WINNING SPACE RACE WITH DATA SCIENCE

Alberto Gonzaga Ramiro
September 2024

Outline

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

Executive Summary

- Summary of Methodologies:
    1. Data Collection
    2. Data Wrangling
    3. Exploratory Data Analysis
    4. Interactive Visual Analytics
    5. Classification
- Summary of Results:
    1. Launch Succes Rate
    2. Payload Impact
    3. Model Accuracy
    4. Geospatial Findings
    5. Time-based Trends

Introduction

- Background and Context:
  Since 1957, the competition for space exploration has intensified, with typical launch costs averaging $165 million. SpaceX has disrupted this market by reducing Falcon 9 launch costs to about $62 million, thanks to its innovative technology that allows the first stage of the rocket to land and be reused. This advancement makes space access more affordable and enhances SpaceX’s competitive edge.
- Find Answers:
  This project aims to predict whether the Falcon 9’s first stage will land successfully. By examining key factors such as payload mass, launch site, and rocket orbit, we seek to identify the variables that impact landing success. These insights will help assess launch costs and determine if competitors should challenge SpaceX's pricing.

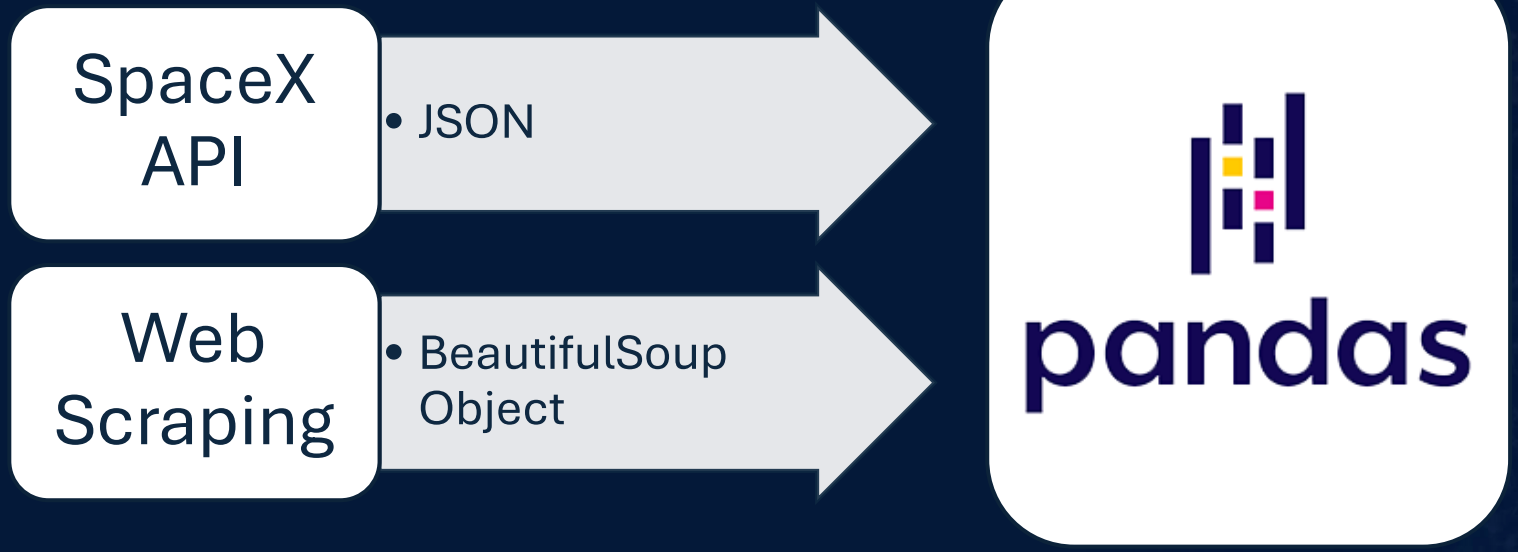
Methodology

Executive Summary

- **Data Collection Methodology:**
  Data was sourced from the SpaceX REST API and web scraping for additional details.
- **Data Wrangling:**
  The data was cleaned by addressing missing values and standardizing formats for analysis.
- **Exploratory Data Analysis (EDA):**
  Visualizations and SQL queries were used to identify trends and relationships in the data.
- **Interactive Visual Analytics:**
  Interactive maps were created with Folium, and a dynamic dashboard was developed using Plotly Dash.
- **Predictive Analysis:**
  Classification models were built and evaluated, with the best model optimized for accuracy

Data Collection

- Two main sources:
  - SpaceX API
  - WebScrapping



Data Collection – SpaceX API

- 1. Get method

```
1  spacex_url="https://api.spacexdata.com/v4/launches/past"
2  response = requests.get(spacex_url)
```

- 2. JSON response file to Pandas Dataframe

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

GitHub code

Data Collection – SpaceX API

- 3. Data stored in lists to create a new dataframe

```
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

GitHub code

Data Collection – SpaceX API

- 4. Combine columns into a dictionary to construct a new dataframe

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

```
launch_df = pd.DataFrame(launch_dict)
```

Data Collection – SpaceX API

- 5. Filter the dataframe to only include Falcon 9 launches

```
data_falcon9 = launch_df[launch_df['BoosterVersion'] != 'Falcon 1']
```

- 6. Replace missing values with the mean

```
# Calculate the mean value of PayloadMass column
mean_payload_mass = launch_df['PayloadMass'].mean()
# Replace the np.nan values with its mean value
launch_df['PayloadMass'].replace(np.nan, mean_payload_mass, inplace=True)
```

GitHub code

Data Collection – Web Scraping

- 1. Get method

```
response = requests.get(static_url)
```

- 2. Create BeautifulSoup object

```
soup = BeautifulSoup(response.text, 'html.parser')
```

GitHub code

Data Collection – Web Scraping

- 3. Create a data frame by parsing the launch HTML tables

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

GitHub code

Data Wrangling

- 1. Calculate the number of launches on each site

```python
import pandas as pd
import numpy as np
# Apply value_counts() on column LaunchSite
launches_per_site = df['LaunchSite'].value_counts()
print(launches_per_site)
```

GitHub code

Data Wrangling

- 2. Calculate the number and occurrence of each orbit

```
# Apply value_counts on Orbit column
orbit_counts = df['Orbit'].value_counts()
print(orbit_counts)
```

- 3. Calculate the number and occurrence of mission outcome of the orbits

```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
print(landing_outcomes)
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
print(bad_outcomes)
```

GitHub code

Data Wrangling

- 4. Create a landing outcome label from Outcome column

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = [0 if outcome in bad_outcomes else 1 for outcome in df['Outcome']]
df['Class']=landing_class
df[['Class']].head(8)
df["Class"].mean()
# Contar los aterrizajes fallidos
failed_landings_count = df['Outcome'].isin(bad_outcomes).sum()
print(f"Total de aterrizajes fallidos: {failed_landings_count}")
```

GitHub code

Exploratory Data Analysis (EDA) - Visualization

Scatter Charts

- Flight Number vs. Launch Site
- Payload vs. Launch Site
- Orbit Type vs. Flight Number
- Payload vs. Orbit Type

Bar Chart

- Success Rate & Orbit Type

Line Chart

- Success Rate & Year

GitHub code

Exploratory Data Analysis (EDA) - SQL

- Objectives to achieve and data to analyze:
    1. Display the names of the unique launch sites in the space mission.
    2. Display 5 records where launch sites begin with the string.
    3. Display the total payload mass carried by boosters launched by NASA (CRS).
    4. Display the average payload mass carried by booster version F9 v1.1.
    5. List the date when the first successful landing outcome in a ground pad was achieved.
    6. List the date when the first successful landing outcome in a ground pad was achieved.
    7. List the total number of successful and failure mission outcomes.
    8. List the names of the booster versions which have carried the maximum payload mass. Use a subquery.
    9. List the records which display the month names, failure landing outcomes in a drone ship, booster versions, and launch sites for the months in the year 2015.
    10. 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.

GitHub code

Interactive Map with Folium

- To visualize the data on an interactive map, the following steps are taken:
  - Mark all launch sites on a map.
  - Mark the successful and failed launches for each site on the map.
  - Calculate the distances between a launch site and its proximities.

GitHub code

Predictive Analysis (Classification)

Load the dataset

- Prepare the data for model development.

Perform data transformations

- Standardize and pre-process the data.

Split the data into training and testing sets

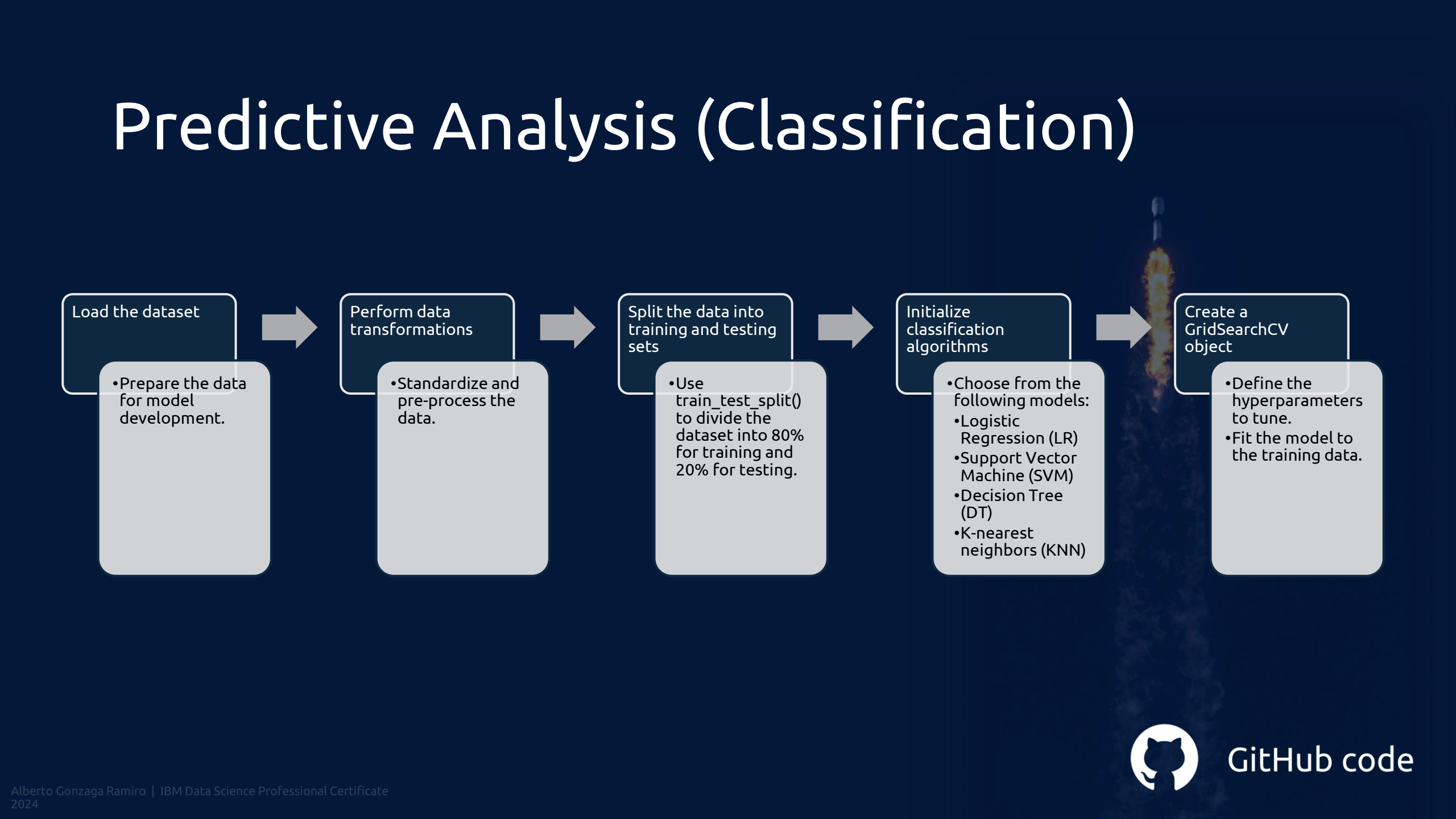
- Use train_test_split() to divide the dataset into 80% for training and 20% for testing.

Initialize classification algorithms

- Choose from the following models:
  - Logistic Regression (LR)
  - Support Vector Machine (SVM)
  - Decision Tree (DT)
  - K-nearest neighbors (KNN)

Create a GridSearchCV object

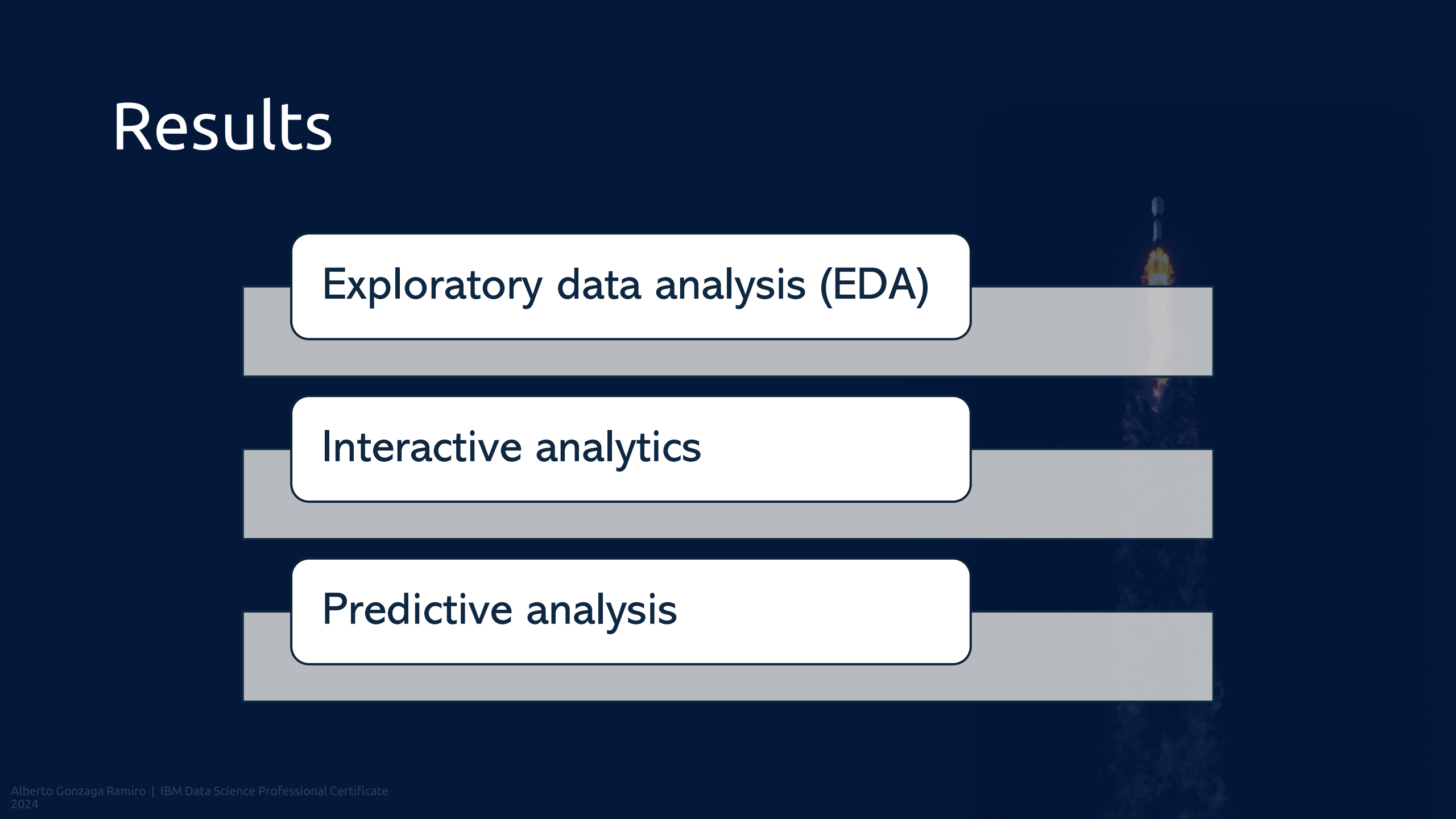
- Define the hyperparameters to tune.
- Fit the model to the training data.



GitHub code

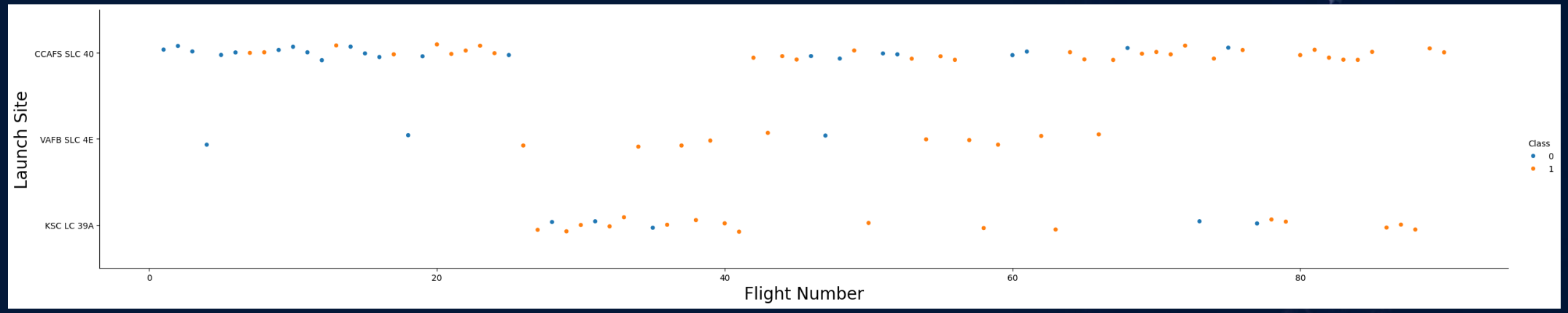
Results

Results



- Exploratory data analysis (EDA)
- Interactive analytics
- Predictive analysis

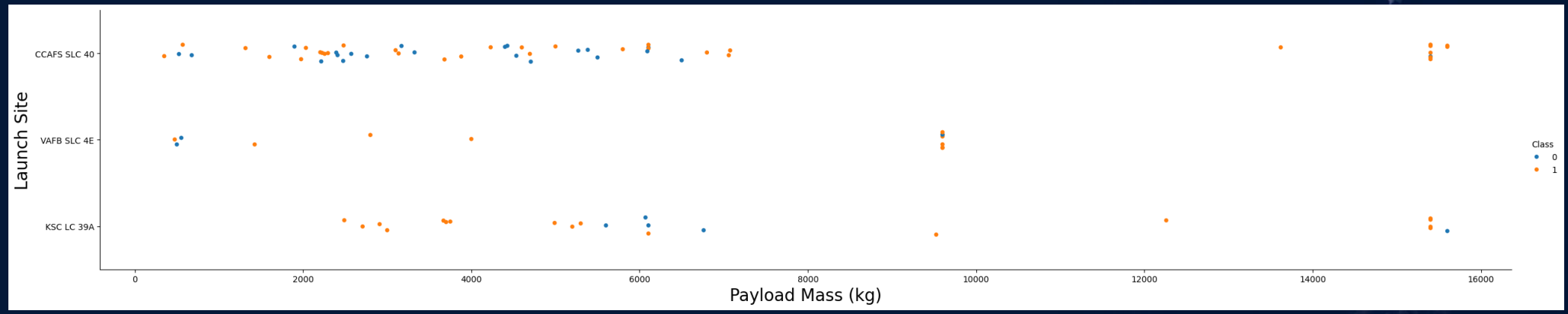
EDA - Flight Number vs. Launch Site



- Launch Site Performance and Trends
  - CCAFS SLC 40: Most active site, but had a lower success rate in earlier flights.
  - VAFB SLC 4E: Shows potential with a higher success rate, though used less frequently.
  - KSC LC 39A: Higher success rate due to a lack of early unsuccessful flights.
- Key Insights:
  - Success rates generally improve with more launches.
  - Early flights had lower success, especially from CCAFS and VAFB.
  - The increase in successful launches after flight number 30 indicates improved performance and experience over time.

GitHub code

EDA – Payload vs. Launch Site

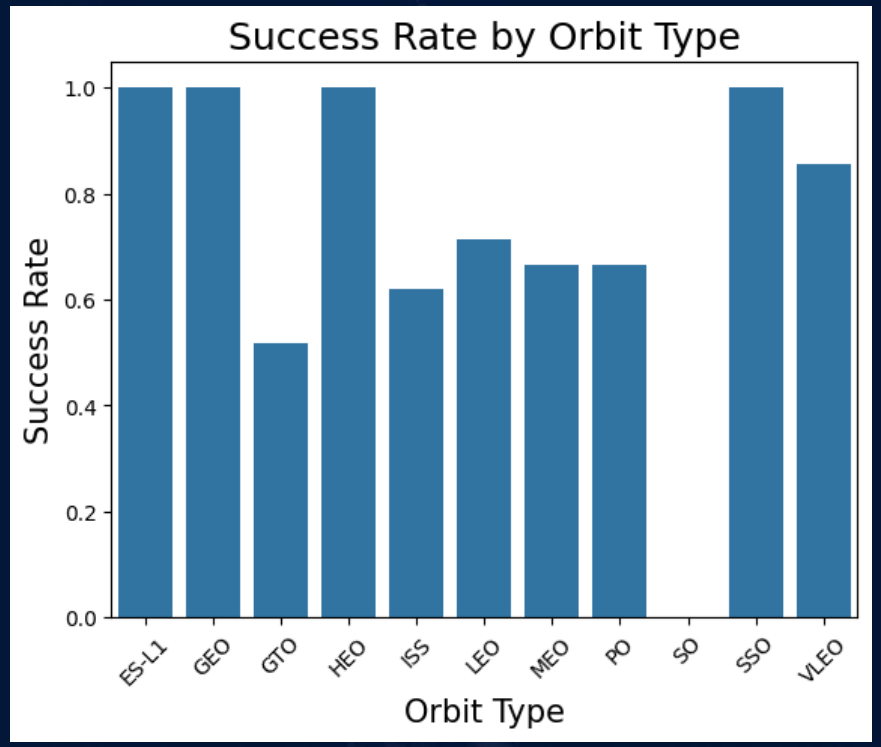


- Payload Mass and Success Insights
  - No strong correlation between payload mass and the success of the first stage return. Both successful and failed landings occurred across various payload weights.
  - Payload Mass Above 7000 kg:
    - Fewer unsuccessful landings, but also limited data for heavier launches.
- Launch Site and Payload Mass Trends
  - No clear relationship between payload mass and success rate across launch sites.
  - CCAFS SLC 40: Primarily lighter payloads, with some outliers for heavier launches.
  - All sites handled a variety of payload masses, but trends in success are not linked directly to the weight of the payload.

GitHub code

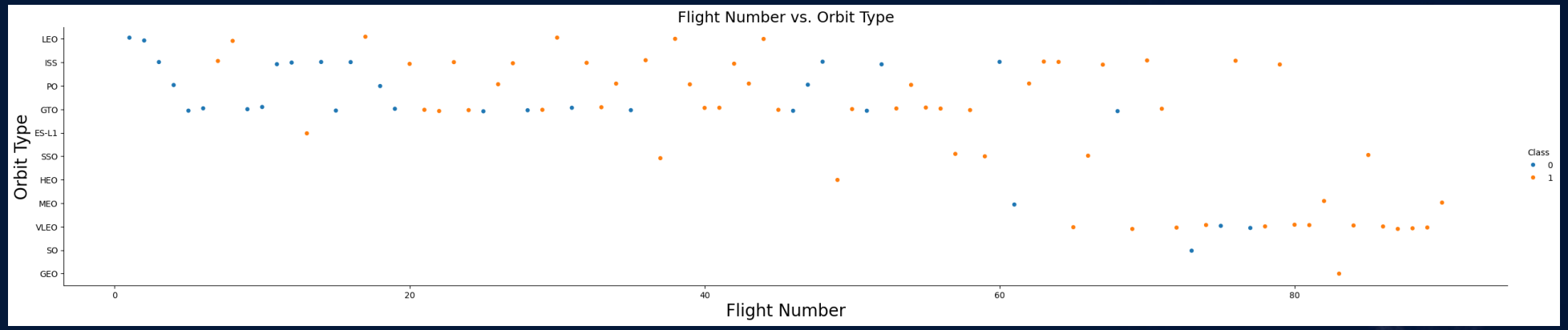
EDA – Success Rate vs. Orbit Type

- Orbit Performance Insights
  - Best Orbits (100% success rate) for first stage returns:
    - ES-L1 (Earth-Sun First Lagrangian Point)
    - GEO (Geostationary Orbit)
    - HEO (High Earth Orbit)
    - SSO (Sun-synchronous Orbit)
- Worst Orbit:
  - GTO (Geostationary Transfer Orbit) has the lowest success rate, requiring further analysis to understand the failures.
- Key Takeaways:
  - Consistent success in specific orbits highlights potential for reliable first stage returns.
  - GTO and SO (Heliocentric Orbit) show no successful returns, indicating areas for improvement or risk mitigation.



GitHub code

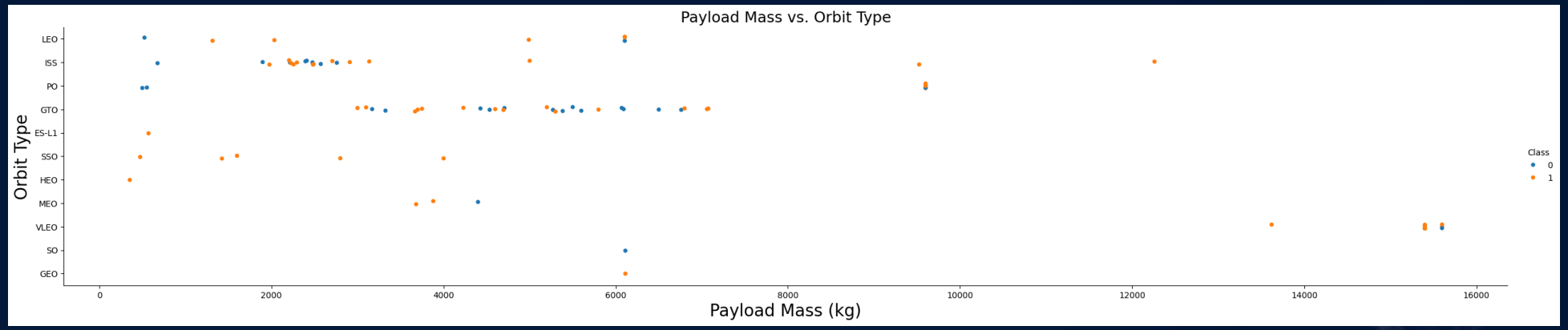
EDA – Flight Number vs. Orbit Type



- Orbit Type and Flight Number Insights
  - LEO Orbit: Success is closely tied to the number of flights—early flights had lower success rates, but success improves as flight numbers increase.
  - GTO Orbit: Shows no clear relationship between flight number and success rate, regardless of the number of flights.
- Key Findings from the Scatter Plot:
  - 100% Success Rate:
    - GEO, HEO, ES-L1: Only 1 flight each, explaining the perfect success rate.
    - SSO: More impressive, with 5 successful flights.
  - General Trend: As flight number increases, so does the success rate, especially in LEO where unsuccessful landings are mostly seen in early launches.

GitHub code

EDA – Payload vs. Orbit Type

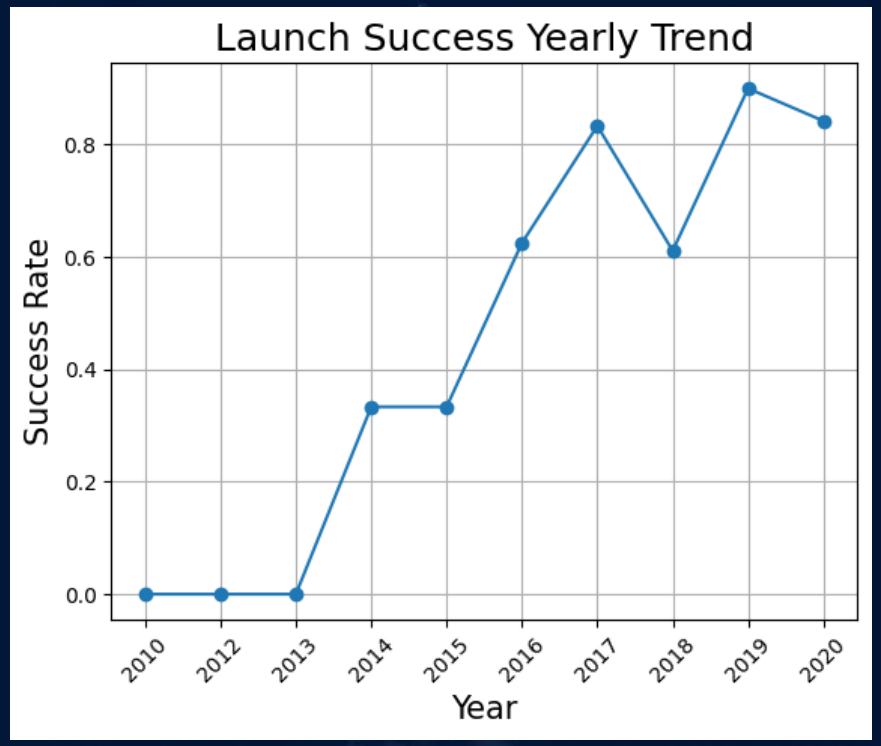


- Payload Mass and Orbit Type Insights
  - Heavy Payloads:
    - Positive influence on ISS, LEO, and Polar (PO) orbits, leading to higher success rates.
    - Negative influence observed in GTO orbits, where success is less consistent with heavier payloads.
- Key Findings from the Scatter Plot:
  - Heavy payloads perform better in ISS, LEO, and PO orbits, despite limited data points for PO.
  - In GTO, the relationship between payload mass and success rate remains unclear.
  - VLEO launches are generally associated with heavier payloads, which aligns with expectations.

GitHub code

EDA – Launch Success Yearly Trend

- Success Rate Over Time
- 2010-2013:
  - All landings were unsuccessful (0% success rate).
- 2013 Onward:
  - The success rate steadily increased, with minor dips in 2018 and 2020.
  - After 2016, the success rate remained above 50%.
- Key Trend:
  - The overall success rate has shown consistent improvement, peaking in recent years, despite occasional fluctuations.



GitHub code

EDA – All Launch Site Names

- Names of the unique launch sites in the space mission:

```
query = '''
    SELECT DISTINCT "Launch_Site"
    FROM SPACEXTBL;
'''
```

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

GitHub code

EDA – Launch Site Names Begin with ‘CCA’

- 5 records where launch sites begin with ‘CCA’

```
query = '''
    SELECT *
    FROM SPACEXTBL
    WHERE "Launch_Site" LIKE 'CCA%'
    LIMIT 5;
'''
```

```
 'CCAFS LC-40',
 'CCAFS LC-40',
'CCAFS LC-40',
'CCAFS LC-40',
 'CCAFS LC-40',
```

GitHub code

EDA – Average Payload Mass by F9 v1.1

- Average payload mass carried by booster version F9 v1.1

```
query = '''
    SELECT AVG("PAYLOAD_MASS__KG_")
    FROM SPACEXTBL
    WHERE "Booster_Version" = "F9 v1.1";
'''
```

```
Average Payload Mass carried by F9 v1.1 boosters: 2928.4 kg
```

GitHub code

EDA – First Successful Ground Landing Date

- Date of the first successful landing outcome on ground pad

```
query = '''
    SELECT MIN("Date")
    FROM SPACEXTBL
    WHERE "Landing_Outcome" = "Success (ground pad)";
'''
```

```
The first successful landing on ground pad was achieved on: 2015-12-22
```

GitHub code

EDA – Successful Drone Ship Landing with Payload between 4000 and 6000

- Names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

```
query = '''
    SELECT "Booster_Version"
    FROM SPACEXTBL
    WHERE "Landing_Outcome" = "Success (drone ship)"
    AND "PAYLOAD_MASS__KG_" > 4000
    AND "PAYLOAD_MASS__KG_" < 6000;
'''
```

```
Boosters with successful landing on drone ship and payload mass between 4000 and 6000 kg:
F9 FT B1022
F9 FT B1026
F9 FT  B1021.2
F9 FT  B1031.2
```

GitHub code

EDA – Total Number of Successful and Failure Mission Outcomes

- Total number of successful and failure mission outcomes

```
query = '''
    SELECT "Landing_Outcome", COUNT(*) as "Count"
    FROM SPACEXTBL
    GROUP BY "Landing_Outcome";
'''
```

```
Total number of successful and failure mission outcomes:
Controlled (ocean): 5
Failure: 3
Failure (drone ship): 5
Failure (parachute): 2
No attempt: 21
No attempt : 1
Precluded (drone ship): 1
Success: 38
Success (drone ship): 14
Success (ground pad): 9
Uncontrolled (ocean): 2
```

GitHub code

EDA – Boosters Carried Maximum Payload

- Names of the booster which have carried the maximum payload mass

```
query = '''
    SELECT "Booster_Version"
    FROM SPACEXTBL
    WHERE "PAYLOAD_MASS__KG_" = (
        SELECT MAX("PAYLOAD_MASS__KG_")
        FROM SPACEXTBL
    );
'''
```

```
Booster versions which have carried the maximum payload mass:
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
```

GitHub code

EDA – 2015 Launch Records

- Failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

```
query = '''
    SELECT
        SUBSTR("Date", 6, 2) AS month,
        "Landing_Outcome",
        "Booster_Version",
        "Launch_Site"
    FROM SPACEXTBL
    WHERE "Landing_Outcome" = "Failure (drone ship)"
    AND SUBSTR("Date", 0, 5) = '2015';
'''
```

```
Records of failed landing outcomes in drone ship for the months in the year 2015:
Month: 01, Landing Outcome: Failure (drone ship), Booster Version: F9 v1.1 B1012, Launch Site: CCAFS LC-40
Month: 04, Landing Outcome: Failure (drone ship), Booster Version: F9 v1.1 B1015, Launch Site: CCAFS LC-40
```

GitHub code

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

- 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

```
query = '''
    SELECT "Landing_Outcome", COUNT(*) as "Count"
    FROM SPACEXTBL
    WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20'
    GROUP BY "Landing_Outcome"
    ORDER BY "Count" DESC;
'''
```

```
Count of landing outcomes between 2010-06-04 and 2017-03-20 (in descending order):
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
```

GitHub code

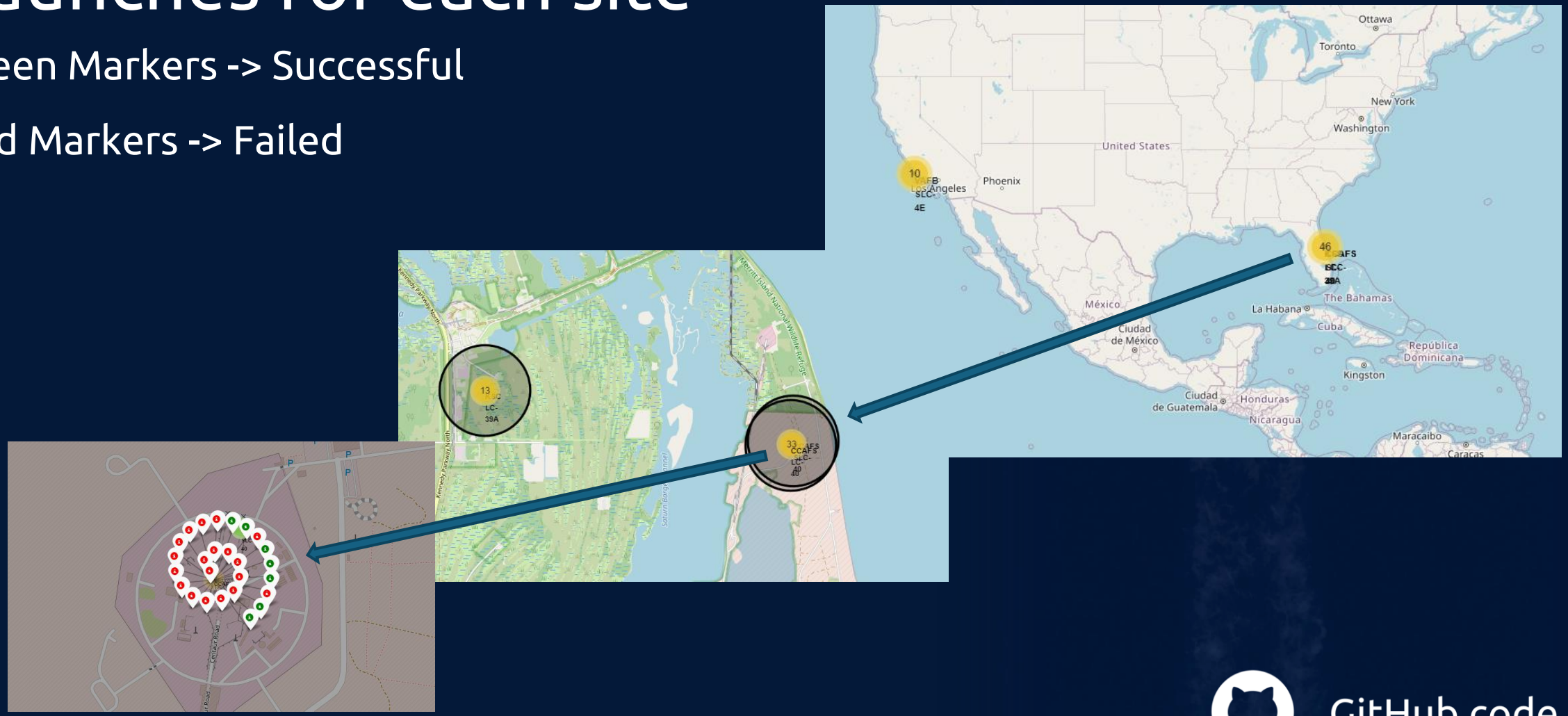
Interactive Analytics – All Launch Site Names

- Launch Site Locations
  - All launch sites are located near the coast and close to the Equator line.
  - Florida and California are the two states where SpaceX launch sites are distributed.
- Key Insights:
  - Proximity to water and the zeroth latitude helps reduce risks and avoid accidents during launches.
  - Coastal locations are chosen for safety and optimal launch conditions.

GitHub code

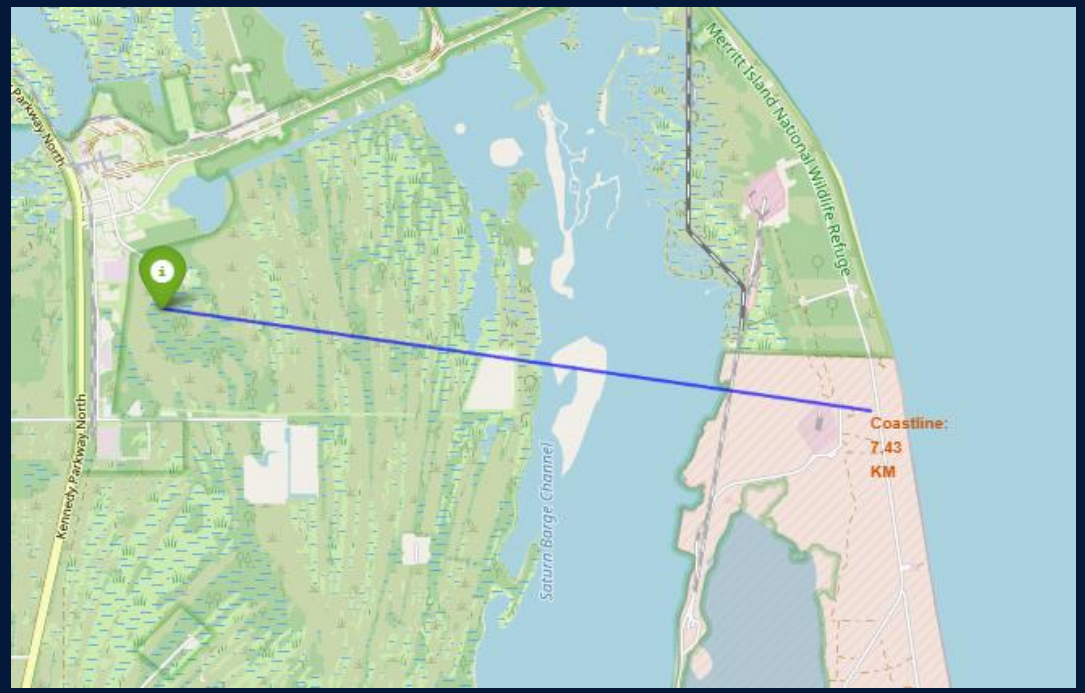
Interactive Analytics – Success/Failed launches for each site

- Green Markers -> Successful
- Red Markers -> Failed



GitHub code

Interactive Analytics – Distances between a launch site to its proximities

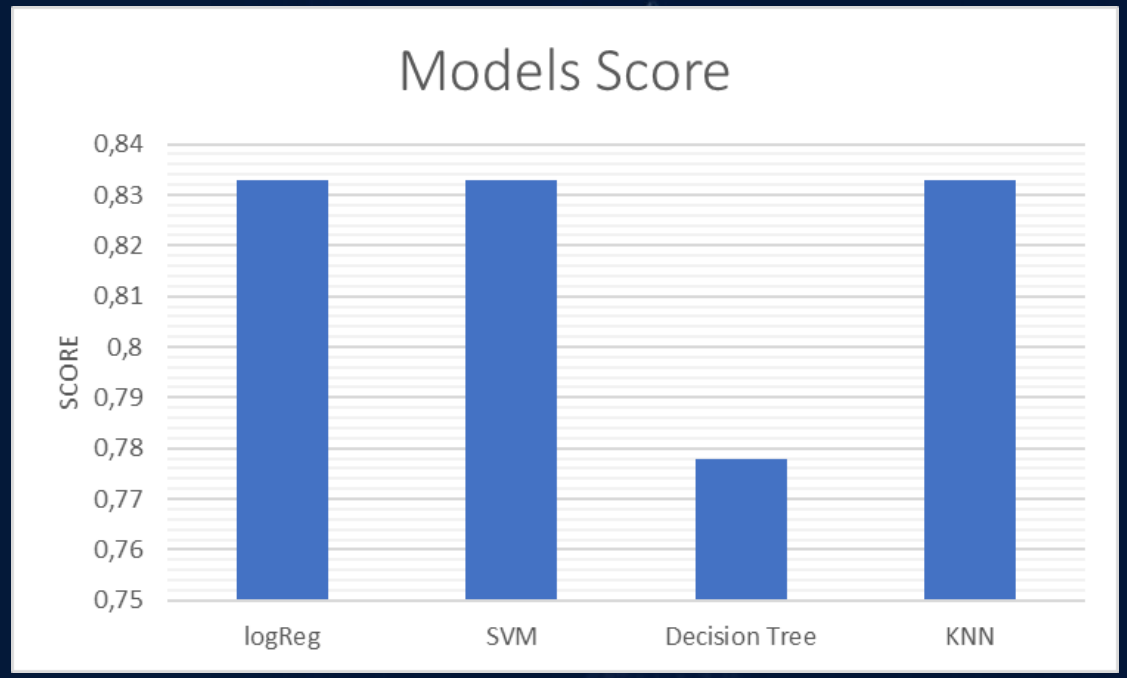



```python
# Create a marker with distance to a closest city, railway, highway, etc.
# Draw a line between the marker to the launch site
import folium
from folium import DivIcon, Marker
from folium.plugins import MarkerCluster
from math import sin, cos, sqrt, atan2, radians

# Función para calcular la distancia entre dos puntos
def calculate_distance(lat1, lon1, lat2, lon2):
    R = 6373.0  # Radio de la Tierra en km
    lat1, lon1, lat2, lon2 = map(radians, [lat1, lon1, lat2, lon2])

    dlon = lon2 - lon1
    dlat = lat2 - lat1

    a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
    c = 2 * atan2(sqrt(a), sqrt(1 - a))

    distance = R * c
    return distance

# Coordenadas del sitio de lanzamiento (KSC LC-39A)
launch_site_lat = 28.573255
launch_site_lon = -80.646895

# Coordenadas de la costa más cercana (cambia a las coordenadas reales)
coastline_lat = 28.56367  # Ejemplo de latitud de la costa
coastline_lon = -80.57163  # Ejemplo de longitud de la costa

# Inicializar el mapa
site_map = folium.Map(location=[launch_site_lat, launch_site_lon], zoom_start=12)

# Calcular la distancia a la costa más cercana
distance_to_coastline = calculate_distance(launch_site_lat, launch_site_lon, coastline_lat, coastline_lon)

# Crear un marcador en la costa más cercana
coastline_marker = folium.Marker(
    location=[coastline_lat, coastline_lon],
    icon=DivIcon(
        icon_size=(20, 20),
        icon_anchor=(0, 0),
        html='<div style="font-size: 12; color:#d35400;"><b>Coastline: {:.2f} KM</b></div>'.format(distance_to_coastline),
    )
)
coastline_marker.add_to(site_map)

# Crear una PolyLine entre el sitio de lanzamiento y la costa
coordinates = [[launch_site_lat, launch_site_lon], [coastline_lat, coastline_lon]]
line_to_coastline = folium.PolyLine(locations=coordinates, weight=2, color='blue', opacity=0.7)

# Agregar la línea al mapa
site_map.add_child(line_to_coastline)

# Crear y agregar un marcador en el sitio de lanzamiento
folium.Marker(
    location=[launch_site_lat, launch_site_lon],
    popup='Launch Site',
    icon=folium.Icon(color='green')
).add_to(site_map)

# Mostrar el mapa
site_map
```

GitHub code

Predictive Analysis – Classification Accuracy

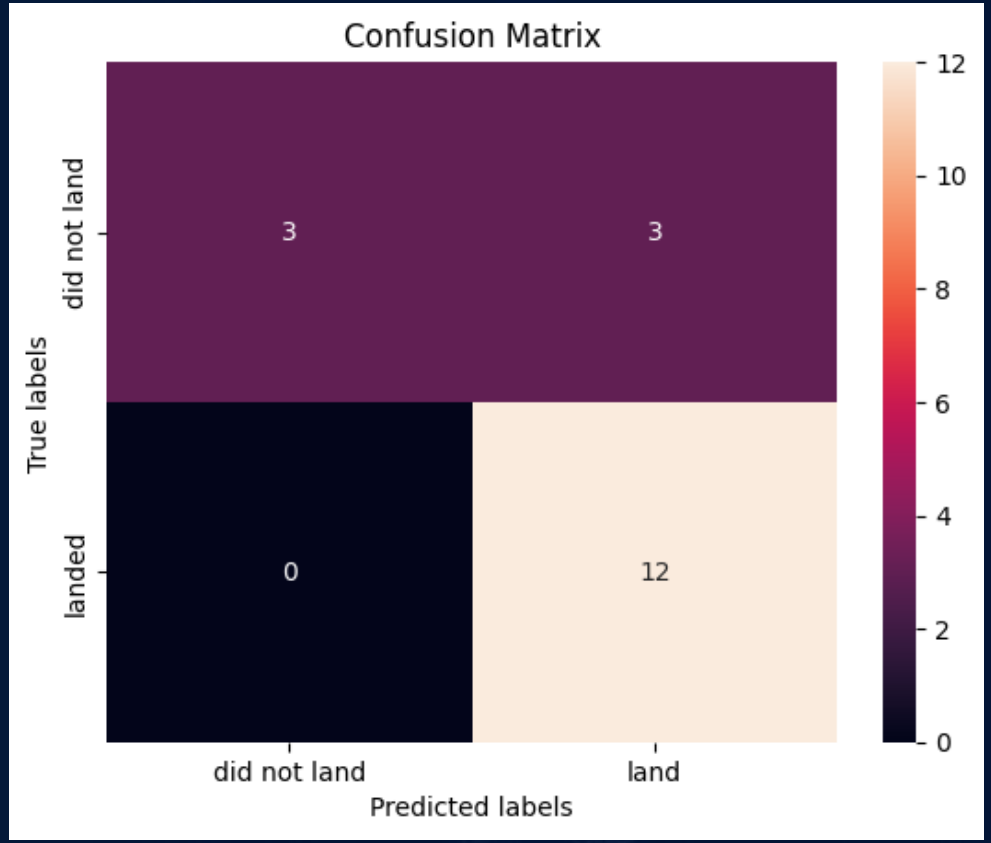
- Model Performance Comparison
  - Logistic Regression, SVM, and KNN:
    - All have the same performance with a Jaccard Score of 0.8.
  - Decision Tree:
    - Shows the worst performance compared to other models.
- Key Insight:
  - Logistic Regression, SVM, and KNN are equally effective, while Decision Tree underperforms in comparison.



GitHub code

Predictive Analysis – Confusion Matrix

- Confusion Matrix for Logistic Regression, SVM, and KNN
  - True Positive (TP): 12
  - False Positive (FP): 0
  - True Negative (TN): 3
  - False Negative (FN): 3
- Key Insight:
  - All three models (Logistic Regression, SVM, and KNN) show identical performance based on the confusion matrix, with no false positives and a good number of true positives.



GitHub code

Conclusions

- Key Success Factors and Trends
- Flight Experience:
  - As the number of flights increases, so does the success rate. Early flights (2010-2013) were mostly unsuccessful, but after 2013, success rates improved, especially post-2016, where the success rate was consistently above 50%.
- Orbit Performance:
  - ES-L1, GEO, HEO, SSO: 100% success rate, though some (GEO, HEO, ES-L1) had only 1 flight each.
  - SSO: More impressive with 5 successful flights.
  - PO, ISS, and LEO: More successful with heavy payloads.
  - VLEO: Typically associated with heavier payloads.
- Launch Sites:
  - KSC LC-39A had the most successful launches (41.7% of total), with a 76.9% success rate—the highest of all sites.
- Payload Impact:
  - Success with massive payloads (over 4000kg) is lower than for lighter payloads.
- Additional Insights:
- First Stage Return: Leads to significant cost savings for rocket launches.
- Wide Range of Factors: 83 attributes were considered in the model, affecting first stage return success.
- Launch Site Proximity: SpaceX sites are close to highways, railways, and coastlines to reduce transportation costs.
- General Trend: SpaceX's success rate has increased over the years, with KSC LC-39A leading in performance. Both orbit type and booster version significantly impact success.

Appendix

SpaceX REST API

```python
# Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
       if x:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

```python
# Takes the dataset and uses the launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['launchpad']:
       if x:
         response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
         Longitude.append(response['longitude'])
         Latitude.append(response['latitude'])
         LaunchSite.append(response['name'])
```

```python
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
       if load:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

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