Winning Space Race with Data Science

Cintia M.
26-Jun-2025

Outline

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

Executive Summary

- Summary of methodologies
  - ✓ Data Collection
  - ✓ Data Wrangling
  - ✓ Exploratory Data Analysis (EDA)
  - ✓ Interactive Visual Analytics
  - ✓ Predictive Analysis (Classification).

- Summary of all results:
  - ✓ EDA results
  - ✓ Interactive analytics
  - ✓ Predictive analysis

Introduction

SpaceX advertises Falcon 9 rocket launches on its website with a cost of $62 million. This is considerably cheaper than other providers (which usually cost upwards of $165 million). Much of the savings are because SpaceX can land, and then re-use the first stage of the rocket.

If we can make predictions on whether the first stage will land, we can determine the cost of a launch, and use this information to assess whether or not an alternate company should bid against SpaceX for a rocket launch.

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

Section 1

Methodology

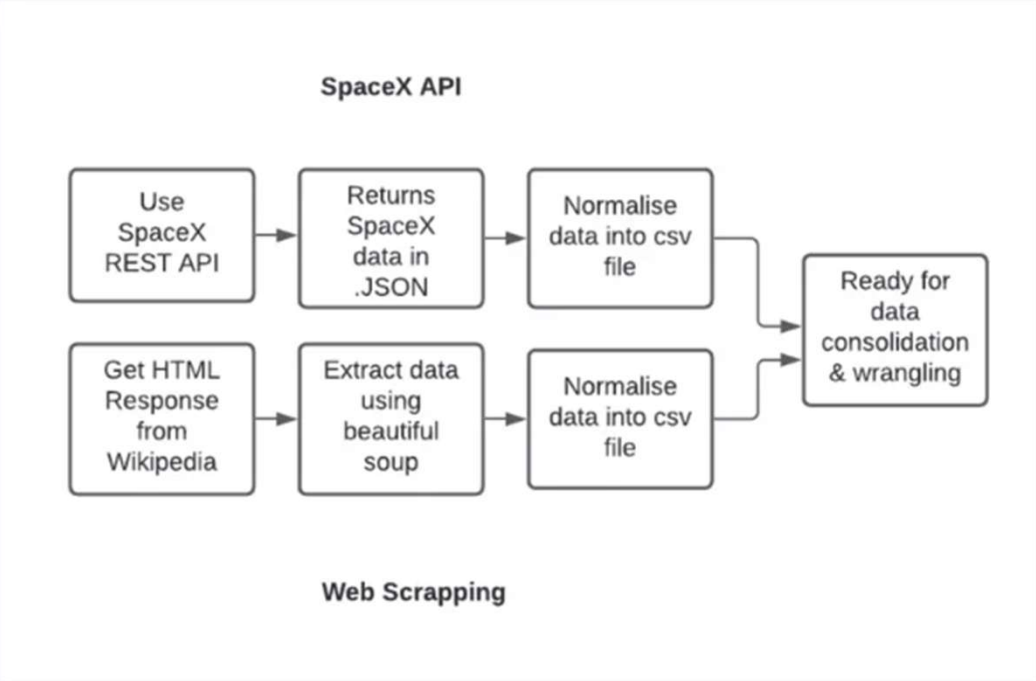
Methodology

1. Data collection
   - Data was collected using SpaceX API and Web Scraping from Wikipedia.
2. Data Wrangling
   - One Hot Encoding and Data Cleaning (of null values and irrelevant features) were applied.
3. Exploratory data analysis
   - EDA was applied using SQL queries and Visualization (Pandas and Matplotlib).
4. Interactive Visual Analytics
   - Geospatial analytics using Folium and Interactive Dashboard creating with Plotly Dash.
5. Predictive analysis
   - LR, KNN, SVM, DT models were built and evaluated for the best classifier.

Data Collection

- **SpaceX REST API** was used to retrieve structured launch data, including information about the rocket used, payload delivered, launchpads, landing specifications, and landing outcome.

- **Web Scraping Wikipedia** was also used to extract Falcon 9 & Falcon Heavy launch records (BeautifulSoup was applied to parse HTML tables).



Data Collection - SpaceX REST API calls

1. Getting Response from API:

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

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

2. Request and decode the response content as a .json file:

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

```
json_data = response.json()
data = json_normalize(json_data)
```

3. Apply custom functions to clean data:

```
getBoosterVersion(data)
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

4. Assign list to dictionary and then to a dataframe (df) :

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
...
'Longitude': Longitude,
'Latitude': Latitude}
```

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

5. Filter df to only include `Falcon 9` launches:

```
data_falcon9 = data[data['BoosterVersion'] == 'Falcon 9'].reset_index(drop=True)
```

GitHub URL with the completed SpaceX API calls:
https://github.com/Cintig/AppliedDataScienceCapstoneCourse/blob/2e7e025abf759b131010fce85b949ebc85f46af4/Lab%201%20-%20Collecting%20the%20data%20-%20Completed%20exercise.ipynb

Data Collection – Web Scraping from Wikipedia

1. Getting Response from HTML and Create a BeautifulSoup object:

```
response = requests.get(static_url)
soup = BeautifulSoup(response.text, 'html.parser')
```

2. Extract column/variable names:

```
html_tables = soup.find_all('table')
first_launch_table = html_tables[2]
th_elements = first_launch_table.find_all('th')

column_names = []
for th in th_elements:
    name = extract_column_from_header(th)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

3. Creation of dictionary by parsing the launch HTML tables:

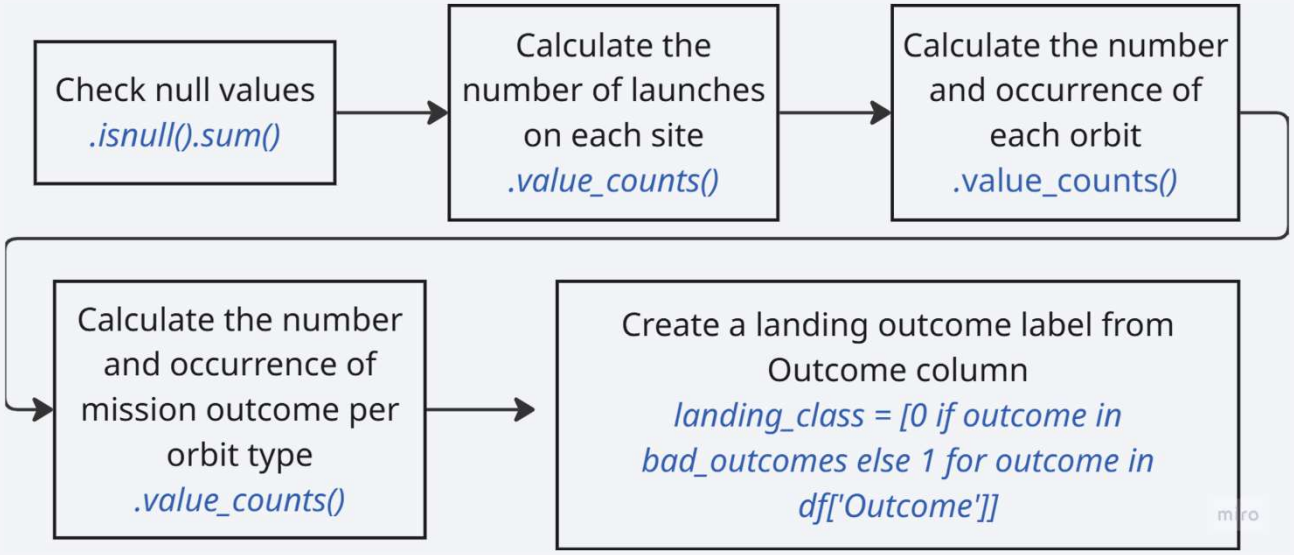
```
launch_dict= dict.fromkeys(column_names)
# Remove an irrelvant column
del launch_dict['Date and time ( )']
# launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
...
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
...
# Append the data into launch_dict
launch_dict['Flight No.'].append(flight_number)
...
```

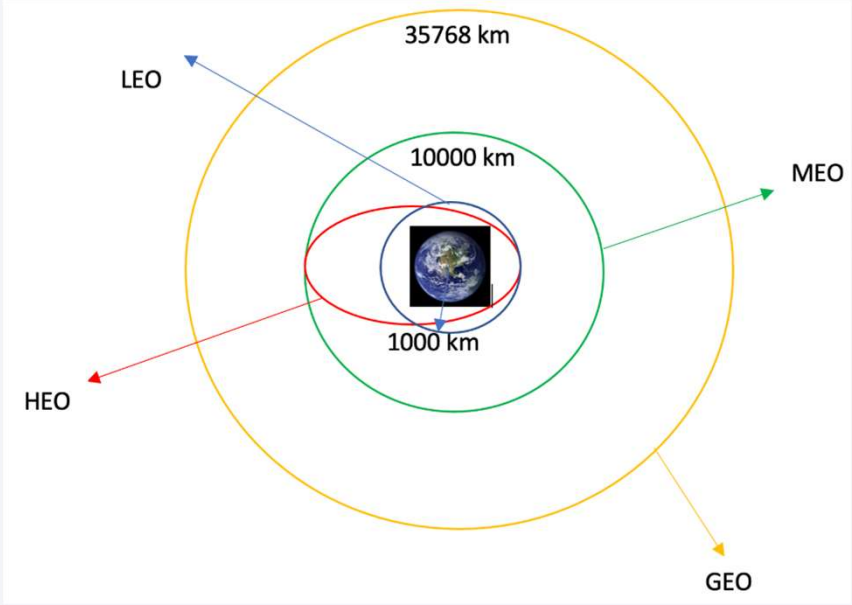
4. Converting the dictionary to dataframe:

```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

GitHub URL with the completed Web Scraping calls:
https://github.com/Cintig/AppliedDataScienceCapstoneCourse/blob/2e7e025abf759b131010fce85b949ebc85f46af4/Lab%201.2%20-%20Web%20Scraping.ipynb

Data Wrangling





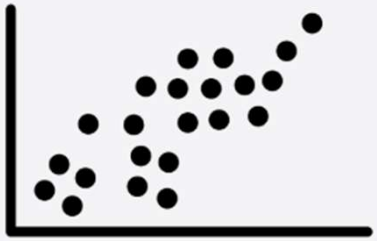
GitHub URL:
https://github.com/Cintig/AppliedDataScienceCapstoneCourse/blob/2e7e025abf759b131010fce85b949ebc85f46af4/Lab%202%20-%20Data%20Wrangling.ipynb

EDA with Data Visualization

SCATTER CHARTS

Scatter charts were produced to visualize the relationships between:

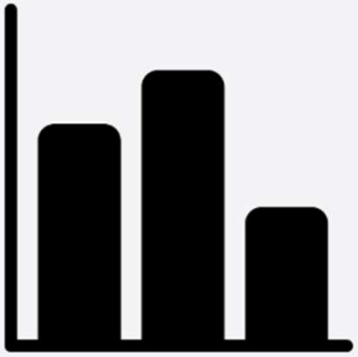
- Flight Number and Launch site
- Payload and Launch Site
- Orbit Type an Flight Number
- Payload and Orbit Type



BAR CHARTS

A bar chart was produced to visualize the relationship between:

- Success Rate and Orbit Type



LINE CHARTS

Line charts were produced to visualize the relationships between:

- Success Rate and Year (launch success yearly trend)

- GitHub URL: https://github.com/Cintig/AppliedDataScienceCapstoneCourse/blob/2e7e025abf759b131010fce85b949ebc85f46af4/Lab%204%20-%20EDA%20with%20Visualization%20Lab%20-%20Exploring%20and%20Preparing%20Data.ipynb

EDA with SQL

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

- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display the average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome on a ground pad was achieved
- List the names of the boosters which had success on a drone ship and a payload mass between 4000 and 6000 kg
- List the total number of successful and failed mission outcomes
- List the names of the booster versions which have carried the maximum payload mass
- List the failed landing outcomes on drone ships, their booster versions, and launch site names for 2015
- Rank the count of landing outcomes, such as Failure (drone ship) or Success (ground pad), in descending order

GitHub URL:
https://github.com/Cintig/AppliedDataScienceCapstoneCourse/blob/595bf5f093df773fd4d7e9f187564aada401630a/Lab%203%20-%20Complete%20the%20EDA%20with%20SQL.ipynb

Interactive Map built with Folium

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

1. **Mark all launch sites on a map**
   - Initialise 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 colour 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_colour* 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.

GitHub URL:
https://github.com/Cintig/AppliedDataScienceCapstoneCourse/blob/2e7e025abf759b131010fce85b949ebc85f46af4/Lab%205%20-%20Interactive%20Visual%20Analytics%20with%20Folium%20lab.ipynb

Dashboard built with 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

GitHub URL:
https://github.com/Cintig/AppliedDataScienceCapstoneCourse/blob/2e7e025abf759b131010fce85b949ebc85f46af4/Lab%206%20-%20Build%20an%20Interactive%20Dashboard%20with%20Plotly%20Dash/spacex-dash-app.py

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_)
- Plot and examine the Confusion Matrix

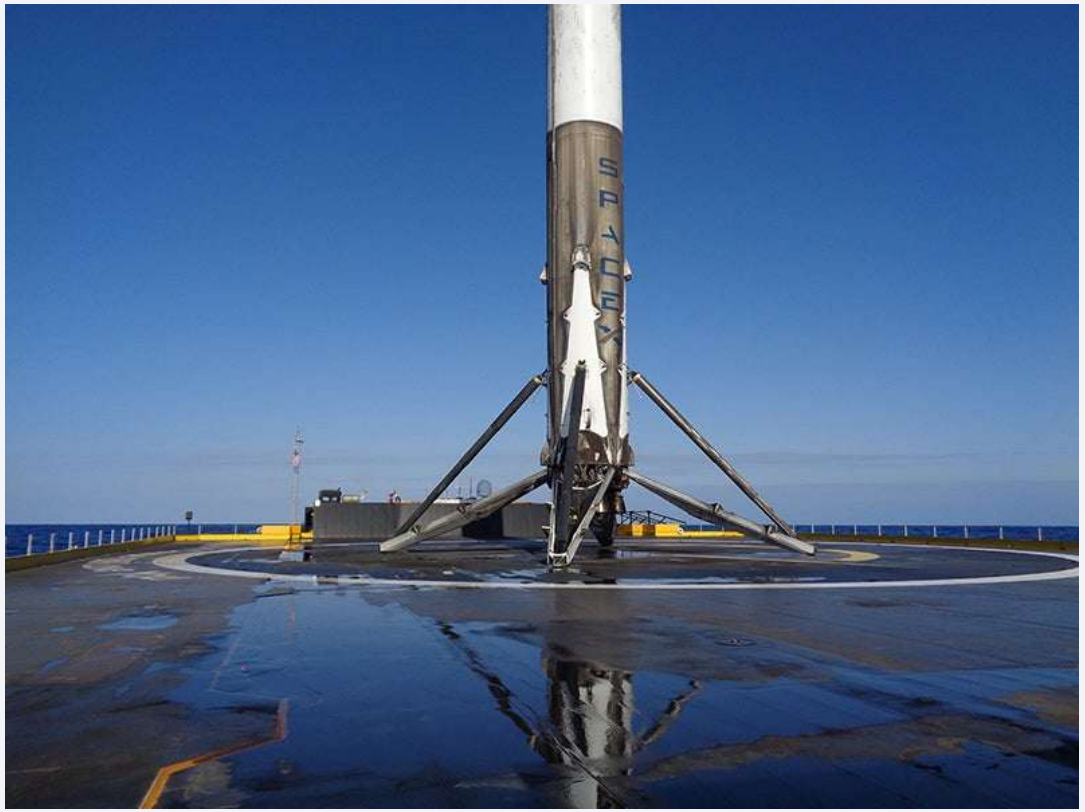
Finding the Best Classification Model

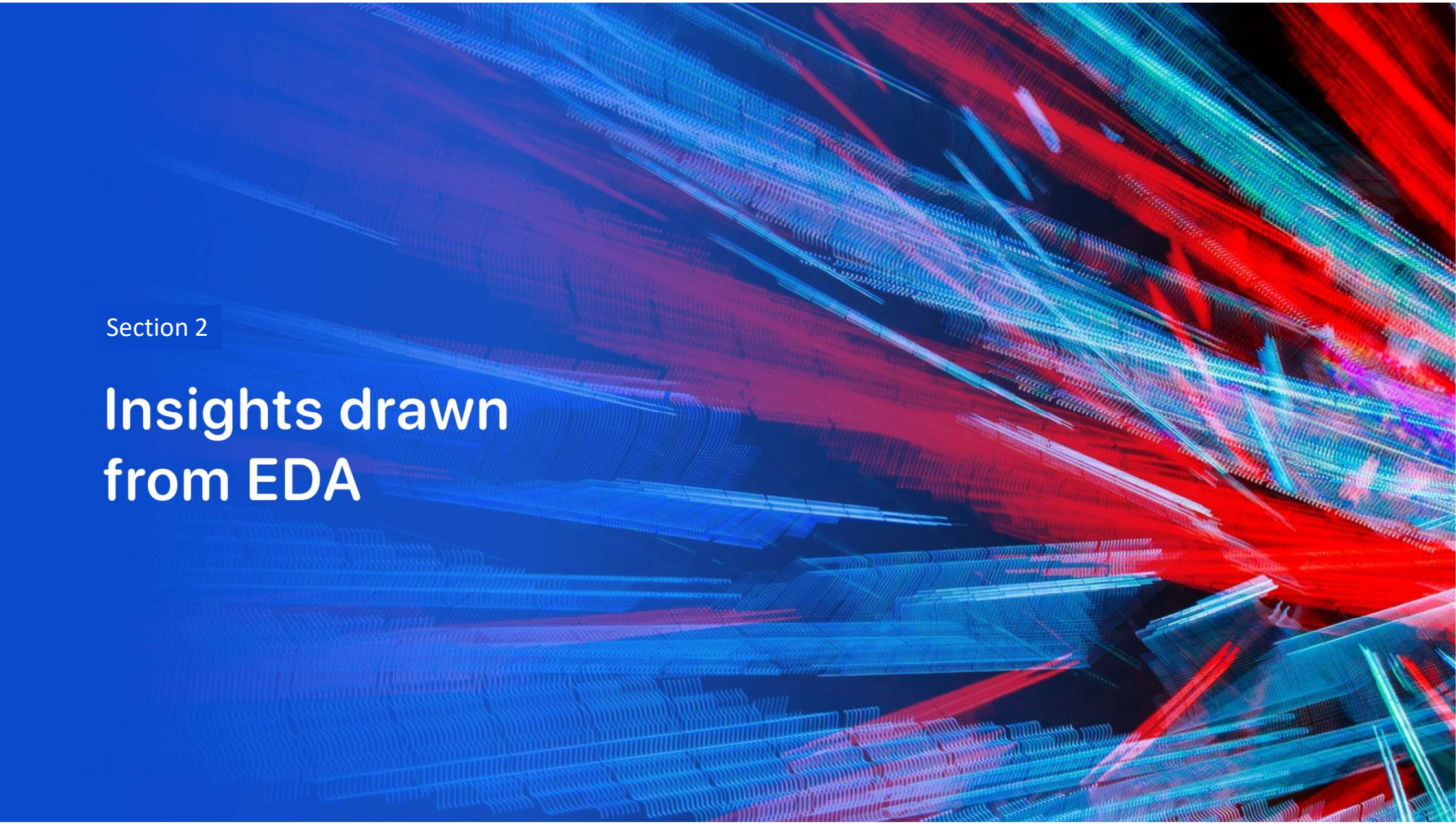
- Review the accuracy scores for all chosen algorithms
- The model with the highest accuracy score is determined as the best performing model

GitHub URL: https://github.com/Cintig/AppliedDataScienceCapstoneCourse/blob/7b2c3cfb848d0925196e2a0f1ec22b1599d15ad5/Lab%207%20-%20Complete%20the%20Machine%20Learning%20Prediction%20lab.ipynb

Results

- Exploratory Data Analysis (EDA)
- Interactive Analytics
- Predictive Analysis

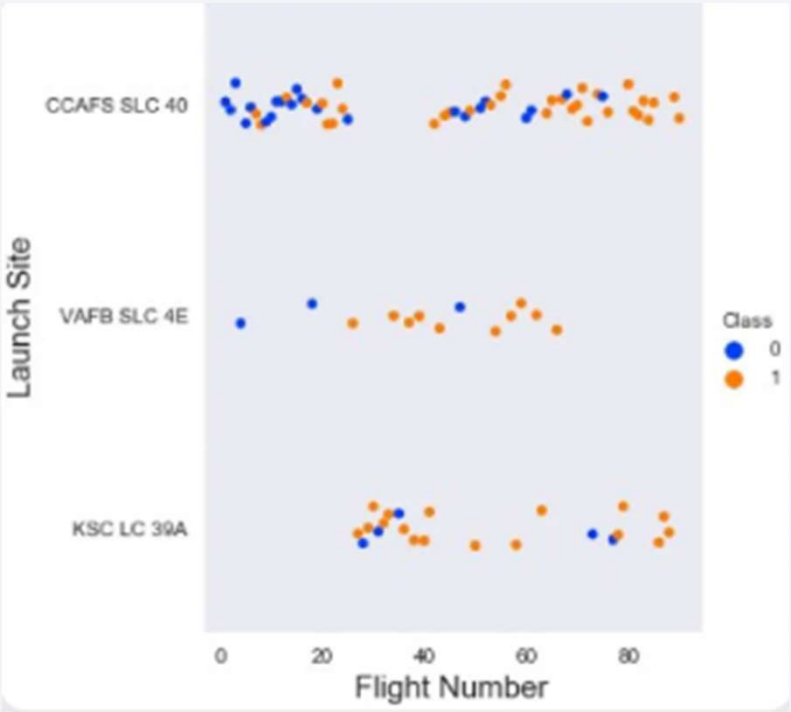



Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

The scatter plot of Launch vs Flight Number shows the following:

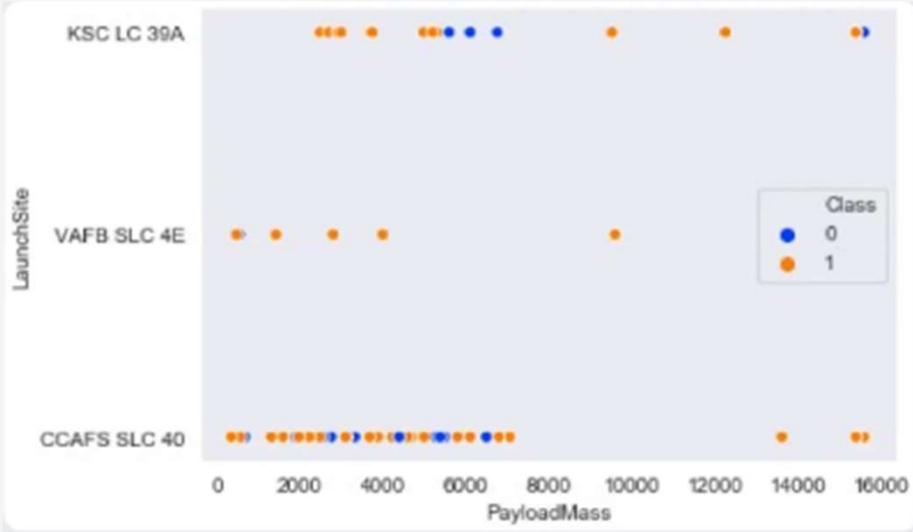
- ✓ As the number of flights increases, the success rate seems to increase for each launch site.
- ✓ Mos of the early flights (Flight Number < 30) were launched from CCAFS SLC 40, and were generally unsuccessful.
- ✓ Launches from VAFB SLC 4E followed a similar pattern.
- ✓ No early flights were launched from KSC LC 39A, so the launches from this site are more successful.
- ✓ From flight number ~30 and onward, successful landings (Class = 1) become more frequent.



Payload vs. Launch Site

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

- ✓ 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).



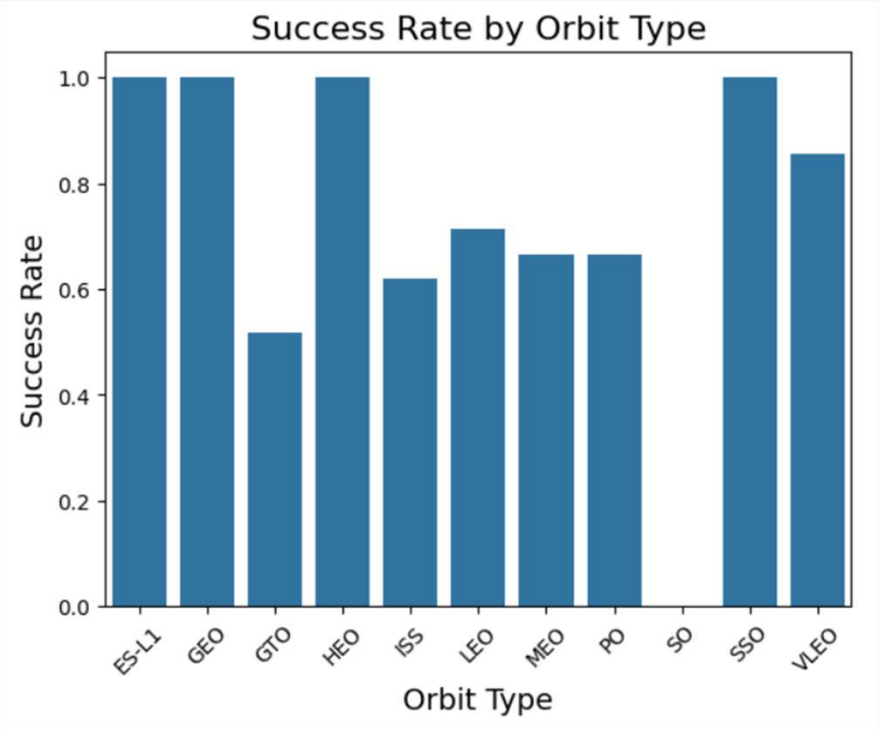
Success Rate vs. Orbit Type

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:

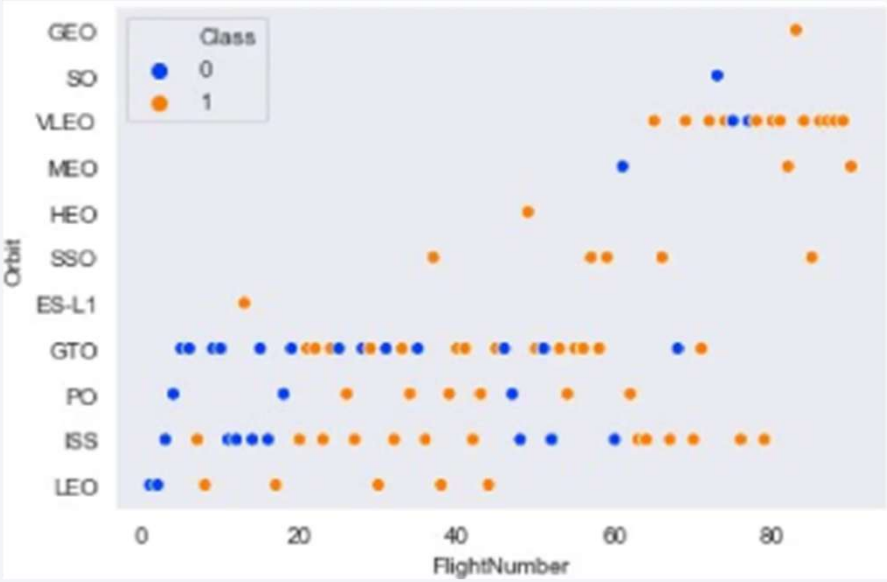
- ✓ SO (Heliocentric Orbit)



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:

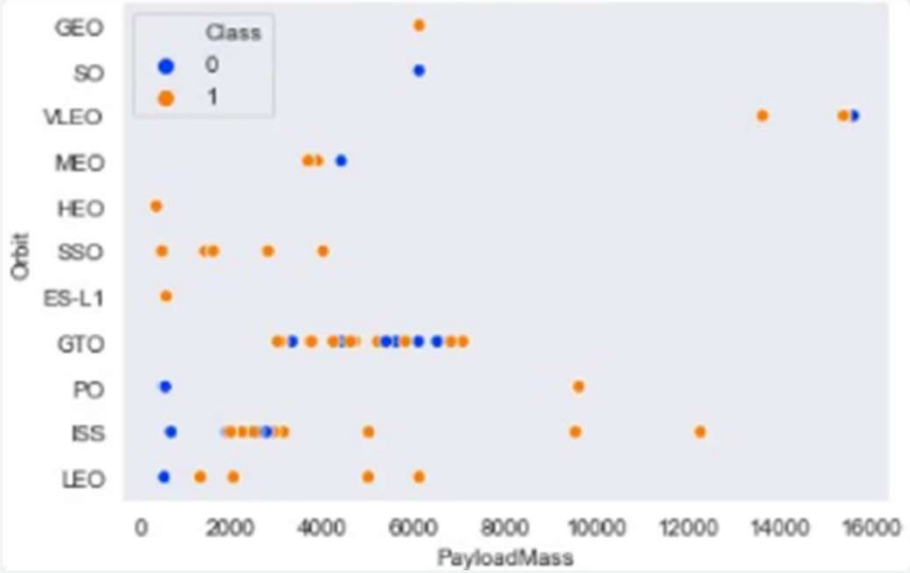
- ✓ The 100% success rate of GEO, HEO, and ES-L1 orbits can be explained by only having 1 flight into the respective orbits.
- ✓ The 100% success rate in SSO is more impressive, with 5 successful flights.
- ✓ There is little relationship between Flight Number and Success Rate for GTO.
- ✓ Generally, 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).



Payload vs. Orbit Type

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

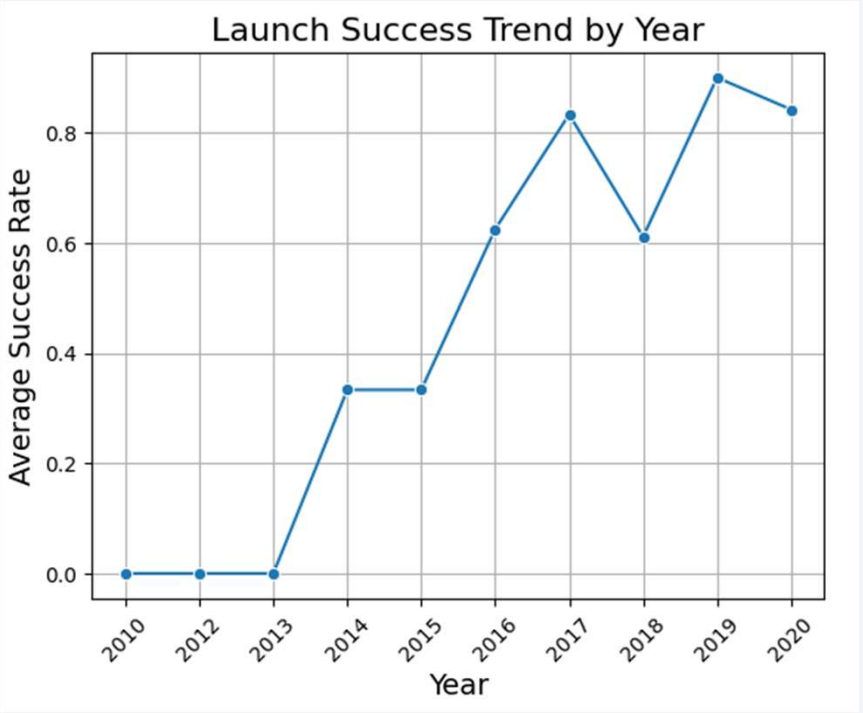
- ✓ The following orbit types have more success with heavy payloads:
  - ISS
  - LEO
  - PO (although the number of data points is small)
- ✓ For GTO, the relationship between payload mass and success rate is unclear.
- ✓ VLEO (Very Low Earth Orbit) launches are associated with heavier payloads.



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.



All Launch Site Names

- The keyword DISTINCT was used to find the names of the unique launch sites.

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

[11]:
```
%%sql
SELECT DISTINCT "Launch_Site"
FROM SPACEXTABLE;
```

```
 * sqlite:///my_data1.db
Done.
```

[11]:

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

Launch Site Names Begin with 'CCA'

- The query below was used to display 5 records where launch sites begin with ‘CCA’:

```
%%sql
SELECT *
FROM SPACEXTABLE
WHERE "Launch_Site" LIKE 'CCA%'
LIMIT 5;
```

```
 * sqlite:///my_data1.db
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	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	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

- The total payload carried by boosters from NASA was calculated.

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

```
%%sql
SELECT SUM("Payload_Mass__kg_") AS Total_Payload_Mass
FROM SPACEXTABLE
WHERE "Customer" = 'NASA (CRS)';
```

```
 * sqlite:///my_data1.db
Done.
```

Total_Payload_Mass
45596

Average Payload Mass by F9 v1.1

- The average payload mass carried by booster version F9 v1.1 was calculated.

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

```
%%sql
SELECT AVG("Payload_Mass__kg_") AS Average_Payload_Mass
FROM SPACEXTABLE
WHERE "Booster_Version" = 'F9 v1.1';
```

```
 * sqlite:///my_data1.db
Done.
```

Average_Payload_Mass
2928.4

First Successful Ground Landing Date

- The date of the first successful landing outcome on ground pad was 22-Dec-2015

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

```
%%sql
SELECT MIN("Date") AS First_Successful_Ground_Pad_Landing
FROM SPACEXTABLE
WHERE "Landing_Outcome" = 'Success (ground pad)';
```

```
 * sqlite:///my_data1.db
Done.
```

First_Successful_Ground_Pad_Landing
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- The **WHERE clause** was used to filter for boosters which have successfully landed on drone ship. Additionally, the **AND condition** was applied to determine successful landing with payload mass greater than 4000 but less than 6000

```
%%sql
SELECT DISTINCT "Booster_Version"
FROM SPACEXTABLE
WHERE "Landing_Outcome" = 'Success (drone ship)'
  AND "Payload_Mass__kg_" > 4000
  AND "Payload_Mass__kg_" < 6000;
```

```
 * sqlite:///my_data1.db
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

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

List the total number of successful and failure mission outcomes

```
%%sql
SELECT "Mission_Outcome", COUNT(*) AS Outcome_Count
FROM SPACEXTABLE
GROUP BY "Mission_Outcome";
```

```
 * sqlite:///my_data1.db
Done.
```

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

Boosters Carried Maximum Payload

- A **subquery in the WHERE clause** and the **MAX() function** were used to determine the booster versions which have carried the maximum payload.

```
%%sql
SELECT DISTINCT "Booster_Version", "Payload_Mass__kg_"
FROM SPACEXTABLE
WHERE "Payload_Mass__kg_" = (
    SELECT MAX("Payload_Mass__kg_")
    FROM SPACEXTABLE
);
```

```
 * sqlite:///my_data1.db
Done.
```

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

2015 Launch Records

- A combination of the WHERE clause, LIKE, and AND conditions were used to get the failed landing outcomes in drone ship, their booster versions, and the launch sites names in year 2015.

```
%%sql
SELECT
    SUBSTR("Date", 6, 2) AS Month,
    "Booster_Version",
    "Launch_Site",
    "Landing_Outcome"
FROM SPACEXTABLE
WHERE "Landing_Outcome" LIKE 'Failure (drone ship)%'
  AND SUBSTR("Date", 1, 4) = '2015';
```

```
 * sqlite:///my_data1.db
Done.
```

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

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

- A combination of the COUNT function, WHERE clause, BETWEEN, and AND conditions were used to get the Landing outcomes between the 4-Jun-2010 and 20-Mar-2017.
- Additionally, the GROUP BY and ORDER BY clauses were used to group de landing outcomes and ordered them in descending order.

```
%%sql
SELECT "Landing_Outcome", COUNT(*) AS Outcome_Count
FROM SPACEXTABLE
WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY "Landing_Outcome"
ORDER BY Outcome_Count DESC;
```

```
 * sqlite:///my_data1.db
Done.
```

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

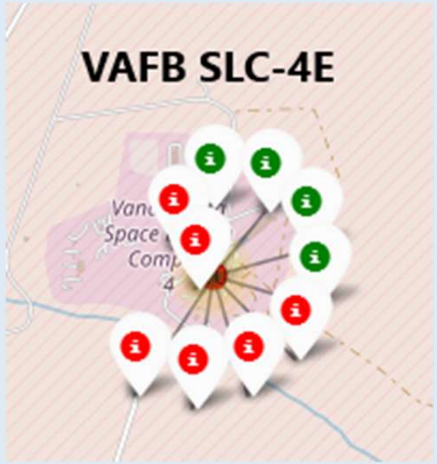

Section 3
Launch Sites
Proximities Analysis

All launch sites global map markers

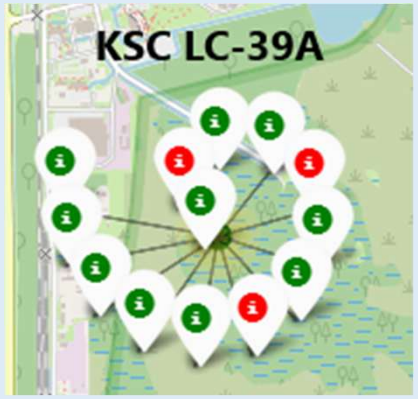
All SpaceX launch sites are on coasts on the United States of America, specifically Florida and California.

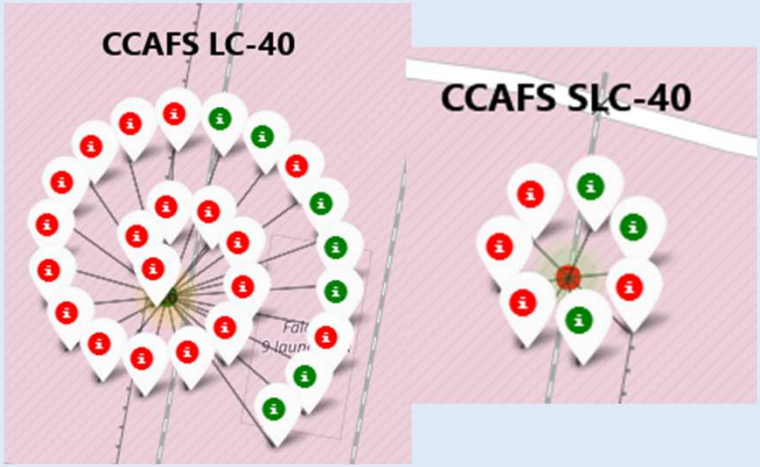
Markers showing launch sites with color labels

California Launch Site



Florida Launch Sites





Green Markers show successful launches and Red Markers show failures.

Proximity of Launch sites to other points of interest

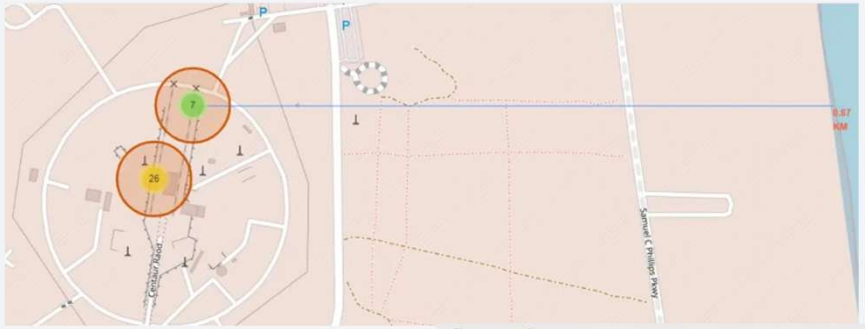
Using the CCAFS SLC-40 launch site as an example, we can understand more about the placement of launch sites.

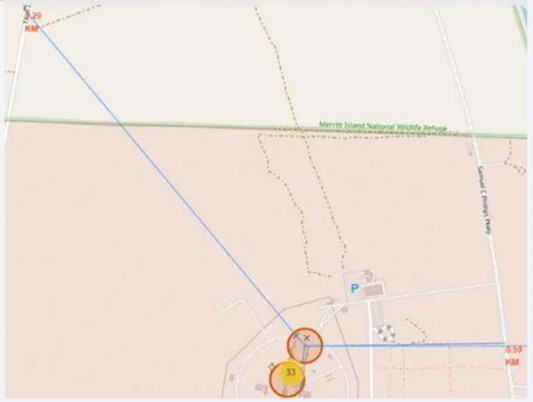
Q1: Are launch sites in close proximity to coastline?
YES. The coastline is only 0.87 km due East.

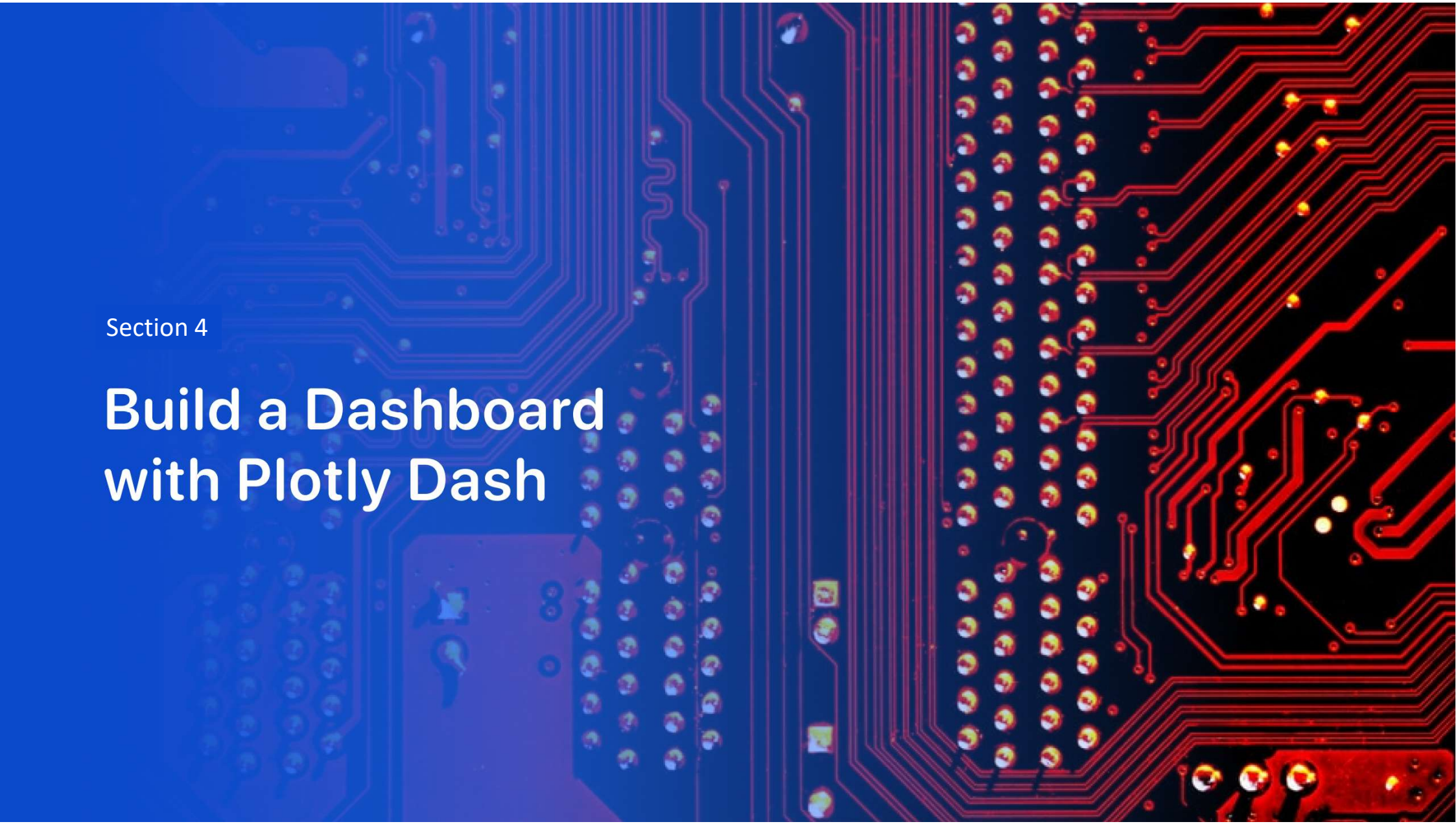
Q2: Are launch sites in close proximity to highways?
YES. The nearest highway is only 0.59 km away.

Q3: Are launch sites in close proximity to railways?
YES. The nearest railway is only 1.29 km away.

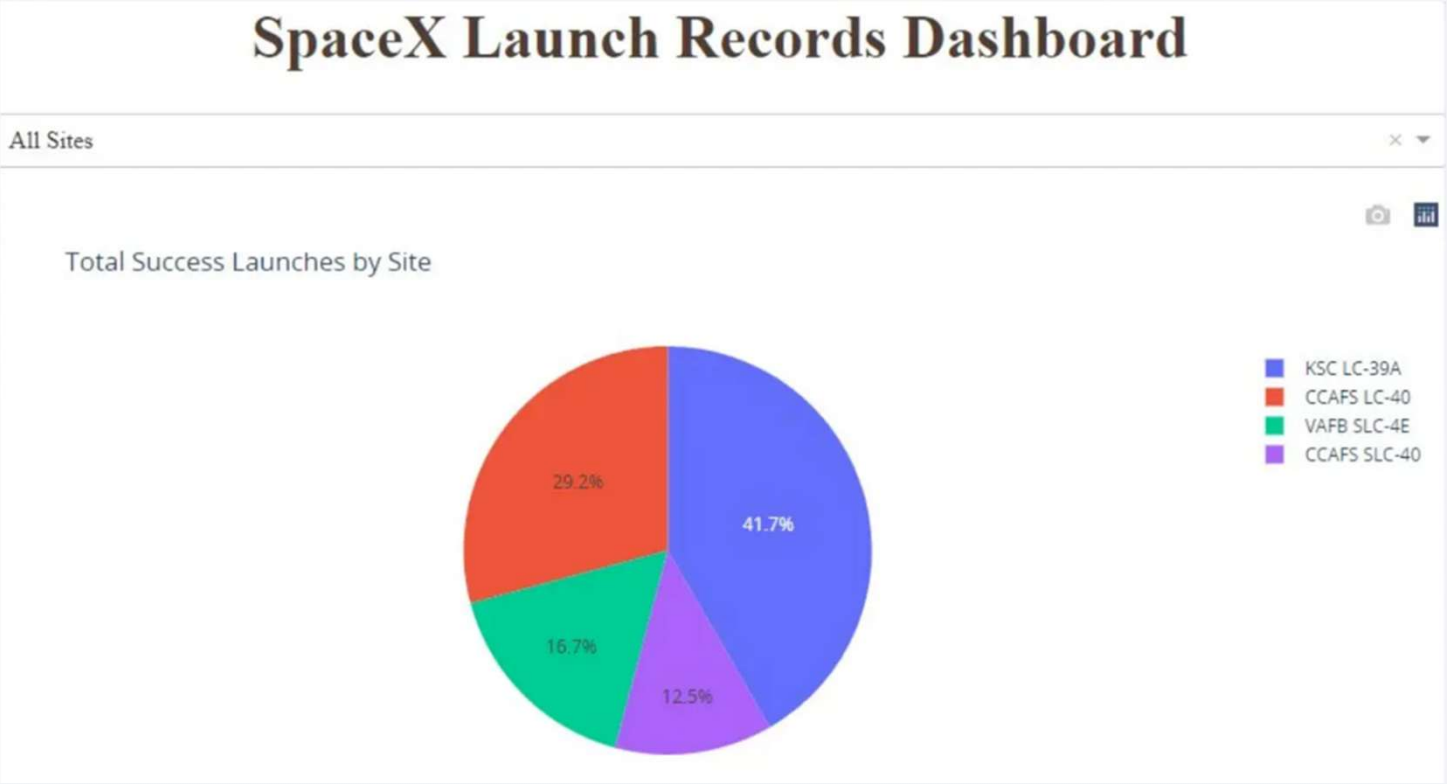
Q4: Do launch sites keep certain distance away from cities?
YES. The nearest city is 51.74 km away.





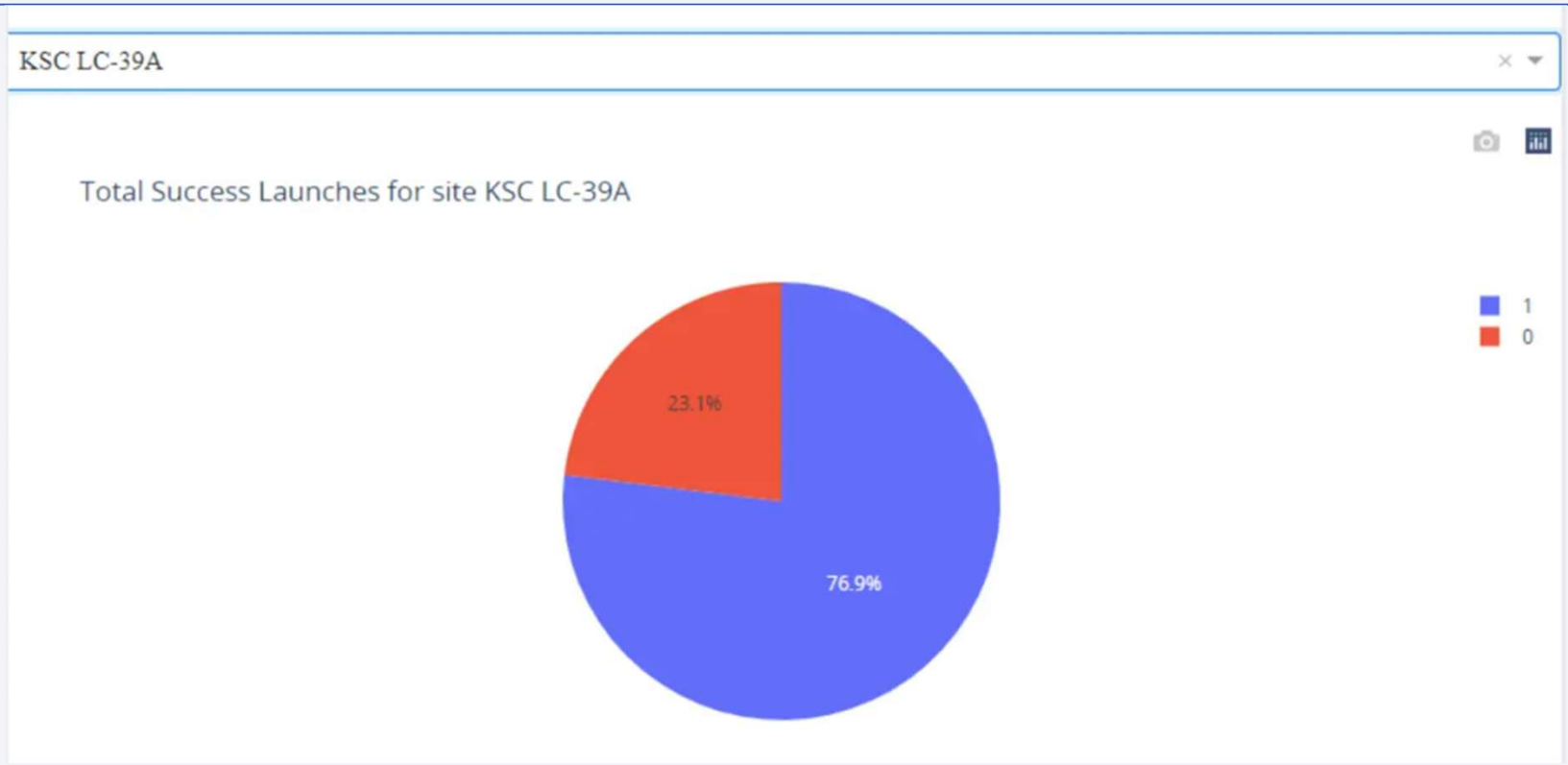

Section 4
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieve by each launch site



- We can observe that KSC LC-39A had the most successful launches from all the sites.

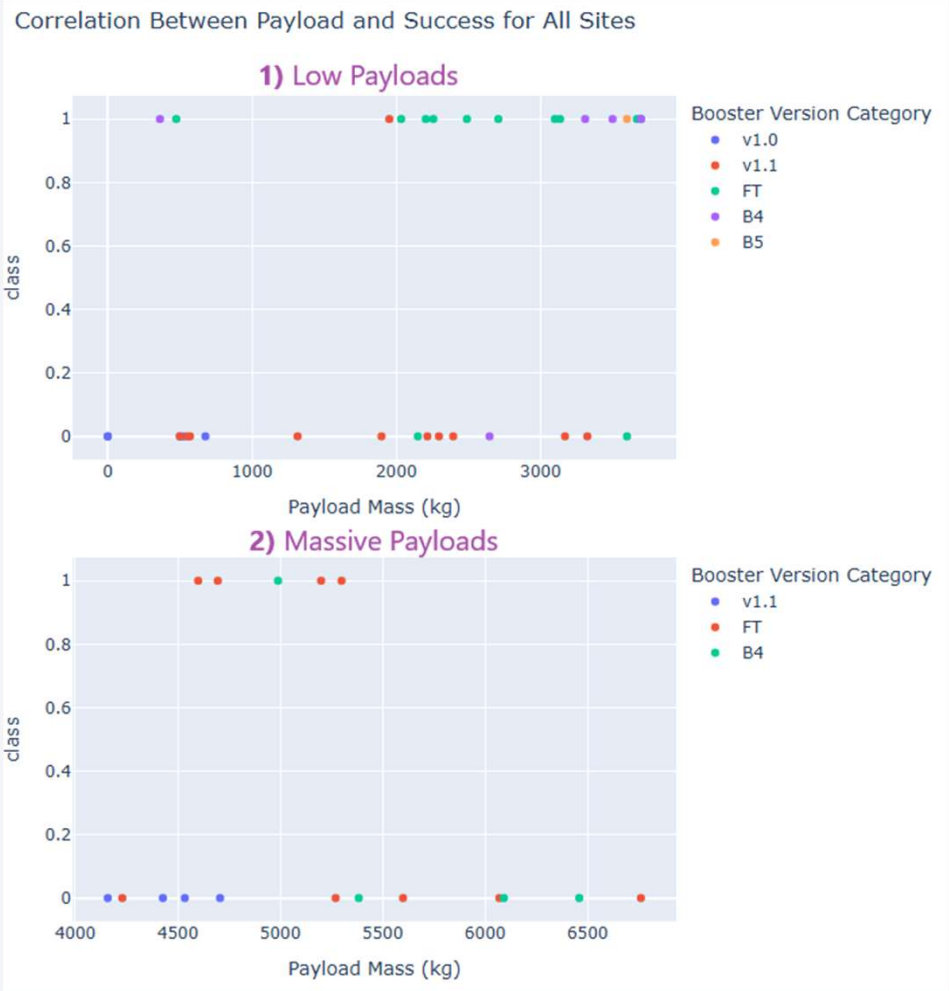
Launch site with highest launch success ratio



- The launch site KSC LC-39A also had the highest rate of successful launches, with a 76,9% success rate.

Payload vs. Launch Outcome 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:
  1) 0 - 4000 kg (low payloads)
  2) 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.

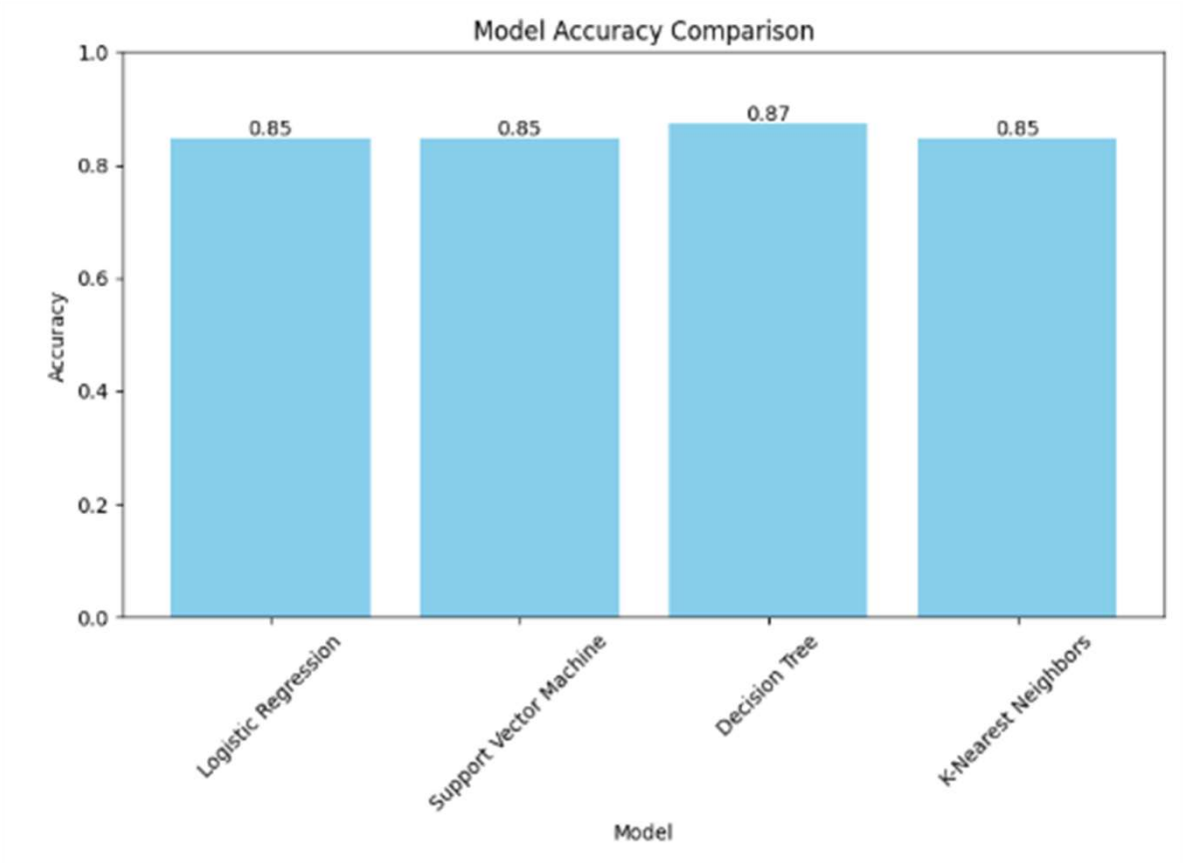


Section 5

Predictive Analysis (Classification)

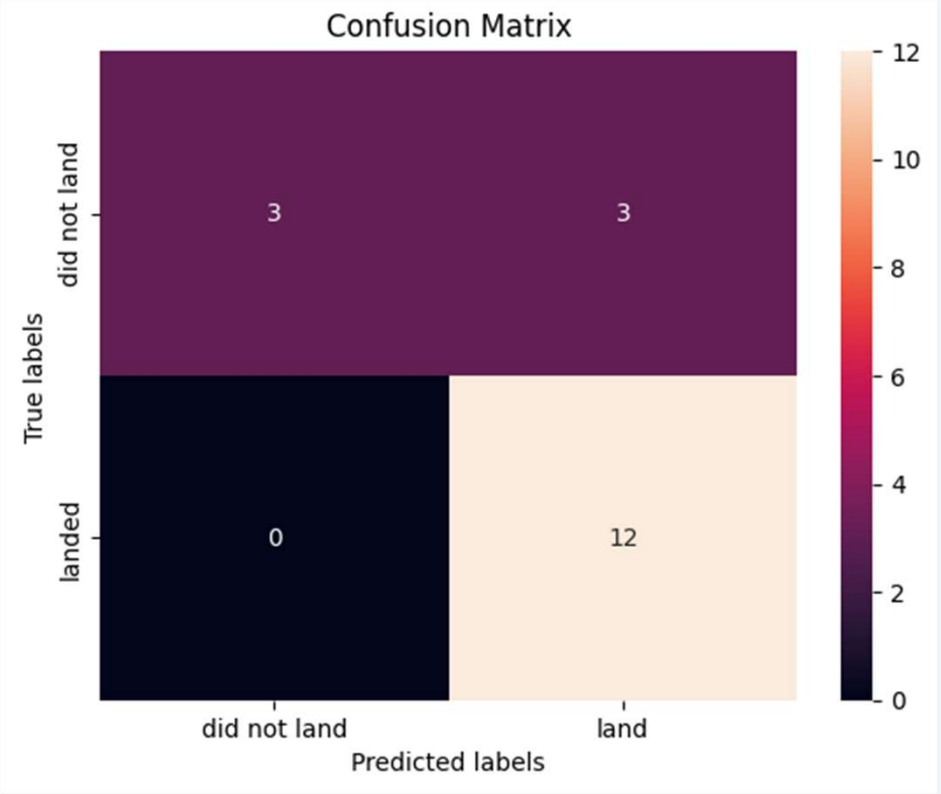
Classification Accuracy

- The **Decision Tree classifier** is the model with the **highest classification accuracy** with a score of **87,32%**



Confusion Matrix

- The **confusion matrix** (for all models) shows that **the classifier can distinguish between the different classes.**
- The **major problem is the false positives** (unsuccessful landing marked as successful by the classifier)



Conclusions

- ✓ Success Rate and Number of Flights: As the number of flights increases, the rate of success at a launch site increases, with most early flights being unsuccessful.

- ✓ Orbit Types and Success Rates: 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 while VLEO (Very Low Earth Orbit) launches are associated with heavier payloads.

- ✓ Launch Site Success: The launch site KSC LC-39A 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.

- ✓ Predictive Analysis: The Decision Tree classifier is the model with the highest classification accuracy with a score of 87,32%.

Appendix 1 – Data Collection SpaceX Rest API

- Customs functions followed to retrieve the required information and clean the data:

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

# To 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])

# 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)]
```

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

```
# 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']:
       if x:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

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

```
# 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']:
       if x:
         response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
         Longitude.append(response['longitude'])
         Latitude.append(response['latitude'])
         LaunchSite.append(response['name'])
```

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

```
# 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']:
       if load:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

Appendix 2 – Data Collection Web Scraping

- Customs functions and custom logic followed to fill up the launch_dict 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
```

```
#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
            launch_dict['Flight No.'].append(flight_number)
            print(flight_number)

            # Date value
            datatimelist=date_time(row[0])
            date = datatimelist[0].strip(',')
            launch_dict['Date'].append(date)
            #print(date)
            print(date)

            # Time value
            time = datatimelist[1]
            launch_dict['Time'].append(time)
            #print(time)
            print(time)

            # Booster version
            bv=booster_version(row[1])
            if not(bv):
                bv=row[1].a.string
            print(bv)
            launch_dict['Version Booster'].append(bv)

            # Launch Site
            launch_site = row[2].a.string if row[2].a else row[2].get_text(strip=True)
            launch_dict['Launch site'].append(launch_site)
            #print(launch_site)
            print(launch_site)
            ...
```

Appendix 3 – Interactive Dashboard with Plotly Dash

- Code used to build the Interactive Dashboard with Plotly Dash

```
# Read the airline data into pandas dataframe
spacex_df = pd.read_csv("spacex_launch_dash.csv")
min_payload = int(spacex_df['Payload Mass (kg)'].min())
max_payload = int(spacex_df['Payload Mass (kg)'].max())
marks = {i: str(i) for i in range(min_payload, max_payload + 1, 1000)}

# Create a dash application
app = dash.Dash(__name__)

# Create an app layout
app.layout = html.Div(children=[html.H1('SpaceX Launch Records Dashboard',
                                        style={'textAlign': 'center', 'color': '#503D36',
                                               'font-size': 40}),
                                # TASK 1: Add a dropdown list to enable Launch Site selection
                                # The default select value is for ALL sites
                                # dcc.Dropdown(id='site-dropdown',...)
                                dcc.Dropdown(
                                    id='site-dropdown',
                                    options=[
                                        {'label': 'All Sites', 'value': 'ALL'},
                                        {'label': 'CCAFS LC-40', 'value': 'CCAFS LC-40'},
                                        {'label': 'VAFB SLC-4E', 'value': 'VAFB SLC-4E'},
                                        {'label': 'KSC LC-39A', 'value': 'KSC LC-39A'},
                                        {'label': 'CCAFS SLC-40', 'value': 'CCAFS SLC-40'}
                                    ],
                                    value='ALL',
                                    placeholder='Select a Launch Site here',
                                    searchable=True
                                ),

                                html.Br(),

                                # TASK 2: Add a pie chart to show the total successful launches count for
                                # If a specific launch site was selected, show the Success vs. Failed coun
                                # This task continues below:
                                html.Div(dcc.Graph(id='success-pie-chart')),
                                html.Br(),

                                html.P("Payload range (Kg):"),
                                # TASK 3: Add a slider to select payload range
                                #dcc.RangeSlider(id='payload-slider',...)
                                dcc.RangeSlider(
                                    id='payload-slider',
                                    min=min_payload,
                                    max=max_payload,
                                    step=1000,
                                    marks=marks,
                                    value=[min_payload, max_payload]
                                ),

                                html.Br(),
                                # TASK 4: Add a scatter chart to show the correlation between payload and
                                # This task continues below:
                                html.Div(dcc.Graph(id='success-payload-scatter-chart')),
                            ])
```

```
@app.callback(
    Output(component_id='success-pie-chart', component_property='figure'),
    Input(component_id='site-dropdown', component_property='value')
)
def update_pie_chart(selected_site):
    if selected_site == 'ALL':
        # For all sites, show total success launches by site
        fig = px.pie(
            spacex_df,
            names='Launch Site',
            values='class',
            title='Total Success Launches by Site'
        )
    else:
        # Filter data for the selected site
        filtered_df = spacex_df[spacex_df['Launch Site'] == selected_site]
        # Count success (1) vs. failure (0)
        outcome_counts = filtered_df['class'].value_counts().reset_index()
        outcome_counts.columns = ['class', 'count']
        # Label classes
        outcome_counts['class'] = outcome_counts['class'].map({1: 'Success', 0: 'Failure'})
        fig = px.pie(
            outcome_counts,
            names='class',
            values='count',
            title=f'Success vs. Failure for {selected_site}'
        )
    return fig
# Add a callback function for `site-dropdown` and `payload-slider` as inputs.
@app.callback(
    Output(component_id='success-payload-scatter-chart', component_property='figure'),
    [
        Input(component_id='site-dropdown', component_property='value'),
        Input(component_id='payload-slider', component_property='value')
    ]
)
def update_scatter_chart(selected_site, selected_payload):
    low, high = selected_payload
    filtered_df = spacex_df[
        (spacex_df['Payload Mass (kg)'] >= low) &
        (spacex_df['Payload Mass (kg)'] <= high)
    ]
    if selected_site == 'ALL':
        fig = px.scatter(
            filtered_df, x='Payload Mass (kg)', y='class',
            color='Booster Version Category',
            title='Correlation Between Payload and Success for All Sites'
        )
    else:
        site_df = filtered_df[filtered_df['Launch Site'] == selected_site]
        fig = px.scatter(
            site_df, x='Payload Mass (kg)', y='class',
            color='Booster Version Category',
            title=f'Correlation Between Payload and Success for {selected_site}'
        )
    return fig
```

Thank you!