IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

IBM Data Science Capstone Project
Sebastian Motsch
7th January 24

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Open Questions
- Outlook and Proposals
- Appendix

Executive Summary

- **Summary of methodologies**
  - Data Collection through APIs
  - Automated Data Collection with Web Scraping
  - Data cleaning and Wrangling
  - Exploratory Data Analysis (EDA) using SQL
  - EDA with Advanced Data Visualization techniques
  - Interactive Visual Analytics using Folium for Geospatial Data
  - Machine Learning Predictive Modeling and Evaluation for Each Classification Algorithm
- **Summary of all results**
  - Exploratory Data Analysis Results Including Interactive Visualizations
  - Screenshots of Interactive Analytics
  - Predictive Analytics Results and Conclusion of the Best Prediction Model

Introduction

- ## Project background and context

  The capstone's main objective is to use machine learning to predict the successful landing of the Falcon 9's first stage and to understand driving success parameters.

  With Space X charging $62 million for launches, which is more than 60% less than competitors' $165 million, the key to this cost advantage is reusing the first stage.

  A reliable prediction model can clarify launch costs and offer a competitive benchmark for other companies. Based on identified driving success parameters, an outlook is given.

- ## Problems you want to find answers

  Which factors influence the successful landing of the first stage of a rocket?

  How do interrelated parameters affect the likelihood of a successful landing?

  What are the optimal operational conditions required for the first stage to land successfully?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX Rest API and by web scraping from Wikipedia
- Perform data wrangling
  - One-hot encoding was applied to categorical features,
    and irrelevant data was either excluded or replaced.
- Perform exploratory data analysis (EDA) using visualization and SQL
  to identify patterns, which were illustrated through graphs.
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Predictive analysis was executed using several classification models,
    focusing on how to build, tune, and evaluate each model effectively.

Data Collection

- Data was sourced through multiple methods:
  - Utilized a GET request to the SpaceX API.
  - Decoded the API response using the .json() function and converted it to a pandas dataframe with .json_normalize().
  - Web scraping from Wikipedia was done for Falcon 9 launch records using BeautifulSoup.
  - The objective of scraping was to:
    - Extract Falcon 9 launch records from an HTML table.
    - Parse the table data and convey it to a pandas dataframe for further analysis.
  - Cleaned the data:
    - Checked for missing values.
    - Filled in missing values as needed.
  - Filtered the dataframes based on specific requirements.
- Exported the cleaned and filtered data to a CSV file.

Data Collection – SpaceX API

- Data was gathered and collected through a GET request from the SpaceX API, and the retrieved data was then cleaned, completed, and formatted using basic and fundamental data wrangling techniques.
- The link to the notebook is https://github.com/sebnotch/presi-and-code-data-science/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Send a GET request to retrieve rocket launch data using the API.

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

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

        # to look at it in human readable form..
        response.json()
```

Use the `json_normalize` method to convert the JSON result into a DataFrame.

```
In [9]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-Skills
```

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

```
In [10]: response.status_code
Out[10]: 200
```

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

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

Perform data cleaning and fill in the missing values.

```
In [36]: # Calculate the mean for the PayloadMass column
         payload_mass_mean = data_falcon9['PayloadMass'].mean()

         # Replace np.nan values in the PayloadMass column with the mean
         data_falcon9['PayloadMass'].replace(np.nan, payload_mass_mean, inplace=True)
```

Data Collection – Web Scraping

- Web scraping with BeautifulSoup was utilized to extract Falcon 9 launch records, and the HTML table was subsequently parsed and transformed into a pandas DataFrame.
- The link to the notebook is https://github.com/sebnotch/presi-and-code-data-science/blob/main/jupyter-labs-webscraping.ipynb

1. Perform an HTTP GET request to retrieve the Falcon 9 Launch HTML page.

```
In [4]: static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

```
In [5]: # use requests.get() method with the provided static_url
        response = requests.get(static_url)
        # assign the response to a object
        if response.status_code == 200:
            print("Request was successful! response.status_code == 200 confirmed")
            content = response.content
        else:
            print(f"Failed to retrieve the page. Status code: {response.status_code}")

        Request was successful! response.status_code == 200 confirmed
```

2. Create a BeautifulSoup object from the HTML response

```
In [6]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
        soup = BeautifulSoup(response.text, 'html.parser')
```

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

```
In [7]: # Use soup.title attribute
        print(soup.title.string)

        List of Falcon 9 and Falcon Heavy launches - Wikipedia
```

```
In [17]: soup.title
Out[17]: <title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

3. Extract column names from the HTML table header.

```
first_launch_table = html_tables[2]

# Find all 'th' elements in the first table
table_headers = first_launch_table.find_all('th')

column_names = []

# Iterate through each 'th' element
for th in table_headers:
    # Extract column name using the provided function
    name = extract_column_from_header(th)
    #column_names.append(name)
    #Append non-empty column names to the list
    if name is not None and len(name) > 0:
        column_names.append(name)

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
```

4. Parse the launch HTML tables to create a DataFrame.

5. Export the DataFrame to a CSV file.

Data Wrangling

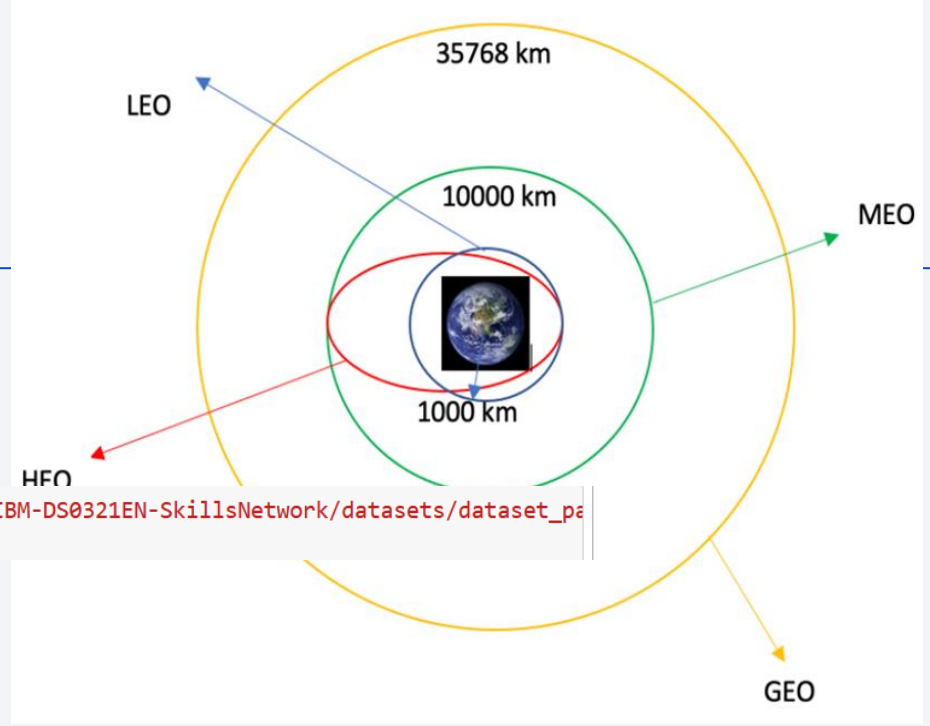
Basic steps of data wrangling applied

1. Import Libraries and Define Auxiliary Functions, load data

```
In [2]: df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_pa
        df.head(10)
```

2. Conduct Data Analysis
3. Load the SpaceX dataset
4. Create a concise, cleaned DataFrame, simplifying to Boolean values where sensible, and understand numerical and categorical columns
    a) Calculate the number of launches at each site
    b) Determine the number and occurrence of each orbit
    c) Calculate the number and occurrence of mission outcomes for the orbits
    d) Generate a landing outcome label from the Outcome column
5. Export the dataset to a flat file as a CSV file

- The link to the notebook is https://github.com/sebnotch/presi-and-code-data-science/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb



a)
```
In [5]: # Apply value_counts() on column LaunchSite
        launch_counts = df['LaunchSite'].value_counts()
```

b)
```
In [7]: # Apply value_counts on Orbit column
        orbit_counts = df['Orbit'].value_counts()

        print(orbit_counts)
```

c)
```
In [8]: # landing_outcomes = values on Outcome column
        landing_outcomes = df['Outcome'].value_counts()

        print(landing_outcomes)
```

d)
```
In [9]: for i,outcome in enumerate(landing_outcomes.keys()):
            print(i,outcome)
```

Exploratory Data Analysis (EDA) with Data Visualization

Data was explored by visualizing

- (Task 0) Flight number vs. payload mass - Scatter point chart (scatter plot)
- (Task 1) Visualizing the relationship between flight number and launch site - Scatter point chart
- (Task 2) Payload and launch site - Scatter point chart
- (Task 3) Success rate of each orbit type - Bar chart
- (Task 4) Flight number and orbit type - Scatter point chart
- (Task 5) Payload and orbit type - Scatter point chart
- (Task 6) Yearly trend of launch success - Line plot chart (line graph)

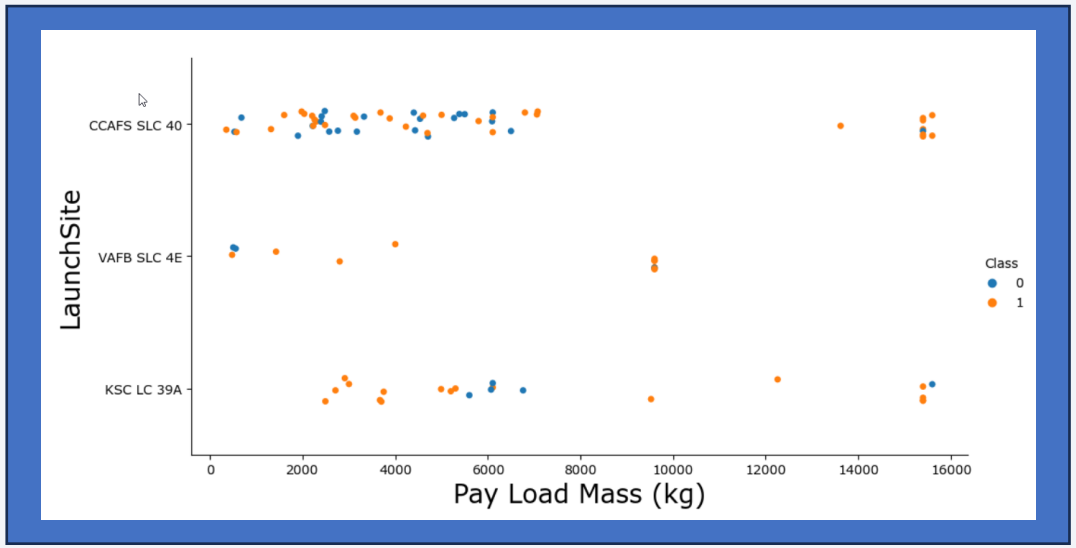
The link to the notebook is https://github.com/sebnotch/presi-and-code-data-science/blob/main/jupyter-labs-eda-dataviz.ipynb

Scatter plots are valuable for visualizing correlations between parameters. This assists in identifying relevant features for predictive modeling. In this case, scatter plots help to estimate the likelihood of success, taking into account both outcomes and landings.

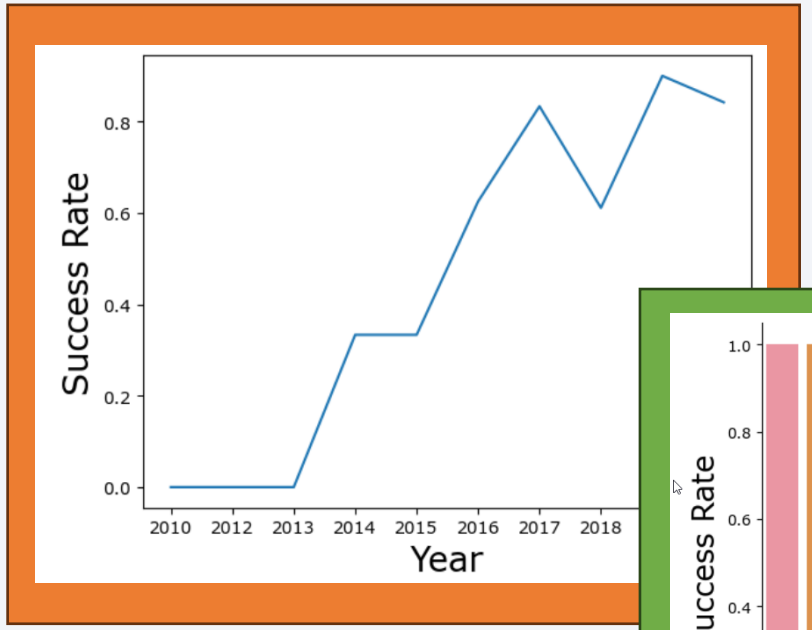
Bar graphs, when applied to preselected attributes, demonstrate the interrelationships among them and indicate which orbits are associated with a higher success rate.

Line graphs are beneficial for tracing trends, such as the annual trend in success rates.

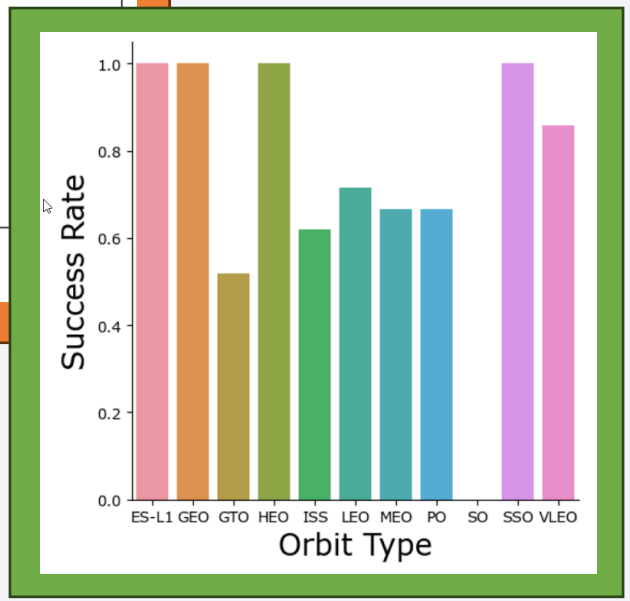
EDA with Data Visualization – continued, depicted



Example **scatter plot**



Example **line graphs**



Example **bar graph**

The link to the notebook is
https://github.com/sebnotch/presi-and-code-data-science/blob/main/jupyter-labs-eda-dataviz.ipynb

Exploratory Data Analysis (EDA) with SQL queries

Examples for SQL queries

- %sql SELECT * FROM SPACEXTABLE LIMIT 5;
- %sql SELECT DISTINCT "LAUNCH_SITE" FROM SPACEXTBL;
- %sql SELECT * FROM SPACEXTABLE WHERE "LAUNCH_SITE" LIKE 'KSC%' LIMIT 5;
- %sql SELECT SUM("PAYLOAD_MASS__KG_") FROM SPACEXTABLE WHERE "Customer" LIKE 'NASA (CRS)';
- %sql SELECT AVG("PAYLOAD_MASS__KG_") FROM SPACEXTABLE WHERE "Booster_Version" LIKE 'F9 v1.1';
- %sql SELECT MIN("DATE") FROM SPACEXTBL WHERE "LANDING__OUTCOME" LIKE "Success (drone ship)" ;
- %sql SELECT "Booster_Version" FROM SPACEXTBL \
  WHERE "LANDING__OUTCOME_" LIKE "Success (drone ship)" \
  AND "PAYLOAD_MASS__KG_" BETWEEN 4000 AND 6000 ;
- %sql SELECT MISSION_OUTCOME, COUNT(*) as total_number \
  FROM SPACEXTBL \
  GROUP BY MISSION_OUTCOME;
- %sql SELECT BOOSTER_VERSION \
  FROM SPACEXTBL \
  WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
- %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';
- %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;

The link to the notebook is https://github.com/sebnotch/presi-and-code-data-science/blob/main/jupyter-labs-eda-sql-edx_sqllite(1)_as_done_on_skills_network.ipynb

EDA with SQL continued - queries explained

With SQL, EDA was conducted to get insights from the data. Queries were formulated to determine:

- The unique launch site names used in the space mission.
- Display 5 records where launch sites begin with the string 'KSC'
- The total payload mass carried by boosters launched for NASA (CRS).
- The average payload mass carried by the booster version F9 v1.1.
- List the date where the first successful landing outcome in drone ship was achieved.
- List the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster versions which have carried the maximum payload mass.
- List the records which will display the month names, successful landing outcomes in ground pad, booster versions, launch site for the months in year 2017
- 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

The link to the notebook is https://github.com/sebnotch/presi-and-code-data-science/blob/main/jupyter-labs-eda-sql-edx_sqllite(1)_as_done_on_skills_network.ipynb

Build an Interactive Map with Folium

Folium provides Python-based tools for creating interactive maps:

- Launch sites are pinpointed on the map with symbols that indicate the success or failure of each launch.
- Launch outcomes are categorized as '0' for failure and '1' for success.
- Color-coded marker clusters are used to identify launch sites with higher success rates.
- The calculated distances address the following:
  - The proximity of launch sites to railways, highways, and coastlines.
  - The maintenance of a certain distance between launch sites and cities.

The link to the notebook is https://github.com/sebnotch/presi-and-code-data-science/blob/main/lab_jupyter_launch_site_location.jupyterlite_SOLVED.ipynb

Build an Interactive Map with Folium - continued

Map Objects	Code	Purpose of adding those objects
Circle marker	Folium.Circle	add a highlighted circle
Map marker	Folium.Marker	Mapping object for marking on the map
Marker cluster object	Marker_cluster	Grouping points of interest which are close to each other
Polyline	Folium.PolyLine	Drawing a line between points.
Icon Marker	Folium.Icon	Create an icon on the map.

The link to the notebook is https://github.com/sebnotch/presi-and-code-data-science/blob/main/lab_jupyter_launch_site_location.jupyterlite_SOLVED.ipynb

Build a Dashboard with Plotly Dash

To visualize and analyze launch data, a dashboard was constructed using Plotly Dash. The development process covers the following steps:

- Developed an interactive dashboard using Plotly Dash for dynamic data exploration.

- Integrated a Launch Site Dropdown Component
  to enable selection of individual launch sites or the option to view the data of all sites.
- Generated pie charts to show the total number of launches by each site,
  enabling understanding of the success and failure rates, such as payload mass.

- Implemented a Payload Range Slider, allowing users to filter launch data based on specified ranges of payload.
- Created a scatter plot to show the relationship between launch outcomes and payload for various booster versions.
  This ebables to understand how booster types affect associated to success rates.

The link to the notebook is https://github.com/sebnotch/presi-and-code-data-science/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

1. Loaded the data with numpy and pandas.
2. Transformed data into NumPy arrays.
3. Divided the dataset into training and testing sets.
4. Developed multiple machine learning models.
5. Tuned model hyperparameters using GridSearchCV and trained the models.

Data Preparation and Model Development

6. Measured model performance using accuracy.
7. Applied feature engineering to improve model performance.
8. Determine optimum hyperparameters for each algorithm
9. Fine-tuned algorithms for better results.
10. Plot the confusion matrix

Model Evaluation and Model Improvement

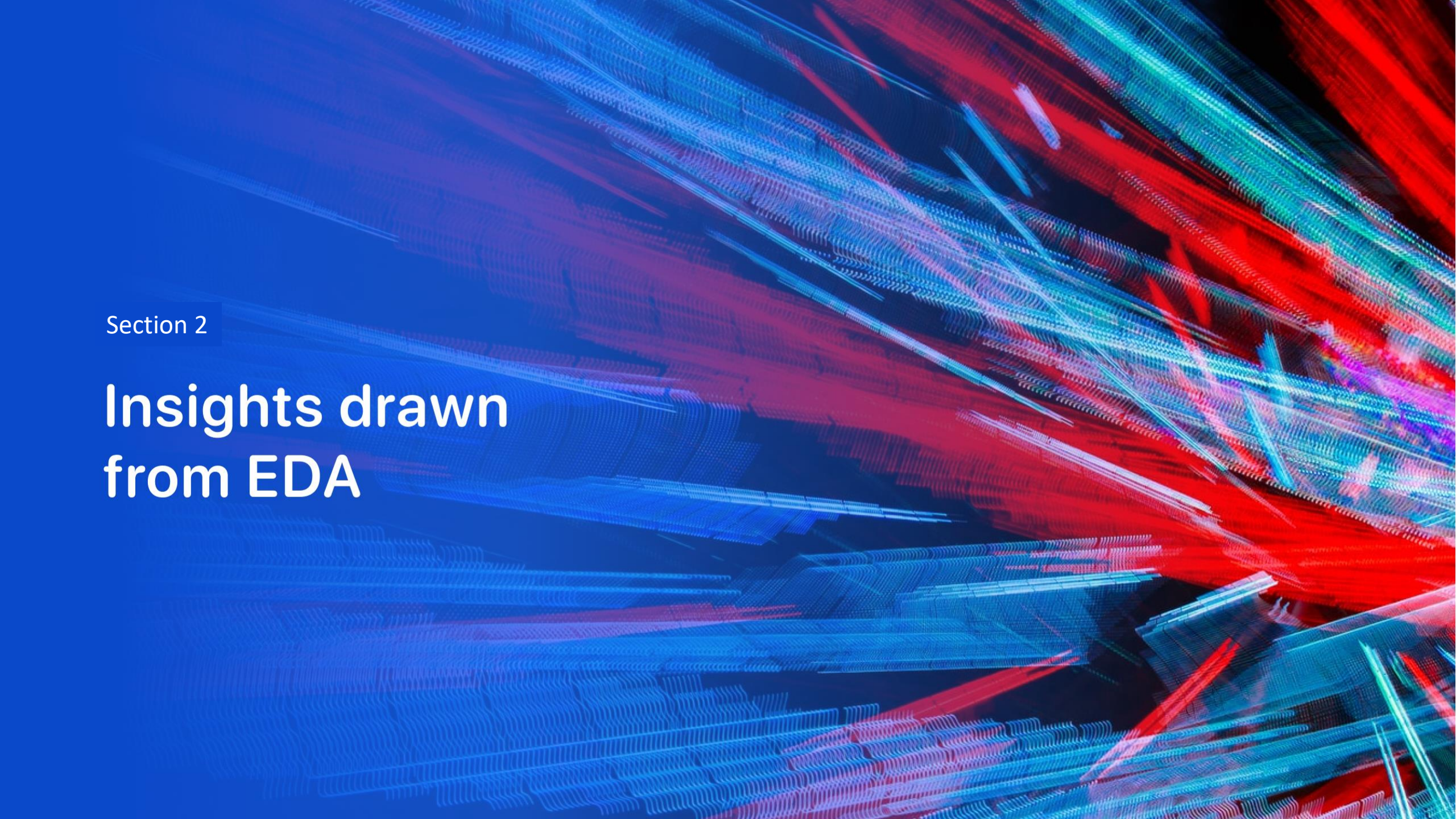
11. Identified the classification model with the highest accuracy.

Final Model Selection

The link to the notebook is https://github.com/sebnotch/presi-and-code-data-science/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

- Exploratory data analysis results
  - Patterns within the data were identified
  - There are clearly success driving parameters
- Interactive analytics demo in screenshots
  - Visualization is an important part of the approach
- Predictive analysis results
  - Predictions could clearly be established at a high confidence rate


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

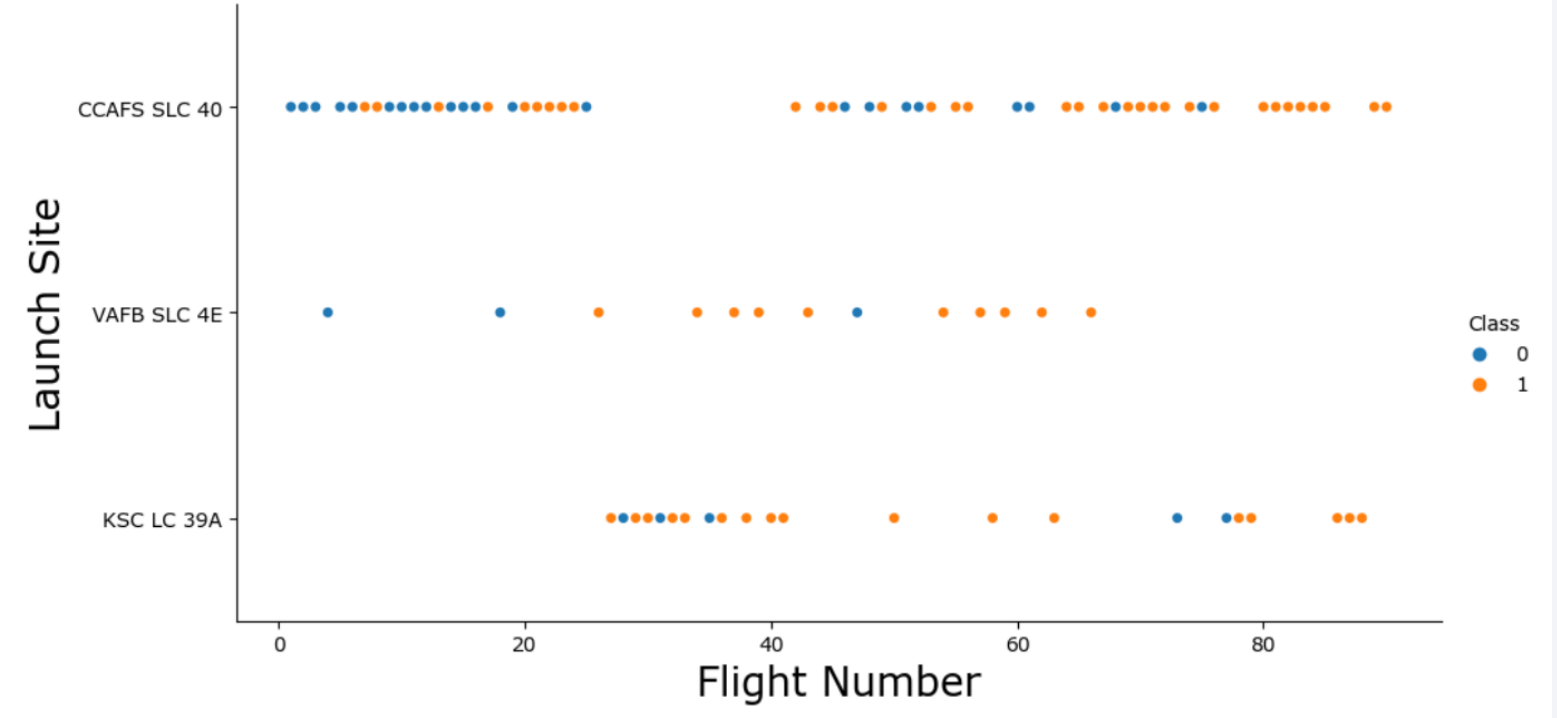


Diagram: scatter plot of Flight Number vs. Launch Site

Conclusion: ➔ Initial flights show a higher incidence of failure, whereas later flights generally achieve more success, independent of the launch site

Payload vs. Launch Site

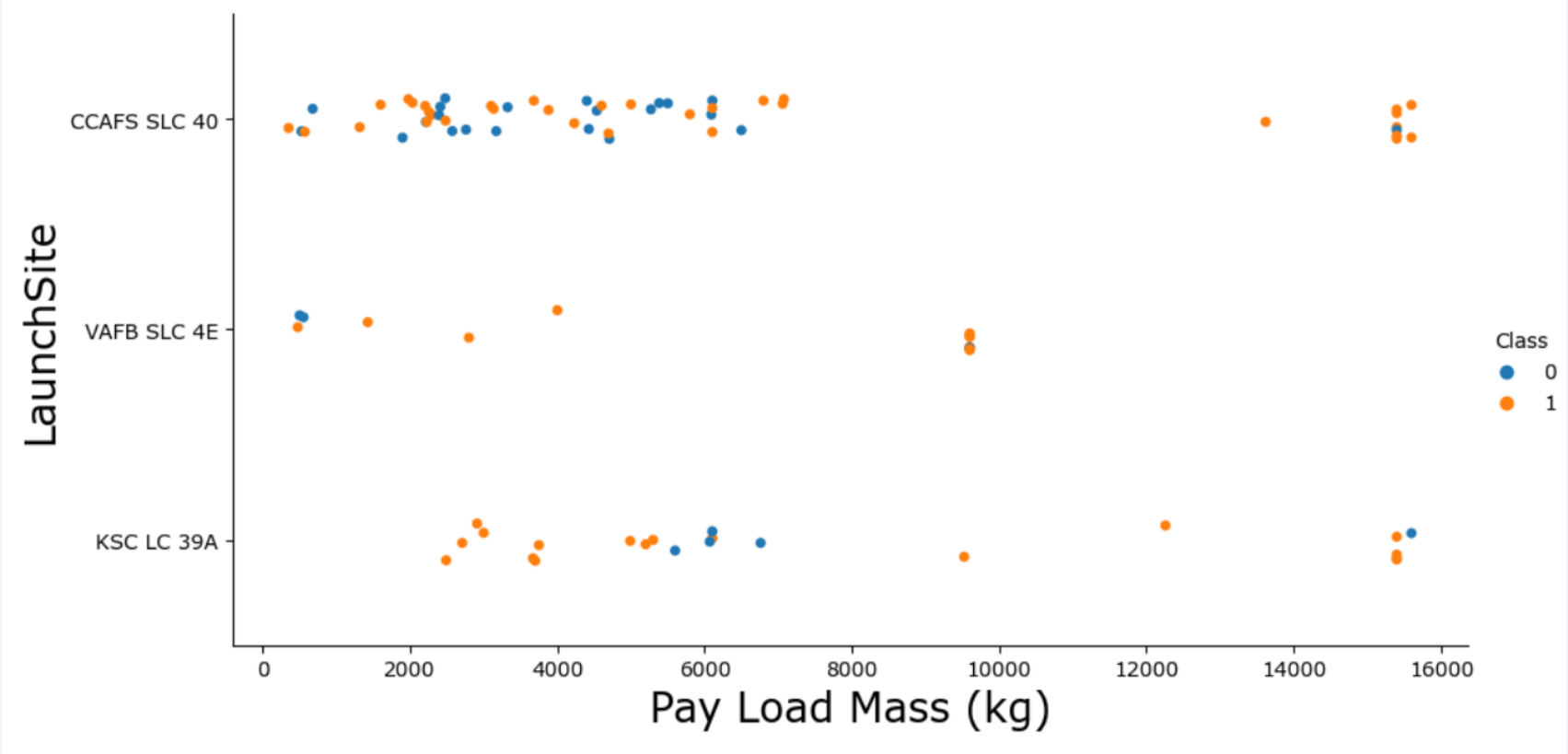


Diagram: scatter plot of Pay Load vs. Launch Site

Conclusion: ➔ Conclusion: Larger payloads usually lead to more successful launches, regardless of where they launch from. However, the CCAFS SLC 40 site shows the highest success rates with heavy payloads.

Success Rate vs. Orbit Type

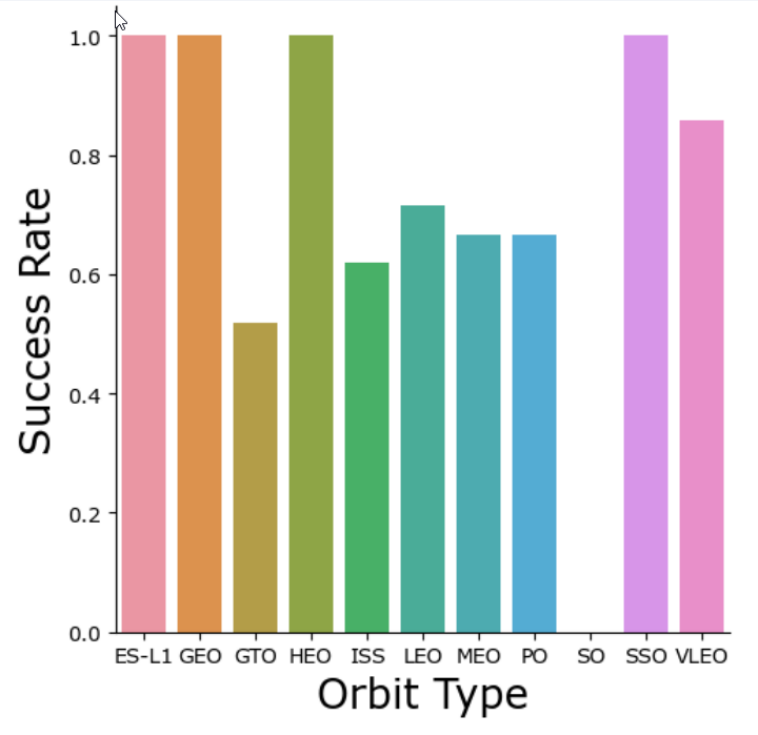


Diagram: bar chart of Success Rate vs. Orbit Type

Conclusion: ➔ ES-L1, GEO, HEO, SSO have biggest success rate, followed by VLEO

Flight Number vs. Orbit Type

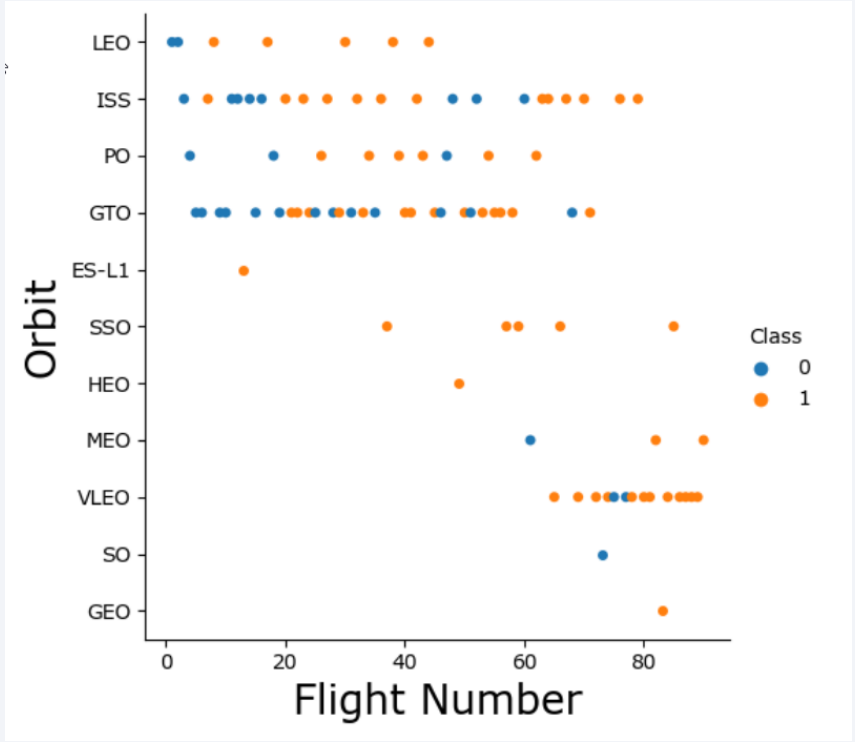


Diagram: scatter plot of Orbit Type vs Flight Number

Conclusion

1.There is a continuous increase in successful launches for Low Earth Orbit (LEO) without exception.

2.Across all launch sites, there is a trend of improvement in the success rate of launches over time.

3.For specific orbits, namely ES-L1, SSO (Sun-Synchronous Orbit), HEO (Highly Elliptical Orbit), and GEO (Geostationary Orbit), there is a record of 100% success of all flights. However, these orbits account for a smaller proportion of the total number of launches.

Payload vs. Orbit Type

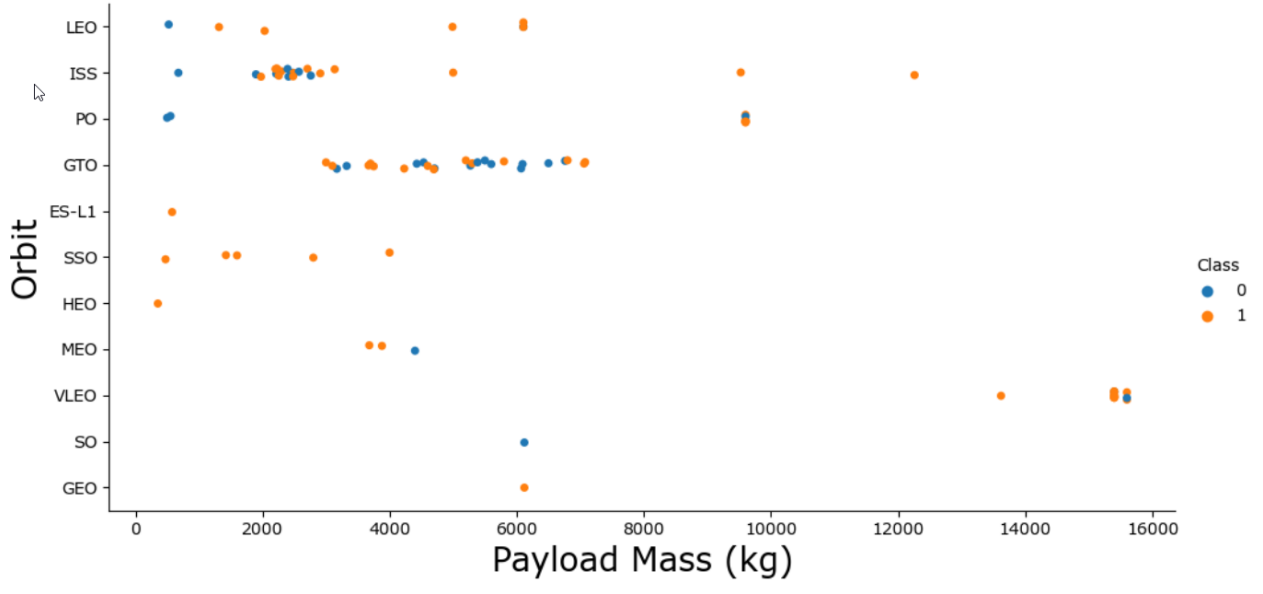


Diagram: scatter plot of Payload vs Orbit

Conclusion

1.Higher payloads are often linked to higher success rates in launches.
At highest payloads, there is only one exception.

2.Orbits like ES-L1, SSO, HEO, and GEO, with fewer flights, appear to be exceptions to this trend.

3.This suggests that smaller payloads might be used for some kind of test flights, while larger payloads are intended for missions expected to succeed.

Launch Success Yearly Trend

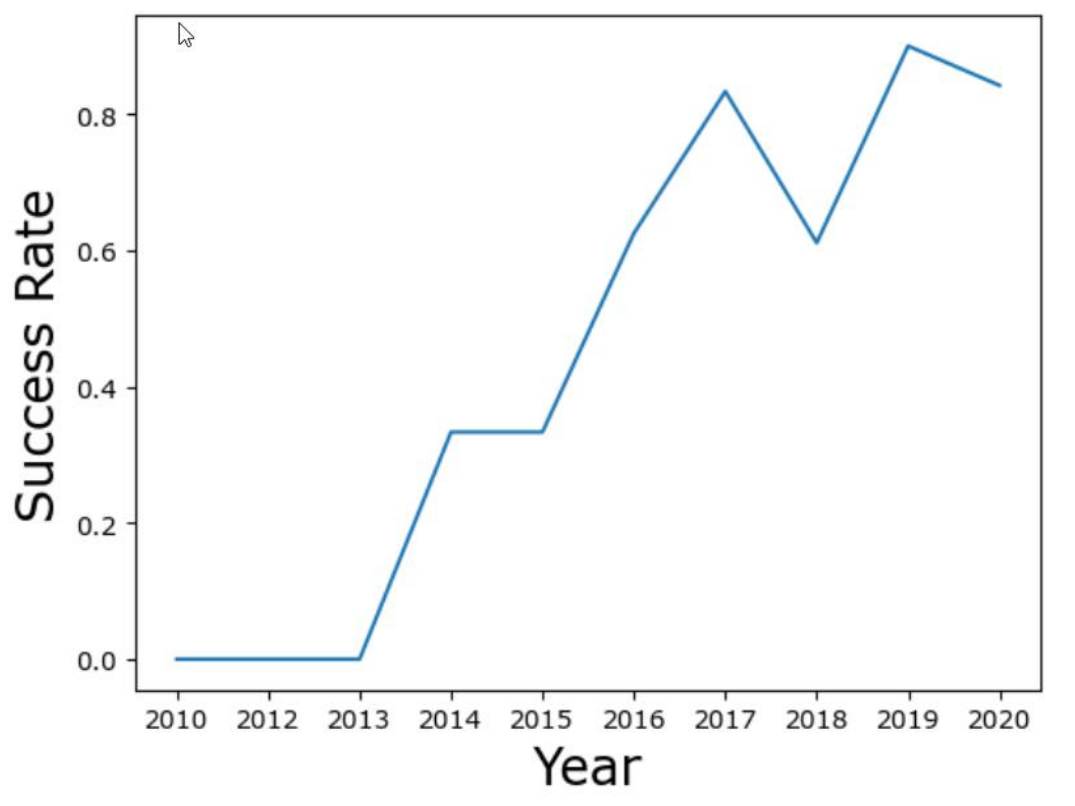


Diagramm: scatter plot of launch success yearly trend

Conclusion

1.Since successful launches are to be observed (2013), the success rate increases

2.The increase of the success rate is steeper in the beginning, roughly up to 2017 than in later years

3.However, the success rate remains quite high at about 80% since 2017

4.One exception to this, the year 2018 at 60% success rate appears to be an outlier, as 2017, 2019 and 2020 success rate are bigger than 80%

All Launch Site Names

```
In [9]: %sql SELECT DISTINCT "LAUNCH_SITE" FROM SPACEXTBL;
 * sqlite:///my_data1.db
Done.
```

Out[9]:

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

The key point of finding the names of the unique launch sites is the use of 'DISTINCT' in the SQL query.

1.CCAFS LC-40: Cape Canaveral Air Force Station Launch Complex 40, located in Florida, USA.
2.VAFB SLC-4E: Vandenberg Air Force Base Space Launch Complex 4E, located in California, USA.
3.KSC LC-39A: Kennedy Space Center Launch Complex 39A, located in Florida, USA.
4.CCAFS SLC-40: Cape Canaveral Air Force Station Space Launch Complex 40, located in Florida, USA.

Launch Site Names Begin with 'KSC'

In [10]:
```
%sql SELECT * FROM SPACEXTABLE WHERE "LAUNCH_SITE" LIKE 'KSC%' LIMIT 5;
```

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

Out[10]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2017-03-16	06:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	5600	GTO	EchoStar	Success	No attempt
2017-03-30	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
2017-01-05	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300	LEO	NRO	Success	Success (ground pad)
2017-05-15	23:21:00	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070	GTO	Inmarsat	Success	No attempt

Points of finding 5 records where launch sites' names start with `KSC` are the use of

- '…**LIKE `KSC%`**… ' in order to select launch sites containing 'KSC' and
- '**LIMIT 5**;' in order to limit the findings to a number of 5

in the SQL query. It was doublechecked, that the output provided data as intended.

Total Payload Mass

```
In [11]: %sql SELECT SUM("PAYLOAD_MASS__KG_") FROM SPACEXTABLE WHERE "Customer" LIKE 'NASA (CRS)';
 * sqlite:///my_data1.db
Done.
```

Out[11]:

SUM("PAYLOAD_MASS__KG_")
45596

The key point of calculating the total payload carried by boosters from NASA is the use of 'SELECT SUM…' in the SQL query.

Average Payload Mass by F9 v1.1

```
In [12]: %sql SELECT AVG("PAYLOAD_MASS__KG_") FROM SPACEXTABLE WHERE "Booster_Version" LIKE 'F9 v1.1';
 * sqlite:///my_data1.db
Done.
```

Out[12]:

AVG("PAYLOAD_MASS__KG_")
2928.4

The key point of calculating the average payload mass carried by booster version F9 v1.1 is the use of 'AVG…' in the SQL query.

First Successful Ground Landing Date

```
In [13]: %sql SELECT MIN(DATE) AS "First Successful Landing Outcome in Ground Pad" \
         FROM SPACEXTBL WHERE "LANDING_OUTCOME" LIKE "Success (drone ship)" ;

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

Out[13]:

First Successful Landing Outcome in Ground Pad
2016-04-08

The key point of finding the dates of the first successful landing outcome on drone ship is the use of

– 'MIN' applied to "DATE" were the

– 'LANDING__OUTCOME' is 'Success (drone ship)'

in the SQL query.

Successful Drone Ship Landing with Payload between 4000 and 6000

```
In [14]: %sql SELECT Booster_Version FROM SPACEXTBL \
         WHERE "LANDING_OUTCOME" LIKE "Success (drone ship)" \
         AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000 ;
          * sqlite:///my_data1.db
         Done.
```

Out[14]:

Booster_Version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

The key point of listing the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000 is to apply 'WHERE' covering:

– 'LANDNG__OUTCOME_ 'Success (drone ship)' and

– 'PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000'

in the SQL query.

Total Number of Successful and Failure Mission Outcomes

```
In [15]: %sql SELECT MISSION_OUTCOME, COUNT(*) as total_number \
         FROM SPACEXTBL \
         GROUP BY MISSION_OUTCOME;
```

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

Out[15]:

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

The key point of calculating the total number of successful and failure mission outcomes is to apply 'COUNT' and 'SELECT MISSION_OUTCOME' in the SQL query.

The conclusion is, that the number of

– Failure missions is 1

– Successful missions is 100

Boosters Carried Maximum Payload

In [16]:
```
%sql SELECT BOOSTER_VERSION \
FROM SPACEXTBL \
WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

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

Out[16]:

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

The key point of listing the names of the booster which have carried the maximum payload mass is to apply SELECT to MAX(PAYLOAD_MASS__KG_ in the SQL query plus subquery.

2017 Success (ground pad) Landing

```
In [20]: %sql SELECT substr(Date,6,2) as month, DATE, BOOSTER_VERSION, LAUNCH_SITE, [Landing_Outcome] \
FROM SPACEXTBL \
WHERE [Landing_Outcome] = 'Success (ground pad)' AND substr(Date,1,4)='2017';

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

Out[20]:

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

The key point of listing the records which will display the month names, succesful landing_outcomes in ground pad, booster versions, launch_site for the months in year 2017 is to apply SELECT in a way that filters the records to only include entries from the year 2017 like "substr(Date,1,4)='2017'".

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

In [21]:
```
%sql SELECT Landing_Outcome as "Landing Outcome", COUNT(Landing_Outcome) as "Total Count" \
FROM SPACEXTBL \
WHERE DATE between '2010-06-04' and '2017-03-20' \
GROUP BY Landing_Outcome ORDER BY COUNT(Landing_Outcome) DESC;
```

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

Out[21]:

Landing Outcome	Total Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

The key point of ranking 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 is to identify the most frequent landing outcomes during that period using GROUP BY Landing_Outcome ORDER BY COUNT(Landing_Outcome) DESC.

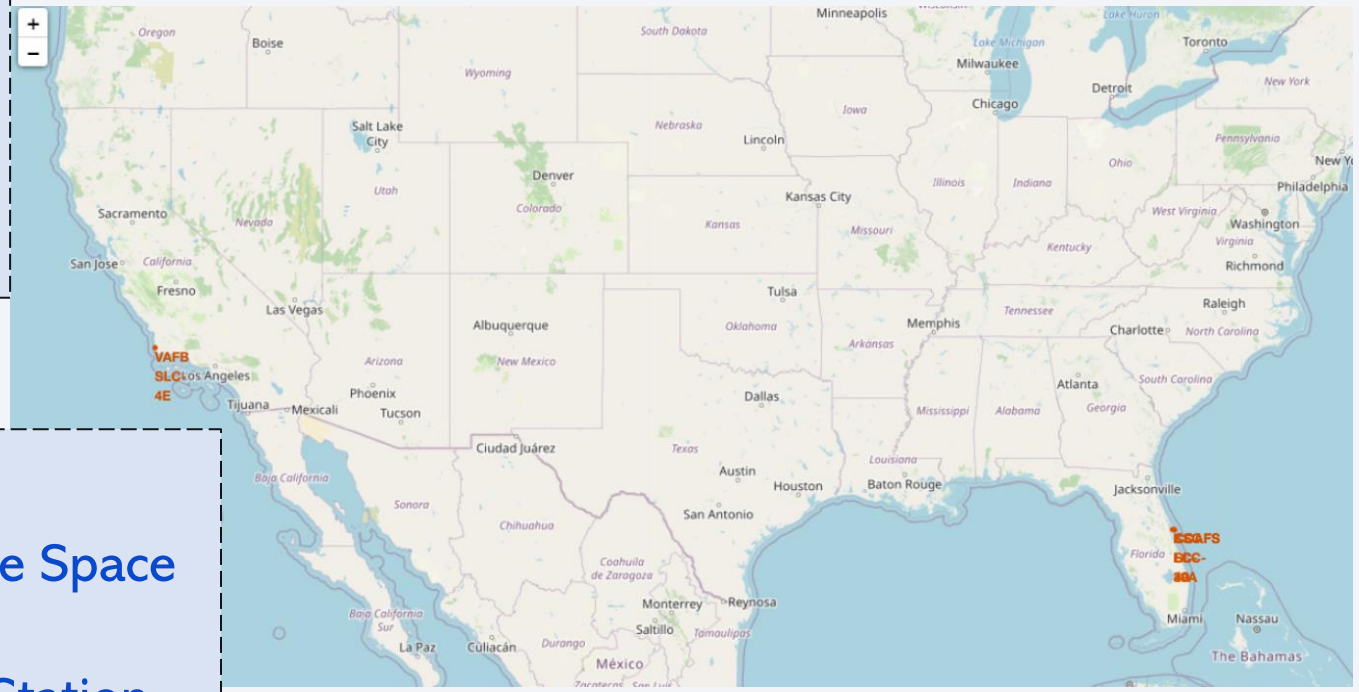
Section 3

Launch Sites Proximities Analysis

Location of SpaceX launch sites

All SpaceX launch sites are located

- Within the USA
- In the southern regions of the USA
- Close to the sea
- Near transportation infrastructure
- Away from large cities



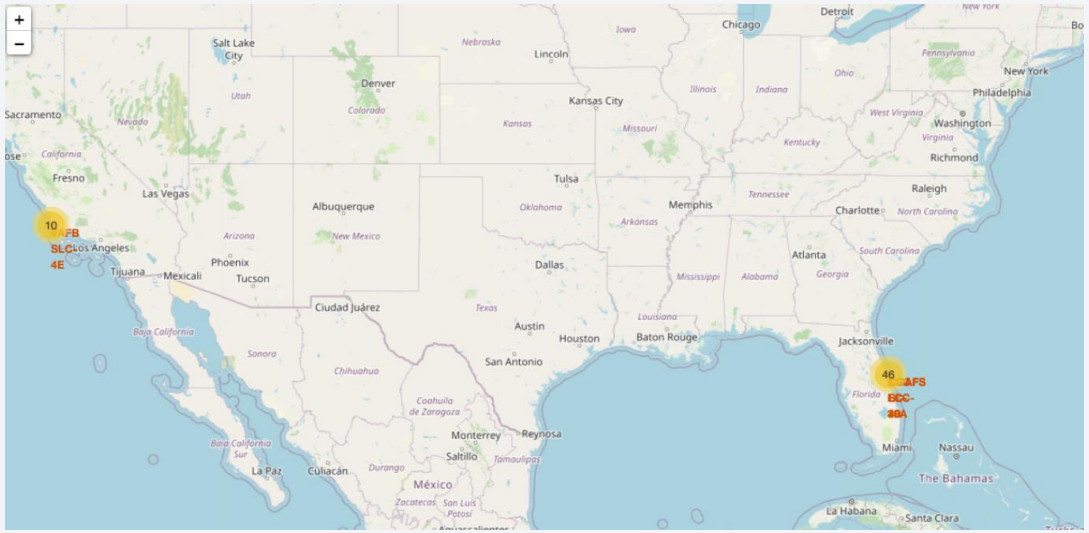
West Coast - California

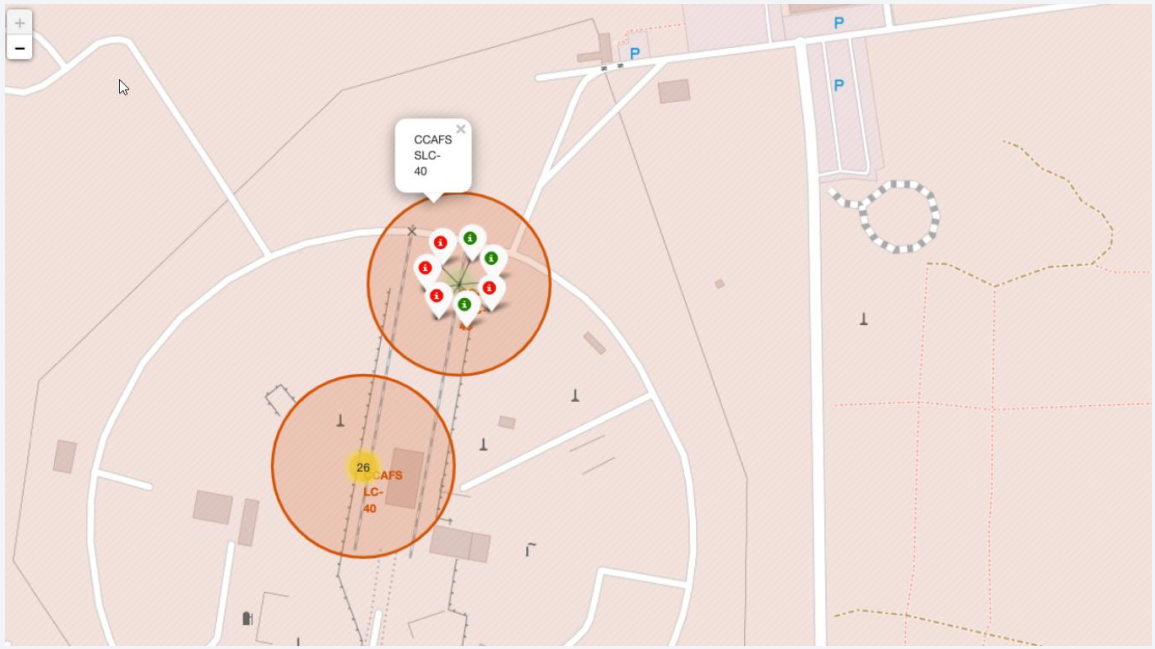
- Vandenberg Air Force Base Space

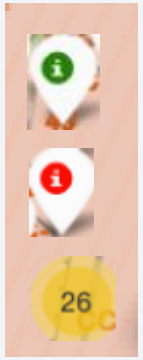
East Coast - Florida

- Cape Canaveral Air Force Station
- Kennedy Space Center

Depicting Launch Outcomes on the Folium Map







- Green markers indicate successful launches.
- Red markers indicate unsuccessful launches.
- Yellow markers indicate overlapping red and green markers, with the count displayed at the center.

Launch Site Proximity Analysis on Folium Map

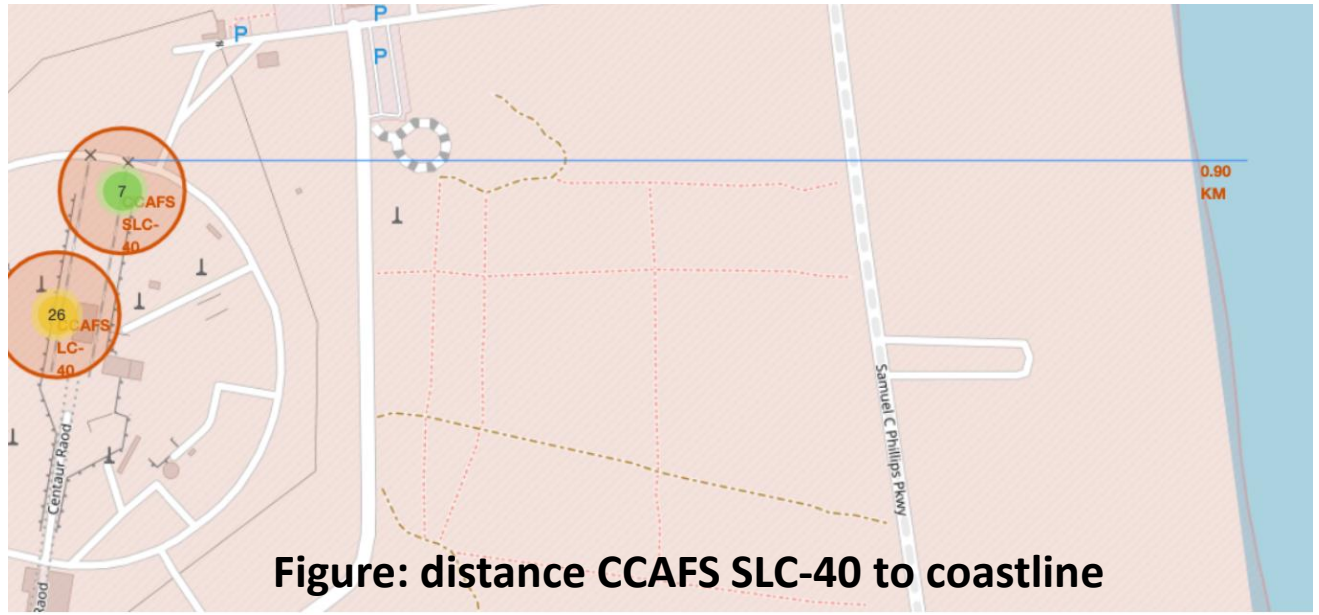


Figure: distance CCAFS SLC-40 to coastline

```
In [10]: EARTH_RADIUS = 6371  # Earth's radius in kilometers

for index, row in launch_sites_df.iterrows():
    coordinate = [row['Lat'], row['Long']]

    # Compute distance from the equator in kilometers
    distance_from_equator = abs(row['Lat'] * (3.14 / 180) * EARTH_RADIUS)

    print(f"Launch Site: {row['Launch Site']}")
    print(f"Coordinates: {coordinate}")
    print(f"Distance from equator: {distance_from_equator:.2f} km")
    print("--------------------------------")
```

```
Launch Site: CCAFS LC-40
Coordinates: [28.56230197, -80.57735648]
Distance from equator: 3174.37 km
--------------------------------
Launch Site: CCAFS SLC-40
Coordinates: [28.56319718, -80.57682003]
Distance from equator: 3174.47 km
--------------------------------
Launch Site: KSC LC-39A
Coordinates: [28.57325457, -80.64689529]
Distance from equator: 3175.59 km
--------------------------------
Launch Site: VAFB SLC-4E
Coordinates: [34.63283416, -120.6107455]
Distance from equator: 3849.04 km
--------------------------------
```

Figure: Measuring Distances to Equator

```
In [23]: # Create a marker with distance to a closest city, railway, highway, etc.
closest_highway = 28.57369, -80.65519
closest_railroad = 28.57369, -80.65396
closest_city = 28.16046, -80.65197

In [24]: distance_highway = calculate_distance(launch_site_lat, launch_site_lon, closest_highway[0], closest_highway[1])
print('distance_highway =',distance_highway, ' km')
distance_railroad = calculate_distance(launch_site_lat, launch_site_lon, closest_railroad[0], closest_railroad[1])
print('distance_railroad =',distance_railroad, ' km')
distance_city = calculate_distance(launch_site_lat, launch_site_lon, closest_city[0], closest_city[1])
print('distance_city =',distance_city, ' km')
```

```
distance_highway = 0.8283938034246578  km
distance_railroad = 0.7084178280903194  km
distance_city = 45.92522356151617  km
```

Figure: Marking Infrastructure and Measuring Distances with Folium

Conclusions from Launch Site Proximity Analysis

All SpaceX Launch Site Locations*

- Close to the sea – approximately 1 km away.
- Within 1 km of transportation infrastructure.
- Away from large cities – at least 45 km away.
- Globally, quite close to the equator – between 3000 km and 4000 km away.
- Positioned as close to the equator as feasible within the USA, considering the constraints mentioned above.

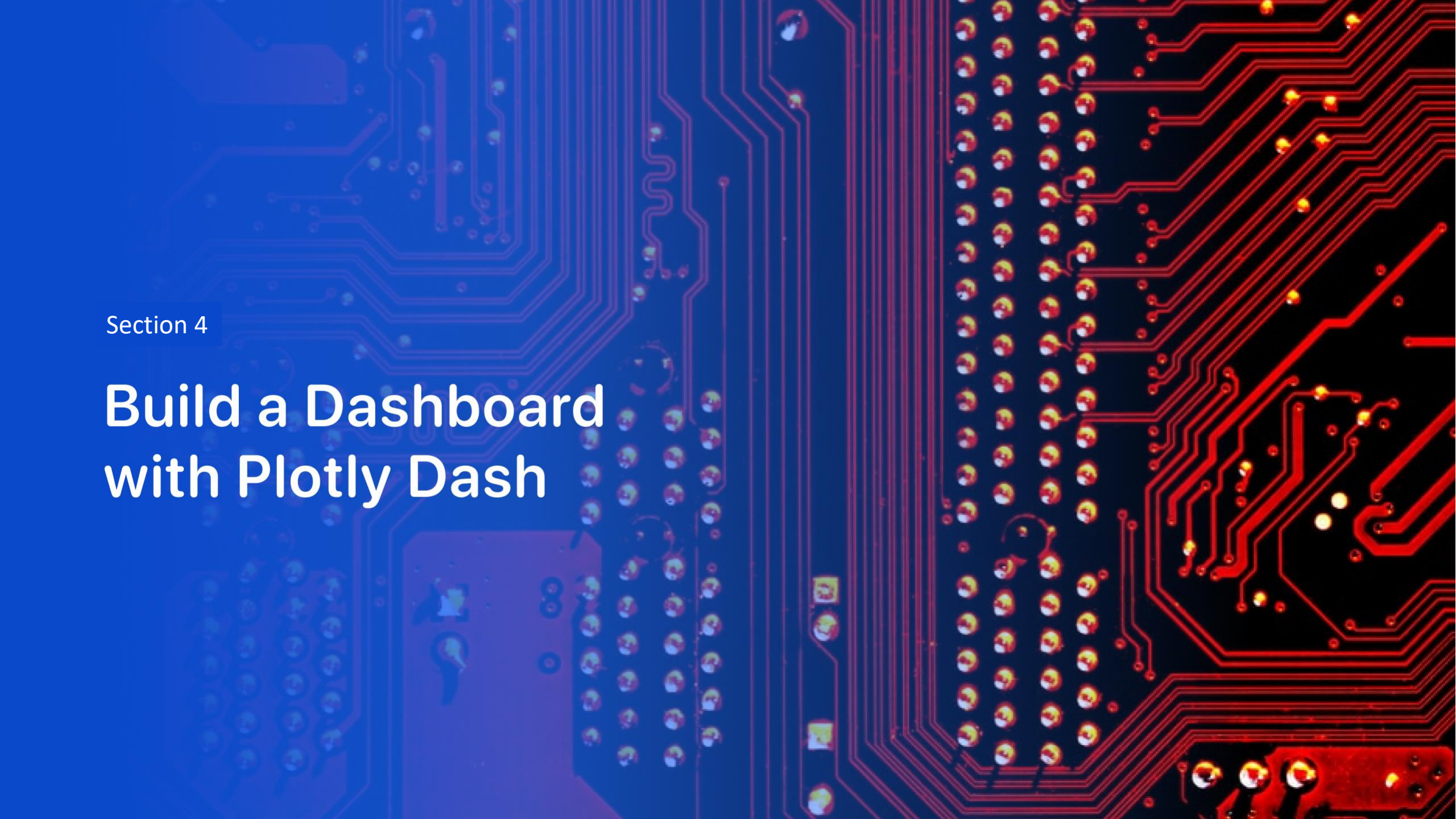
*All measurements were conducted at the CCAFS SLC-40 launch site. The conclusions were visually compared to other identified launch sites and were found to be consistent.

Conclusions from Launch Site Proximity Analysis

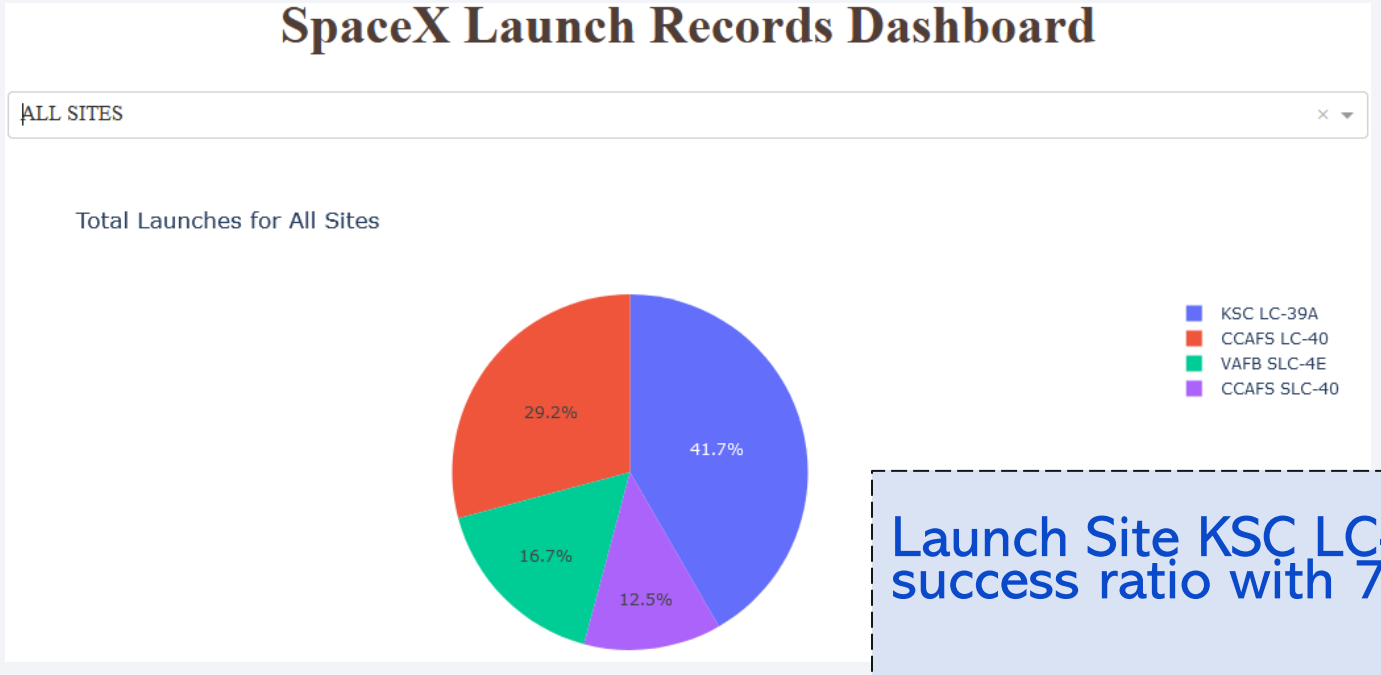
Rationale for SpaceX Launch Site Locations*

- Proximity to railways enables efficient transportation of cargo and staff.
- Proximity to highways enables efficient transportation of cargo and staff.
- Proximity to coastlines:
  - enables sea transportation, crucial for landing platforms (!).
  - reduces the risk of harm to civilians, as the sea is not inhabited.
- Safe distance to populated areas minimizes risks to civilians in case of flight events.
- Positioned relatively close to the equator to use some rotational energy from planet Earth.
- Located within the USA, mainly for safety, security and regulatory reasons.

*All measurements were conducted at the CCAFS SLC-40 launch site. The conclusions were visually compared to other identified launch sites and were found to be consistent.


Section 4
Build a Dashboard with Plotly Dash

Success Count for all Sites

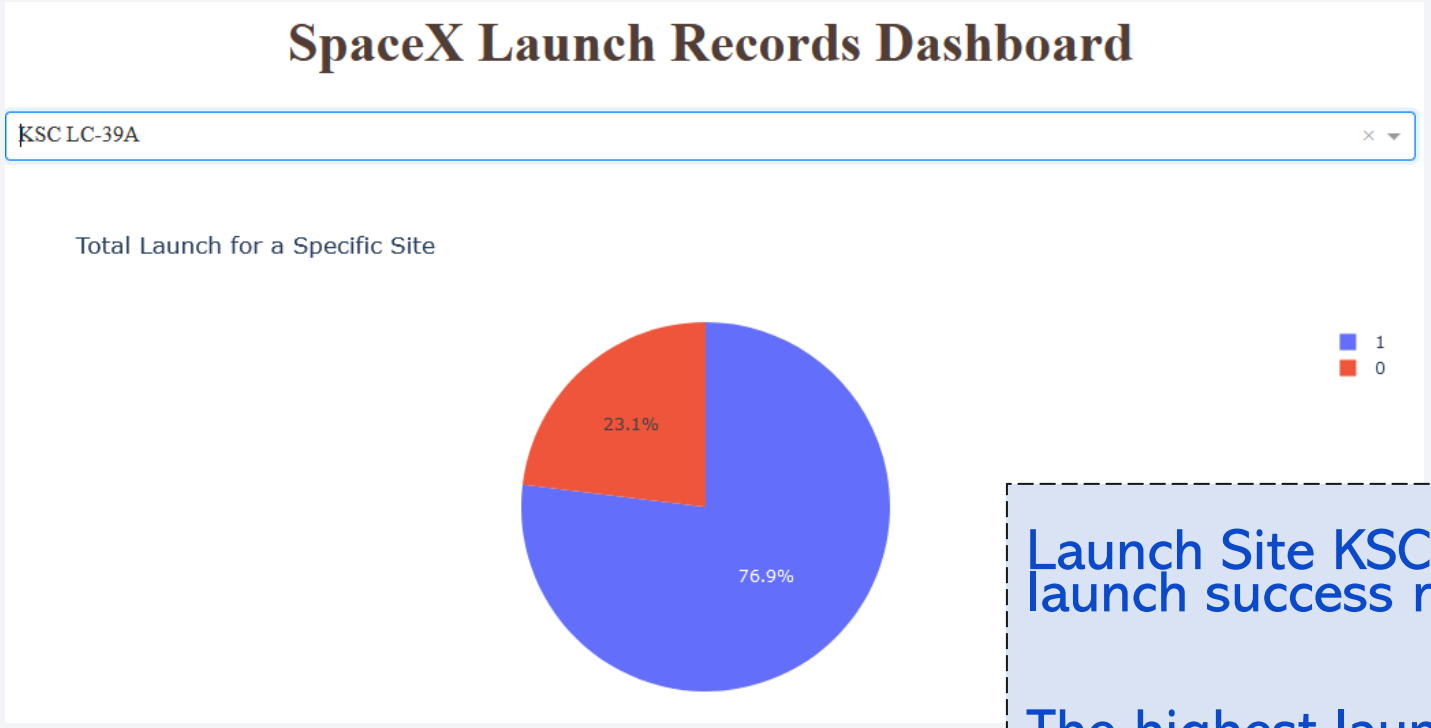


Launch Site KSC LC-39A has the highest launch success ratio with 76,9% (see also next slide)

It is followed by CCAFS LC-40 with 73,1%

where low success ratios are associated with
CCAFS SLC-40 42,9%
VAFB SLC-4E 40%

Launch Site with Highest Launch Success Ratio



Launch Site KSC LC-39A has the highest launch success ratio with 76,9%

The highest launch success ratio is also associated with:

Booster type FT and big pay load.

Payload vs. Launch Outcome for All Sites

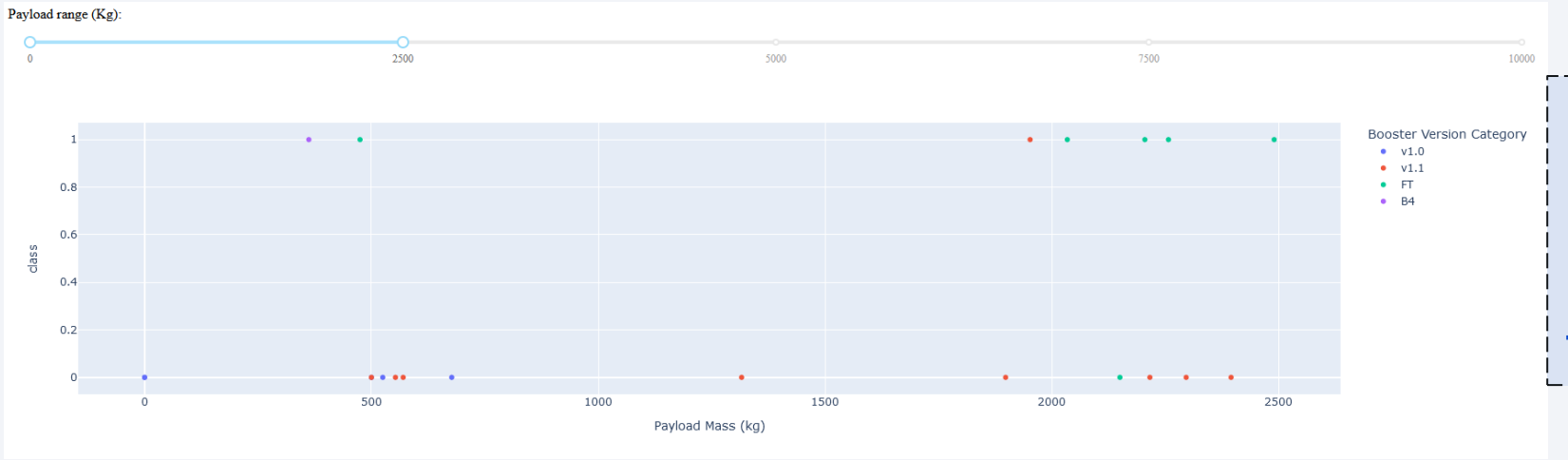
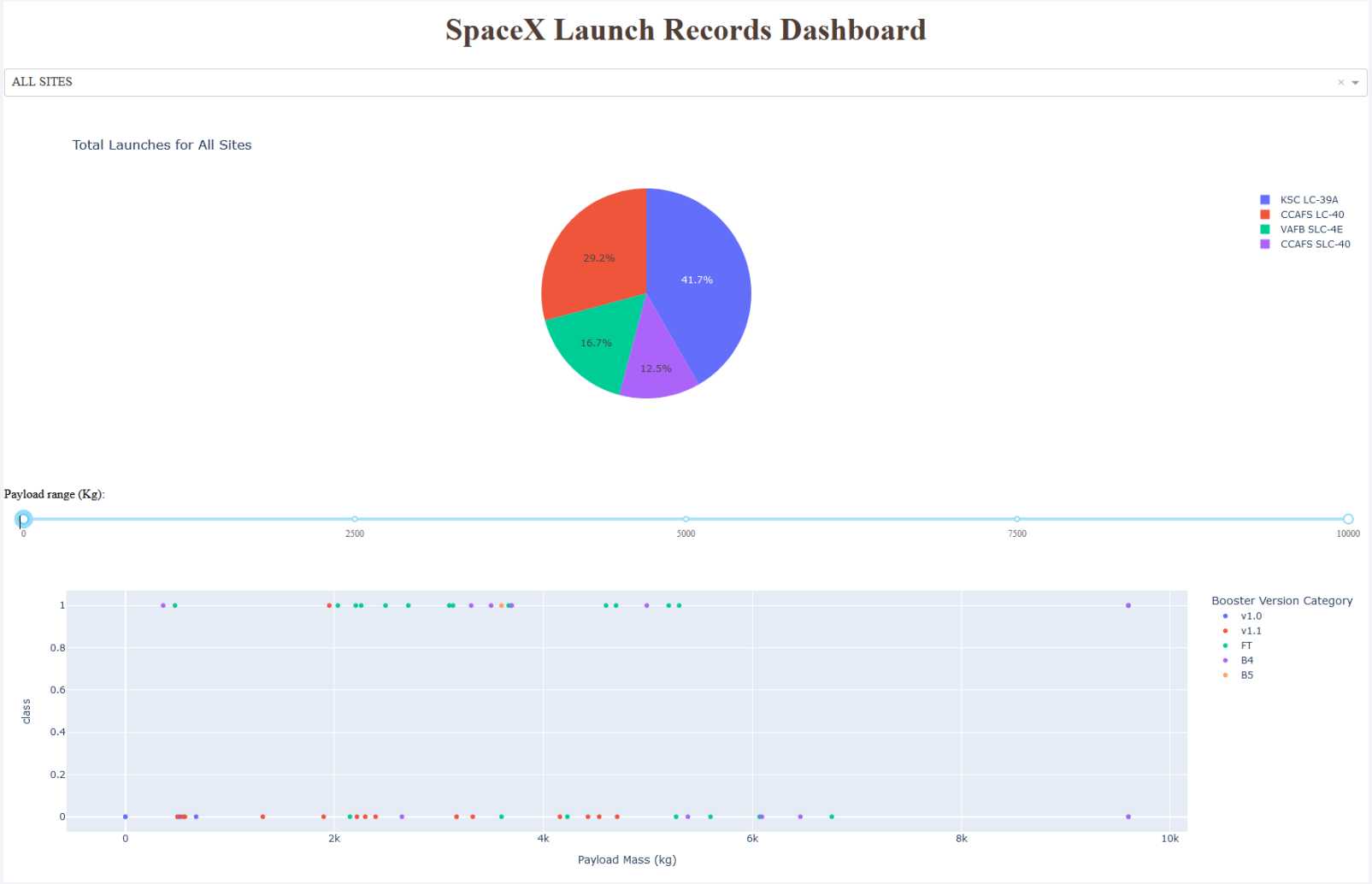


Figure depicting payload vs. Launch for low payloads up to 2500 kg

Figure depicting payload vs. Launch for big payloads between 2500 kg and 10000 kg

Payload vs. Launch Outcome continued



Baseline figure

Depicting payload vs. Launch Outcome for all Payloads from min to max Payloads

Section 5

Predictive Analysis (Classification)

Classification Accuracy

Algorithm	Accuracy	Accuracy on Test Data	Tuned Hyperparameters
Logistic Regression	0.8464285714285713	0.8333333333333334	{'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
SVM	0.8482142857142856	0.8333333333333334	{'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
Decision Tree I	0.9053571428571429	0.8333333333333334	{'criterion': '**entropy**', 'max_depth': 4, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5, 'splitter': 'random'}
Decision Tree II	0.875	0.8333333333333334	{'criterion': '**gini**', 'max_depth': 4, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5, 'splitter': 'random'}
KNN	0.8482142857142858	0.8333333333333334	{'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}

The decision tree model outperforms others in classification accuracy, achieving approximately 90%, closely followed by other models at around 85%

The link to the notebook is https://github.com/sebnotch/presi-and-code-data-science/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Classification Accuracy continued

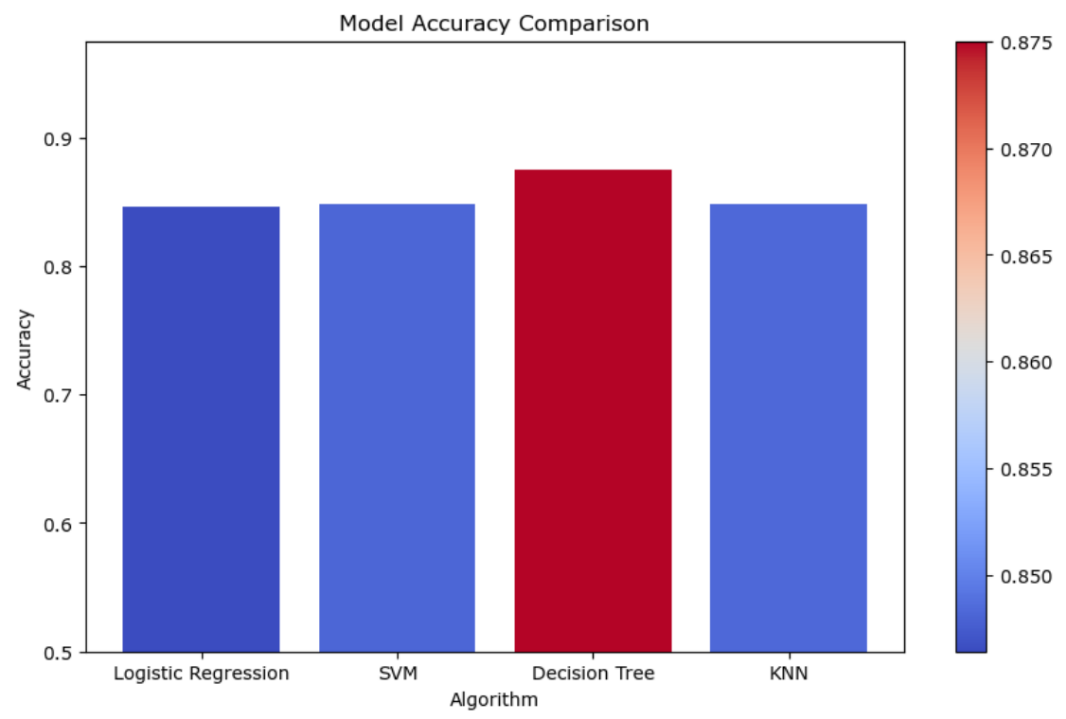


Figure: Algorithm vs Accuracy

The decision tree model outperforms others in classification accuracy, achieving approximately 90%, closely followed by other models at around 85%.

Side note: the accuracy of the decision tree algorithm seems to vary for several runs within jupyter slightly. However, the decision tree model is always better than all the other checked models

This behaviour should be further investigated, compare slide 54 on “Open questions”.

The link to the notebook is https://github.com/sebnotch/presi-and-code-data-science/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Confusion Matrix

- The confusion matrix of the best performing model is identical to the confusion matrices of all other applied models
- In other words, all models are predicting
  - Not landed 3 times correctly (true negatives, upper left square)
  - Landed 12 times correctly (true positives, lower right square)
  - Landed 3 time incorrectly (false positive, upper right square) (this indicates the main weakness of the models)
  - Did not land 0 times incorrectly (false negative, lower left square) (which in fact is correct)

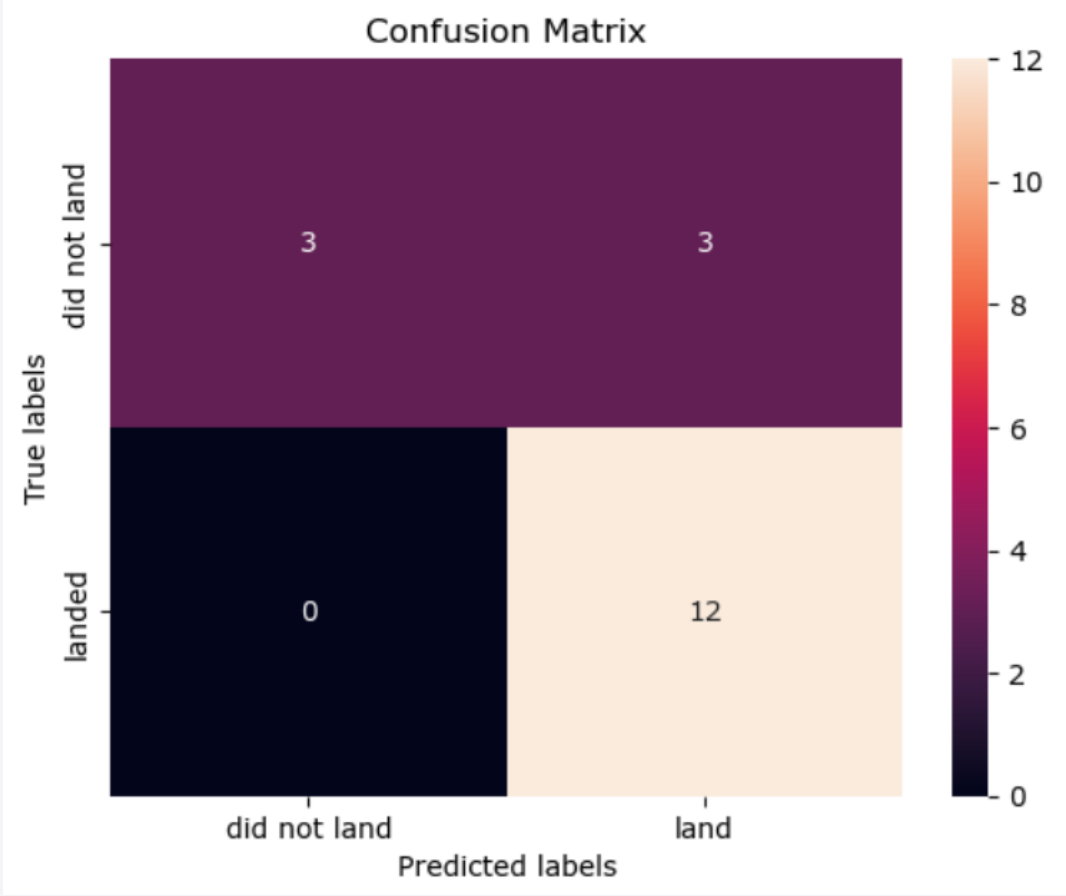


Figure: confusion matrix of all models

Confusion Matrix continued

- Accuracy:
  (TP + TN) / Total
  =(3+12)/18 = 0.833
- Misclassification Rate:
  (FP + FN) / Total
  =(3+0)/18 = 0.1667

Accuracy + Misclassification Rate = 1

- Precision:
  TP/ all positive predicted
  =12 / 15 = 0.8
- Prevalence:
  actual all positive / Total
  =0 + 12 / 18 = 0.667

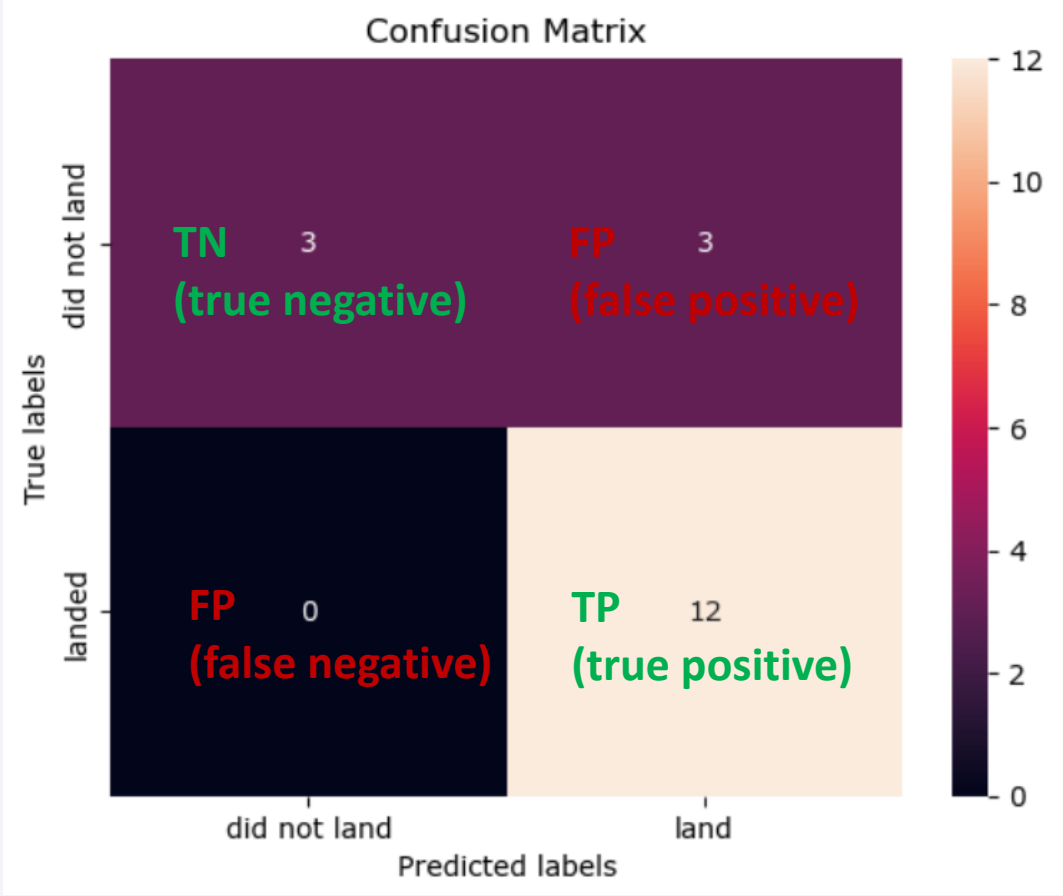


Figure: confusion matrix of all tested models

Conclusions

- Attributes contributing to higher launch success rates were identified and include:
  - Orbits such as ES-L1, GEO, HEO, SSO, VLEO
  - Launches conducted in the later project phases
  - Larger payloads
  - ➢ The presentation suggests possible explanations for these factors.
- A decision tree classifier was identified as the optimum machine learning model with an accuracy exceeding 90%.
  - ➢ The ML model helps to scrutinize and to quantify the findings mentioned above.
- Based on the above findings, further improvements to the SpaceX program are suggested, such as incorporating additional design parameters, including launch height, detailed in the next slides.

Open Questions - Areas for deeper data analysis should include

Model / Data related items

– as models demonstrated similar accuracy –

- Consider investigating alternative models which may score much better (including sensitivity against several decision tree runs)
- Explore what additional data could have the potential to improve the model performance.
- Consider segmenting the dataset into
    - distinct categories,
    - such as low and high payloads and
    - apply and optimize separate models for each.

The latter strategy assumes that each category has unique behavior which can be captured with a specific model.

It this turns out to be true, that would be a way for further improved accuracy.

Open Questions - Areas … continued

Pay load related items

- The data suggests a higher payload mass correlates with higher success rates. The cause behind this correlation need clarification.
- Comprehending why larger payloads often result in successful missions could provide insights for enhancing future launches.
- Examining the payload mass and cost relationship to determine if the pay load cost affects the mission success.

➔ Payload cost analysis could show that lower costs are associated with test flights (possibly planned aborts), while higher costs imply missions of high importance (possibly no planned aborts).

Open Questions - Areas … continued

Flight length related items

- The length of the flight before an abort may reflect the degree of command and control over the Falcon 9.
  A short flight suggests less control, while a longer flight implies more effective command and control.
- The type of early flight termination — whether intentional for safety or due to technical problems — also indicates the level of command and control.

Outlook and Proposal

Elevated Launch Sites for Energy Saving at Launch

Core Concept: Utilizing high-altitude regions like mountains or lakes may offer advantages for launches by reducing the distance to orbit (which reduces gravitation and friction in the atmosphere).

1) **Strategic Approach:** Deploy Folium for an advanced search to pinpoint geographical areas that align with established Criteria (infrastructure availability, Proximity to the coastline and to the equatorial line)

   plus a new Criterion: **high-altitude terrain** for launch site.

2) **Pragmatic Exploration:** Searching for potential new launch locations at higher altitudes, with initial candidates including regions in India, Hawaii, the Rocky Mountains, Lake Victoria, and Lake Titicaca.

3) **Objective:** Document exact positions and elevations to enable further assessments regarding the feasibility and quantification of possible energy savings at each proposed site.

Vision of energy saving with SPACEX

The prospect from the future reads:

Figure: high altitude launch, reducing start mass & saving energy

SpaceX, initially represented by "**_X**", is now taking on a new identity as "**|^{X}**". This represents a new era of innovation and commitment to sustainability, beneficial for the company and the planet – a true "Musk-Win Situation".

Technology already exists to move materials from sea level to higher ground for launches. This development can cut down the initial fuel weight or allow for more payload. In short, it achieves a more effective use of fuel per payload, steering SpaceX towards an even greener space travel.

Appendix

- All requested details are provided in this presentation (in some cases on slides "… continued") or through links to GitHub.
- For additional background knowledge, it is recommended to consult the 'Falcon User's Guide' (falcon-users-guide-2021-09.pdf , see snapshot on the right).

Figure: snapshot of 1st page from falcon-users-guide-2021-09.pdf

Appendix

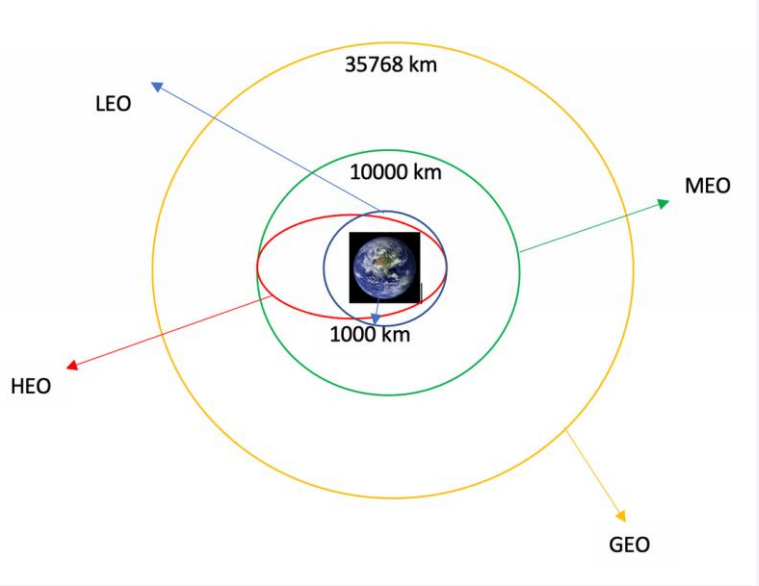
- Nomenclature (from jupyter notebooks as per GitHub storage)

The data contains several Space X launch facilities: Cape Canaveral Space Launch Complex 40 **VAFB SLC 4E** , Vandenberg Air Force Base Space Launch Complex 4E **(SLC-4E)**, Kennedy Space Center Launch Complex 39A **KSC LC 39A** .The location of each Launch Is placed in the column `LaunchSite`

Each launch aims to an dedicated orbit, and here are some common orbit types:

- **LEO**: Low Earth orbit (LEO)is an Earth-centred orbit with an altitude of 2,000 km (1,200 mi) or less (approximately one-third of the radius of Earth),[1] or with at least 11.25 periods per day (an orbital period of 128 minutes or less) and an eccentricity less than 0.25.[2] Most of the manmade objects in outer space are in LEO [1].
- **VLEO**: Very Low Earth Orbits (VLEO) can be defined as the orbits with a mean altitude below 450 km. Operating in these orbits can provide a number of benefits to Earth observation spacecraft as the spacecraft operates closer to the observation[2].
- **GTO** A geosynchronous orbit is a high Earth orbit that allows satellites to match Earth's rotation. Located at 22,236 miles (35,786 kilometers) above Earth's equator, this position is a valuable spot for monitoring weather, communications and surveillance. Because the satellite orbits at the same speed that the Earth is turning, the satellite seems to stay in place over a single longitude, though it may drift north to south," NASA wrote on its Earth Observatory website [3] .
- **SSO (or SO)**: It is a Sun-synchronous orbit also called a heliosynchronous orbit is a nearly polar orbit around a planet, in which the satellite passes over any given point of the planet's surface at the same local mean solar time [4] .
- **ES-L1** :At the Lagrange points the gravitational forces of the two large bodies cancel out in such a way that a small object placed in orbit there is in equilibrium relative to the center of mass of the large bodies. L1 is one such point between the sun and the earth [5] .
- **HEO** A highly elliptical orbit, is an elliptic orbit with high eccentricity, usually referring to one around Earth [6].
- **ISS** A modular space station (habitable artificial satellite) in low Earth orbit. It is a multinational collaborative project between five participating space agencies: NASA (United States), Roscosmos (Russia), JAXA (Japan), ESA (Europe), and CSA (Canada) [7]
- **MEO** Geocentric orbits ranging in altitude from 2,000 km (1,200 mi) to just below geosynchronous orbit at 35,786 kilometers (22,236 mi). Also known as an intermediate circular orbit. These are "most commonly at 20,200 kilometers (12,600 mi), or 20,650 kilometers (12,830 mi), with an orbital period of 12 hours [8]
- **HEO** Geocentric orbits above the altitude of geosynchronous orbit (35,786 km or 22,236 mi) [9]
- **GEO** It is a circular geosynchronous orbit 35,786 kilometres (22,236 miles) above Earth's equator and following the direction of Earth's rotation [10]
- **PO** It is one type of satellites in which a satellite passes above or nearly above both poles of the body being orbited (usually a planet such as the Earth [11]

some are shown in the following plot:



Thanks to IBM and the team behind

Thank you!