IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

Mark Khusid
December 03, 2022

# Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion

# Executive Summary

- In this study, we attempt to predict whether a Falcon 9 booster will land successfully.
- This prediction will allow a SpaceX competitor to come up with a competitive launch price.
- In this study, we will look at how the payload mass, launch site, number of first stage flights and orbits affect the success rate of the first stage landing.
- The methodology employed was:
  - Obtaining data from the SpaceX API and Wikipedia entry.
  - Exploratory Data Analysis (EDA) by plotting and Structured Query Language (SQL) queries.
  - Visual analytics with Folium and interactive dashboards with Dash.
  - Predicting mission success with machine learning models and selecting the best model type.
- Summary of all results:
  - It appears that VAFB SLC 4E and KSC LC 39A had superior mission success rates.
    - KSC LC-39A had the highest percentage of successful launches, where over ¾ of the launches from this site were successful.
  - Over time there is a trend towards increasing mission success rates and the plots of the data suggest that there are ranges of payload masses and specific locations where mission success is more prevalent.
  - Plots of the data also suggested that the ES-L1, GEO, HEO, and SSO orbits had the best results, while the SO orbit had the worst.
  - SQL queries of the data showed that the boosters which have successfully landed on a drone ship had payload masses greater than 2000kg, but less than 6000kg;  particularly for the “FT” booster version.
  - The K Nearest Neighbors and the Support Vector Machine models had the highest and most reliable accuracy and best scores.

# Introduction

- Project background and context:
- From Week 1 of the course:

  “SpaceX advertises Falcon 9 rocket launches on its website for a cost of 65 million dollars; other providers cost upwards of 165 million dollars each.” Much of the savings is derived from (sic) SpaceX’s ability to reuse the first stage.” [1]

- The goal of this project is to predict if the Falcon 9 first stage will land successfully.
- If we can determine if the first stage will land, then we can determine the cost of the launch.
- This information can be used by a competitor of SpaceX to bid against SpaceX.
- We wish to find the answers to:
  - How the payload mass, launch site, number of first stage flights and orbits affect the success rate of the first stage landing.



[1] IBM Data Science Capstone Project Course Material

Section 1

# Methodology

# Methodology: Obtain and Clean the Data (1/2)

- Data was collected from:
  - SpaceX REST API.
  - Web scraping from the SpaceX Wikipedia entry.
- After collection, data wrangling was performed:
  - SpaceX REST API:
    - Filtering the data to only include Falcon 9 launches.
    - Dealing with missing values.
  - Web scraping from Wikipedia:
    - Extracting column and variable names from HTML table headers.
    - Extracting the table data.
    - Creating a Pandas DataFrame from the column names, variable names and table data.

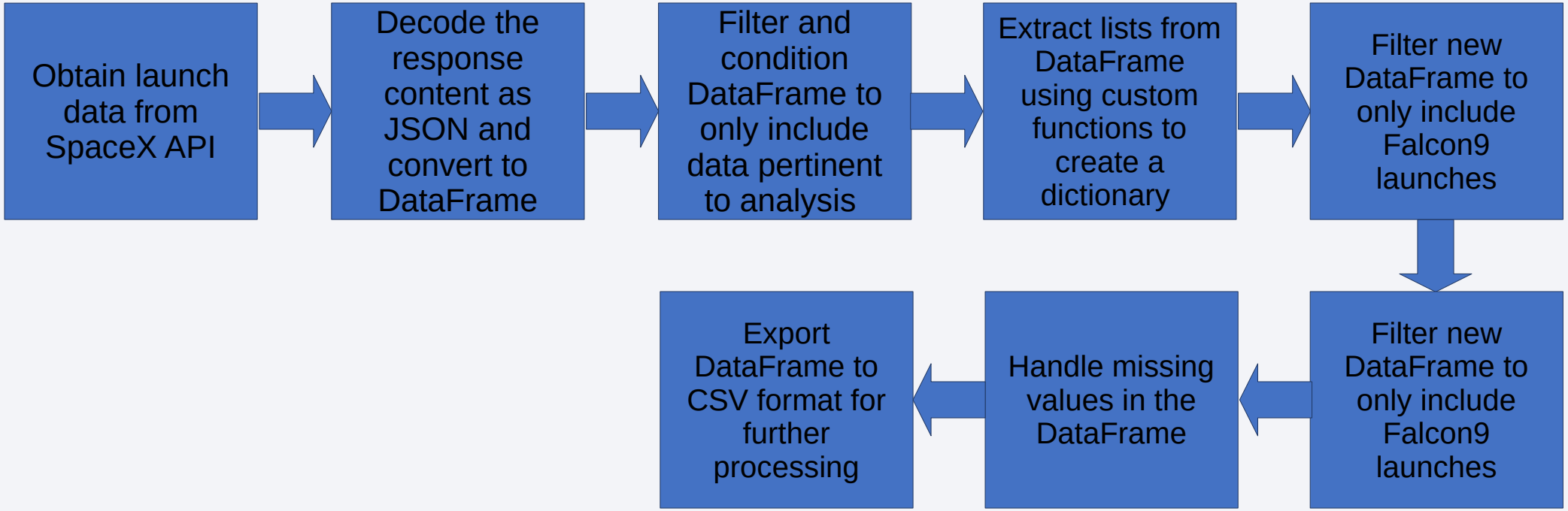
# Methodology: Analyze the Data (2/2)

- After the data was obtained and cleaned, it was analyzed by:
  - Performing exploratory data analysis (EDA) using visualization and SQL.
  - Performing interactive visual analytics using Folium and Plotly Dash.
  - Performing predictive analysis using classification models.

# Data Collection: SpaceX API Data

GitHub Link: Data Collection API

# Data Collection: SpaceX API Data Collection Flowchart



# Data Collection: SpaceX API Data Collection (1/8)

- Method of obtaining SpaceX API data
  - A static URL was used for consistent output on this project:

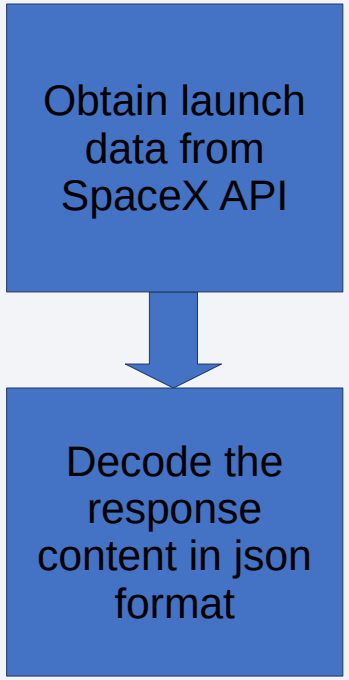
    static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
  - The response was captured as shown below:

    response = requests.get(static_json_url)

    response.json()


```
[{'fairings': {'reused': False,
   'recovery_attempt': False,
   'recovered': False,
   'ships': []},
  'links': {'patch': {'small': 'https://images2.imgbox.com/3c/0e/T8iJcSN3_o.png',
    'large': 'https://images2.imgbox.com/40/e3/GypSkayF_o.png'},
   'reddit': {'campaign': None,
    'launch': None,
    'media': None,
    'recovery': None},
...
```


  - Complete data can be viewed at:
    https://github.com/markkhusid/Applied_Data_Science_Capstone



# Data Collection: SpaceX API Data Collection (2/8)

- The JSON data was then converted into a Pandas Dataframe.

```
# Use json_normalize method to convert the json result into a dataframe

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

Convert JSON and content into a DataFrame

```
# Get the head of the dataframe
data.head(3)
```

| | static_fire_date_utc | static_fire_date_unix | tbd | net | window | rocket | success | details | crew | ships | capsules | payloads | launchpad | auto_update | failures | flight_number | name | date_utc | date_unix | date_local | date_precision | upcoming | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2006-03-17T00:00:00.000Z | 1.142554e+09 | False | False | 0.0 | 5e9d0d95eda69955f709d1eb | False | Engine failure at 33 seconds and loss of vehicle | [] | [] | [] | [5eb0e4b5b6c3bb0006eeb1e1] | 5e9e4502f5090995de566f86 | True | [{'time': 33, 'altitude': None, 'reason': 'merlin engine failure'}] | 1 | FalconSat | 2006-03-24T22:30:00.000Z | 1143239400 | 2006-03-25T10:30:00+12:00 | hour | False | '5 |
| 1 | None | NaN | False | False | 0.0 | 5e9d0d95eda69955f709d1eb | False | Successful first stage burn and transition to second stage, maximum altitude 289 km, Premature engine shutdown at T+7 min 30 s, Failed to reach orbit, Failed to recover first stage | [] | [] | [] | [5eb0e4b6b6c3bb0006eeb1e2] | 5e9e4502f5090995de566f86 | True | [{'time': 301, 'altitude': 289, 'reason': 'harmonic oscillation leading to premature engine shutdown'}] | 2 | DemoSat | 2007-03-21T01:10:00.000Z | 1174439400 | 2007-03-21T13:10:00+12:00 | hour | False | '5 |
| 2 | None | NaN | False | False | 0.0 | 5e9d0d95eda69955f709d1eb | False | Residual stage 1 thrust led to collision between stage 1 and stage 2 | [] | [] | [] | [5eb0e4b6b6c3bb0006eeb1e3, 5eb0e4b6b6c3bb0006eeb1e4] | 5e9e4502f5090995de566f86 | True | [{'time': 140, 'altitude': 35, 'reason': 'residual stage-1 thrust led to collision between stage 1 and stage 2'}] | 3 | Trailblazer | 2008-08-03T03:34:00.000Z | 1217734440 | 2008-08-03T15:34:00+12:00 | hour | False | '5 |

```
data.shape
```

```
(107, 42)
```

# Data Collection: SpaceX API Data Collection (3/8)

- The raw dataframe contained allot of information that was not pertinent to our analysis.
- This information was filtered out.
- Relevant data was conditioned to remove redundancies for multiple cores and was converted into the proper data structures:

Filter and condition DataFrame to only include data pertinent to analysis

```
# 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)]
```

# Data Collection: SpaceX API Data Collection (4/8)

- The conditioned dataframe was then parsed out by column to create a dictionary data structure
- The keys for the dictionary are the final column names, and the values are lists that were extracted using custom functions.

Extract lists from DataFrame using custom functions to create a dictionary

```
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}
```

  - The custom functions can be viewed at: https://github.com/markkhusid/Applied_Data_Science_Capstone

# Data Collection: SpaceX API Data Collection (5/8)

- The new Dataframe:

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

Show the summary of the dataframe

```
# Show the head of the dataframe
launch_df.head(10)
```

| | FlightNumber | Date | BoosterVersion | PayloadMass | Orbit | LaunchSite | Outcome | Flights | GridFins | Reused | Legs | LandingPad | Block | Reus |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2006-03-24 | Falcon 1 | 20.0 | LEO | Kwajalein Atoll | None None | 1 | False | False | False | None | NaN | |
| 1 | 2 | 2007-03-21 | Falcon 1 | NaN | LEO | Kwajalein Atoll | None None | 1 | False | False | False | None | NaN | |
| 2 | 4 | 2008-09-28 | Falcon 1 | 165.0 | LEO | Kwajalein Atoll | None None | 1 | False | False | False | None | NaN | |
| 3 | 5 | 2009-07-13 | Falcon 1 | 200.0 | LEO | Kwajalein Atoll | None None | 1 | False | False | False | None | NaN | |
| 4 | 6 | 2010-06-04 | Falcon 9 | NaN | LEO | CCSFS SLC 40 | None None | 1 | False | False | False | None | 1.0 | |
| 5 | 8 | 2012-05-22 | Falcon 9 | 525.0 | LEO | CCSFS SLC 40 | None None | 1 | False | False | False | None | 1.0 | |
| 6 | 10 | 2013-03-01 | Falcon 9 | 677.0 | ISS | CCSFS SLC 40 | None None | 1 | False | False | False | None | 1.0 | |
| 7 | 11 | 2013-09-29 | Falcon 9 | 500.0 | PO | VAFB SLC 4E | False Ocean | 1 | False | False | False | None | 1.0 | |
| 8 | 12 | 2013-12-03 | Falcon 9 | 3170.0 | GTO | CCSFS SLC 40 | None None | 1 | False | False | False | None | 1.0 | |
| 9 | 13 | 2014-01-06 | Falcon 9 | 3325.0 | GTO | CCSFS SLC 40 | None None | 1 | False | False | False | None | 1.0 | |

Extract lists from DataFrame using custom functions to create a dictionary

The complete DataFrame can be viewed at:

https://github.com/markkhusid/Applied_Data_Science_Capstone

# Data Collection: SpaceX API Data Collection (6/8)

- The new contains entries for both the Falcon1 and Falcon9 launch vehicles.
- Since our study is only focused on the Falcon9, we filter the DataFrame to only include Falcon9 launches.

```
launch_df_falcon9 = launch_df[launch_df['BoosterVersion'] != 'Falcon 1']
```

```
launch_df_falcon9.head(10)
```

| | FlightNumber | Date | BoosterVersion | PayloadMass | Orbit | LaunchSite | Outcome | Flights | GridFins | Reused | Legs | LandingPad | Block | Reu: |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 6 | 2010-06-04 | Falcon 9 | NaN | LEO | CCSFS SLC 40 | None None | 1 | False | False | False | None | 1.0 | |
| 5 | 8 | 2012-05-22 | Falcon 9 | 525.0 | LEO | CCSFS SLC 40 | None None | 1 | False | False | False | None | 1.0 | |
| 6 | 10 | 2013-03-01 | Falcon 9 | 677.0 | ISS | CCSFS SLC 40 | None None | 1 | False | False | False | None | 1.0 | |
| 7 | 11 | 2013-09-29 | Falcon 9 | 500.0 | PO | VAFB SLC 4E | False Ocean | 1 | False | False | False | None | 1.0 | |
| 8 | 12 | 2013-12-03 | Falcon 9 | 3170.0 | GTO | CCSFS SLC 40 | None None | 1 | False | False | False | None | 1.0 | |
| 9 | 13 | 2014-01-06 | Falcon 9 | 3325.0 | GTO | CCSFS SLC 40 | None None | 1 | False | False | False | None | 1.0 | |
| 10 | 14 | 2014-04-18 | Falcon 9 | 2296.0 | ISS | CCSFS SLC 40 | True Ocean | 1 | False | False | True | None | 1.0 | |
| 11 | 15 | 2014-07-14 | Falcon 9 | 1316.0 | LEO | CCSFS SLC 40 | True Ocean | 1 | False | False | True | None | 1.0 | |
| 12 | 16 | 2014-08-05 | Falcon 9 | 4535.0 | GTO | CCSFS SLC 40 | None None | 1 | False | False | False | None | 1.0 | |
| 13 | 17 | 2014-09-07 | Falcon 9 | 4428.0 | GTO | CCSFS SLC 40 | None None | 1 | False | False | False | None | 1.0 | |

Filter new DataFrame to only include Falcon9 launches

The complete DataFrame can be viewed at:

https://github.com/markkhusid/Applied_Data_Science_Capstone

# Data Collection: SpaceX API Data Collection (7/8)

- The Falcon9 DataFrame contains some missing values.
- Specifically, the Payload Mass column has 5 entries that were NaN.
  - These will be filled in with the mean of the column.

Handle missing values in the DataFrame

```
# Calculate the mean value of PayloadMass column
payload_mass_mean = data_falcon9['PayloadMass'].mean();
# Replace the np.nan values with its mean value
```

```
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(to_replace = np.nan, value=payload_mass_mean);
```

```
data_falcon9.isnull().sum()
```

```
FlightNumber       0
Date               0
BoosterVersion     0
PayloadMass        0
Orbit              0
LaunchSite         0
Outcome            0
Flights            0
GridFins           0
Reused             0
Legs               0
LandingPad        26
Block              0
ReusedCount        0
Serial             0
Longitude          0
Latitude           0
dtype: int64
```

# Data Collection: SpaceX API Data Collection (8/8)

- Finally, the processed DataFrame is exported as a CSV file for further processing.

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

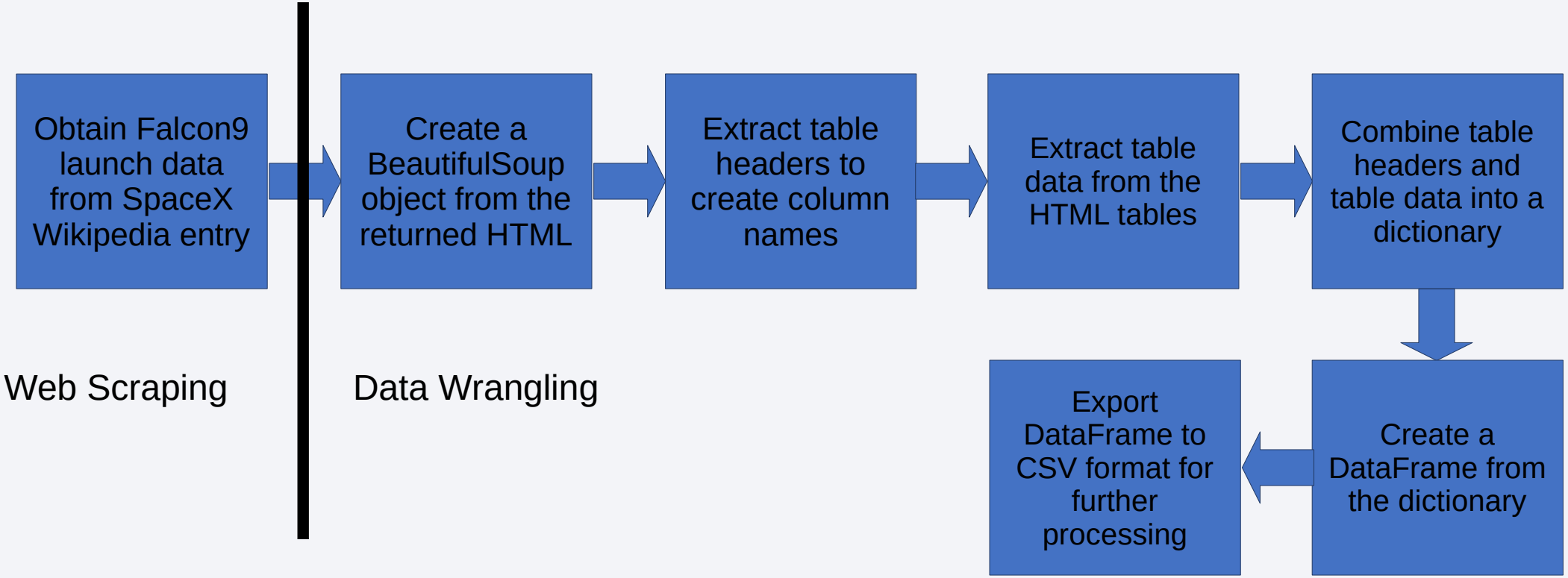
| FlightNumber | Date | BoosterVersion | PayloadMass | Orbit | LaunchSite | Outcome |
|---|---|---|---|---|---|---|
| 1 | 2010-06-04 | Falcon 9 | 6104.959411764706 | LEO | CCAFS SLC 40 | None None |
| 2 | 2012-05-22 | Falcon 9 | 525.0 | LEO | CCAFS SLC 40 | None None |
| 3 | 2013-03-01 | Falcon 9 | 677.0 | ISS | CCAFS SLC 40 | None None |
| 4 | 2013-09-29 | Falcon 9 | 500.0 | PO | VAFB SLC 4E | False Ocean |
| 5 | 2013-12-03 | Falcon 9 | 3170.0 | GTO | CCAFS SLC 40 | None None |
| 6 | 2014-01-06 | Falcon 9 | 3325.0 | GTO | CCAFS SLC 40 | None None |
| 7 | 2014-04-18 | Falcon 9 | 2296.0 | ISS | CCAFS SLC 40 | True Ocean |
| 8 | 2014-07-14 | Falcon 9 | 1316.0 | LEO | CCAFS SLC 40 | True Ocean |
| 9 | 2014-08-05 | Falcon 9 | 4535.0 | GTO | CCAFS SLC 40 | None None |
| 10 | 2014-09-07 | Falcon 9 | 4428.0 | GTO | CCAFS SLC 40 | None None |
| 11 | 2014-09-21 | Falcon 9 | 2216.0 | ISS | CCAFS SLC 40 | False Ocean |
| 12 | 2015-01-10 | Falcon 9 | 2395.0 | ISS | CCAFS SLC 40 | False ASDS |
| 13 | 2015-02-11 | Falcon 9 | 570.0 | ES-L1 | CCAFS SLC 40 | True Ocean |
| 14 | 2015-04-14 | Falcon 9 | 1898.0 | ISS | CCAFS SLC 40 | False ASDS |
| 15 | 2015-04-27 | Falcon 9 | 4707.0 | GTO | CCAFS SLC 40 | None None |
| 16 | 2015-06-28 | Falcon 9 | 2477.0 | ISS | CCAFS SLC 40 | None ASDS |
| 17 | 2015-12-22 | Falcon 9 | 2034.0 | LEO | CCAFS SLC 40 | True RTLS |
| 18 | 2016-01-17 | Falcon 9 | 553.0 | PO | VAFB SLC 4E | False ASDS |
| 19 | 2016-03-04 | Falcon 9 | 5271.0 | GTO | CCAFS SLC 40 | False ASDS |

Export DataFrame to CSV format for further processing

# Data Collection: SpaceX Wikipedia Entry Scraping

GitHub Link: Web Scraping

# Data Collection/Wrangling: SpaceX Wikipedia Entry Scraping  Flowchart




We will now discuss web scraping data from the SpaceX Wikipedia entry.

# Data Collection: SpaceX Wikipedia Entry Scraping (1/1)

- To obtain additional data for our analysis, the SpaceX Wikipedia entry's tables were scraped.
  - This involved loading the entire HTML contents of the Wikipedia webpage into a variable "launch_data".

Obtain Falcon9 launch data from SpaceX Wikipedia entry

static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"

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

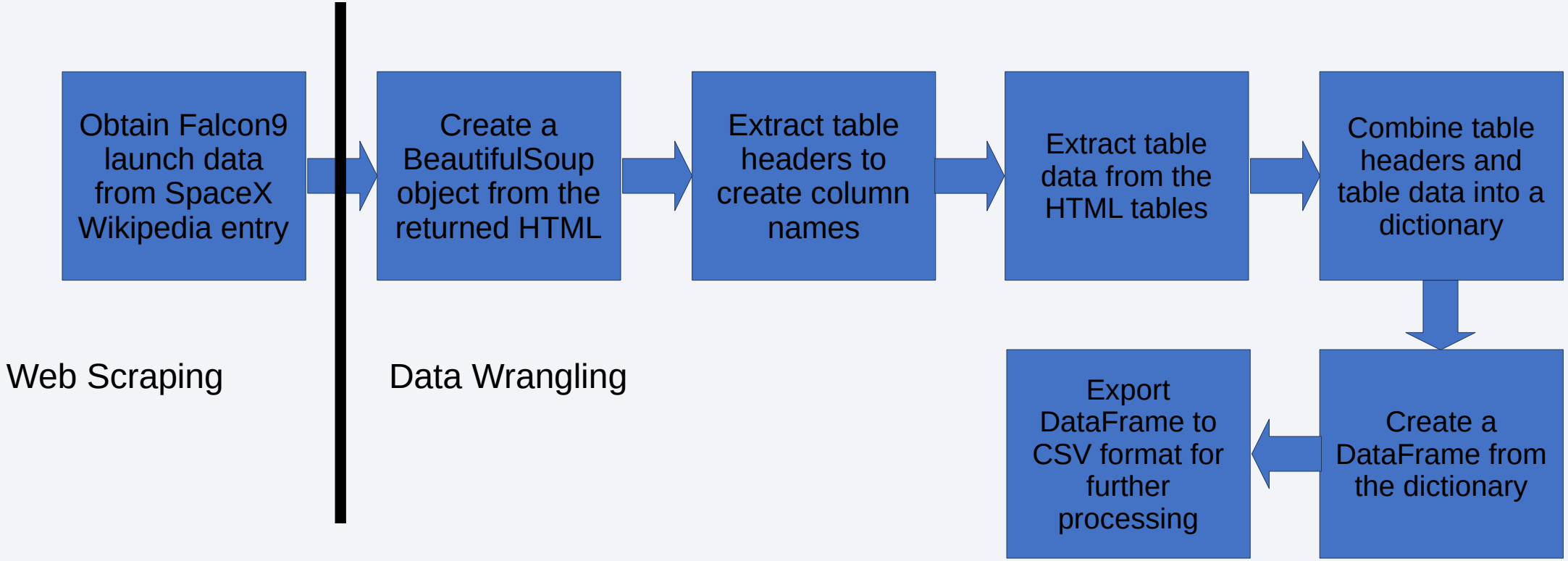
```
launch_data.text
```


```
'<!DOCTYPE html>\n<html class="client-nojs" lang="en" dir="ltr">\n<head>\n<meta charset="UTF-8"/>\n<title>List of Falcon 9 and
Falcon Heavy launches - Wikipedia</title>\n<script>document.documentElement.className="client-js";RLCONF={"wgBreakFrames":fals
e,"wgSeparatorTransformTable":["",""],"wgDigitTransformTable":["",""],"wgDefaultDateFormat":"dmy","wgMonthNames":["","Januar
y","February","March","April","May","June","July","August","September","October","November","December"],"wgRequestId":"c4b72ab
b-036e-47e8-a6c4-1adf374fef02","wgCSPNonce":false,"wgCanonicalNamespace":"","wgCanonicalSpecialPageName":false,"wgNamespaceNum
ber":0,"wgPageName":"List_of_Falcon_9_and_Falcon_Heavy_launches","wgTitle":"List of Falcon 9 and Falcon Heavy launches","wgCur
RevisionId":1117642752,"wgRevisionId":1027686922,"wgArticleId":37574004,"wgIsArticle":true,"wgIsRedirect":false,"wgAction":"vi
ew","wgUserName":null,"wgUserGroups":["*"],"wgCategories":["Source attribution","All articles with dead external links","Artic
les with dead external links from February 2021","Articles with permanently dead external links","CS1 Spanish-language sources
(es)","CS1 Indonesian-language sources (id)"\n,"CS1 errors: external links","CS1 maint: url-status","CS1 German-language sourc
es (de)","CS1 Korean-language sources (ko)","Articles with short description","Short description is different from Wikidata","
Use American English from January 2021","All Wikipedia articles written in American English","Use dmy dates from January 202
1","Articles containing potentially dated statements from March 2018","All articles containing potentially dated statements","
All articles with failed verification","Articles with failed verification from May 2021","Articles containing potentially date
d statements from April 2020","Pages using multiple image with auto scaled images","Featured lists","Articles containing video
clips","Falcon 9 and Falcon Heavy launches","SpaceX launch vehicles","Lists of rocket launches"],"wgPageContentLanguage":"e
n","wgPageContentModel":"wikitext","wgRelevantPageName":"List_of_Falcon_9_and_Falcon_Heavy_launches","wgRelevantArticleId":375
74004,"wgIsProbablyEditable":true,\n"wgRelevantPageIsProbablyEditable":true,"wgRestrictionEdit":[],"wgRestrictionMove":[],"wgF
laggedRevsParams":{"tags":{"status":{"levels":1}}},"wgVisualEditor":{"pageLanguageCode":"en","pageLanguageDir":"ltr","pageVari
```

# Data Wrangling

GitHub Link: Data Wrangling

# Data Collection/Wrangling: SpaceX Wikipedia Entry Scraping  Flowchart




We will now discuss Data Wrangling the Web Scraped Data from the SpaceX Wikipedia entry.

# Data Wrangling: SpaceX Wikipedia Entry Scraping (1/8)

- In order to parse the raw HTML data, the BeautifulSoup Python module was used

Create a BeautifulSoup object from the returned HTML

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

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

```
# Use soup.title attribute
soup.title
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

# Data Wrangling: SpaceX Wikipedia Entry Scraping (2/8)

- BeautifulSoup is used to extract all of the tables from the HTML.
- The data we are interested in is contained in the third table:

```python
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

Starting from the third table is our target table contains the actual launch records.

```python
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

```
<table class="wikitable plainrowheaders collapsible" style="width: 100%;">
<tbody><tr>
<th scope="col">Flight No.
</th>
<th scope="col">Date and<br/>time (<a href="/wiki/Coordinated_Universal_Time" title="Coordinated Universal Time">UTC</a>)
</th>
<th scope="col"><a href="/wiki/List_of_Falcon_9_first-stage_boosters" title="List of Falcon 9 first-stage boosters">Version,<b
r/>Booster</a> <sup class="reference" id="cite_ref-booster_11-0"><a href="#cite_note-booster-11">[b]</a></sup>
</th>
<th scope="col">Launch site
</th>
<th scope="col">Payload<sup class="reference" id="cite_ref-Dragon_12-0"><a href="#cite_note-Dragon-12">[c]</a></sup>
</th>
<th scope="col">Payload mass
</th>
<th scope="col">Orbit
</th>
<th scope="col">Customer
</th>
```

Extract table headers to create column names

# Data Wrangling: SpaceX Wikipedia Entry Scraping (3/8)

- A loop and an extraction function are used to extract the column names from the table headers:

Extract table headers to create column names

```
for table_header in first_launch_table_headers:
    extracted_header = extract_column_from_header(table_header)
    if (extracted_header != None) and (len(extracted_header) > 0):
        print(extracted_header)
        column_names.append(extracted_header)
```

```
Flight No.
Date and time ( )
Launch site
Payload
Payload mass
Orbit
Customer
Launch outcome
```

Check the extracted column names

```
print(column_names)
```

```
['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome']
```

# Data Wrangling: SpaceX Wikipedia Entry Scraping (4/8)

- Now that the column names are extracted, a dictionary is setup where the keys are the column names and the values are empty lists that will hold the extracted table data:

Extract table data from the HTML tables

We will create an empty dictionary with keys from the extracted column names in the previous task. Later, this dictionary will be converted into a Pandas dataframe

```
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']            = []
```

```
launch_dict
```

```
{'Flight No.': [],
 'Launch site': [],
 'Payload': [],
 'Payload mass': [],
 'Orbit': [],
 'Customer': [],
 'Launch outcome': [],
 'Version Booster': [],
 'Booster landing': [],
 'Date': [],
 'Time': []}
```

# Data Wrangling: SpaceX Wikipedia Entry Scraping (5/8)

- A loop is setup and in each run a table row is processed so that the appropriate entry lists in the dictionary are appended.

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            launch_dict['Flight No.'].append(flight_number)
            #print(flight_number)

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

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

            # Booster version
            # TODO: 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)
            #print(bv)
```

Combine table headers and table data into a dictionary

# Data Wrangling: SpaceX Wikipedia Entry Scraping (6/8)

- A loop is setup and in each run a table row is processed so that the appropriate entry lists in the dictionary are appended.

Combine table headers and table data into a dictionary

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            launch_dict['Flight No.'].append(flight_number)
            #print(flight_number)

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

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

            # Booster version
            # TODO: 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)
            #print(bv)
```

# Data Wrangling: SpaceX Wikipedia Entry Scraping (7/8)

- Now that the dictionary is filled up with the data that is pertinent to this analysis, we create a DataFrame from the dictionary:

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

```
df
```

| | Flight No. | Launch site | Payload | Payload mass | Orbit | Customer | Launch outcome | Version Booster | Booster landing | Date | Time |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | CCAFS | Dragon Spacecraft Qualification Unit | 0 | LEO | SpaceX | Success\n | F9 v1.0B0003.1 | Failure | 4 June 2010 | 18:45 |
| **1** | 2 | CCAFS | Dragon | 0 | LEO | NASA | Success | F9 v1.0B0004.1 | Failure | 8 December 2010 | 15:43 |
| **2** | 3 | CCAFS | Dragon | 525 kg | LEO | NASA | Success | F9 v1.0B0005.1 | No attempt\n | 22 May 2012 | 07:44 |
| **3** | 4 | CCAFS | SpaceX CRS-1 | 4,700 kg | LEO | NASA | Success\n | F9 v1.0B0006.1 | No attempt | 8 October 2012 | 00:35 |
| **4** | 5 | CCAFS | SpaceX CRS-2 | 4,877 kg | LEO | NASA | Success\n | F9 v1.0B0007.1 | No attempt\n | 1 March 2013 | 15:10 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **116** | 117 | CCSFS | Starlink | 15,600 kg | LEO | SpaceX | Success\n | F9 B5B1051.10 | Success | 9 May 2021 | 06:42 |
| **117** | 118 | KSC | Starlink | ~14,000 kg | LEO | SpaceX | Success\n | F9 B5B1058.8 | Success | 15 May 2021 | 22:56 |
| **118** | 119 | CCSFS | Starlink | 15,600 kg | LEO | SpaceX | Success\n | F9 B5B1063.2 | Success | 26 May 2021 | 18:59 |
| **119** | 120 | KSC | SpaceX CRS-22 | 3,328 kg | LEO | NASA | Success\n | F9 B5B1067.1 | Success | 3 June 2021 | 17:29 |
| **120** | 121 | CCSFS | SXM-8 | 7,000 kg | GTO | Sirius XM | Success\n | F9 B5 | Success | 6 June 2021 | 04:26 |

121 rows × 11 columns

Create a DataFrame from the dictionary

# Data Wrangling: SpaceX Wikipedia Entry Scraping (8/8)

- Finally, we export the DataFrame as a CSV file for further processing

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Export DataFrame to CSV format for further processing

| Flight No. | Launch site | Payload | Payload mass | Orbit | Customer | Launch outcome |
|---|---|---|---|---|---|---|
| 1 | CCAFS | Dragon Spacecraft ... | 0 | LEO | SpaceX | Success |
| 2 | CCAFS | Dragon | 0 | LEO | NASA | Success |
| 3 | CCAFS | Dragon | 525 kg | LEO | NASA | Success |
| 4 | CCAFS | SpaceX CRS-1 | 4,700 kg | LEO | NASA | Success |
| 5 | CCAFS | SpaceX CRS-2 | 4,877 kg | LEO | NASA | Success |
| 6 | VAFB | CASSIOPE | 500 kg | Polar orbit | MDA | Success |
| 7 | CCAFS | SES-8 | 3,170 kg | GTO | SES | Success |
| 8 | CCAFS | Thaicom 6 | 3,325 kg | GTO | Thaicom | Success |
| 9 | Cape Canaveral | SpaceX CRS-3 | 2,296 kg | LEO | NASA | Success |
| 10 | Cape Canaveral | Orbcomm-OG2 | 1,316 kg | LEO | Orbcomm | Success |
| 11 | Cape Canaveral | AsiaSat 8 | 4,535 kg | GTO | AsiaSat | Success |
| 12 | Cape Canaveral | AsiaSat 6 | 4,428 kg | GTO | AsiaSat | Success |
| 13 | Cape Canaveral | SpaceX CRS-4 | 2,216 kg | LEO | NASA | Success |
| 14 | Cape Canaveral | SpaceX CRS-5 | 2,395 kg | LEO | NASA | Success |
| 15 | Cape Canaveral | DSCOVR | 570 kg | HEO | USAF | Success |
| 16 | Cape Canaveral | ABS-3A | 4,159 kg | GTO | ABS | Success |
| 17 | Cape Canaveral | SpaceX CRS-6 | 1,898 kg | LEO | NASA | Success |
| 18 | Cape Canaveral | TürkmenÄlem 52°E... | 4,707 kg | GTO | | Success |
| 19 | Cape Canaveral | SpaceX CRS-7 | 1,952 kg | LEO | NASA | Failure |

# Exploratory Data Analysis (EDA) with Data Visualization

GitHub Link: EDA with Data Visualization

# EDA with Data Visualization (1/2)

- To facilitate the study, an Exploratory Data Analysis (EDA) was undertaken by plotting the data to glean relationships between features.
- A summary of plotted charts is as follows:
  - A category plot of Payload Mass vs. Flight Number, with the category being mission success or failure.
    - This is to glean a relationship between mission success and payload mass.
  - A category plot of Launch Site vs. Payload Mass, with the category being mission success or failure.
    - This is to glean a relationship between the launch site, payload mass, and mission success or failure.
  - A bar plot of the normalized average success rate over all orbit types vs. Orbit Type.
    - This is to assess which orbit types have a higher mission success rate.

# EDA with Data Visualization (2/2)

- A category plot of the Orbit Type vs Flight Number
  - This is to glean a relationship between the Orbit Type and the mission success rate.
- A category plot of the Orbit Type and the Payload Mass, with the category being the mission success rate.
  - This is to glean a relationship between the Orbit Type, the Payload Mass and the mission success rate
- A line plot of the normalized mission success rate vs time in years.
  - This is to see the relationship of the reliability of a successful mission over time.

# Exploratory Data Analysis (EDA) with Structured Query Language (SQL)

GitHub Link: EDA with SQL

# EDA with  SQL (1/2)

For Exploratory Data Analysis (EDA), the data was entered into a database to enable search queries.  The following queries were performed:

- Task 1: A listing of all distinct launch sites.
- Task 2: A listing of 5 records from the data where the launch sites begin with the string “CCA”
- Task 3: A display of the total payload mass in kg launched by NASA (CRS)
- Task 4: A display of the average payload mass in kg carried by the booster version F9 v1.1
- Task 5: A display of the first date that a successful outcome of landing on a ground pad was achieved.

# EDA with SQL (2/2)

- Task 6: A listing of the names of the boosters which have had success in landing on a drone ship, and have payload masses between 4000 kg and 6000 kg.
- Task 7: A listing of the total number of successful and failure mission outcomes.
- Task 8: A listing of the names of the booster versions which have carried the maximum payload mass in kg.
- Task 9: A listing of the records which will display the month names, failed landing outcomes in drone ship, booster versions, and the launch site for the months in year 2015.
- Task 10: Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

# Interactive Visual Analytics with Folium

GitHub Link: Interactive Visual Analytics with Folium

# Build an Interactive Map with Folium (1/2)

- In order to assess the affects of launch site to mission success, the launch sites were plotted on an interactive map, along with distance calculations to various topographical features.  The process was:
  - Task 1: Mark All Launch Sites on a Map
    - Folium’s Circle object was used to add circle map markers to the map at the locations of the launch sites
  - Task 2: Mark the Success/Failed Launches for Each Site on the Map
    - Folium’s MarkerCluster object was used to mark the mission successes or failures for each launch site.
  - Task 3: Calculate the Distances Between a Launch Site and Various Map Items
    - Folium’s mouse position meter was added to the map and the coordinates of various map items near to the launch sites were noted.
    - Using the geometry of rectangular coordinates, the distances from the launch sites to the various map items were calculated.
    - A line was then drawn from the launch site to each map item and a distance marker was added.
- Finally, questions were asked about the launch site’s distance to the various items on the map.

# Build an Interactive Map with Folium (2/2)

- The questions to answer were:
  Question 1) Are launch sites in close proximity to railways?
  Answer 1) Yes, CCAFS SLC-40 is about 1.28km away from a railway.  A railway is critical to supplying material to the launch site.

  Question 2) Are launch sites in close proximity to highways?
  Answer 2) Yes, CCAFS SLC-40 is about 0.6km away from a highway.  A highway is critical to supplying personnel and material to the launch site.

  Question 3) Are launch sites in close proximity to coastline?
  Answer 3) Yes, CCAFS SLC-40 is about 0.87kn away from the coastline.  The vehicle is launched Eastward over the ocean for safety reasons; therefore, close proximity to a coast is advantageous.

  Question 4) Do launch sites keep certain distance away from cities?
  Answer 4) Yes, CCAFS SLC-40 is about 51.74km away from the city of Melbourne.  This is done for safety reasons.

# Interactive Visual Analytics with Plotly Dash

GitHub Link: Interactive Visual Analytics with Plotly Dash

# Build a Dashboard with Plotly Dash

To enhance this study, an interactive dashboard application was created with Plotly Dash. The dashboard items are as follows:

- Task 1: A dropdown box that allows for the selection of all or specific launch sites. When a choice is selected, this triggers the replotting of plots on the dashboard.
- Task 2: A pie chart showing the:
  - Success rate of all launch sites if “All Sites” is selected in the dropdown box, or
  - The success and failure rate of a specific launch site when that launch site is selected from the dropdown box.
- Task 3: A range slider that allows for the filtering of payload masses to values between 0kg to 10,000kg. Upon selection of a range, the scatter plot in Task 4 is updated.
- Task 4: A scatter plot of:
  - All mission success or failures vs payload mass as selected by the range slider in Task 3 if “All Sites” is selected from the drop down box, or
  - The mission success or failures vs payload mass as selected by the range slider in Task 3 for a specific launch site as selected by the dropdown box.
  - In all cases of the dropdown box and range slider settings, the color of the plotted points represents the mission success or failure.

# Predictive Analysis (Classification)

GitHub Link: Predictive Analysis (Classification)

# Predictive Analysis (Classification) (1/3)

In furtherance of the study, predictive analysis was used to create several machine learning models that attempt to predict mission success vs various features, such as:

- Flight number
- Payload mass
- Number of flights
- Booster version
- Orbit type
- Whether booster was resused
- If booster was equipped with grid fins and/or legs.

# Predictive Analysis (Classification) (2/3)

- These features were fed into various models as training data, with the target being mission success or not.
- Task 1,2: The features were normalized using SciKit Preprocessing StandardScaler.
- Task 3: The features data was then split into a training and test set by a ratio of 80% training and 20% testing.
  - This resulted in 72 rows of training data and 18 rows of test data, for a total of 90 rows.
  - The training and test features data had 83 columns.

# Predictive Analysis (Classification) (3/3)

- With training and test data ready, the following models were created and tested:
  - Task 4,5: Logistic Regression
  - Task 6,7: Space Vector Machine
  - Task 8,9: Decision Tree Classifier
  - Task 10, 11: K Nearest Nearest Neighbor
- For all models:
  - A grid search was setup to tune for the best hyperparameters specific to the model.
  - A confusion matrix was generated to assess the model’s accuracy.
- Finally, the accuracy scores and best run scores were plotted to assess which model produced the best performance.

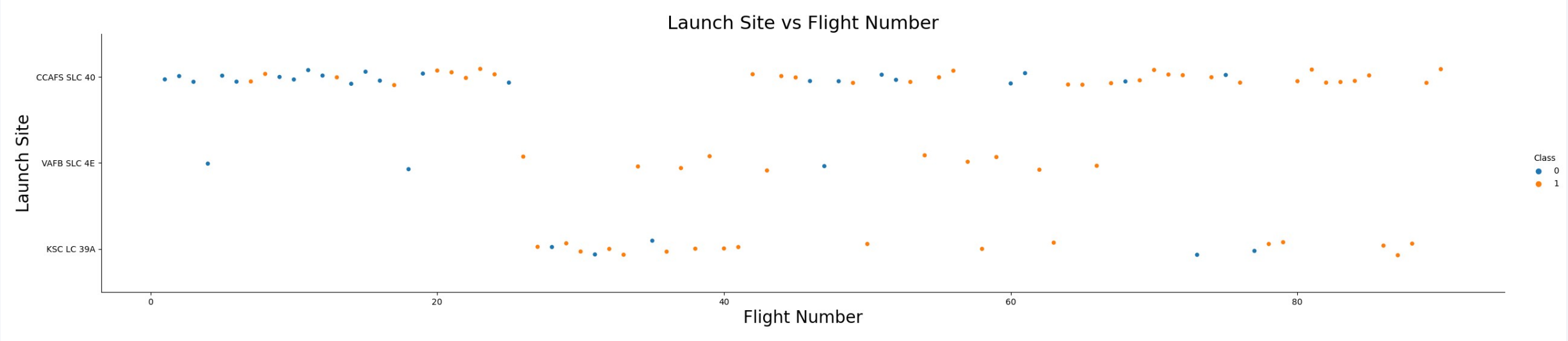
Section 2

# Insights drawn from EDA

# Exploratory Data Analysis (EDA) with Data Visualization

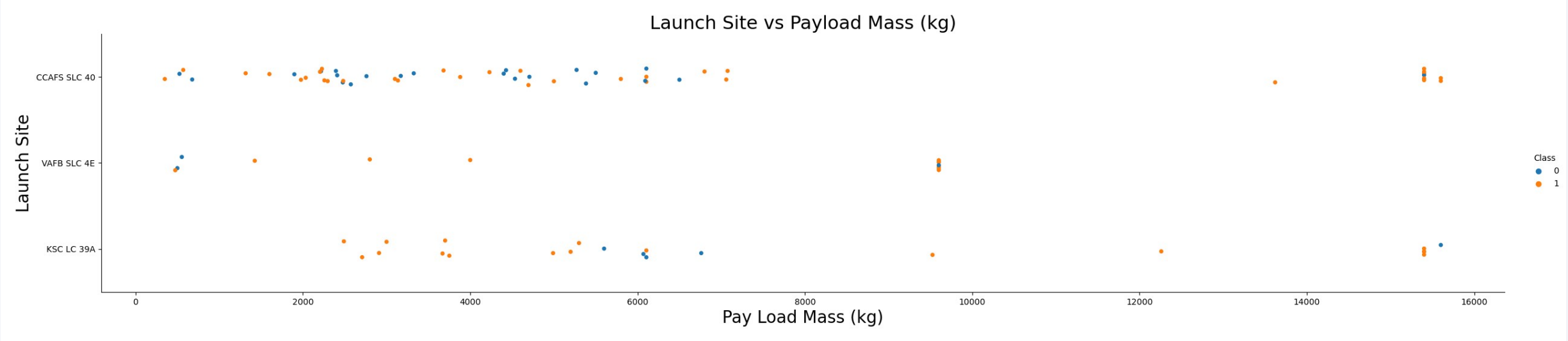
GitHub Link: EDA with Data Visualization

# EDA with Data Visualization: Launch Site vs Flight Number



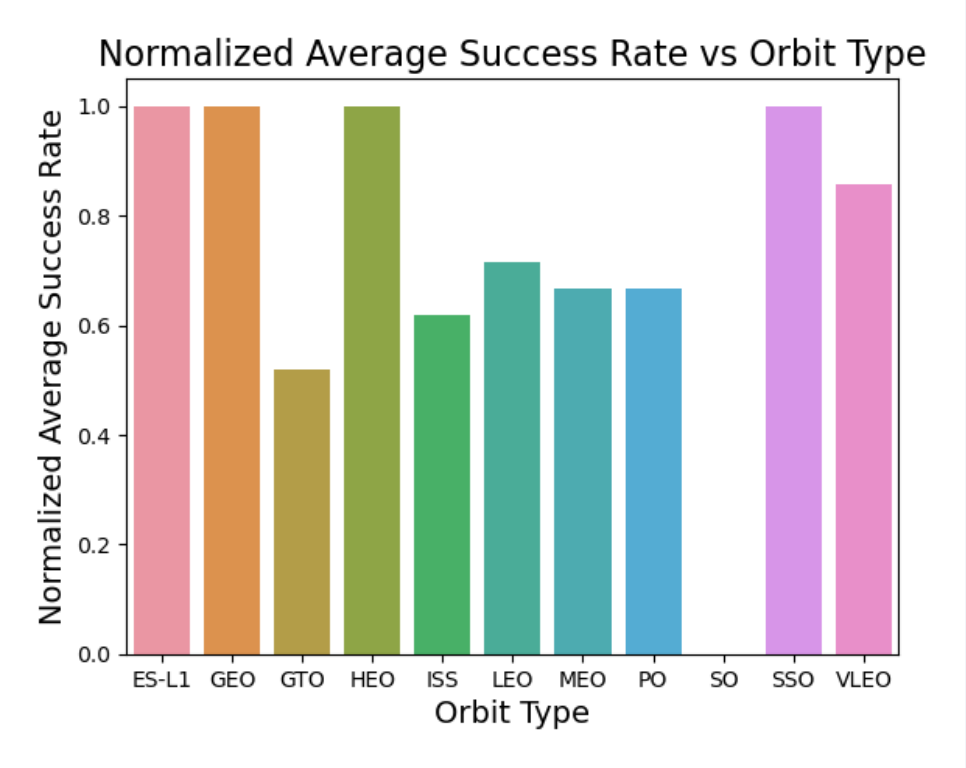

- In the above figure, we can see the relationship between the various launch sites, flight numbers and mission success.
- In the above figure, it appears that VAFB SLC 4E and KSC LC 39A had good mission success rates, while CCAFS SLC 40 appears to be 50/50.
- The figure also shows a trend towards increasing mission success rates.

# EDA with Data Visualization: Launch Site vs Payload Mass (kg)



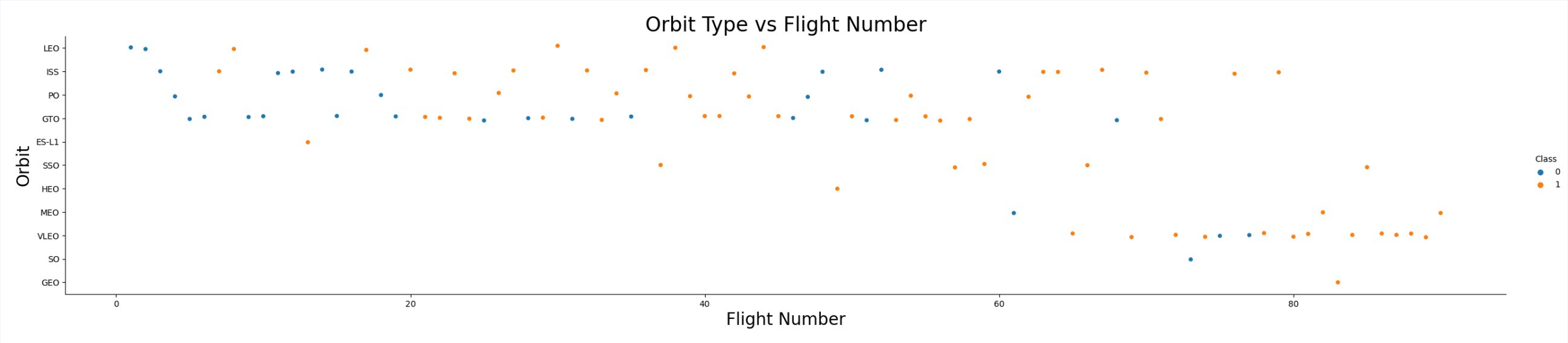

- In the above figure, we can see the relationship between the various launch sites, payload mass in kg and mission success.
- The figure suggests that there are ranges of payload masses and specific locations where mission success is more prevalent.

# EDA with Data Visualization: Normalized Average Success Rate vs Orbit Type



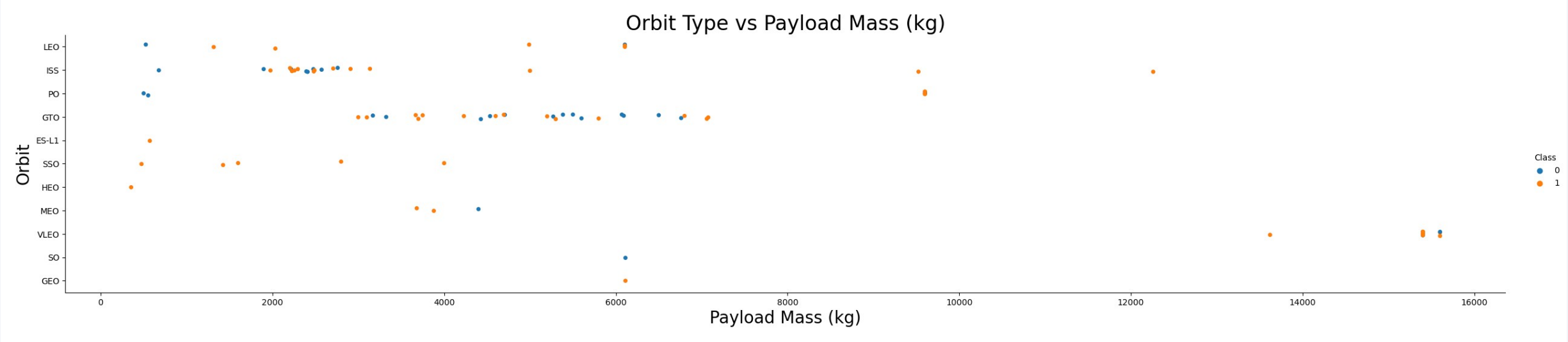

- To make the above figure, the data were grouped by orbit type, and the normalized average success rate for that orbit was calculated.
- From the above figure, we can see that the ES-L1, GEO, HEO, and SSO orbits had the best results, while the SO orbit had the worst.

# EDA with Data Visualization: Orbit Type vs Flight Number



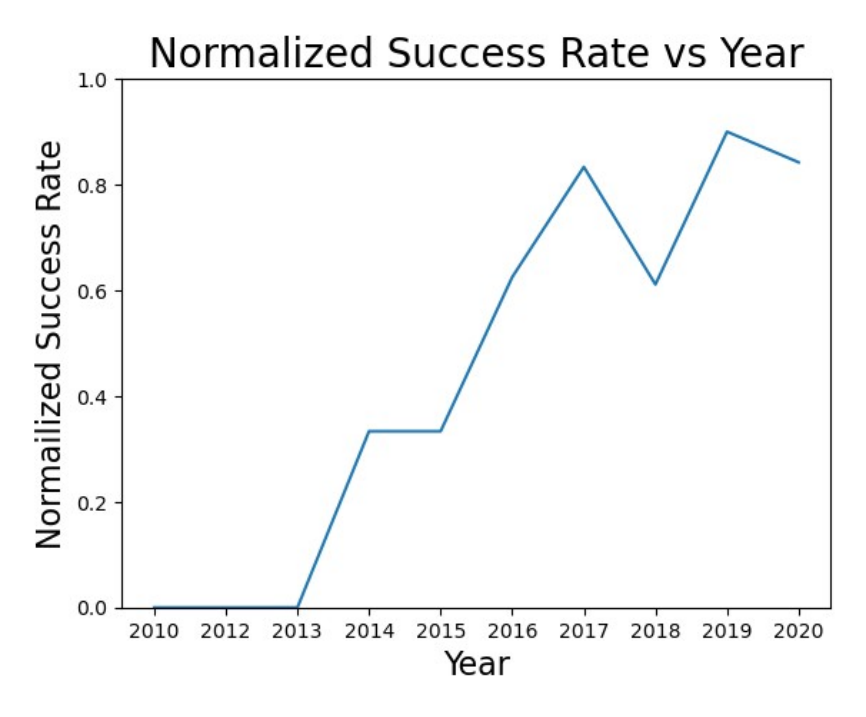

- In the above figure, we can see the relationship between the various orbit types, flight numbers and mission success.
- It can be seen that the graph on the previous slide doesn’t tell the whole story with regards to high normalized average success rate.
- A better insight in the mission success rate can be obtained by looking at orbit types that have more data points.

# EDA with Data Visualization: Orbit Type vs Payload Mass (kg)




- In the above figure, we can see the relationship between the various orbit types, payload mass and mission success.
- The figure suggests that there are ranges of payload masses and orbit types where mission success is more prevalent.

# EDA with Data Visualization: Normalized Average Success Rate vs Orbit Type




- To make the above figure, the data were grouped by launch year, filtered for success rate and the average of the success rate was taken for that year.
- From the above figure, we can see that mission success rate generally trends upwards in time, despite a drop in the 2018 and 2020 time periods.

# Exploratory Data Analysis (EDA) with Structured Query Language (SQL)

GitHub Link: EDA with SQL

# EDA with SQL: Create Table from DataFrame

The conditioned Dataframe was converted into a SQLite Table as shown below:

```
df.to_sql("SPACEXTBL", con, if_exists='replace', index=False, method="multi")
```

```
101
```

```
%sql select * from SPACEXTBL
```

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 18-07-2016 | 04:45:00 | F9 FT B1025.1 | CCAFS LC-40 | SpaceX CRS-9 | 2257 | LEO (ISS) | NASA (CRS) |
| 14-08-2016 | 05:26:00 | F9 FT B1026 | CCAFS LC-40 | JCSAT-16 | 4600 | GTO | SKY Perfect JSAT Group |
| 14-01-2017 | 17:54:00 | F9 FT B1029.1 | VAFB SLC-4E | Iridium NEXT 1 | 9600 | Polar LEO | Iridium Communications |
| 19-02-2017 | 14:39:00 | F9 FT B1031.1 | KSC LC-39A | SpaceX CRS-10 | 2490 | LEO (ISS) | NASA (CRS) |
| 16-03-2017 | 06:00:00 | F9 FT B1030 | KSC LC-39A | EchoStar 23 | 5600 | GTO | EchoStar |
| 30-03-2017 | 22:27:00 | F9 FT B1021.2 | KSC LC-39A | SES-10 | 5300 | GTO | SES |
| 01-05-2017 | 11:15:00 | F9 FT B1032.1 | KSC LC-39A | NROL-76 | 5300 | LEO | NRO |

# EDA with SQL: All Launch Site Names

## Task 1

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

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

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

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

From the above query response, we can see the names of the distinct launch sites used by SpaceX in this study.

# EDA with SQL: Launch Site Names that Begin with 'CCA'

## Task 2

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

```
%sql SELECT * FROM SPACEXTBL 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 |
|---|---|---|---|---|---|---|---|---|---|
| 04-06-2010 | 18:45:00 | F9 v1.0 B0003 | CCAFS LC-40 | Dragon Spacecraft Qualification Unit | 0 | LEO | SpaceX | Success | Failure (parachute) |
| 08-12-2010 | 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) |
| 22-05-2012 | 07:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525 | LEO (ISS) | NASA (COTS) | Success | No attempt |
| 08-10-2012 | 00:35:00 | F9 v1.0 B0006 | CCAFS LC-40 | SpaceX CRS-1 | 500 | LEO (ISS) | NASA (CRS) | Success | No attempt |
| 01-03-2013 | 15:10:00 | F9 v1.0 B0007 | CCAFS LC-40 | SpaceX CRS-2 | 677 | LEO (ISS) | NASA (CRS) | Success | No attempt |

In this query request, we find 5 records where launch sites begin with “CCA”.

# EDA with SQL: Total Payload Mass

## Task 3

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

```
%%sql
SELECT

SUM(PAYLOAD_MASS__KG_) AS Total_Payload_Mass_kg

FROM SPACEXTBL

WHERE Customer = "NASA (CRS)";
```

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

| Total_Payload_Mass_kg |
|---|
| 45596 |

The total payload mass in kg launched from NASA (CRS) is calculated and shown as a query results.

# EDA with SQL: Average Payload Mass by F9 v1.1

## Task 4

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

```
%%sql
SELECT

AVG(PAYLOAD_MASS__KG_) AS Average_Payload_Mass_kg

FROM SPACEXTBL

WHERE Booster_Version like "F9 v1.1%";
```

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

| Average_Payload_Mass_kg |
|---|
| 2534.6666666666665 |

The average payload mass carried by booster version F9 v1.1 was then queried and the results are shown above.

# EDA with SQL: First Successful Ground Landing Date

## Task 5

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

*Hint:Use min function*

```
# Landing _Outcome # has a space in it, haha :) fix by enclosing in brackets
```

```
%%sql
SELECT

MIN(Date) AS Earliest_Succssful_Landing

FROM SPACEXTBL

WHERE [Landing _Outcome] = "Success (ground pad)";
```

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

| Earliest_Succssful_Landing |
|---|
| 01-05-2017 |

The dates of the first successful landing outcome on ground pad was then queried and the results are shown above.

# EDA with SQL: Successful Drone Ship Landing with Payload between 4000 and 6000

## Task 6

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

```
%%sql
SELECT Booster_Version, [Landing _Outcome], PAYLOAD_MASS__KG_

FROM SPACEXTBL

WHERE ([Landing _Outcome] = "Success (drone ship)" AND
       PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000);
```

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

| Booster_Version | Landing _Outcome | PAYLOAD_MASS__KG_ |
|---|---|---|
| F9 FT B1022 | Success (drone ship) | 4696 |
| F9 FT B1026 | Success (drone ship) | 4600 |
| F9 FT B1021.2 | Success (drone ship) | 5300 |
| F9 FT B1031.2 | Success (drone ship) | 5200 |

In the above query, the names of boosters which have successfully landed on a drone ship and had payload mass greater than 4000 but less than 6000 are shown.

# EDA with SQL: Total Number of Successful and Failure  Mission Outcomes

## Task 7

List the total number of successful and failure mission outcomes

```
%%sql
SELECT DISTINCT Mission_Outcome, COUNT(*) AS Total

FROM SPACEXTBL

GROUP BY Mission_Outcome;
```

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

| Mission_Outcome | Total |
|---|---|
| Failure (in flight) | 1 |
| Success | 99 |
| Success (payload status unclear) | 1 |

The above query shows the total number of successful and failure mission outcomes.

# EDA with SQL: Boosters Carried Maximum Payload

Task 8

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

```
%%sql
SELECT Booster_Version, PAYLOAD_MASS__KG_

FROM SPACEXTBL

WHERE PAYLOAD_MASS__KG_ =

(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

```
 * 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 |

The query shown lists the names of the booster which have carried the maximum payload mass.

# EDA with SQL: 2015 Launch Records

## Task 9

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.

**Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.**

```
%%sql
SELECT
substr(Date, 4, 2) as Month,
Date,
[Landing _Outcome],
Booster_Version,
Launch_Site

FROM SPACEXTBL

WHERE
substr(Date, 7, 4) = '2015' AND
[Landing _Outcome] = 'Failure (drone ship)';
```

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

| Month | Date | Landing _Outcome | Booster_Version | Launch_Site |
|---|---|---|---|---|
| 01 | 10-01-2015 | Failure (drone ship) | F9 v1.1 B1012 | CCAFS LC-40 |
| 04 | 14-04-2015 | Failure (drone ship) | F9 v1.1 B1015 | CCAFS LC-40 |

The query shown lists the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

# EDA with SQL: Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

## Task 10

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

```
%%sql
SELECT
substr(Date, 4, 2) as Month,
substr(Date, 7, 4) as Year,
[Landing _Outcome],
COUNT([Landing _Outcome]) as Landing_Outcomes

FROM SPACEXTBL

WHERE
(Year BETWEEN "2010" and "2017") and
([Landing _Outcome] LIKE "Success%")

GROUP BY [Landing _Outcome]

ORDER BY Landing_Outcomes DESC;
```

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

| Month | Year | Landing _Outcome | Landing_Outcomes |
|---|---|---|---|
| 04 | 2016 | Success (drone ship) | 12 |
| 12 | 2015 | Success (ground pad) | 8 |

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.

Section 3

# Launch Sites Proximities Analysis

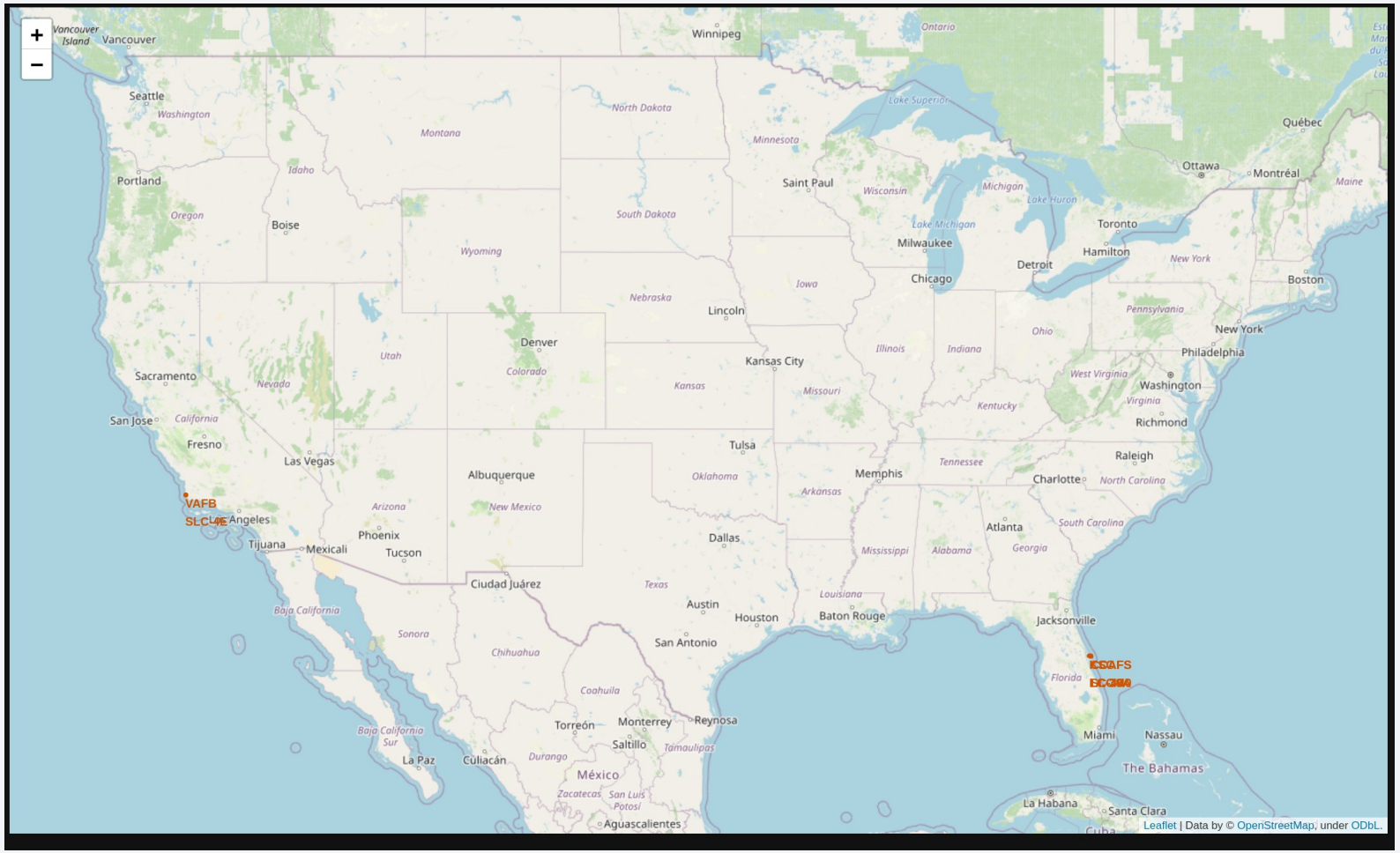
# Interactive Visual Analytics with Folium

GitHub Link: Interactive Visual Analytics with Folium

# Launch Site Proximities Analysis: SpaceX Launch Sites

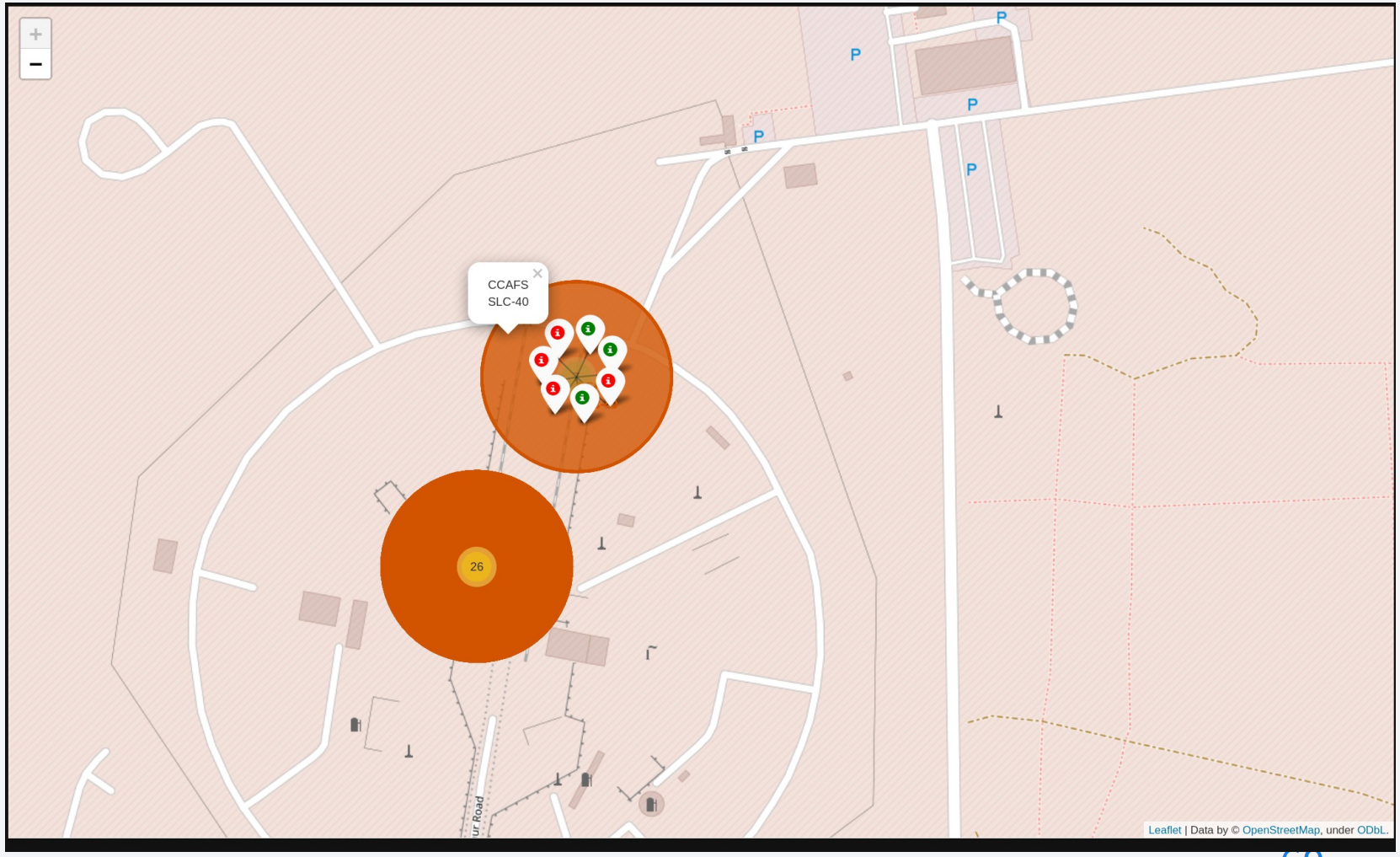
- The generated map shows the SpaceX launch sites considered in this study.

- As can be seen in the screenshots, there are launch sites on the West and East coasts of the Unite States.



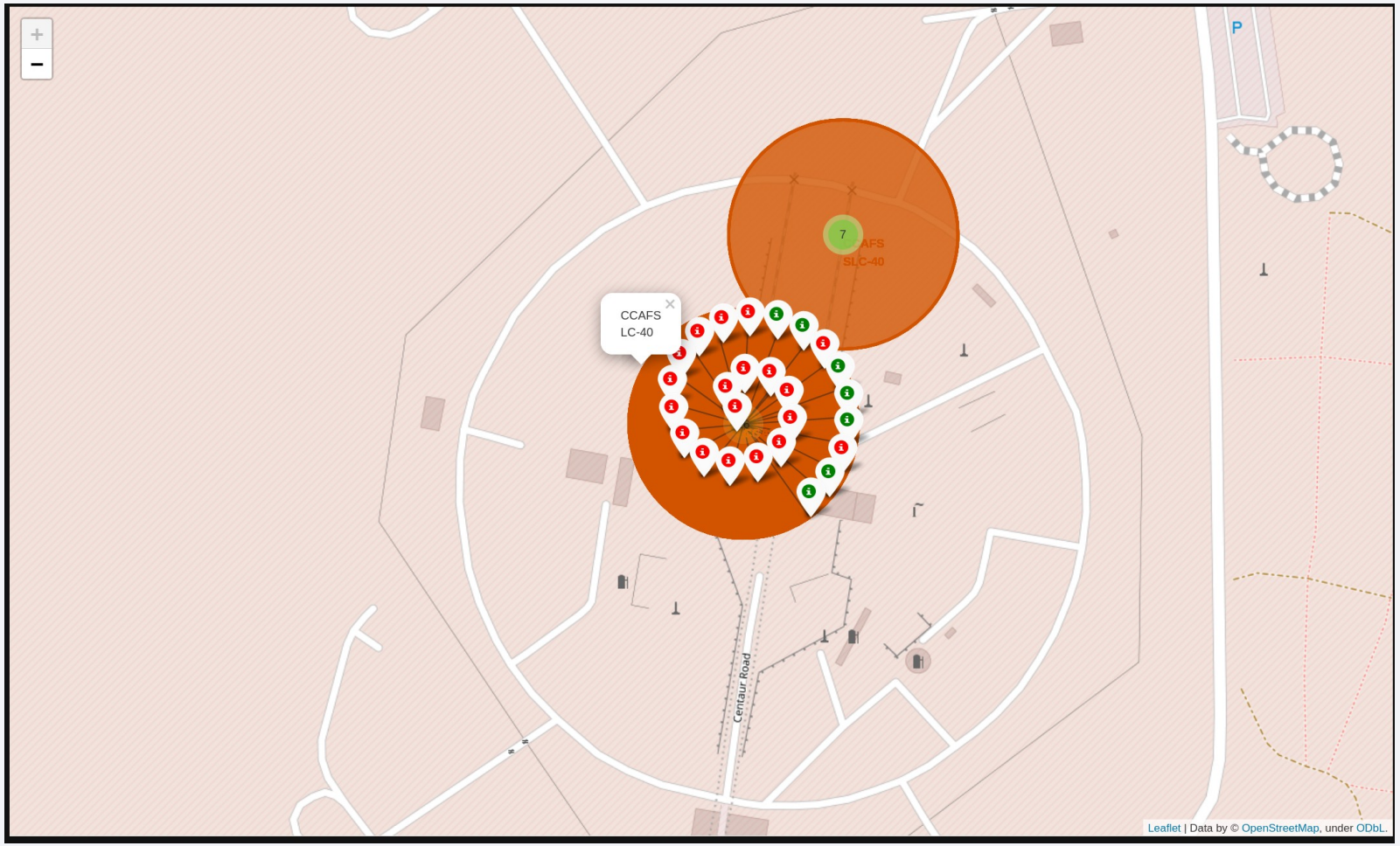
# Launch Site Proximities Analysis: Mission Success or Failure (1/2)

- The map to the right shows the mission successes (in green) and the mission failures (in red) for the launch site CCAFS SLC-40.
- There were 7 total launches at this site.
- The site adjacent to CCAFS SLC-40 is called CCAFS LC-40 and had a total of 26 launches.
- This site is shown on the next slide.



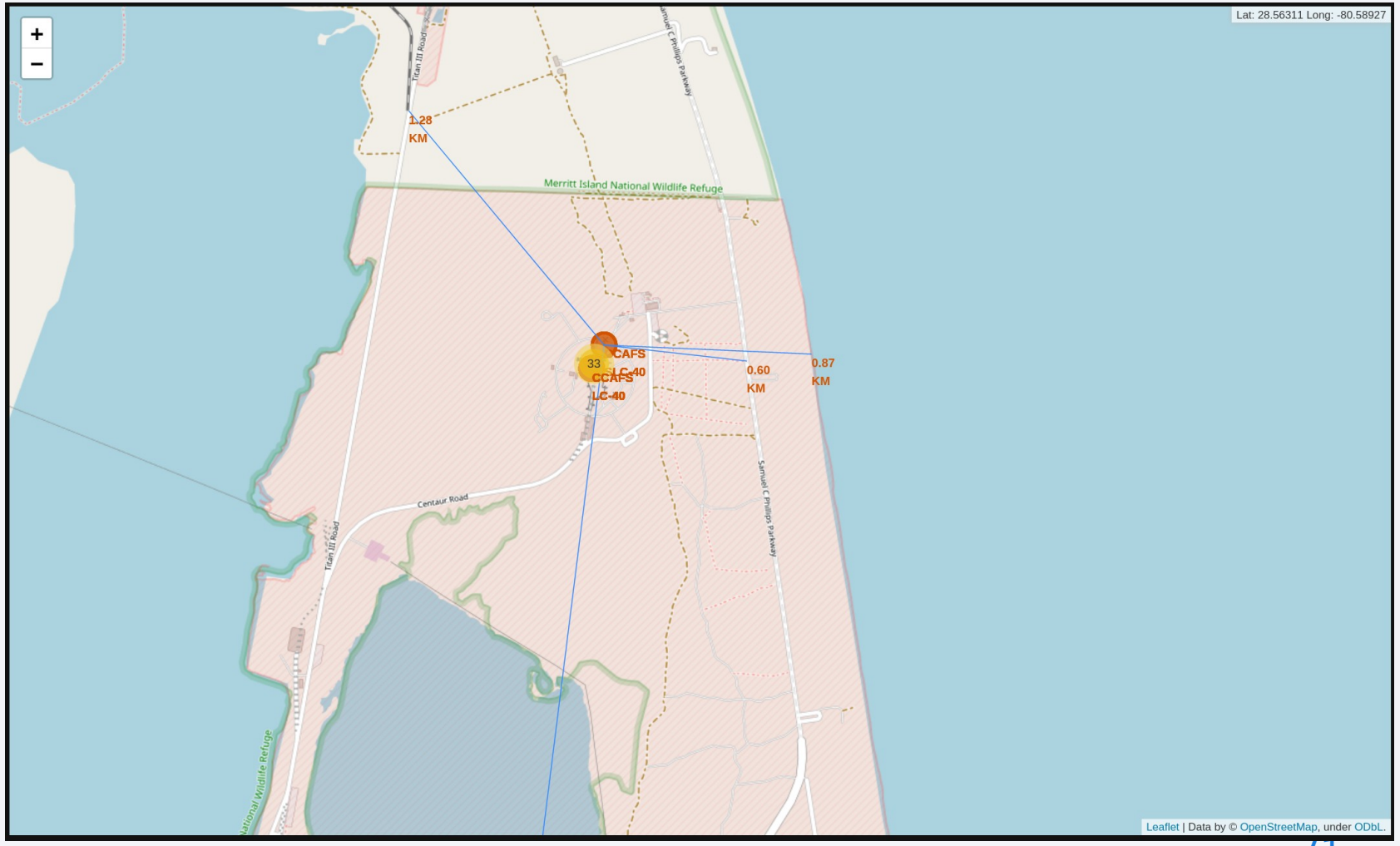
# Launch Site Proximities Analysis: Mission Success or Failure (2/2)

- The map to the right shows the mission successes (in green) and the mission failures (in red) for the launch site CCAFS LC-40.
- There were 26 total launches at this site.
- Greater than half of the launches at this site resulted in mission failure.



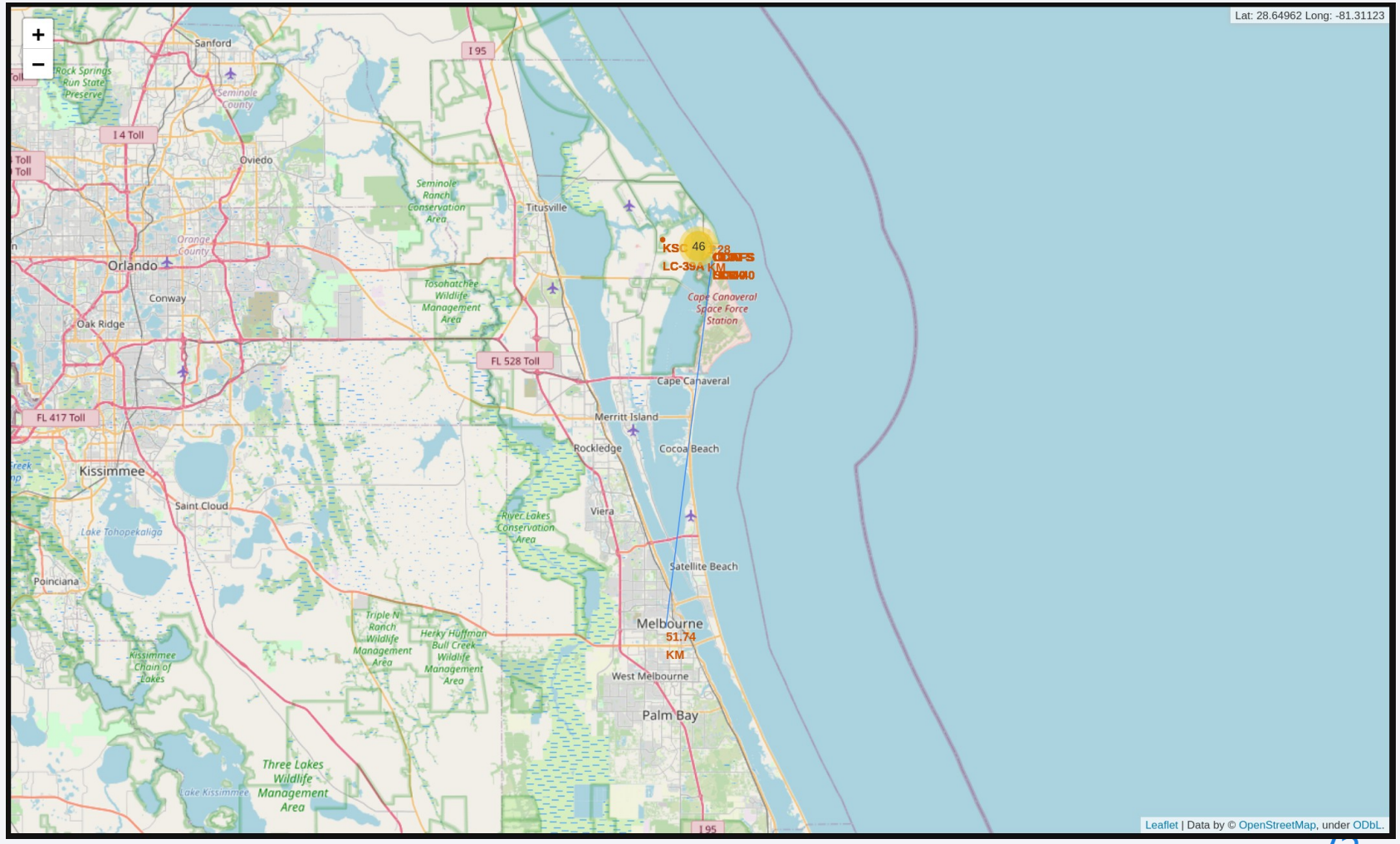
# Launch Site Proximities Analysis: Vectors to Map Features (1/2)

- As can be seen from the annotated map on the right, the launch site CCAFS SLC-40 is about:
  - 1.28km from a railroad
  - 0.6km away from a highway
  - 0.87km away from the coast



# Launch Site Proximities Analysis: Vectors to Map Features (2/2)

- As can be seen from the annotated map on the right, the launch site CCAFS SLC-40 is about:
  - 51.74km from the City of Melbourne



Section 4

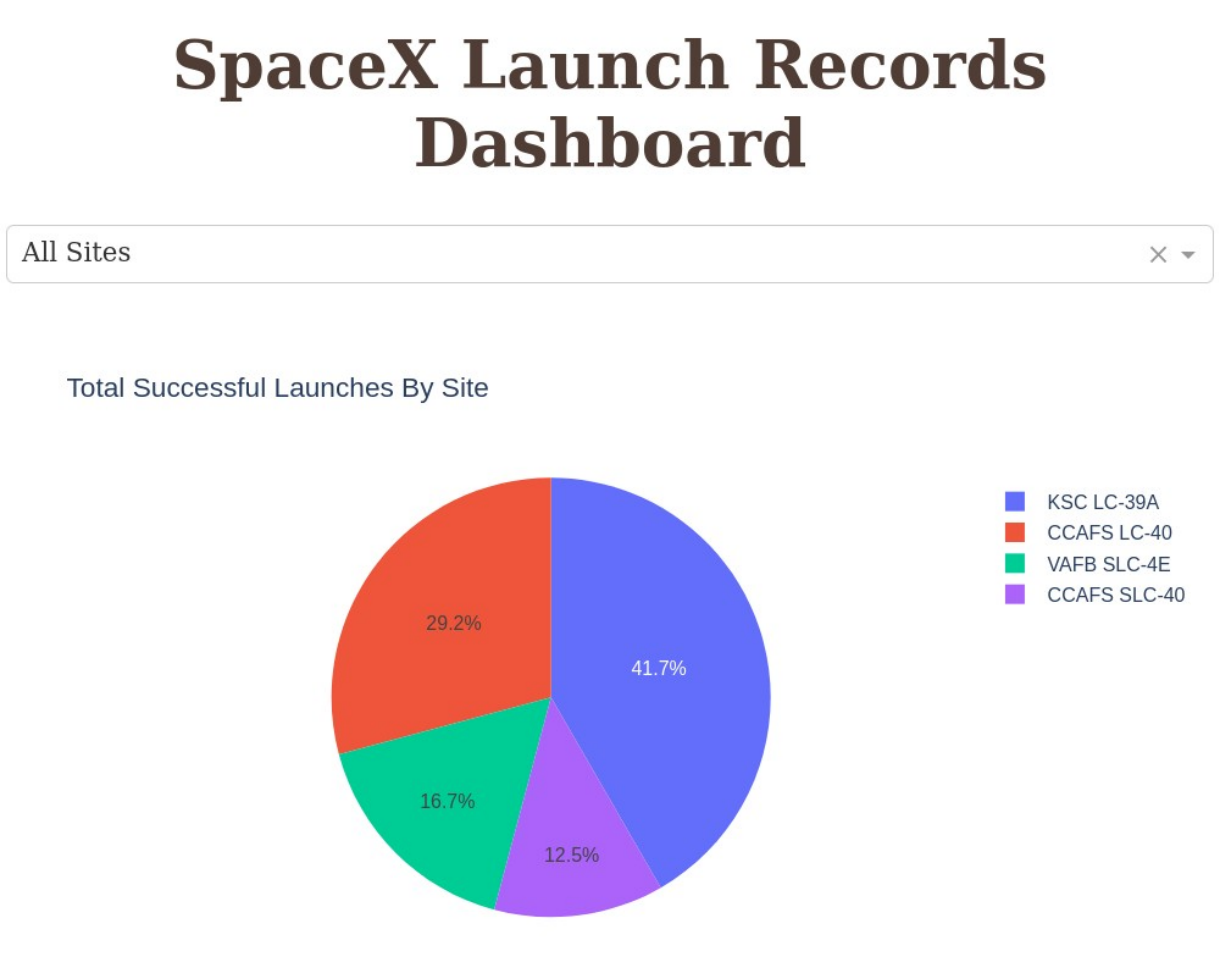
# Build a Dashboard with Plotly Dash

# Build a Dashboard with Plotly Dash

GitHub Link: Build a Dashboard Application with Plotly Dash

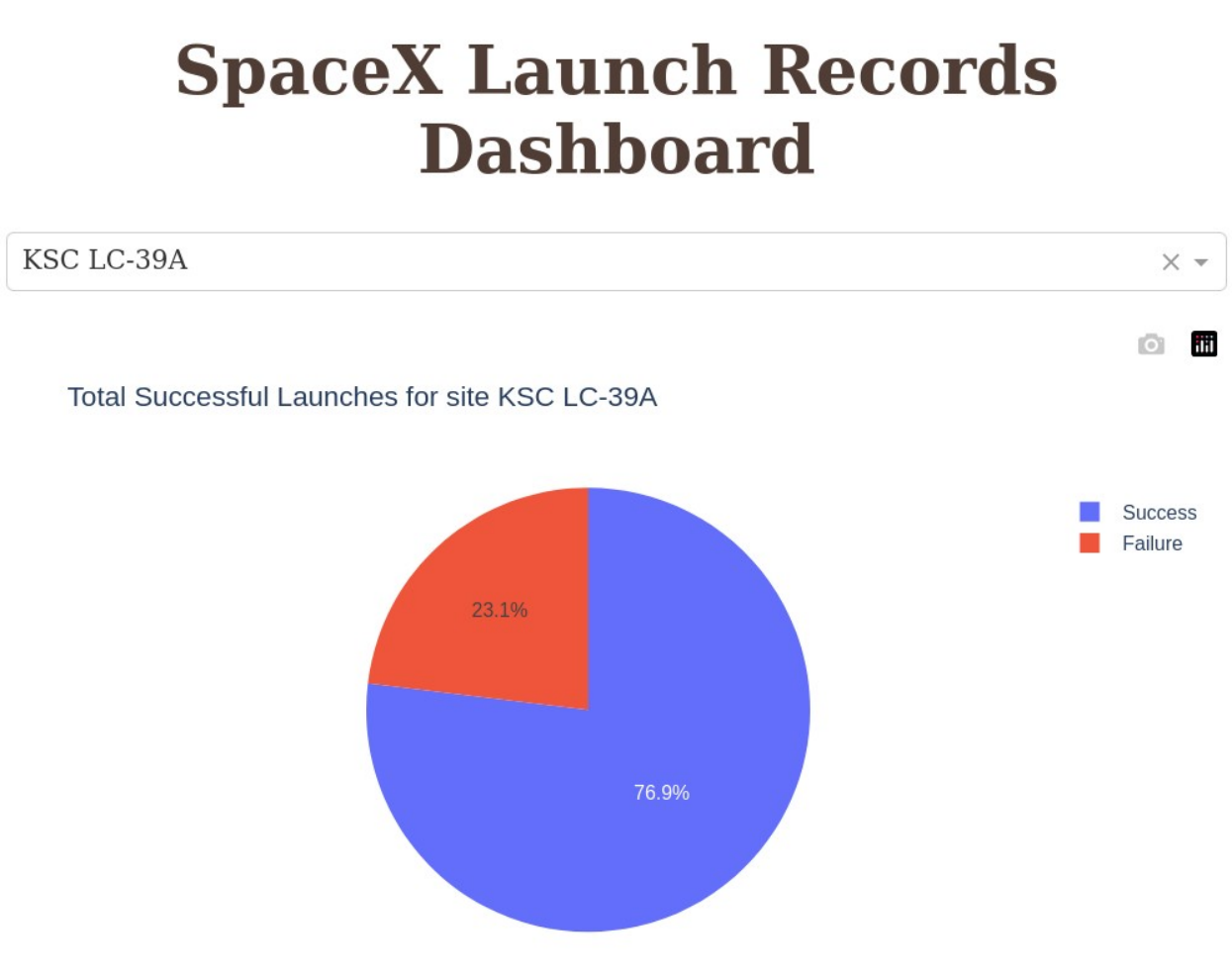
# Plotly Dash Dashboard: Total Successful Launches by Site Pie Chart

- The pie chart on the right displays , on a percentage basis, the total successful launches by site.
- As can be seen from the pie chart, a large percentage of successful launches originated from KSC LC-39A and CCAFS LC-40.



# Plotly Dash Dashboard: Total Successful Launches from KSC-LC39A

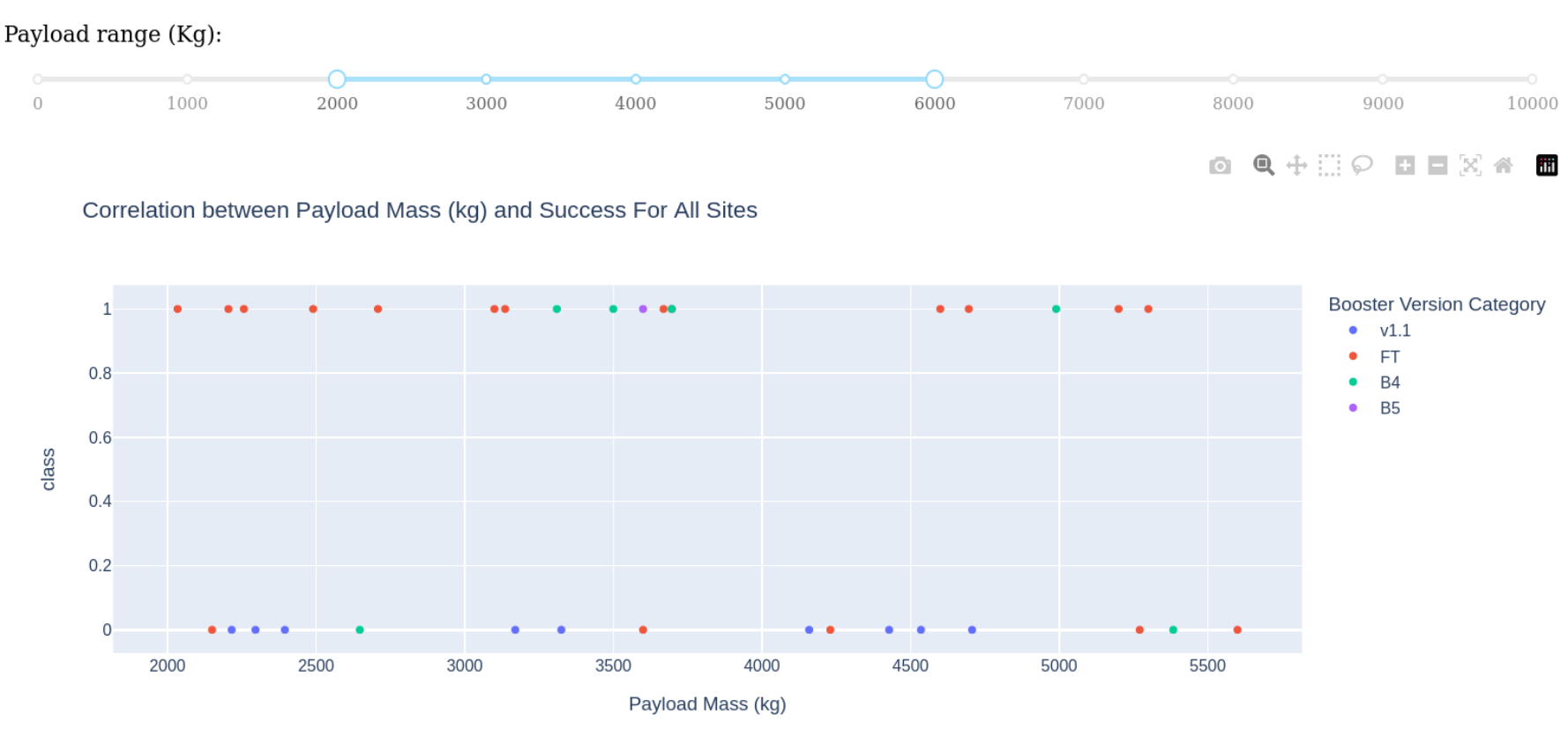
- Since KSC LC-39A had the highest percentage of successful launches, we examine it in greater detail.
- By selecting this launch site from the drop down list, we can see that over ¾ of the launches from this site were successful.



# Plotly Dash Dashboard: Payload Mass vs Launch Outcome for All Sites

- We now look for the launch outcomes based on payload mass in kg and booster version.
- From the dashboard above, we can see a clustering of successful launch outcomes for payload masses between 2000kg and 6000kg.
- In the next slide, we will use the slider to select this payload mass range.

# Plotly Dash Dashboard: Payload Mass vs Launch Outcome for All Sites




- Zooming into the payload mass range of 2000kg to 6000kg, we can see a high mission success rate, particularly for the “FT” booster version.

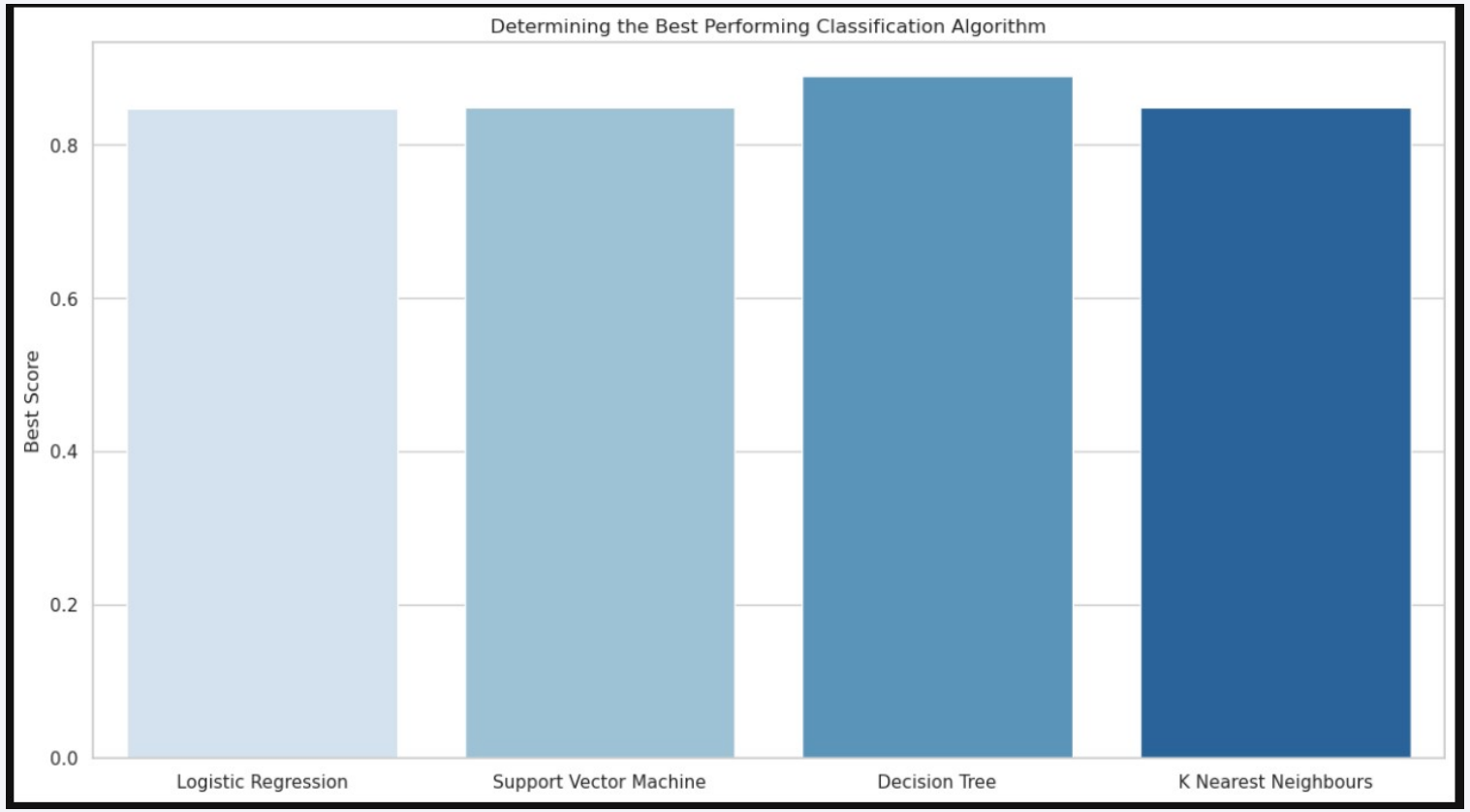
Section 5

# Predictive Analysis (Classification)

# Predictive Analysis (Classification)

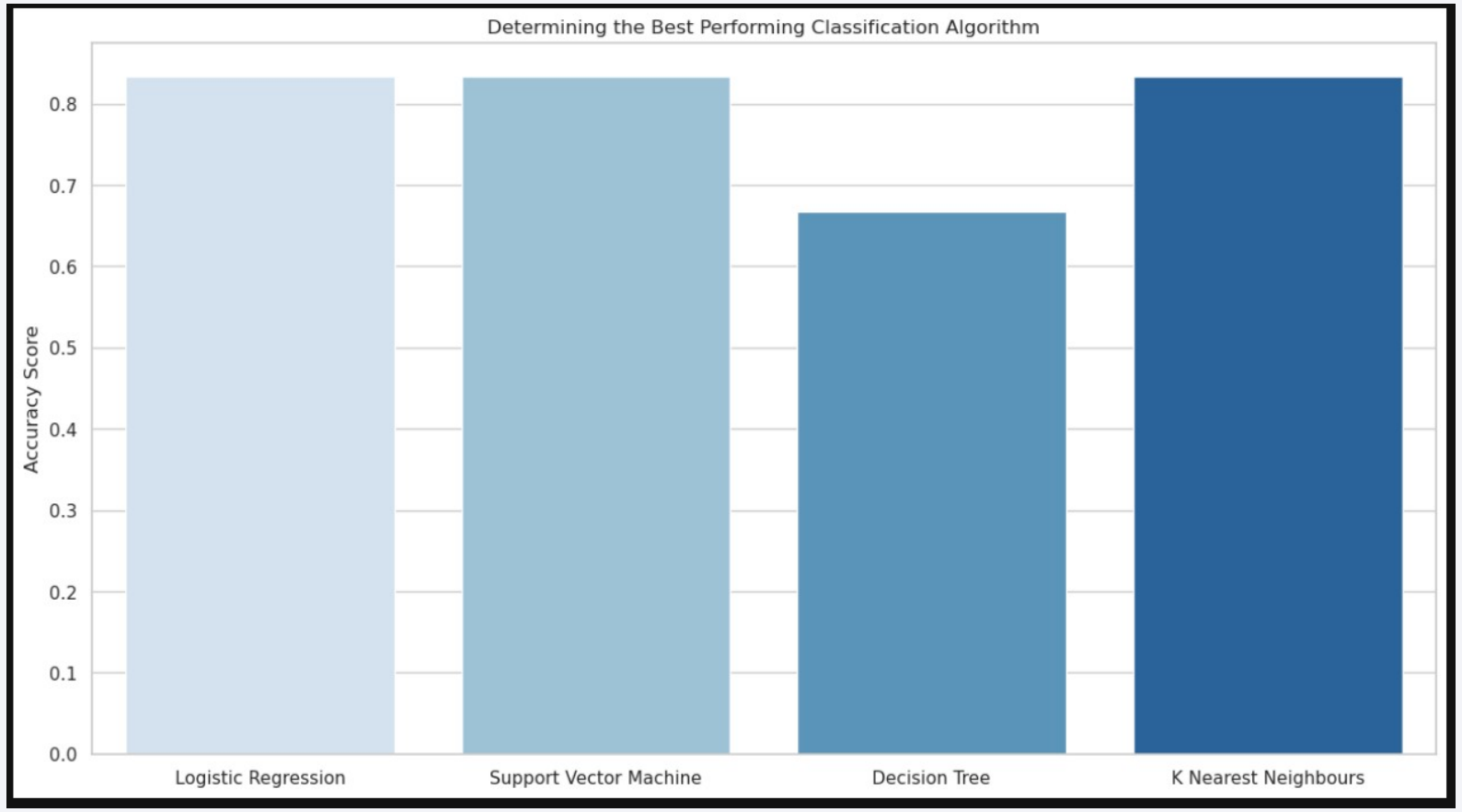
GitHub Link: Predictive Analysis Classification

# Classification Accuracy: Best Score



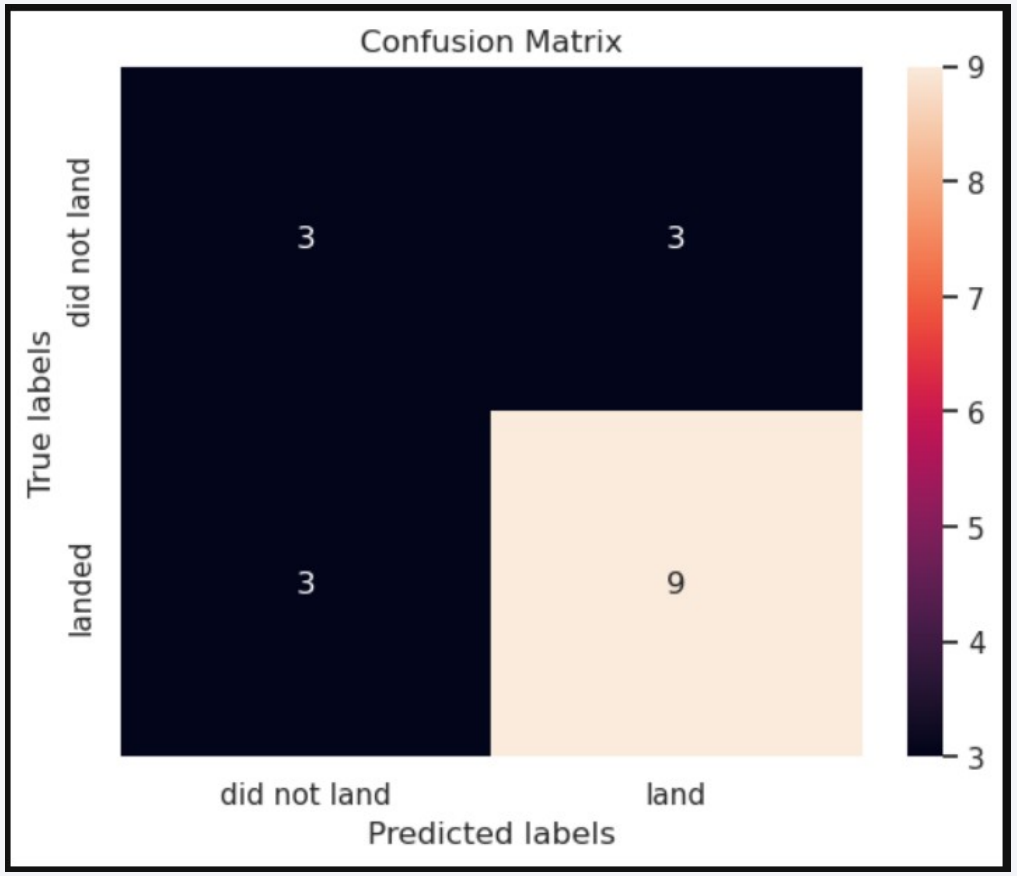

- From the bar chart above, we can determine that the Decision Tree model had the highest best score; however, this may be misleading.
- In the next chart we will see why...

# Classification Accuracy: Accuracy Score



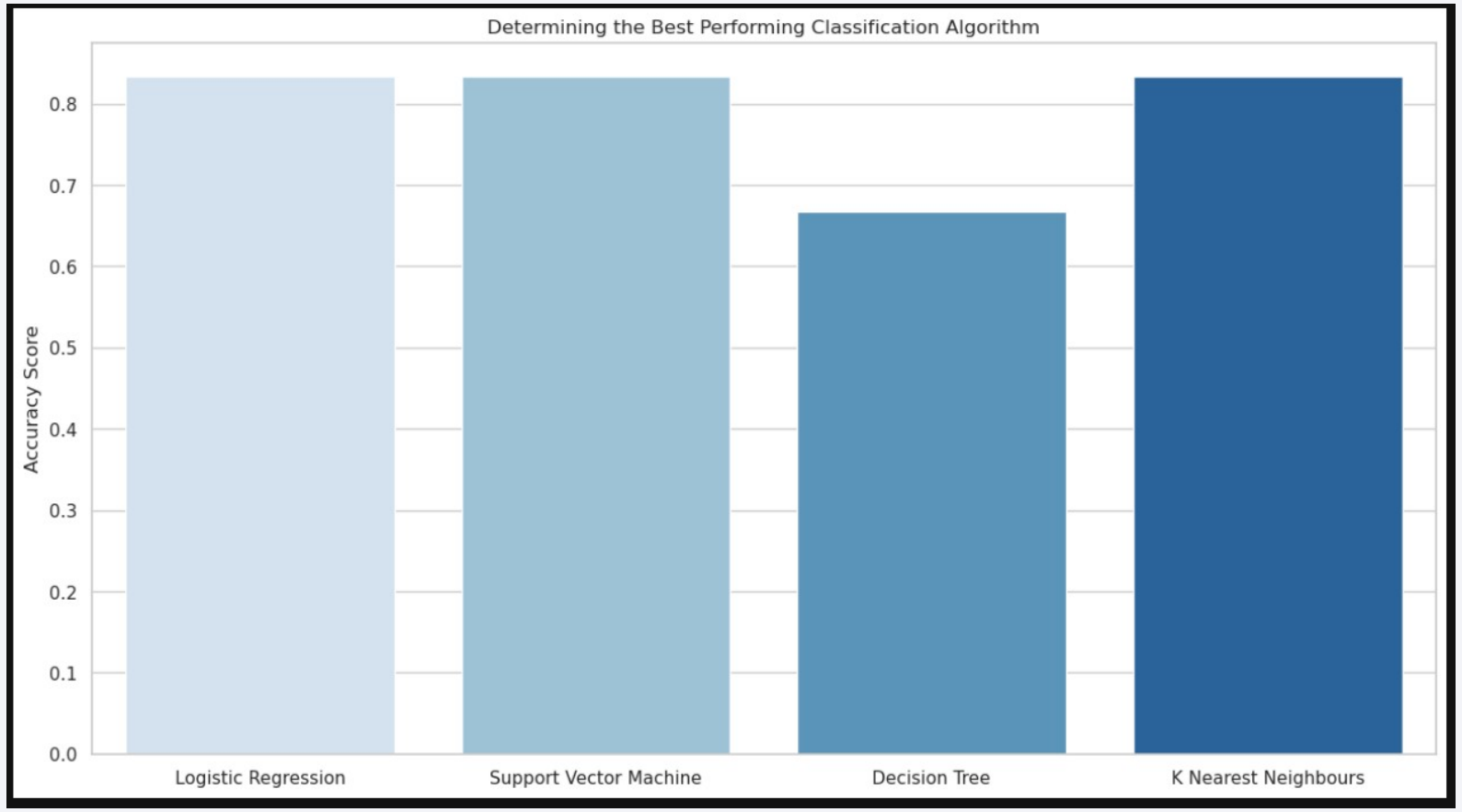

- In this chart we can see that although the Decision Tree model had the best score for a particular run, it does not have the best overall accuracy.

# Classification Accuracy: Decision Tree Confusion Matrix




- In this chart we can see the Confusion Matrix for the Decision Tree model.
- The Confusion Matrix shows that the model incorrectly classified 3 as landed which actually did not land, and vice versa.

# Classification Accuracy: Best Overall Model




- Since the Decision Tree model had a lower accuracy score, we can look at the other models for higher accuracy.

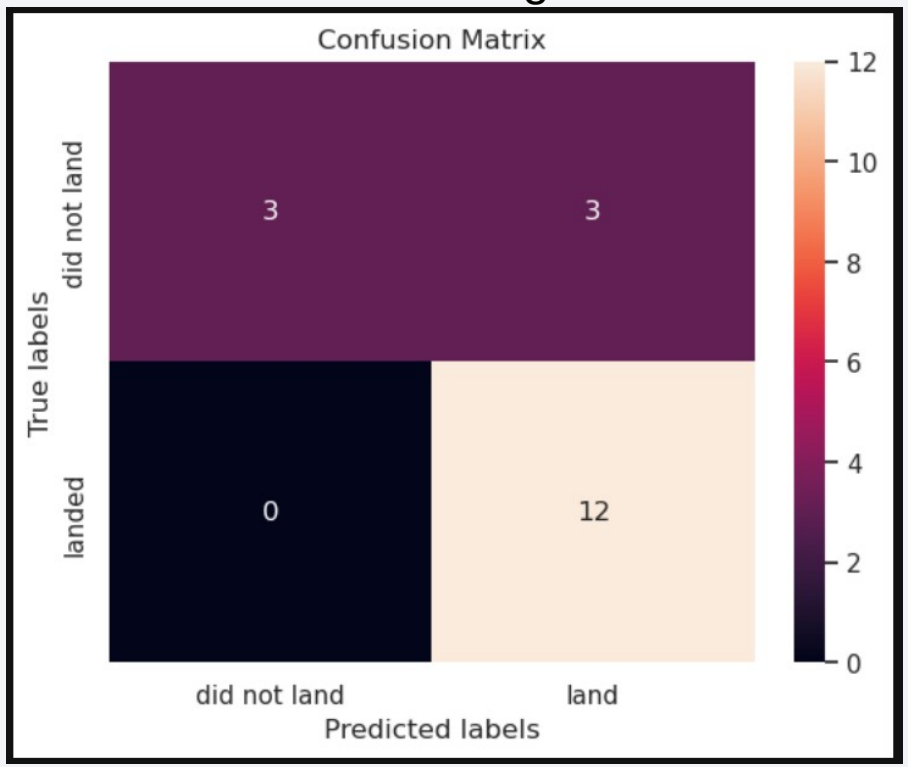
# Classification Accuracy: Table of Model Performance

| | Algorithm | Accuracy Score | Best Score |
|---|---|---|---|
| 0 | Logistic Regression | 0.833333 | 0.846429 |
| 1 | Support Vector Machine | 0.833333 | 0.848214 |
| 2 | Decision Tree | 0.666667 | 0.889286 |
| 3 | K Nearest Neighbours | 0.833333 | 0.848214 |

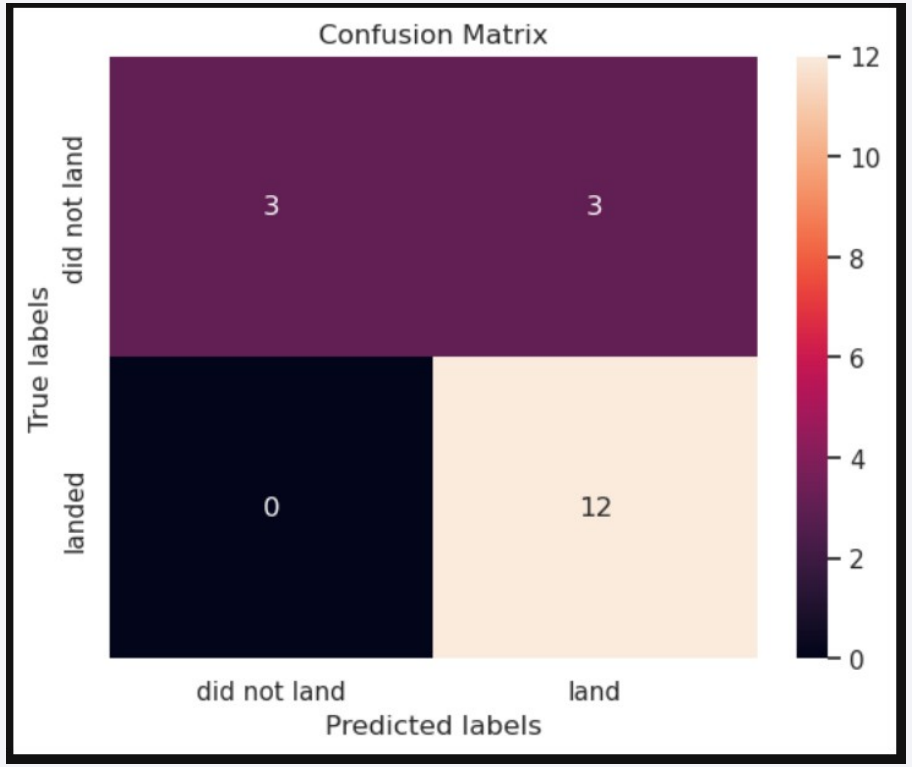
- In this chart we can see that the K Nearest Neighbors and the Support Vector Machine models had the highest and more reliable accuracy and best scores.

# Confusion Matrix

K Nearest Neighbor




Support Vector Machine




- From the above Confusion Matrices, we can see that the KNN and SVM models had similar performances.

# Conclusions

# Conclusions from Study: EDA with Pandas and Matplotlib

- Insights drawn from EDA with Pandas and Matplotlib:
  - It appears that VAFB SLC 4E and KSC LC 39A had good mission success rates, while CCAFS SLC 40 appears to be 50/50.
  - Over time there is a trend towards increasing mission success rates.
  - The plots of the data suggest that there are ranges of payload masses and specific locations where mission success is more prevalent.
  - Plots of the data also suggested that the ES-L1, GEO, HEO, and SSO orbits had the best results, while the SO orbit had the worst.

# Conclusions from Study: EDA with SQL

- Insights drawn from EDA with SQL:
  - Since the data suggests a range of payload masses where mission success is more prevalent, we used an SQL query to home in on this mass range.
  - The query showed that the boosters which have successfully landed on a drone ship had payload masses greater than 4000kg. but less than 6000kg.

## Conclusions from Study: Interactive Visual Analytics with Folium Analysis

- Insights drawn from the Interactive Visual Analytics with Folium Analysis:
  - The study looked at two launch sites:
    - CCAFS LC-40 and
      - There were 7 total launches at this site.
      - 3/7 of the launches at this site resulted in mission success
    - CCAFS SLC-40
      - There were 26 total launches at this site.
      - Less than half resulted in mission success.
  - An analysis of the proximity of these launch sites to other features on the map such as highways, the coast, railroads and nearest city showed that the launch sites were close to a highway, a railroad, the coast, but were far away from the city of Melbourne.

# Conclusions from Study: Dashboard with Plotly Dash Analysis

- Insights drawn from Dashboard with Plotly Dash Analysis:
  - A pie chart plot indicates that a large percentage of successful launches originated from KSC LC-39A and CCAFS LC-40.
  - KSC LC-39A had the highest percentage of successful launches, where over ¾ of the launches from this site were successful.
  - With a payload mass range of 2000kg to 6000kg, we can see a high mission success rate, particularly for the “FT” booster version.

# Conclusions from Study: Predictive Analysis (Classification)

- Insights drawn from Predictive Analysis (Classification):
  - The Decision Tree model had the best predictive score but lowest overall accuracy.
  - The K Nearest Neighbors and the Support Vector Machine models had the highest and more reliable accuracy and best scores.

# Thank you!