IBM Developer
SKILLS NETWORK

IBM DATA SCIENCE CAPSTONE PROJECT

Space X Falcon 9 Landing Analysis

Francisco Javier Ordonez Araujo
September 2024

Outline

Executive Summary

Summary of Methodologies:

This project follows these steps:

- Data Collection
- Data Wrangling
- Exploratory Data Analysis
- Interactive Visual Analytics
- Predictive Analysis (Classification)

Summary of Results:

This project produced the following outputs and visualizations:

1. Exploratory Data Analysis (EDA) results
2. Geospatial analytics
3. Interactive dashboard
4. Predictive analysis of classification models
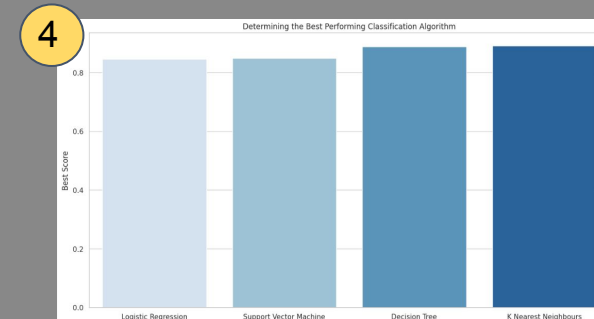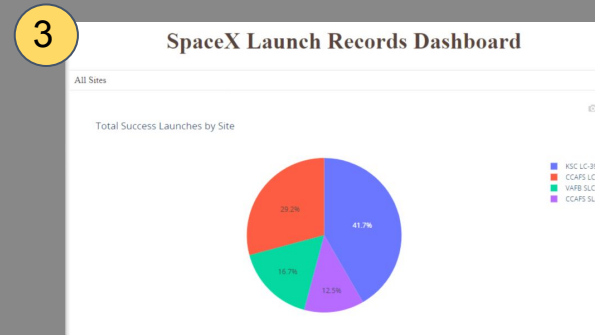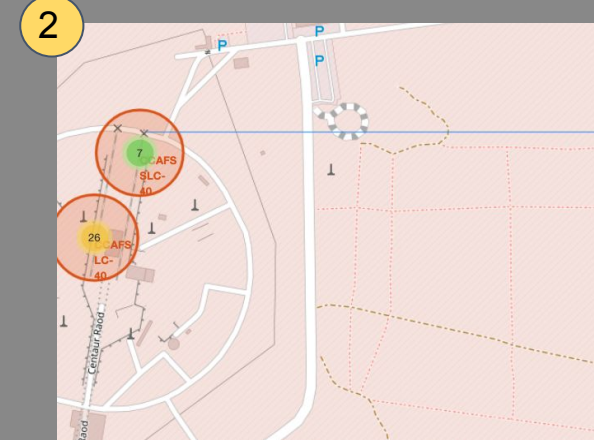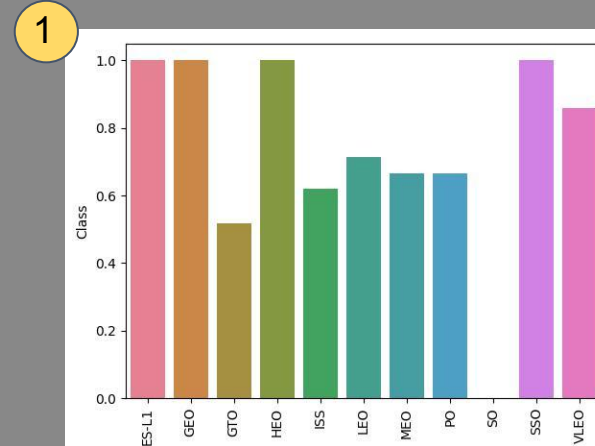
Introduction

- **SpaceX's Cost Advantage**: The Falcon 9 rocket costs approximately $62 million per launch, significantly lower than the industry average of $165 million. This is largely due to SpaceX's ability to land and reuse the first stage of the rocket, making it a cost-efficient solution in space exploration.

- **Predicting Launch Costs**: Accurately predicting whether the first stage will land allows us to refine the launch cost estimate, which is critical in determining if an alternative company should bid against SpaceX for a launch contract.

- **Project Goal**: This project focuses on **predicting the success of Falcon 9 first stage landings**, enabling better cost analysis and decision-making for future space missions.

Methodology

- **Data collection**
  - Taken from SpaceX REST API.
  - Web Scraping.

- **Perform data wrangling**
  - Cleaned data by handling missing values using .fillna().
  - Analyzed data distributions using .value_counts() for launch sites, orbits, and mission outcomes.
  - Created landing outcome labels: 0 for unsuccessful booster landings, 1 for successful landings.

- **Exploratory Data Analysis (EDA)**
  - Executed SQL queries to manipulate and explore the SpaceX dataset.
  - Visualized relationships between variables and identified patterns using Pandas and Matplotlib.

- **Interactive Visual Analysis**
  - Performed geospatial analysis with Folium.
  - Built an interactive dashboard using Plotly Dash for real-time data exploration.

- **Data Modeling and Evaluation**
  - Pre-processed data using Scikit-Learn, including standardization and train-test split.
  - Trained various classification models, optimized using GridSearchCV, and evaluated performance using confusion matrices and accuracy metrics.

Data Collection-SPACE X REST API

1 Data Acquisition

- Send a **GET** request to the SpaceX REST API and convert the JSON response into a Pandas DataFrame for structured analysis.

2 Data Cleaning

- Use custom logic and functions to clean and organize the data, storing key fields like BoosterVersion, PayloadMass, and LaunchSite in lists.

- Construct a dictionary from these lists for dataset creation.

3 DataFrame Creation

- Create a Pandas DataFrame from the constructed dictionary for easy manipulation.

4 Filtering and Refinement

- Filter the dataset for Falcon 9 launches, reset the *FlightNumber* column, and replace missing PayloadMass values with the column's mean.

1
```
ow let's start requesting rocket launch data from SpaceX API with the following URL:

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

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

2
```
#Global variables              # Call getBoosterVersion    launch_dict = {'FlightNumber': list(data['flight_number']),
BoosterVersion = []            getBoosterVersion(data)                   'Date': list(data['date']),
PayloadMass = []                                                         'BoosterVersion':BoosterVersion,
Orbit = []                     BoosterVersion[0:5]                       'PayloadMass':PayloadMass,
LaunchSite = []                                                          'Orbit':Orbit,
Outcome = []                   ['Falcon 1', 'Falcon 1', 'Fal(           'LaunchSite':LaunchSite,
Flights = []                                                             'Outcome':Outcome,
GridFins = []                                                            'Flights':Flights,
Reused = []                    # Call getLaunchSite                      'GridFins':GridFins,
Legs = []                      getLaunchSite(data)                       'Reused':Reused,
LandingPad = []                                                          'Legs':Legs,
Block = []                                                               'LandingPad':LandingPad,
ReusedCount = []               # Call getPayloadData                     'Block':Block,
Serial = []                    getPayloadData(data)                      'ReusedCount':ReusedCount,
Longitude = []                                                           'Serial':Serial,
Latitude = []                  # Call getCoreData                        'Longitude': Longitude,
                               getCoreData(data)                         'Latitude': Latitude}
```

3
```
# Create a data from launch_dict
df = pd.DataFrame.from_dict(launch_dict)
```

4
```
data_falcon9 = df[df['BoosterVersion']!='Falcon 1']
```

```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

Data Collection-WEB SCRAPING

- Request the HTML page from the static URL.

① ● Assign the response to an object.

② ● Create a BeautifulSoup object from the HTML response object.

- Find all tables within the HTML page.

③ ● Collect all column header names from the tables found within the HTML page.

④ ● Use the column names as keys in a dictionary.

- Use custom functions and logic to parse all launch tables to fill the dictionary values.

⑤ ● Convert the dictionary to a Pandas DataFrame ready for export.

①
```python
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
# use requests.get() method with the provided static_url
response = requests.get(static_url)
# assign the response to a object
data = response.text
```

②
```python
# Use BeautifulSoup() to create a BeautifulSoup
# object from a response text content
soup = BeautifulSoup(data, 'html5lib')
print(soup.title)
```

③
```python
column_names = []
# 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
for row in first_launch_table.find_all('th'):
    name = extract_column_from_header(row)
    if(name != None and len(name) > 0):
        column_names.append(name)
```

④
```python
launch_dict= dict.fromkeys(column_names)

# Remove an irrelevant 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']=[]
```

⑤
```python
df = pd.DataFrame(launch_dict)
df.head()
```

Data Manipulation & Wrangling – PANDAS

- **Context**
  - Dataset includes SpaceX launch facilities (LaunchSite column) and orbit types (Orbit column).
  - Each launch has a landing outcome (Outcome column) classified as success or failure.

- **Data Wrangling**
  - **(1)** Defining a set of unsuccessful (bad) outcomes called bad_outcome.

  - **(2)** Creating a list landing_class, where the element is 0 if the corresponding row in the Outcome column is in bad_outcome, otherwise it's 1.

  - **(3)** Creating a Class column that contains the values from the landing_class list.

  - Exporting the DataFrame as a .csv file for further analysis.
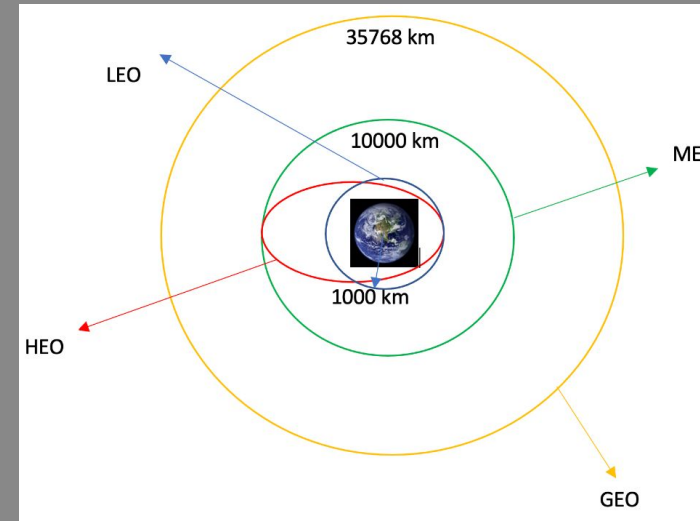

```python
# Apply value_counts on Orbit column
df['Orbit'].value_counts()

GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1

# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
landing_outcomes

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

(1) We create a set of outcomes where the second stage did not land successfully:
```python
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```
```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

(2)
```python
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise

landing_class = []

for outcome in df['Outcome']:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

(3)
```python
df['Class'] = landing_class

# Export the DataFrame as a CSV file
df.to_csv("dataset_part_2.csv", index=False)
```

EDA with Data Visualization

- **Scatter Charts**: Utilized to examine relationships between **flight number**, **launch site**, and **payload**, as well as correlations between orbit type and flight number, offering insights into the distribution of numeric variables.

- **Bar Charts**: Bar charts are used to visually compare success rates across different orbit types, making it easier to understand the performance of various mission categories. By representing categorical data alongside numerical success rates, the bar charts allow for a quick comparison, highlighting which orbits show higher or lower success rates over time and offering insights into the relative reliability of each orbit type.

- **Line Charts**: Employed to display trends in **launch success rates over time**, helping to visualize changes across multiple years (2010–2020).

EDA with SQL

- Names of Unique launch sites in the space mission.

- Five records where launch sites begin with string 'CCA' .

- Total payload mass by **NASA (CRS).**

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

- First successful landing outcome in ground pad • Success in drone ship, payload **mass >4000** but **<6000**

- Total number of successful and failure mission outcomes.

- Booster_versions with maximum payload mass.

- Month names of failure landing outcomes in drone ship in year 2015.

- Landing outcome counts between "2010-06-04" and "2017-03-20" in descending order.

Build an Interactive Map with Folium

In the interactive map created with Folium, I integrated various elements to enhance the visualization. Markers were added to pinpoint specific launch locations, such as the NASA JSC space station launch site. Circles were used to highlight key areas, each accompanied by text labels at specified coordinates for clarity. Additionally, lines were drawn to represent the proximity between launch sites, providing a clear depiction of spatial relationships.

Build a Dashboard with Plotly Dash

In the Plotly Dash user application I have added a dropdown list and a range slider to allow a user to interact with a pie chart and the scatter point chart.

Predictive Analysis (Classification)

- The cleaned dataset was imported, with features assigned to X and the target variable to Y. Using sklearn's **StandardScaler**, the data was normalized, followed by a train-test split where 80% was used for training and 20% for testing.

- Multiple models were instantiated, and hyperparameter tuning was performed using **GridSearchCV** to identify the best-performing parameters for each model.

- After training, the **DecisionTreeClassifier** delivered the best results, achieving an accuracy of 88.88% and an F1-score of 88.21% on the test set. The performance was optimized through **GridSearchCV** for hyperparameter tuning, ensuring robust evaluation on unseen data.

RESULT

Exploratory Data Analysis

Interactive Analytics

Predictive Analysis

EDA with Data Visualization

# Launch Site VS Flight Number

- Success rates improve as the number of flights increases at each launch site.

- Early flights (Flight Number < 30) from CCAFS SLC 40 and VAFB SLC 4E had a higher rate of failure, with fewer successful landings.

- KSC LC 39A shows a higher success rate overall, especially since no early flights were launched from this site.

- Notably, after Flight Number 30, the majority of flights across all launch sites show a clear increase in successful landings (Class = 1).
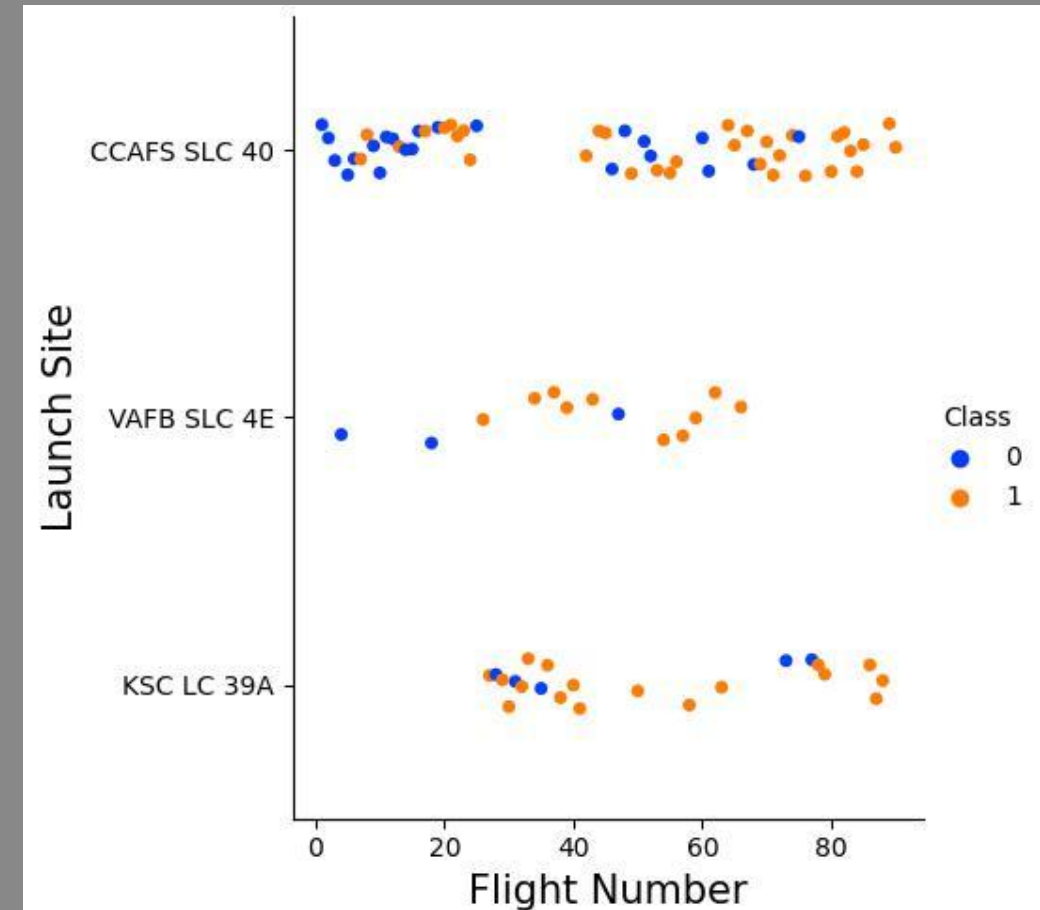
Success Rate VS Orbit Type

The bar chart provides a comprehensive view of the success rates across various orbital types. It is evident that certain orbits exhibit near-perfect performance, while others face significant challenges. This information can guide future mission planning by highlighting which orbits tend to be more reliable and which may require further technological advancements to improve success rates.

- Orbits with a 100% success rate include

  - ES-L1 (Earth-Sun First Lagrangian Point).

  - GEO (Geostationary Orbit).

  - HEO (High Earth Orbit).

  - SSO (Sun-synchronous Orbit).

- The orbit with the lowest success rate (0%) is:

  - SO (Heliocentric Orbit)
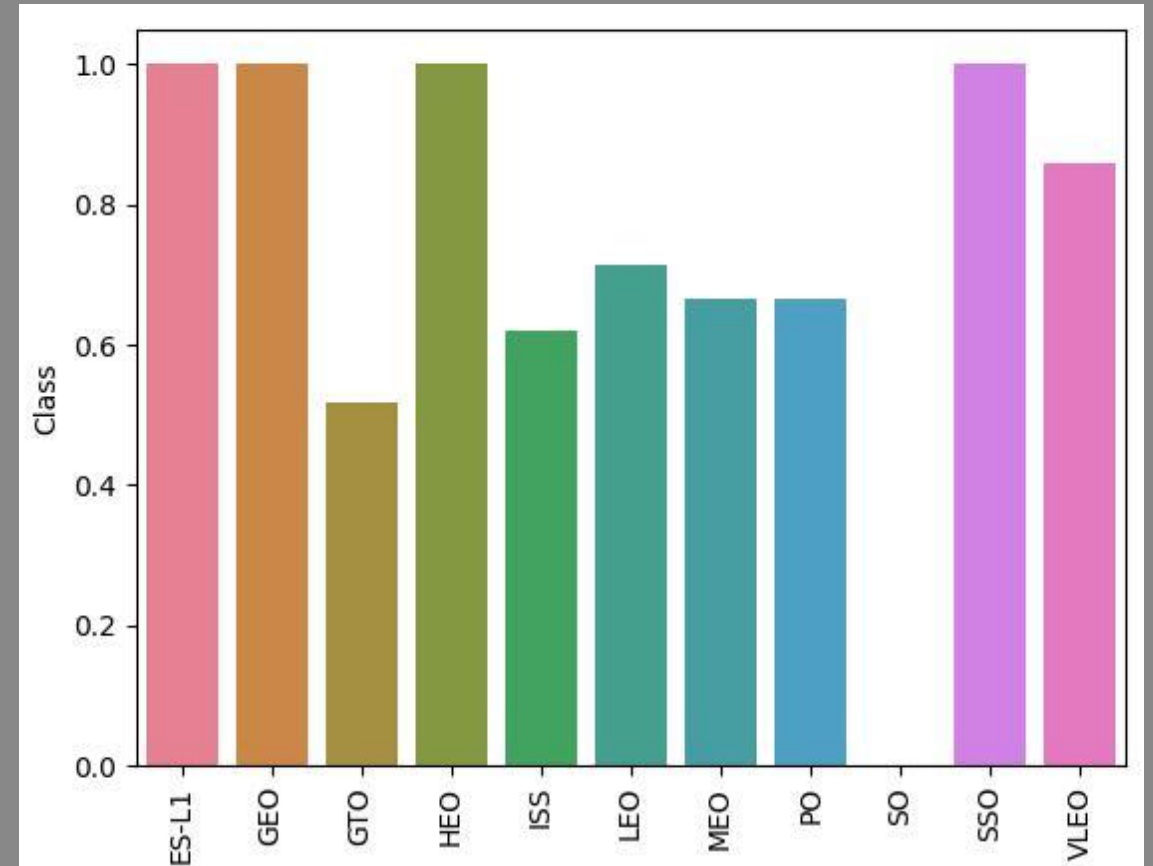
Orbit Type VS Flight Number

This scatter plot illustrates the relationship between Orbit Type and Flight Number, highlighting several key insights:

- GEO, HEO, and ES-L1 orbits have a 100% success rate, though it's worth noting that each of these categories only had a single flight.

- SSO orbits demonstrate an impressive 100% success rate over 5 successful flights.

- For GTO orbits, there is minimal correlation between the number of flights and the success rate.

- LEO orbits exhibit a trend where higher flight numbers are associated with increased success rates. Early flights, with lower flight numbers, experienced a greater number of unsuccessful landings.
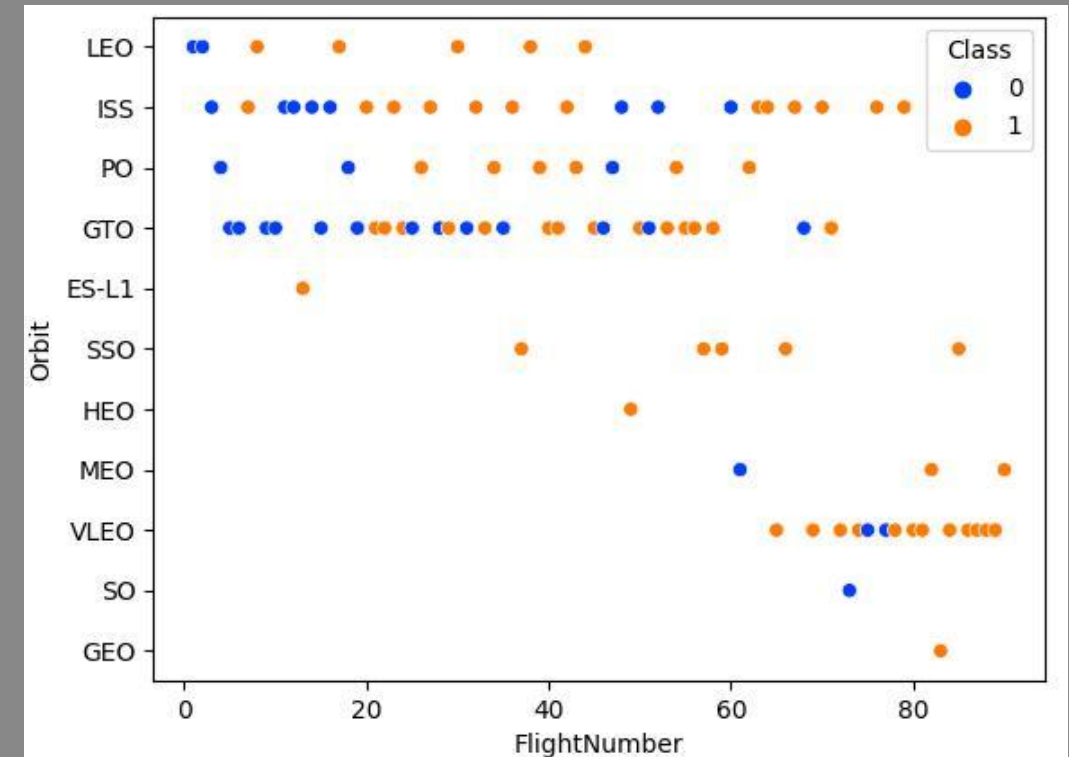
Orbit Type VS Payload Mass

This scatter plot shows the relationship between Orbit Type and Payload Mass, highlighting:

- PO, ISS, and LEO achieve higher success rates with heavier payloads, though data for PO is limited.

- For GTO, there's no clear pattern between payload mass and success rate.

- VLEO missions often involve heavier payloads, which makes sense given their proximity to Earth.
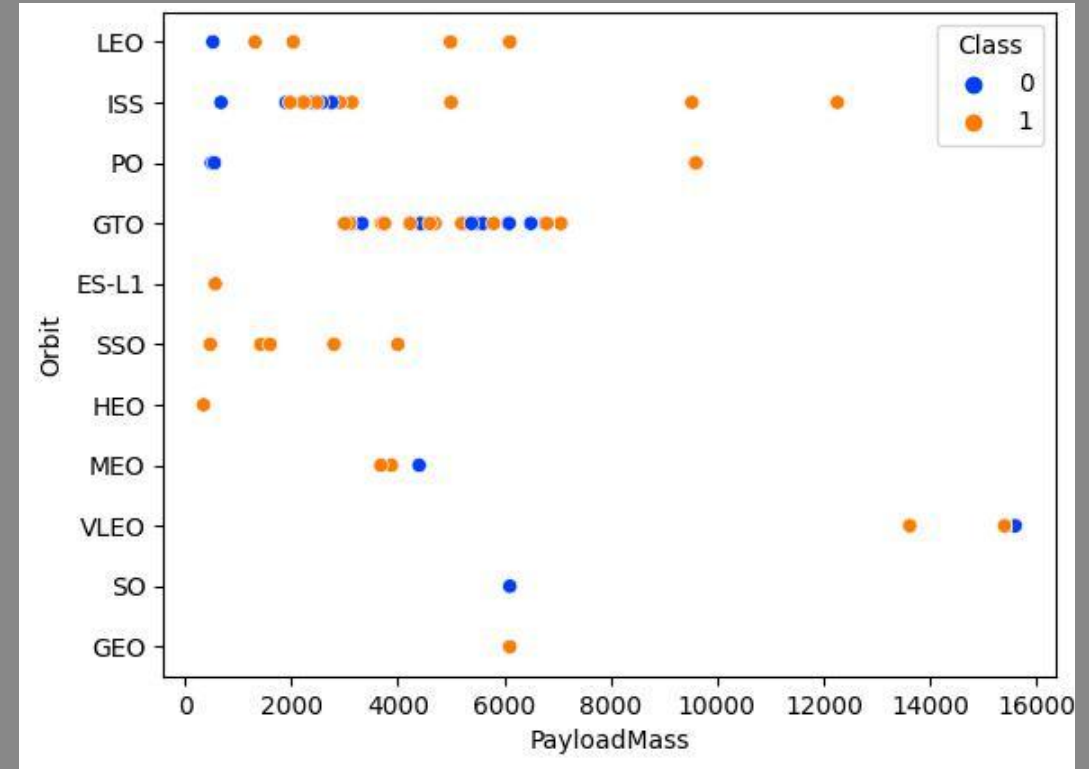
Launch Success Yearly Trend

This line chart illustrates the yearly average success rate of landings over the period from 2010 to 2020. The data provides a clear view of the progression in launch reliability and highlights key trends:

- 2010-2013: Success rate remained at 0%, indicating early challenges in technology.

- However, from **2013 onwards**, a notable improvement is evident. The success rate began to rise steadily, reaching over **60% by 2015**. This growth indicates significant advancements in booster technology and mission execution.

- 2016-2017: Peak success rate, consistently above 50%.

- Notably, since **2016**, the success rate has remained consistently **above 50%**, demonstrating the maturity and reliability of the landing technology. By **2020**, the success rate was nearing **90%**, showing remarkable progress from earlier years.
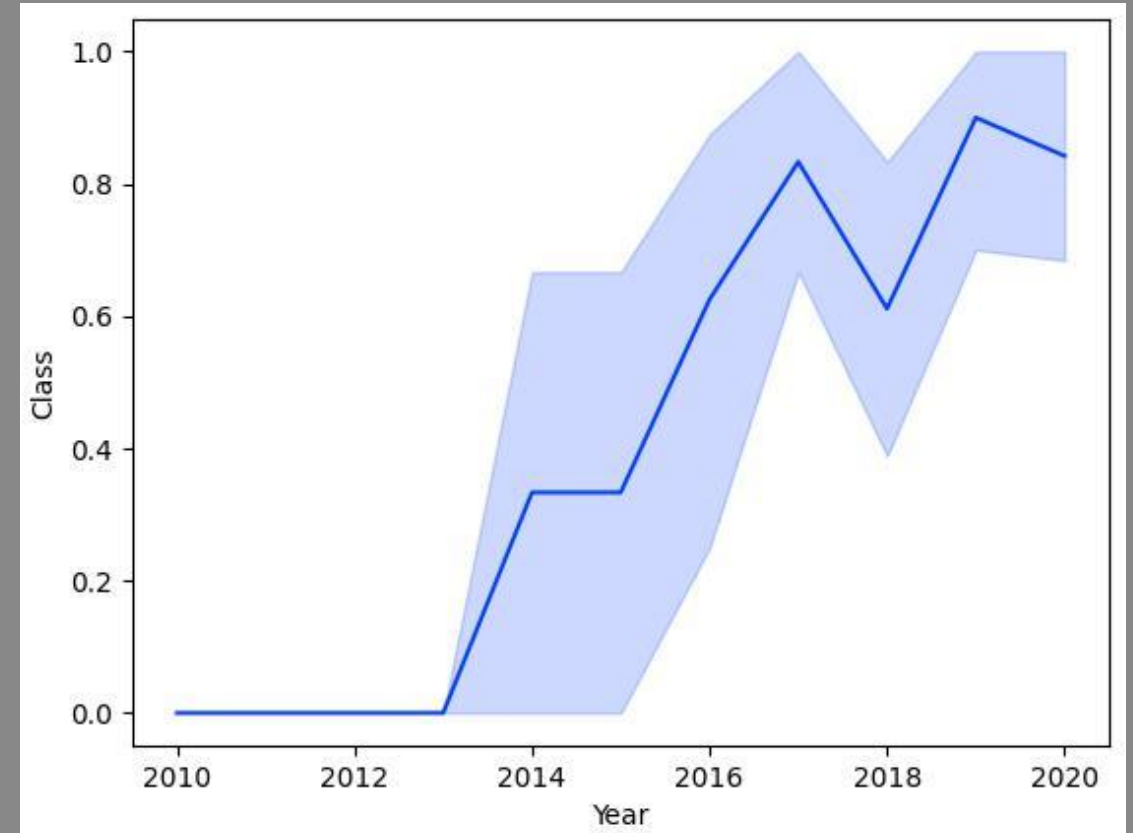
EDA with SQL

All Launch Site Names

Find the names of the unique launch sites.


```
Display the names of the unique launch sites in the space mission
   %sql SELECT DISTINCT(LAUNCH_SITE) FROM SPACEXTBL;
7]    ✓  0.0s
```

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

The word DISTINCT returns only unique values from the LAUNCH_SITE column of the SPACEXTBL

Launch Site Names Begin with 'CCA'

Find 5 records where launch sites begin with 'CCA'

```
%sql SELECT LAUNCH_SITE FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

LIMIT 5 fetches only 5 records, and the LIKE keyword is used with the wild card 'CCA%' to retrieve

string values beginning with 'CCA'.

Total Payload Mass

Calculate the total payload carried by boosters from NASA.

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

TOTAL_PAYLOAD_MASS
45596

The SUM keyword is used to calculate the total of the LAUNCH column, and the SUM keyword (and the associated condition) filters the results to only boosters from NASA (CRS)

Average Payload Mass by F9 V1.1

Calculate the average payload mass carried by booster version F9 v1.1

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

AVERAGE_PAYLOAD_MASS
2928.4

The AVG keyword is used to calculate the average of the PAYLOAD_MASS__KG_ column, and the WHERE keyword (and the associated condition) filters the results to only the F9 v1.1 booster version.

First Successful Ground Landing Date

Find the dates of the first successful landing outcome on ground pad.

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

FIRST_SUCCESSFUL_GROUND_LANDING

2015-12-22

The MIN keyword is used to calculate the minimum of the DATE column, i.e. the first date, and the WHERE keyword (and the associated condition) filters the results to only the successful ground pad landings

Successful Drone Ship Landing with Payload between 4000 and 6000

List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000.

```sql
%%sql
SELECT BOOSTER_VERSION FROM SPACEXTBL
WHERE LANDING_OUTCOME = "Success (drone ship)" and   PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000;
```

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

The WHERE keyword is used to filter the results to include only those that satisfy both conditions in

the brackets (as the AND keyword is also used). The BETWEEN keyword allows for 4000 < x < 6000

values to be selected.

Total Number of Successful and Failure Mission Outcomes

Calculate the total number of successful and failure mission outcome

```sql
%%sql
    SELECT Mission_Outcome, COUNT(Mission_Outcome) AS TOTAL_NUMBER
    FROM SPACEXTBL GROUP BY Mission_Outcome;
```
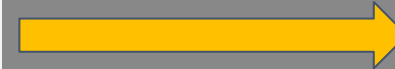
Mission_Outcome	TOTAL_NUMBER
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

The COUNT keyword is used to calculate the total number of mission outcomes, and the GROUPBY

keyword is also used to group these results by the type of mission outcome

Boosters Carried Maximum Payload

List the names of the booster which have carried the maximum payload mass.

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

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

The WHERE keyword is used to filter the results for only failed landing outcomes, AND only for the year of 2015.

2015 Launch Records

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015.

```sql
%%sql
    SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL
    WHERE (Landing_Outcome = 'Failure (drone ship)') AND strftime('%Y', Date) = '2015';
```

Booster_Version	Launch_Site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

A subquery is used here. The SELECT statement within the brackets finds the

maximum payload, and this value is used in the WHERE condition. The DISTINCT

keyword is then used to retrieve only distinct /unique booster versions.

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

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

```sql
%%sql SELECT Landing_Outcome, COUNT(Landing_Outcome) AS TOTAL_NUMBER
FROM SPACEXTBL
WHERE Date BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY Landing_Outcome
ORDER BY TOTAL_NUMBER DESC;
```
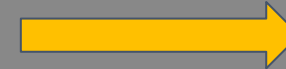
Landing_Outcome	TOTAL_NUMBER
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

The WHERE keyword is used with the BETWEEN keyword to filter the results to dates only within those specified. The results are then grouped and ordered, using the keywords GROUP BY and ORDER BY, respectively, where DESC is used to specify the descending order

Proximities

Proximities such as railway, highway, coastline, with distance calculated and displayed.


```python
def calculate_distance(lat1, lon1, lat2, lon2):
    # approximate radius of earth in km
    R = 6373.0

    lat1 = radians(lat1)
    lon1 = radians(lon1)
    lat2 = radians(lat2)
    lon2 = radians(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
```

```python
# find coordinate of the closet coastline
launch_site_lat = 28.56319
launch_site_long = -80.57683
launch_site_coordinates = [launch_site_lat, launch_site_long]

coastline_lat = launch_site_lat #same latitude, so the line is directly east
coastline_long = -80.56794
coastline_coordinates = [coastline_lat, coastline_long]

distance_coastline = calculate_distance(launch_site_lat, launch_site_long, coastline_lat, coastline_long)


print(f"The coastline is {distance_coastline:.2f} km due East from the launch site")
```

```
The coastline is 0.87 km due East from the launch site
```

Launch Success Count for all Sites

Total Success Launches by Site
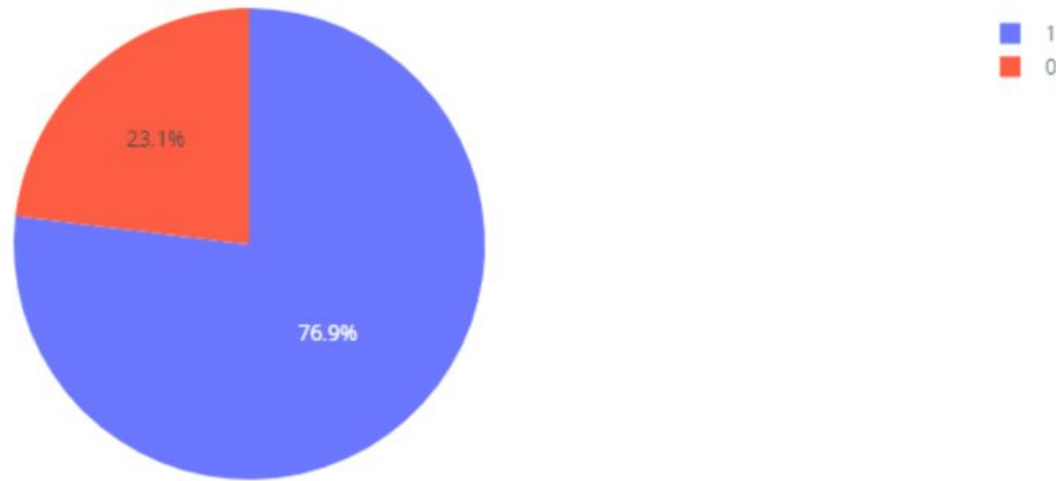
- KSC LC-39A
- CCAFS LC-40
- VAFB SLC-4E
- CCAFS SLC-40

41.7%
29.2%
16.7%
12.5%
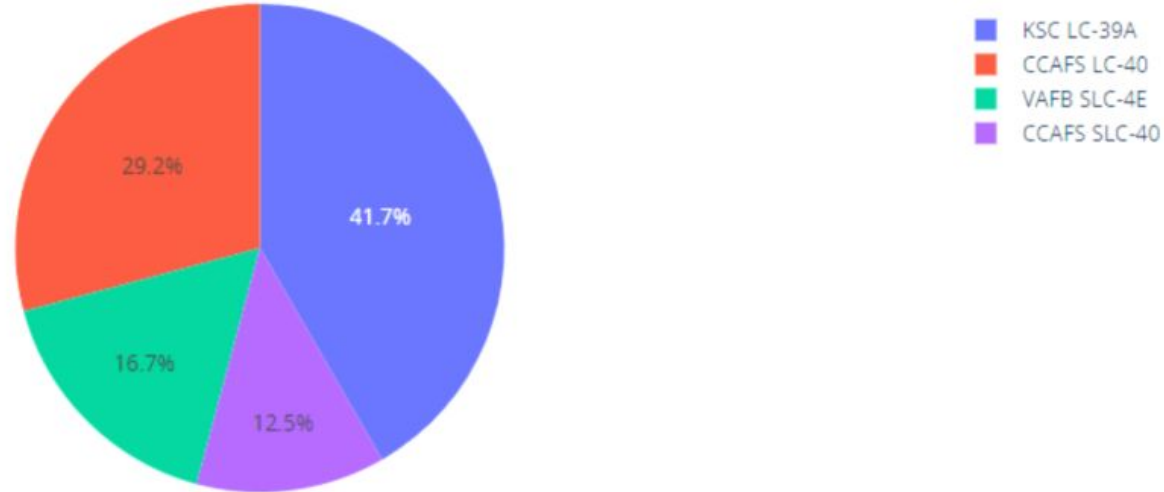
The launch site KSC LC-39A had the most successful launches, with 41.7% of the total successful launches.
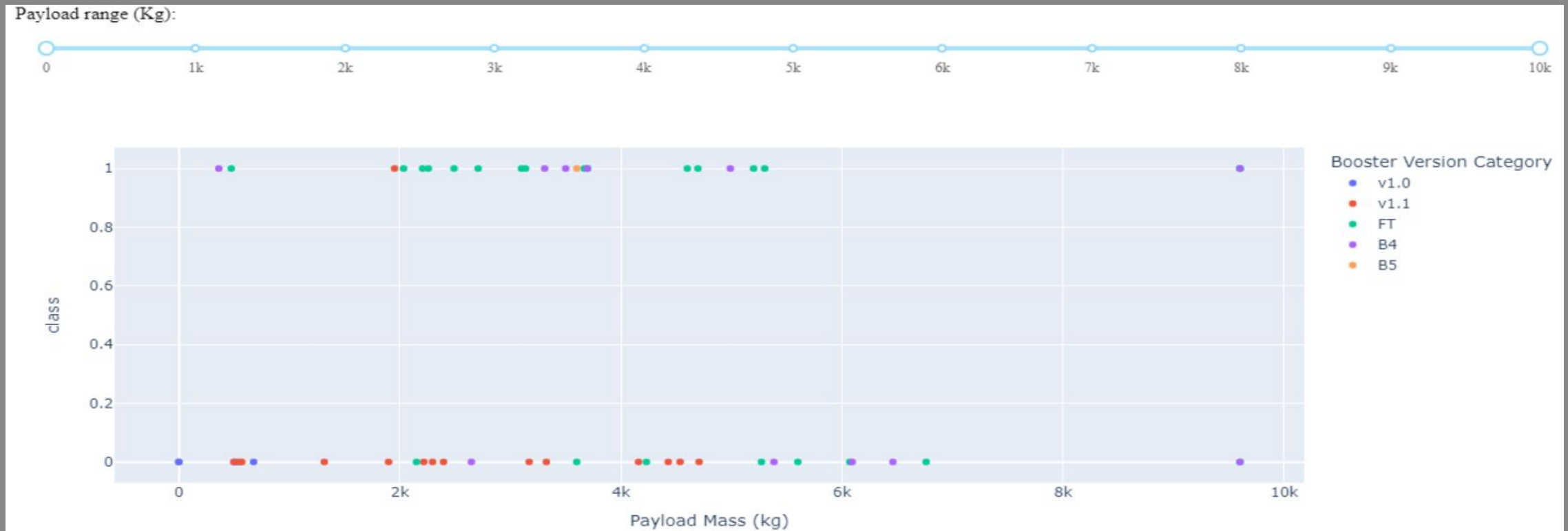
Pie chart of KSC LC 39A

Total Success Launches for site KSC LC-39A

23.1%

76.9%

The launch site KSC LC-39 A also had the highest rate of successful launches, with a 76.9% success rate.

Payload vs Launch Outcome scatter

The scatter plot reveals that as payload mass increases, the success rate improves for the FT booster version. In contrast, the v1.1 booster version shows the opposite trend, with a higher rate of failures for heavier payloads.

Predictive Analysis- Classification

Payload vs Launch Outcome scatter

Plotting the Accuracy Score and Best Score for each classification algorithm yields the following insights:

- Logistic Regression, Support Vector Machine, and K Nearest Neighbours all achieve an accuracy score of 83.33%, making them the top-performing models in this set.
- The Decision Tree algorithm lags slightly behind with an accuracy score of 77.78%.
- In terms of the Best Score, the Decision Tree stands out with the highest value at 88.93%, indicating its potential for improvement with optimal parameters.
- While Logistic Regression has a slightly lower best score of 84.64%, it still performs consistently well overall.
- Support Vector Machine and K Nearest Neighbours follow closely with best scores of 84.82% and 87.50%, respectively, showing robust performance across different tests.

|	Algorithm	Accuracy Score	Best Score
0	Logistic Regression	0.833333	0.846429
1	Support Vector Machine	0.833333	0.848214
2	Decision Tree	0.777778	0.889286
3	K Nearest Neighbours	0.833333	0.875000
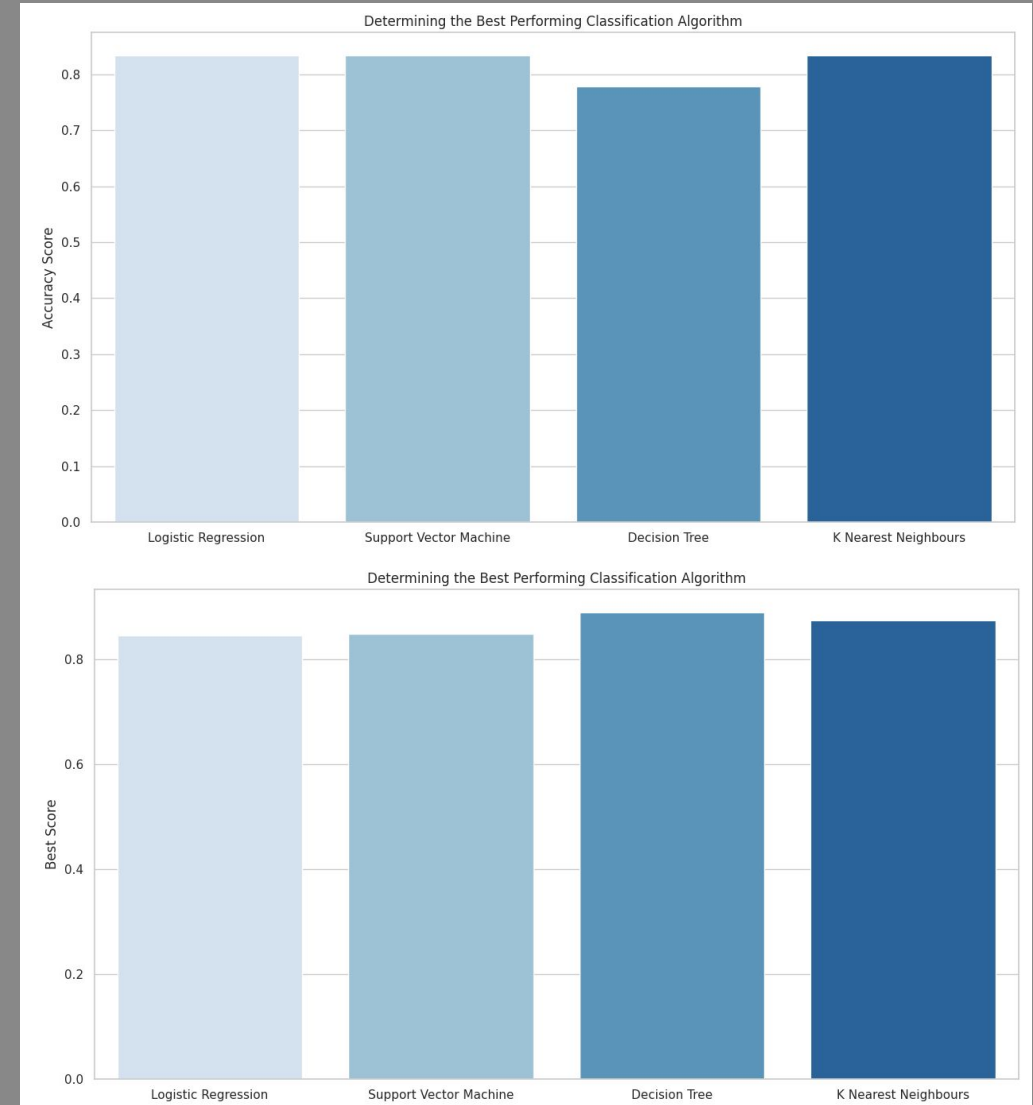
Payload vs Launch Outcome scatter

The **best performing classification model**, **Logistic Regression**, achieves an accuracy of **83.33%**.
This performance is well-explained by the **confusion matrix**, where we observe:

- Out of **18 predictions**, **15 are correct**:
  - **12 true positives** (correctly predicting landings)
  - **3 true negatives** (correctly predicting non-landings)
- There are **3 misclassifications**, all of which are **false positives** (incorrectly predicting landings when they did not occur), shown in the top-right corner.

The confusion matrix highlights the model's strong ability to predict successful landings, but with a minor tendency to overestimate landings in cases where they did not happen.

Confusion Matrix
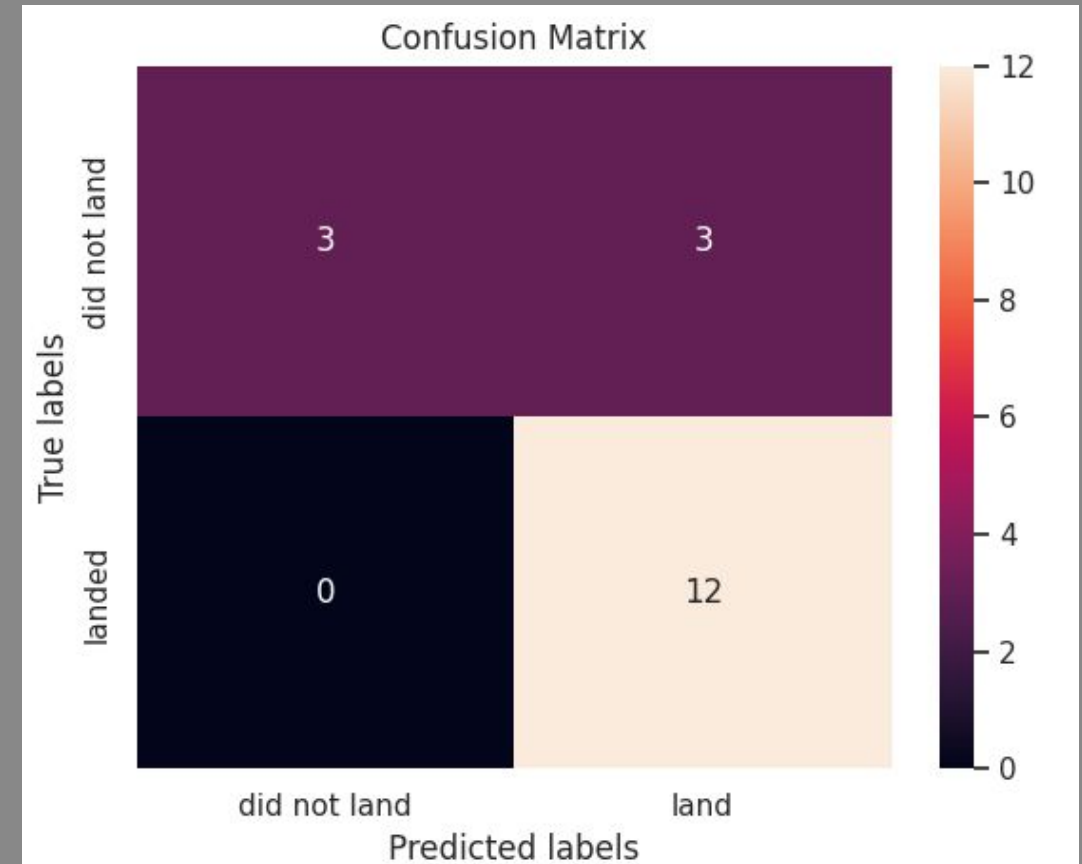
Conclusion

• As the flight number increases there's more success rate for Launch Site CCAFS SLC 40, the same relationship was observed for payload mass.

• There's 100% success rate of landing for Orbit type ES-L1, SSO, HEO, and GEO

• There's been steady increase in the success rate of Falcon 9 landing since 2010-2020

• With a very high accuracy oF 83.33%, we know that our model can predict the outcome of the landing.

Appendix

All relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, and data sets included in this presentation can be found on my [GitHub](#).

Thank you!