IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Reynard Etwaroo
10/22/2023

Outline

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

Executive Summary

Summary of methodologies

- Data collection
- Data wrangling
- Exploratory Data Analysis with Data Visualization
- Exploratory Data Analysis with SQL
- Building an interactive map with Folium
- Building a Dashboard with Plotly Dash
- Predictive analysis (Classification)

Summary of all results

- Exploratory Data Analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Introduction

The commercial space industry is rapidly evolving, with companies like Virgin Galactic, Rocket Lab, Blue Origin, and SpaceX leading the way. Among these, SpaceX stands out due to its relatively inexpensive rocket launches, which are primarily due to the company's capability to reuse the first stage of its Falcon 9 rocket. In this presentation, we will delve into the factors influencing the cost of space launches and the ability of SpaceX to reuse the first stage of its Falcon 9 rocket. We will explore how these factors can be predicted using machine learning and data science techniques, and how these insights can be used to compete with SpaceX in the commercial space industry.

The identified problems include:

- Identifying all factors that influence the landing outcome.
- The relationship between each variable and how it affects the outcome.
- The best condition needed to increase the probability of successful landing.

Section 1

Methodology

Methodology

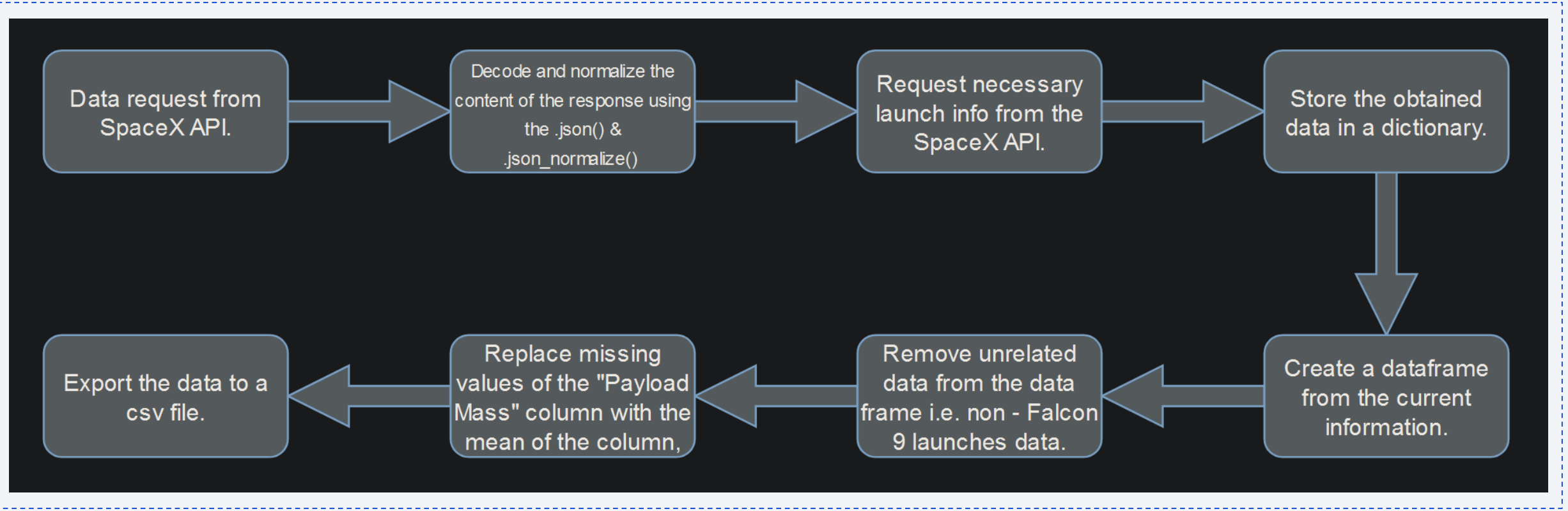
Executive Summary

- Data collection methodology:
  - The data was obtained from Wikipedia via web scraping and SpaceX REST API.
- Perform data wrangling
  - The data was processed by categorical features through one-hot encoding.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

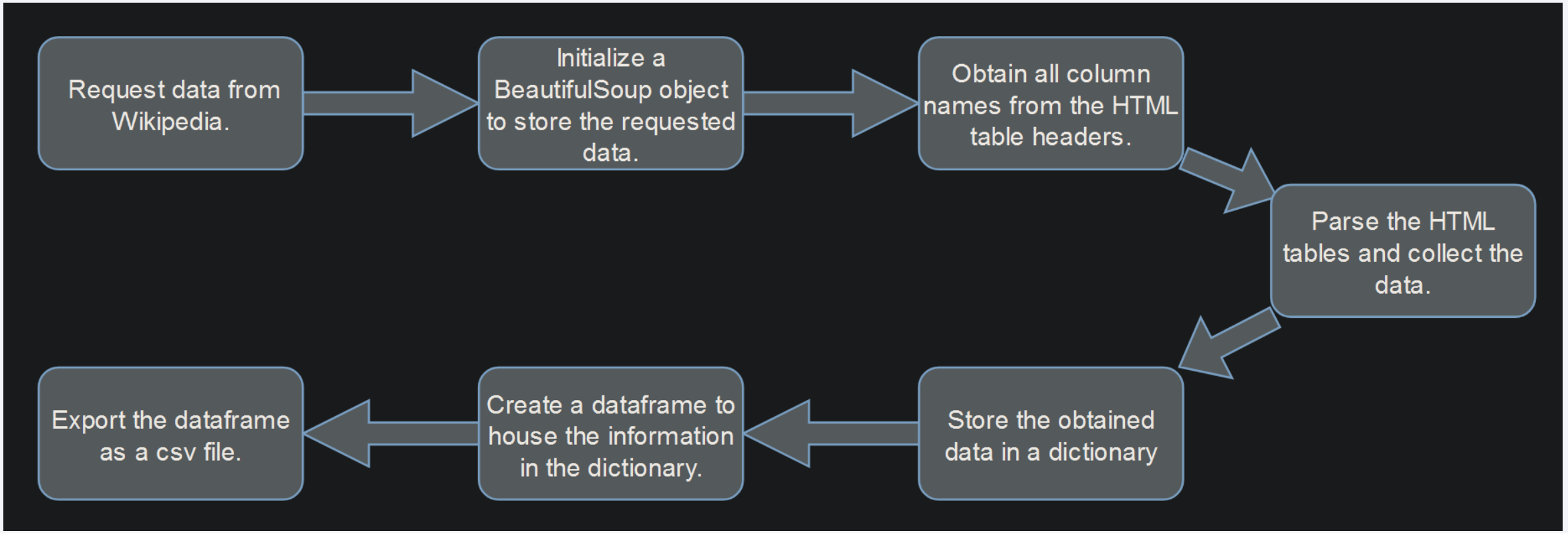
The data was collected from the SpaceX REST API, specifically the endpoint api.spacexdata.com/v4/launches/past. A GET request with the requests library in Python was used to retrieve it, resulting in a JSON response that was converted into a dataframe using the json_normalize function.

Data Collection - SpaceX API



SpaceX API calls notebook

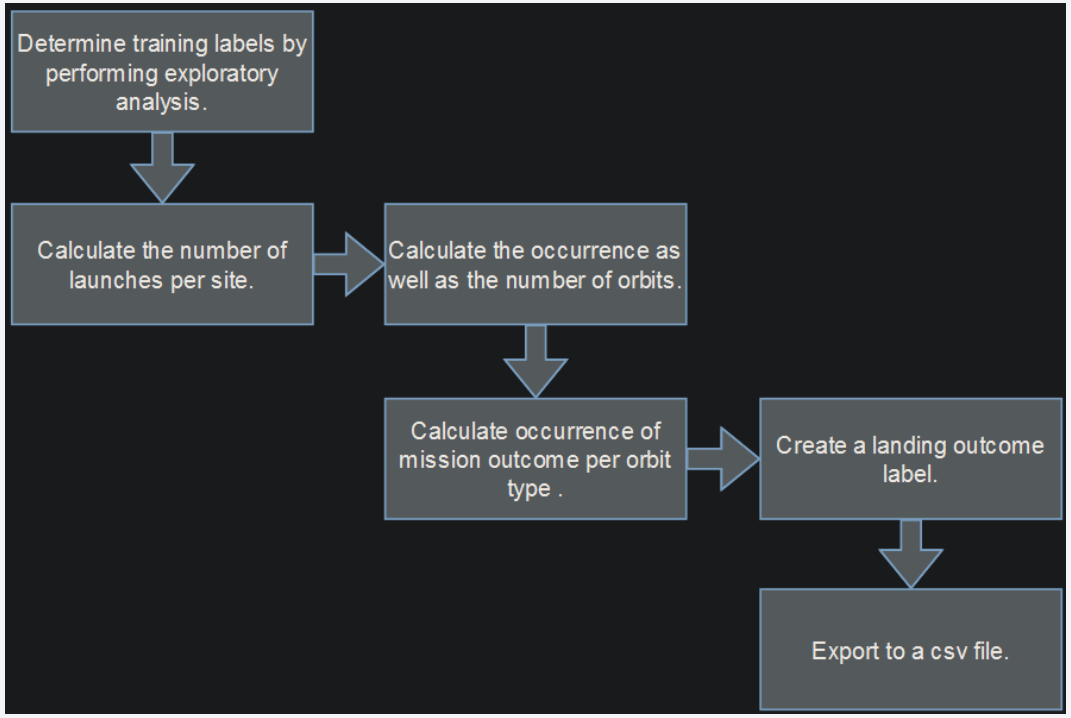
Data Collection - Scraping



Web scraping notebook

Data Wrangling

The data was wrangled by dealing with identification numbers in some columns, such as rocket, which were replaced with specific data using additional API endpoints. The data was also filtered to exclude Falcon 1 launches, and null values in the PayloadMass column were replaced with the mean of the PayloadMass data.



Data wrangling notebook

EDA with Data Visualization

The types of charts plotted were:

- **Scatter plots**: these were selected because they are able to show the relationship between variables.
- **Bar charts**: selected due to their ability of being able to compare discrete categories of data.
- **Line plots**: selected to visualize trends and changes over time.

EDA with Data Visualization Notebook

EDA with SQL

SQL queries performed:

- Displaying the names of the unique launch sites in the space mission.
- Displaying 5 records where launch sites begin with the string ‘CCA’
- Displaying the total payload mass carried by boosters launched by NASA.
- Displaying average payload mass carried by booster version F9 v1.1.
- Listing the date when the first successful landing outcome in ground pad was achieved.
- Listing the names of the boosters which have success in drone ship and payload mass > 4000 but < 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.
- Listing the failed landing outcomes in drone ship, their booster versions and launch site names for the months in year 2015.
- 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.

EDA with SQL Notebook

Build an Interactive Map with Folium

The following information describes the creation and addition of various map objects to a Folium map, and the reasons behind their inclusion:

- **Launch Sites Markers**: Markers were added to each Launch Site using their latitude and longitude coordinates. These markers included a circle, a popup label, and a text label. The marker for the NASA Johnson Space Center was specifically highlighted as the starting location. This was done to visualize the geographical locations of each Launch Site and their proximity to the Equator and coasts.
- **Launch Outcomes Markers**: Colored markers were added to represent the success or failure of each launch. Green markers were used for successful launches, while red markers were used for failed launches. A Marker Cluster was used to group these markers, allowing for easier identification of launch sites with relatively high success rates.
- **Distances to Proximities**: Colored lines were added to represent the distances between a Launch Site and its proximities, such as Railway, Highway, Coastline, and Closest City. This was done to provide a more detailed view of the spatial relationships between different elements.

The inclusion of these objects and their representation serves to provide a comprehensive and interactive visualization of the data, making it easier to understand the geographical locations of the Launch Sites, the outcomes of their launches, and their proximities to various elements

Interactive Visual Analytics with Folium

Build a Dashboard with Plotly Dash

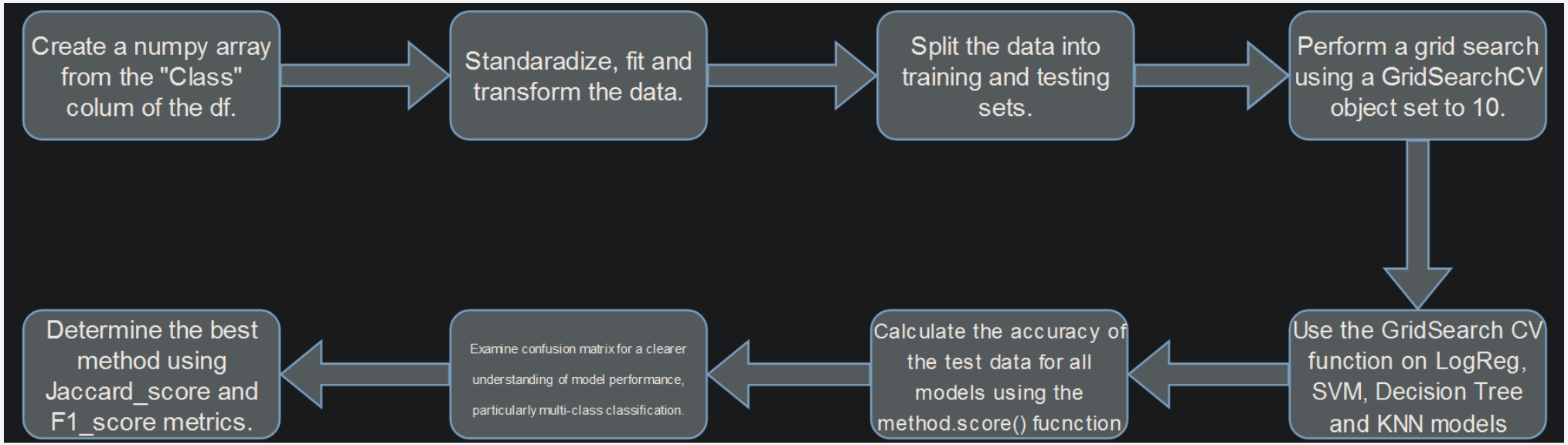
The following objects were added to a Folium map:

- **Launch Sites Dropdown List**: This dropdown list was introduced to allow users to select a Launch Site. The reason for its addition was to provide users with a convenient way to filter the data based on the Launch Site.
- **Pie Chart showing Success Launches (All Sites/Certain Site)**: A pie chart was added to display the total successful launches count for all sites and the Success vs. Failed counts for the site, if a specific Launch Site was selected. This was done to provide a visual representation of the success rate of launches, which can be particularly useful for comparing the success rates across different launch sites.
- **Slider of Payload Mass Range**: A slider was added to allow users to select a range of Payload Mass. The inclusion of this feature allows users to filter the data based on the payload mass, which can be helpful in analyzing the impact of payload mass on the success rate of launches.
- **Scatter Chart of Payload Mass vs. Success Rate for the different Booster Versions**: A scatter chart was added to show the correlation between Payload and Launch Success. This chart was included to visualize the relationship between the payload mass and the success rate of launches, which can help identify any patterns or trends in the data.

These objects were added to enhance the user experience by providing additional ways to interact with and analyze the data. Each object serves a specific purpose and contributes to the overall functionality and usability of the map.

Plotly Dashboard

Predictive Analysis (Classification)



The process begins by creating a NumPy array from the "Class" column and standardizing it using the StandardScaler function from the sklearn library.

- This involves fitting and transforming the data.

Then, the data is split into training and testing sets using the train_test_split function to ensure accurate model evaluation.

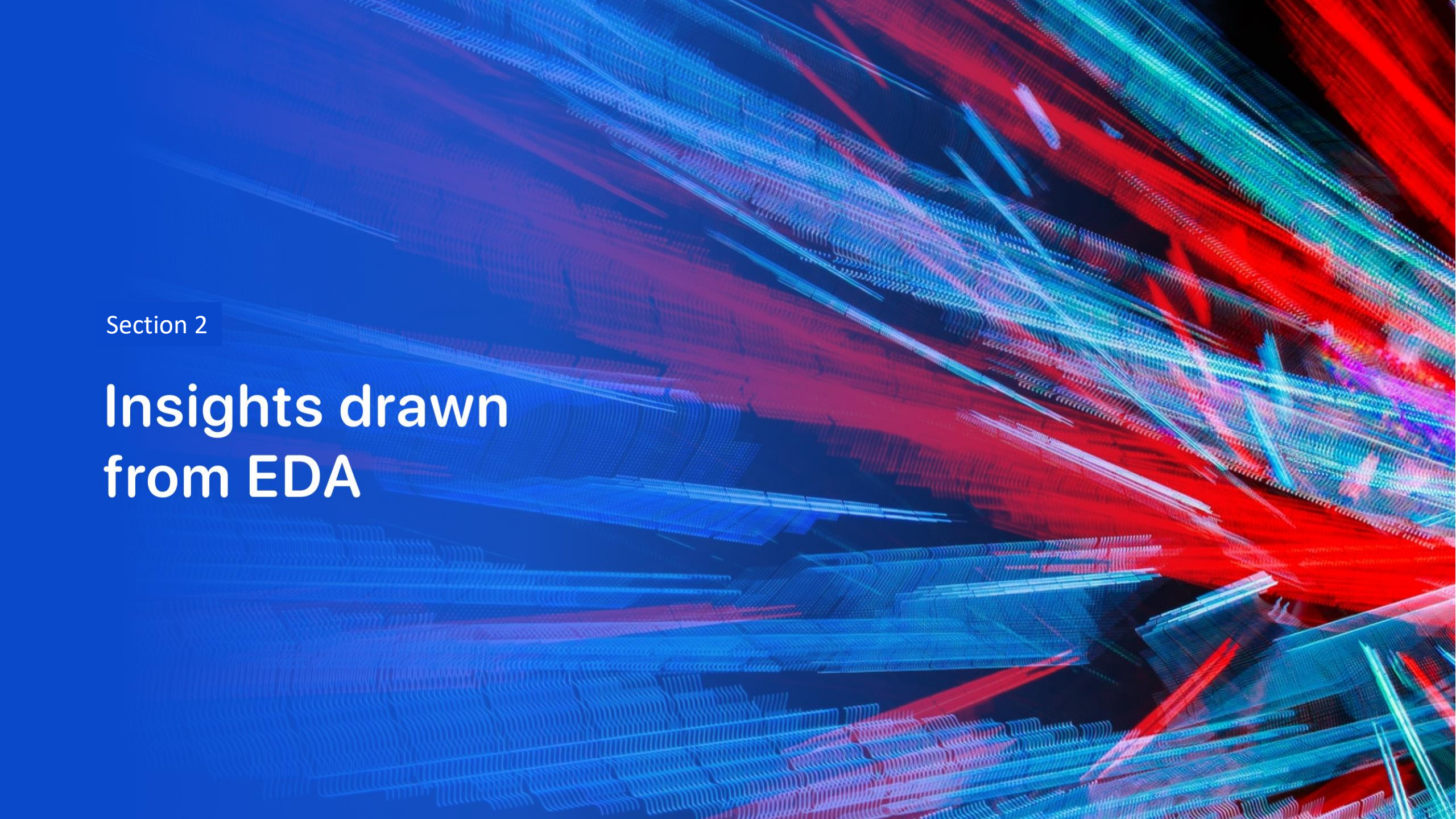
A GridSearchCV object is created with 10-fold cross-validation to perform an exhaustive search over specified parameter values for models like LogReg, SVM, Decision Tree, and KNN.

Performance is assessed using **Jaccard_score** and **F1_score metrics**, which evaluate *classification algorithms*.

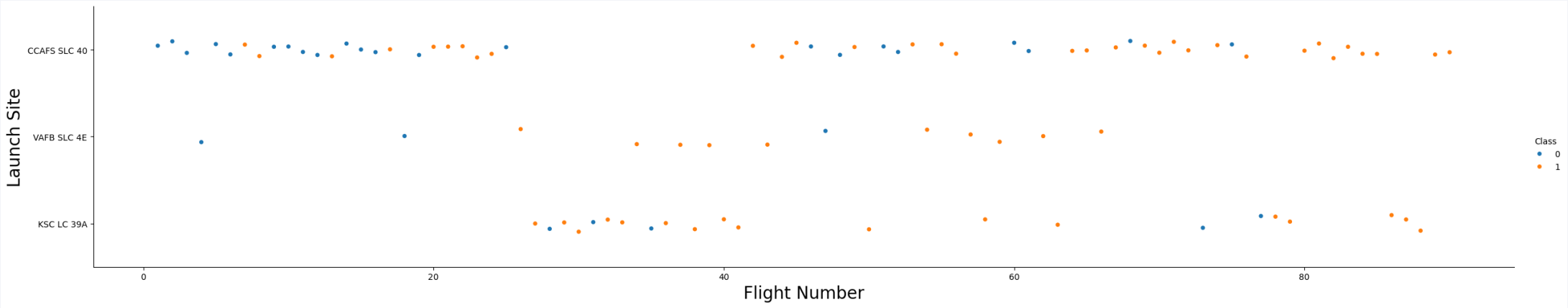
The confusion matrix is examined for a clearer understanding of model performance, particularly for multi-class classification. Finally, the models' accuracy on test data is calculated using the .score() method.

Results

- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results


Section 2
Insights drawn from EDA

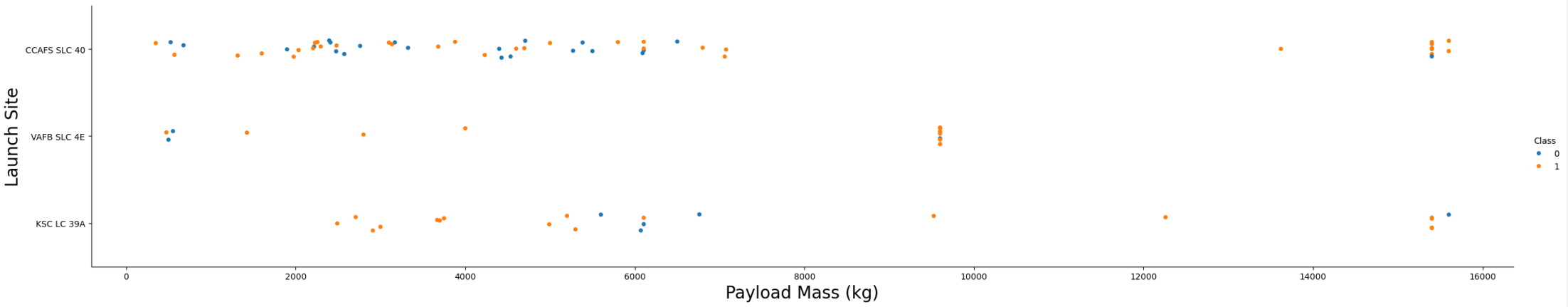
Flight Number vs. Launch Site



The data suggests that:

- The earliest launches had a failure rate, while the most recent ones were successful.
- The Cape Canaveral Space Force Station (CCAFS) Space Launch Complex 40 (SLC 40) has been involved in approximately half of the launches.
- The Vandenberg Space Force Base (VAFB) Space Launch Complex 4E and the Kennedy Space Center (KSC) Launch Complex 39A have demonstrated higher success rates. It's also observed that each subsequent launch has had a higher success rate.

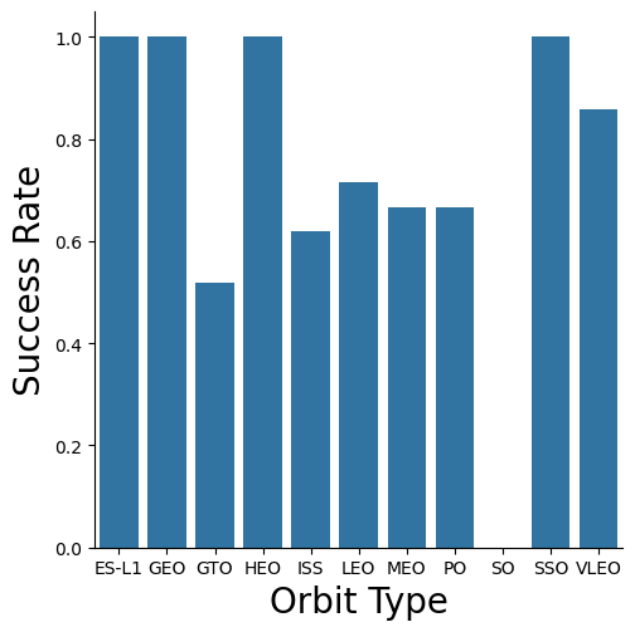
Payload vs. Launch Site



From the above chart it is evident that:

- For every launch site the higher the payload mass, the higher the success rate.
- Most of the launches with payload mass over 7000 kg were successful.
- KSC LC 39A has a 100% success rate for payload mass under 5500 kg too.

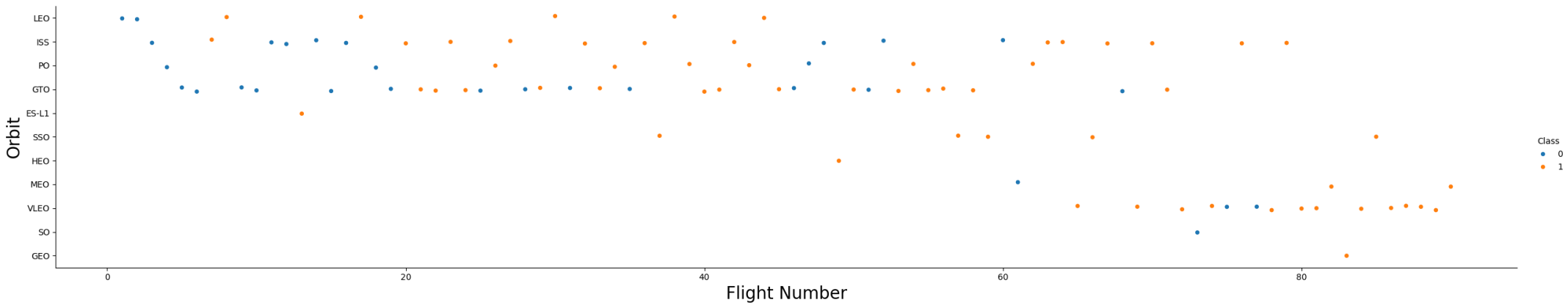
Success Rate vs. Orbit Type



The bar chart above shows:

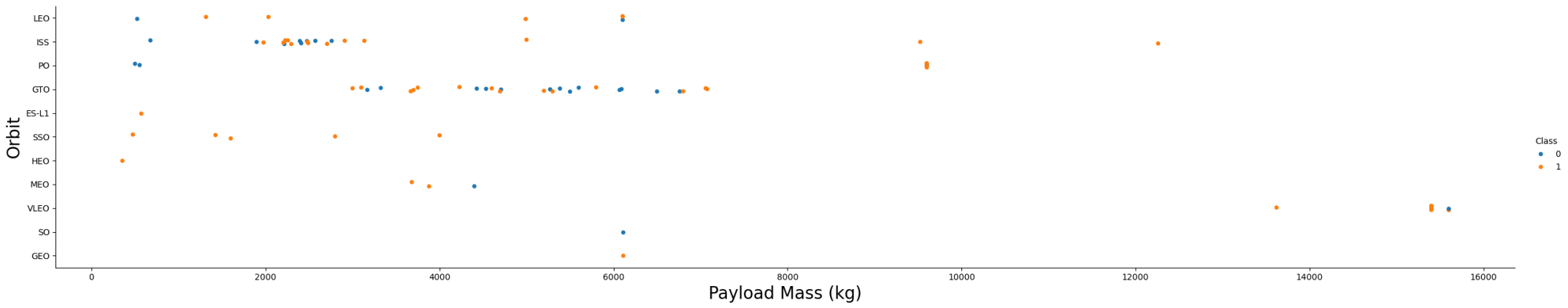
- The orbits with a 100% success rate include ES-L1, GEO, HEO, and SSO.
- Conversely, the orbit with a 0% success rate is SO.
- The orbits with a success rate between 50% and 85% include GTO, ISS, LEO, MEO, and PO.

Flight Number vs. Orbit Type



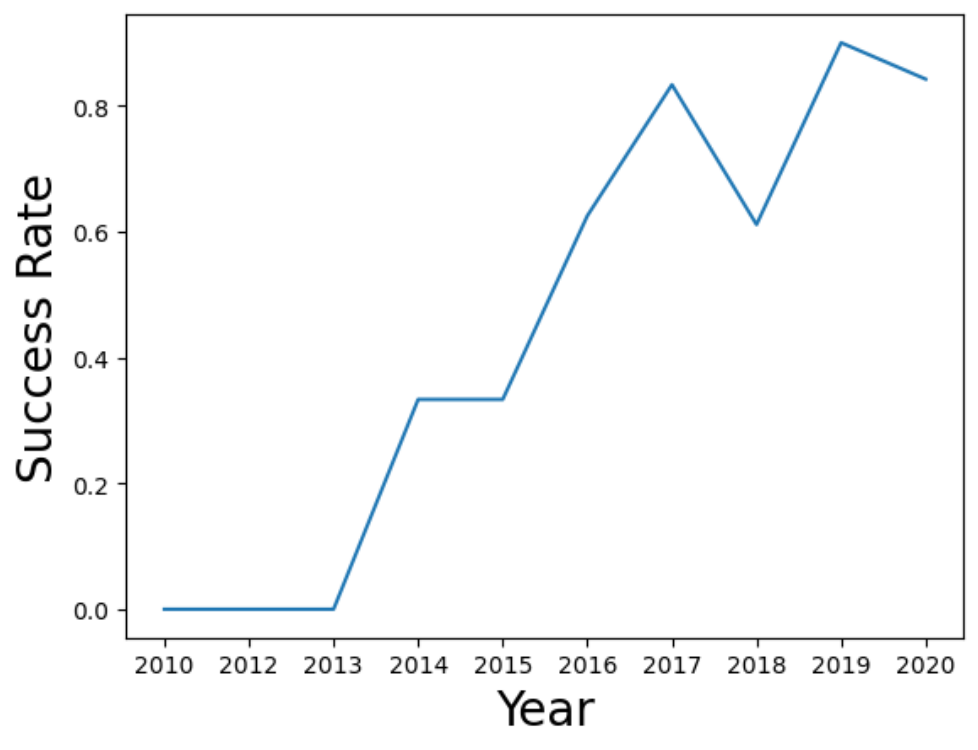
The line graph indicates a correlation between the number of flights and success in Low Earth Orbit (LEO). However, no such correlation is observed between flight number and success in Geostationary Transfer Orbit (GTO).

Payload vs. Orbit Type



It is evident from the above data that heavy payloads can have a detrimental effect on Geostationary Transfer Orbit (GTO) orbits, but they can be beneficial for Geostationary Transfer Orbit and Polar Low Earth Orbit (GTO and Polar LEO) orbits, such as those used by the International Space Station (ISS).

Launch Success Yearly Trend



The line graph suggests that from 2013 to 2020, the success rate has been consistently on the rise.

All Launch Site Names

```
%sql select distinct launch_site from SPACEXTABLE;
```

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

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

This SQL query retrieves unique values from the column "launch_site" in the table "SPACEXTABLE". It specifically selects distinct (unique) entries, meaning it won't include duplicates.

Launch Site Names Begin with 'CCA'

```
%sql select * from SPACEXTABLE where launch_site like 'CCA%' limit 5;
```

Python

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

This SQL retrieves data from the SPACEXTABLE table, specifically where records in the launch_site column start with the characters 'CCA‘.It utilizes the wildcard symbol (%) to achieve this. Additionally, the limit 5 clause restricts the output to only show the first 5 records that match the condition.

Total Payload Mass

```
%sql select sum(payload_mass__kg_) as total_payload_mass from SPACEXTABLE where customer = 'NASA (CRS)';
```

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

total_payload_mass
45596

This SQL query calculates the total payload mass for missions where the customer is 'NASA (CRS)'. The sum(payload_mass__kg_) function adds up the values in the payload_mass__kg_ column, and it assigns the result to a new column called total_payload_mass.

Average Payload Mass by F9 v1.1

```
%sql select avg(payload_mass__kg_) as average_payload_mass from SPACEXTABLE where booster_version like '%F9 v1.1%';
```

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

average_payload_mass
2534.6666666666665

This SQL calculates the average (avg) of the column payload_mass__kg_ and labels the result as average_payload_mass. The condition where booster_version like '%F9 v1.1%' filters the data, only including rows where the booster_version contains the substring "F9 v1.1".

First Successful Ground Landing Date

```
%sql select min(date) as first_successful_landing from SPACEXTABLE where "Landing__Outcome" = 'Success (ground pad)';
```

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

first_successful_landing
None

This SQL query searches for the first date in the SPACETABLE table (using the built-in SQL function ‘min’) where the landing outcome is a successful ground landing.

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql select booster_version from SPACEXTABLE where "Landing__Outcome" = 'Success (drone ship)' and payload_mass__kg_ between 4000 and 6000;
```

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

Booster_Version

This SQL query searches for the names of boosters from the SPACEXTABLE table which have success in drone ship and have payload mass greater than 4000 but less than 6000.

Total Number of Successful and Failure Mission Outcomes

```
%sql select mission_outcome, count(*) as total_number from SPACEXTABLE group by mission_outcome;
```

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

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

This SQL query lists the total number of successful and failure mission outcomes from the SPACEXTABLE table.

Boosters Carried Maximum Payload

```
%sql select booster_version from SPACEXTABLE where payload_mass__kg_ = (select max(payload_mass__kg_) from SPACEXTABLE);
```

```
 * 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

This SQL query lists the names of the booster versions which have carried the maximum payload mass through the use of a subquery.

2015 Launch Records

```
%%sql select substr(Date, 6,2) as month, date, booster_version, launch_site, "Landing__Outcome" from SPACEXTABLE
      where "Landing__Outcome" = 'Failure (drone ship)' and substr(Date, 0,5)='2015';
```

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

month	Date	Booster_Version	Launch_Site	"Landing__Outcome"

This SQL query lists the records which will display the month names, failed landing outcomes in drone ship, booster versions and launch sites for the months of the year 2015.

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

```
%%sql select "Landing__Outcome", count(*) as count_outcomes from SPACEXTABLE
      where date between '2010-06-04' and '2017-03-20'
      group by "Landing__Outcome"
      order by count_outcomes desc;
```

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

"Landing__Outcome"	count_outcomes
Landing__Outcome	32

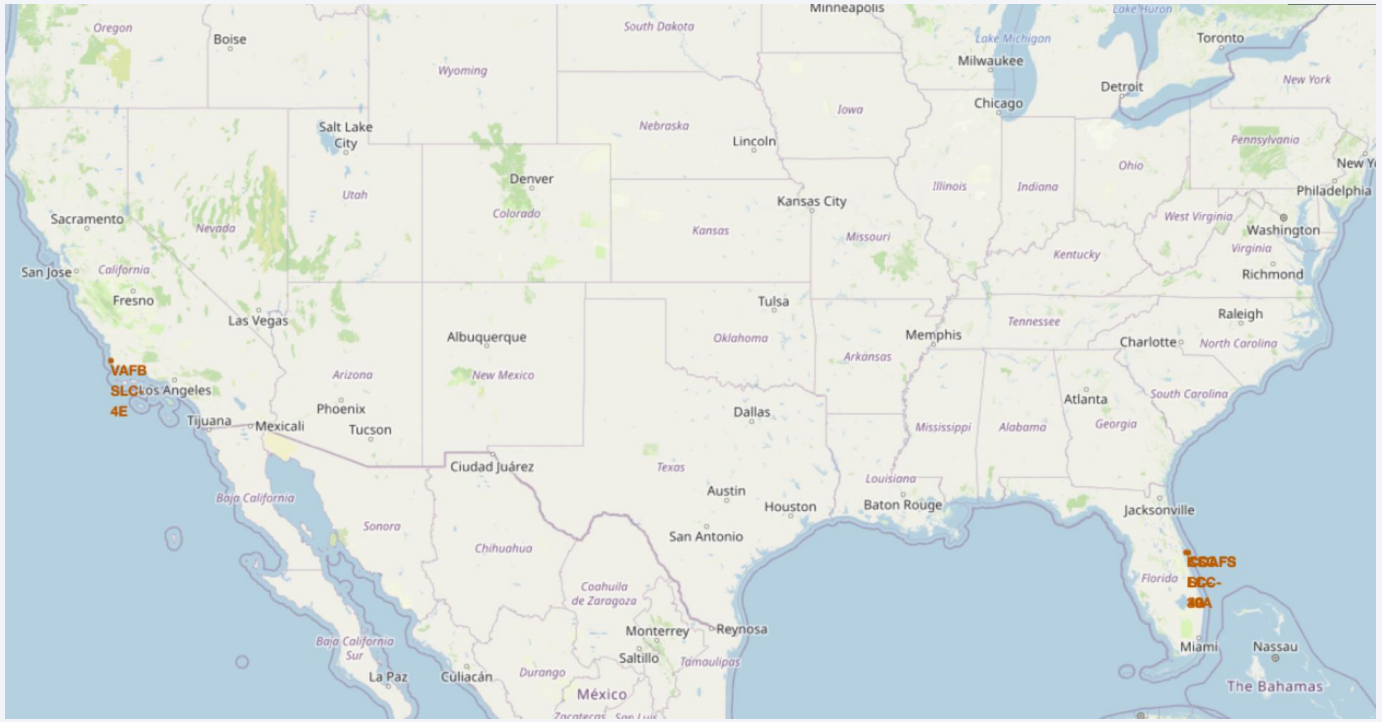
This SQL query ranks the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

Section 3

Launch Sites Proximities Analysis

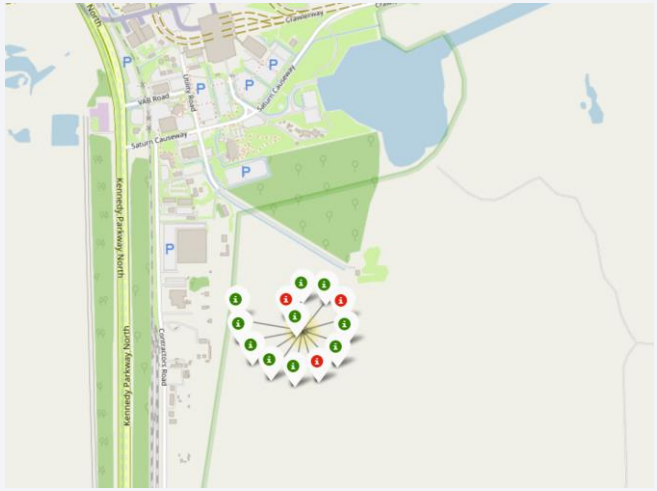
Global Launch Site Locations

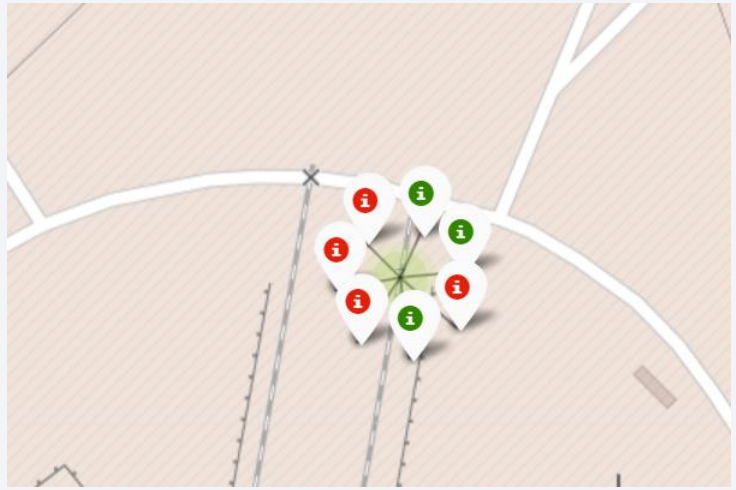
- Most rocket launch sites are situated near the equator due to the Earth's rotation. At the equator, the land moves at a speed of 1670 km/hour, which is faster than any other location on Earth. When a rocket is launched from the equator, it carries this speed into space, aiding in maintaining a sufficient velocity to stay in orbit. This is due to the principle of inertia, which prevents the rocket from slowing down once it has been launched.
- Launch sites are typically located close to the coast, which minimizes the risk of debris from the launch impacting populated areas. Launching rockets towards the ocean also reduces the risk of debris exploding near people.



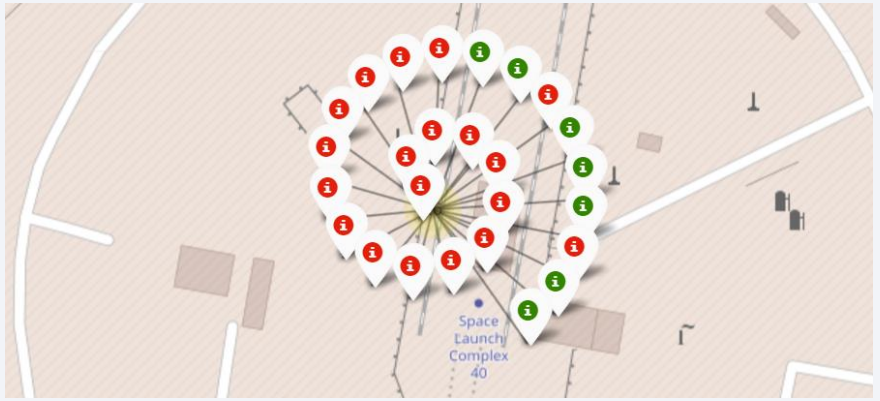
Map Analysis: Launch Outcomes and Findings

- The colored map markers illustrate the successful and failed launches of sites by utilizing green and red appropriately.



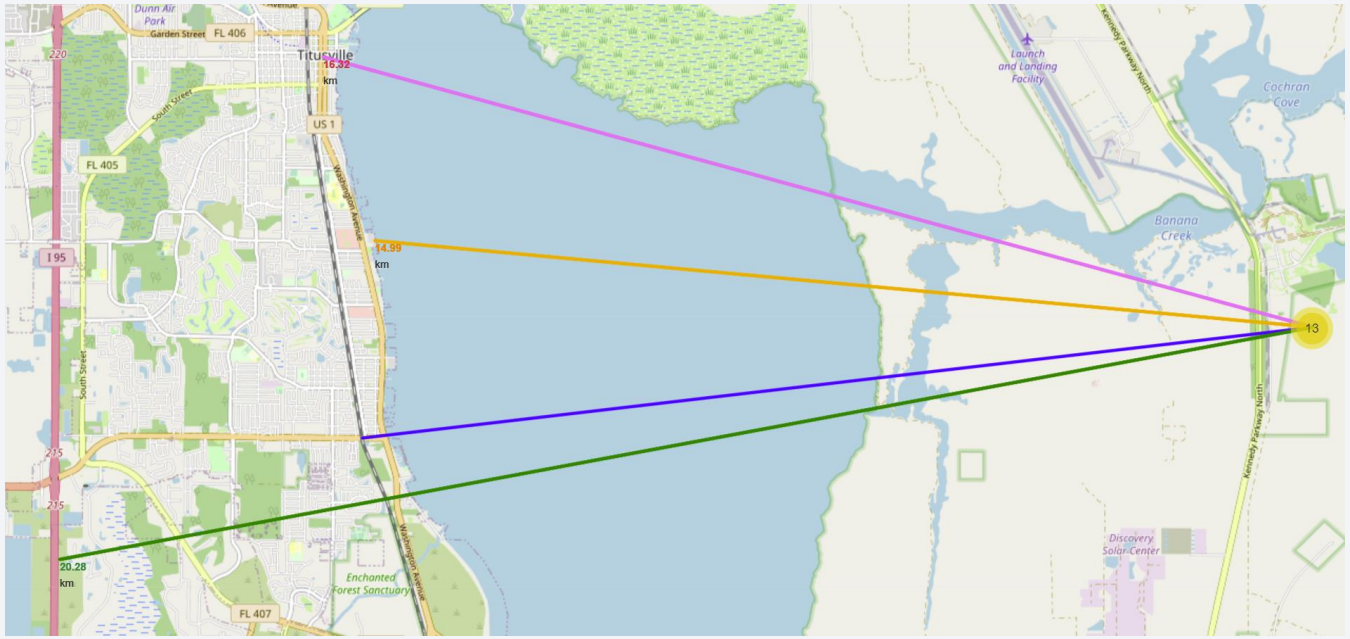


- Launch site KSC LC-39A has a very high success rate (top left) conversely launch site SLC-40 has a substantially lower success rate (bottom right).

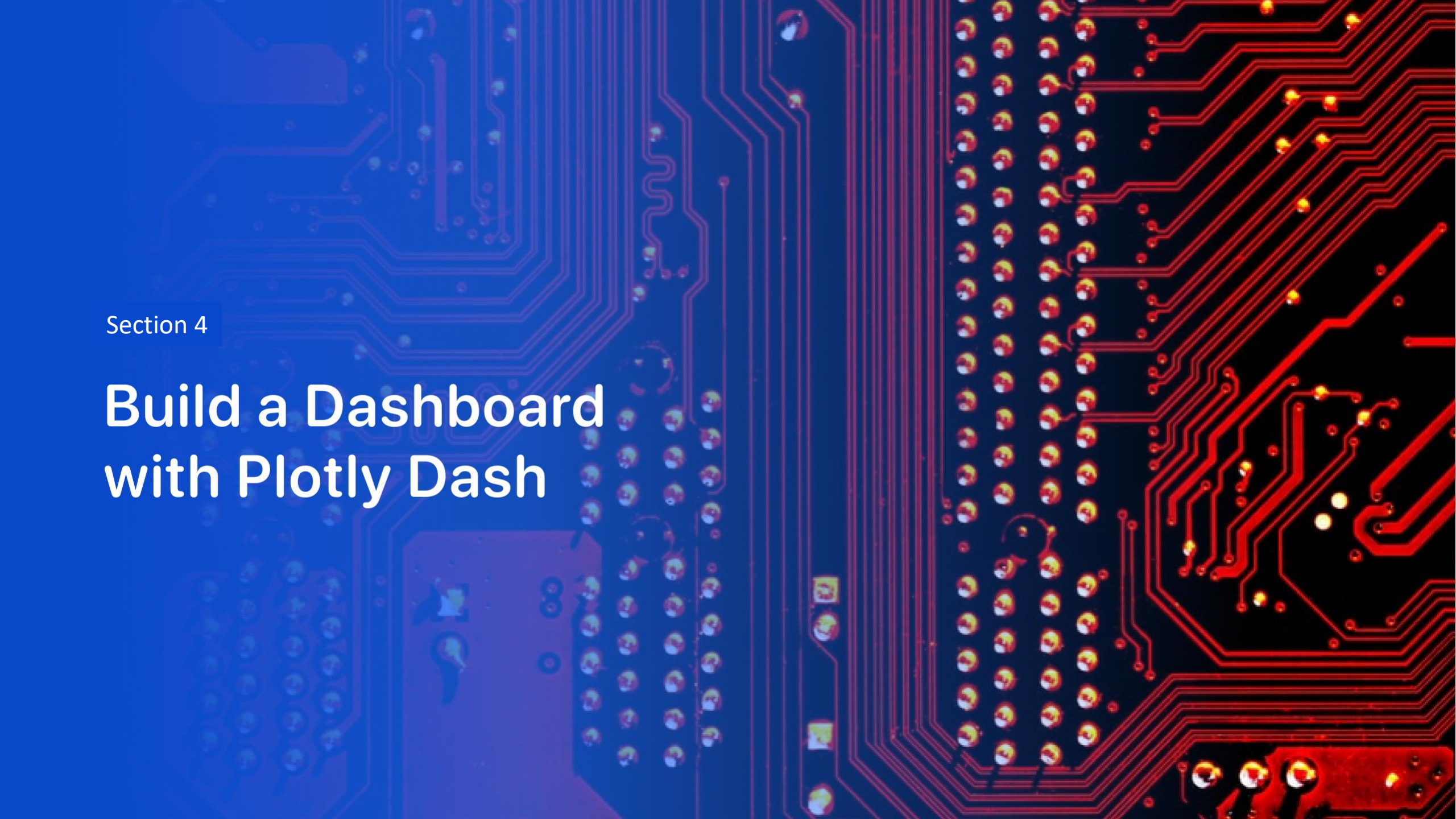


Launch Site Proximity Analysis (KSC LC-39A)

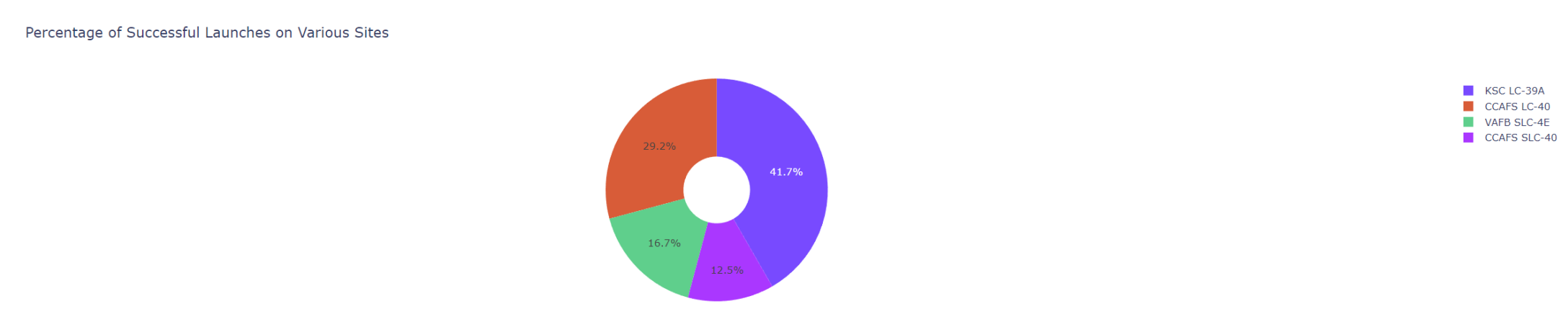
The Kennedy Space Center's Launch Complex 39A (LC-39A) is strategically located near various key features. It is approximately 15.23 km from a railway, 20.28 km from a highway, and 14.99 km from the coastline. Additionally, it is relatively close to its nearest city, Titusville, which is 16.32 km away.



In the event of a rocket failure, the high-speed debris could potentially reach populated areas within a 15-20 km radius in just a few seconds, highlighting the importance of safety measures and the need for effective disaster response plans.

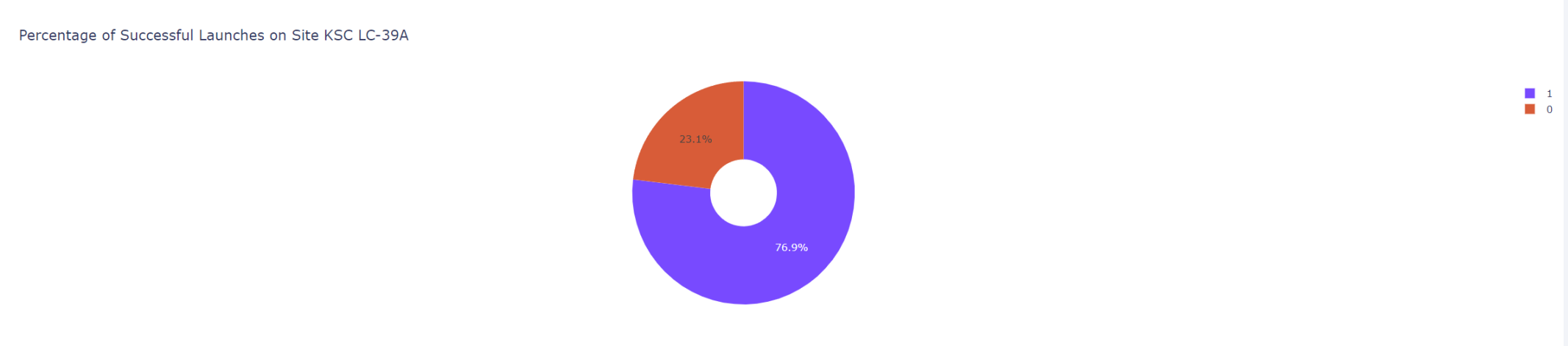

Section 4
Build a Dashboard with Plotly Dash

Dashboard Overview: Launch Success Count



The chart shows that from all the sites, KSC LC-39A has the most successful launches (41.7%).

Dashboard Overview: KSC LC-39A Launch Success Ratio



Kennedy Space Center's Launch Complex 39A boasts the highest launch success rate, with a staggering 76.9% success rate. This is demonstrated by the 10 successful launches and only 3 failed ones.

Dashboard Overview: Payload vs. Launch Outcome

The charts indicate that payloads within the range of 2000 to 5500 kg exhibit the highest rate of success.

Section 5

Predictive Analysis (Classification)

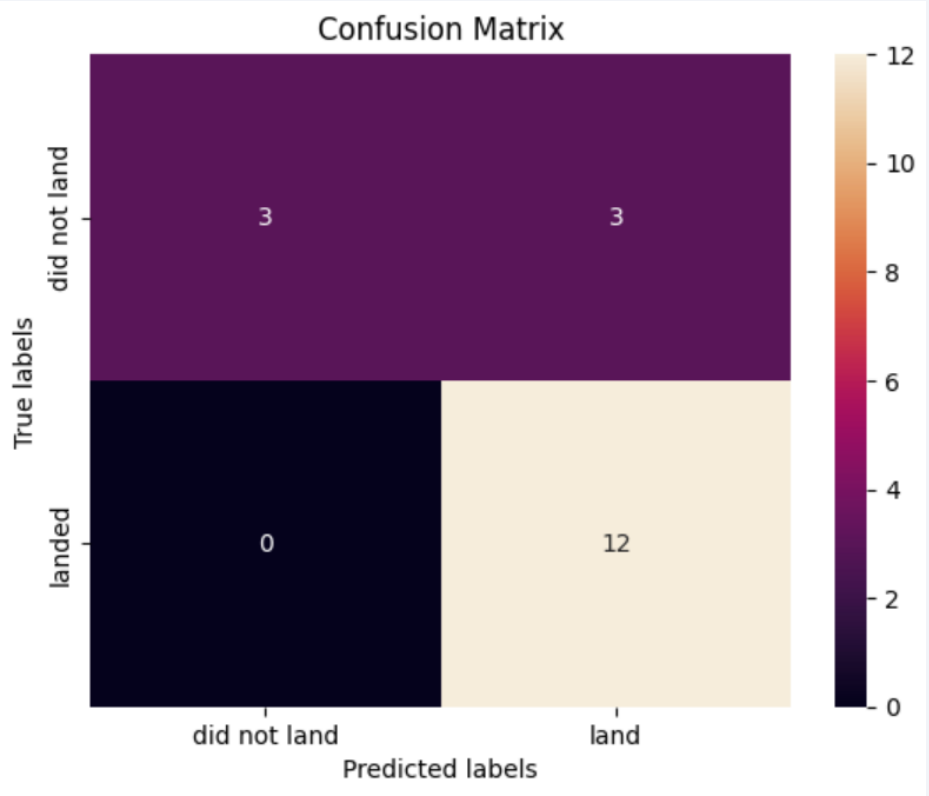
Classification Accuracy

```
x = []
y1 = []
y2 = []
for meth in comparison.keys():
    x.append(meth)
    y1.append(comparison[meth]['Accuracy'])
    y2.append(comparison[meth]['TestAccuracy'])


x_axis = np.arange(len(x))

plt.bar(x_axis - 0.2, y1, 0.4, label = 'Accuracy')
plt.bar(x_axis + 0.2, y2, 0.4, label = 'Test Accuracy')

plt.ylim([0,1])
plt.xticks(x_axis, x)

plt.xlabel("Methods")
plt.ylabel("Accuracy")
plt.title("Accuracy of Each Method")
plt.legend(loc='lower left')
plt.show()
```

The code execution resulted in errors, preventing the generation of a chart. However, the decision tree model demonstrated the highest accuracy among the tested models. This suggests that despite the errors, the decision tree model would likely yield the highest classification accuracy if it were to run successfully.

Confusion Matrix

Observing the confusion matrix, it becomes evident that logistic regression is capable of discerning between the distinct classes. It is apparent that the primary issue lies in the occurrence of false positives



Conclusions

- The Decision Tree Model is the most suitable algorithm for this dataset.
- Launches with a low payload mass show better results than launches with a larger payload mass due to various factors such as fuel efficiency, payload capacity, and the cost of the launch.
- Most of the launch sites are in proximity to the Equator line and all the sites are in very close proximity to the coast. This might be due to the gravitational pull of the Earth at the Equator, which could make launches easier and more cost-effective.
- The success rate of launches increases over the years. This could be due to advancements in technology, improvements in launch procedures, or increased experience and expertise in the field.
- KSC LC-39A has the highest success rate of the launches from all the sites. This is due to its location, facilities, and the specific launch conditions at this site.
- Orbits ES-L1, GEO, HEO and SSO have 100% success rate. This is due to the specific characteristics of these orbits, such as their altitude, inclination, or the type of launch vehicle used.

Appendix

Error encountered during the completion of task four in the machine learning prediction lab:

TASK 4

Create a logistic regression object then create a GridSearchCV object `logreg_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

```
parameters ={'C':[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}
```

```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()
# Instantiate the GridSearchCV object: logreg_cv
logreg_cv = GridSearchCV(lr, parameters, cv=10)

# Fit it to the data
logreg_cv.fit(X_train, Y_train)
```

```
/lib/python3.11/site-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/lib/python3.11/site-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/lib/python3.11/site-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/lib/python3.11/site-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/lib/python3.11/site-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Thank you!

Special Thanks to:
Instructors
Coursera
IBM