Winning Space Race with Data Science

Franco Stuer
14-03-2023

Outline

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

Executive Summary

The current project aims to build a model that is able to predict outcomes of rocket launches from the company Space X by using machine learning algorithms applied in past launches data.
Due to the technical complexity of a rocket launch, there are a number of variables that can impact the outcome of a launch. Therefore, a preliminary analisis is needed to select the significant variables to feed to the model.
The data was retrieved from previous Space X launches as well as other public sources, and it needed to be adapted to the required inputs of the different models.
Model selection was done by comparing different classification algorithms and looking at it’s accuracy on the test data. The involved models were a Logistic Regression model, a Support Vector Machine model, a Decision Tree and a K-nearest neighbour model.

Introduction

- The main goal of the project is to create a machine learning model that is able to accurately predict the outcome of a given launch.

- Some questions that had to be answered to achieved this were:
- Which machine learning model adjusts best to the problem?
- What variables should be used to predict the outcome?
- Which variables should be left out?

- In order to answer those questions, data wrangling, exploratory data analysis and data visualization were used, to finally have the means to construct the machine learning model.

Section
1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - SPACE X API, Wikipedia web scraping
- Perform data wrangling
  - Removal and replacement of null values
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - GridSearchCV, sklearn metrics

Data Collection

There were two main sources of data:

1 – The SpaceX Data API: https://api.spacexdata.com/v4/launches/past

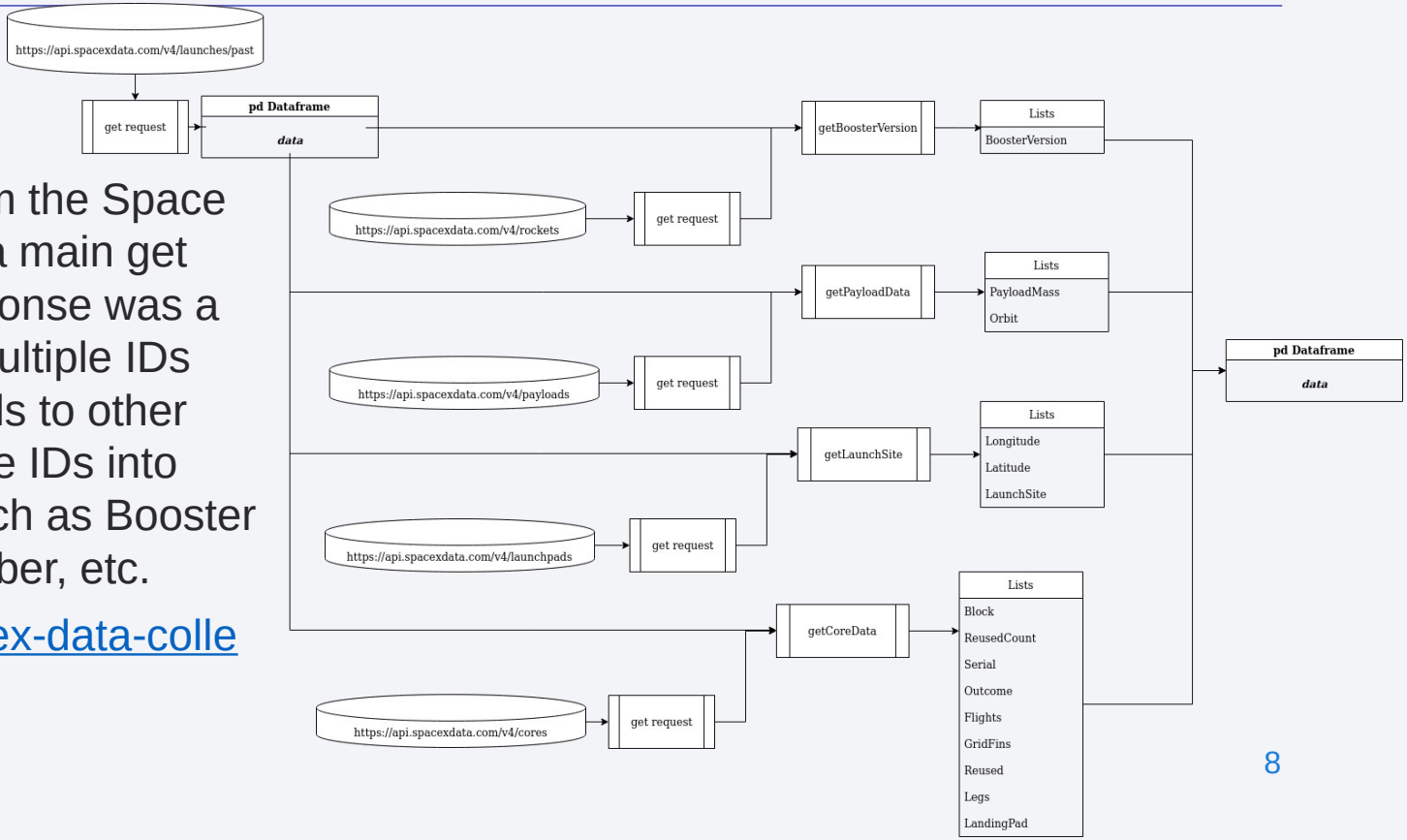
2 – The Wikipedia *List of Falcon 9 and Falcon Heavy launches page*

Whereas the first one provided an API to download json data that can be parsed/ normalized using the Pandas library, the Wikipedia page had to be scraped to obtain data from its tables using Beautiful Soup.

Data Collection – SpaceX API

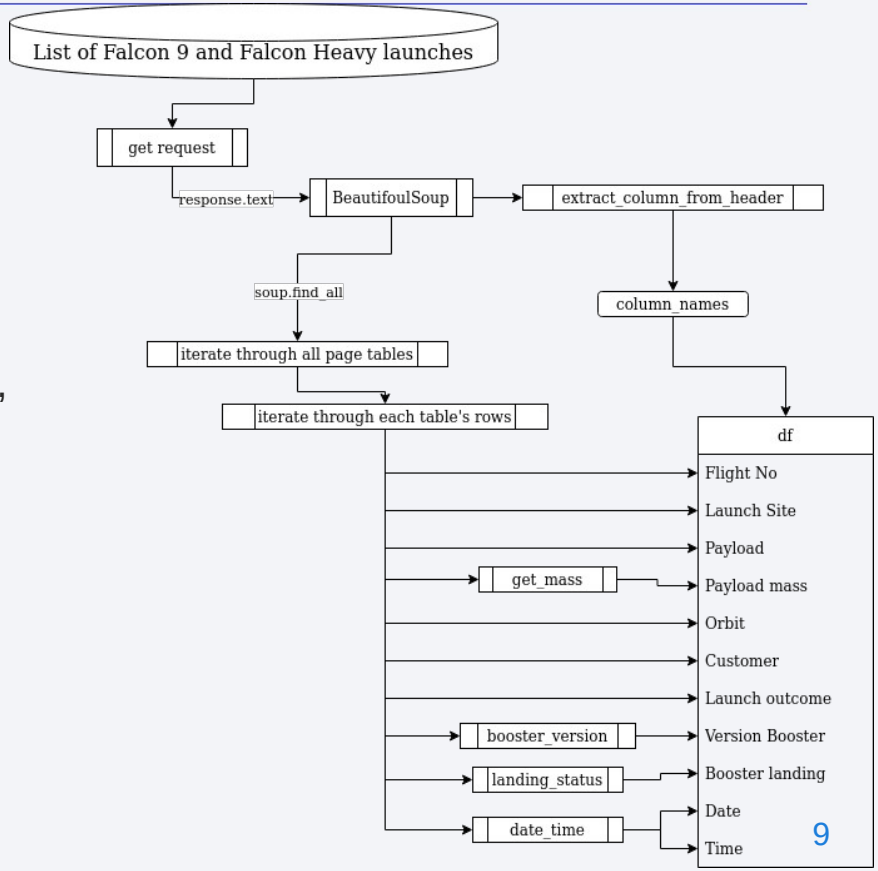
- Collecting data from the Space X API consisted of a main get request whose response was a json file including multiple IDs and subsequent calls to other files to decode those IDs into known variables such as Booster Version, Serial number, etc.

[00_jupyter-labs-spacex-data-collection-api.ipynb]



Data Collection - Scraping

- After the get request from the Wikipedia URL, a BeautifulSoup object is created which allows to iterate over the different HTML objects of the response text.
- It is then possible to identify the columns by a helper function (extract_column_from_header) and with the help of another four helper functions, all the fields are parsed and stored as a Pandas DataFrame



[01_jupyter-labs-webscraping.ipynb]

Data Wrangling

The main Data Wrangling tasks performed were:

- Used `df.isnull().sum()` to identify the number of null values on each column.
- The null values in “PayloadMass” column were replaced by its mean value by doing `df.loc[df["PayloadMass"].isnull()] = df.loc[:,"PayloadMass"].mean()`
- Calculated the number of launches on each site with `df["LaunchSite"].value_counts()`
- Calculated the number and occurrence of each orbit with `df["Orbit"].value_counts()`
- Calculated the number and occurence of mission outcome per launch site type
- Created a new column named Class to represent the outcome of each launch

[02_labs-jupyter-spacex-Data%20wrangling.ipynb]

EDA with Data Visualization

- Scatter plots
  - Flight number vs. payload mass, including the outcome as hue.
  - Flight number vs launch site, including the outcome as hue.
  - Payload mass vs launch site, to display relationships between the payload, launch site and outcome
  - Flight number vs orbit type, including the outcome as hue.
  - Payload vs orbit type, including the outcome as hue.
- Bar plot showing success rate for each orbit
- Line plot showing success trend over the years

04_jupyter-labs-eda-dataviz.ipynb

EDA with SQL

SQL Queries were used to performed the following tasks:

- Displayed the names of the unique launch sites.
- Displayed 5 records where launch sites begin with 'CCA'.
- Displayed the total payload mass carried by boosters launched by NASA (CRS).
- Displayed average payload mass carried by booster version F9 v1.1.
- Listed the date when the first successful landing outcome in ground pad was achieved.
- Listed the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- Listed the total number of successful and failure mission outcomes
- Used a subquery to list the names of the booster_versions which have carried the maximum payload mass.
- Listed the records which will display the month names, failure landing outcomes in drone ship ,booster versions, launch _site for the months in year 2015.
- Ranked the count of successful landing outcomes between the date 04-06-2010 and 20-03-2017 in descending order

03_jupyter-labs-eda-sql-coursera_sqllite2.ipynb

Build an Interactive Map with Folium

Below is a summary of the folium objects used:

- folium.Circle() draws circle overlays on a map in the specified location and format, it was used to mark different launch sites in a map.
- folium.map.Marker() creates a marker with optional text or an icon that can be specified by DivIcon(). They were also used to mark different launch sites.
- MarkerCluster() creates a cluster to group multiple markers, for more visual clarity.
- MousePosition() was used to display the latitude and longitude of the mouse position.
- folium.PolyLine() was used to create a line between different points in the map, to join the closes point on the coastline, road, railway and city.

05_lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

A **dash** application was built to perform interactive visual analytics on SpaceX launch data in real-time.

It contains input components such as a dropdown to select amongst all launch sites list and a range slider to select the payload range. Those input components interact with a pie chart and a scatter point chart that react to the selection.

The Pie chart displays the success/failure rate of launches in the selected launch site or the percentage of successful launches if “All Sites” is selected.

The scatter plot shows the payload for each launch for the select site (or all sites) versus the launch outcome sets a different colour for each booster version

https://github.com/francostuer/ibm_capstone_project/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

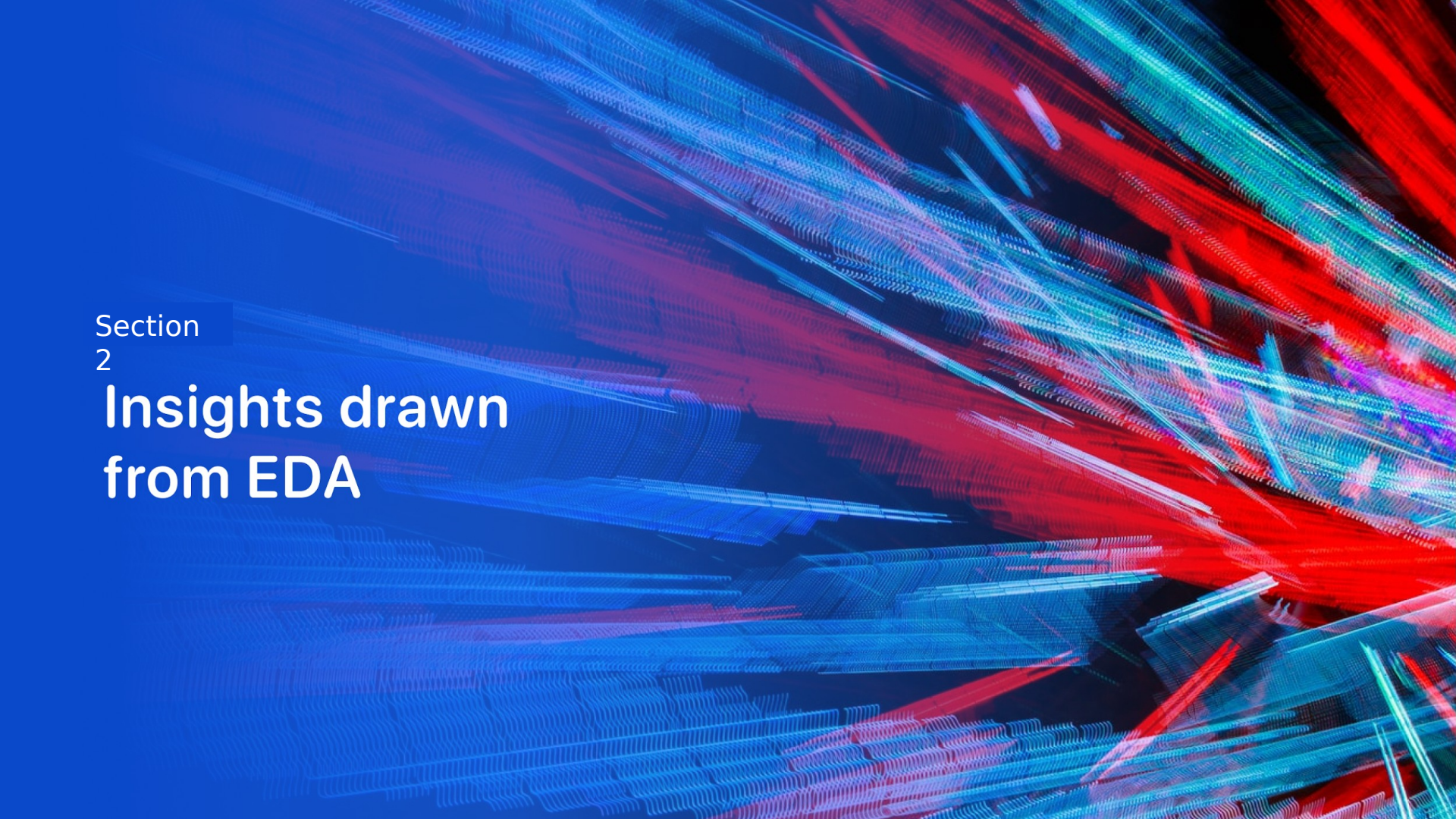
The outcome of the launch, namely `df["Class"]` was set to be the label and `StandardScaler()` was used to standardize all the features by removing the mean and scaling to unit variance. `train_test_split()` was used to randomly split the data into train and test sets with a test size of 20% .

Four different classifiers were trained:

- Logistic Regression,
- Support Vector Machine,
- Decision Tree,
- K-nearest Neighbour.

`GridSearchCV()` was used to perform cross validation over a given set of parameters for each type of classifier and determine which parameters provide the most accurate results for each classifier. `Best_params_` and `best_score_` were used to obtain the best set of parameters and the respective best score.

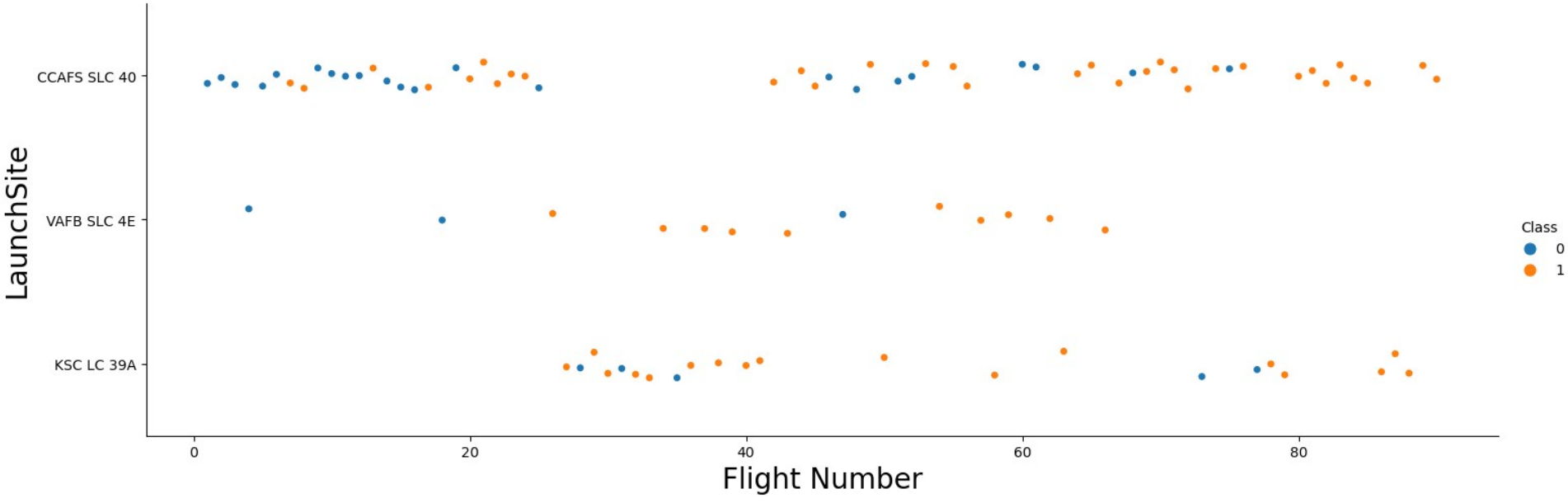
06_SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb



Section 2

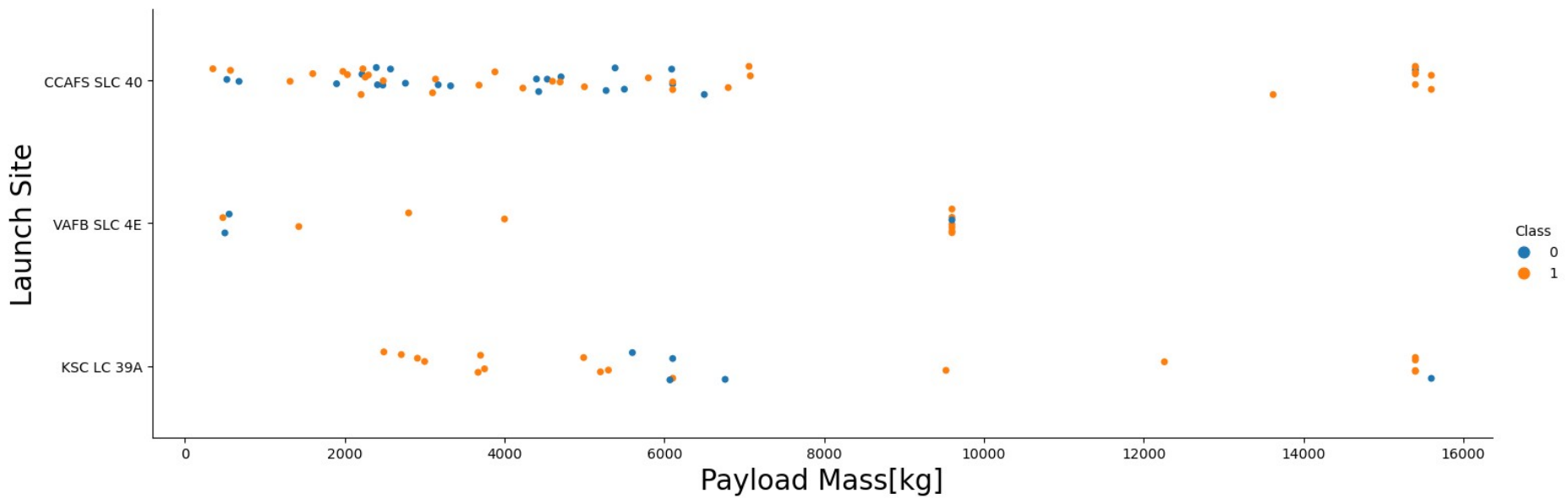
Insights drawn from EDA

Flight Number vs. Launch Site



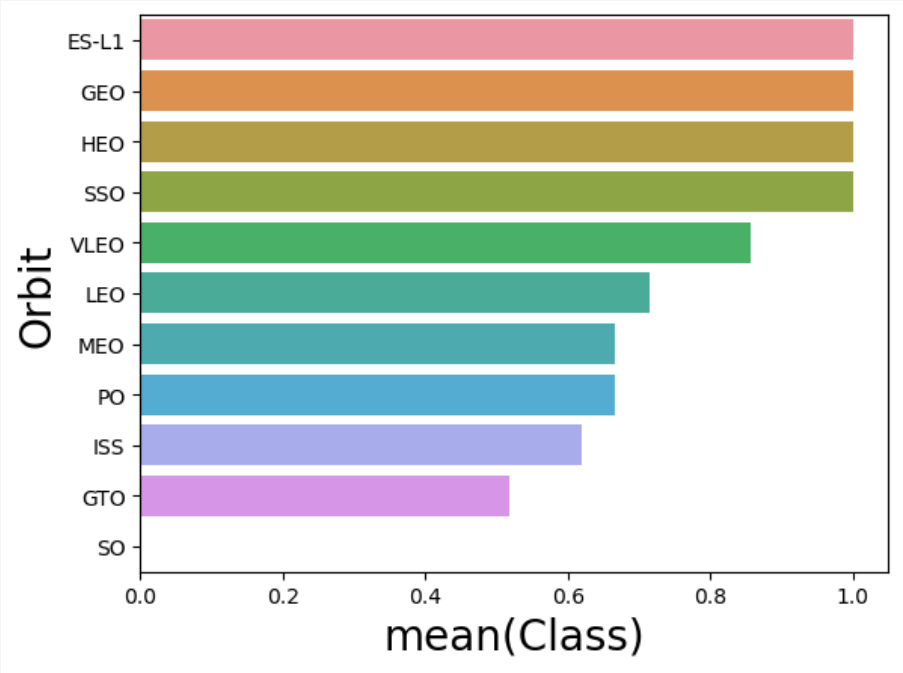
This plot makes evident an increase in the success rate with the number of flights, particularly for launches in CCAFS SLC40

Payload vs. Launch Site

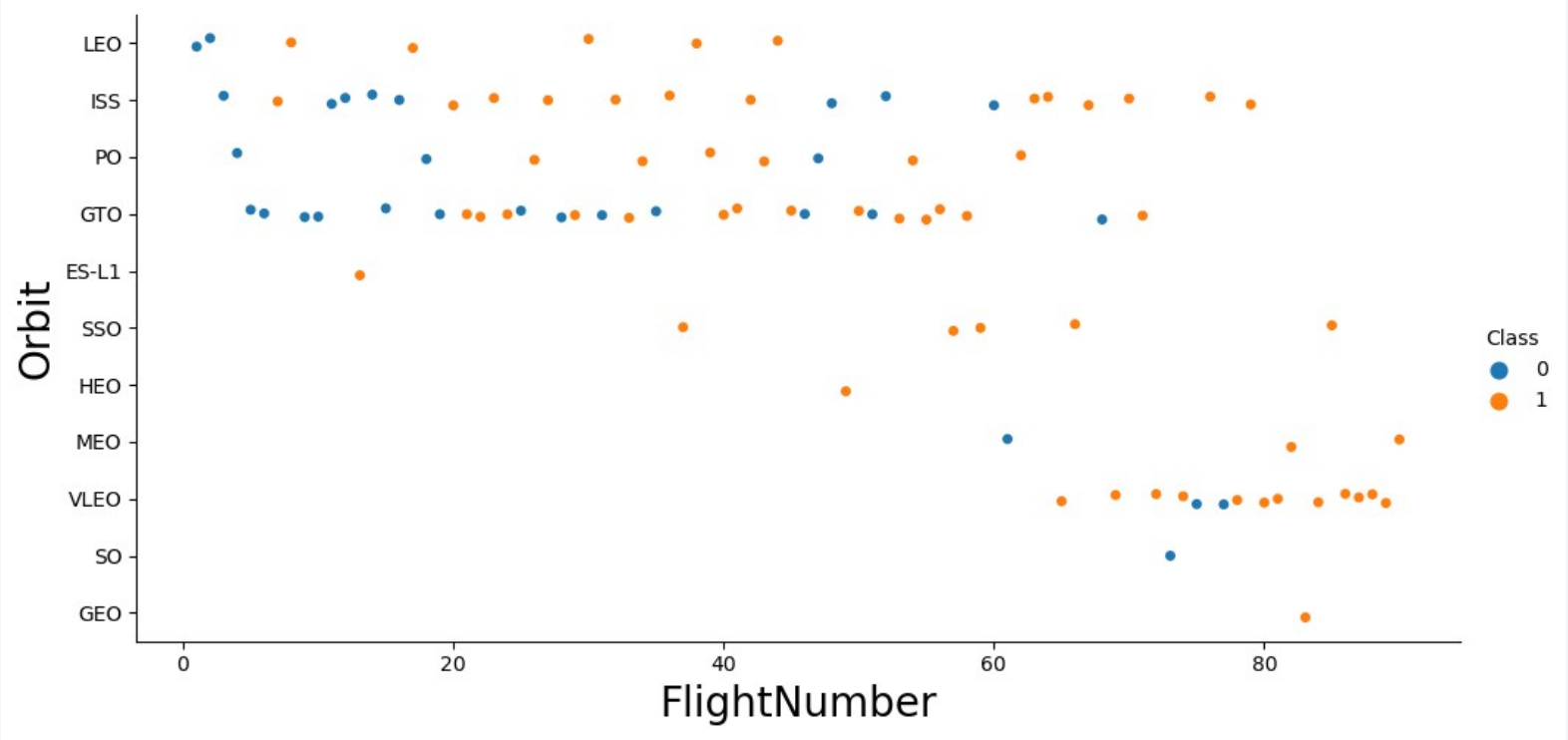


We can see that no flights on the VAFB SLC 4E had a payload over 10.000kg

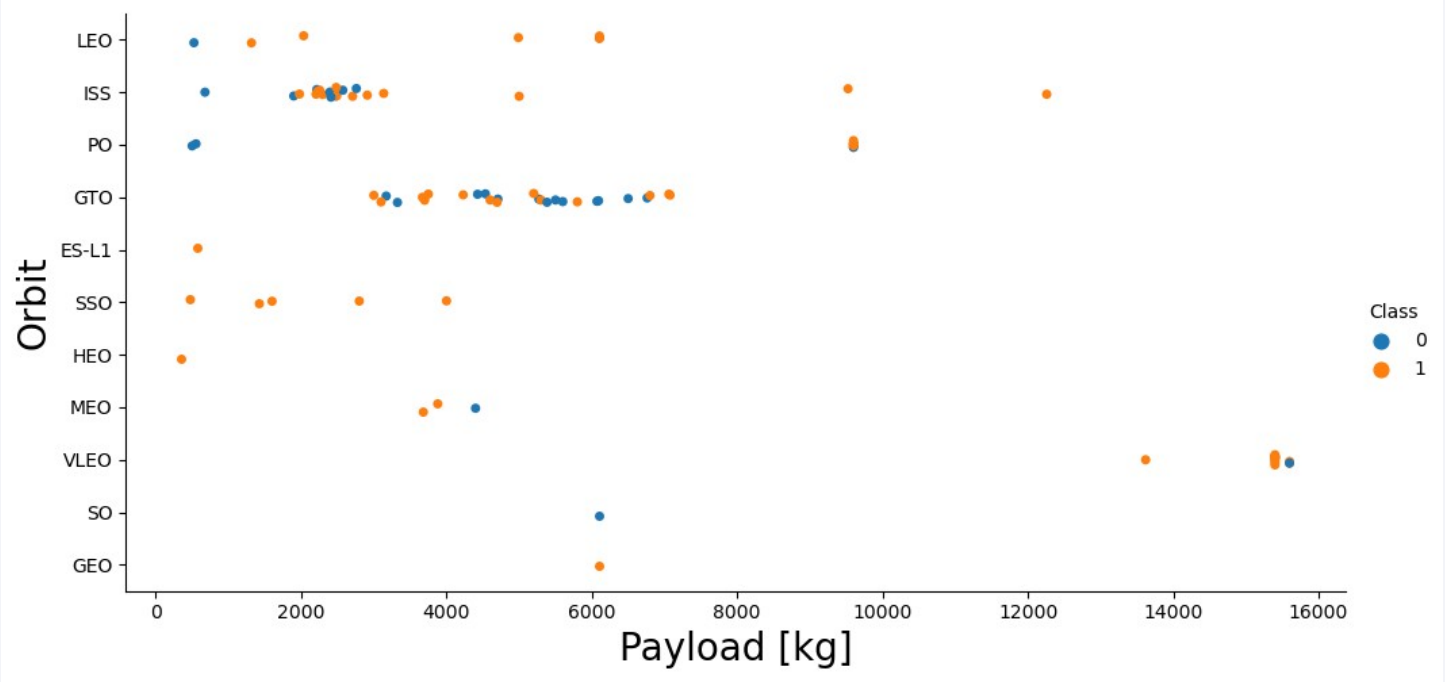
Success Rate vs. Orbit Type



Flight Number vs. Orbit Type

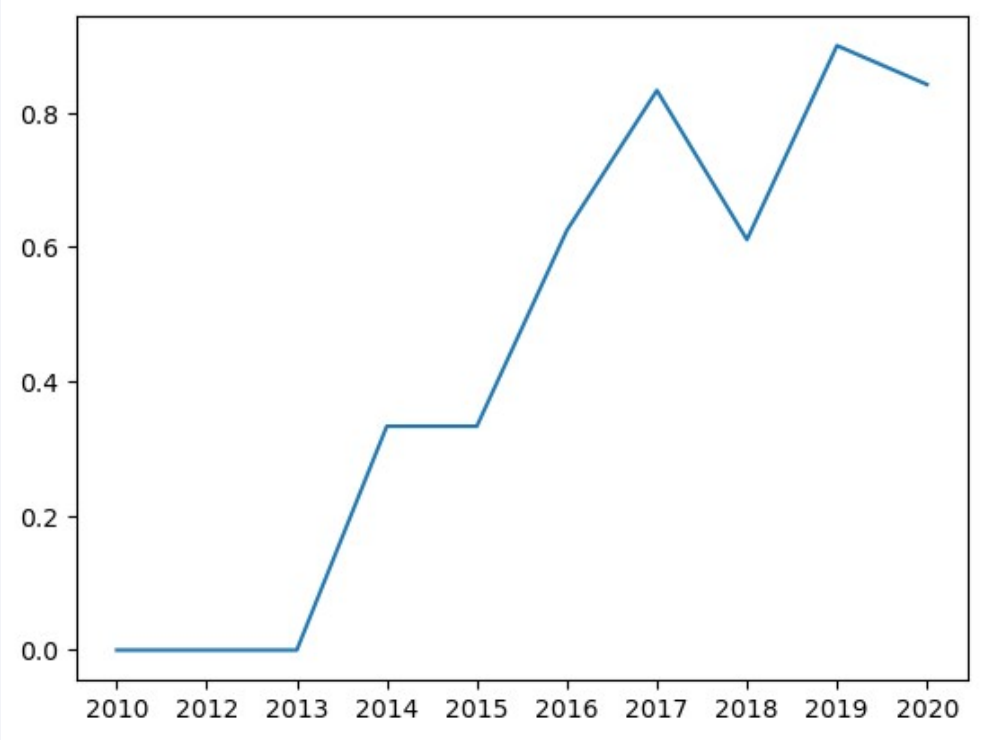


Payload vs. Orbit Type



Heaviest payloads were sent to Very Low Orbits or ISS

Launch Success Yearly Trend



The plot shows a clear improvement in the success rate over the years

All Launch Site Names

```
%%sql
SELECT DISTINCT("Launch_Site") FROM "SPACEXTBL"
```

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

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

The DISTINCT function was applied to the “Launch_Site” column of the “SPACEXTBL” table, which returns all unique values

Launch Site Names Begin with 'CCA'

```
%%sql
SELECT * FROM "SPACEXTBL"
WHERE("Launch_Site" LIKE "CCA%")
LIMIT 5
```

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

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

The WHERE … LIKE condition is applied to select only records where the “Launch_Site” column value starts with “CCA”

Total Payload Mass

```
%%sql
WITH "SUBTABLE" AS
    (SELECT "Customer", SUM("PAYLOAD_MASS__KG_") AS TOTAL_PAYLOAD
    FROM "SPACEXTBL" GROUP BY "Customer")
SELECT "Customer", "TOTAL_PAYLOAD"
FROM "SUBTABLE"
WHERE "Customer"= "NASA (CRS)"
```

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

Customer	TOTAL_PAYLOAD
NASA (CRS)	45596

A subtable is created with “Customer” and the sum of the payload mass grouped by each customer. In the outer query, the customer is set to match NASA (CRS)

Average Payload Mass by F9 v1.1

```
%%sql
WITH "SUBTABLE" AS
    (SELECT "Booster_Version", "PAYLOAD_MASS__KG_"
     FROM "SPACEXTBL"
     WHERE "Booster_Version" LIKE "F9 v1.1%")

SELECT AVG("PAYLOAD_MASS__KG_") AS "F9 v1.1 AVG PAYLOAD" FROM "SUBTABLE"
```

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

F9 v1.1 AVG PAYLOAD
2534.6666666666665

Again a subtable including only Booster Versions starting with v1.1.
Then the average value is displayed.

First Successful Ground Landing Date

```
%%sql
WITH "SUBTABLE" AS
    (SELECT "Date",
        SUBSTRING("Date",1,2) AS "DAY",
        SUBSTRING("Date",4,2) AS "MONTH",
        SUBSTRING("Date",7,4) AS "YEAR",
        "Landing _Outcome"
    FROM "SPACEXTBL" WHERE "Landing _Outcome" = "Success (ground pad)")
SELECT "DAY","MONTH",MIN("YEAR") AS "YEAR" FROM "SUBTABLE"
```

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

DAY	MONTH	YEAR
22	12	2015

The SUBSTRING function is used to strip the date values. The MIN function was then applied to the YEAR column which returned a single value

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%%sql
WITH "SUBTABLE" AS
    (SELECT "Booster_Version", "PAYLOAD_MASS__KG_", "Landing _Outcome"
    FROM "SPACEXTBL"
    WHERE "Landing _Outcome" = "Success (drone ship)")
SELECT DISTINCT("Booster_Version") FROM "SUBTABLE" WHERE "PAYLOAD_MASS__KG_" BETWEEN 4000 AND 6000;
```

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

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

First all the successful flights are selected as a subtable, then the unique booster versions that carried a payload between 4000 and 6000kg are retrieved.

Total Number of Successful and Failure Mission Outcomes

```
%%sql
SELECT( SELECT COUNT("Mission_Outcome") AS "Success count"
    FROM "SPACEXTBL"
    WHERE "Mission_Outcome" LIKE "Success%") AS "Success count",

    (SELECT COUNT("Mission_Outcome") AS "Failure count"
    FROM "SPACEXTBL"
    WHERE "Mission_Outcome" LIKE "Failure%") AS "Failure count"
```

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

Success count	Failure count
100	1

The COUNT function is together with a WHERE … LIKE clause. To find Outcomes starting with Success or Failure.

Boosters Carried Maximum Payload

```
%%sql
SELECT DISTINCT("Booster_Version")
FROM "SPACEXTBL"
WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_MASS__KG_") FROM "SPACEXTBL")
```

```
 * 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

The unique booster versions that carried a payload equal to the maximum are selected

2015 Launch Records

```
%%sql
WITH "SUBTABLE" AS
    (SELECT SUBSTR("Date", 4, 2) AS "Month",
    "Landing _Outcome",
    "Booster_Version",
    "Launch_Site"
    FROM "SPACEXTBL"
    WHERE SUBSTR("Date", 7, 4)="2015")
SELECT * FROM "SUBTABLE" WHERE "Landing _Outcome"="Failure (drone ship)"
```

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

Month	Landing _Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

Used SUBSTR to select the Date string from the 7th position with a length of 4 characters to retrieve the year. Then selected the records where the outcome was a Failure.

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

```
%%sql
WITH SUBTABLE AS (SELECT "Date", SUBSTR("Date", 7, 4) || SUBSTR("Date", 4, 2) || SUBSTR("Date", 1, 2)
    AS "Date_Formatted", "Landing _Outcome"
FROM SPACEXTBL
WHERE "Date_Formatted" BETWEEN "20100604" AND "20170320"
ORDER BY "Date_Formatted" DESC)
SELECT "Date", "Landing _Outcome" FROM "SUBTABLE" WHERE "Landing _Outcome" LIKE "Success%"
```

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

Date	Landing _Outcome
19-02-2017	Success (ground pad)
14-01-2017	Success (drone ship)
14-08-2016	Success (drone ship)
18-07-2016	Success (ground pad)
27-05-2016	Success (drone ship)
06-05-2016	Success (drone ship)
08-04-2016	Success (drone ship)
22-12-2015	Success (ground pad)

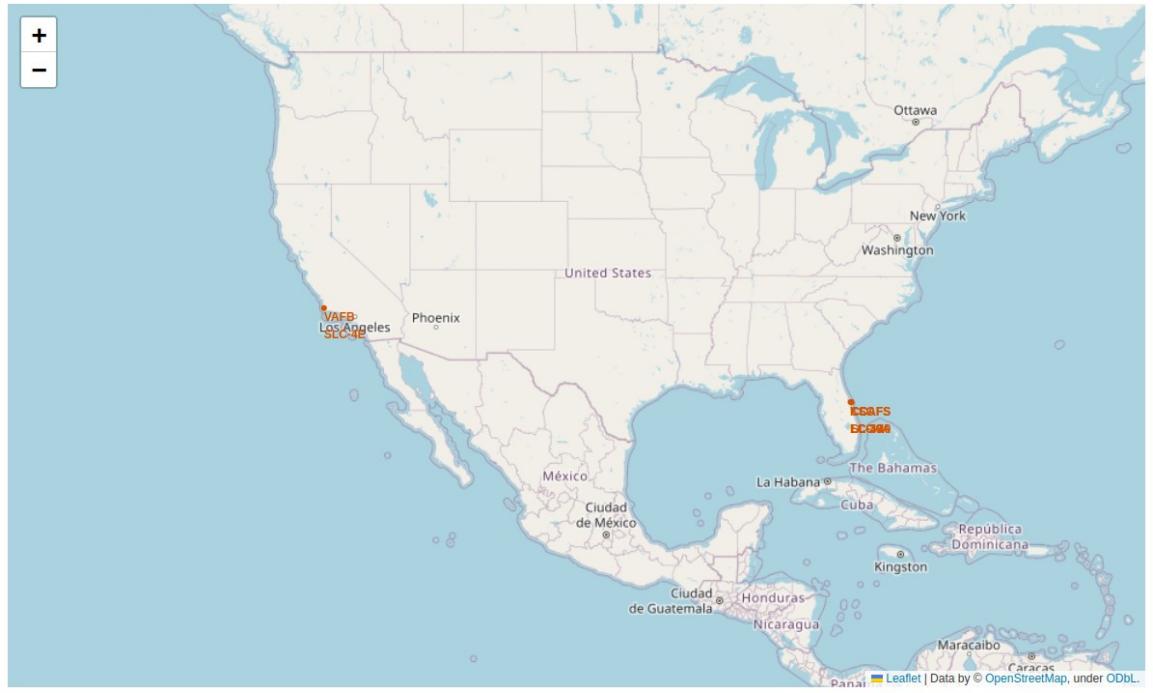
Inverted the date to a YYYYMMDD format to be able to use a WHERE… BETWEEN clause. Then the outcomes where filtered to start with “Success”

Section 3

Launch Sites Proximities Analysis

Launch Sites in Folium

The map displays the location of VAFB SLC-4E in California, and KSC LC-39A, CCAFS SLC-40 and CCAFS LC-40 in Florida

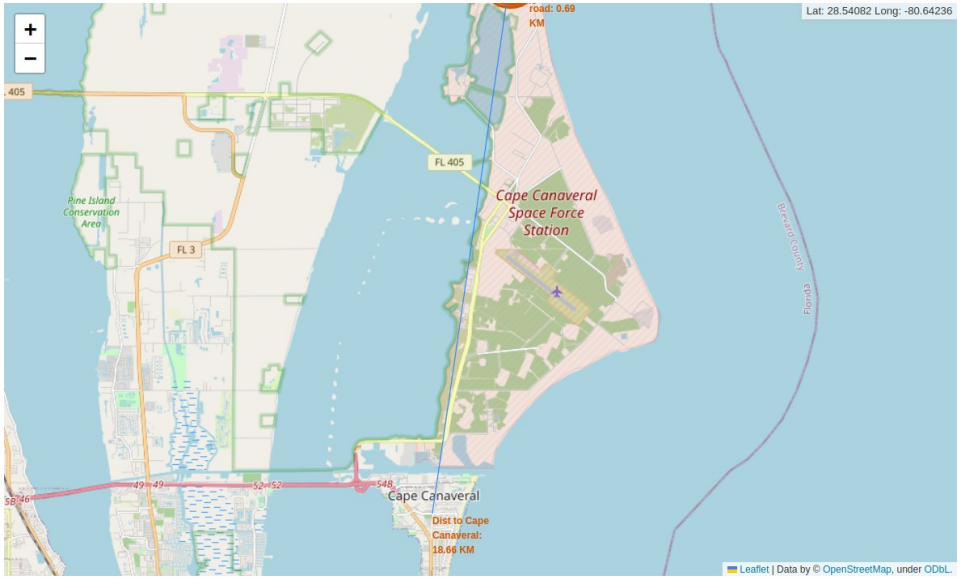


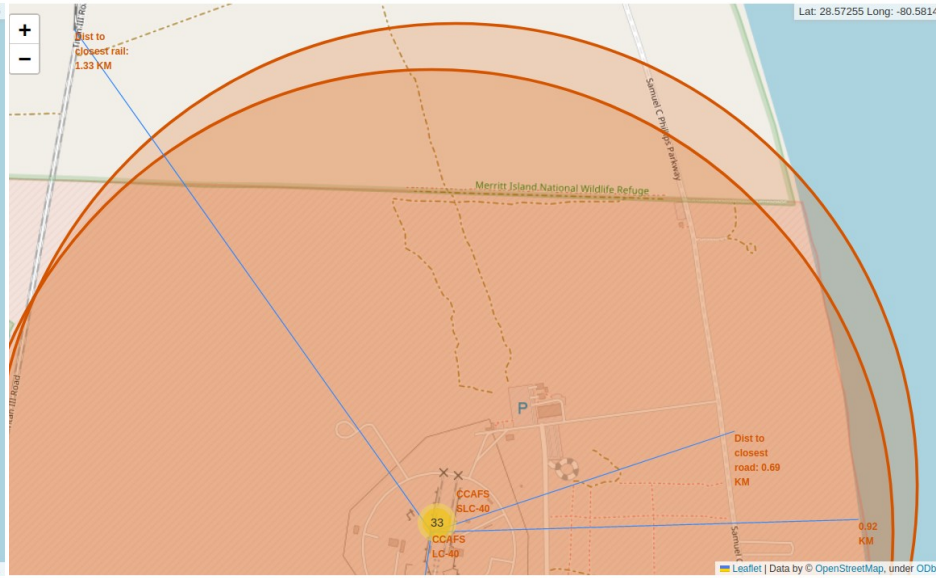
Flight Outcome in a Folium Map

The outcome of each flight for each location can be seen based in the color of the marker.

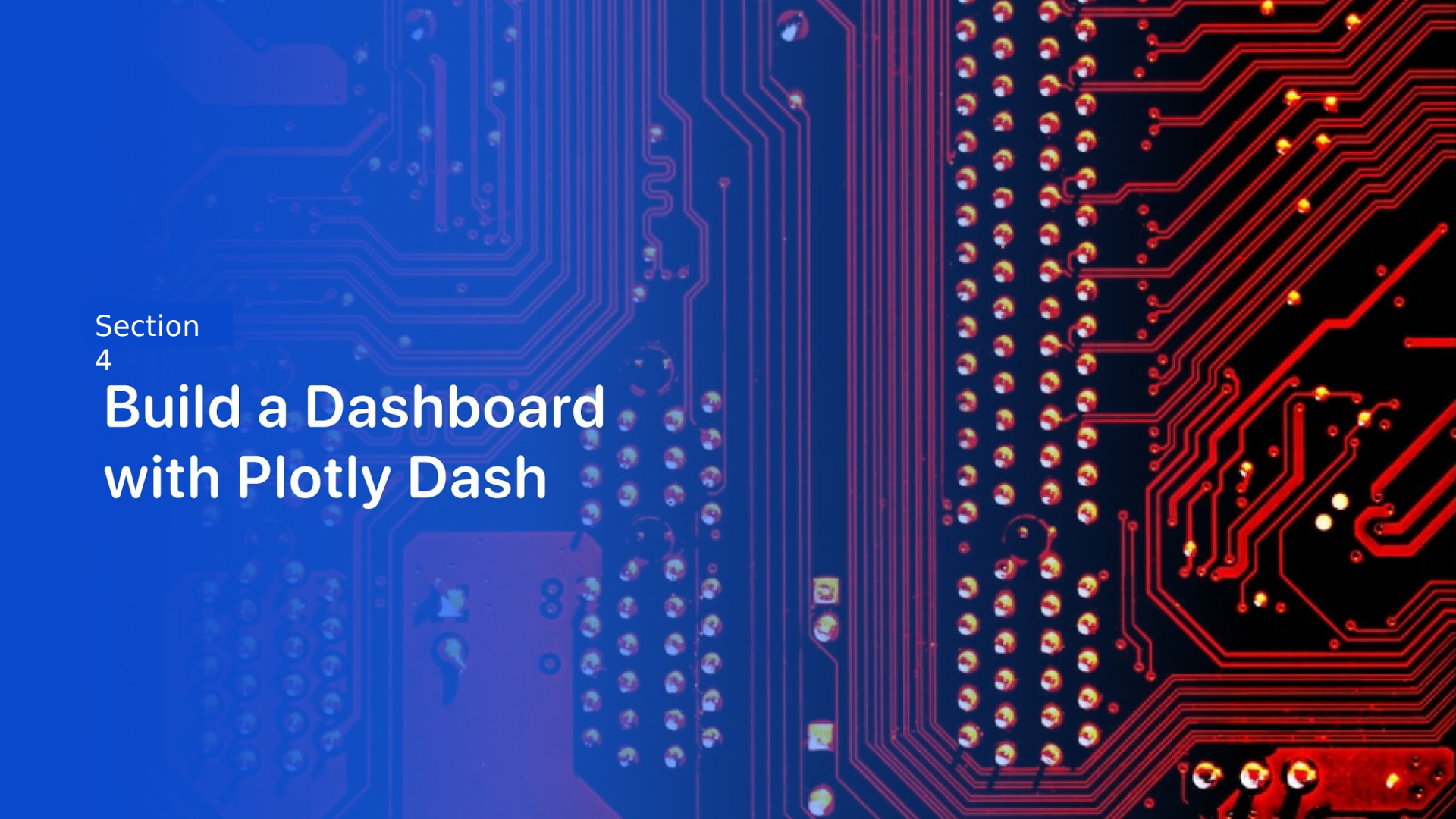
For example, only 4 out of 10 flights were successful for launches from Vandenberg SFB

Analysis of Surroundings with Folium



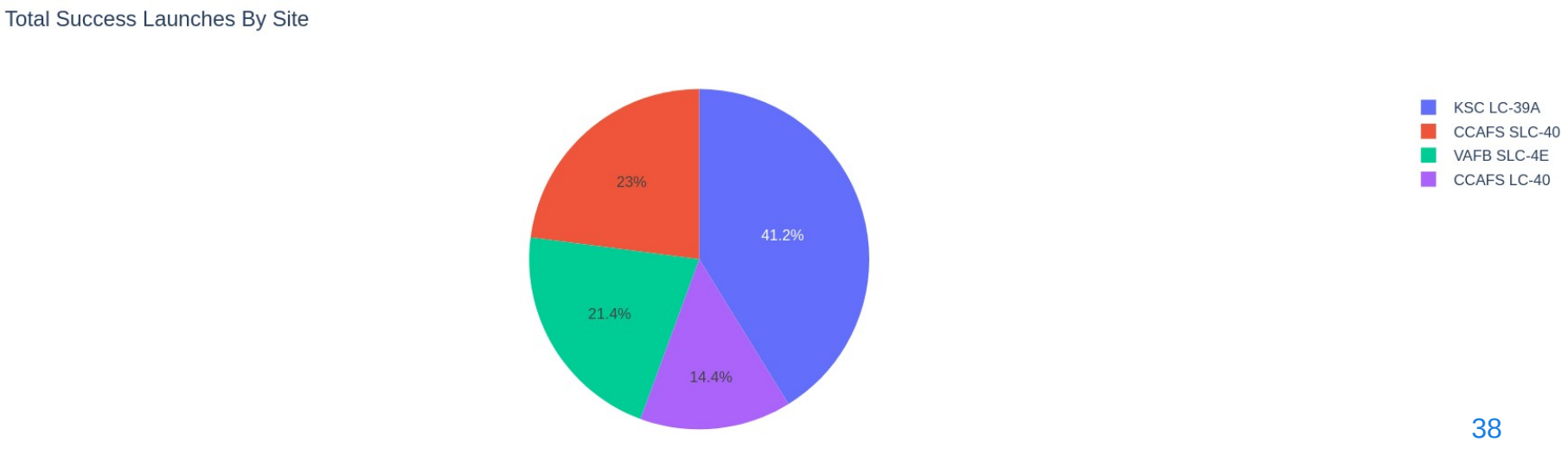


The closest road is located 0.69 km from SLC-40, whereas the closes rail is 1.33 km away. The closest city, Cape Canaveral, is 18.66 km away.


Section
4
Build a Dashboard
with Plotly Dash

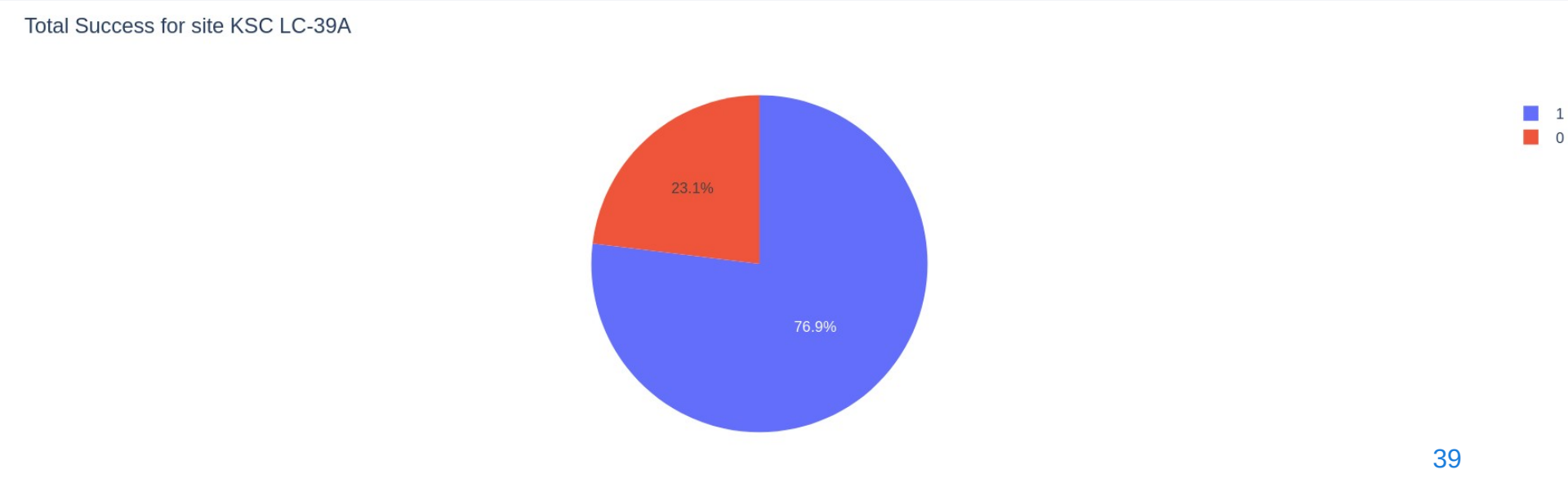
Dashboard Application - Pie Chart

Below is the resulting pie chart when the All Sites are selected in the dropdown menu



Dashboard Application - Pie Chart

The below pie chart shows the success rate for KSC LC-39A launch site, where 76.9% of launches were successful.



Dashboard Application - Scatter plot

The scatter plot shows the flights and their outcome for a given payload (top 0-10000kg, bottom 3000-6000kg).

The dots are colour coded to each of the booster versions.

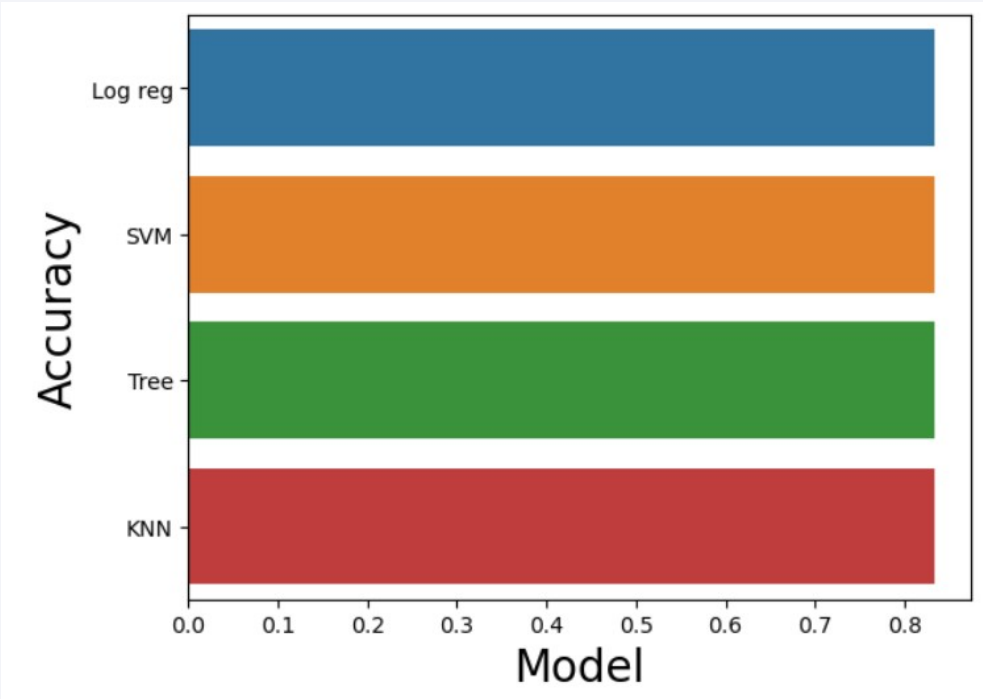
Section
5

Predictive Analysis (Classification)

Classification Accuracy

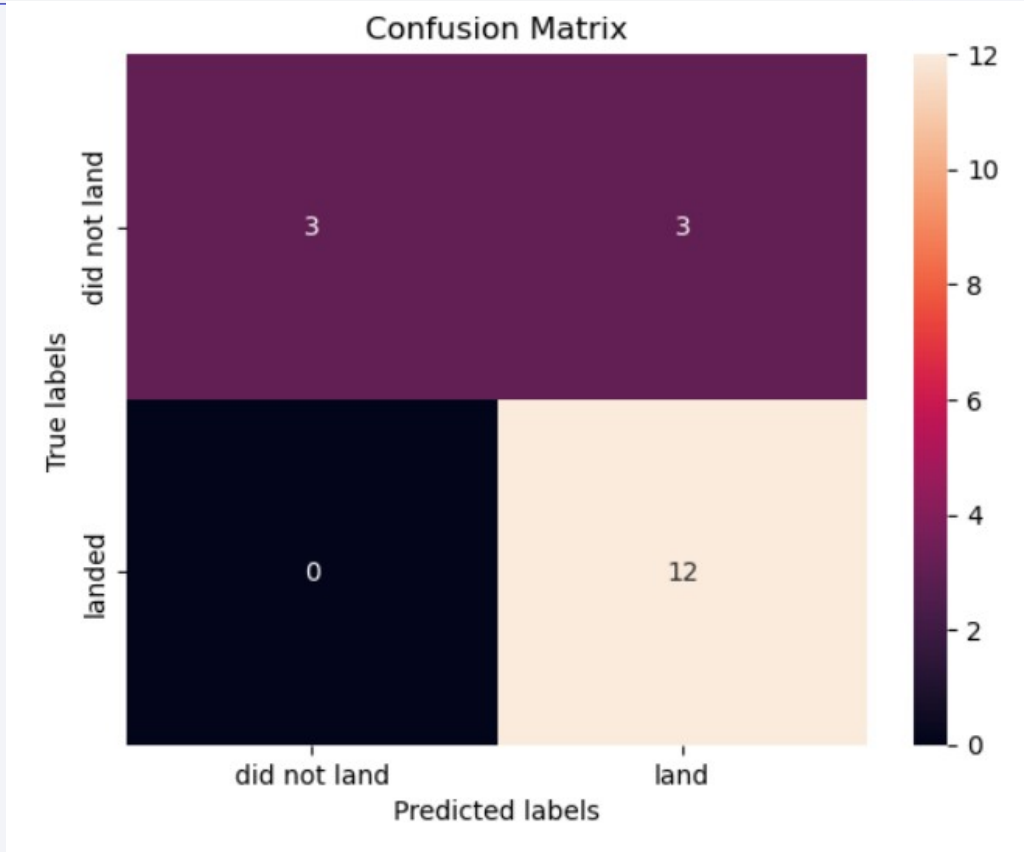
The bar char shows the accuracy for each classification model: Logistic Regression, Support Vector Machine, Decision Tree and K-nearest neighbour.

They all show the same accuracy (0.833) probably due to the relatively small amount of data, they were train and tested with (only 90 records)



Confusion Matrix

In agreement with the previously mentioned, all four models display the same confusion matrix, where three false positives are reported.



Conclusions

Due to the similarity of the results in the four models, when a new prediction is needed it might be interesting to create the same prediction with different models to get a better estimation of the outcome.

Appendix

The below links are files needed in case the notebooks or dash application need to be re run

https://github.com/francostuer/ibm_capstone_project/blob/main/spacex_launch_dash.csv
https://github.com/francostuer/ibm_capstone_project/blob/main/dataset_part_3.csv
https://github.com/francostuer/ibm_capstone_project/blob/main/spacex_launch_geo.csv
https://github.com/francostuer/ibm_capstone_project/blob/main/my_data1.db

Thank you!