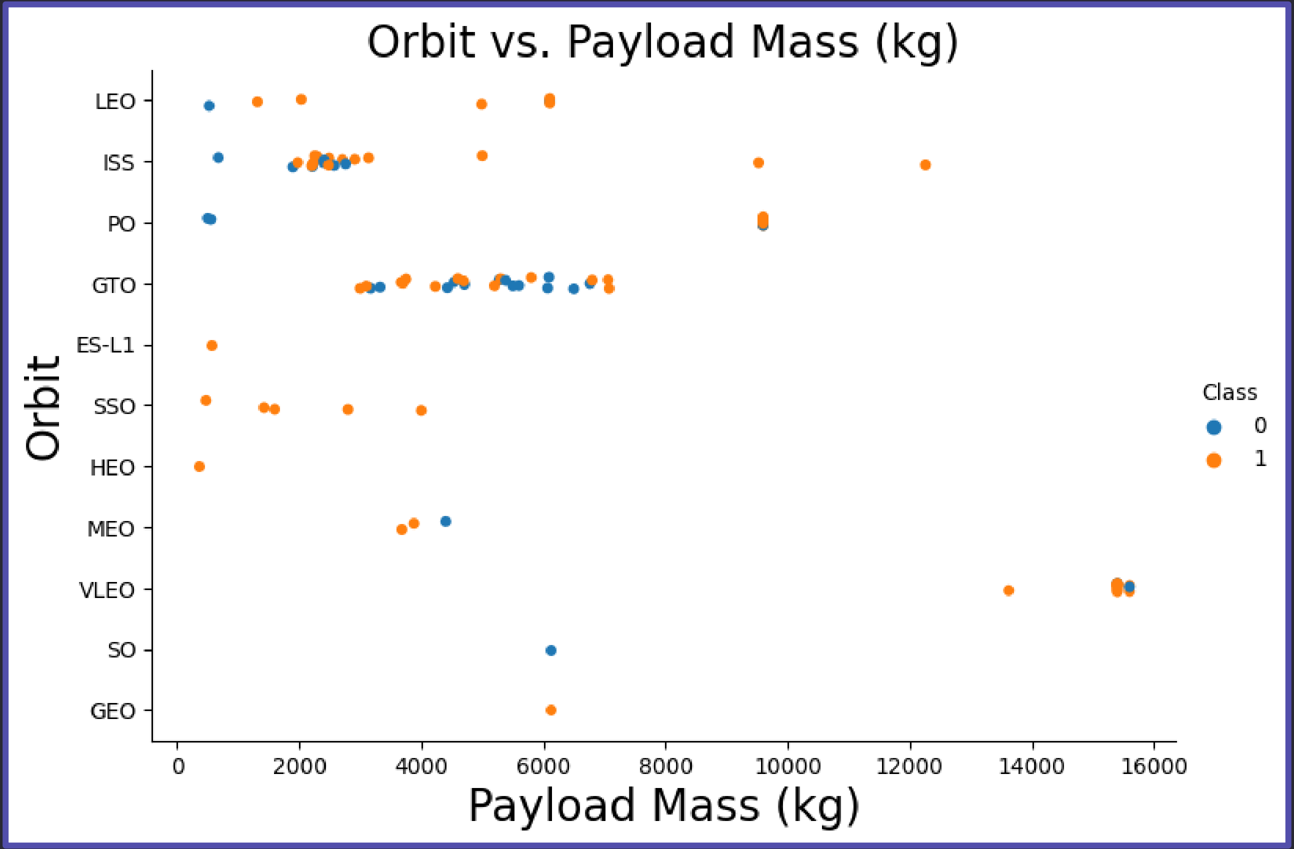
- The 100% success rate of GEO, HEO, and ES-L1 orbits can be explained by only having 1 flight into the respective orbits.
- The 100% success rate in SSO is more impressive, with 5 successful flights.
- There is little relationship between Flight Number and Success Rate for GTO.
- Generally, as Flight Number increases, the success rate increases. This is most extreme for LEO, where unsuccessful landings only occurred for the low flight numbers (early launches).



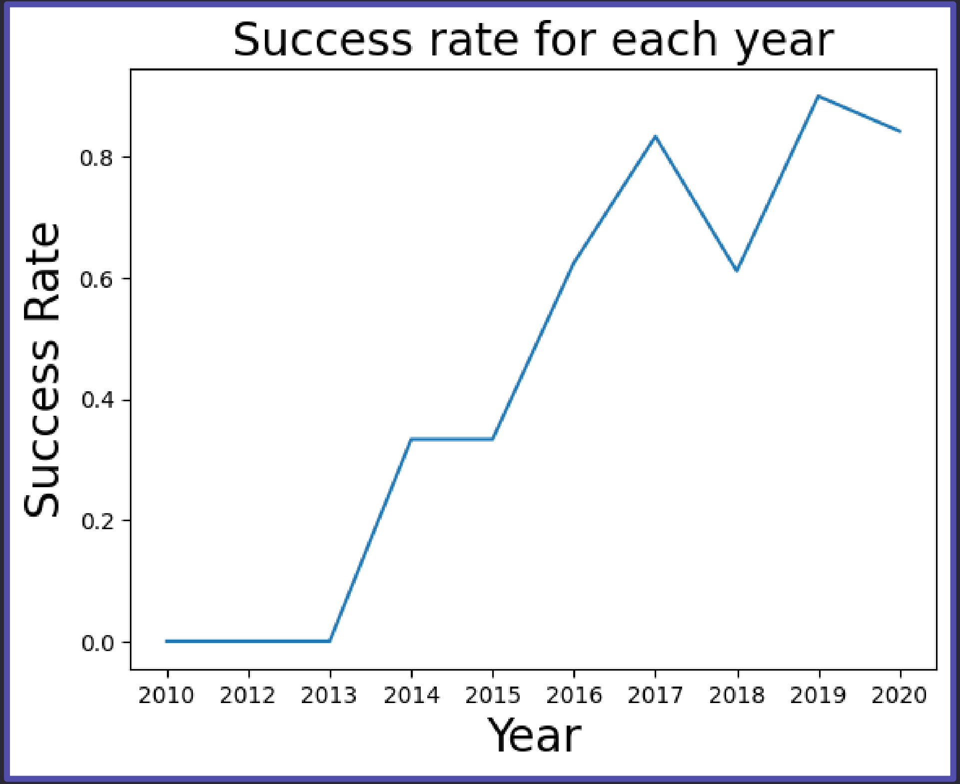
ORBIT TYPE VS. PAYLOAD MASS

This scatter plot of Orbit Type vs. Payload Mass shows that:

- The following orbit types have more success with heavy payloads:
    - PO
    - ISS
    - LEO
- For GTO, the relationship between payload mass and success rate is unclear.
- VLEO (Very Low Earth Orbit) launches are associated with heavier payloads, which makes intuitive sense.

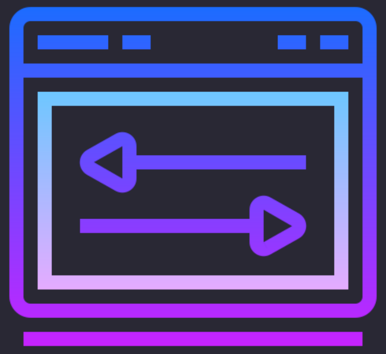


LAUNCH SUCCESS YEARLY TREND



- The line chart of yearly average success rate shows that:
- Between 2010 and 2013, all landings were unsuccessful (as the success rate is 0).
- After 2013, the success rate generally increased, despite small dips in 2018 and 2020.
- After 2016, there was always a greater than 50% chance of success.

EDA USING SQL



OVERVIEW OF THE DATASET USING SQL

1. Display the names of the unique launch sites in the space mission
2. Display 5 records where launch sites begin with the string 'CCA'
3. Display the total payload mass carried by boosters launched by NASA (CRS)
4. Display average payload mass carried by booster version F9 v1.1
5. List the date when the first successful landing outcome in ground pad was achieved.
6. List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

1

```
%%sql
SELECT DISTINCT launch_site
FROM spacextbl;
```

Launch_Site
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40
None

2

```
%%sql
SELECT launch_site
FROM spacextbl
WHERE launch_site LIKE '%CCA%'
LIMIT 5;
```

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

3

```
%%sql
SELECT SUM(payload_mass__kg_) AS total_payload_mass
FROM spacextbl
WHERE customer IS 'NASA (CRS)';
```

total_payload_mass
45596.0

4

```
%%sql
SELECT MIN(date) AS first_ground_pad_success
FROM spacextbl
WHERE landing_outcome = 'Success (ground pad)';
```

first_ground_pad_success
01/08/2018

5

```
%%sql
SELECT AVG(payload_mass__kg_) AS avg_payload_mass
FROM spacextbl
WHERE booster_version IS 'F9 v1.1';
```

avg_payload_mass
2928.4

6

```
%%sql
SELECT booster_version
FROM spacextbl
WHERE landing_outcome IS 'Success (drone ship)' AND payload_mass__kg_ BETWEEN 4000 AND 6000
GROUP BY booster_version;
```

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

OVERVIEW OF THE DATASET USING SQL

1. List the total number of successful and failure mission outcomes
2. List the names of the booster_versions which have carried the maximum payload mass using a subquery
3. List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.
4. 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.

1

```
%%sql
SELECT  mission_outcome
        ,COUNT(mission_outcome) AS total_number
FROM spacextbl
WHERE mission_outcome <> 'None'
GROUP BY mission_outcome;
```

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

2

```
%%sql
SELECT DISTINCT(booster_version)
FROM spacextbl
WHERE payload_mass__kg_ =  (SELECT MAX(payload_mass__kg_)
                            FROM spacextbl);
```

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

3

```
%%sql
SELECT  substr(date, 4, 2) AS Month
        ,landing_outcome
        ,booster_version
        ,launch_site
FROM spacextbl
WHERE landing_outcome = 'Failure (drone ship)' AND substr(date, 7, 4) = '2015';
```

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

4

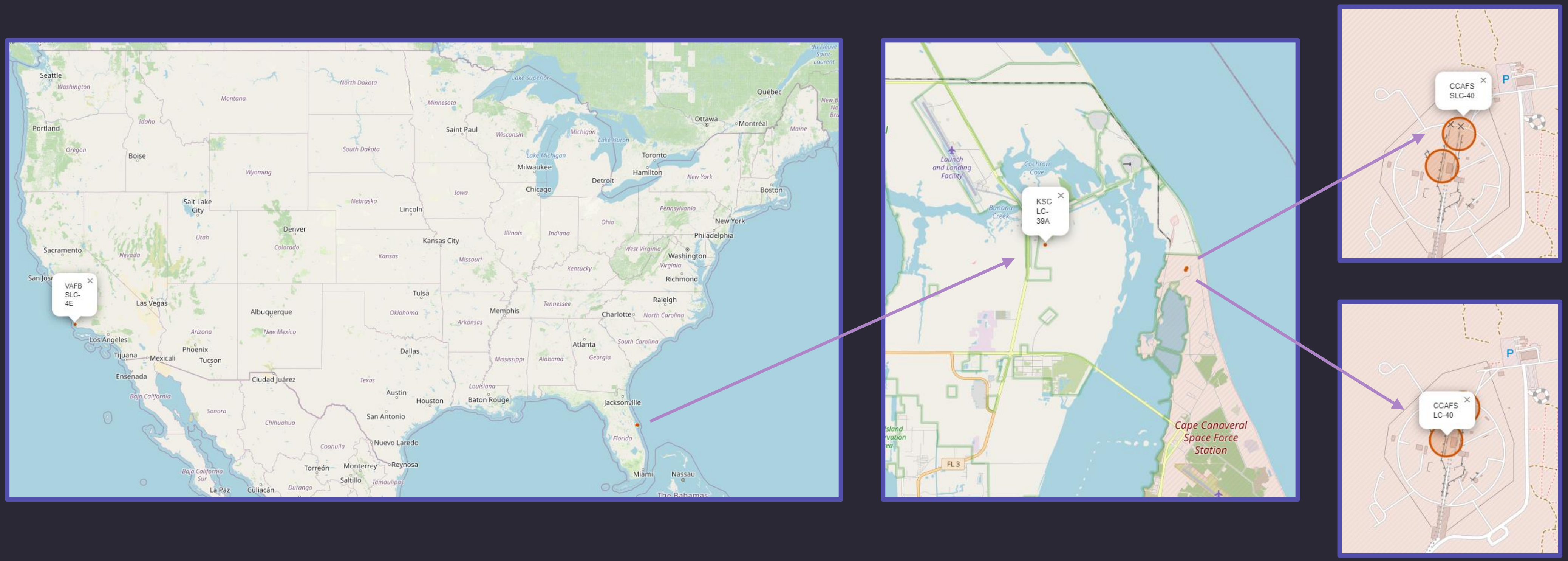
```
%%sql
SELECT  landing_outcome
        ,COUNT(landing_outcome) AS total_number
FROM SPACEXTBL
WHERE date BETWEEN '04/06/2010' AND '20/03/2020'
GROUP BY landing_outcome
ORDER BY total_number DESC;
```

Landing_Outcome	total_number
Success	20
No attempt	9
Success (drone ship)	8
Success (ground pad)	7
Failure (drone ship)	3
Failure	3
Failure (parachute)	2
Controlled (ocean)	2
No attempt	1

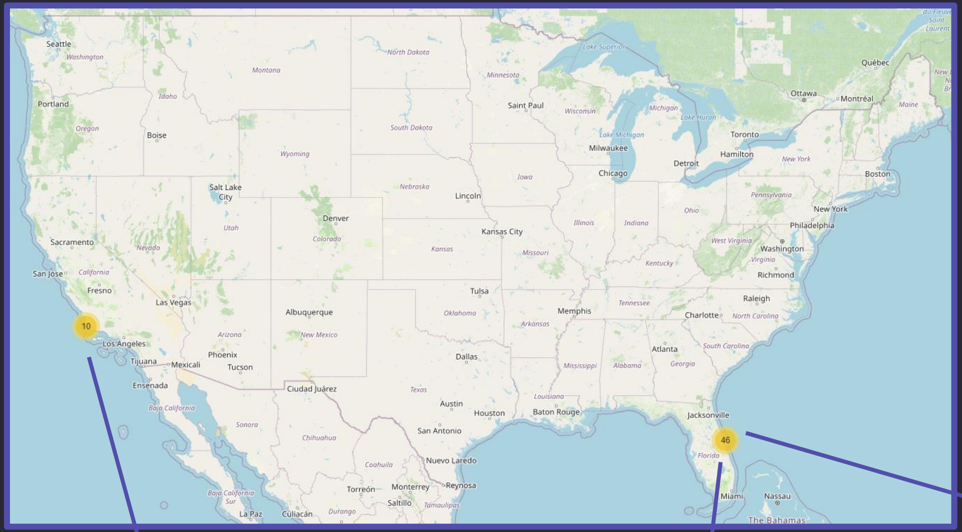
LAUNCH SITES PROXIMITY ANALYSIS:
FOLIUM INTERACTIVE MAP

ALL LAUNCH SITES ON A MAP

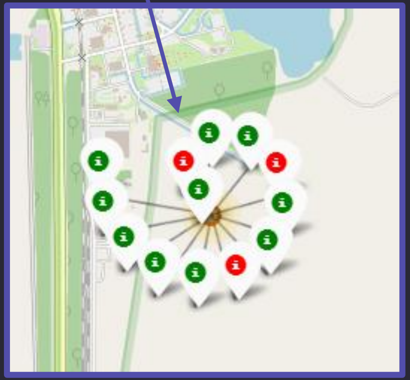
All SpaceX launch sites are on the coasts of the United States, specifically Florida and California.



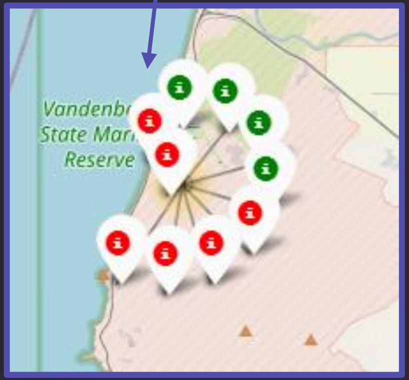
SUCCESSFUL AND FAILED LAUNCHES FOR EACH SITE



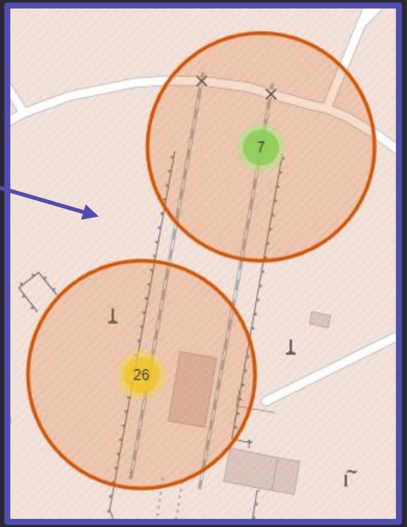
Launches have been grouped into clusters, where green icons are successful launches, and red icons are failed launches.



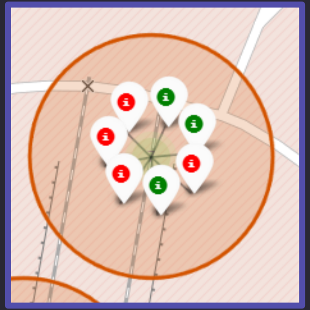
VAFB SLC-4E



KSC LC-39A



=



CCAFS SLC-40

CCAFS LC-40

PROXIMITY OF LAUNCH SITES TO POINTS OF INTEREST

Using CCAFS SLC-40 as an example site, the following questions can be asked to understand more about the placement of launch sites:

Are launch sites in close proximity to railways?

- Yes, the coastline is only 0.87 km due East.

Are launch sites in close proximity to highways?

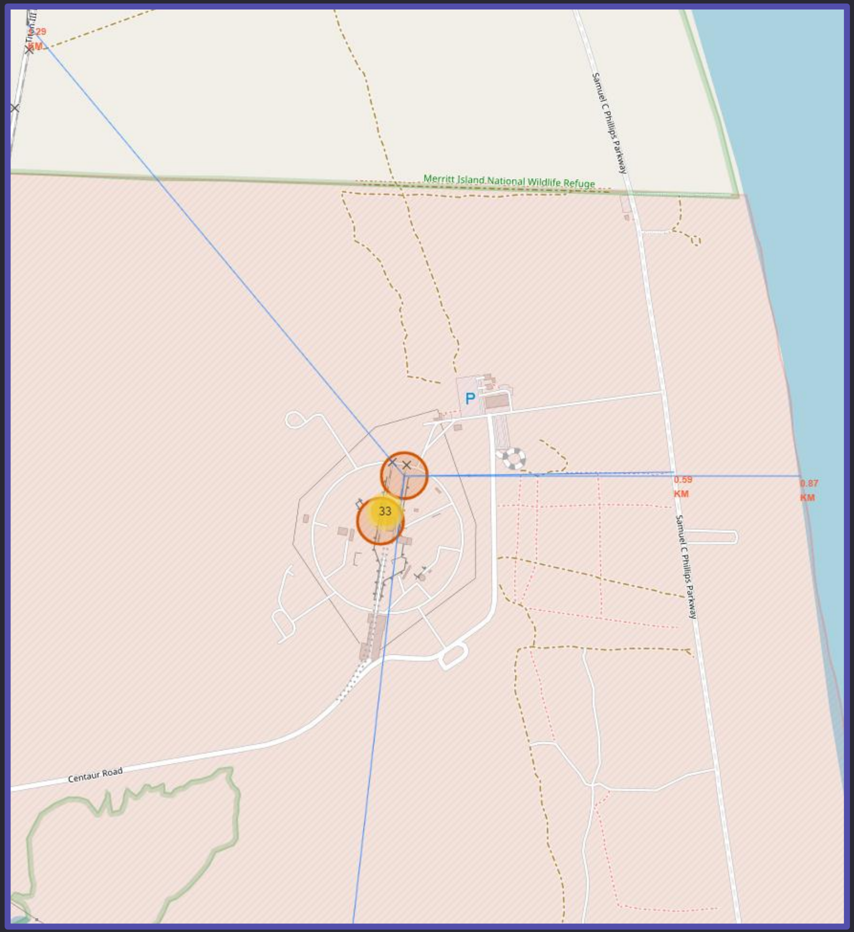
- Yes, the nearest highway is only 0.59 km away.

Are launch sites in close proximity to railways?

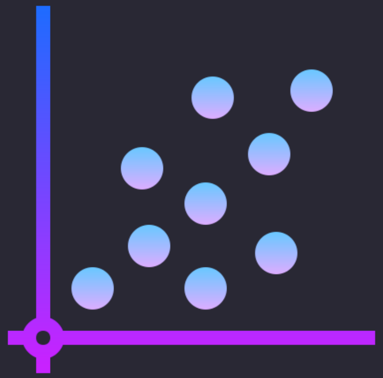
- Yes, the nearest railway is only 1.29 km away.

Do launch sites keep a certain distance away from cities?

- Yes, the nearest city is 51.74 km away.



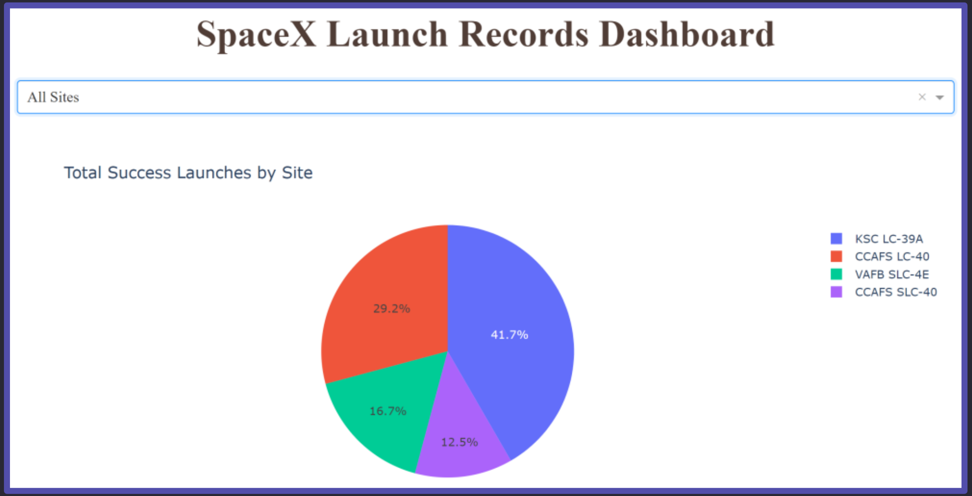
INTERACTIVE DASHBOARD ANALYSIS:
PLOTLY DASH

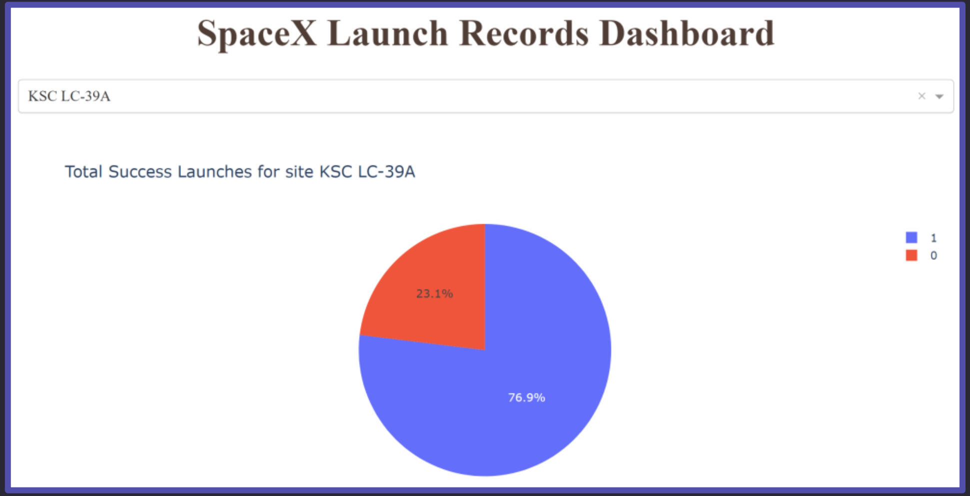


LAUNCH SITE SUCCESS RATES

The launch site KSC LC-39 A had the most successful launches, with 41.7% of the total successful launches.

The launch site KSC LC-39 A also had the highest rate of successful launches, with a 76.9% success rate.





SCATTER PLOT (ALL SITES):
LAUNCH OUTCOME VS. PAYLOAD MASS

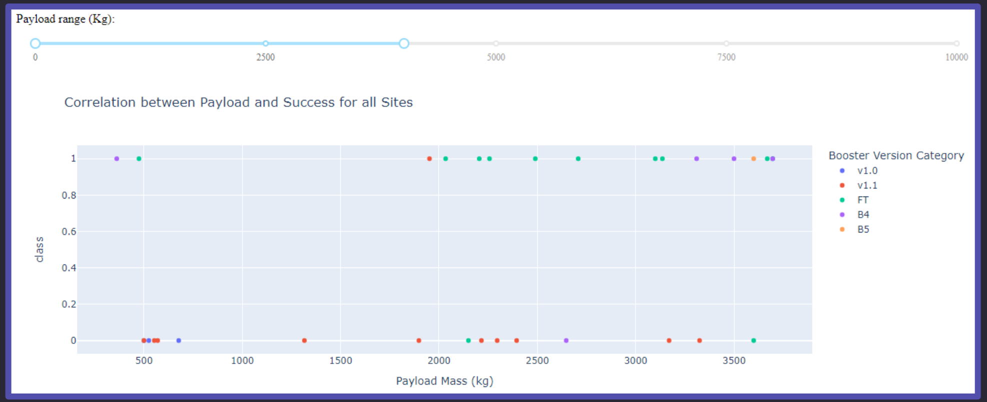
Plotting the launch outcome vs. payload mass for all sites shows a gap around 4000 kg, so it makes sense to split the data into two ranges:

1. 0 – 4000 kg (low payload mass)
2. 4001 – 10000 kg (high payload mass)

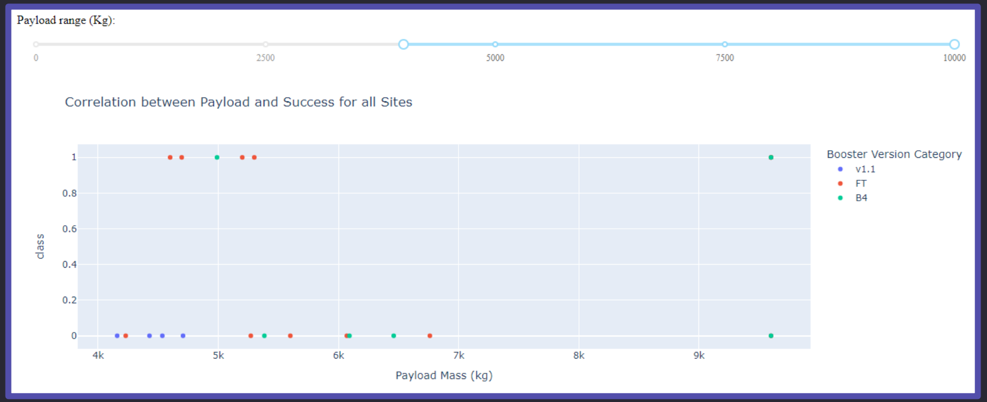
From these two plots, it can be observed that the success for higher payload mass is lower than that for lower payload mass.

It is also worth noting that some booster types (v 1.0 and B5) have not been launched with high payload mass.

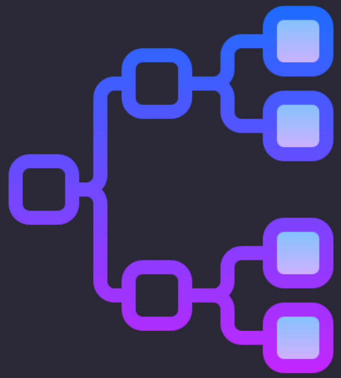
1



2



PREDICTIVE ANALYSIS
USING CLASSIFICATION



CLASSIFICATION ACCURACY

Plotting the Accuracy Score and Best Score for each classification algorithm produces the following result:

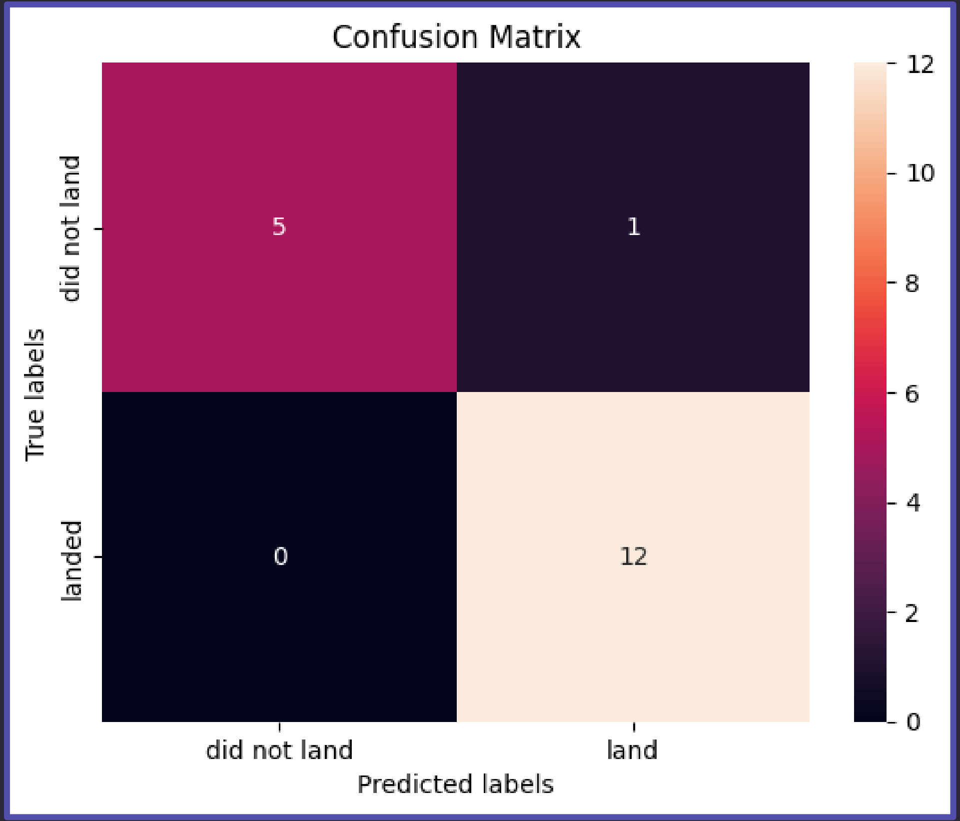
- The K-Nearest Neighbors model has the highest classification accuracy
  1. The Accuracy Score is 94.44%
  2. The Best Score is 88.75%

	Algorithm	Accuracy Score	Best Score
0	Logistic Regression	0.833333	0.846429
1	Support Vector Machine	0.833333	0.848214
2	Decision Tree	0.833333	0.864286
3	K Nearest Neighbours	0.944444	0.887500

1

2

CONFUSION MATRIX



- As shown previously, best performing classification model is the K-Nearest Neighbors model, with an accuracy of 94.44%.
- This is explained by the confusion matrix, which shows only 1 out of 18 total results classified incorrectly (a false positive, shown in the top-right corner).
- The other 17 results are correctly classified (5 did not land, 12 did land).

CONCLUSIONS

- As the number of flights increases, the rate of success at a launch site increases, with most early flights being unsuccessful (i.e. with more experience, the success rate increases.)
  - Between 2010 and 2013, all landings were unsuccessful (as the success rate is 0).
  - After 2013, the success rate generally increased, despite small dips in 2018 and 2020.
  - After 2016, there was always a greater than 50% chance of success.
- Orbit types ES-L1, GEO, HEO, and SSO, have the highest (100%) success rate.
  - The 100% success rate of GEO, HEO, and ES-L1 orbits can be explained by only having 1 flight into the respective orbits.
  - The 100% success rate in SSO is more impressive, with 5 successful flights.
  - The orbit types PO, ISS, and LEO, have more success with heavy payloads:
    - VLEO (Very Low Earth Orbit) launches are associated with heavier payloads, which makes intuitive sense.
- The launch site KSC LC-39 A had the most successful launches, with 41.7% of the total successful launches, and also the highest rate of successful launches, with a 76.9% success rate.
- The success for massive payloads (over 4000 kg) is lower than that for low payloads (this means a smaller payload translates to a higher success rate).
- The best performing classification model is the K-Nearest Neighbors model, with an accuracy of 94.44%.