IBM Developer
SKILLS NETWORK

Predicting SpaceX's Falcon 9 First Stage Landings with Data Science & Machine Learning

IBM – Data Science and Machine Learning Capstone Project

Björn Thomsen, M.A.

June 18th, 2023

Outline

Executive Summary

This project aims to predict the successful landing of the Falcon 9 first stage.

Summary of Methodologies:

- **Data Collection** with the SpaceX Rest API & Web Scraping of Falcon 9 and Falcon Heavy launches records from Wikipedia
- **Data Wrangling** to find patterns in the data & determine what would be the label for training supervised models
- **Exploratory Data Analysis** (EDA) with **SQL** & with **Data Visualizations** in order to prepare Data Feature Engineering
- **Interactive Visual Analytics** using Folium to find geographical patterns about launch sites
- **Interactive Dashboard** with Plotly Dash to visualize different success factors like payload and launch site
- **Machine Learning** pipeline to predict if the first stage will land

Summary of the Results:

Factors for a successful landing:

- Among all launch sites **KSC LC-39A** has the highest success rate
- Among all orbits **ES-L1, GEO, HEO, and SSO** have with 100% the highest success rate
- The higher the **payload**, the greater the success rate
- **Success rate for launches** grows over time

Introduction

Project Background

This project aims to predict the successful landing of the Falcon 9 first stage. SpaceX's cost-effective approach to rocket launches, with a price tag of $62 million compared to competitors' $165 million, hinges on reusing this stage. By accurately determining the landing outcome, we can estimate launch costs. Predictive insight like this is invaluable for alternate companies vying for rocket launch bids against SpaceX.

Problems to we want to solve

Through this project, our objective is to examine the impact of payload, launch sites, number of flights, and orbits on the success rate of controlled stage one landings. We demonstrate our proficiency in achieving this goal utilizing Python, employing various data science tools and techniques such as Matplotlib, Folium, and machine learning methodologies.

Section 1

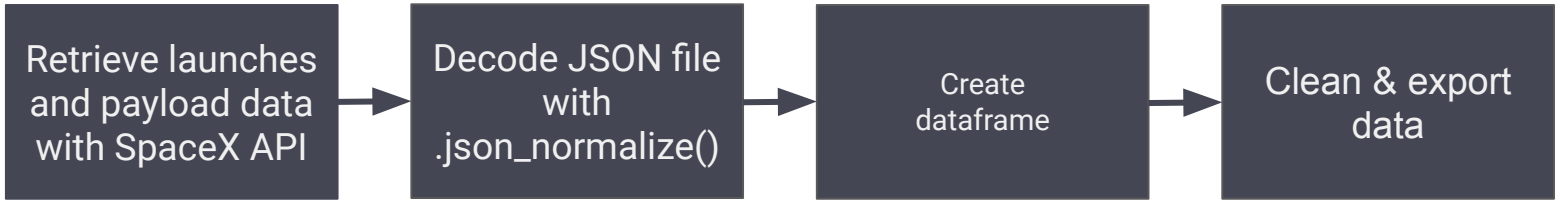
Methodology

Methodology

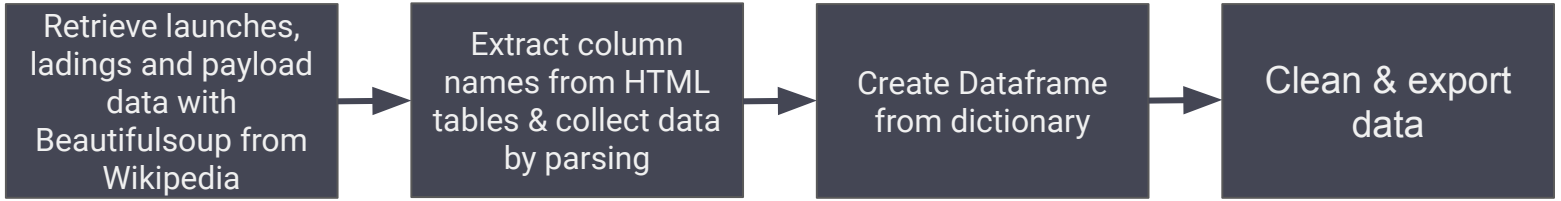
- **Data Collection** with:
  - SpaceX REST API
  - Beautifulsoup for web scraping from Wikipedia
- **Data Wrangling** with:
  - hot encoding, adding missing and deleting unnecessary values
- **Data Exploration** with:
  - Exploratory Data Analysis (EDA) using SQL and Visualization with Seaborn
- **Interactive Visualization** with:
  - Folium to find and mark geographical patterns
  - Plotly Dash
- **Machine Learning** with:
  - sklearn to perform predictive analysis using classification models

Data Collection

Data Collection from SpaceX REST API*



Data Collection with Web Scraping from Wikipedia**



*https://api.spacexdata.com/v4/
**https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922

Data Collection API (1/2)

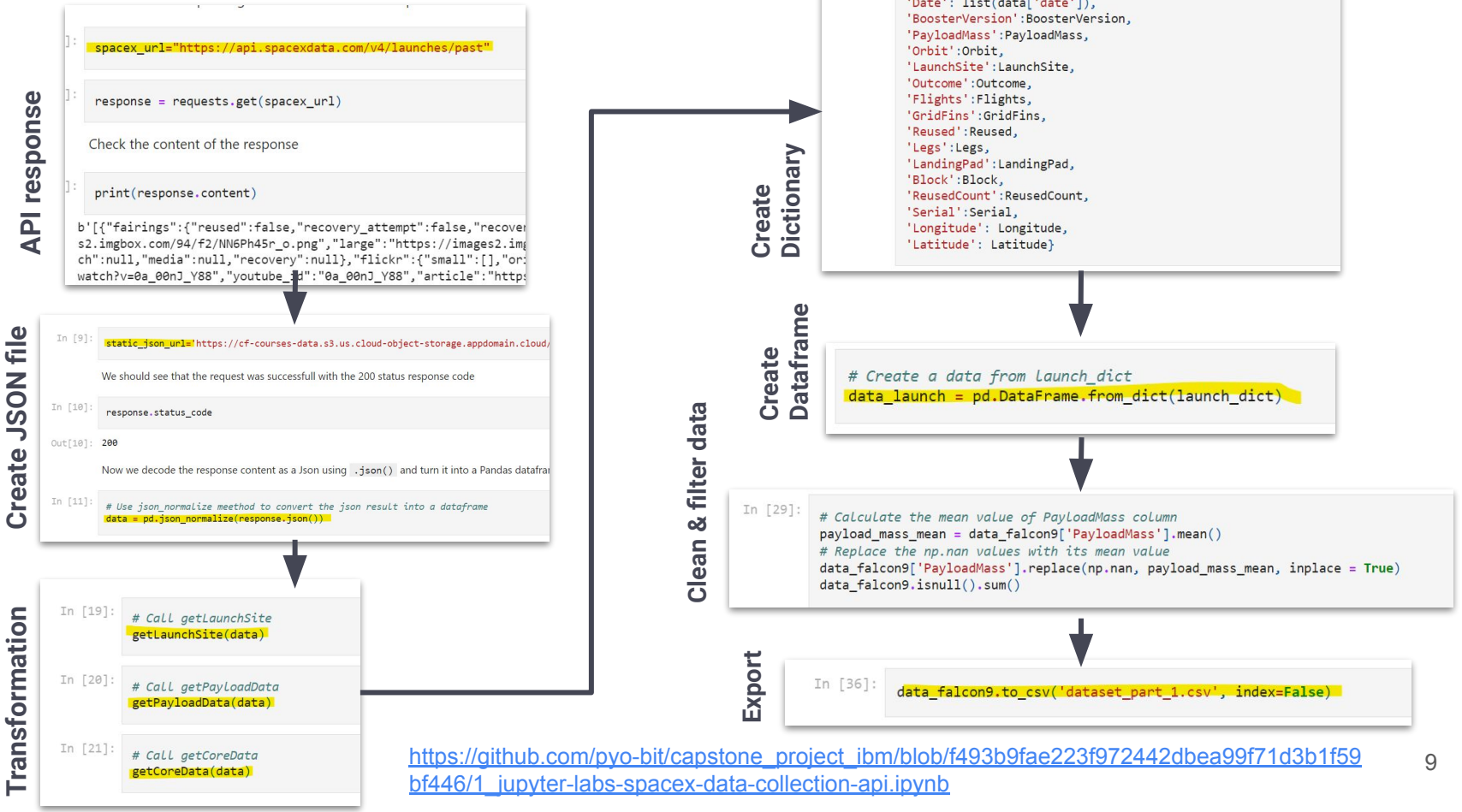
The first task involves collecting data from the SpaceX REST API and ensuring it is in the correct format. Several helper functions were defined to extract specific information from the API, such as the booster name, launch site details, payload information, and core data.

After importing the necessary libraries, a GET request was made to the SpaceX API to fetch the rocket launch data. Additionally, a static response object was used to ensure consistent JSON results for the project.

The next task was to request and parse the SpaceX launch data using the GET request. The response content was decoded as JSON and converted into a Pandas dataframe using the json_normalize() method. The DataFrame contained information about past SpaceX launches, including details such as static fire dates, success status, failures, crew, payloads, launchpads, and more.

https://github.com/pyo-bit/capstone_project_ibm/blob/f493b9fae223f972442dbea99f71d3b1f59bf446/1_jupyter-labs-spacex-data-collection-api.ipynb

Data Collection API (2/2)



https://github.com/pyo-bit/capstone_project_ibm/blob/f493b9fae223f972442dbea99f71d3b1f59bf446/1_jupyter-labs-spacex-data-collection-api.ipynb

Data Collection Web Scraping (1/2)

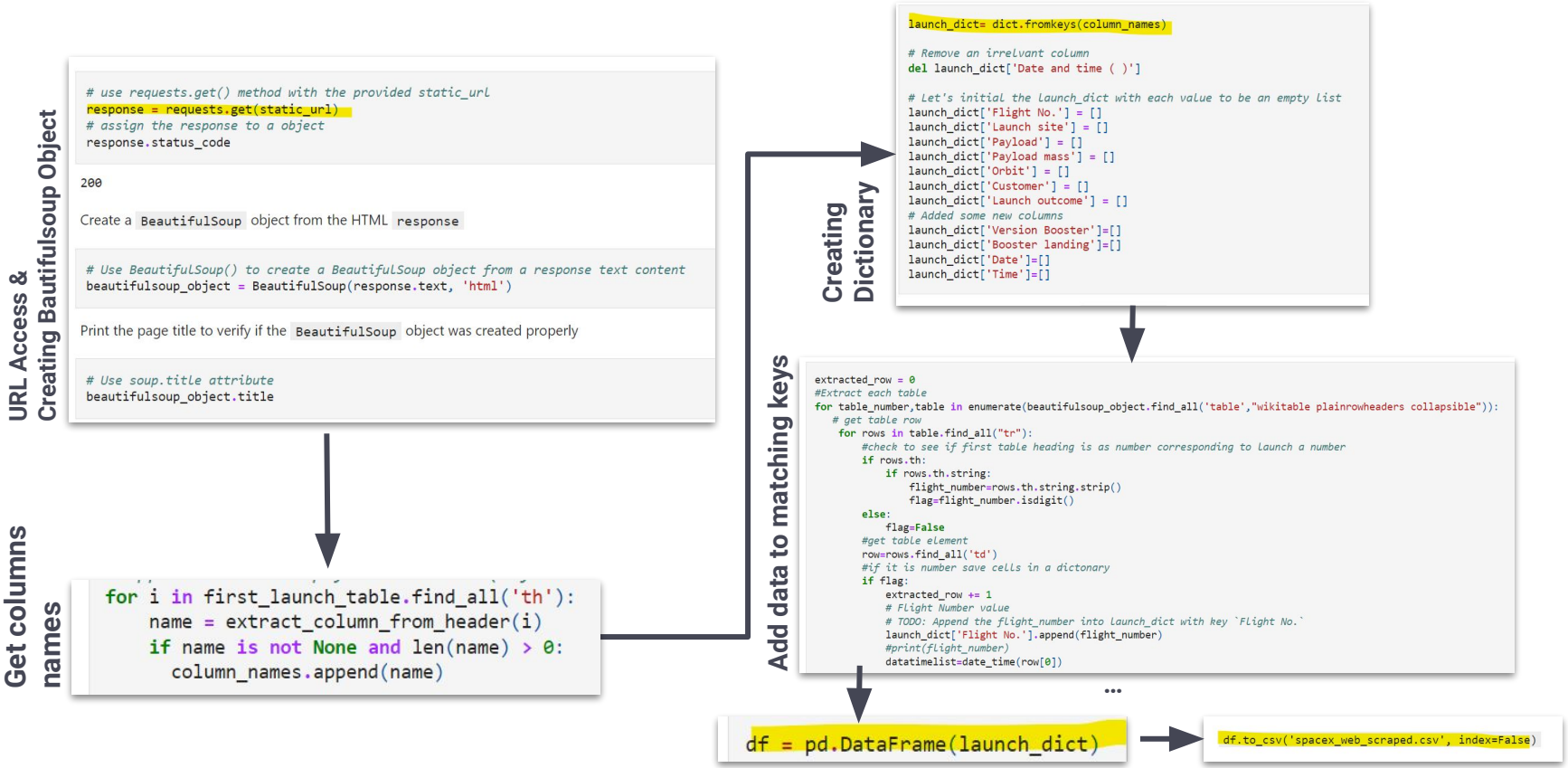
To collect historical launch records, we are conducting web scraping from a Wikipedia page titled "List of Falcon 9 and Falcon Heavy launches."

The objective is to extract the launch records table from the Wikipedia page, parse it, and convert it into a Pandas DataFrame. We are using the BeautifulSoup library for web scraping and the requests library for making HTTP requests. We have provided helper functions to process the scraped HTML table.

We scrape the launch records, including flight number, launch site, payload, payload mass, orbit, customer, launch outcome, booster version, booster landing, date, and time. Finally, we export the DataFrame to a CSV file.

https://github.com/pyo-bit/capstone_project_ibm/blob/0f3450907c80ca74864883993a817de4fb546f47/2_jupyter-labs-webscraping.ipynb

Data Collection Web Scraping (2/2)



https://github.com/pyo-bit/capstone_project_ibm/blob/0f3450907c80ca74864883993a817de4fb546f47/2_jupyter-labs-webscraping.ipynb

Data Wrangling

The dataset covers cases where the first stage did not land successfully (false ocean/false RTLS, false ASDS). These strings have to be translated into 0 (=failure) and 1 (=success).

After determining the data labels, the launches or each site, occurrences in orbit, occurrences of mission have to be calculate, followed by transforming the landing outcome. These actions included following steps:

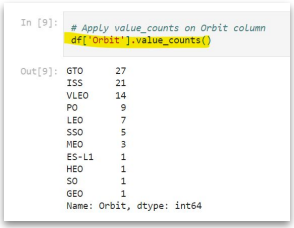
1. Calculating the number of launches on each site
2. Calculating the number and occurrence of each orbit
3. Calculate the number and occurence of mission outcome per orbit type
4. Create a landing outcome label from Outcome column

1.

```
In [8]:  # Apply value_counts() on column LaunchSite
         df['LaunchSite'].value_counts()

Out[8]:  CCAFS SLC 40    55
         KSC LC 39A      22
         VAFB SLC 4E     13
         Name: LaunchSite, dtype: int64
```

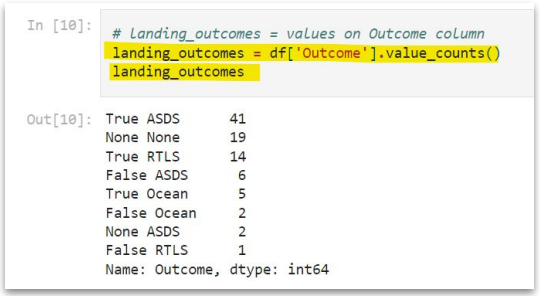
2.



```
In [9]:  # Apply value_counts on Orbit column
         df['Orbit'].value_counts()

Out[9]:  GTO     27
         ISS     21
         VLEO    14
         PO       9
         LEO      7
         SSO      5
         MEO      3
         ES-L1    1
         HEO      1
         SO       1
         GEO      1
         Name: Orbit, dtype: int64
```

3.



```
In [10]: # landing_outcomes = values on Outcome column
         landing_outcomes = df['Outcome'].value_counts()
         landing_outcomes

Out[10]: True ASDS      41
         None None      19
         True RTLS      14
         False ASDS      6
         True Ocean      5
         False Ocean     2
         None ASDS       2
         False RTLS      1
         Name: Outcome, dtype: int64
```

4.

```
In [14]: # landing_class = 0 if bad_outcome
         # landing_class = 1 otherwise
         landing_class = []
         # appending outcome to list
         for key, value in df['Outcome'].items():
           if value in bad_outcomes:
             landing_class.append(0)
           else:
             landing_class.append(1)
```

https://github.com/pyo-bit/capstone_project_ibm/blob/0a74185c8558a6a2ee6bc4e0298dc851a34ae331/3_jupyter-labs-spacex-data-wrangling.ipynb

EDA with Data Visualization

Following charts have been created:

- **Scatter Graphs**
  - number vs. payload
  - number vs. launch site
  - payload vs. launch site
  - payload vs. orbit

- **Bar Graph**
  - success rate vs. orbit
- **Line Graph**
  - success rate vs. year

https://github.com/pyo-bit/capstone_project_ibm/blob/03be254c6c0abf3ac1d98271b14aeed21668e416/5_jupyter_lab_eda_with_visualization.ipynb

EDA with SQL

Following SQL queries were performed in order to gain more insights:

- Displaying the names of the unique launch sites in the space mission
- Displaying 5 records where launch sites begin with the string 'KSC'
- Displaying the total payload mass carried by boosters launched by NASA (CRS)
- Displaying average payload mass carried by booster version F9 v1.1
- Listing the date where the successful landing outcome in drone ship was achieved.
- Listing the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000
- Listing the total number of successful and failure mission outcomes
- Listing the names of the booster_versions which have carried the maximum payload mass. Use a subquery
- Listing the records which will display the month names, successful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017
- Ranking 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, DATE,BOOSTER_VERSION, LAUNCH_SITE, [LANDING_OUTCOME] \
FROM SPACEXTBL where [Landing_Outcome] = 'Success (ground pad)' and substr(Date,7,4)='2017' \
ORDER BY month;
```

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

month	Date	Booster_Version	Launch_Site	Landing_Outcome
01	05/01/2017	F9 FT B1032.1	KSC LC-39A	Success (ground pad)
02	19/02/2017	F9 FT B1031.1	KSC LC-39A	Success (ground pad)
03	06/03/2017	F9 FT B1035.1	KSC LC-39A	Success (ground pad)
07	09/07/2017	F9 B4 B1040.1	KSC LC-39A	Success (ground pad)
08	14/08/2017	F9 B4 B1039.1	KSC LC-39A	Success (ground pad)
12	15/12/2017	F9 FT B1035.2	CCAFS SLC-40	Success (ground pad)

Example: Listing the records which will display the month names, successful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017

https://github.com/pyo-bit/capstone_project_ibm/blob/b8bc810db86c6e32ffd626de11127e24dd33c2ca/4_jupyter-labs-eda-sql-edx_sqllite.ipynb

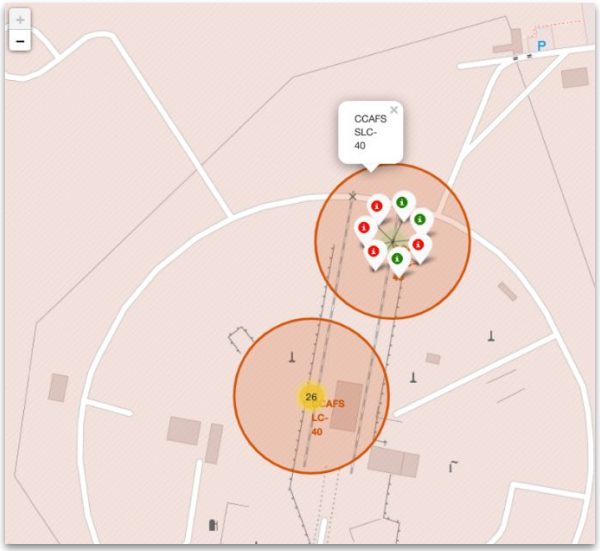
Building an Interactive Map with Folium

Using a Folium object, we generate a map centered on the NASA Johnson Space Center in Houston, Texas.

- Additionally, we plot a **blue** marker on the NASA Johnson Space Center location and **red** markers at each launch site for the Falcon 9, accompanied by labels indicating the respective launch site names.
- Furthermore, we mark successful landings **green** and unsuccessful landings **red**.
- …and calculate the distance between the launch site and closest key locations, like railways, highways, cities, etc.

Methods:
folium.Circle, folium.map.Marker, folium.plugin.MarkerCluster, folium.PolyLine, folium.features.DivIcon

Explanation:
By utilizing Folium, we can easily generate interactive maps that provide a comprehensive view of launch site locations and their surrounding areas. This enables us to visually analyze the spatial distribution of launch sites and identify any noticeable patterns or trends.



Example: Folium Marker Cluster over CCAFS SLC-40 launch site.

https://github.com/pyo-bit/capstone_project_ibm/blob/3d4de4c5054ee0c57d9ff7e6126a68288157dc12/6_interactive_visual_analytics_with_folium.ipynb

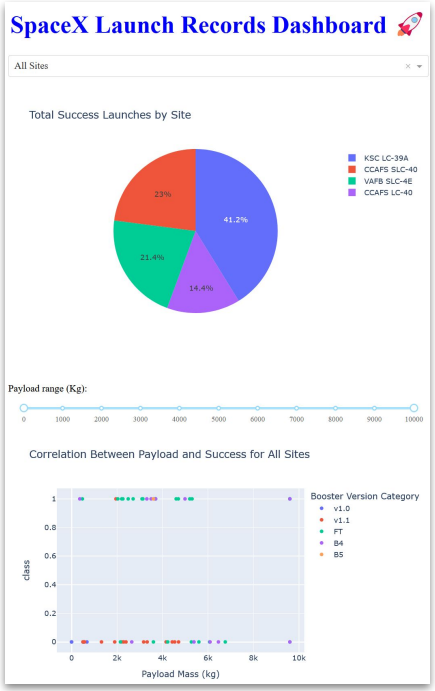
Building a Dashboard with Plotly Dash

Plotly Dash is a **Python framework** that enables the creation of interactive web-based dashboards. It provides an integration of data visualization components and interactivity, allowing users to build and deploy dynamic dashboards for data exploration.

To gain insights into the relationship between success rates, payload mass, and launch sites, we have developed a dashboard using Plotly Dash. This dashboard incorporates various components created within a Python file, including:

- **Dropdown** to choose launch sites
- **Pie Chart** to compare success / failure rates by launch site
- **Range Slider** to select a payload range
- **Scatter Chart** to show the success rate for different payload masses.

Methods:
dash_core_components.Dropdown, plotly.express.pie,
dash_core_components_RangeSlider, plotly.express.scatter



Screenshot: SpaceX Launch Records Dashboard with Plotly Dash

https://github.com/pyo-bit/capstone_project_ibm/blob/95c53468d14c7d4c859071f952194c893a758551/7_interactive_visual_analytics_with_plotly_dashboard.py

Machine Learning

To predict if the first stage will land, we are creating a Machine Learning pipeline. To achieve this objective, we import the necessary libraries such as pandas, numpy, matplotlib.pyplot, seaborn, and scikit-learn. These libraries enable us to manipulate and analyze the data, standardize it, split it into training and test datasets, and implement various classification algorithms.

We load the dataset containing information about Falcon 9 rocket launches from SpaceX's website. The dataset includes features such as flight number, date, booster version, payload mass, orbit, launch site, outcome, flights, grid fins, reuse status, legs, landing pad, block, reuse count, serial number, longitude, latitude, and class. Next, we create a NumPy array from the "Class" column of the data and assign it to the variable "Y." Then, we standardize the data in the "X" variable using the preprocessing.StandardScaler() transform. We split the standardized data into training and test datasets using the train_test_split function, with a test size of 0.2 and a random state of 2.

For each classification algorithm (Logistic Regression, SVM, and Decision Tree), we create a GridSearchCV object to find the best hyperparameters. We specify the parameter grids to search for the best combination of hyperparameters. Finally, we evaluate the performance of each model by calculating the accuracy on the test dataset using the score method. We also plot the confusion matrix to visualize the model's performance in distinguishing between successful and unsuccessful landings.

https://github.com/pyo-bit/capstone_project_ibm/blob/4463bdc0fc67f62a78ad87b34b830f19c3f0d2fa/8_machine_learning_prediction_lab.ipynb

Predictive Analysis (Classification) & Results

Get & prepare data

- Load dataset into a NumPy array
- Normalize the data with 'StandardSaler'
- Split the data into training and test set

Prepare the models

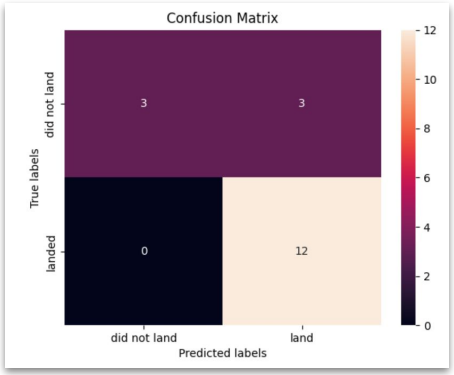
- Choose a ML algorithm
- Create a GridSearchCV object for parameters optimization
- Train the data with GridSearchModel with SVM(), DecisionTreeClassifier(), LogisticRegression (), and KNeighborsClassifier()

Evaluate the models

- Retrieve the best hyperparameters
- Using .score() to calculate the accuracy for all models
- Plot a Confusion Matrix

Result

- Compare the models regarding their accuracy
- Here: Decision Tree had highest accuracy (=0.877)



We used and compared multiple machine learning methods and presented the results in a dataframe.Finally, the we identified **Decision Tree** with an **accuracy = 0.877** as the best model.

https://github.com/pyo-bit/capstone_project_ibm/blob/4463bdc0fc67f62a78ad87b34b830f19c3f0d2fa/8_machine_learning_prediction_lab.ipynb

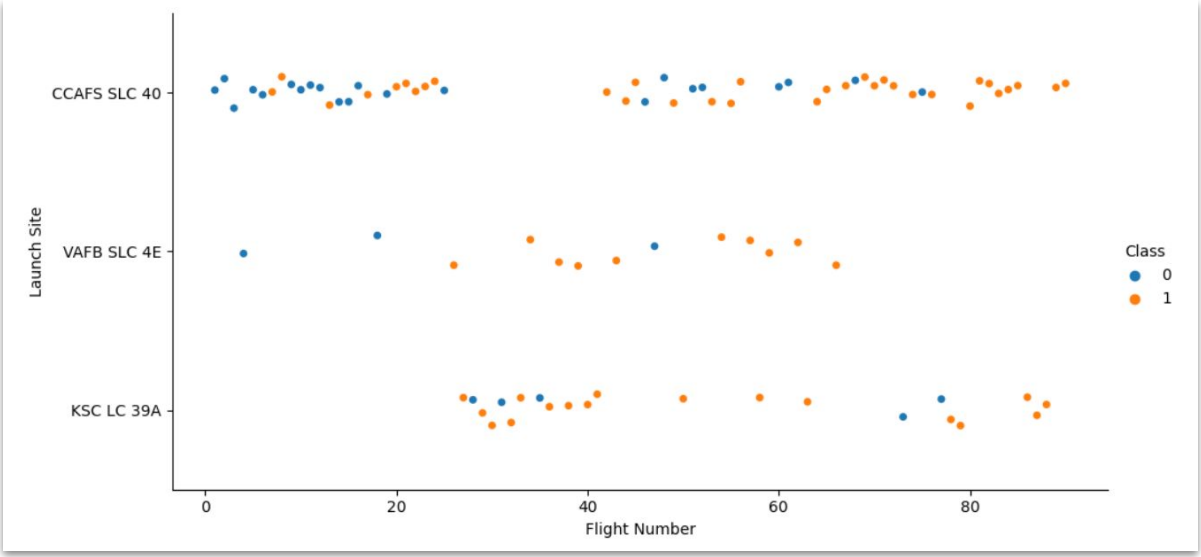
Section 2

Insights Drawn from EDA

Flight Number vs. Launch Site

Results:

- We can observe that the **success rate increases** over time.
- Errors are marked in **blue** (class=0) while successful landings are depicted in orange (class=1) .
- Approximately half of the launches took place from **CCAFS SLC-40**.
- Overall, **KSC LC-39A** and **VAFB SLC-4E** exhibit higher success rates compared to other launch sites.

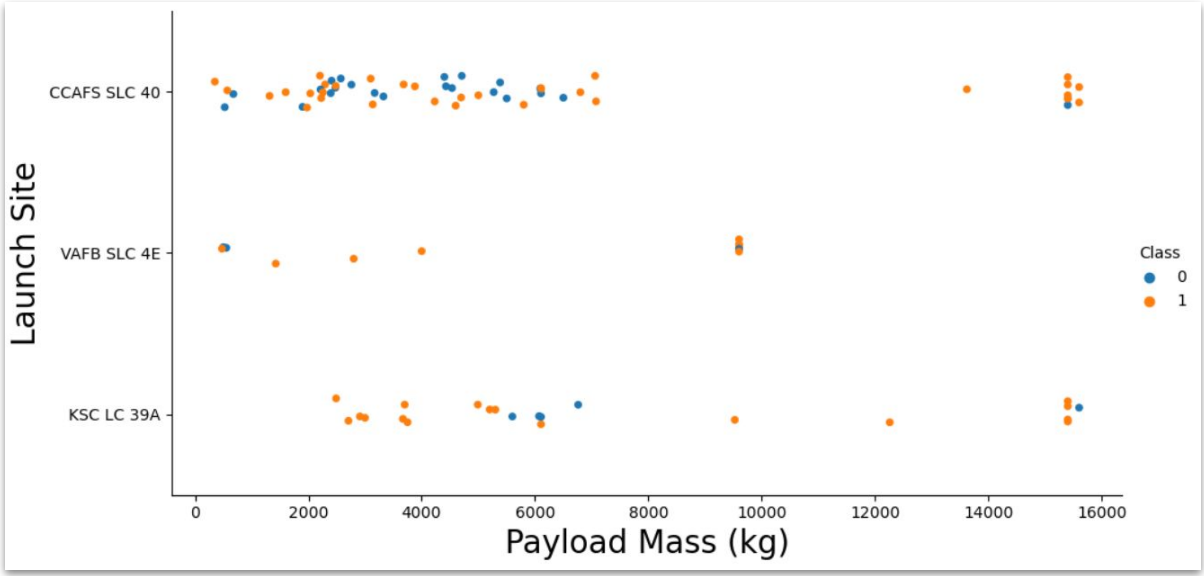


Scatter Plot: Flight Number vs. Launch Site

Payload vs. Launch Site

Results:

- The higher the payload, the higher the success rate; nearly all launches with a payload exceeding 7,000 kg were successful.
- **KSC LC-39A** maintains a 100% success rate for payloads exceeding 5,000 kg.

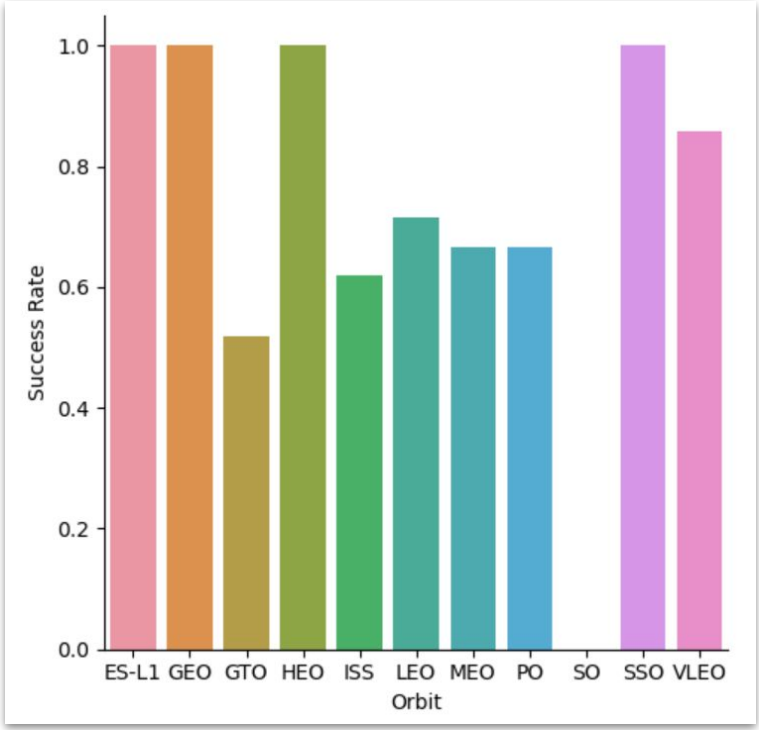


Scatter Plot: Payload vs. Launch Site

Success Rate vs. Orbit Type

Results:

- The orbit types ES-L1, GEO, HEO, and SSO exhibit a **100% success** rate.
- The orbit type SO, on the other hand, stands out with a **0% success rate**, being the only category with such a distinction.

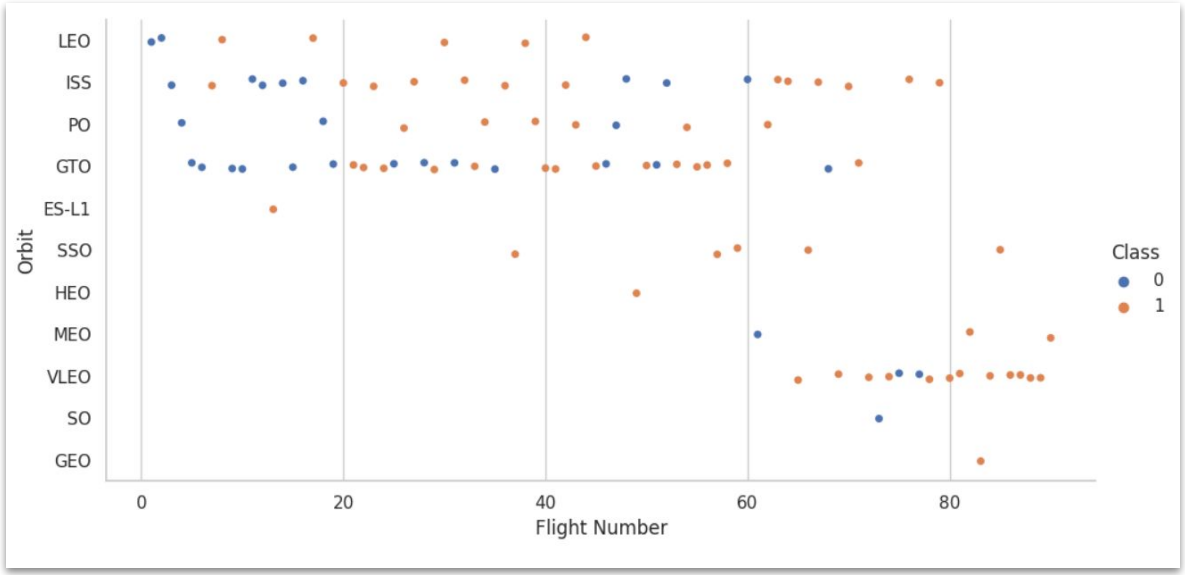


Bar Chart: Success Rate for each Orbit Type

Flight Number vs. Orbit Type

Results:

- The **success rate (orange)** **increases with the number of launches**.
- The **LEO orbit** demonstrates a particularly strong **positive correlation** between the number of **previous launches and the success rate**.
- **GTO** exhibits little to **no correlation** between the number of previous launches and the success rate.

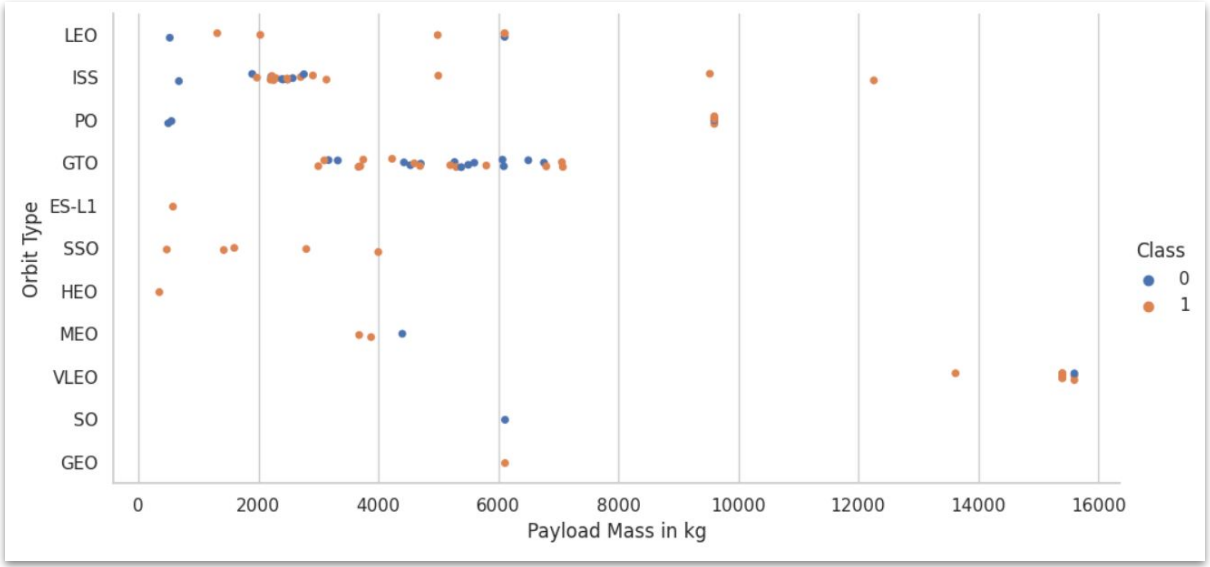


Scatter Plot: Flight Number vs. Orbit Type

Payload vs. Orbit Typ

Results:

- Payloads and **success rate (orange)** are positively related. Particularly, the **ISS, LEO, and PO** orbits benefit from **higher payloads**, showing an **increase in success rate**.
- However, **GTO deviates from this trend**, experiencing higher **failure rates (blue)** for higher payloads.

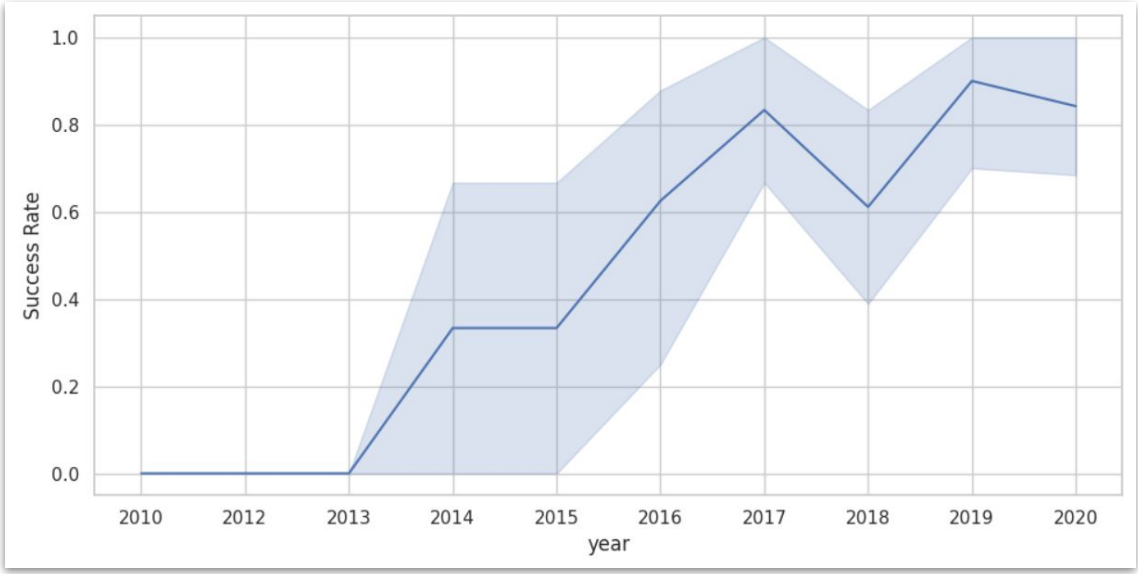


Scatter Plot: Payload vs. Orbit Type

Launch Success Yearly Trend

Results:

- Since 2013, the success rate has shown a **positive trend**. The only exceptions occurred between 2017 and 2018, as well as between 2019 and 2020, during which it experienced a slight decline.



Line Chart: yearly average success rate

All Launch Site Names

```
print(df["Launch_Site"].unique())
#Or: %sql SELECT Unique(LAUNCH_SITE) FROM SPACEXTBL;
```

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

Launch sites can be retrieved either from a dataframe or using an SQL command. When using SQL, we retrieve the LAUNCH_SITE column from the SPACEXTBL table, ensuring uniqueness with the addition of the UNIQUE keyword to eliminate duplicate launch sites.

Launch Site Names Begin with ‘KSC’

In [40]:
```
%sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE'KSC%' LIMIT 5;
```

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

Out[40]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcom
19/02/2017	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490.0	LEO (ISS)	NASA (CRS)	Success	Success (grour pa
16/03/2017	6:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	5600.0	GTO	EchoStar	Success	No attemp
30/03/2017	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300.0	GTO	SES	Success	Success (dror shi
05/01/2017	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300.0	LEO	NRO	Success	Success (grour pa
15/05/2017	23:21:00	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070.0	GTO	Inmarsat	Success	No attemp

We are performing an SQL query to retrieve and present five records from the SPACEXTBL table where the launch sites are identified by a string starting with 'KSC'. By employing the LIKE operator with the pattern 'KSC%', the query selectively matches launch sites beginning with 'KSC'.

The inclusion of the LIMIT 5 clause ensures that only a limited number of records are fetched.

Total Payload Mass

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

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE CUSTOMER = 'NASA (CRS)';
```

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

SUM(PAYLOAD_MASS__KG_)
45596.0

We are executing an SQL query to retrieve the total payload mass of boosters launched by NASA under the CRS (Commercial Resupply Services) program. The query filters the SPACEXTBL table based on the CUSTOMER column, specifically selecting records where the customer is 'NASA (CRS)'.

The resulting sum of the PAYLOAD_MASS__KG_ column is calculated and returned as the total payload mass, which amounts to 45,596.0 kilograms.

Average Payload Mass by F9 v1.1

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

```
# print(df[df["Booster_Version"] == "F9 v1.1"]["PAYLOAD_MASS__KG_"].mean())
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE BOOSTER_VERSION = 'F9 v1.1';
```

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

AVG(PAYLOAD_MASS__KG_)
2928.4

We are executing an SQL query to calculate and display the average payload mass carried by boosters of version F9 v1.1. The query filters the SPACEXTBL table based on the BOOSTER_VERSION column, specifically selecting records where the booster version is 'F9 v1.1'.

The resulting average payload mass, which amounts to 2928.4 kilograms, is calculated and returned.

First Successful Ground Landing Date

List the date where the succesful landing outcome in drone ship was acheived.

Hint:Use min function

```
%sql SELECT MIN(DATE) FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (drone ship)'
```

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

MIN(DATE)
04/08/2016

We are executing an SQL query to retrieve the earliest date on which a successful landing outcome was achieved on a drone ship. The query filters the SPACEXTBL table based on the condition specified in the WHERE clause, selecting records where the landing outcome is 'Success (drone ship)'. The MIN function is used to calculate and return the minimum (earliest) date from the matching records.

The result indicates that the successful landing on a drone ship occurred on April 8, 2016.

Successful Drone Ship Landing with Payload between 4000 and 6000

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

```
%sql SELECT PAYLOAD FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (ground pad)' AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 600
```

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

Payload
NROL-76
Boeing X-37B OTV-5
Zuma

We are executing an SQL query to retrieve the names of boosters that achieved success in ground pad landings and had a payload mass greater than 4000 kilograms but less than 6000 kilograms. The query filters the SPACEXTBL table based on the criteria specified in the WHERE clause, specifically matching records where the landing outcome is 'Success (ground pad)' and the payload mass falls within the range of 4000 to 6000 kilograms.

The resulting names of the boosters meeting these conditions are returned as 'Payload', which include NROL-76, Boeing X-37B OTV-5, and Zuma.

Total Number of Successful and Failure Mission Outcomes

We are executing an SQL query to obtain the total number of mission outcomes categorized as successful and failure. The query operates on the SPACEXTBL table and employs the GROUP BY clause to group the records based on the MISSION_OUTCOME column. The COUNT(*) function is used to calculate the total number of occurrences for each distinct mission outcome.

The results show that there are 898 missions with no specified outcome, 1 failure in flight, 98 successful missions, 1 ambiguous success, and 1 success with unclear payload status.

List the total number of successful and failure mission outcomes

```
%sql SELECT MISSION_OUTCOME, COUNT(*) as total_number \
FROM SPACEXTBL GROUP BY MISSION_OUTCOME;
```

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

Mission_Outcome	total_number
None	898
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

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

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

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

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

We are executing an SQL query to retrieve the names of booster versions that have carried the maximum payload mass. The query utilizes a subquery to find the maximum payload mass value from the SPACEXTBL table. The main query then selects the booster versions from the table where the payload mass matches the maximum value obtained from the subquery.

The result displays the names of the booster versions that achieved this maximum payload mass, such as F9 B5 B1048.4, F9 B5 B1049.4, F9 B5 B1051.3, and others.

Landing Outcomes in 2017

List the records which will display the month names, succesful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017

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)='2017' for year.

```
%sql SELECT substr(Date,4,2) as month, DATE,BOOSTER_VERSION, LAUNCH_SITE, [LANDING_OUTCOME] \
FROM SPACEXTBL where [Landing_Outcome] = 'Success (ground pad)' and substr(Date,7,4)='2017' \
ORDER BY month;
```

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

month	Date	Booster_Version	Launch_Site	Landing_Outcome
01	05/01/2017	F9 FT B1032.1	KSC LC-39A	Success (ground pad)
02	19/02/2017	F9 FT B1031.1	KSC LC-39A	Success (ground pad)
03	06/03/2017	F9 FT B1035.1	KSC LC-39A	Success (ground pad)
07	09/07/2017	F9 B4 B1040.1	KSC LC-39A	Success (ground pad)
08	14/08/2017	F9 B4 B1039.1	KSC LC-39A	Success (ground pad)
12	15/12/2017	F9 FT B1035.2	CCAFS SLC-40	Success (ground pad)

We are executing an SQL query to retrieve records that display the month names, dates, booster versions, launch sites, and successful landing outcomes in ground pad for the months in the year 2017. As SQLite does not have a built-in month name function, we are using the substr(Date, 4, 2) expression to extract the month from the date column. Additionally, we filter the results to only include records with a landing outcome of 'Success (ground pad)' and where the year portion of the date is '2017' using substr(Date, 7, 4)='2017'.

The result is ordered by the month column, showing the month, date, booster version, launch site, and landing outcome for each record.

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

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

```
%sql SELECT [Landing_Outcome], count(*) as count_outcomes FROM SPACEXTBL \
WHERE DATE between '04-06-2010' and '20-03-2017' group by [Landing_Outcome] \
order by count_outcomes DESC;
```

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

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

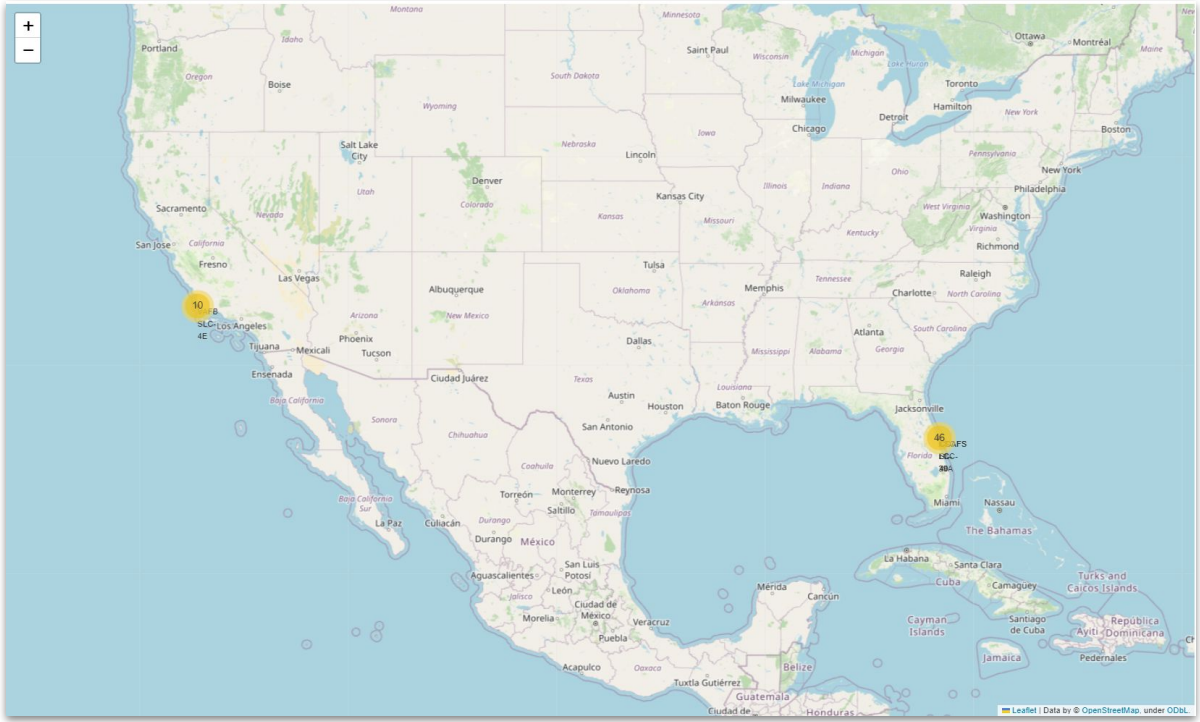
We are executing an SQL query to rank the count of successful landing outcomes between the dates '04-06-2010' and '20-03-2017' in descending order. The query operates on the SPACEXTBL table and uses the WHERE clause to filter the records based on the date range specified. The COUNT(*) function is used to calculate the total count of each landing outcome category, and the results are grouped by the landing outcome. The ORDER BY clause is applied to sort the results in descending order based on the count of outcomes.

The output displays the landing outcomes and their corresponding counts, with the most frequent successful landing outcomes appearing first.

Section 3

Launch Sites Proximities Analysis

Folium Map Launch Site

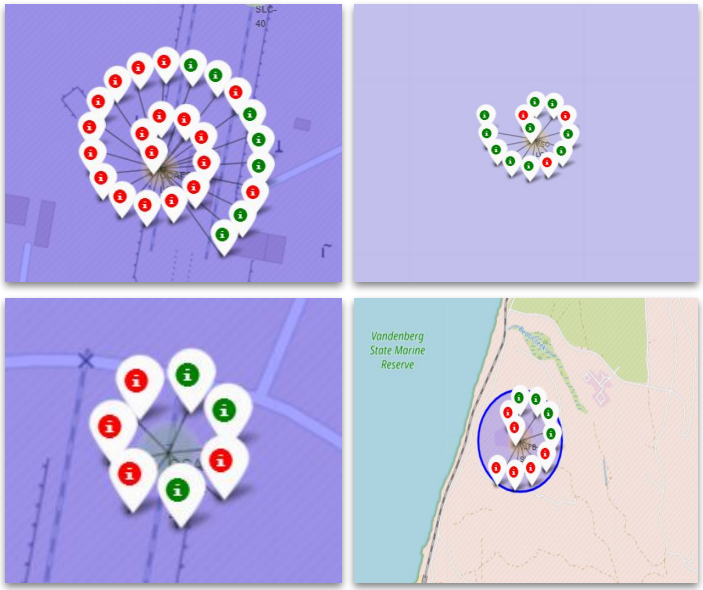


Map with Folium: showing all Falcon 9 launch sites.

We have marked the launch sites, both on the east and west coast of the U.S.A., **yellow**.

The launch sites are close to the equator because this offers advantages such as increased orbital velocity, efficient energy use, safety considerations, and accessibility to various orbital inclinations.

Folium Map Markers for Launch Success



Map with Folium: launch success by launch site

WIn the visualization provided, we can observe the launch outcomes categorized by different launch sites. Successful launches are represented by **green markers**, while failed launches are depicted by **red markers**.

For instance, upon examination, it is evident that CCAF SLC-40 exhibits a success rate of 42.9%. This information offers insights into the performance of each launch site in terms of successful mission outcomes.

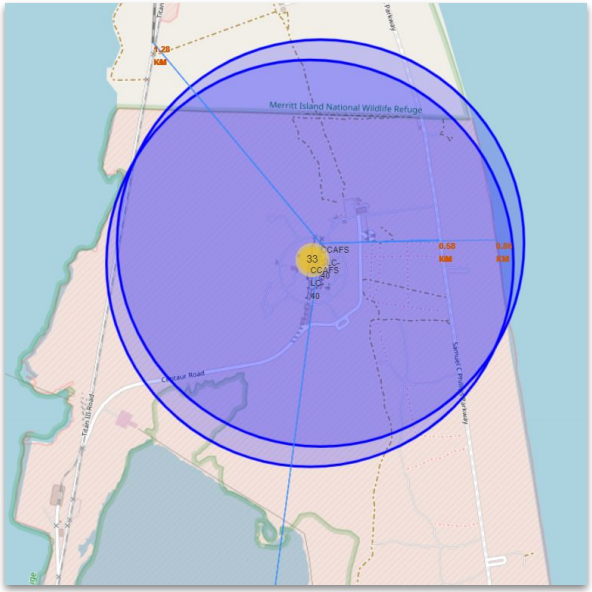
Distances to Proximities

Using the visualization capabilities of Folium, we are able to draw lines between coordinates and measure distances. As a result, we have determined that CCAFS SLC-40 is located:

- 21.96 km away from the nearest railway line,
- 0.86 km from the coastline,
- 23.23 km from the nearest city, and
- 26.88 km from the nearest highway.

This information provides valuable insights into the geographical positioning of CCAFS SLC-40, highlighting its distances from key landmarks and transportation infrastructure.

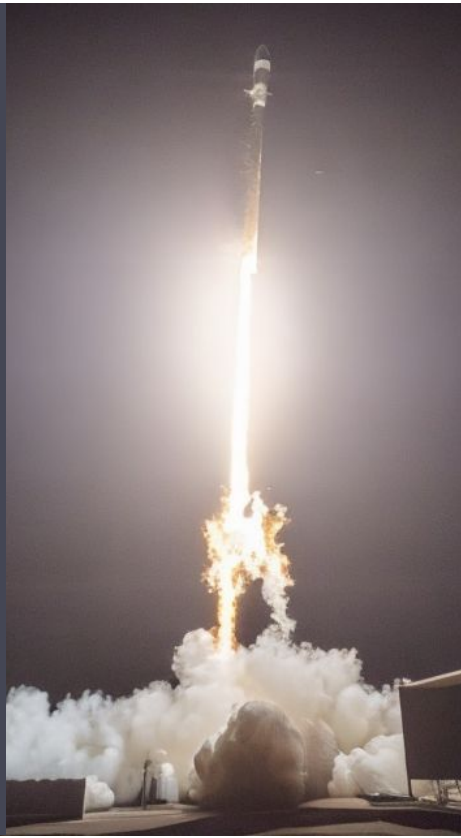
These distances inform logistical planning, safety considerations, and impact assessment, ensuring effective and responsible launch operations.



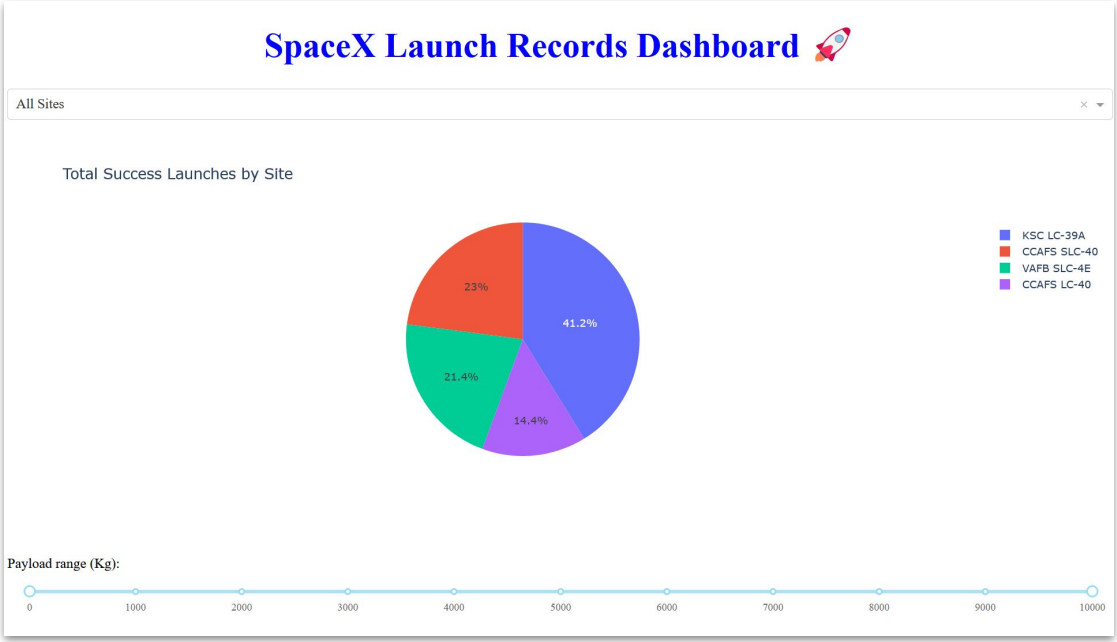
Map with Folium: drawing lines to proximities likes railroads and cities

Section 4

Build a Dashboard with Plotly Dash



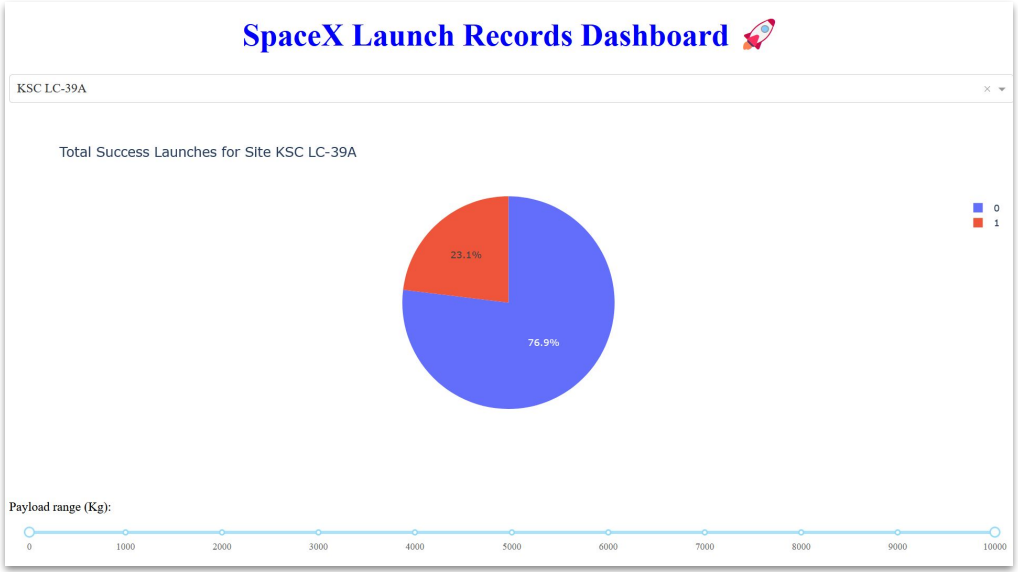
Launch Success Rates by Site



Result:

KSC LC-39 boasts the highest success rate among all launch sites.

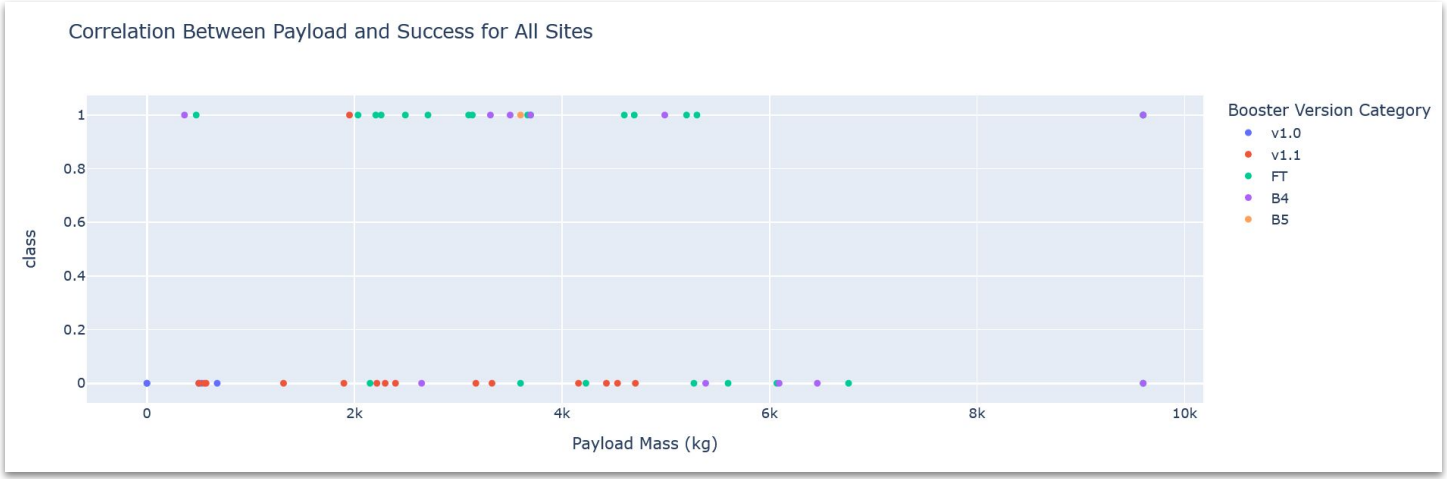
Launch Success Rate of KSC LC-39A



Result:

KSC LC-39A holds the highest success rate at 76.9%, while the failure rate stands at a significantly lower 23.1%.

Launch Success Rate: Payload vs. Mass



Result:

KSC LC-39A holds the highest success rate at 76.9%, while the failure rate stands at a significantly lower 23.1%.From this graph, we can infer that payloads up to 5000 have a higher success rate.

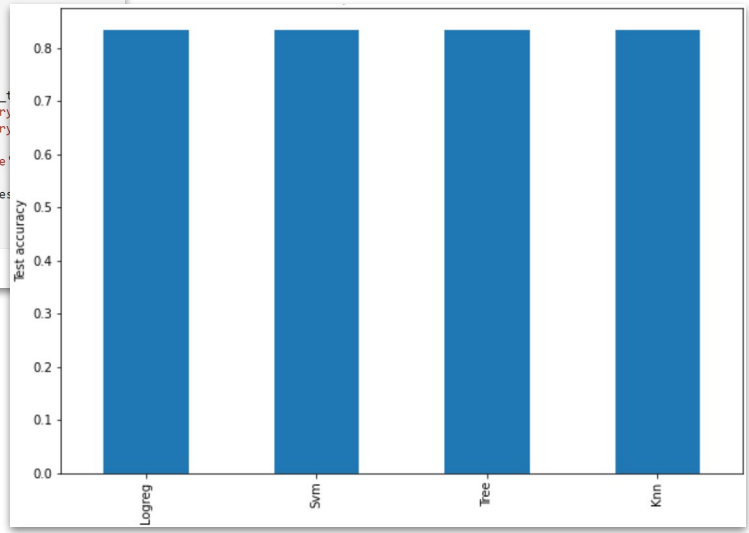
Section 5

Predictive Analysis (Classification)

Classification Accuracy

```
from sklearn.metrics import f1_score, jaccard_score
#Calculationg accuracy
accuracy = [svm_cv_score, logreg_score, knn_cv_score, tree_cv_score]
accuracy = [i * 100 for i in accuracy]
#Comparing methods
method = ['SVM', 'Logistic Regression', 'K Nearest', 'Decision Tree']
models = {'ML Method':method, 'Accuracy Score (%)':accuracy}
#Show results
results=pd.DataFrame(models)
print("ML method accuracy: ")
results
f1_scores =[f1_score(Y_test, yhat, average='binary'),f1_score(Y_test, yhat, average='binary'), f1_score(Y_t
jaccard_scores = [jaccard_score(Y_test, yhat, average='binary'),jaccard_score(Y_test, yhat, average='binary
                  jaccard_score(Y_test, yhat, average='binary'),jaccard_score(Y_test, yhat, average='binary
results2 = [logreg_score, svm_cv_score, tree_cv_score, knn_cv_score]
results3 = pd.DataFrame(np.array([f1_scores, jaccard_scores, accuracy]), index=['Jaccard_Score', 'F1_Score'
results3
models = {'Decision Tree':tree_cv.best_score_,'Logistic Regression':logreg_cv.best_score_,'SVM': svm_cv.bes
results4=max(models, key=models.get)
print("Best Model: ",results4," with ",models[results4])
```

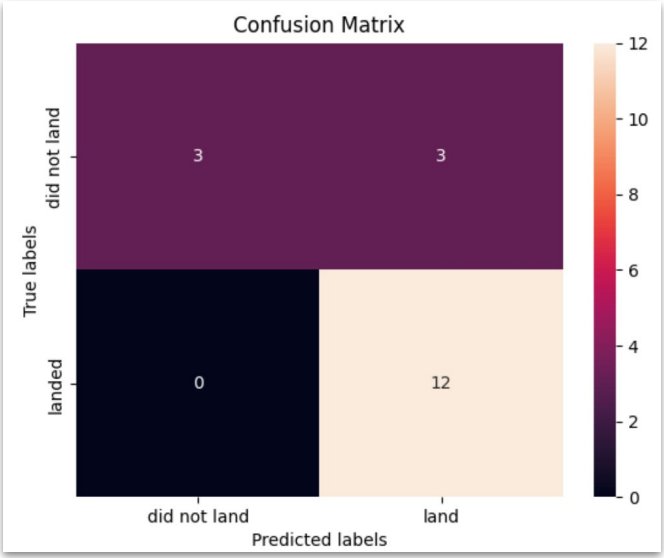
```
ML method accuracy:
Best Model:  Decision Tree  with  0.8767857142857143
```



Result:

The performance of the ML models is consistently high, with an accuracy of over 80%. Among the models, the Decision Tree performs the best, achieving an accuracy of 87.7% in our case.

Confusion Matrix of Best Performing Model



Here, we plot the confusion matrix by predicting the labels (yhat) using the tree_cv model and passing the true labels (Y_test) to the plot_confusion_matrix function.

Result:

We can see that 3 “did not land” and 12 “land” were predicted correctly. 3 true labels “did not land” were predicted incorrectly.

Conclusion

This project aims to uncover factors that influence the success or failure of a Stage One landing. Based on the data provided by the SpaceX REST API and Wikipedia, the following observations can be made:

- The success rate has significantly improved over time.
- KSC LC-39A is the launch site with the highest success rate. It achieves a 100% success rate for payloads below 5000 kg.
- The ES-L1, GEO, HEO, and SSO orbits have a 100% success rate.
- Higher or lower payloads lead to higher success rates in different cases. Further investigation could be conducted to explore the relationship between payload, velocity, and altitude for specific orbits.

Furthermore:

- The dataset does not include the latest data from the 2020s.
- The data does not explain why some launch sites have higher success rates than others.
- The various ML models perform similarly, but the Decision Tree model is preferred in this case due to its slight advantage.

Thank you!

IBM – Data Science and Machine Learning Capstone Project

Björn Thomsen, M.A.

June 18th, 2023

Appendix

Github Repository Master:
https://github.com/pyo-bit/capstone_project_ibm/tree/master

Presentation:
https://github.com/pyo-bit/capstone_project_ibm/blob/02f3c41d38709b9f052b6ec686f440b074f91d43/Predicting%20SpaceX%E2%80%99s%20Falcon%209.pdf

Jupyter Files by Chapter:
https://github.com/pyo-bit/capstone_project_ibm/blob/f493b9fae223f972442dbea99f71d3b1f59bf446/1_jupyter-labs-spacex-data-collection-api.ipynb

https://github.com/pyo-bit/capstone_project_ibm/blob/0f3450907c80ca74864883993a817de4fb546f47/2_jupyter-labs-webscraping.ipynb

https://github.com/pyo-bit/capstone_project_ibm/blob/0a74185c8558a6a2ee6bc4e0298dc851a34ae331/3_jupyter-labs-spacex-data-wrangling.ipynb

https://github.com/pyo-bit/capstone_project_ibm/blob/b8bc810db86c6e32ffd626de11127e24dd33c2ca/4_jupyter-labs-eda-sql-edx_sqllite.ipynb

https://github.com/pyo-bit/capstone_project_ibm/blob/03be254c6c0abf3ac1d98271b14aeed21668e416/5_jupyter_lab_eda_with_visualization.ipynb

https://github.com/pyo-bit/capstone_project_ibm/blob/3d4de4c5054ee0c57d9ff7e6126a68288157dc12/6_interactive_visual_analytics_with_folium.ipynb

https://github.com/pyo-bit/capstone_project_ibm/blob/95c53468d14c7d4c859071f952194c893a758551/7_interactive_visual_analytics_with_plotly_dashboard.py

https://github.com/pyo-bit/capstone_project_ibm/blob/4463bdc0fc67f62a78ad87b34b830f19c3f0d2fa/8_machine_learning_prediction_lab.ipynb

Additional Files:
https://github.com/pyo-bit/capstone_project_ibm/blob/4463bdc0fc67f62a78ad87b34b830f19c3f0d2fa/spacex_launch_dash.csv