



SPACE Y FIRST STAGE REUSE

Venugopal Thottempudi

06/25/2023

CONTENTS

- **Executive Summary**
- **Introduction**
- **Methodology**
- **Results**
 - **Visualization – Charts**
 - **Dashboard**
- **Discussion**
 - **Findings & Implications**
- **Conclusion**
- **Appendix**

EXECUTIVE SUMMARY

Summary of Methodologies

The Capstone project follows these steps:

1. Data Collection
2. Data Wrangling
3. Exploratory Data Analysis
4. Interactive Visual Analytics
5. Predictive Analysis

- Summary of Results:

- This project produced the following outputs and visualizations:
 1. Exploratory Data Analysis (EDA) results
 2. Geospatial analytics
 3. Interactive dashboard
 4. Predictive analysis of classification models

INTRODUCTION

- SpaceX launches Falcon 9 rockets at a cost of around
- \$62m. This is considerably cheaper than other providers (which usually cost upwards of \$165m), and 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 and SpaceX for a rocket launch.
- This project will ultimately predict if the Space X Falcon 9 first stage will land successfully.



METHODOLOGY

1. Data Collection

- I used GET requests to the SpaceX REST API
- Web Scraping

2. Data Wrangling

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

3. Exploratory Data Analysis

- Using SQL queries to manipulate and evaluate the SpaceX dataset
- Using Pandas and Matplotlib to visualize

4. Interactive Visual Analytics

- Geospatial analytics using Folium
- Creating an interactive dashboard using Plotly Dash

5. Data Modelling and Evaluation

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

DATA COLLECTION SPACE X REST API

Here I have used the SpaceX API to retrieve data about launches, including information about the rocket used, payload etc.

1

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

2

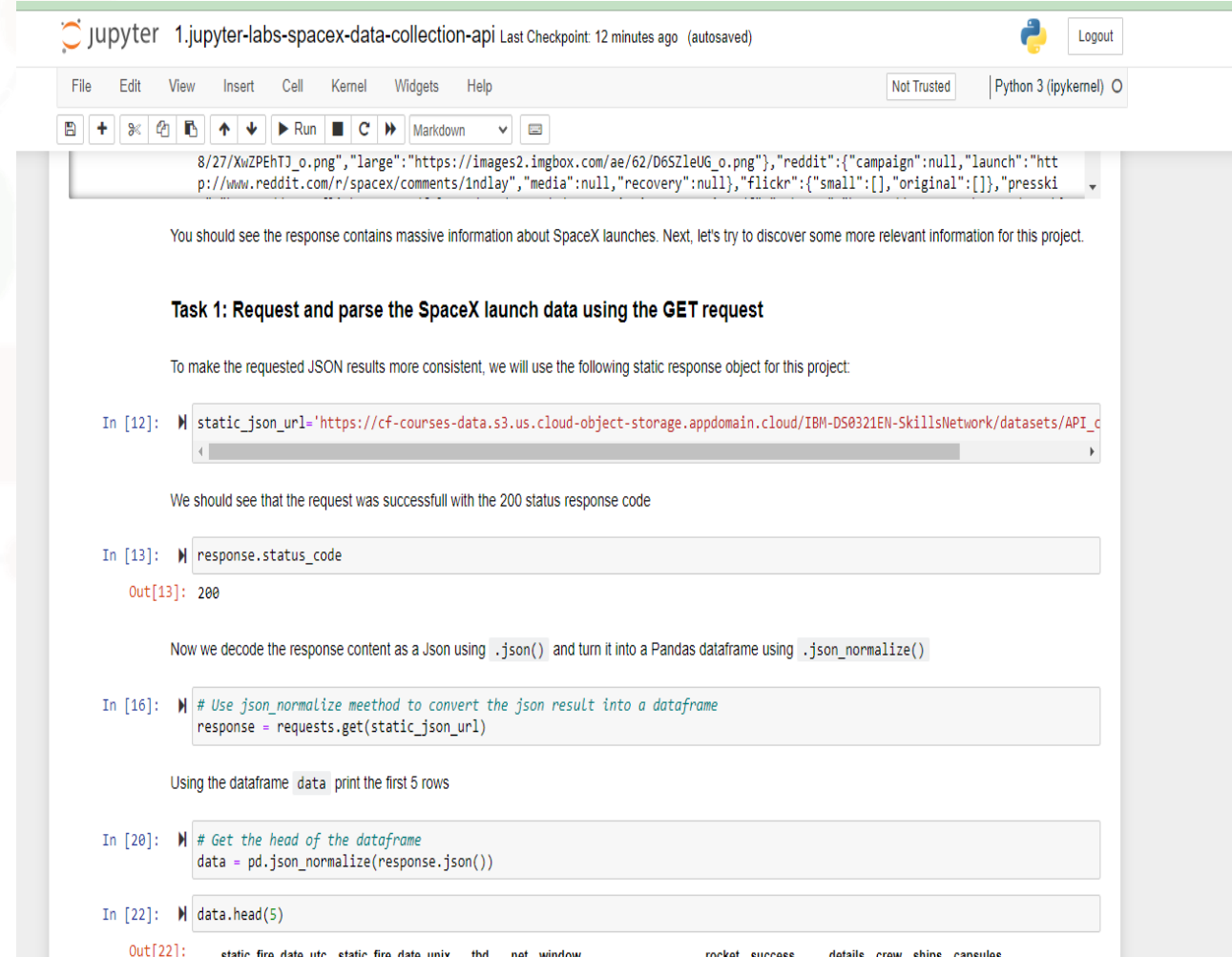
- Use custom logic to clean the data
- Define lists for data to be stored in
- Call custom functions to retrieve data and fill the lists
- Use these lists as values in a dictionary and construct the dataset

3

- Create a Pandas DataFrame from the constructed dictionary dataset

4

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



The screenshot shows a Jupyter Notebook titled "1.jupyter-labs-spacex-data-collection-api". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a toolbar with icons for file operations and execution, and a status bar indicating "Not Trusted" and "Python 3 (ipykernel)".

The notebook content includes a text block with a URL snippet, a paragraph explaining the task, and several code cells:

```
In [12]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_c
8/27/XwZPEHTJ_o.png', "large": "https://images2.imgbox.com/ae/62/D6S21eUG_o.png"}, "reddit": {"campaign": null, "launch": "http://www.reddit.com/r/spacex/comments/1ndlay", "media": null, "recovery": null}, "flickr": {"small": [], "original": []}, "presski
```

You should see the response contains massive information about SpaceX launches. Next, let's try to discover some more relevant information for this project.

Task 1: Request and parse the SpaceX launch data using the GET request

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

```
In [12]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_c
8/27/XwZPEHTJ_o.png', "large": "https://images2.imgbox.com/ae/62/D6S21eUG_o.png"}, "reddit": {"campaign": null, "launch": "http://www.reddit.com/r/spacex/comments/1ndlay", "media": null, "recovery": null}, "flickr": {"small": [], "original": []}, "presski
```

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

```
In [13]: response.status_code
Out[13]: 200
```

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

```
In [16]: # Use json_normalize method to convert the json result into a dataframe
response = requests.get(static_json_url)

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

Using the dataframe `data` print the first 5 rows

```
In [20]: # Get the head of the dataframe
data = pd.json_normalize(response.json())

In [22]: data.head(5)
Out[22]: static fire data utr static fire data unix thd not window rocket success details crew shine capsules
```


DATA COLLECTION SPACE X REST API

1

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

2

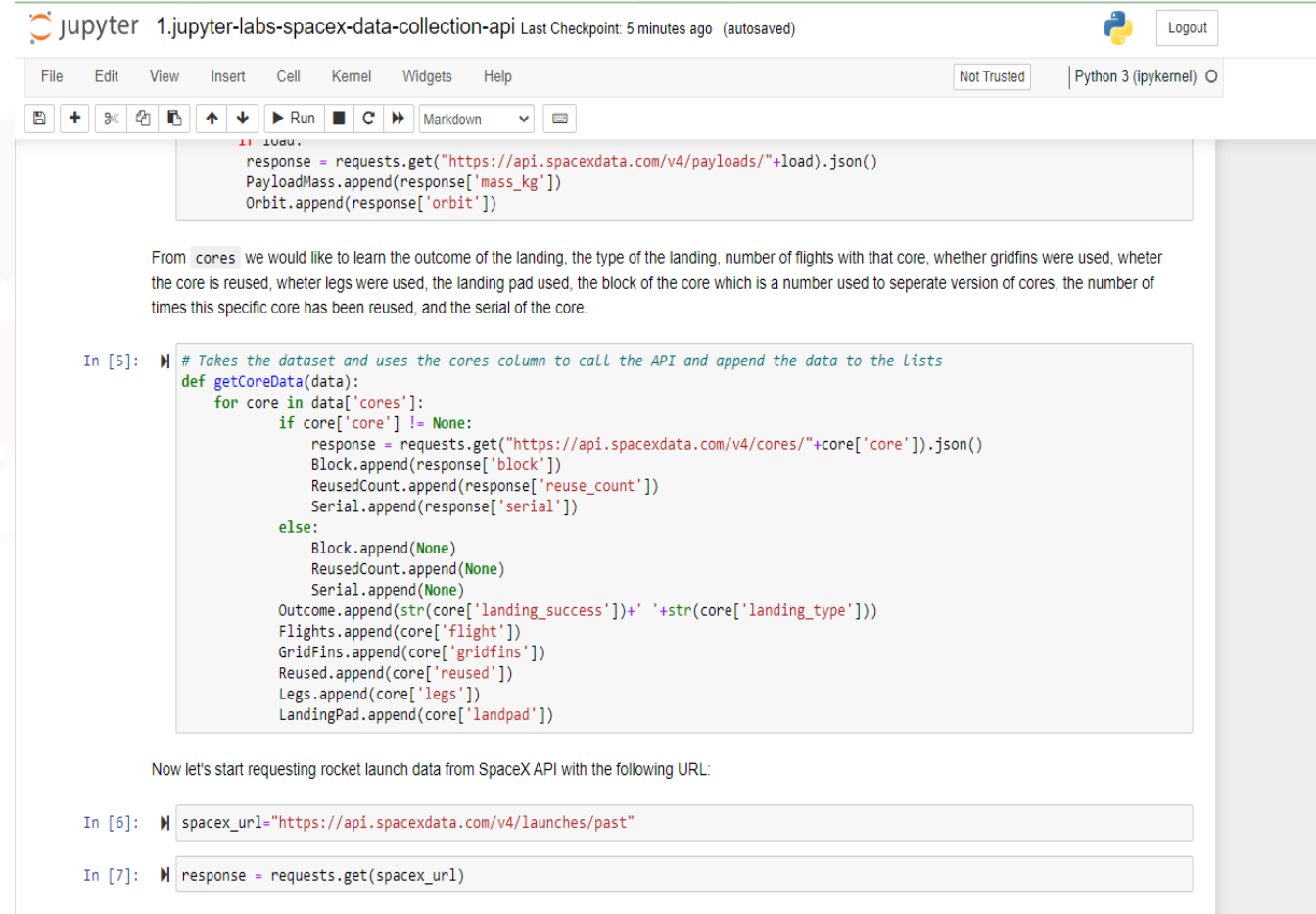
- Use custom logic to clean the data (see Appendix)
- Define lists for data to be stored in
- Call custom functions (see Appendix) to retrieve data and fill the lists
- Use these lists as values in a dictionary and construct the dataset

3

- Create a Pandas DataFrame from the constructed dictionary dataset

4

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



The screenshot shows a Jupyter Notebook titled "1.jupyter-labs-spacex-data-collection-api". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a toolbar with icons for file operations and execution, and a status bar indicating "Not Trusted" and "Python 3 (ipykernel)".

The notebook contains the following code and text:

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

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

```
In [5]: # Takes the dataset and uses the cores column to call the API and append the data to the lists
def getCoreData(data):
    for core in data['cores']:
        if core['core'] != None:
            response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
            Block.append(response['block'])
            ReusedCount.append(response['reuse_count'])
            Serial.append(response['serial'])
        else:
            Block.append(None)
            ReusedCount.append(None)
            Serial.append(None)
    Outcome.append(str(core['landing_success'])+' '+str(core['landing_type']))
    Flights.append(core['flight'])
    GridFins.append(core['gridfins'])
    Reused.append(core['reused'])
    Legs.append(core['legs'])
    LandingPad.append(core['landpad'])
```

Now let's start requesting rocket launch data from SpaceX API with the following URL:

```
In [6]: spacex_url="https://api.spacexdata.com/v4/launches/past"

In [7]: response = requests.get(spacex_url)
```

DATA COLLECTION – WEB SCRAPING

Web scraping to collect Falcon 9 historical launch records from a Wikipedia page titled List of Falcon 9 and Falcon Heavy launches.

- Request the HTML page from the static URL
- Assign the response to an object
- Create a BeautifulSoup object from the HTML response object
- Find all tables within the HTML page
- Collect all column header names from the tables found within the HTML page
- Use the column names as keys in a dictionary
- Use custom functions and logic to parse all launch tables to fill the dictionary values
- Convert the dictionary to a Pandas DataFrame ready for export

jupyter 2. jupyter-labs-webscraping Last Checkpoint: a few seconds ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

More specifically, the launch records are stored in a HTML table shown below:

2020 [edit]

In late 2019, [Gwynne Showell](#) stated that SpaceX hoped for as many as 24 launches for Starlink satellites in 2020,^[49] in addition to 14 or 15 non-Starlink launches. At 29 launches, 13 of which for Starlink satellites, Falcon 9 had its most prolific year, and Falcon rockets were second most prolific rocket family of 2020, only behind China's Long March rocket family.^[49]

Flight No.	Date and time (UTC)	Version, Booster ^[2]	Launch site	Payload ^[1]	Payload mass	Orbit	Customer	Launch outcome	Booster landing
78	7 January 2020, 02:19:21 ^[49]	FB B5 Δ, B1048.4	CCAFS, SLC-40	Starlink 2 v1.0 (80 satellites)	15,600 kg (34,400 lb) ^[5]	LEO	SpaceX	Success	Success (drone ship)
Third large batch and second operational flight of Starlink constellation. One of the 60 satellites included a test coating to make the satellite less reflective, and thus less likely to interfere with ground-based astronomical observations. ^[48]									
79	19 January 2020, 15:30 ^[49]	FB B5 Δ, B1048.4	KSC, LC-39A	Crew Dragon in-flight abort test ^[49] (Dragon C205.1)	12,050 kg (26,570 lb)	Sub-orbital ^[49]	NASA (CTS) ^[49]	Success	No attempt
An atmospheric test of the Dragon 2 abort system after Max Q. The capsule fired its SuperDraco engines, reached an apogee of 40 km (25 mi), deployed parachutes after reentry, and splashed down in the ocean 31 km (19 mi) downrange from the launch site. The test was previously slated to be accomplished with the Crew Dragon Demo-1 capsule ^[49] but that test article exploded during a ground test of SuperDraco engines on 20 April 2019. ^[49] The abort test used the capsule originally intended for the first crewed flight ^[49] . As expected, the booster was destroyed by aerodynamic forces after the capsule aborted. ^[49] First flight of a Falcon 9 with only one functional stage — the second stage had a mass simulator in place of its engine.									
80	29 January 2020, 14:07 ^[50]	FB B5 Δ, B1051.3	CCAFS, SLC-40	Starlink 3 v1.0 (80 satellites)	15,600 kg (34,400 lb) ^[5]	LEO	SpaceX	Success	Success (drone ship)
Third operational and fourth large batch of Starlink satellites, deployed in a circular 290 km (180 mi) orbit. One of the fairing halves was caught, while the other was fished out of the ocean. ^[49]									
81	17 February 2020, 15:05 ^[50]	FB B5 Δ, B1056.4	CCAFS, SLC-40	Starlink 4 v1.0 (80 satellites)	15,600 kg (34,400 lb) ^[5]	LEO	SpaceX	Success	Failure (drone ship)
Fourth operational and fifth large batch of Starlink satellites. Used a new flight profile which deployed into a 212 km × 386 km (132 mi × 240 mi) elliptical orbit instead of launching into a circular orbit and firing the second stage engine twice. The first stage booster failed to land on the drone ship ^[49] due to incorrect wind data. ^[50] This was the first time a flight proven booster failed to land.									
82	7 March 2020, 04:50 ^[50]	FB B5 Δ, B1059.2	CCAFS, SLC-40	SpaceX CRS-20 (Dragon C112.3 Δ)	1,977 kg (4,359 lb) ^[50]	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
Last launch of phase 1 of the CRS contract. Carries Bartolomeo, an ESA platform for hosting external payloads onto ISS. ^[50] Originally scheduled to launch on 2 March 2020, the launch date was pushed back due to a second stage engine failure. SpaceX decided to swap out the second stage instead of replacing the faulty part. ^[50] It was SpaceX's 50th successful landing of a first stage booster, the third flight of the Dragon C112 and the last launch of the cargo Dragon spacecraft.									
83	18 March 2020, 12:19 ^[51]	FB B5 Δ, B1048.5	KSC, LC-39A	Starlink 5 v1.0 (80 satellites)	15,600 kg (34,400 lb) ^[5]	LEO	SpaceX	Success	Failure (drone ship)
Fifth operational launch of Starlink satellites. It was the first time a first stage booster flew for a fifth time and the second time the fairings were reused (Starlink flight in May 2019). ^[51] Towards the end of the first stage burn, the booster suffered premature shut down of an engine, the first of a Merlin 1D variant and first since the CRS-1 mission in October 2012. However, the payload still reached the targeted orbit. ^[51] This was the second Starlink launch booster landing failure in a row, later revealed to be caused by residual cleaning fluid trapped inside a sensor. ^[51]									
84	22 April 2020, 19:30 ^[51]	FB B5 Δ, B1051.4	KSC, LC-39A	Starlink 6 v1.0 (80 satellites)	15,600 kg (34,400 lb) ^[5]	LEO	SpaceX	Success	Success (drone ship)

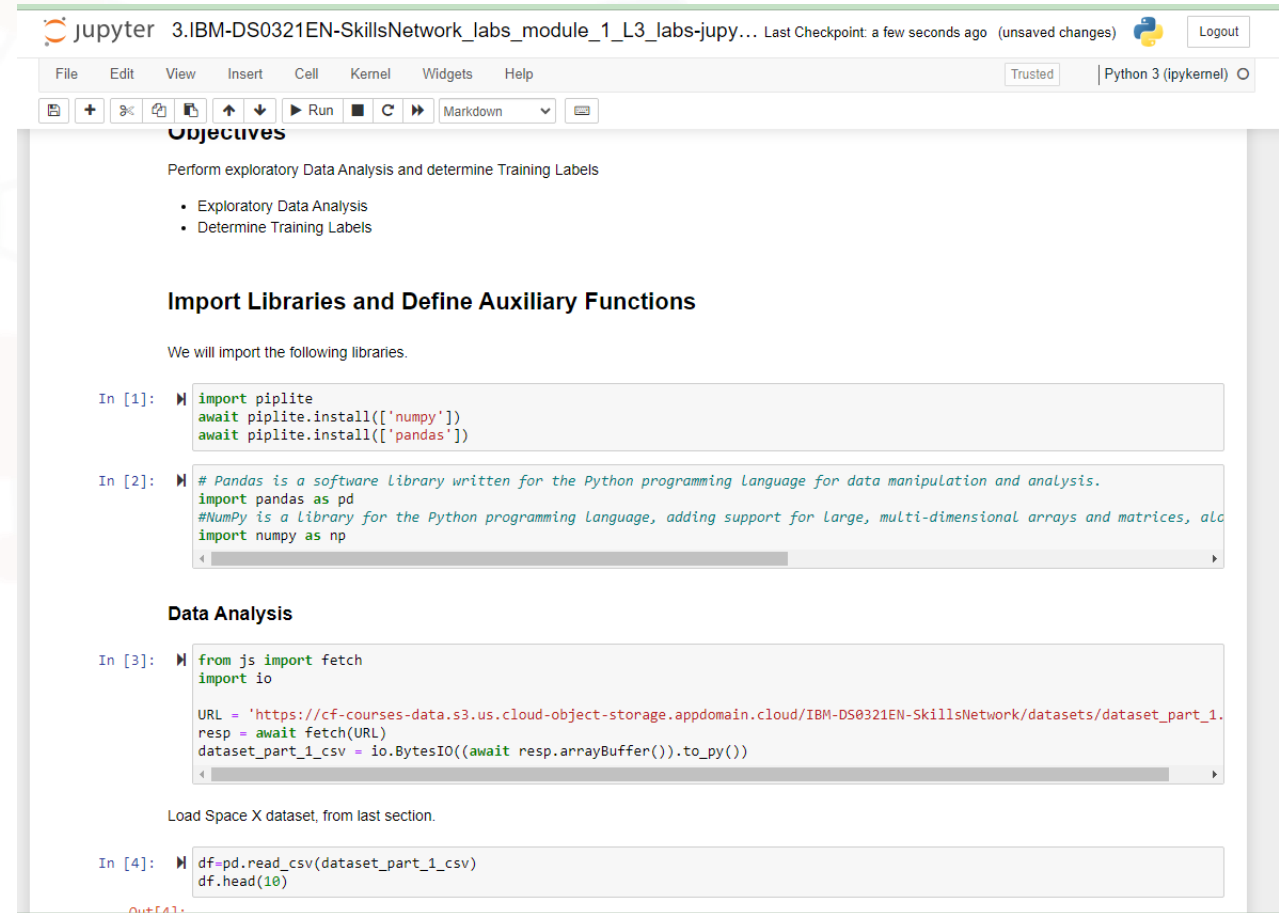
Objectives

Web scrap Falcon 9 launch records with BeautifulSoup:

- Extract a Falcon 9 launch records HTML table from Wikipedia
- Parse the table and convert it into a Pandas data frame

DATA MANIPULATION/WRANGLING-PANDAS

1. Defining a set of unsuccessful outcomes
2. Creating a list, `landing_class`, where the element is 0 if the corresponding row in `Outcome` is in the set `bad_outcome`, otherwise, it's 1.
3. Create a `Class` column that contains the values from the list `landing_class`
4. Export the `DataFrame` as a `.csv` file



The screenshot shows a Jupyter Notebook titled "3.IBM-DS0321EN-SkillsNetwork_labs_module_1_L3_labs-jupy...". The interface includes a top bar with "jupyter", the title, and a "Logout" button. Below the top bar is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". A toolbar contains icons for file operations, running, and markdown. The notebook content is divided into sections: "Objectives", "Import Libraries and Define Auxiliary Functions", and "Data Analysis".

Objectives

Perform exploratory Data Analysis and determine Training Labels

- Exploratory Data Analysis
- Determine Training Labels

Import Libraries and Define Auxiliary Functions

We will import the following libraries.

```
In [1]: import piplite
        await piplite.install(['numpy'])
        await piplite.install(['pandas'])

In [2]: # Pandas is a software library written for the Python programming language for data manipulation and analysis.
        import pandas as pd
        # NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large number of mathematical functions.
        import numpy as np
```

Data Analysis

```
In [3]: from js import fetch
        import io

        URL = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv'
        resp = await fetch(URL)
        dataset_part_1_csv = io.BytesIO((await resp.arrayBuffer()).to_py())

Load Space X dataset, from last section.

In [4]: df = pd.read_csv(dataset_part_1_csv)
        df.head(10)
```

Out[4]:

EXPLORATORY DATA ANALYSIS(EDA) -VISUALIZATION

SCATTER CHARTS

Scatter charts were produced to visualize the relationships between:

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

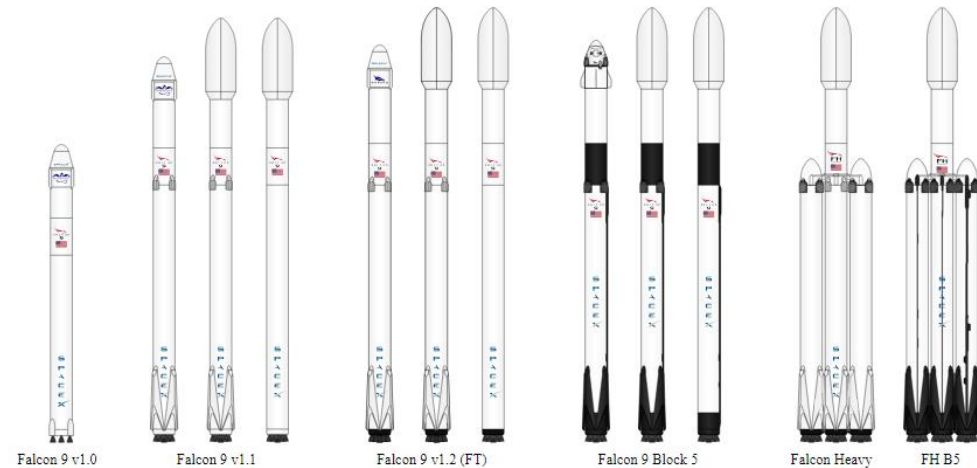
BAR CHART

A bar chart was produced to visualize the relationship between:

- Success Rate and Orbit Type

LINE CHARTS

- Success Rate and



Results

- Exploratory Data Analytics
- Interactive Data Analytics
- Predictive Analytics



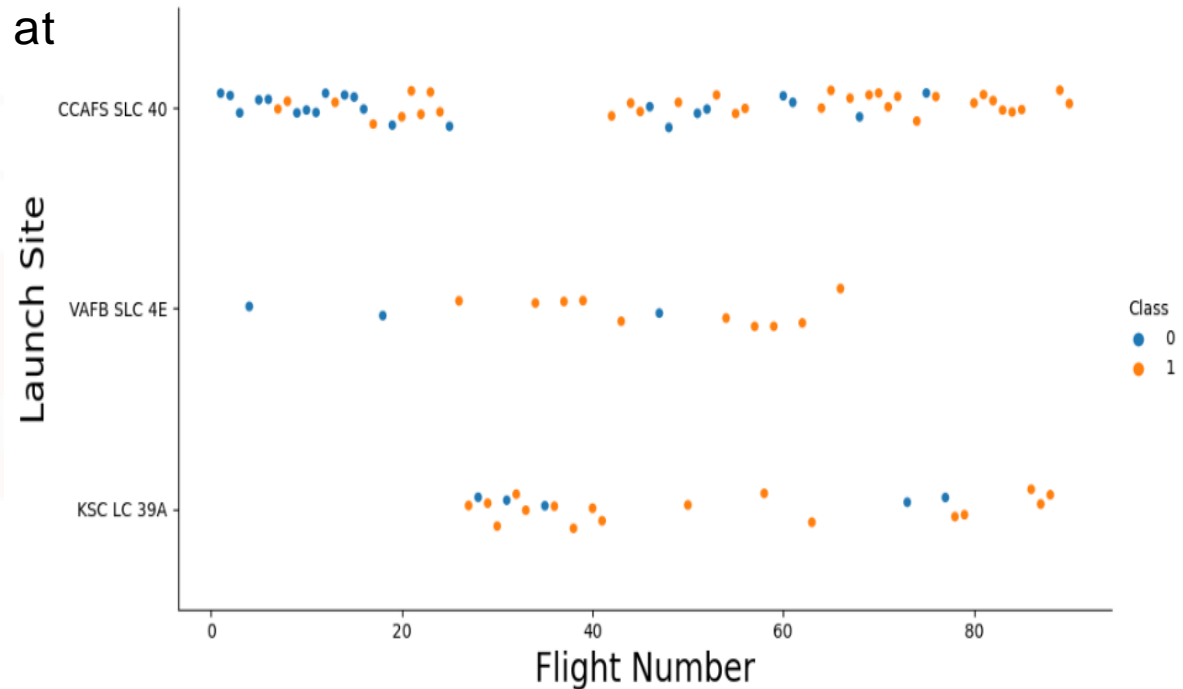
EDA with Visualizations



Launch Site Vs Flight Number (Cat plot)

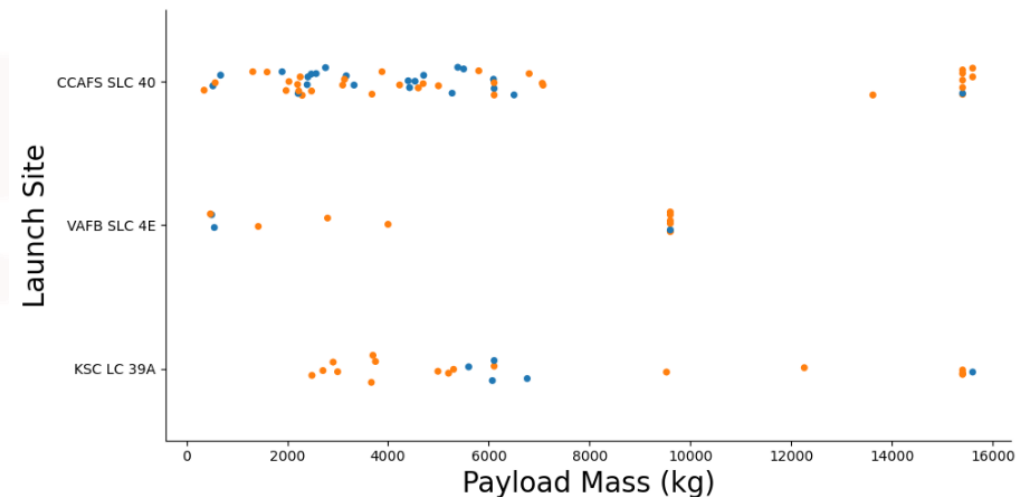
The Cat plot of Launch Site vs. Flight Number shows that:

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



Launch Site vs Payload Mass

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

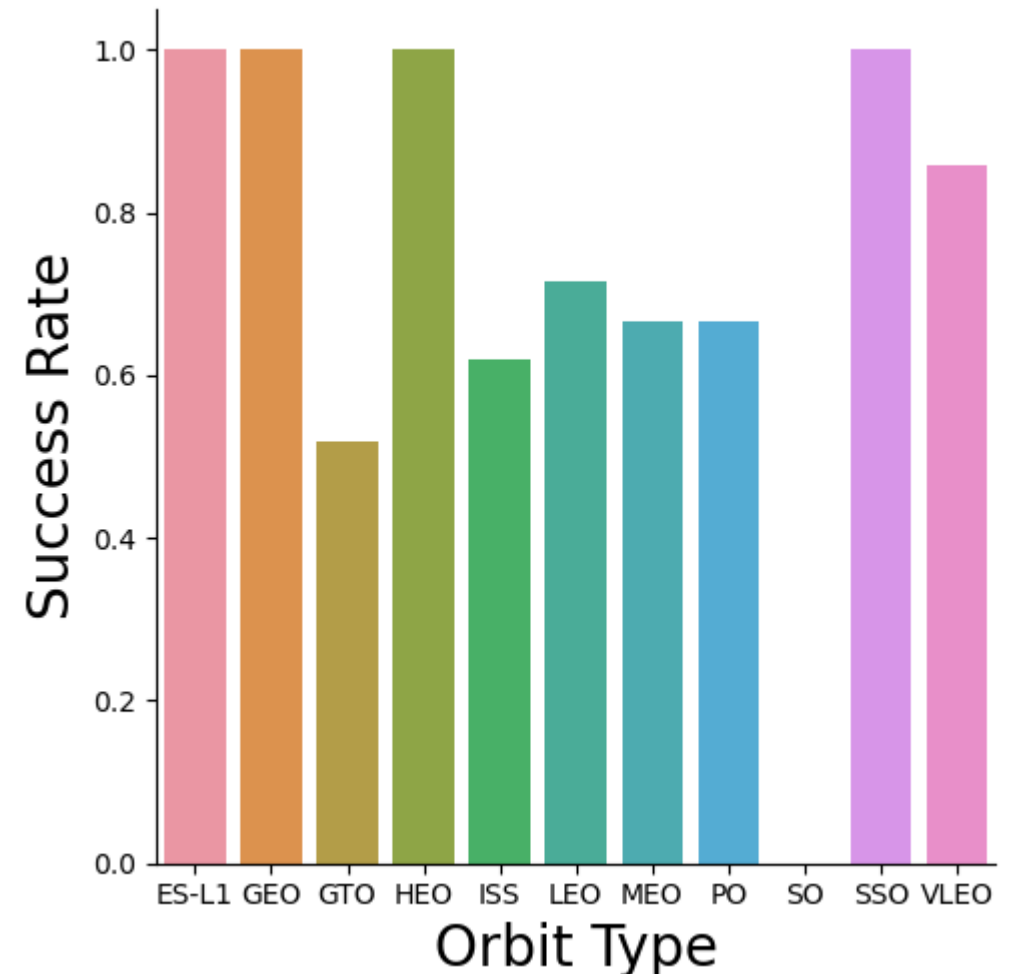


Success Rate vs Orbit Type

The bar chart of Success Rate vs. Orbit Type shows that the following orbits have the highest (100%) success rate:

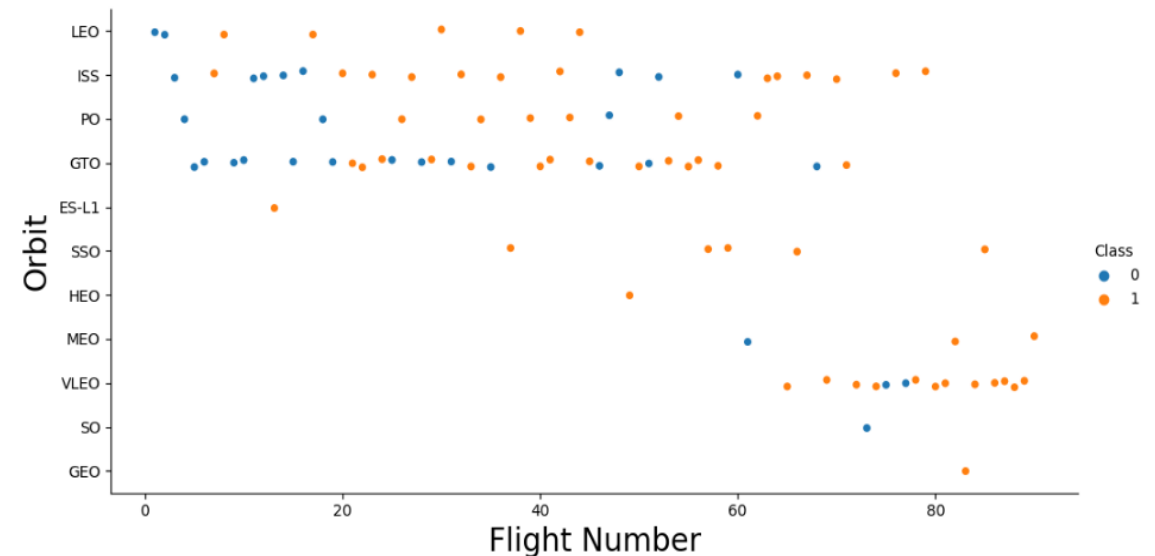
- ES-L1 (Earth-Sun First Lagrangian Point)
- GEO (Geostationary Orbit)
- HEO (High Earth Orbit)
- SSO (Sun-synchronous Orbit)

Observation: SO = 0% success rate



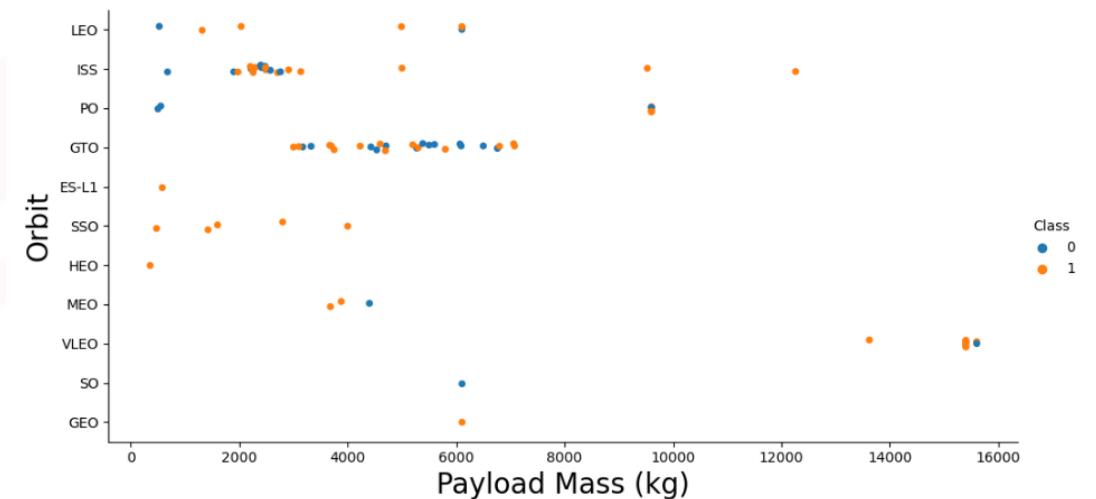
Orbit Type vs Flight Number

- This scatter plot of Orbit Type vs. Flight number shows a few useful things that the previous plots did not, such as:
- The 100% success rate of GEO, HEO, and ES-L1 orbits can be explained by only having 1 flight into the respective orbits.
- The 100% success rate in SSO is more impressive, with 5 successful flights.
- There is little relationship between Flight Number and Success Rate for GTO.



Orbit Type vs Payload Mass

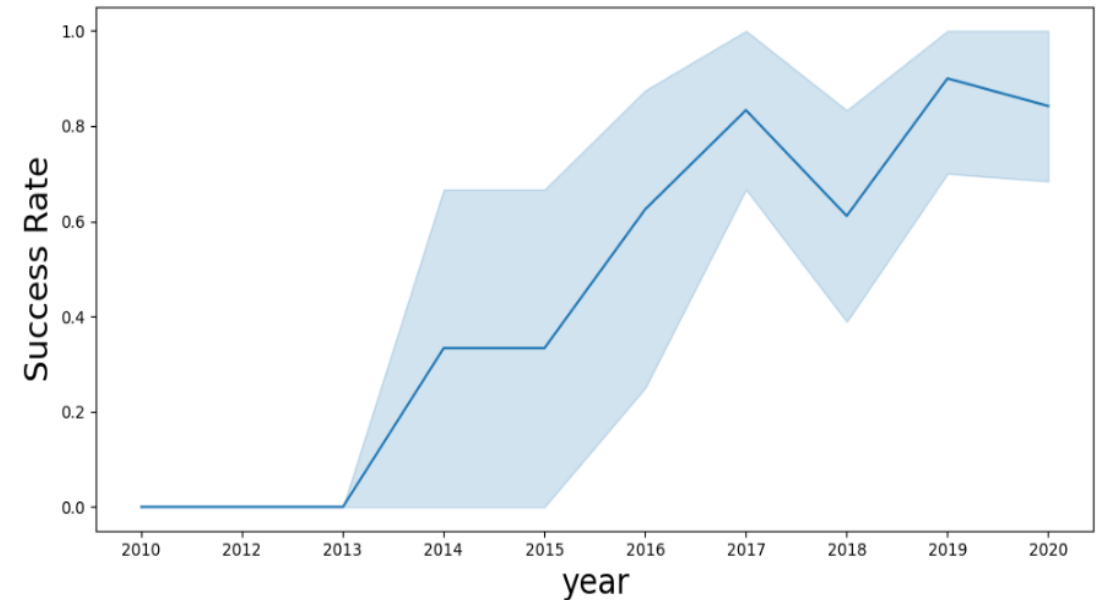
- This scatter plot of Orbit Type vs. Payload Mass shows that:
- The following orbit types have more success with
- heavy payloads:
 - PO (although the number of data points is small)
 - ISS
 - LEO
- For GTO, the relationship between payload mass and success rate is unclear.
- VLEO (Very Low Earth Orbit) launches are associated with heavier payloads, which makes intuitive sense.



Success Rate vs Year

The line chart of yearly average success rate shows that:

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



EDA with SQL

Launch Sites

```
Out[18]: ['CCAFS LC-40', 'VAFB SLC-4E', 'KSC LC-39A', 'CCAFS SLC-40', nan]
```

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

```
[17]: %sql SELECT LAUNCH_SITE FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
* sqlite:///my_data1.db
Done.
```

```
Out[17]: Launch_Site
         CCAFS LC-40
         CCAFS LC-40
         CCAFS LC-40
         CCAFS LC-40
         CCAFS LC-40
```


Avg Payload Mass

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

```
In [19]: %sql SELECT SUM(PAYLOAD_MASS_KG_) AS TOTAL_PAYLOAD_MASS FROM SPACEXTBL \
        WHERE CUSTOMER = 'NASA (CRS)';

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

```
Out[19]: TOTAL_PAYLOAD_MASS
        45596.0
```

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

```
] : %sql SELECT AVG(PAYLOAD_MASS_KG_) AS AVERAGE_PAYLOAD_MASS FROM SPACEXTBL \
    WHERE BOOSTER_VERSION = 'F9 v1.1';

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

```
t[20]: AVERAGE_PAYLOAD_MASS
        2928.4
```

Successful Landing Dates

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

Hint: Use min function

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

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

```
0]: FIRST_SUCCESSFUL_GROUND_LANDING
```

```
01/08/2018
```

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

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

```
1]: FIRST_SUCCESSFUL_GROUND_LANDING
```

```
22/12/2015
```

Booster Versions

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 FROM SPACEXTBL WHERE (LANDING_OUTCOME = 'Success (drone ship)') AND (PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000)
```

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

9]: **Booster_Version**

F9 FT B1022

F9 FT B1026

F9 FT B1021.2

F9 FT B1031.2

Successful and Failure Missions

List the total number of successful and failure mission outcomes

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

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

[42]:

Mission_Outcome	TOTAL_NUMBER
None	0
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Booster Versions with Payloads

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

```
%sql SELECT DISTINCT(BOOSTER_VERSION) FROM SPACEXTBL \
      WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

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

4]: **Booster_Version**

F9 B5 B1048.4

F9 B5 B1049.4

F9 B5 B1051.3

F9 B5 B1056.4

F9 B5 B1048.5

F9 B5 B1051.4

F9 B5 B1049.5

F9 B5 B1060.2

F9 B5 B1058.3

F9 B5 B1051.6

F9 B5 B1060.3

F9 B5 B1049.7

Month and Year of Failure

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: SQLite 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,BOOSTER_VERSION, LAUNCH_SITE, [LANDING_OUTCOME] FROM SPACEXTBL where [LANDING_OUTCOME] = 'Failure (drone ship)'

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

```
[1]:
```

	month	Date	Booster_Version	Launch_Site	Landing_Outcome
	10	01/10/2015	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
	04	14/04/2015	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

Count of Outcomes

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

```
%sql SELECT [LANDING_OUTCOME], count(*) as count_outcomes FROM SPACEXTBL WHERE DATE between '04-06-2010' and '20-03-2017' gro
```

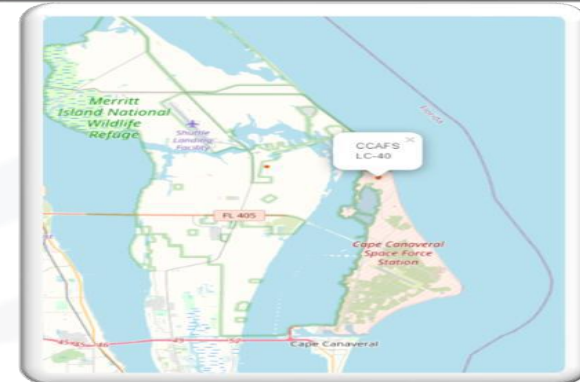
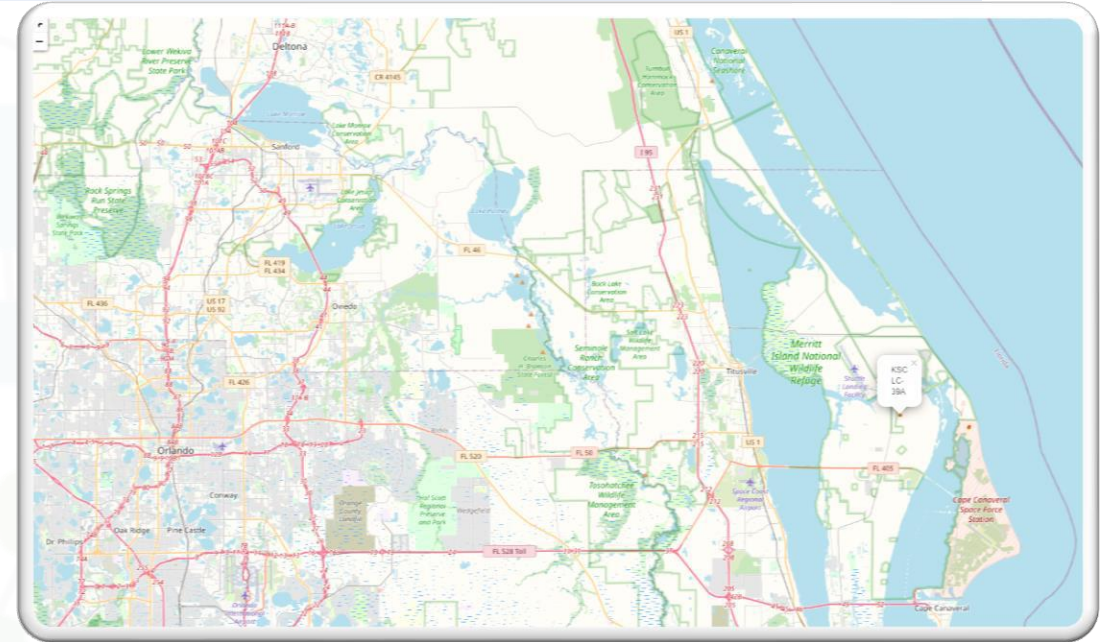
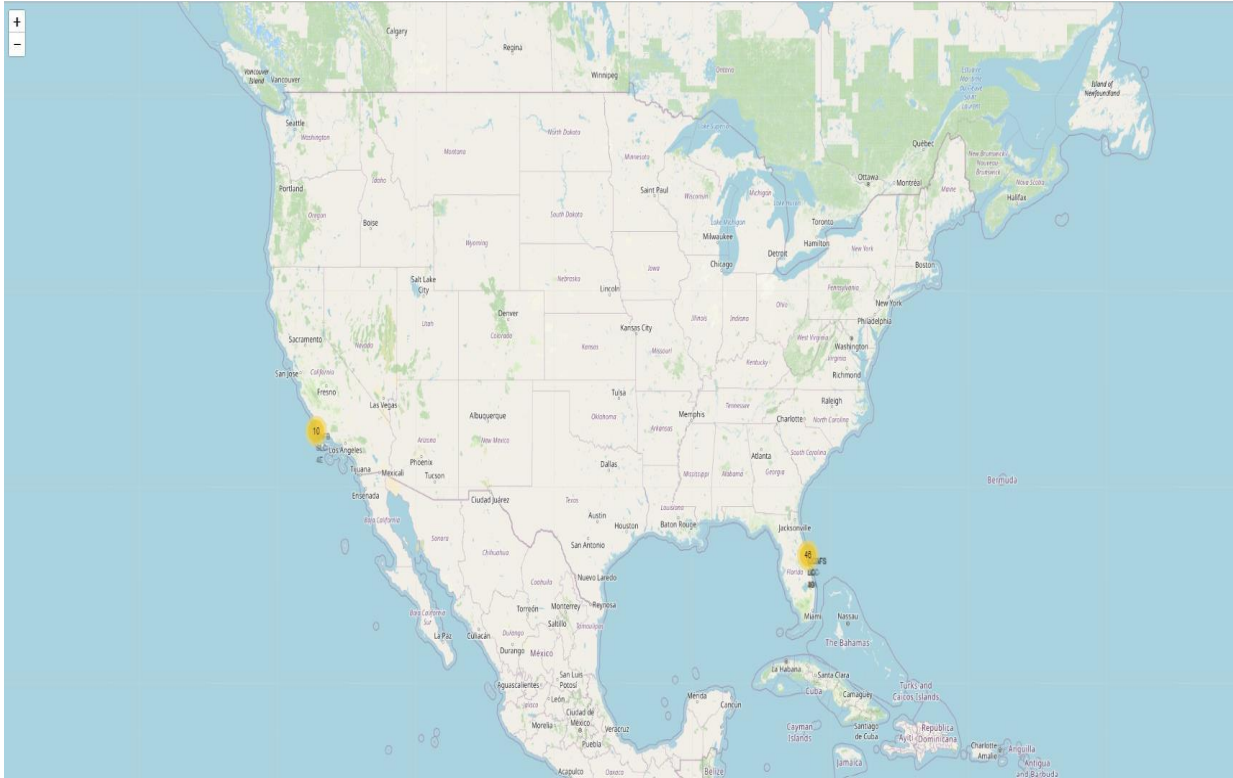
```
* sqlite:///my_data1.db  
Done.
```

```
53]:
```

Landing_Outcome	count_outcomes
Success	20
No attempt	10
Success (drone ship)	8
Success (ground pad)	7
Failure (drone ship)	3
Failure	3
Failure (parachute)	2
Controlled (ocean)	2
No attempt	1

Launch Site Analysis- Folium Interactive Map

Launching Sites



Launch Outcomes

At Each Launch Site

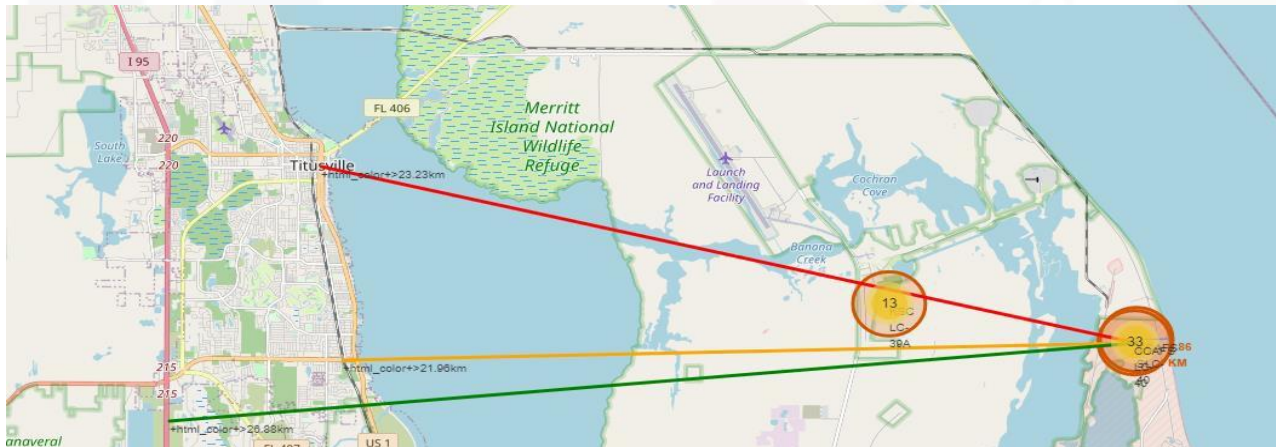
- **Green** markers for successful launches
- **Red** markers for unsuccessful launches
- Launch site **CCAFS SLC-40** has a **3/7 success rate (42.9%)**



Distance to Proximities

CCAFS SLC-40

- **Coasts:** help ensure that spent stages dropped along the launch path or failed launches don't fall on people or property.
- **Safety / Security:** needs to be an exclusion zone around the launch site to keep unauthorized people away and keep people safe.
- **Transportation/Infrastructure and Cities:** need to be away from anything a failed launch can damage, but still close enough to roads/rails/docks to be able to bring people and material to or from it in support of launch activities.



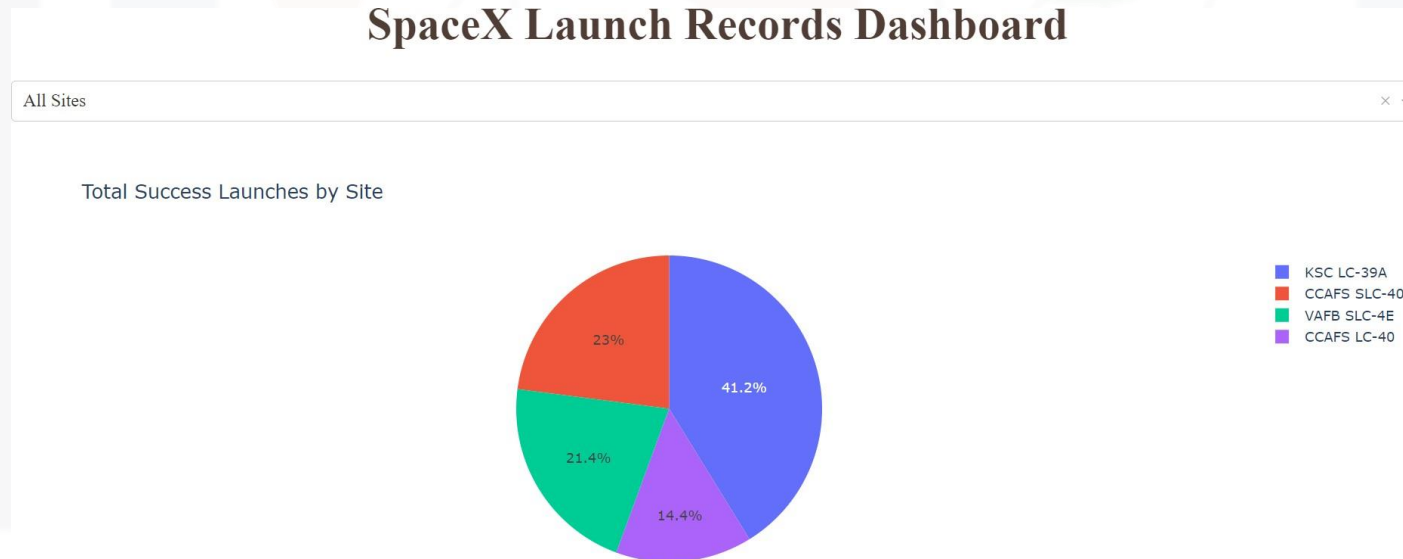
Plotly Analytics-Dashboards

Launch Success by Site

Success as Percent of Total

- **KSC LC-39A** has the **most successful launches** amongst launch sites (**41.2%**)

SpaceX Launch Records Dashboard

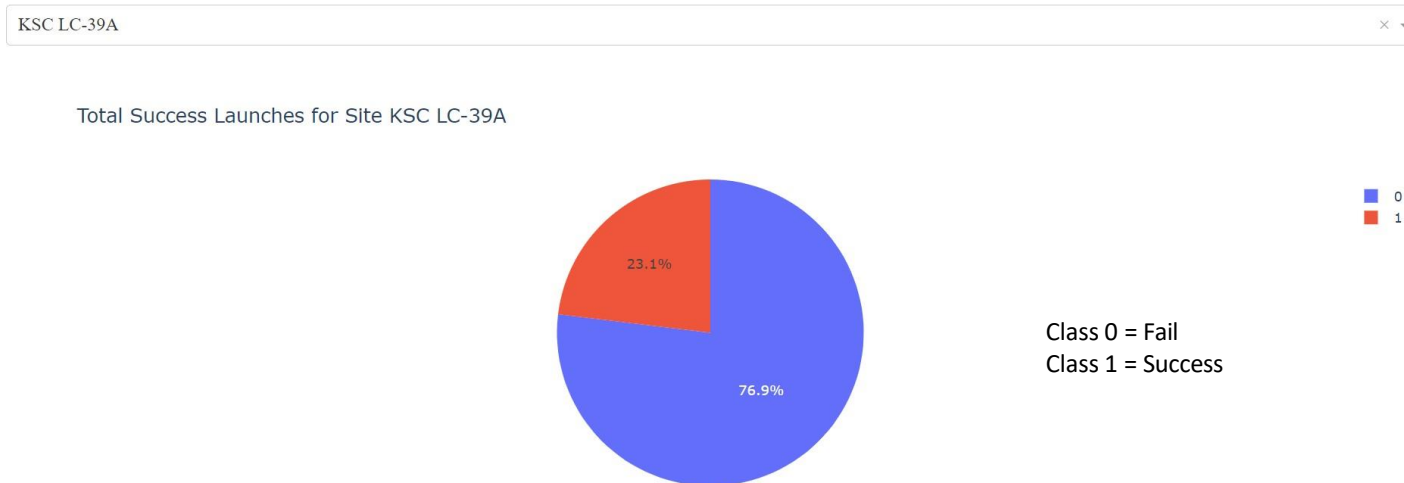


Launch Success (KSC LC-29A)

Success as Percent of Total

- **KSC LC-39A** has the **highest success rate** amongst launch sites (**76.9%**)
- 10 successful launches and 3 failed launches

SpaceX Launch Records Dashboard



Payload Mass and Success

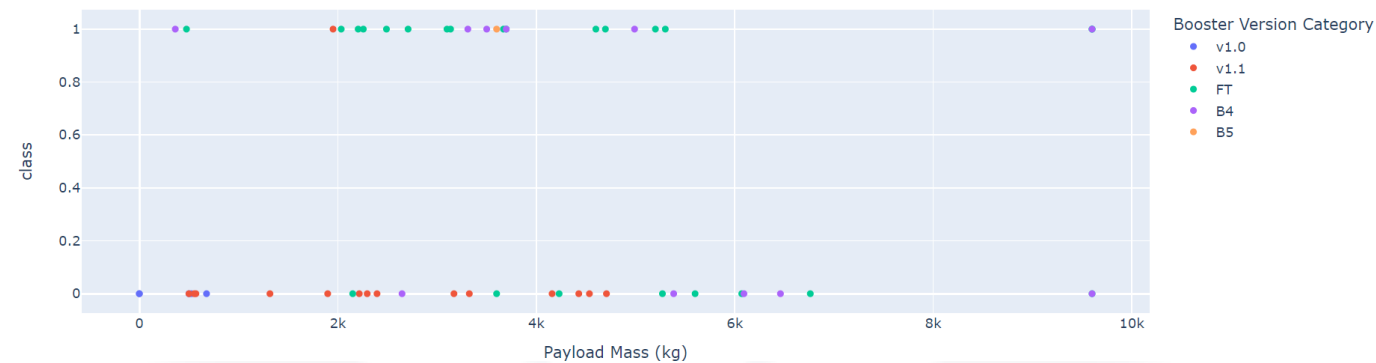
By Booster Version

- **Payloads between 2,000 kg and 5,000 kg have the highest success rate**
- 1 indicating successful outcome and 0 indicating an unsuccessful outcome

Payload range (Kg):



Correlation Between Payload and Success for All Sites



Predictive Analytics

Classification

Accuracy

- All the **models** performed at about the same level and had the **same scores** and **accuracy**. This is likely due to the **small dataset**. The **Decision Tree model slightly outperformed** the rest when looking at `.best_score_`
- `.best_score_` is the average of all cv folds for a single combination of the parameters

```
: models = {'KNeighbors': knn_cv.best_score_,
            'DecisionTree': tree_cv.best_score_,
            'LogisticRegression': logreg_cv.best_score_,
            'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm, 'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)

Best model is DecisionTree with a score of 0.9017857142857142
Best params is : {'criterion': 'gini', 'max_depth': 16, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 10, 'splitter': 'random'}
```

Confusion Matrices

- Confusion Matrix Outputs:

- 12 True positive
- 3 True negative
- **3 False positive**
- 0 False Negative

- **Precision** = $TP / (TP + FP)$

- $12 / 15 = .80$

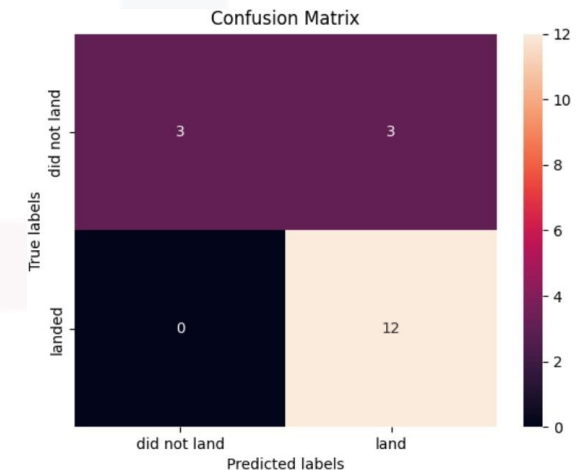
- **Recall** = $TP / (TP + FN)$

- $12 / 12 = 1$

- **F1 Score** = $2 * (Precision * Recall) / (Precision + Recall)$

- $2 * (.8 * 1) / (.8 + 1) = .89$

- **Accuracy** = $(TP + TN) / (TP + TN + FP + FN) = .833$



Conclusion

Observations:

- **Model Performance:** The models performed similarly on the test set with the decision tree model slightly outperforming
- **Equator:** Most of the launch sites are near the equator for an additional natural boost - due to the rotational speed of earth - which helps save the cost of putting in extra fuel and boosters
- **Coast:** All the launch sites are close to the coast
- **Launch Success:** Increases over time
- **KSC LC-39A:** Has the highest success rate among launch sites. Has a 100% success rate for launches less than 5,500 kg
- **Orbits:** ES-L1, GEO, HEO, and SSO have a 100% success rate
- **Payload Mass:** Across all launch sites, the higher the payload mass (kg), the higher the success rate

Conclusion

Things to Consider

- **Dataset:** A larger dataset will help build on the predictive analytics results to help understand if the findings can be generalizable to a larger data set
- **Feature Analysis / PCA:** Additional feature analysis or principal component analysis should be conducted to see if it can help improve accuracy

APPENDIX

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