IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Markus Sloth
06/04/2023

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection
  - Data wrangling
  - EDA with Data visualisation and SQL
  - Interactive map, Plotly Dash,
  - Classification of predictive analysis
- Summary of all results
  - EDA results
  - Interactive analysis
  - Predictive analysis

Introduction

- Project background and context

  SpaceX advertises Falcon 9 rocket launches on its website with of cost of 62 million dollars

- Problems you want to find answers

  The project is about predicting if the first stage of SpaceX Falcon 9 rocket will land successfully

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:

  Web scrapping from Website

  SpaceX Rest API

- Perform data wrangling

  One Hot Encoding data fields for Machine learning and data cleaning of null values and irrelevant columns

- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models

  LR,KNN,SVM,DT models have been built and evaluated for the best classifier

Data Collection

Web scraping to collect Falcon 9 historical launch records from a **Wikipedia** page titled – "List of Falcon 9 and Falcon Heavy launches."

- Under HTML page, requesting the URL
- Find the respective table from the URL
- Collect all the data of the columns from the table found in the above URL
- Use the column name as keys
- Panda Data frame are ready to be exported

```
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
```

```
soup = BeautifulSoup(data, 'html5lib')
```

```
html_tables = soup.find_all('table')
```

```
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)
```

```
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']=[]
```

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

Data Collection – SpaceX API

- The following datasets were collected:

  SpaceX launch data that is gathered from the SpaceX API and this gives us data about launches and its specifications along with outcome.

```python
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())
```

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

```python
# Call getBoosterVersion
getBoosterVersion(data)

# Call getLaunchSite
getLaunchSite(data)

# Call getPayloadData
getPayloadData(data)

# Call getCoreData
getCoreData(data)
```

```python
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}
```

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

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

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

# Calculate the mean value of PayloadMass column and Replace the np.nan values with its mean value
data_falcon9 = data_falcon9.fillna(value={'PayloadMass': data_falcon9['PayloadMass'].mean()})
```

Data Wrangling

Context:

- The SpaceX dataset contains several Space X launch facilities, and each location is in the LaunchSite column.
- Each launch aims to a dedicated orbit, and some of the common orbit types are shown in the figure below. The orbit type is in the Orbit column.

Initial Data Exploration:

- Using the .value_counts() method to determine the following:

1. Number of launches on each site
2. Number and occurrence of each orbit
3. Number and occurrence of landing outcome per orbit type

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

CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
```

```
# 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
GEO       1
SO        1
HEO       1
Name: Orbit, dtype: int64
```

```
# 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
None ASDS       2
False Ocean     2
False RTLS      1
```

Context:

- The landing outcome is shown in the Outcome column:

True Ocean – the mission outcome was successfully landed to a specific region of the ocean

False Ocean – the mission outcome was unsuccessfully landed to a specific region of the ocean.

True RTLS – the mission outcome was successfully landed to a ground pad

False RTLS – the mission outcome was unsuccessfully landed to a ground pad.

True ASDS – the mission outcome was successfully landed to a drone ship

False ASDS – the mission outcome was unsuccessfully landed to a drone ship.

None ASDS and None None – these represent a failure to land.

Data Wrangling:

- To determine whether a booster will successfully land, it is best to have a binary column, i.e., where the value is 1 or 0, representing the success of the landing.
- This is done by:

1. Defining a set of unsuccessful (bad) outcomes, bad_outcome
2. Creating a list, landing_class, where the element is 0 if the corresponding row in Outcome is in the set bad_outcome, otherwise, it's 1.
3. Create a Class column that contains the values from the list landing_class
4. Export the DataFrame as a .csv file.

```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

```
.anding_class = 0 if bad_out
landing_class = 1 otherwise

.anding_class = []

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

```
df['Class']=landing_class
```

```
df.to_csv("dataset_part\_2.csv", index=False)
```

EDA with Data Visualization

Scatter Charts

- Scatter charts are useful to observe relationships, or correlations, between two numeric variables.
- Scatter charts were produced to visualize the relationships between:
  - Flight Number and Launch Site
  - Payload and Launch Site
  - Orbit Type and Flight Number
  - Payload and Orbit Type

Bar Charts

- Bar charts are used to compare a numerical value to a categorical variable. Horizontal or vertical bar charts can be used, depending on the size of the data
- A bar chart was produced to visualize the relationship between:
  - Success Rate and Orbit Type

Line Charts

- Line charts contain numerical values on both axes, and are generally used to show the change of a variable over time.
- Line charts were produced to visualize the relationships between:
  - Success Rate and Year (i.e. the launch success yearly trend)

EDA with SQL

- **The SQL queries performed on the data set were used to:**

1. **Display the names of the unique launch sites in the space mission**
2. **Display 5 records where launch sites begin with the string 'CCA'**
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 on a ground pad was achieved**
6. **List the names of the boosters which had success on a drone ship and a payload mass between 4000 and 6000 kg**
7. **List the total number of successful and failed mission outcomes**
8. **List the names of the booster versions which have carried the maximum payload mass**
9. **List the failed landing outcomes on drone ships, their booster versions, and launch site names for 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**

Build an Interactive Map with Folium

- The following steps were taken to visualize the launch data on an interactive map:

1. Mark all launch sites on a map
   - Initialise the map using a Folium Map object
   - Add a folium.Circle and folium.Marker for each launch site on the launch map

2. Mark the success/failed launches for each site on a map
   - As many launches have the same coordinates, it makes sense to cluster them together.
   - Before clustering them, assign a marker colour of successful (class = 1) as green, and failed (class = 0) as red.
   - To put the launches into clusters, for each launch, add a folium.Marker to the MarkerCluster() object.
   - Create an icon as a text label, assigning the icon_color as the marker_colour determined previously.

3. Calculate the distances between a launch site to its proximities
   - To explore the proximities of launch sites, calculations of distances between points can be made using the Lat and Long values.
   - After marking a point using the Lat and Long values, create a folium.Marker object to show the distance.
   - To display the distance line between two points, draw a folium.PolyLine and add this to the map.

Build a Dashboard with Plotly Dash

- SThe following plots were added to a Plotly Dash dashboard to have an interactive visualisation of the data:

1. Pie chart (px.pie()) showing the total successful launches per site
   - This makes it clear to see which sites are most successful
   - The chart could also be filtered (using a dcc.Dropdown() object) to see the success/failure ratio for an individual site

2. Scatter graph (px.scatter()) to show the correlation between outcome (success or not) and payload mass (kg)
   - This could be filtered (using a RangeSlider() object) by ranges of payload masses
   - It could also be filtered by booster version

Predictive Analysis (Classification)

- **<u>Model Development:</u>**
- To prepare the dataset for model development:
  - Load dataset
  - Perform necessary data transformations (standardise and pre-process)
  - Split data into training and test data sets, using `train_test_split()`
  - Decide which type of machine learning algorithms are most appropriate
- For each chosen algorithm:
  - Create a `GridSearchCV` object and a dictionary of parameters
  - Fit the object to the parameters
  - Use the training data set to train the model
- **<u>Model Evaluation:</u>**
- For each chosen algorithm:
  - Using the output GridSearchCV object:
    - Check the tuned hyperparameters (`best_params_`)
    - Check the accuracy (`score` and `best_score_`)
  - Plot and examine the Confusion Matrix

F**<u>inding the best model:</u>**

- Review the accuracy scores for all chosen algorithms
- The model with the highest accuracy score is determined as the best performing model

Results

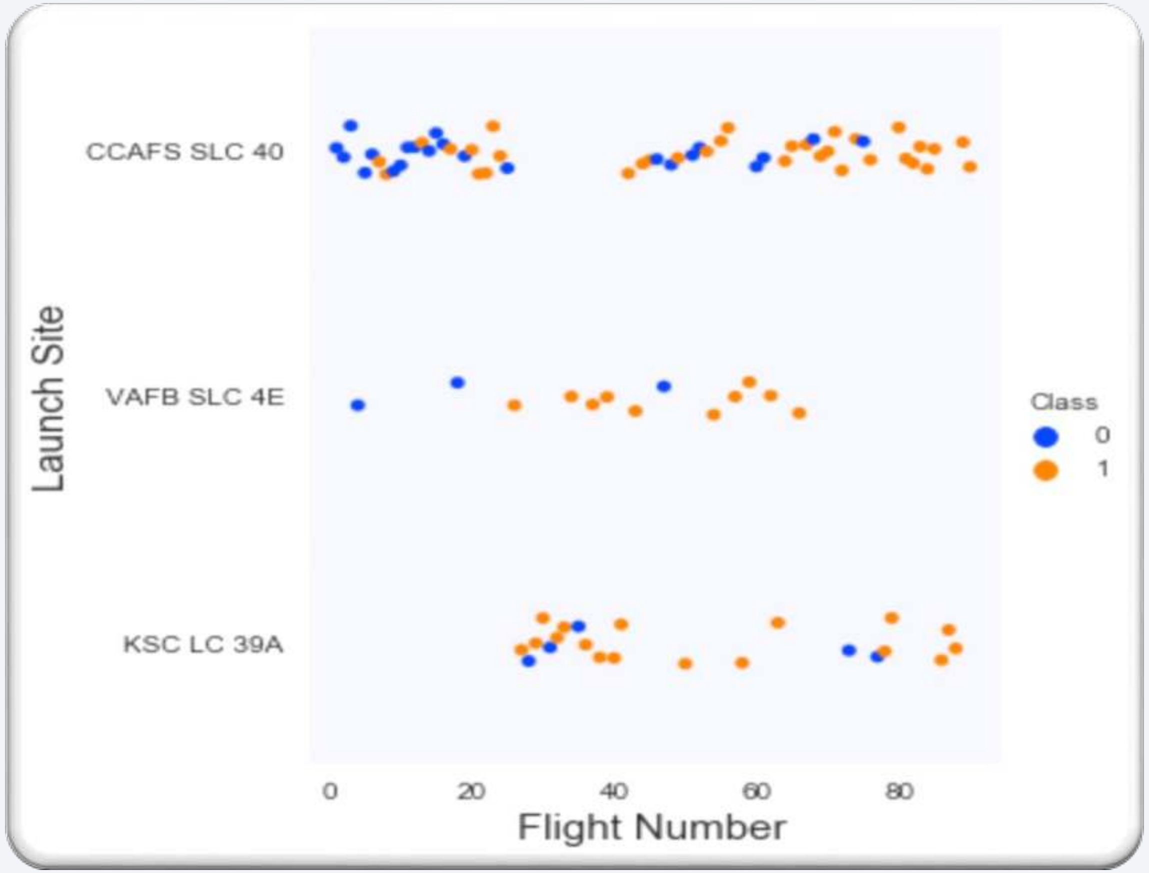
- Exploratory Data Analysis
- Interactive Analysis
- Predictive Analysis

Section 2

Insights drawn from EDA

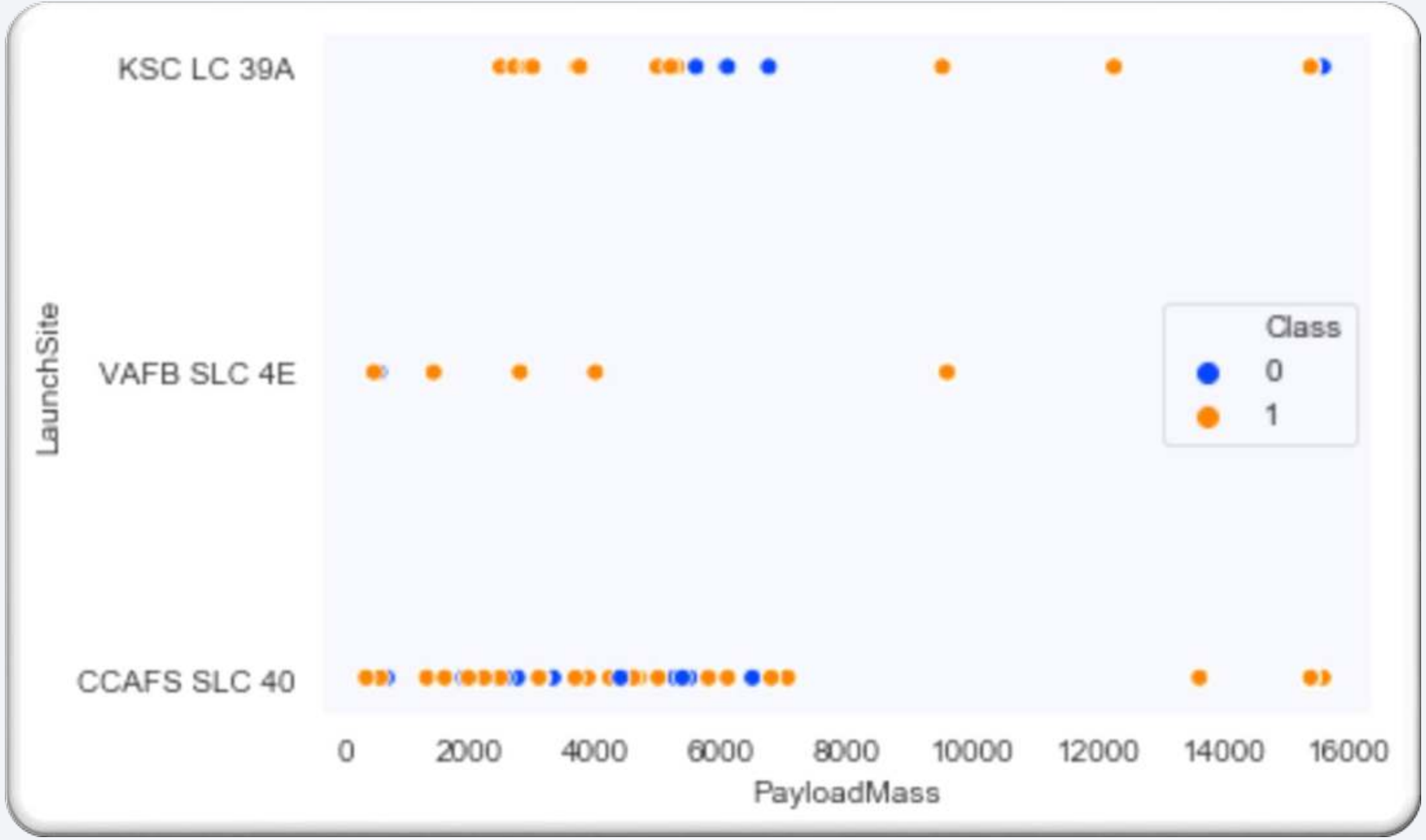
Flight Number vs. Launch Site

- The scatter plot of Launch Site vs. Flight Number shows that:
- As the number of flights increases, the rate of success at a launch site increases.
- Most of the early flights (flight numbers < 30) were launched from CCAFS SLC 40, and were generally unsuccessful.
- The flights from VAFB SLC 4E also show this trend, that earlier flights were less successful.
- No early flights were launched from KSC LC 39A, so the launches from this site are more successful.
- Above a flight number of around 30, there are significantly more successful landings (Class = 1).



Payload vs. Launch Site

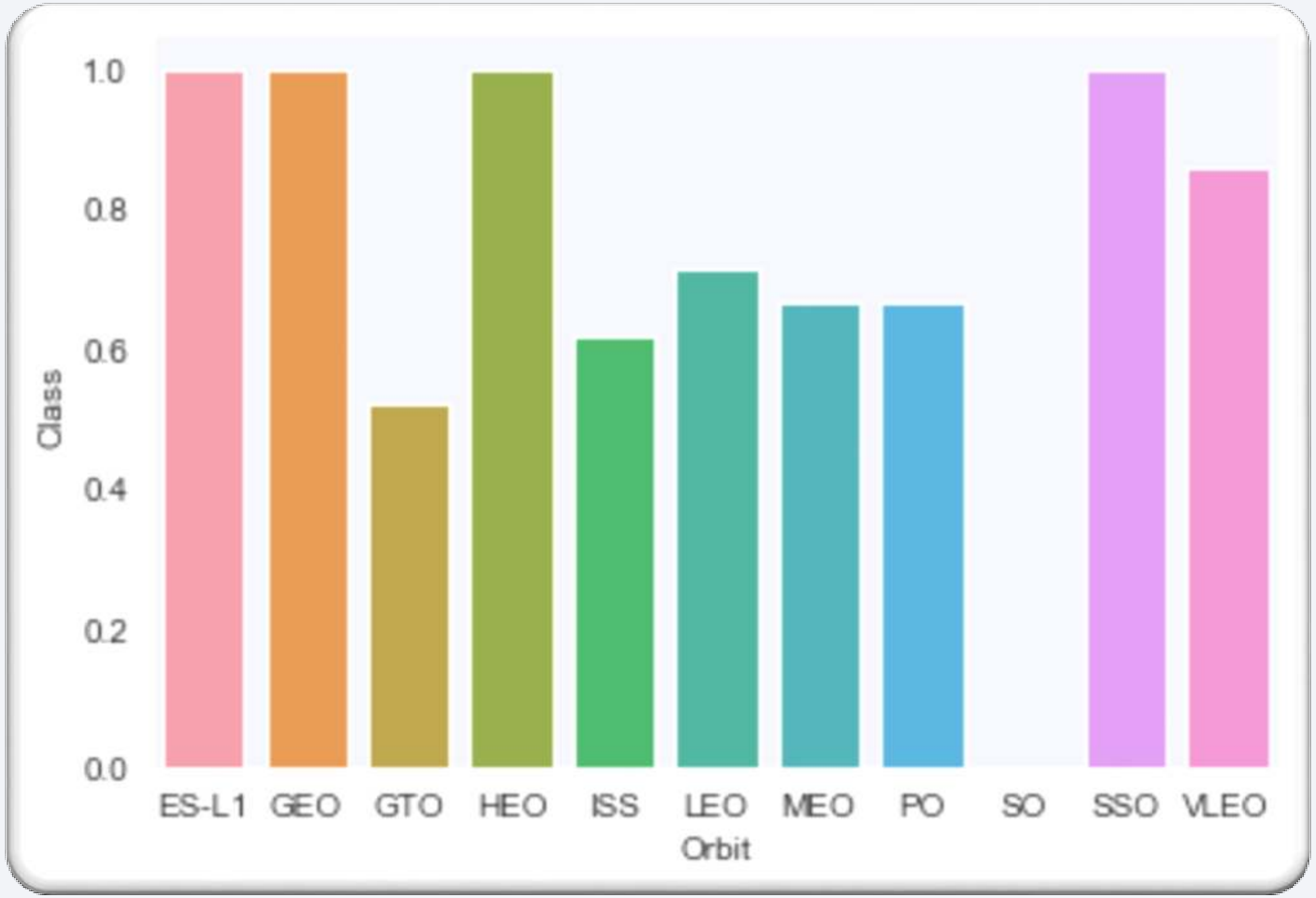
- The scatter plot of Launch Site vs. Payload Mass shows that:
- Above a payload mass of around 7000 kg, there are very few unsuccessful landings, but there is also far less data for these heavier launches.
- There is no clear correlation between payload mass and success rate for a given launch site.
- All sites launched a variety of payload masses, with most of the launches from CCAFS SLC 40 being comparatively lighter payloads (with some outliers).



Success Rate vs. Orbit Type

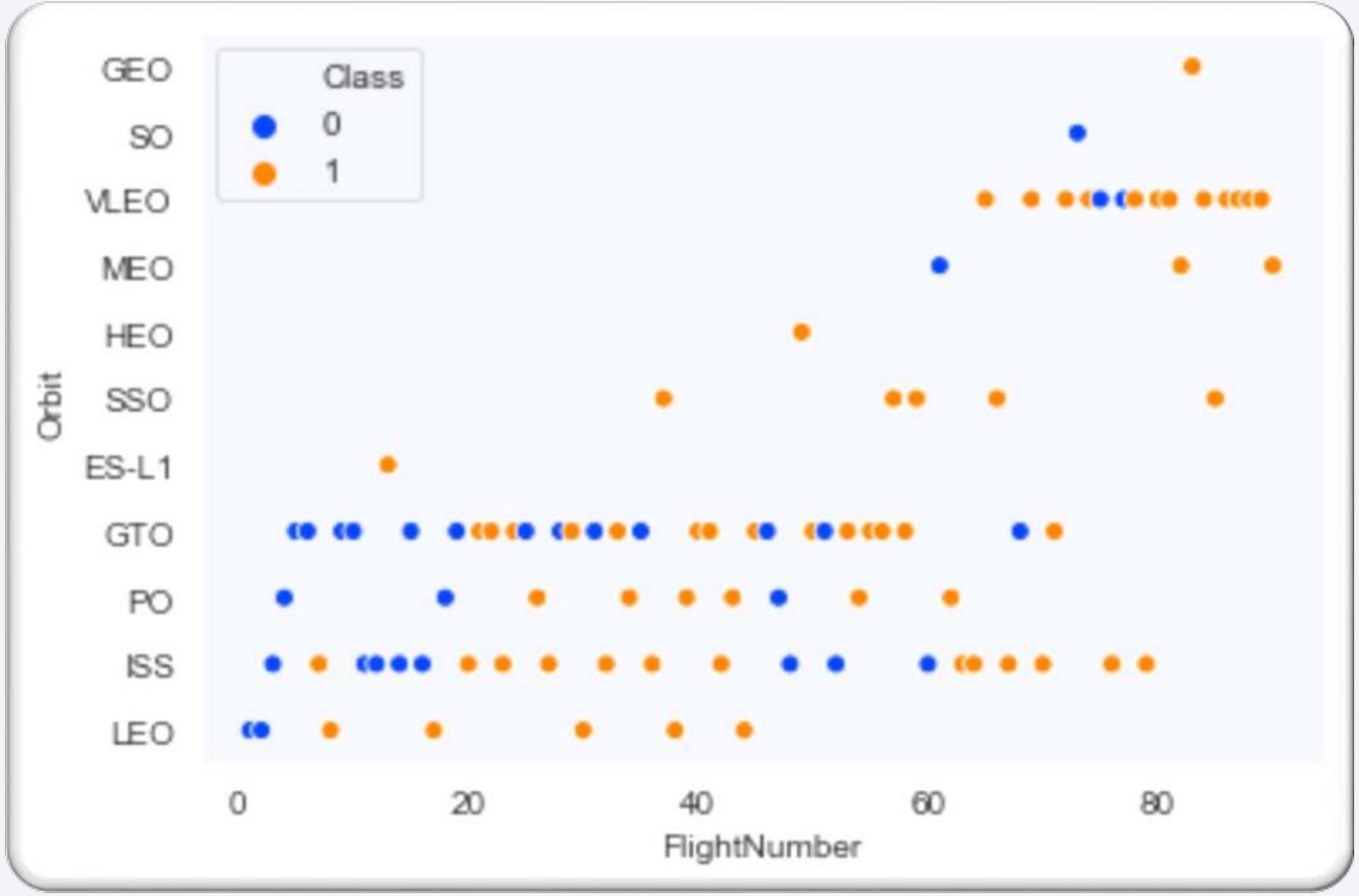
- The bar chart of Success Rate vs. Orbit Type shows that the following orbits have the highest (100%) success rate:
- ES-L1 (Earth-Sun First Lagrangian Point)
- GEO (Geostationary Orbit)
- HEO (High Earth Orbit)
- SSO (Sun-synchronous Orbit)

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



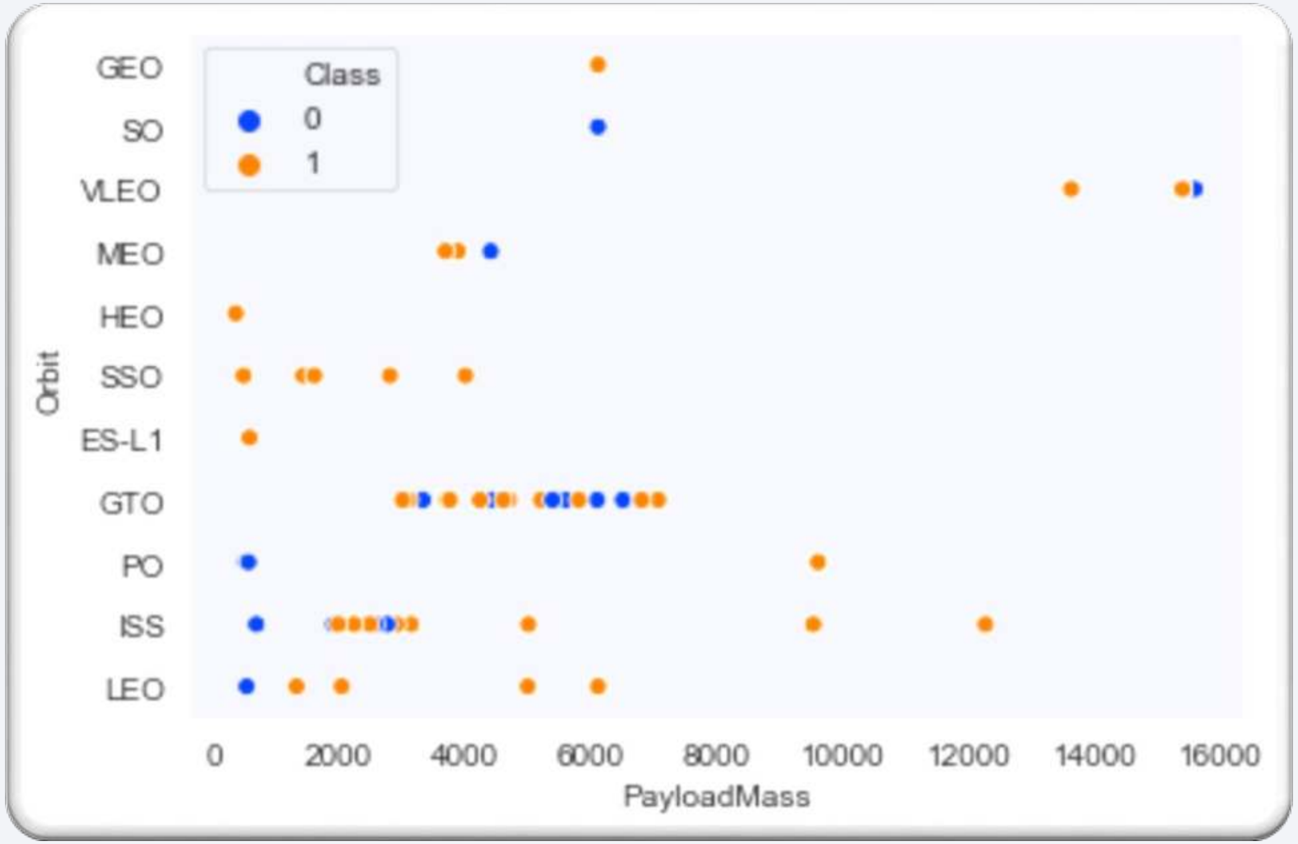
Flight Number vs. Orbit Type

- This scatter plot of Orbit Type vs. Flight number shows a few useful things that the previous plots did not, such as:
- The 100% success rate of GEO, HEO, and ES-L1 orbits can be explained by only having 1 flight into the respective orbits.
- The 100% success rate in SSO is more impressive, with 5 successful flights.
- There is little relationship between Flight Number and Success Rate for GTO.
- Generally, as Flight Number increases, the success rate increases. This is most extreme for LEO, where unsuccessful landings only occurred for the low flight numbers (early launches).



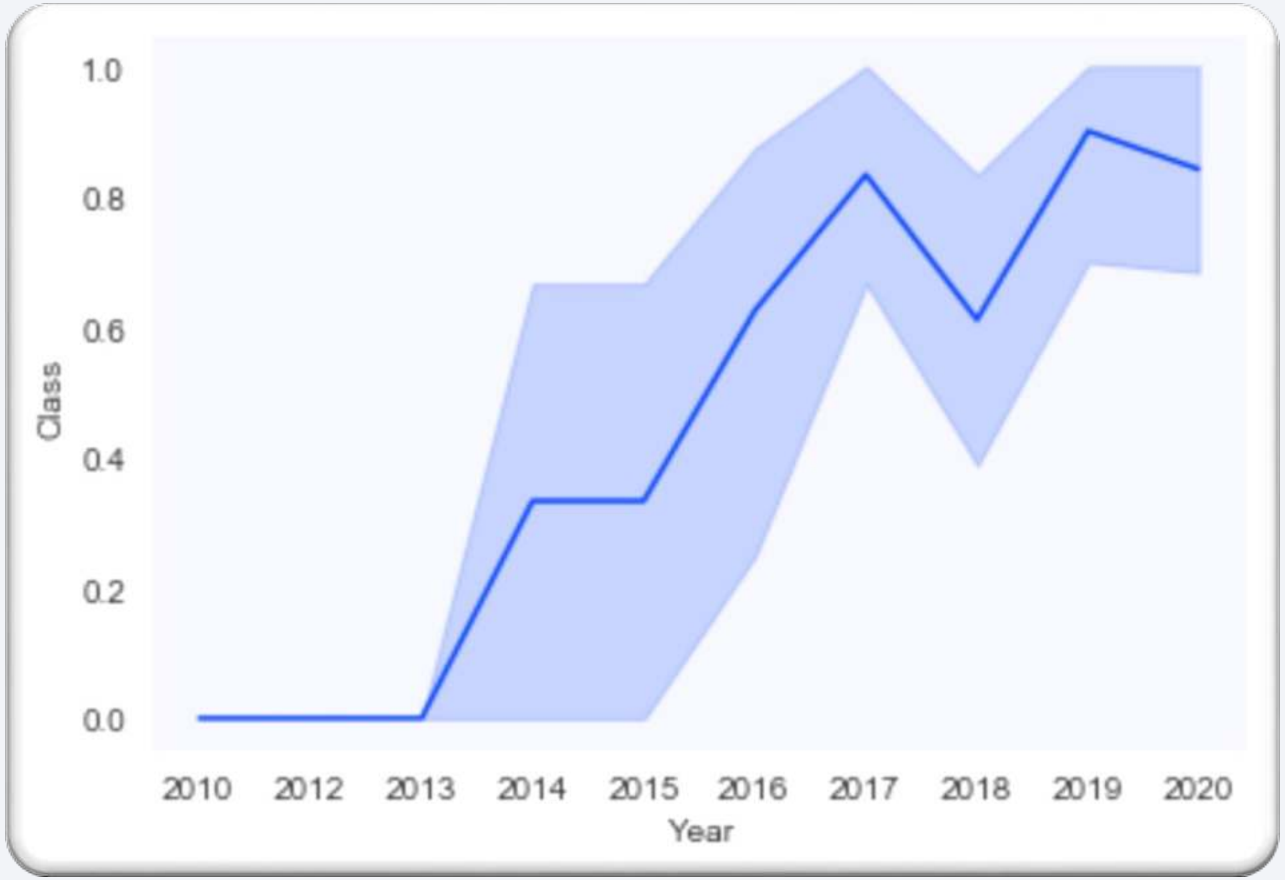
Payload vs. Orbit Type

- This scatter plot of Orbit Type vs. Payload Mass shows that:
- The following orbit types have more success with heavy payloads:
  - PO (although the number of data points is small)
  - ISS
  - LEO
- For GTO, the relationship between payload mass and success rate is unclear.
- VLEO (Very Low Earth Orbit) launches are associated with heavier payloads, which makes intuitive sense.



Launch Success Yearly Trend

- The line chart of yearly average success rate shows that:
- Between 2010 and 2013, all landings were unsuccessful (as the success rate is 0).
- After 2013, the success rate generally increased, despite small dips in 2018 and 2020.
- After 2016, there was always a greater than 50% chance of success.



All Launch Site Names

```
1 %sql SELECT UNIQUE(LAUNCH_SITE) FROM SPACEXTBL;
```

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

The word `UNIQUE` returns only unique values from the `LAUNCH_SITE` column of the `SPACEXTBL` table.

Launch Site Names 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

```
%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

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

average_payload_mass
2928

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

```
%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

```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

```
%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	99
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

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

booster_version
F9 B5 B1048.4
F9 B5 B1048.5
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1049.7
F9 B5 B1051.3
F9 B5 B1051.4
F9 B5 B1051.6
F9 B5 B1056.4
F9 B5 B1058.3
F9 B5 B1060.2
F9 B5 B1060.3

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.

2015 Launch Records

```
%sql SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL \
    WHERE (LANDING__OUTCOME = 'Failure (drone ship)') AND (EXTRACT(YEAR FROM DATE) = '2015');
```

booster_version	launch_site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

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

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

```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
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	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.

Section 3

Launch Sites Proximities Analysis

<Folium Map Screenshot 1>

- Replace <Folium map screenshot 1> title with an appropriate title
- Explore the generated folium map and make a proper screenshot to include all launch sites' location markers on a global map
- Explain the important elements and findings on the screenshot

<Folium Map Screenshot 2>

- Replace <Folium map screenshot 2> title with an appropriate title

- Explore the folium map and make a proper screenshot to show the color-labeled launch outcomes on the map

- Explain the important elements and findings on the screenshot

<Folium Map Screenshot 3>

- Replace <Folium map screenshot 3> title with an appropriate title

- Explore the generated folium map and show the screenshot of a selected launch site to its proximities such as railway, highway, coastline, with distance calculated and displayed

- Explain the important elements and findings on the screenshot

Section 4

Build a Dashboard with Plotly Dash

<Dashboard Screenshot 1>

- Replace <Dashboard screenshot 1> title with an appropriate title
- Show the screenshot of launch success count for all sites, in a piechart
- Explain the important elements and findings on the screenshot

<Dashboard Screenshot 2>

- Replace <Dashboard screenshot 2> title with an appropriate title
- Show the screenshot of the piechart for the launch site with highest launch success ratio
- Explain the important elements and findings on the screenshot

<Dashboard Screenshot 3>

- Replace <Dashboard screenshot 3> title with an appropriate title
- Show screenshots of Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider
- Explain the important elements and findings on the screenshot, such as which payload range or booster version have the largest success rate, etc.

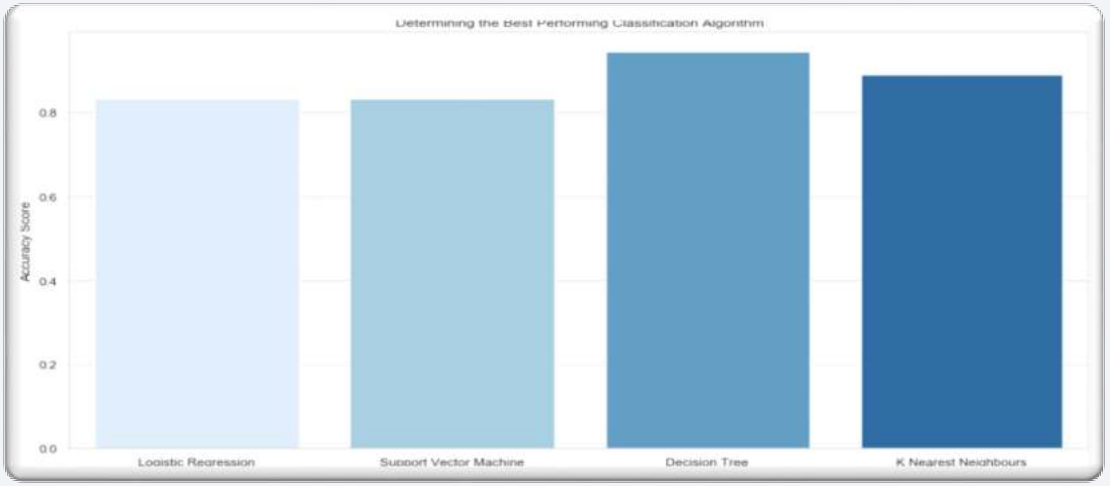
Section 5

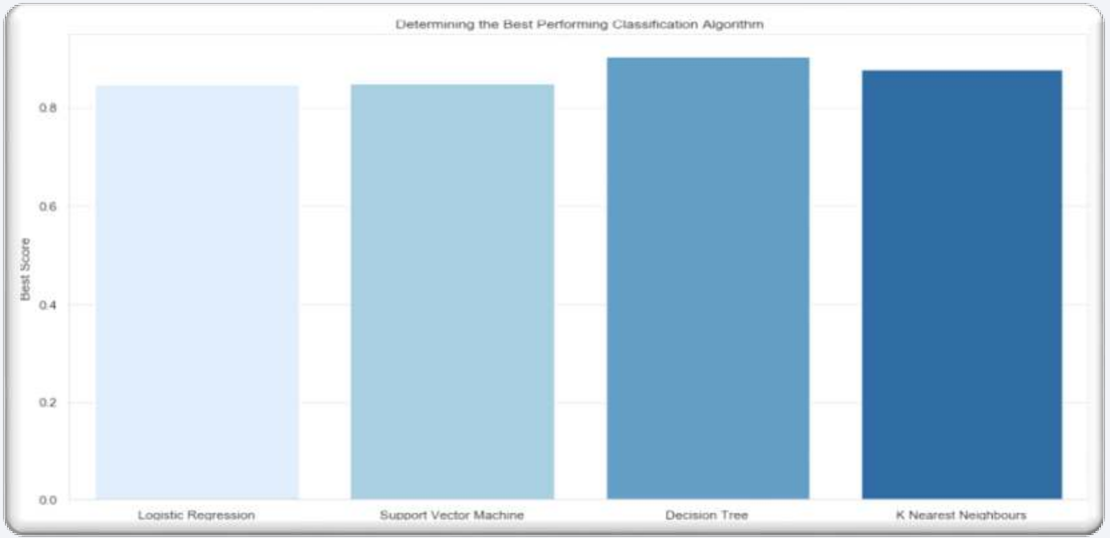
Predictive Analysis (Classification)

Classification Accuracy

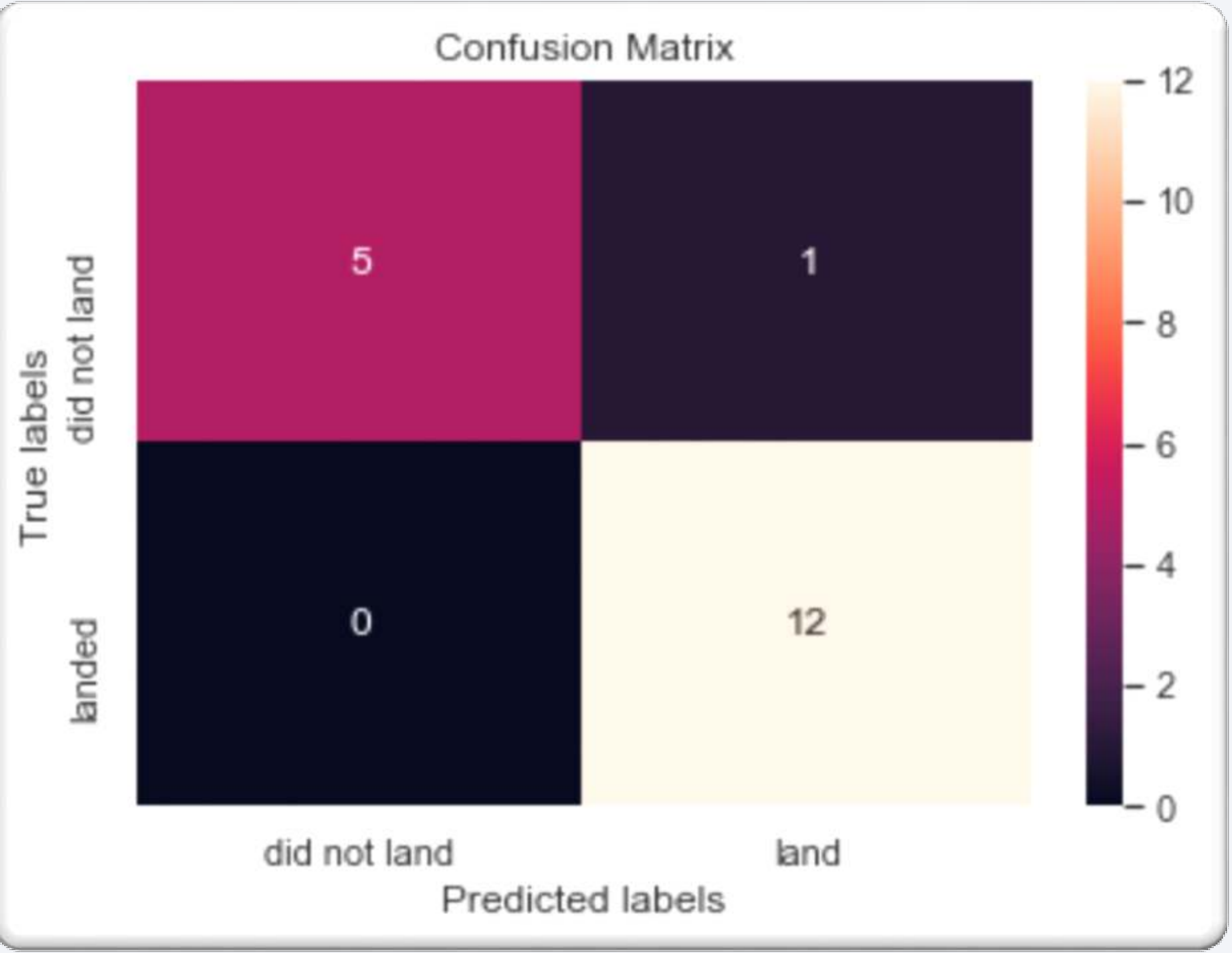
- Plotting the Accuracy Score and Best Score for each classification algorithm produces the following result:
- The Decision Tree model has the highest classification accuracy
  - The Accuracy Score is 94.44%
  - The Best Score is 90.36%

Algorithm	Accuracy Score	Best Score
Logistic Regression	0.833333	0.846429
Support Vector Machine	0.833333	0.848214
Decision Tree	0.944444	0.903571
K Nearest Neighbours	0.888889	0.876786





Confusion Matrix



- As shown previously, best performing classification model is the Decision Tree model, with an accuracy of 94.44%.
- This is explained by the confusion matrix, which shows only 1 out of 18 total results classified incorrectly (a false positive, shown in the top-right corner).
- The other 17 results are correctly classified (5 did not land, 12 did land).

Conclusions

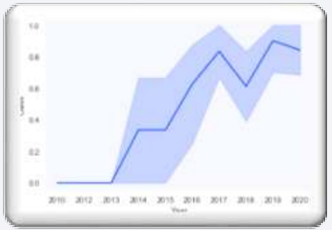
- **As the number of flights increases, the rate of success at a launch site increases, with most early flights being unsuccessful. I.e. with more experience, the success rate increases.**
  - **Between 2010 and 2013, all landings were unsuccessful (as the success rate is 0).**
  - **After 2013, the success rate generally increased, despite small dips in 2018 and 2020.**
  - **After 2016, there was always a greater than 50% chance of success.**

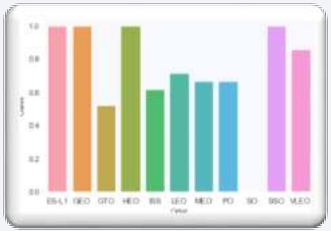
- **Orbit types ES-L1, GEO, HEO, and SSO, have the highest (100%) success rate.**
  - **The 100% success rate of GEO, HEO, and ES-L1 orbits can be explained by only having 1 flight into the respective orbits.**
  - **The 100% success rate in SSO is more impressive, with 5 successful flights.**
  - **The orbit types PO, ISS, and LEO, have more success with heavy payloads:**
  - **VLEO (Very Low Earth Orbit) launches are associated with heavier payloads, which makes intuitive sense.**

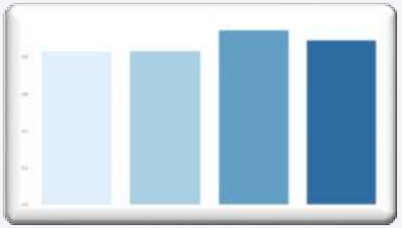
- **The launch site KSC LC-39 A had the most successful launches, with 41.7% of the total successful launches, and also the highest rate of successful launches, with a 76.9% success rate.**

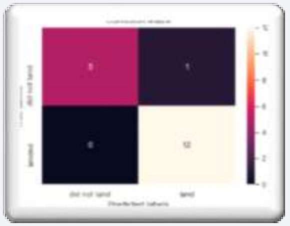
- **The success for massive payloads (over 4000kg) is lower than that for low payloads.**

- **The best performing classification model is the Decision Tree model, with an accuracy of 94.44%.**









Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project