IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

ChunFu Hsieh

08/29/2023

Outline

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

Executive Summary

Summary of methodologies

- Data Collecting, API and Web Scripting
- Data Wrangling / Clean Data
- Analysis, SQL
- Visualization, Panda and matplotlib
- Predictions, Classification Regression

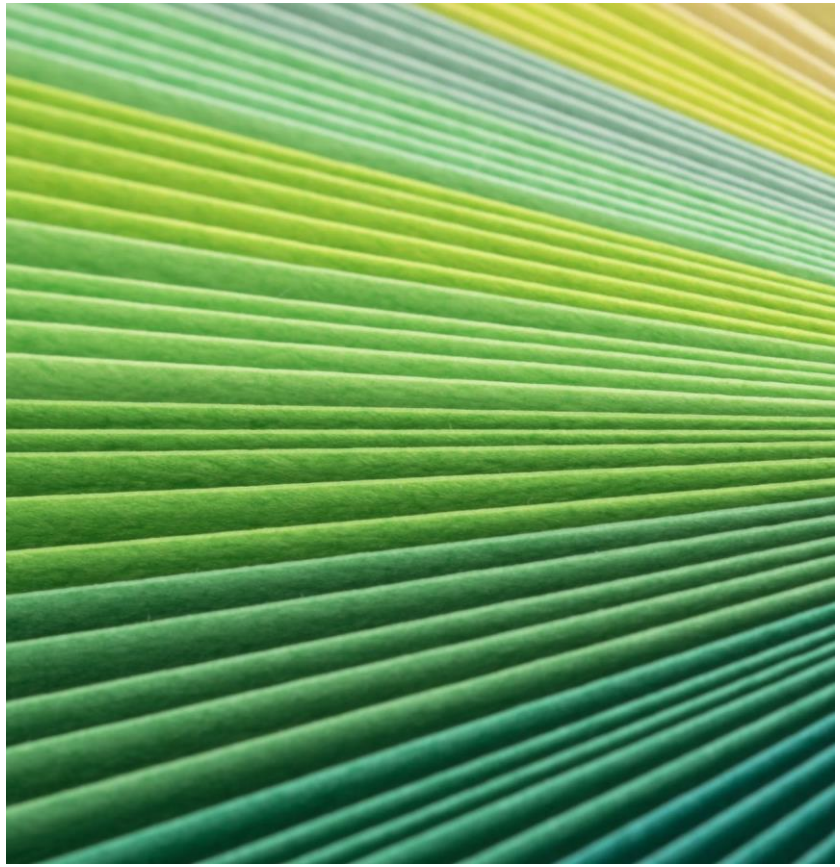
Summary of all results

- Data Wrangling the need Data
- Analysis the result
- Visualization easy to see the result
- Predictions the future result

Introduction

Predict if the Falcon 9 first stage will land successfully.

Determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.



Section 1

Methodology

Methodology

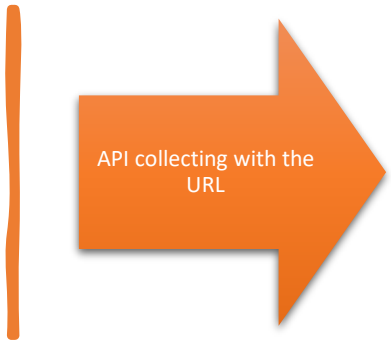
- Executive Summary
- Data collection methodology:
  - Using the API and Web Scarping as the source of Data collection
    - API Import the libraries and define the Functions
    - Web Scarping on the Wikipedia pages, getting the data of first landing stage
- Perform data wrangling
  - Remove all the Null information, transpose the data to certain columns
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Using panda tools and matplotlib visualization, SQL for the fata analysis to find get up to the right data
- Perform interactive visual analytics using Folium and Plotly Dash
  - Interactive Dashboard with Ploty Dash
- Perform predictive analysis using classification models
  - Find the method performs best using test data

Data Collection

- API
  - **Request and parse the SpaceX launch data using the GET request**
  - Filter the Dataframe to only include Falcon 9 launches¶
  - **Dealing with Missing Values**

- Web Scraping
  - **Request the Falcon9 Launch Wiki page from its URL**
  - **Extract all column/variable names from the HTML table header**
  - **Create a data frame by parsing the launch HTML tables**

Data Collection – SpaceX API



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

[8]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

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

[9]:
```
response.status_code
```

[9]:
```
200
```

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

[10]:
```
response = requests.get(static_json_url)
```

[11]:
```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

Using the dataframe `data` print the first 5 rows

[12]:
```
# Get the head of the dataframe
data.head()
```

Data Collection - Scraping

- Present your web scraping process using key phrases and flowcharts

- Add the GitHub URL of the completed web scraping notebook, as an external reference and peer-review purpose

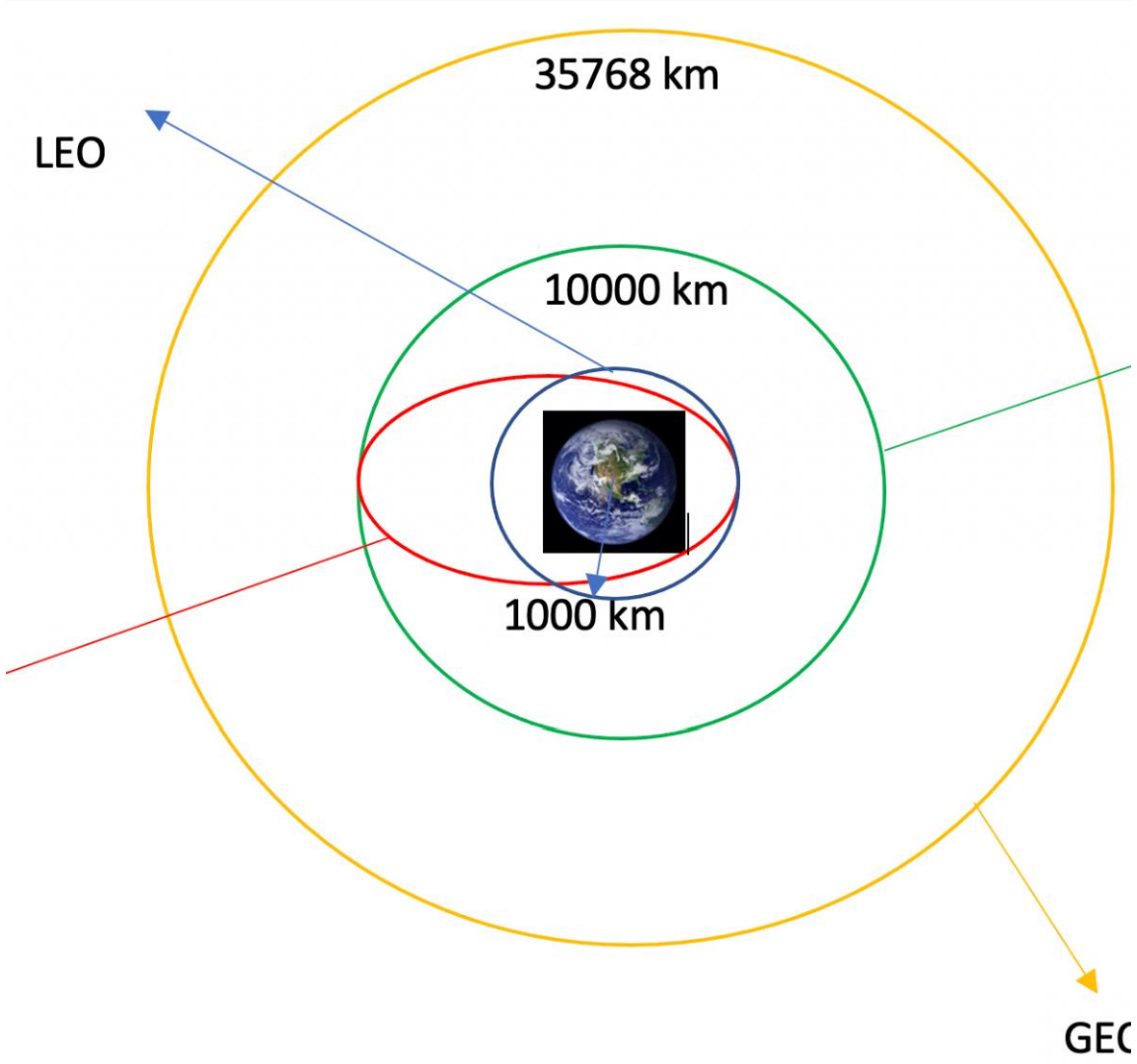
```
def date_time(table_cells):
    """
    This function returns the data and time from the HTML  table cell
    Input: the  element of a table data cell extracts extra row
    """
    return [data_time.strip() for data_time in list(table_cells.strings)][0:2]

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()

    column_name = ' '.join(row.contents)

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

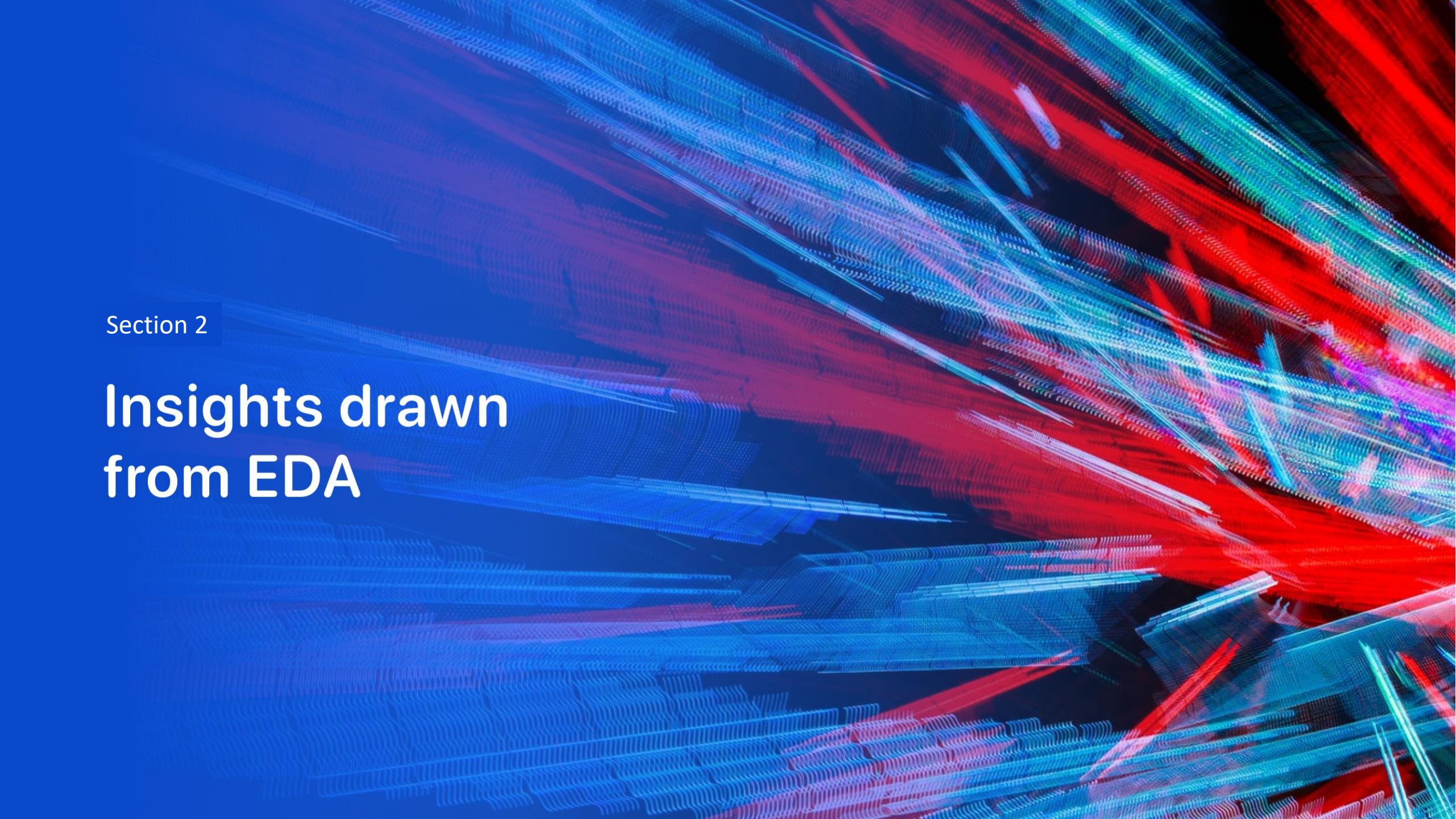
To keep the lab tasks consistent, you will be asked to scrape the data from a snapshot of the `List of Falcon 9 and Falcon Heavy launches` Wikipage updated on `9th June 2021`

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

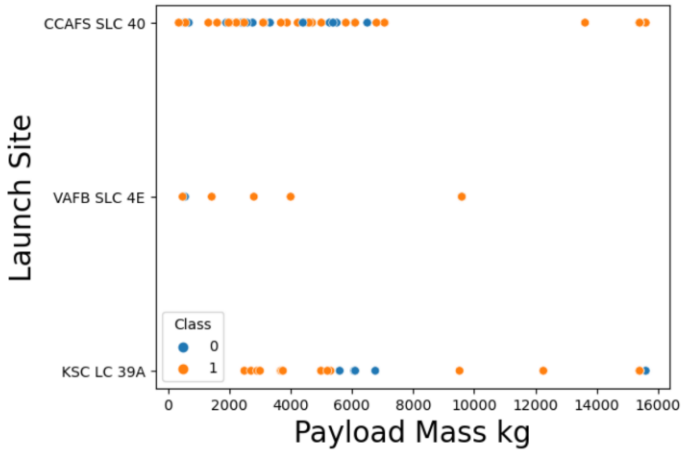
Data Wrangling

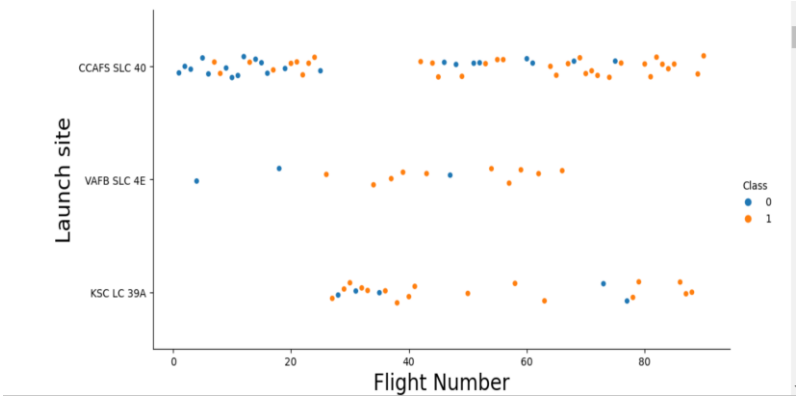
- Processed Stage
  - Import the csv file creating data frame
  - Cleaning the Null
  - Calculate the number and occurrence of each orbit
  - Calculate the number and occurrence of mission outcome per orbit type
  - Create a landing outcome label from Outcome column¶



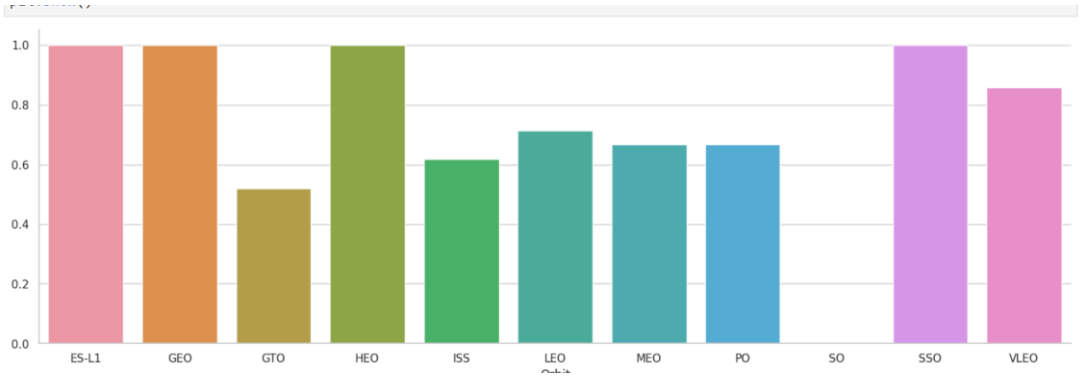

Section 2
Insights drawn from EDA

EDA with Data Visualization





- Plot out the FlightNumber vs. PayloadMass
  - Finding the result that might effect the first stage
- Plot out the relationship between Payload and Launch Site
  - Launch Site also matter on the first stage landing
- Bar out the Orbit success rate



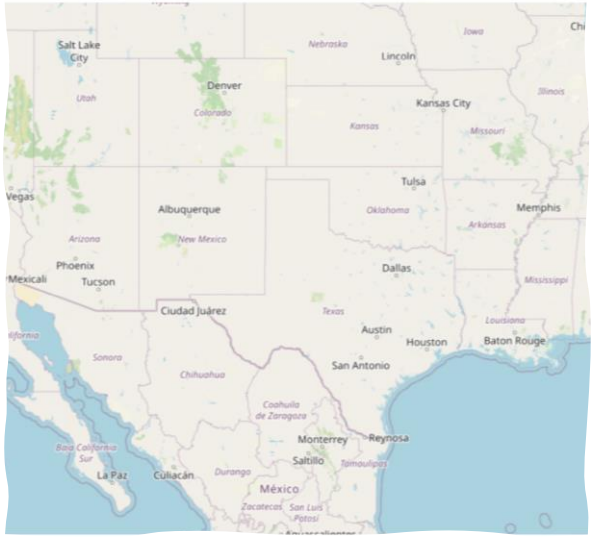
EDA with SQL

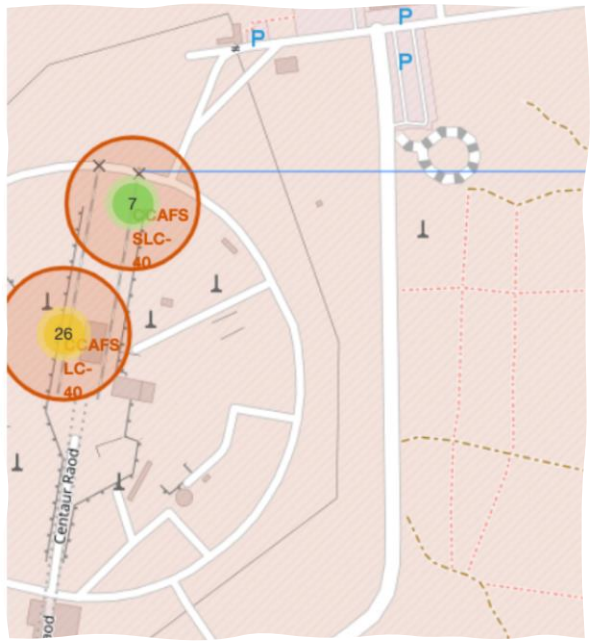
1. Display the names of the unique launch sites in the space mission
2. Display the total payload mass carried by boosters launched by NASA (CRS)
3. Display average payload mass carried by booster
4. List the date when the first succesful landing outcome in ground pad was acheived.
5. List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
6. List the total number of successful and failure mission outcomes

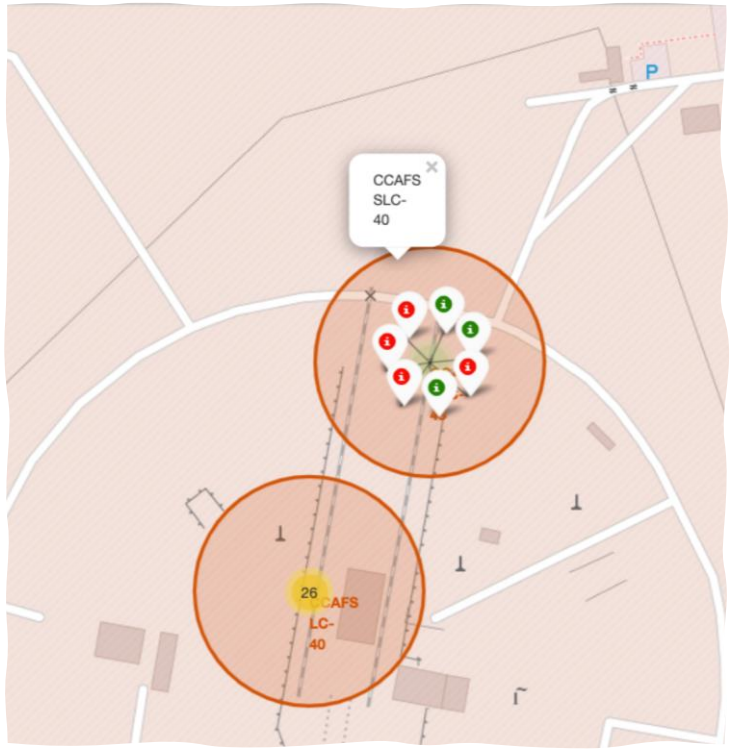

Section 3
Launch Sites
Proximities Analysis

Build an Interactive Map with Folium

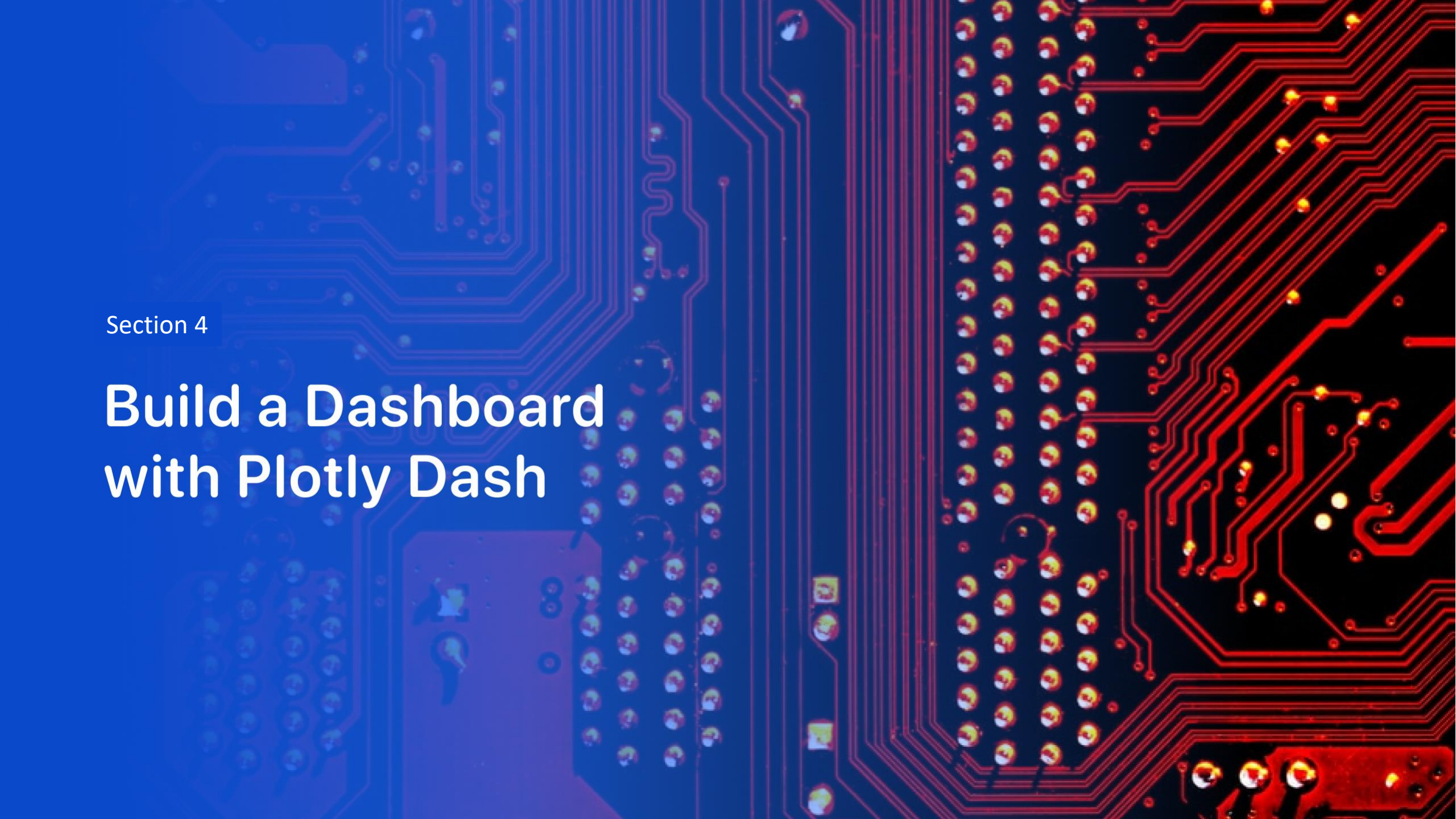
- Mark all launch sites on a map
- Mark the success/failed launches for each site on the map
- Calculate the distances between a launch site to its proximities



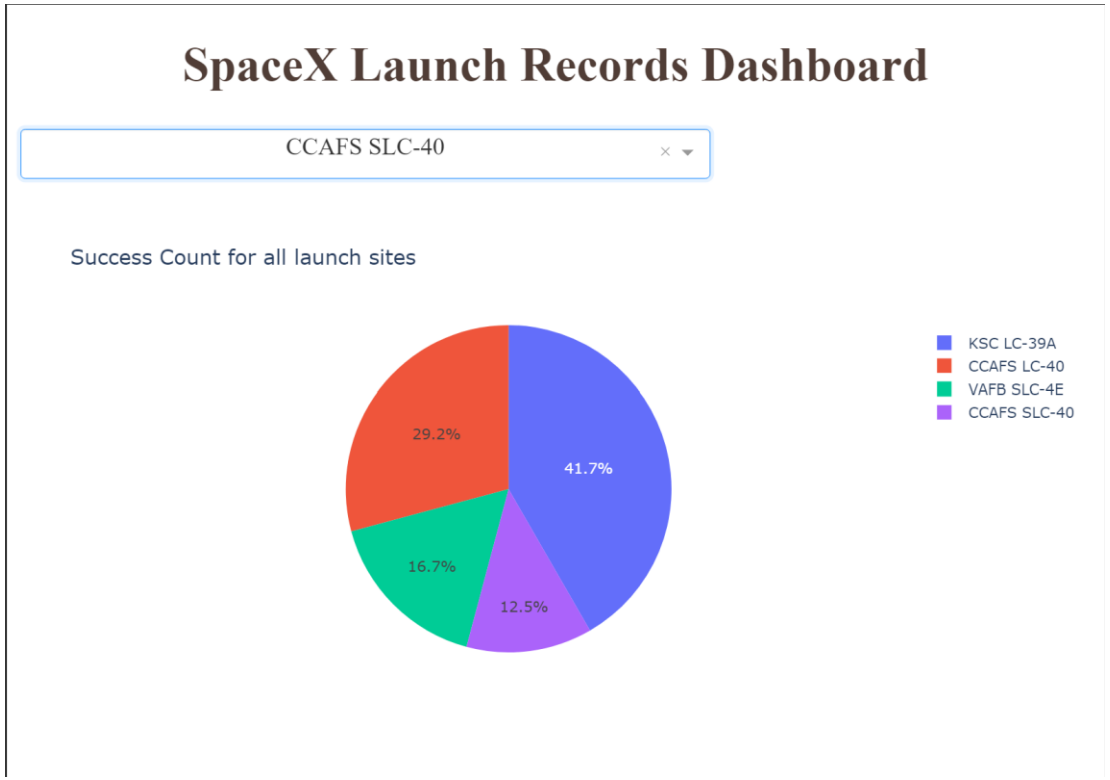




46	KSC LC-39A	28.573255	-80.646895	1	green
47	KSC LC-39A	28.573255	-80.646895	1	green
48	KSC LC-39A	28.573255	-80.646895	1	green
49	CCAFS SLC-40	28.563197	-80.576820	1	green
50	CCAFS SLC-40	28.563197	-80.576820	1	green
51	CCAFS SLC-40	28.563197	-80.576820	0	red
52	CCAFS SLC-40	28.563197	-80.576820	0	red
53	CCAFS SLC-40	28.563197	-80.576820	0	red
54	CCAFS SLC-40	28.563197	-80.576820	1	green


Section 4
Build a Dashboard with Plotly Dash

Build a Dashboard with Plotly Dash



Section 5

Predictive Analysis (Classification)

Predictive Analysis (Classification)

- Transforming the data
- Build different model with hyperparameters using GridSearchCV.
- Metric to prossces on the number
- Which perform the best model

Results

We can conclude that:

- Larger site create greater success rate
- Launch success rate started to increase in 2013 till 2020.
- Orbits ES-L1, GEO, HEO, SSO, VLEO is more success then other
- KSC LC-39A is the most successful on launch
- Doing decision tree is better than other model use

Thank you!