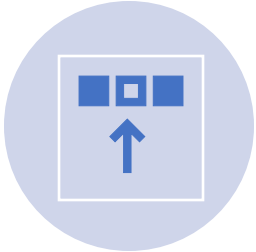
IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Marvin Duran
April 2024

Outline



EXECUTIVE SUMMARY

INTRODUCTION

METHODOLOGY

RESULTS

CONCLUSION

APPENDIX

Executive Summary:
Enhancing SpaceX's Landing Success Prediction

This is a comprehensive data-driven journey to improve the accuracy of predicting successful rocket landings. Here are the key insights:

Summary of Methodologies:

This 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 outputs and visualizations:

1. Exploratory Data Analysis (EDA) results
2. Geospatical analysis
3. Interactive dashboard
4. Predictive analysis of classification models

Insights and Recommendations

- Predictive Power: Leveraging discernible patterns and correlations, this predictive model provides reliable forcecasts for successful landings.

Predicting SpaceX's First-Stage Rocket Retrieval Success

Project Objective

Our mission: **Forecast the Success of First-Phase Rocket Retrieval**. Why? Because this predictive capability holds the key to optimizing resource allocation and enhancing mission success rates.

Cost Savings and Competitive Edge

- **SpaceX's Advantage**: Their Falcon 9 rocket launches, priced at $62 million, outshine competitors (costing upward of $165 million) due to first-stage reusability.
- **Bid Wars**: Armed with our predictions, alternate companies can strategically bid against SpaceX for rocket launches.

This project will ultimately **predict if the Space X Falcon 9 first stage will land successfully.**

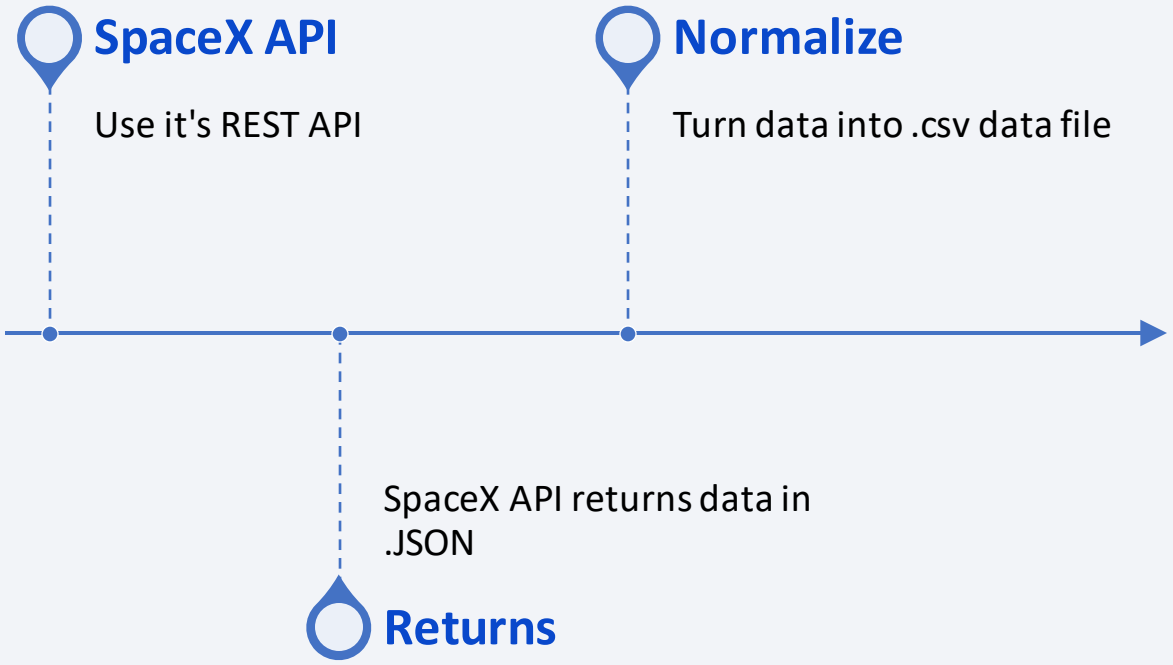
Section 1

Methodology

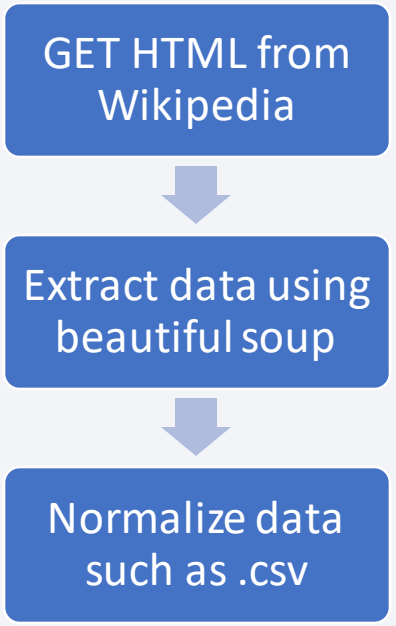
Data Collection

- Gather data from SpaceX Rest API
- Scrape data from Wikipedia about Falcon 9 launches

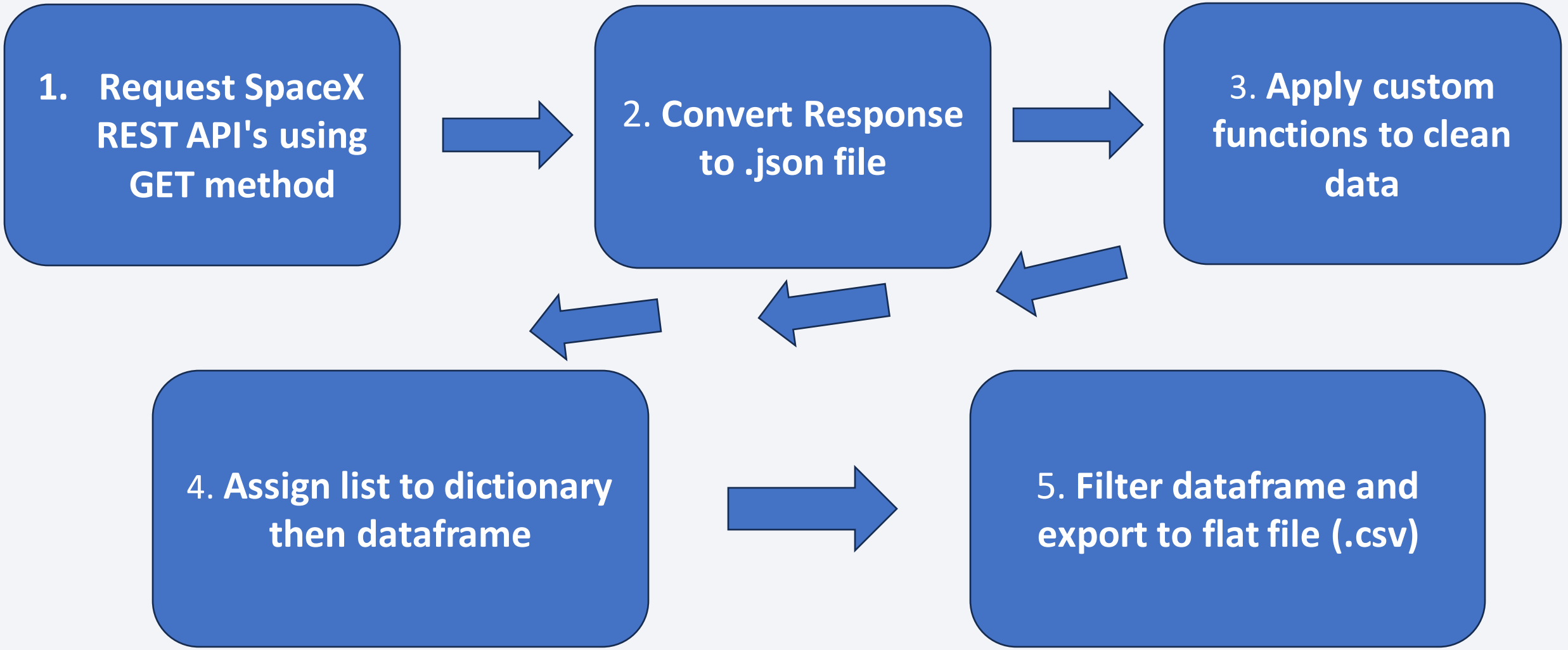
SpaceX API



Web Scraping

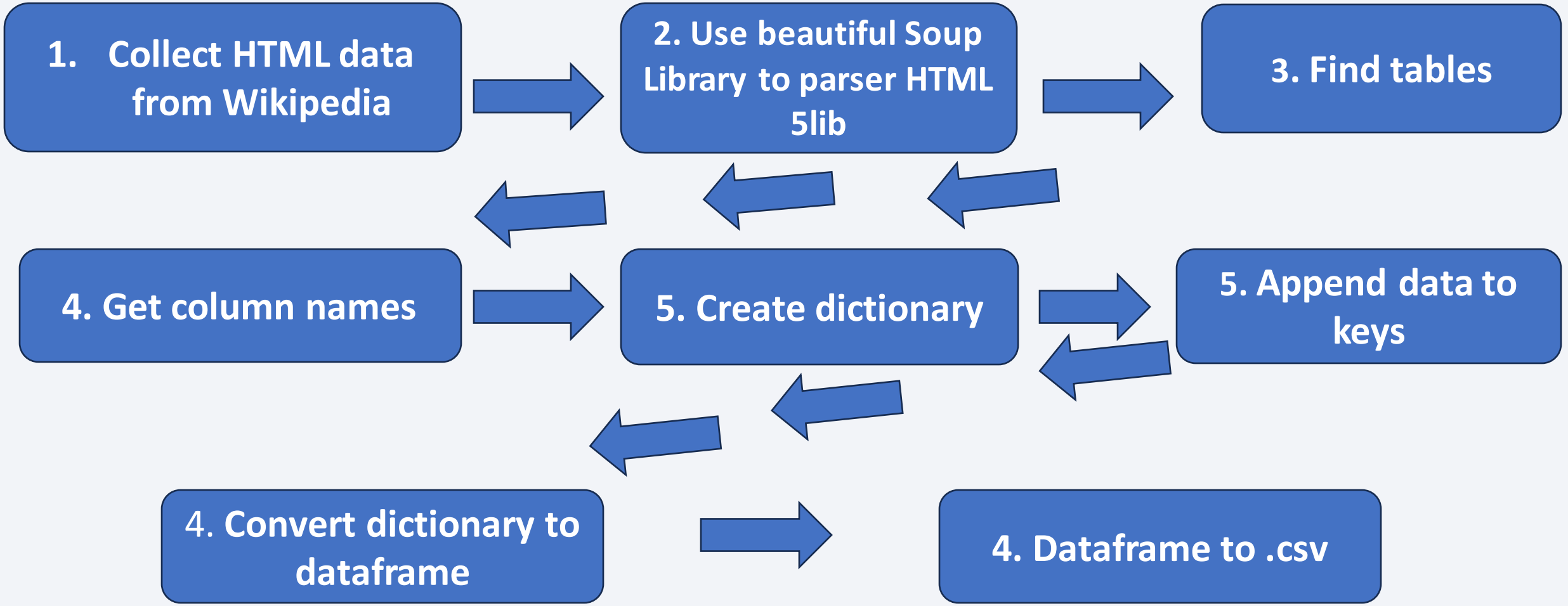


Data Collection – SpaceX API



GitHub URL

Data Collection – Scraping

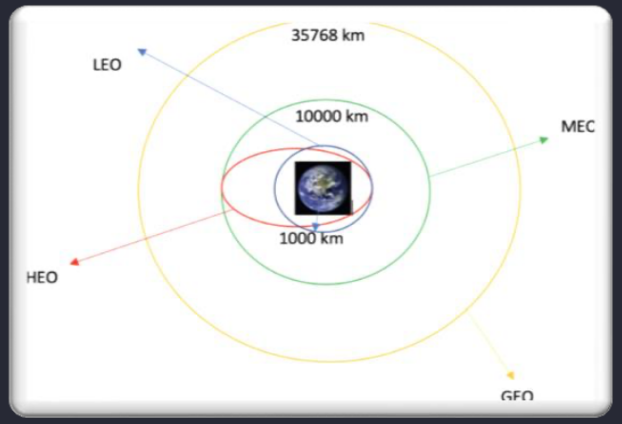


GitHub URL

DATA MANIPULATION/WRANGLING – PANDAS

Context:

- The SpaceX dataset contains several Space X launch facilities, and each location is in the `LaunchSite` column.
- Each launch aims to a dedicated orbit, and some of the common orbit types are shown in the figure below. The orbit type is in the `Orbit` column.



Initial Data Exploration:

- Using the `.value_counts()` method to determine the following:
  1. Number of launches on each site
  2. Number and occurrence of each orbit
  3. Number and occurrence of landing outcome per orbit type

1

```
# 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
```

2

```
# 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
GEO       1
SO        1
HEO       1
Name: Orbit, dtype: int64
```

3

```
# 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
None ASDS       2
False Ocean     2
False RTLS      1
Name: Outcome, dtype: int64
```

Data Manipulation/ Wrangling - PANDAS

GitHub URL

Context:

- 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.

Data Wrangling:

- 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 success of the landing.
- This is done by:
  1. Defining a set of unsuccessful (bad) outcomes, `bad_outcome`
  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.
  3. Create a `Class` column that contains the values from the list `landing_class`
  4. Export the DataFrame as a .csv file.

1

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

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

3

```
# 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)
```

3

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

4

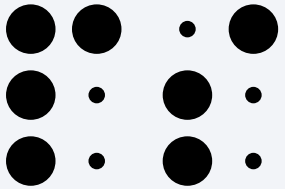
```
df.to_csv("dataset_part\_2.csv", index=False)
```

EDA with Data Visualization

Scatterplot:

Scatter plot is a great tool to depict correlation.

- **FlightNumber vs. PayloadMass**
- **FlightNumber vs LaunchSite**
- **FlightNumber vs Orbit type**
- **Payload vs Launch Site**
- **Payload vs Orbit type**



Bar Chart:

Bar chart compares the measure of categorical dimension

- **Success rate vs Orbit type**

Line Chart:

Display trends and developments of numeric data over time.

- **Launch success yearly trend**

GitHub URL

EXPLORATORY DATA ANALYSIS (EDA) with SQL

Summary of the SQL queries performed

- **SQL'eled Unique Launch Sites**
  - Kennedy Space Center(Florida)
  - Cape Cavaeral Space Force Station(Florida)
  - Vandenber Space Force Base Station(California)
  - Wallops Flight Facility (Virginia)
  - Reagan Test Site – Kwajalein Atoll(Pacific Ocean)
- Boosters with Drone Ship Success and Payload Mass Criteria
- Total Number of Succesful and Failure Mission Outcomes
- Total Payload Mass Carried by NASA Boosters
- Average Payload Mass Carried by Booster Version F9
- First Successful Ground Pad Landing Date

GitHub URL

GEOSPATIAL ANALYSIS Map with Folium

- **Launch sites marked on Folium interactive map**
  - Each site was marked with a circle
  - Used latitude and longitude coordinates of each launch site
- **Successful and failed launches marked**
  - Launch outcomes
    - Failure = 0 **Red**
    - Success = 1 **Green**

GitHub URL

- **Distances calcualted between launch sites and its proximities**
  - Calculated proximities such as coastines and highways.

Interactive Dashboard with Plotly Dash

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

Graphs

- **Scatter plot displays the relationship Payload Mass (kg) vs Launch Outcome for the different booster versions**
  - **Shows correlation**
  - **Range of data, maximum and minimum value can be filtered**
- **Pie Chart shows the total launches by a specific site or all**
  - Displayed relative proposrtions of multiple classes of data
  - Size of the circle can be made proportional to the toal quantity it represents

Predictive Analysis - 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:

- Load dataset
- Perform necessary data transformations(standardise and pre-process)
- Split data into training and test data sets, using train_test_split()
- Decide Which type of machine learning algorithms are most appropriate

For each chosen algorithm:

- Create a GridSearchCV object and a dictionary of parameters
- Fit the object to the parameters
- 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_)

Finding the best Classification Model

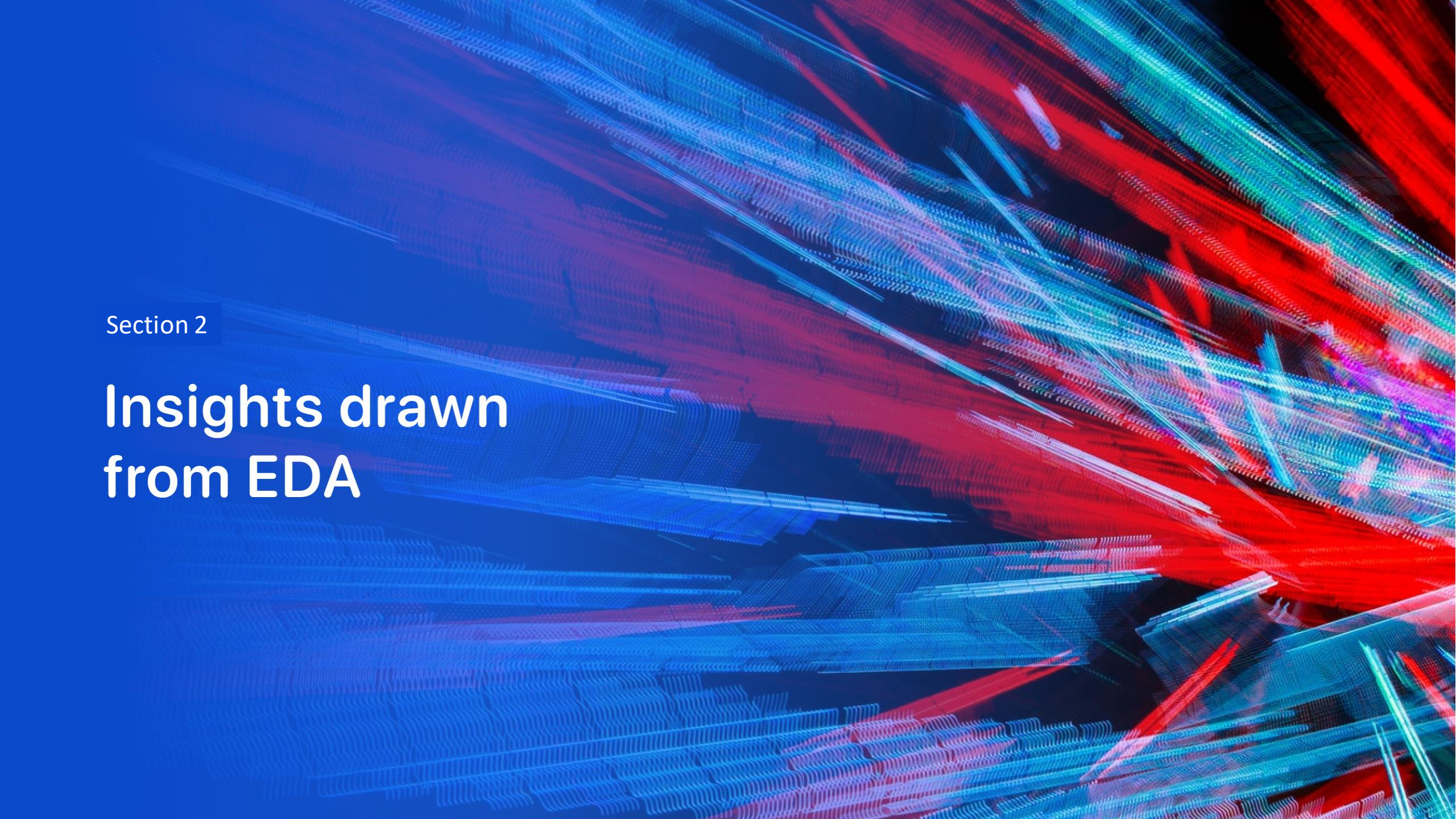
- Review the accuracy scores for all chosen algorithms.
- The model with the highest accuracy score is determined aas the best performing model

Results

- Exploratory Data Analysis
- Interactive Analytics
- Predictive Analysis

Section 2

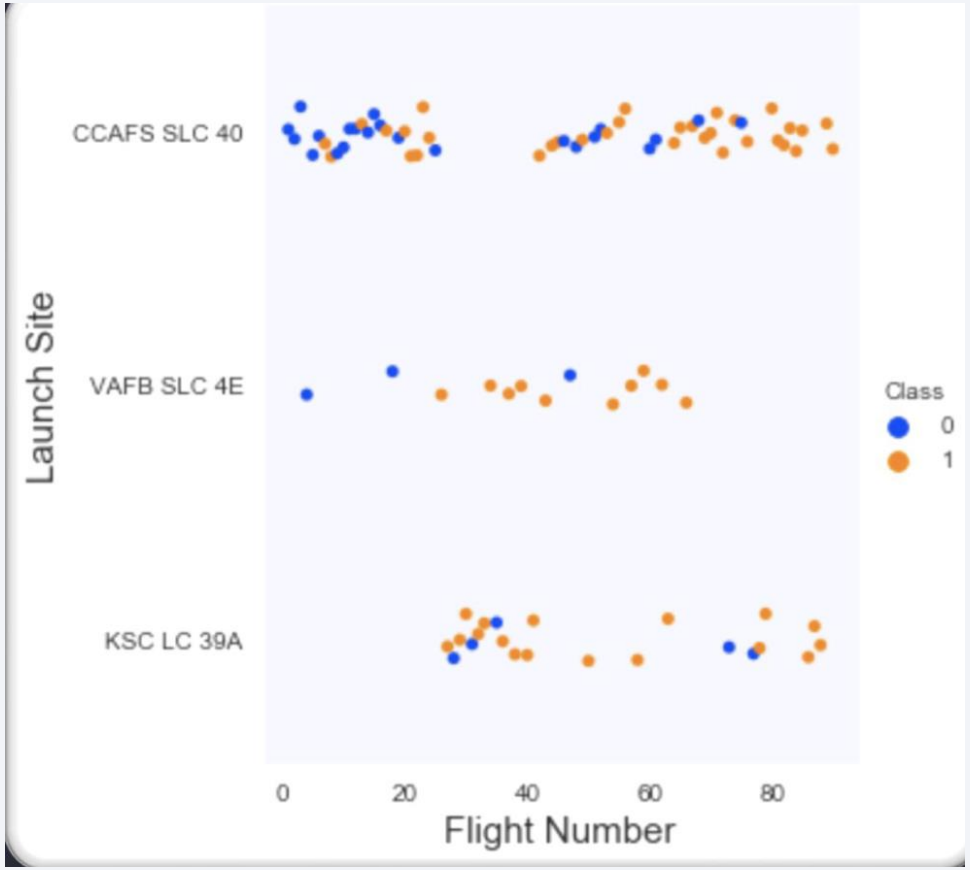
Insights drawn from EDA



Flight Number vs. Launch Site

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

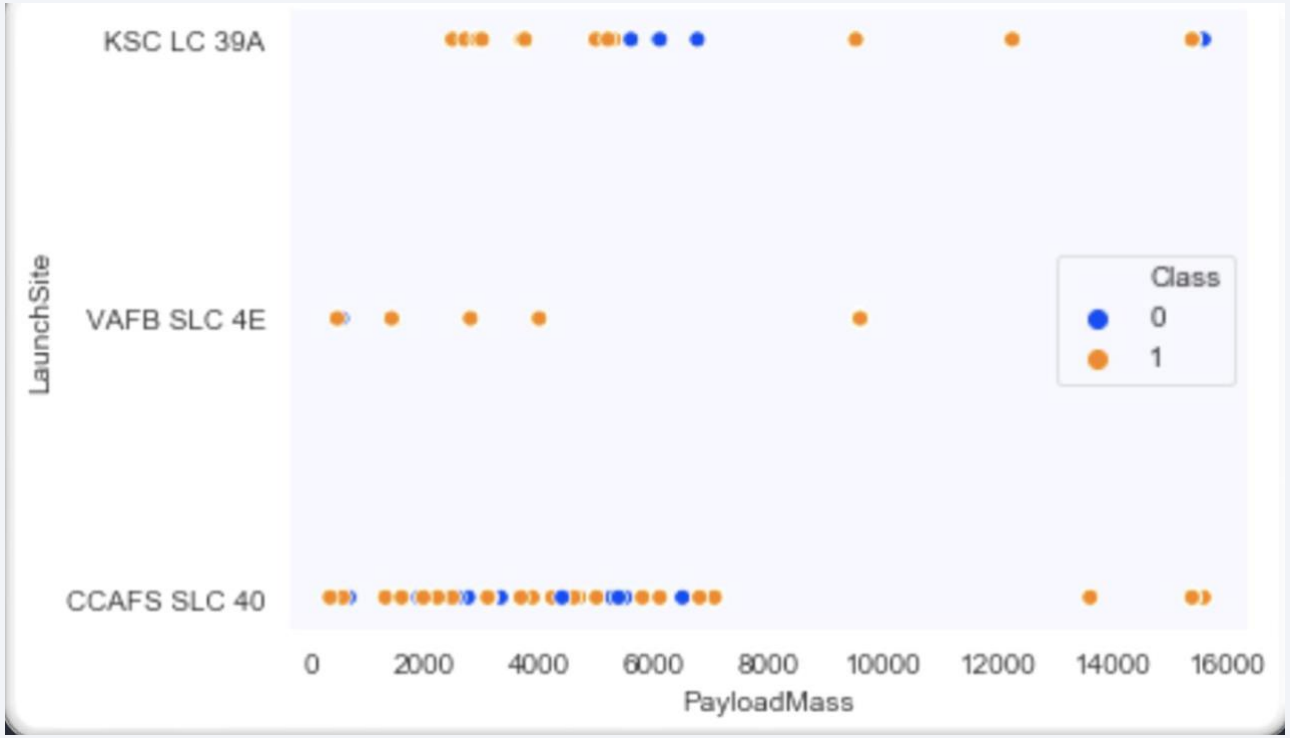
- As the number of flights increases, the rate of success at a launch site increases.
- Most of the early flights (flight number <30) were launched from CCAFS SLC 40, and were generally unsuccessful.
- The flight from VAFB SLC 4E show this trend, that earlier flights were less successful.
- No early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



Payload vs. Launch Site

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

- Above a payload mass of around 7000kg, 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).



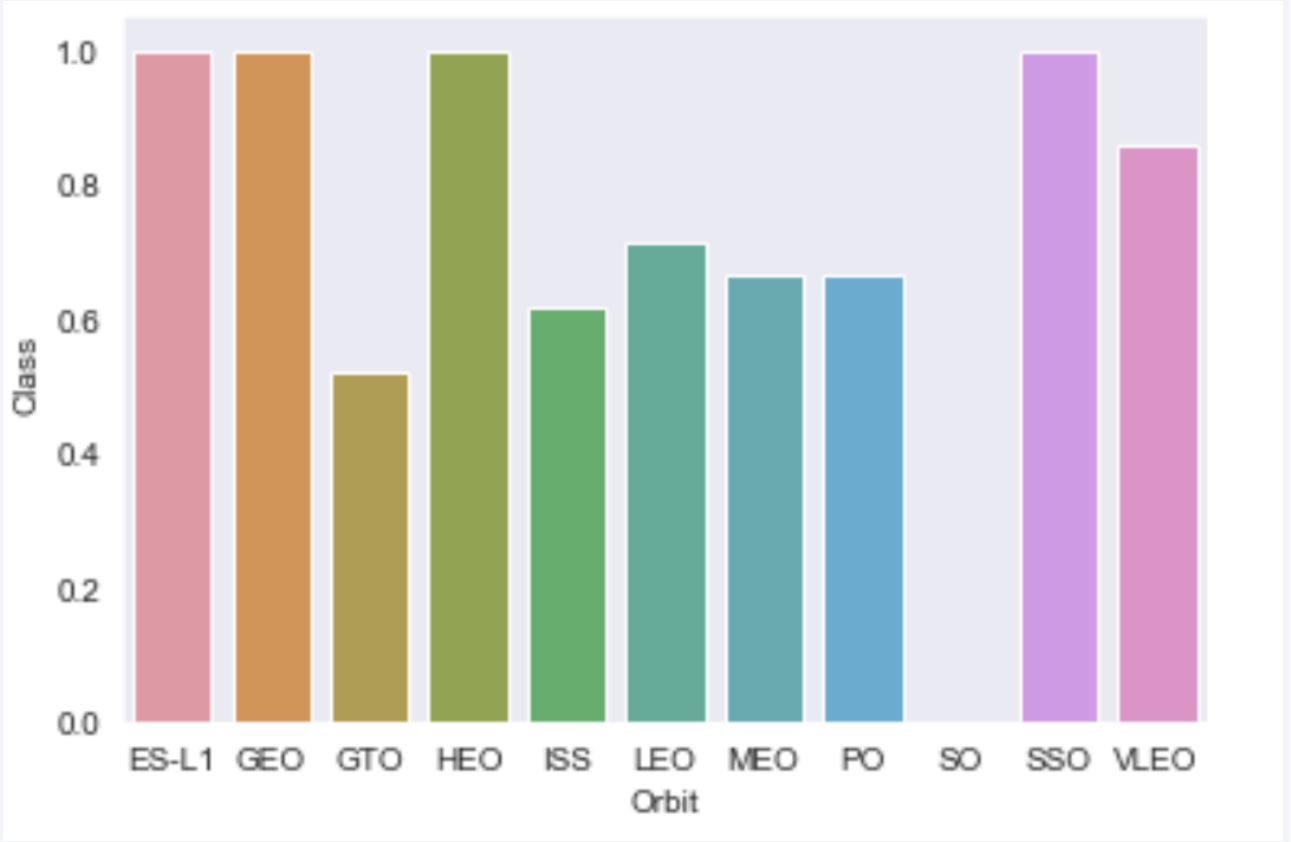
Success Rate vs. Orbit Type

The bar chart of Success Rate vs. Orbit Typs 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:

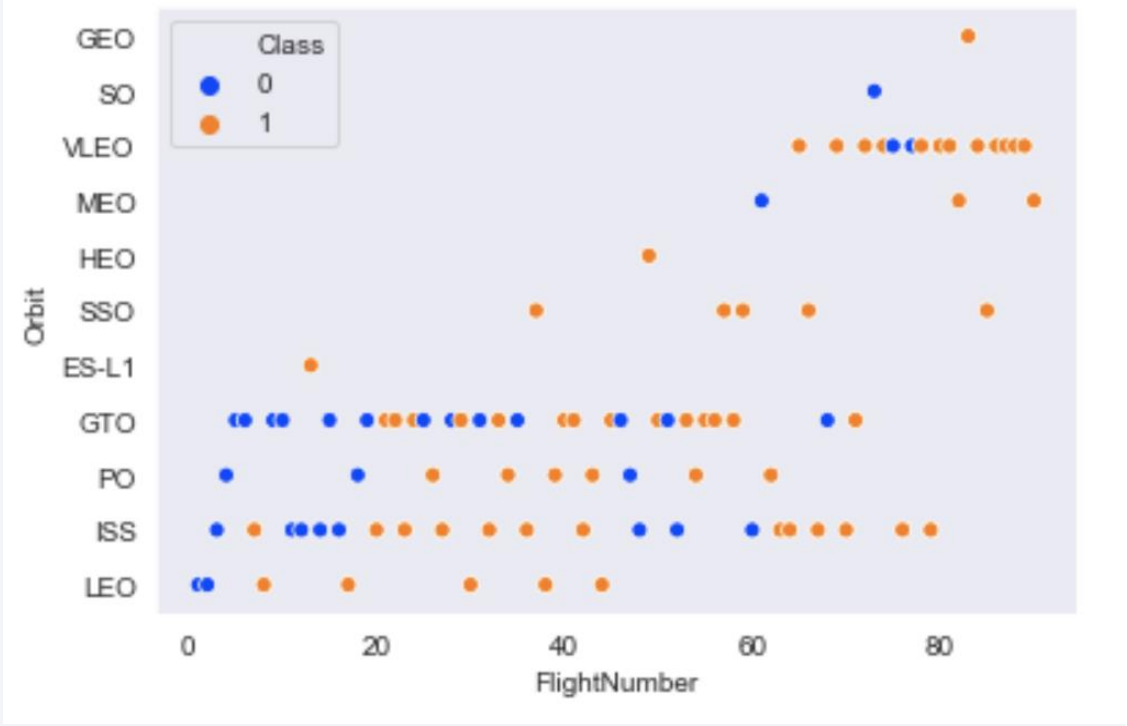
- SO (Heliocentric Orbit)



Flight Number vs. Orbit Type

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-L 1 orbits can be explained by only having 1 flight into the respective orbits.
- The 100 % success rate in sso is more impresive with 5 successful flights.
- There is litle relationshops between Flight Number and success rate for GTO.
- Generally, as Flight Number Increases, the success rate increases. This is most extreme LEO, where unsuccessful landing only occurred the the low flight numbers (early launches).



Payload vs. Orbit Type

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

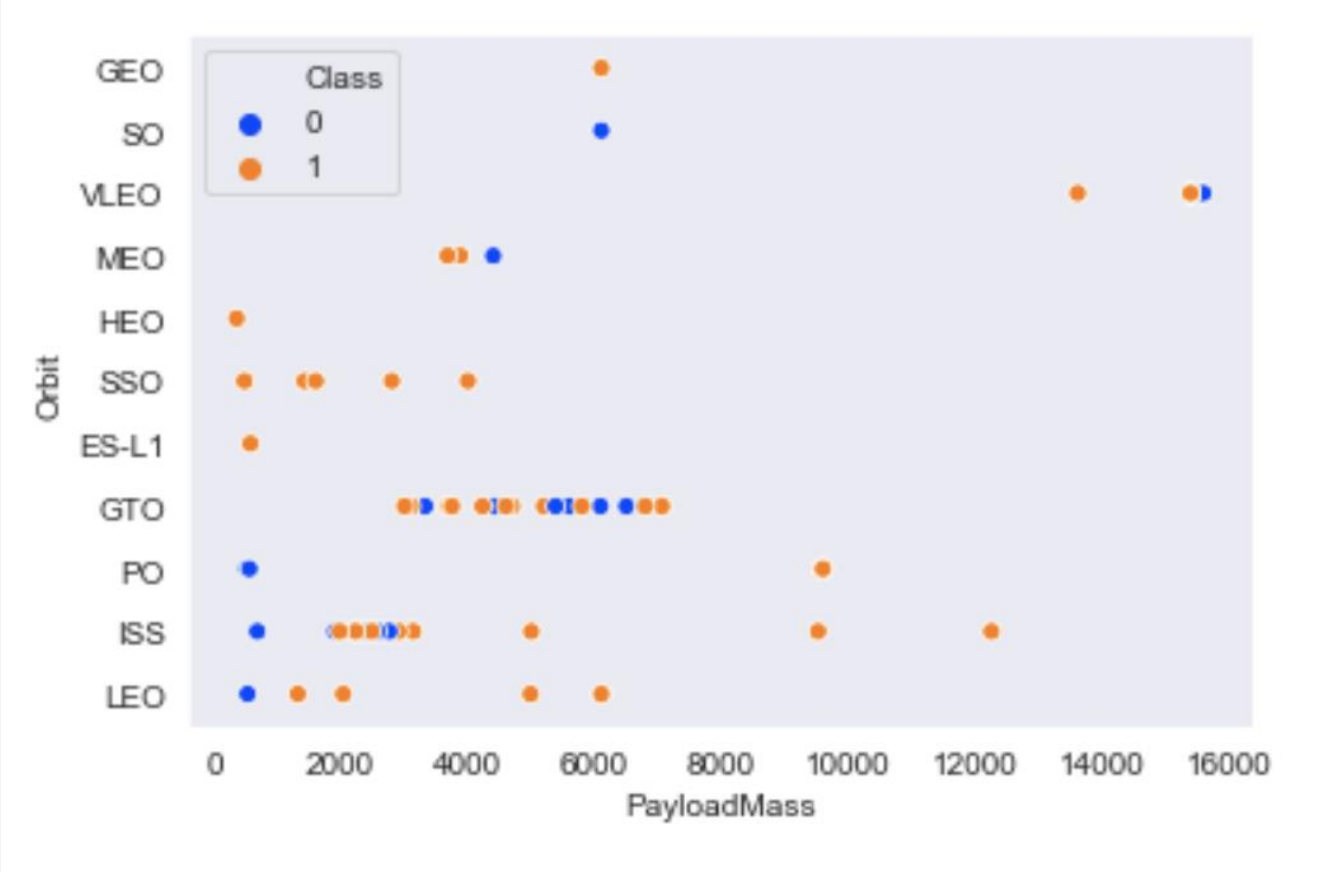
- The following orbit types have more success with heavy payloads:

  PO (although the number of data points is small)

  ISS

  LEO

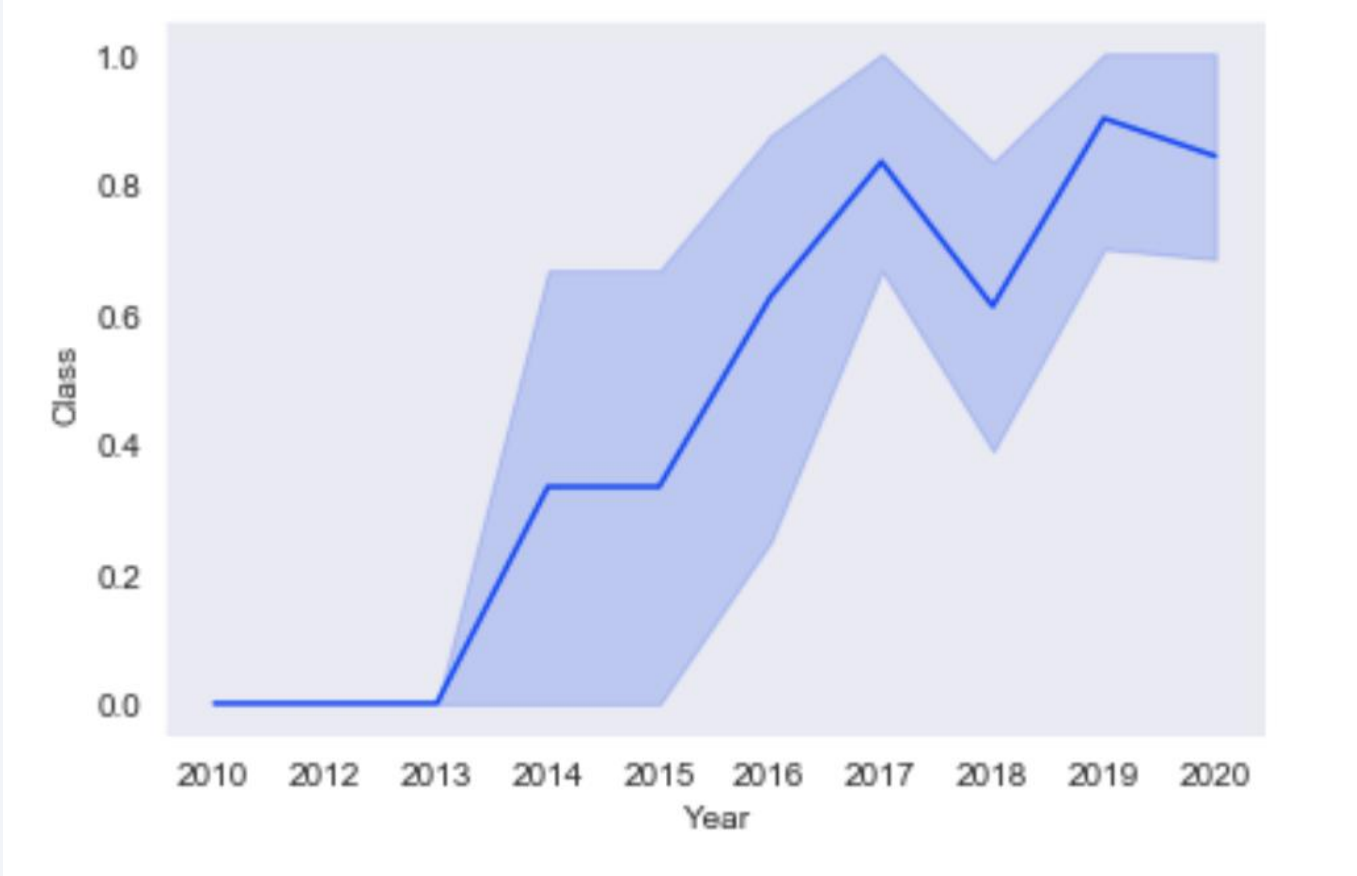
- For GTO, the relationship between paylaod mass and success rate is unclear
- VELO (Very Low Earth Orbit) launches are associated with heavier payloads, which makes intuitive sense.



Launch Success Yearly Trend

The line chart of yearly average success rate shows that:

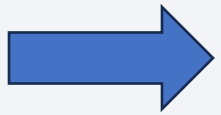
- Between 2010 and 2013, all landings were unsuccessful (as the success rate is 0).
- After 2013, the success rate generally increased, despite small dips in 2018 and 2020.
- After 2016, there was always a greater than 50% chance of success.



All Launch Site Names

Find the names of the unique launc sites.

```
% sql SELECT UNIQUE (LAUNCH_SITE) FROM SPACEXTBL;
```



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

The word **UNIQUE** returns only unique values from the **LAUNCH_SITE** column of the **SPACEXTBL** table.

Launch Site Names Begin with 'CCA'

Find 5 records where launch sites begin with `CCA`

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

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

Query Explanation

Using the word limit 5 in the query means that it will only show 5 records from SpaxeXTBL and LIKE keyword has a wild card to suggest that the Launch_Site name must start with CCA

Total Payload Mass

Calculate the total payload carried by boosters from NASA

```
%sql SELECT SUM(PAYLOAD_MASS_KG_) AS TOTAL_PAYLOAD_MASS FROM SPACEXBL \ WHERE
CUSTOMER = 'NASA' (CRS)';
```

Total_payload_mass
45596

The SUM KEYWORD is sued to calculate the total of the LAUNCH column, and the SUM keyword (and the associated condition) filters the results to only boosters from NASA (CRS).

Average Payload Mass by F9 v1.1

Calculate the average payload mass carried by booster version F9 v1.1

```
%sql SELECT AVG(PAYLOAD_MASS_KG_) AS AVERAGE_PAYLOAD_MASS FRO SPACEXTBL \
   WHERE BOOSTER_VERSION = 'FY V1.1';
```

Average_payload_mass

2928

The AVG keyword is used to calculate the average of the PAYLOAD_MASS_KG column, and the WHERE keyword (and the associated condition) filters the results to only the F9 v1.1 booster version.

First Successful Ground Landing Date

Find the dates of the first successful landing outcome on ground pad

```
%sql SELECT MIN(DATE) AS FIRST_SUCCESSFUL_GROUND_LANDING FROM SPACEXTBL \
    WHERE LANDING_OUTCOME = 'Success (ground pad)';
```

First_successful_ground_landing
2015-12-22

The MIN keyword is used to calculate the minimum of the DATE column, i.e. the first date, and the WHERE keyword (and the associated condition) filters the results to only the successful ground pad landings.

Successful Drone Ship Landing with Payload between 4000 and 6000

List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

```
%sql SELECT BOOSTER)VERSION FROM SPACEXTBL \ WHERE (LANDING_OUTCOME = 'Success
(drone ship)' AND (PAYLOAD_MASS_KG_ BETWEEN 4000 AND 6000);
```

Booster_version

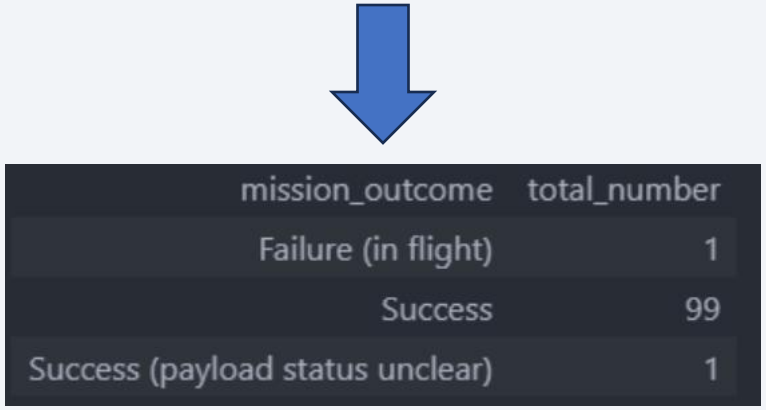
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

The WHERE keyword is used to filter the results to include only those that satisfy both conditions in the brackets (as the AND keyword is also used). The BETWEEN keyword allows for 4000 < x < 6000 values to be selected.

Total Number of Successful and Failure Mission Outcomes

Calculate the total number of successful and failure mission outcomes

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



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

The COUNT keyword is used to calculate the total number of mission outcomes, and the GROUPBY keyword is also used to group these results by the type of mission outcome.

Boosters Carried Maximum Payload

List the names of the booster which have carried the maximum payload mass.

```
%sql SELECT DISTINCT(BOOSTER_VERSION) FRO SPACEXTBL \
    WHERE PAYLOAD_MASS_KG_ = (SELECT MAX(PAYLOAD_MASS_KG_);
```

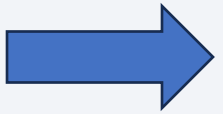
booster_version
F9 B5 B1048.4
F9 B5 B1048.5
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1049.7
F9 B5 B1051.3
F9 B5 B1051.4
F9 B5 B1051.6
F9 B5 B1056.4
F9 B5 B1058.3
F9 B5 B1060.2
F9 B5 B1060.3

A subquery is used here. The SELECT statement within the brackets finds the maximum payload, and this value is used in the WHERE condition. The DISTINCT keyword is then used to retrieve only distinct/unique booster versions.

2015 Launch Records

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

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



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

The WHERE keyword is used to filter the results for only failed landing outcomes, AND only for the of 2015.

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

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

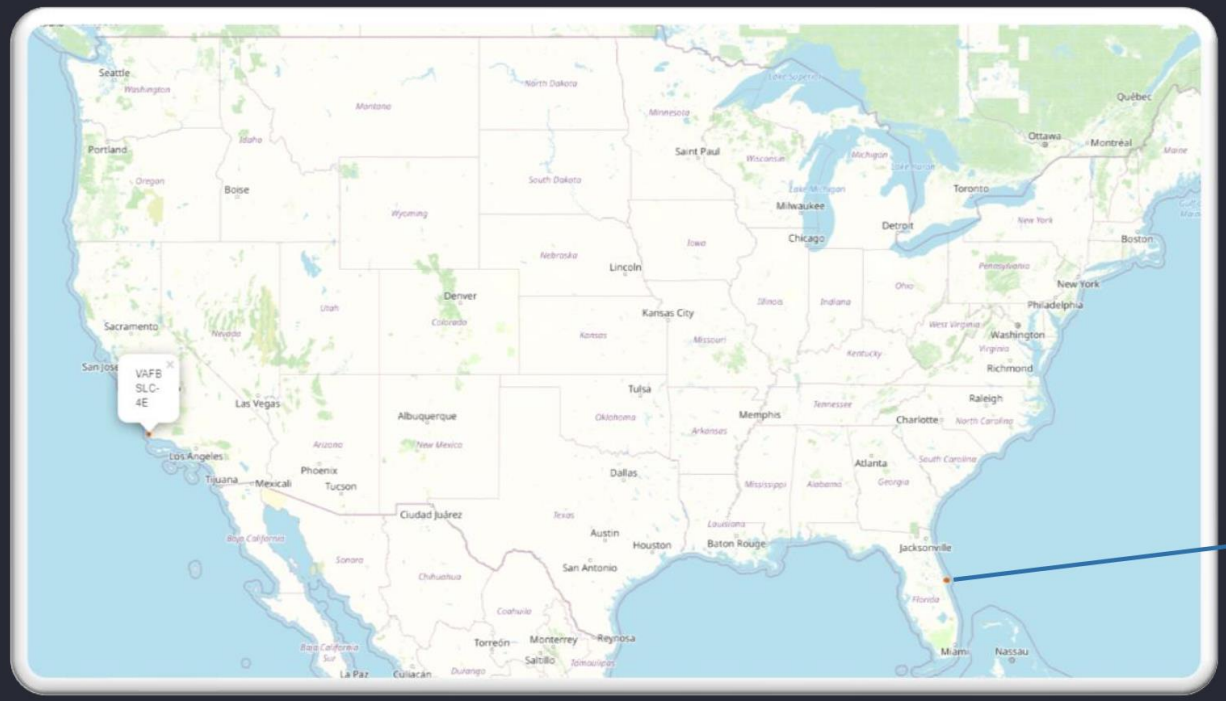
```
%sql SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) AS TOTAL_NUMBER FROM SPACEXTBL \
    WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
    GROUP BY LANDING__OUTCOME \
    ORDER BY TOTAL_NUMBER DESC;
```

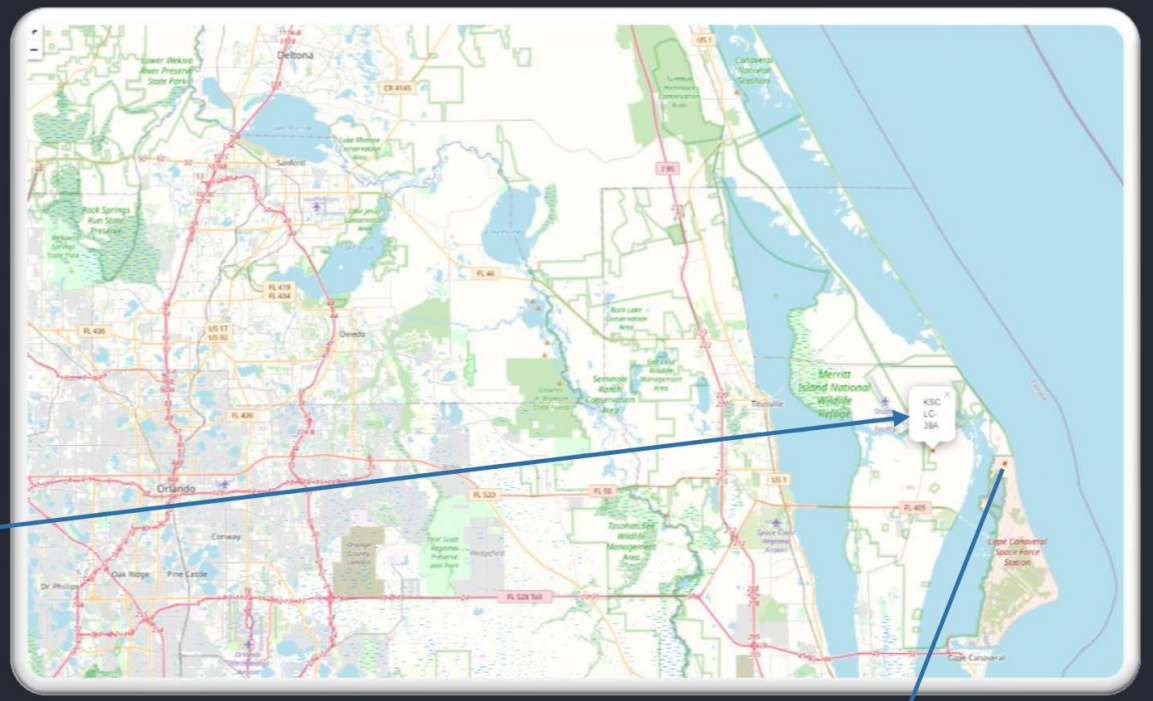
landing__outcome	total_number
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

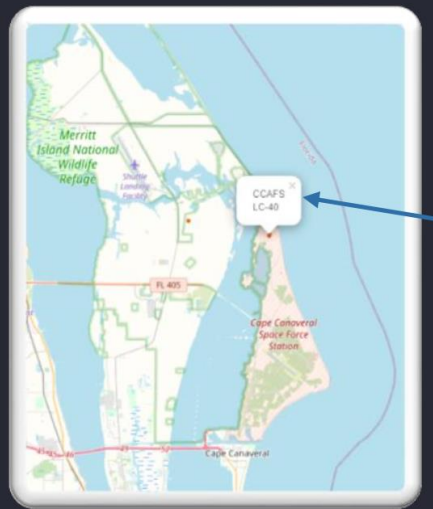
The WHERE keyword is used with the BETWEEN keyword to filter the results to dates only within those specified. The resuls are then grouped and ordered, using the keywords GROUP BY and ORDER BY, respectively, where DESC is used to specifiy the descending order.

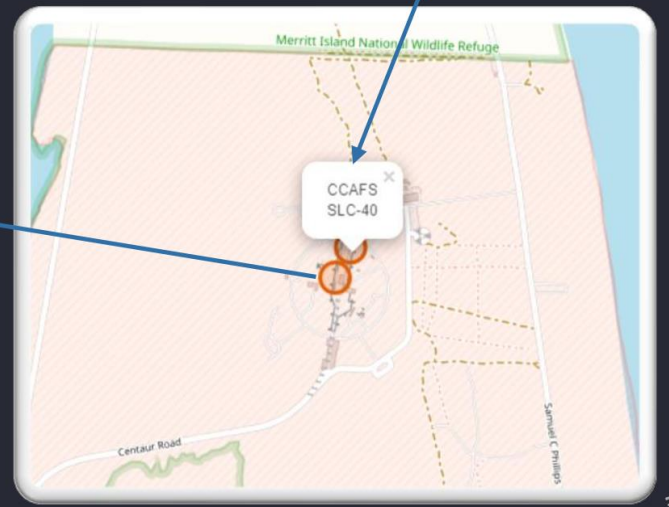

Section 3
Launch Sites
Proximities Analysis

All Launch Sites



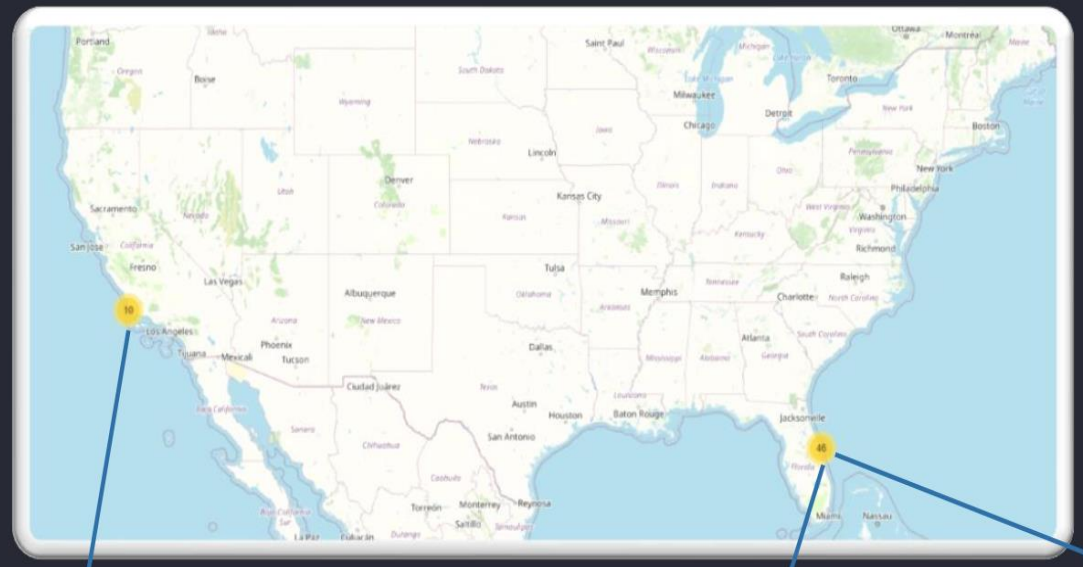






All SpaceX launch sites are on coasts of the United States of America, specifically Florida and California.

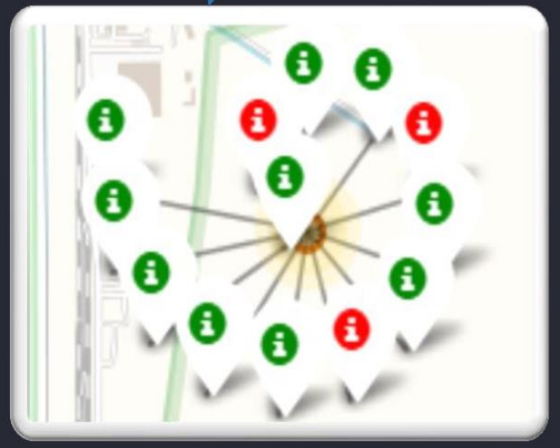
Success/Failed Launches for each site



Launches have been grouped into clusters, and annotated with green icons for successful launches, and red icons for failed launches.

VAFB SLC-4E

KSC LC-39A

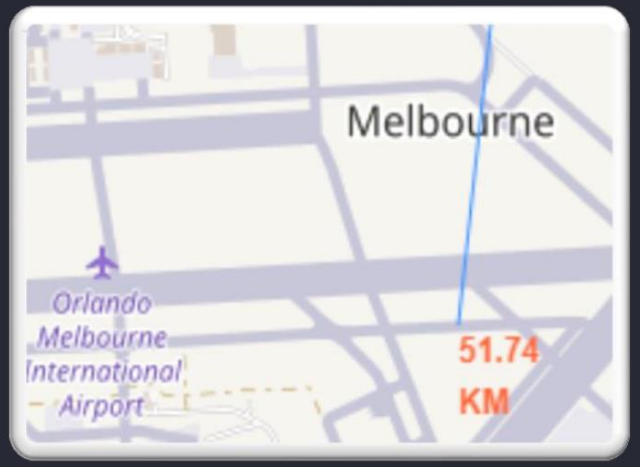


CCAFS SLC-40 and CCAFS LC-40

=

Launch Sites distance to landmarks its proximities

Using the CCAFS SLC-40 launch site as an example site, we can 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?

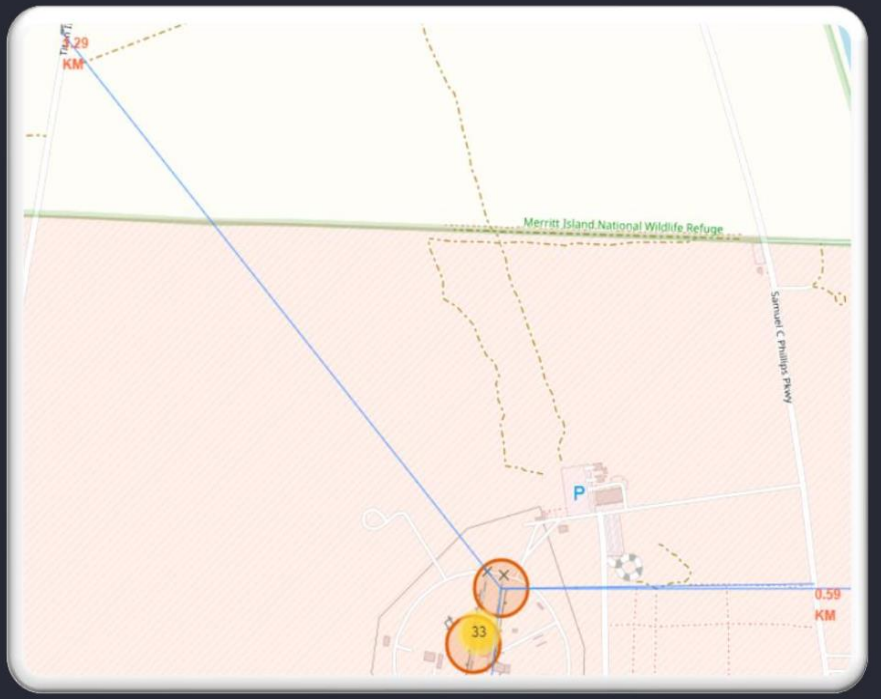
- YES. The nearest highway is only 0.59km away.

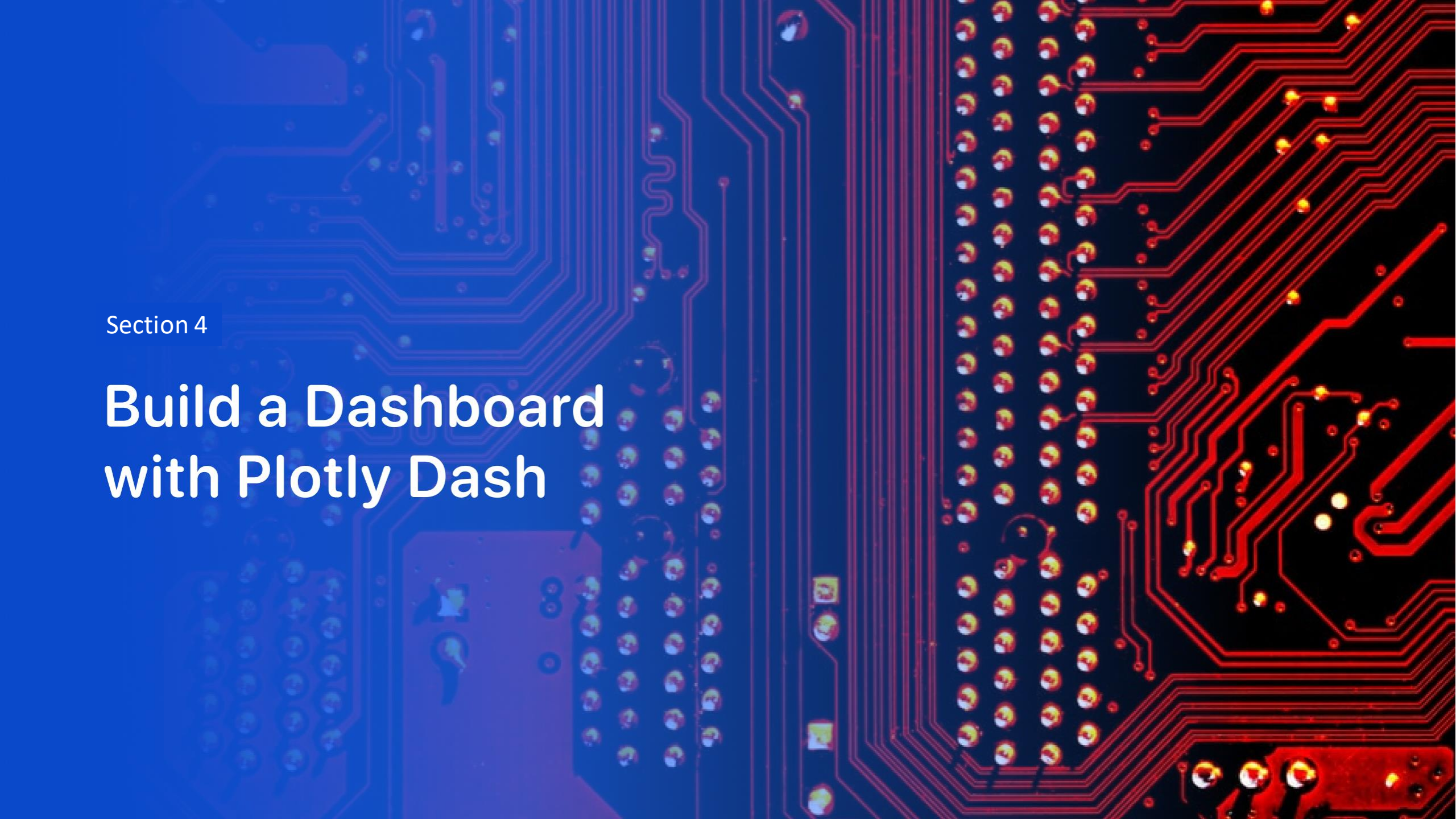
Are launch sites in close proximity to railways?

- YES. The nearest railway is only 1.29 km away.

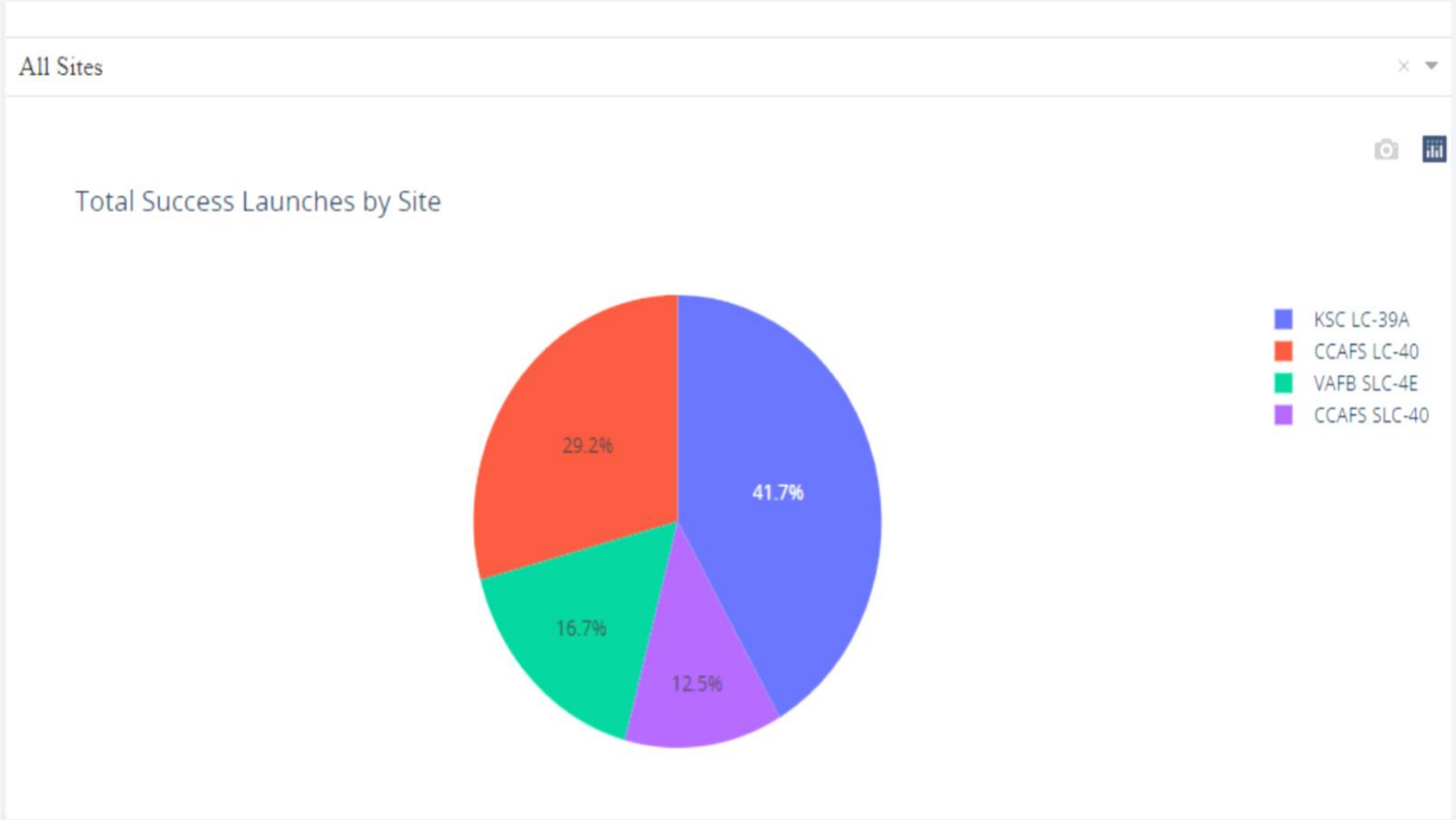
Do launch sites keep certain distance away from cities?

- YES. The nearest city is 51.74 km away.



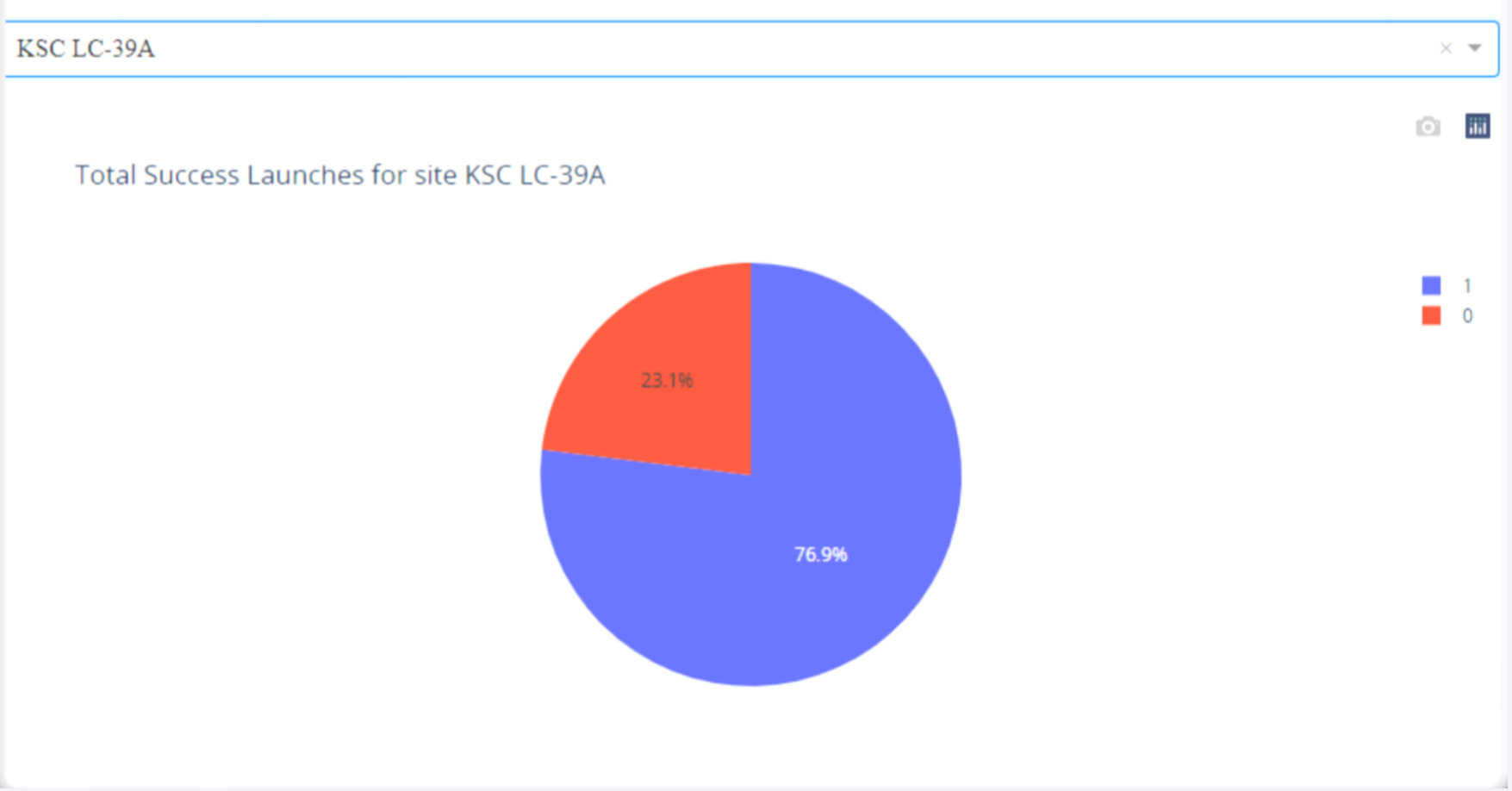

Section 4
Build a Dashboard with Plotly Dash

Total Success by each launch site



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

Launch Site Success Ratio



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

Launch Outcome vs. Payload Scatter Plot for All Sites

- Plotting the launch outcome vs. payload for all sites shows a gap around 4000 kg, so it makes sense to split the data into 2 ranges:
  - 0 – 4000 kg (low payloads)
  - 4000 – 10000 kg (massive payloads)
- From these 2 plots, it can be shown that the success for massive payloads is lower than that for low payloads.
- It is also worth noting that some booster types (v1.0 and B5) have not been launched with massive payloads.

1

2

Section 5

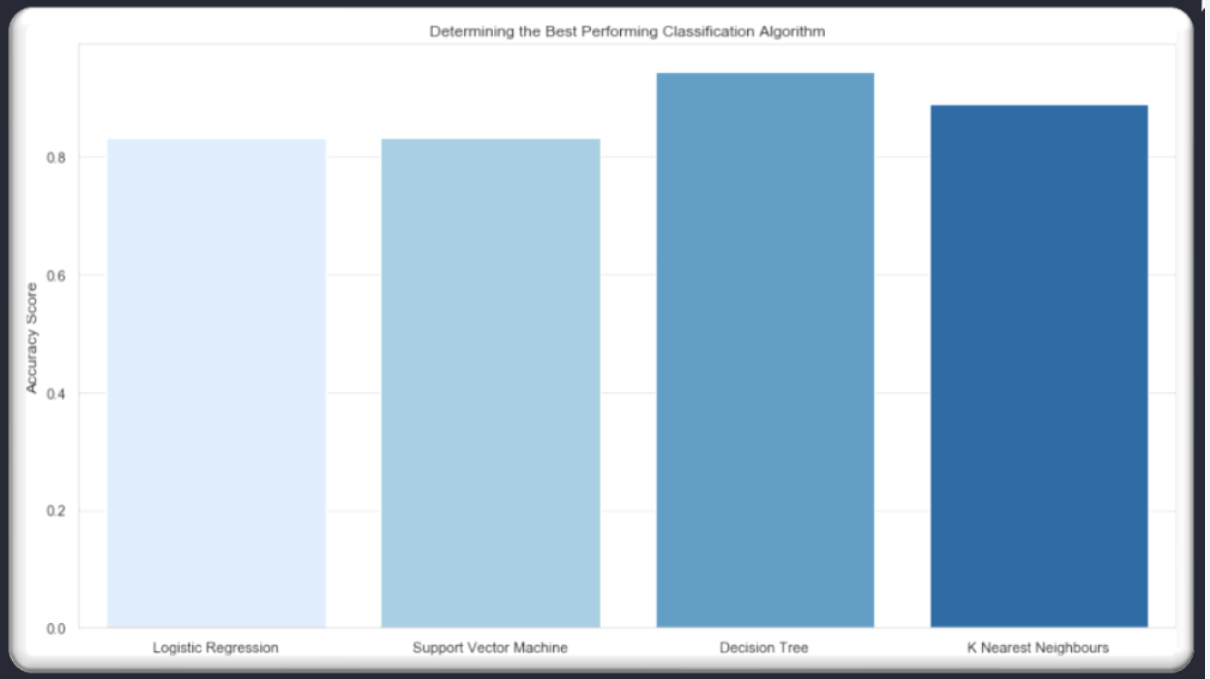
Predictive Analysis (Classification)

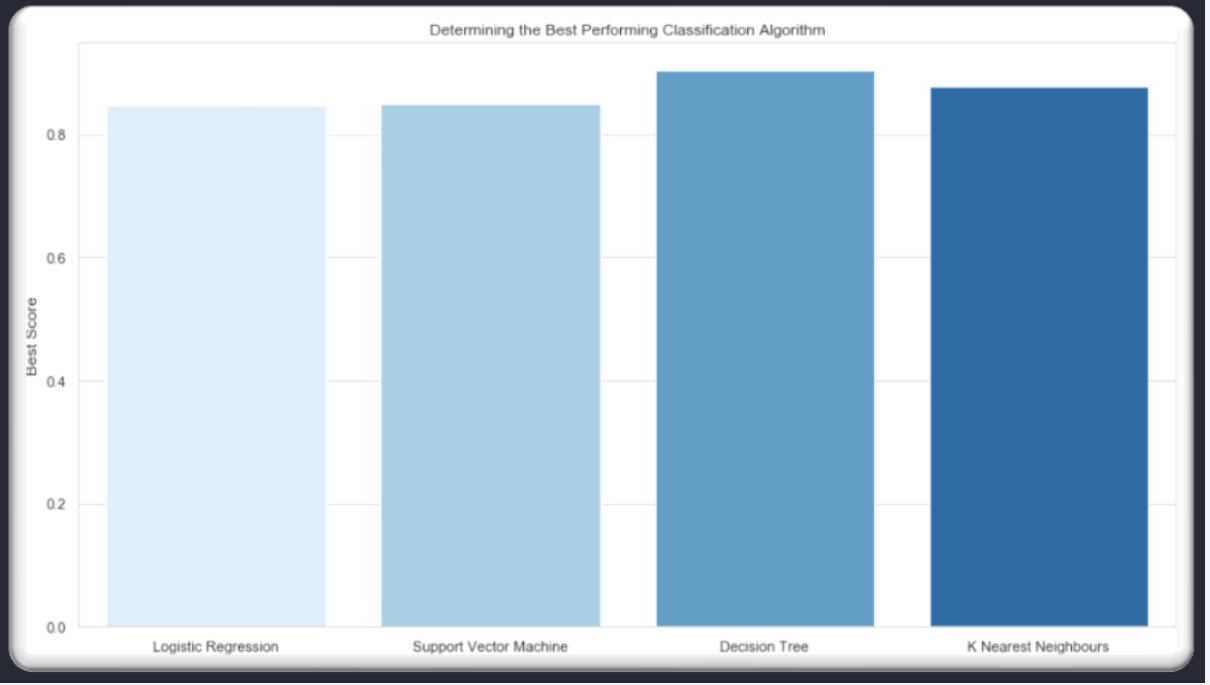
CLASSIFICATION ACCURACY

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

- The Decision Tree model has the highest classification accuracy
  - The Accuracy Score is 94.44%
  - The Best Score is 90.36%

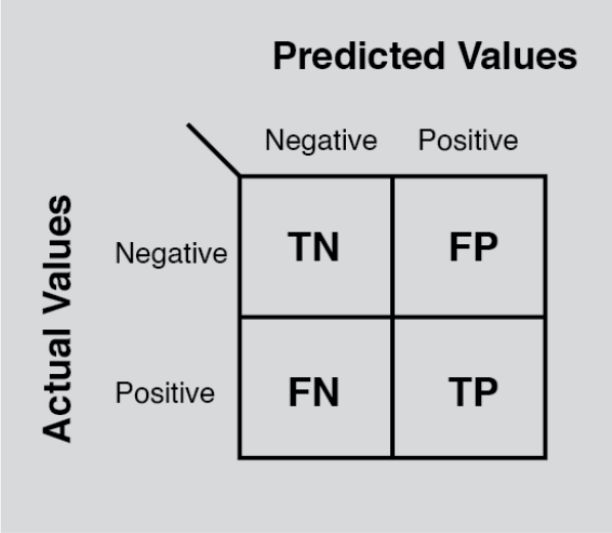
Algorithm	Accuracy Score	Best Score
Logistic Regression	0.833333	0.846429
Support Vector Machine	0.833333	0.848214
Decision Tree	0.944444	0.903571
K Nearest Neighbours	0.888889	0.876786

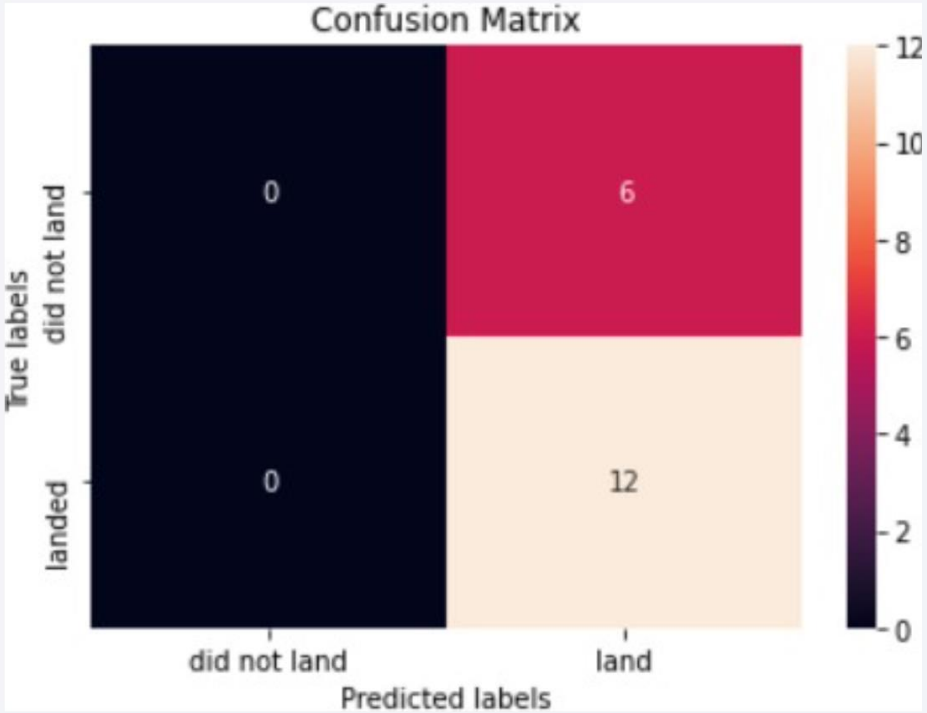




Confusion Matrix

Examining the confusion matrix, we see that Tree can distinguish between the different classes. We see that the major problem is false positives.



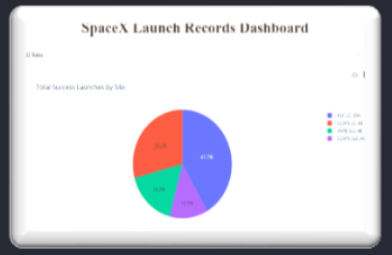


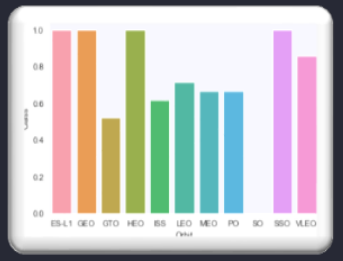
APPENDIX

Conclusions

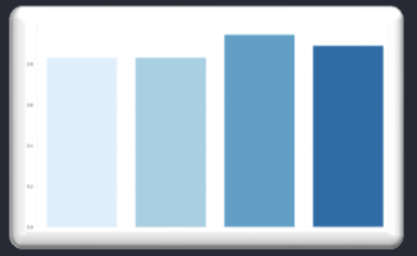
- As the number of flights increases, the rate of success at a launch site increases, with most early flights being unsuccessful. I.e. with more experience, the success rate increases.
  - Between 2010 and 2013, all landings were unsuccessful (as the success rate is 0).
  - After 2013, the success rate generally increased, despite small dips in 2018 and 2020.
  - After 2016, there was always a greater than 50% chance of success.

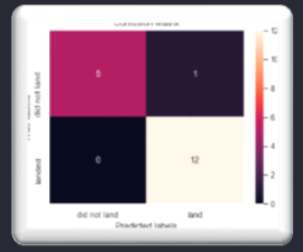
- Orbit types ES-L1, GEO, HEO, and SSO, have the highest (100%) success rate.
  - The 100% success rate of GEO, HEO, and ES-L1 orbits can be explained by only having 1 flight into the respective orbits.
  - The 100% success rate in SSO is more impressive, with 5 successful flights.
  - The orbit types 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 4000kg) is lower than that for low payloads.

- The best performing classification model is the Decision Tree model, with an accuracy of 94.44%.

DATA COLLECTION – WEB SCRAPING

- Custom functions for web scraping
- Custom logic to fill up the launch_dict values with values from the launch tables

```
def date_time(table_cells):
    """
    This function returns the data and time from the HTML  table cell
    Input: the  element of a table data cell extracts extra row
    """
    return [data_time.strip() for data_time in list(table_cells.strings)][0:2]

def booster_version(table_cells):
    """
    This function returns the booster version from the HTML  table cell
    Input: the  element of a table data cell extracts extra row
    """
    out=''.join([booster_version for i,booster_version in enumerate( table_cells.strings) if i%2==0][0:-1])
    return out

def landing_status(table_cells):
    """
    This function returns the landing status from the HTML table cell
    Input: the  element of a table data cell extracts extra row
    """
    out=[i for i in table_cells.strings][0]
    return out


def get_mass(table_cells):
    mass=unicodedata.normalize("NFKD", table_cells.text).strip()
    if mass:
        mass.find("kg")
        new_mass=mass[0:mass.find("kg")+2]
    else:
        new_mass=0
    return new_mass


def extract_column_from_header(row):
    """
    This function returns the landing status from the HTML table cell
    Input: the  element of a table data cell extracts extra row
    """
    if (row.br):
        row.br.extract()
    if row.a:
        row.a.extract()
    if row.sup:
        row.sup.extract()

    colunm_name = ' '.join(row.contents)

    # Filter the digit and empty names
    if not(colunm_name.strip().isdigit()):
        colunm_name = colunm_name.strip()
        return colunm_name
```

```
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
            # Append the flight_number into launch_dict with key `Flight No.`
            launch_dict["Flight No."].append(flight_number)

            # Date value
            #Append the date into launch_dict with key `Date`
            datatimelist=date_time(row[0])
            date = datatimelist[0].strip(',')
            launch_dict["Date"].append(date)

            # Time value
            #Append the time into launch_dict with key `Time`
            time = datatimelist[1]
            launch_dict["Time"].append(time)

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

            # Launch Site
            #Append the bv into launch_dict with key `Launch site`
            launch_site = row[2].a.string
            launch_dict['Launch site'].append(launch_site)

            # Payload
            #Append the payload into launch_dict with key `Payload`
            payload = row[3].a.string
            launch_dict['Payload'].append(payload)

            # Payload Mass
            #Append the payload_mass into launch_dict with key `Payload mass`
            payload_mass = get_mass(row[4])
            launch_dict['Payload mass'].append(payload_mass)

            # Orbit
            #Append the orbit into launch_dict with key `Orbit`
            orbit = row[5].a.string
            launch_dict['Orbit'].append(orbit)

            # Customer
            #Append the customer into launch_dict with key `Customer`
            if row[6].a != None:
                customer = row[6].a.string
            else:
                customer = 'None'

            launch_dict['Customer'].append(customer)

            # Launch outcome
            #Append the Launch_outcome into launch_dict with key `Launch outcome`
            launch_outcome = list(row[7].strings)[0]
            launch_dict['Launch outcome'].append(launch_outcome)

            # Booster Landing
            #Append the booster landing into launch_dict with key `Booster landing`
            booster_landing = landing_status(row[8])
            launch_dict['Booster landing'].append(booster_landing)


            print(f"Flight Number: {flight_number}, Date: {date}, Time: {time} \n \
            Booster Version {bv}, Launch Site: {launch_site} \n \
            Payload: {payload}, Orbit: {orbit} \n \
            Customer: {customer}, Launch Outcome: {launch_outcome}\
            Booster Landing: {booster_landing} \n \
            *** ")
```

DATA COLLECTION – SPACE X REST API

- Custom functions to retrieve the required information
- Custom logic to clean the data

```python
# 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 payloads in a single rocket.
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)]
```

Python

From the `rocket` column we would like to learn the booster name.

```python
# Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

Python

From the `launchpad` we would like to know the name of the launch site being used, the longitude, and the latitude.

```python
# Takes the dataset and uses the launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['launchpad']:
        response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
        Longitude.append(response['longitude'])
        Latitude.append(response['latitude'])
        LaunchSite.append(response['name'])
```

Python

From the `payload` we would like to learn the mass of the payload and the orbit that it is going to.

```python
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

Python

From `cores` we would like to learn the outcome of the landing, the type of the landing, number of flights with that core, whether gridfins were used, whether the core is reused, whether legs were used, the landing pad used, the block of the core (which is a number used to separate versions of cores), the number of times this specific core has been reused, and the serial of the core.

```python
# Takes the dataset and uses the cores column to call the API and append the data to the lists
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
                Block.append(response['block'])
                ReusedCount.append(response['reuse_count'])
                Serial.append(response['serial'])
            else:
                Block.append(None)
                ReusedCount.append(None)
                Serial.append(None)
            Outcome.append(str(core['landing_success'])+' '+str(core['landing_type']))
            Flights.append(core['flight'])
            GridFins.append(core['gridfins'])
            Reused.append(core['reused'])
            Legs.append(core['legs'])
            LandingPad.append(core['landpad'])
```

Python

Thank you!