Data Science Capstone project

<Zihaohan Sang>

<Sep26, 2021>

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection
  - Data wrangling
  - EDA with data extraction from SQL
  - Data visualization
  - Machine learning
- Summary of all results
  - Organized data
  - Visualization
  - Predictions and pattern analysis

Introduction

- Project background and context

In this capstone, we will predict if the Falcon 9 first stage will land successfully. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

- Problems you want to find answers

Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

Methodology

- Data collection methodology:
  - Describe how data were collected
- Perform data wrangling
  - Describe how data were processed
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Methodology

Data collection

- Request to the SpaceX API

  1. We extracted information using identification numbers in the launch data from SpaceX API with the following URL 'https://api.spacexdata.com/v4/launches/past'. Followed record were extracted: rocket, Launchpad, payload, and cores

- Clean the requested data using Pandas
  1. take a subset of our dataframe keeping only the features we want and the flight number, and date_utc;
  2. remove rows with multiple cores;
  3. remove the Falcon 1 launches keeping only the Falcon 9 launches
  4. deal with missing values

Data collection – SpaceX API

GitHub URL of the completed SpaceX API calls notebook (https://github.com/zihhSang/IBM/blob/main/week1.ipynb)

SpaceX REST API → return as .json → Normalized data into flat data file

simplified flow chart

1 .Getting Response from API

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

2. Converting Response to a .json file

```
response = requests.get(static_json_url).json()
data = pd.json_normalize(response)
```

3. Apply custom functions to clean data

```
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

```
getBoosterVersion(data)
```

4. Assign list to dictionary then 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}
```

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

5. Filter dataframe and export to flat file (.csv)

```
data_falcon9 = df.loc[df['BoosterVersion']!="Falcon 1"]
```

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

8

Data collection Web scraping

Html response → Extract using beautifulsoup → Clean data using pandas

GitHub URL of the completed web scraping noteboo (https://github.com/zihhSang/IBM/blob/main/week1_2.ipynb)

simplified flow chart

1 .Getting Response from HTML

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

2. Creating BeautifulSoup Object

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

3. Finding tables

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

4. Getting column names

```
column_names = []
temp = soup.find_all('th')
for x in range(len(temp)):
    try:
     name = extract_column_from_header(temp[x])
     if (name is not None and len(name) > 0):
        column_names.append(name)
    except:
     pass
```

5. Creation of dictionary

```
launch_dict= dict.fromkeys(column_names)

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


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'] = []
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

6. Appending data to keys (refer) to notebook block 12

```
In [12]: extracted_row = 0
         #Extract each table
         for table_number,table in enumerate(
            # get table row
             for rows in table.find_all("tr")
                 #check to see if first table
```

7. Converting dictionary to dataframe

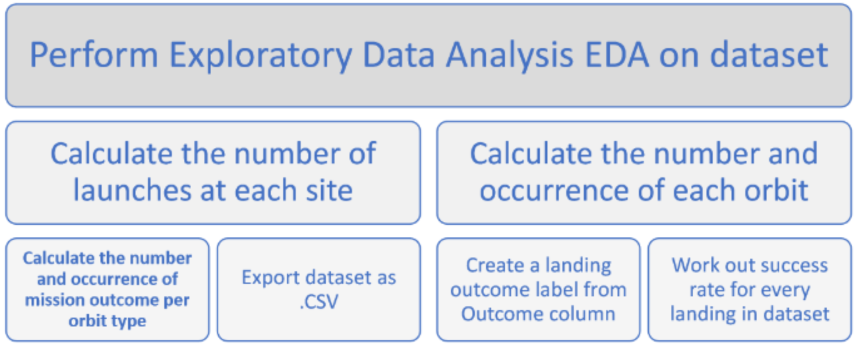
```
df = pd.DataFrame.from_dict(launch_dict)
```

8. Dataframe to .CSV

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Data wrangling

- Introduction: In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident; for example, True Ocean means the mission outcome was successfully landed to a specific region of the ocean while False Ocean means the mission outcome was unsuccessfully landed to a specific region of the ocean. True RTLS means the mission outcome was successfully landed to a ground pad False RTLS means the mission outcome was unsuccessfully landed to a ground pad.True ASDS means the mission outcome was successfully landed on a drone ship False ASDS means the mission outcome was unsuccessfully landed on a drone ship.
- Process:



- GitHub URL of your completed data wrangling related notebooks (https://github.com/zihhSang/IBM/blob/main/week1_3.ipynb)

EDA with data visualization

- Summarize what charts were plotted and why used those charts
  - 1. scatter plots
    - Flight number vs. payload mass
    - Flight number vs. launch site
    - Payload vs. launch site
    - Orbit vs. flight number
    - Payload vs. orbit type

  2. Bar graph:
    - Mean vs. orbit

- GitHub URL of your completed EDA with data visualization notebook (https://github.com/zihhSang/IBM/blob/main/week2_2.ipynb)

EDA with SQL

- Summarize performed SQL queries using bullet points
  - Display the names of unique launched sites
  - Display launch sites begin with 'KSC'
  - Display the total payload mass launched by CRS
  - Display the mean payload mass of booster version F9 V11
  - List of names of boosters with payload mass between 4000 and 6000
  - List the number of successful and failure missions
  - The name of booster_version carried the maximum payload mass
  - Rank the count of successful missions

- GitHub URL of your completed EDA with SQL notebook(https://github.com/zihhSang/IBM/blob/main/week2_1.ipynb)

Build an interactive map with Folium

- Step:
  - 1. added latitude and longitude coordinates at each launch site and added a circle marker with a label with their names
  - 2. assigned the dataframe given the outcome (fail or success) as 0 and 1, and colors them with green and red markers
  - 3. added the distance from sites to markers

- These objects were added to show the distance to railways, highways, coastline and cities

- GitHub URL of your completed interactive map with Folium map (https://github.com/zihhSang/IBM/blob/main/week3_1.ipynb)

Build a Dashboard with Plotly Dash

- Pie chart showing the total launches by sites
  - Displaying the proportion of classes
  - Size of circle represent the total sample size
- Scatter plot showing the relationship between outcome and payload mass for boosters
  - Show linear relationship
  - Show the range of values
- GitHub URL of your completed Plotly Dash lab (https://github.com/zihhSang/IBM/blob/main/Dashboard%20SpaceX%20Dataset.ipynb)

Predictive analysis (Classification)

- Building model
  - Load and transform data
  - Split data into training and test sets
  - Decide the algorithms of machine learning
  - Set parameters and fit the model

- Evaluating model
  - Compute accuracy for model
  - Tune parameters
  - Improving model if necessary

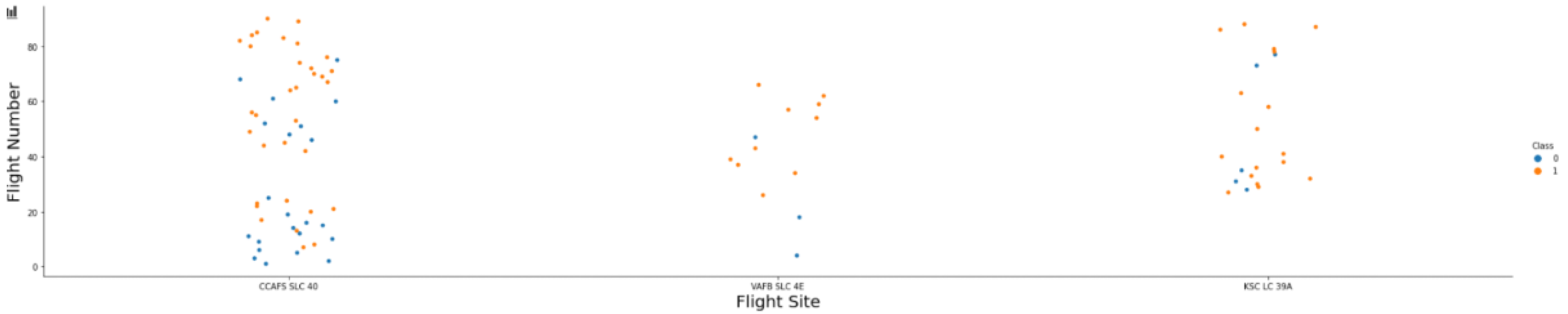
- GitHub URL of your completed predictive analysis lab (https://github.com/zihhSang/IBM/blob/main/week4.ipynb)

Results

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

EDA with Visualization

Flight Number vs. Launch Site

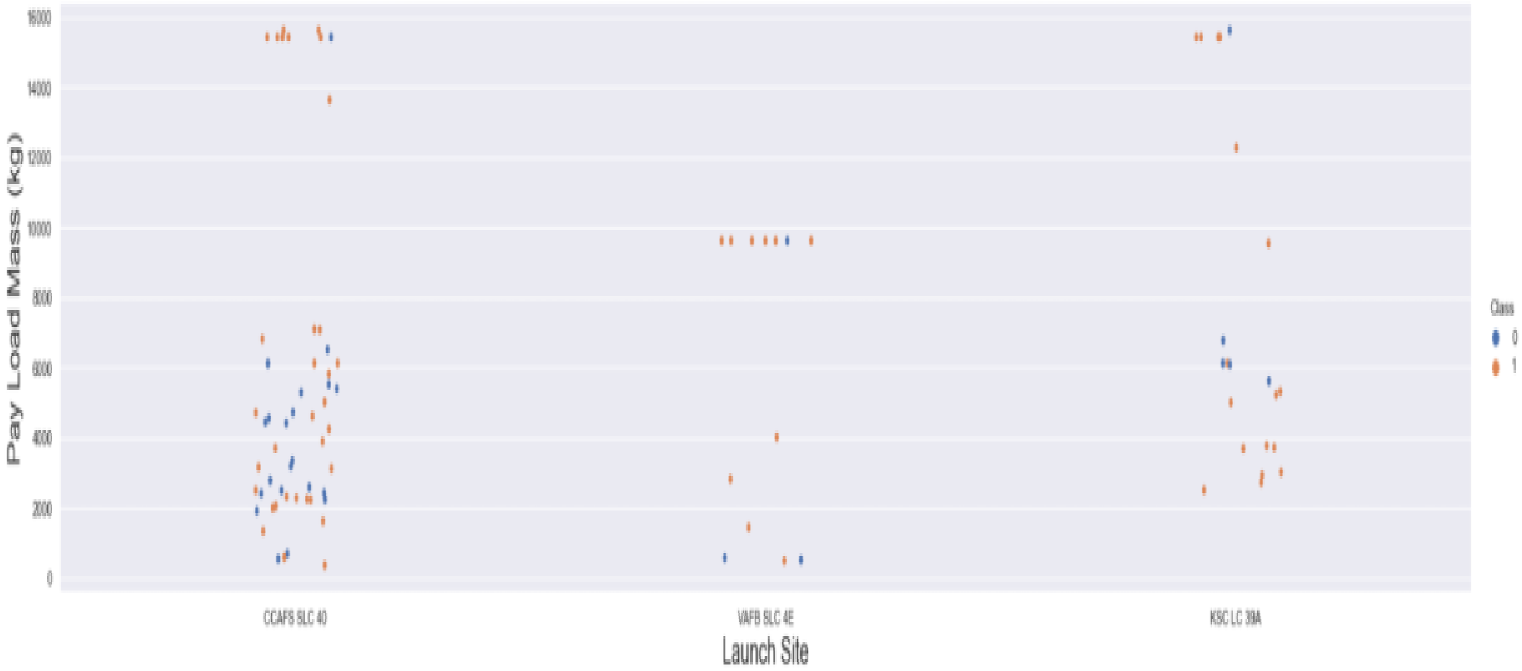


The more amount of flights at a launch site the greater the success rate at a launch site.

Payload vs. Launch Site

Show a scatter plot of Payload vs. Launch Site

Show the screenshot of the scatter plot with explanations

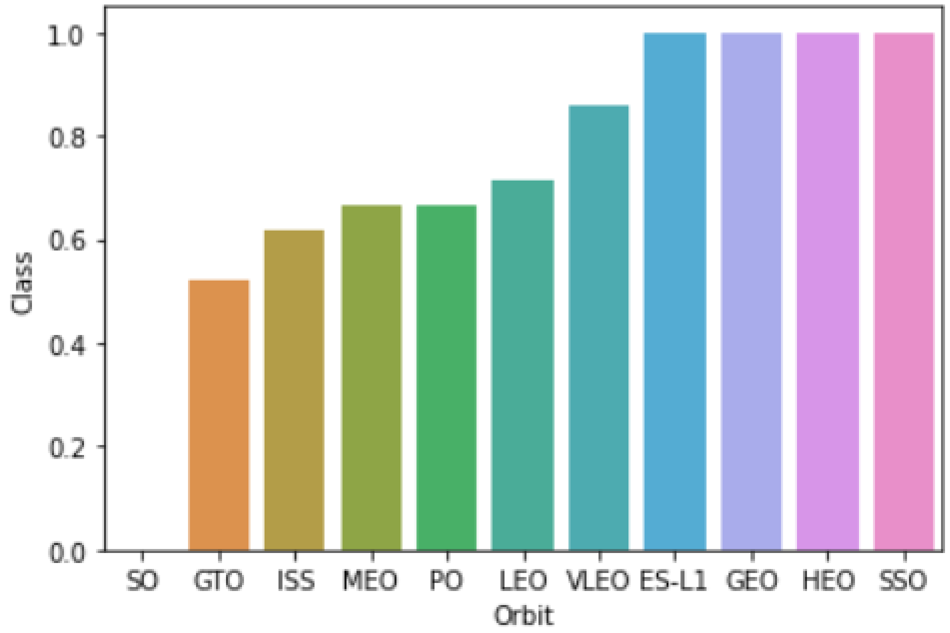


Greater the payload mass, higher the success rate for the mission, although there is no quite clear pattern whether the launch site id dependent on payload mass for a success launch.

Success rate vs. Orbit type

Show a barchart for the success rate of each orbit type

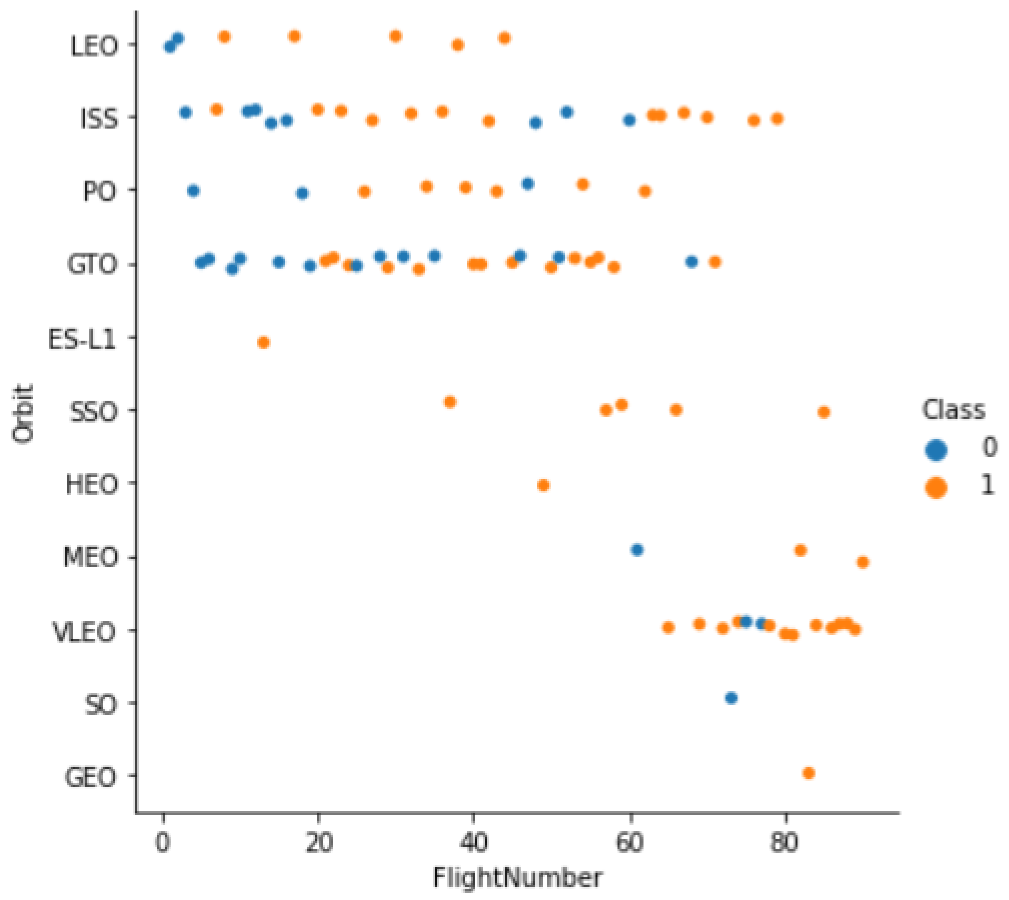
Orbit VLEO, ES-L1, GEO, HEO and SSO have the most performance.



Flight Number vs. Orbit type

Show a scatter point of Flight number vs. Orbit type

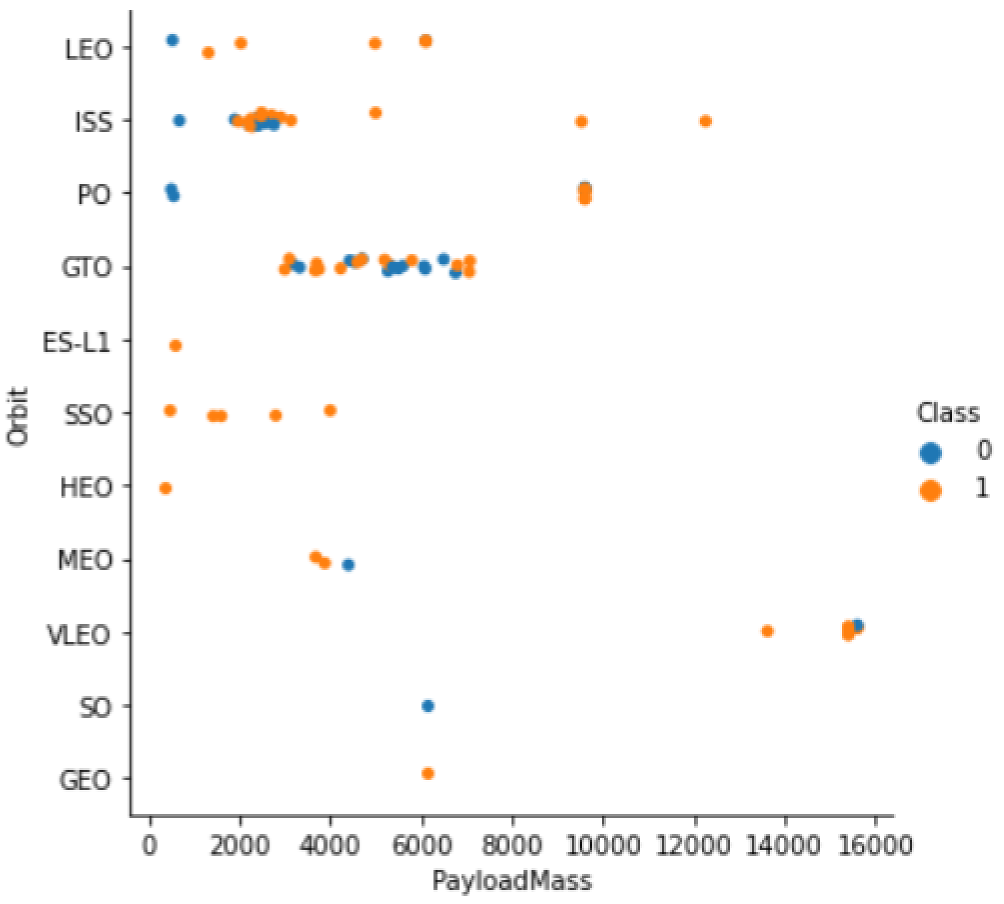
Orbit SSO showed all success records across flight numbers. LEO have high success in high flight number, and high fails for lower numbers. Instead, GTO shows no clear pattern of the relationship between output and flightnumber.



Payload vs. Orbit type

Show a scatter point of payload vs. orbit type

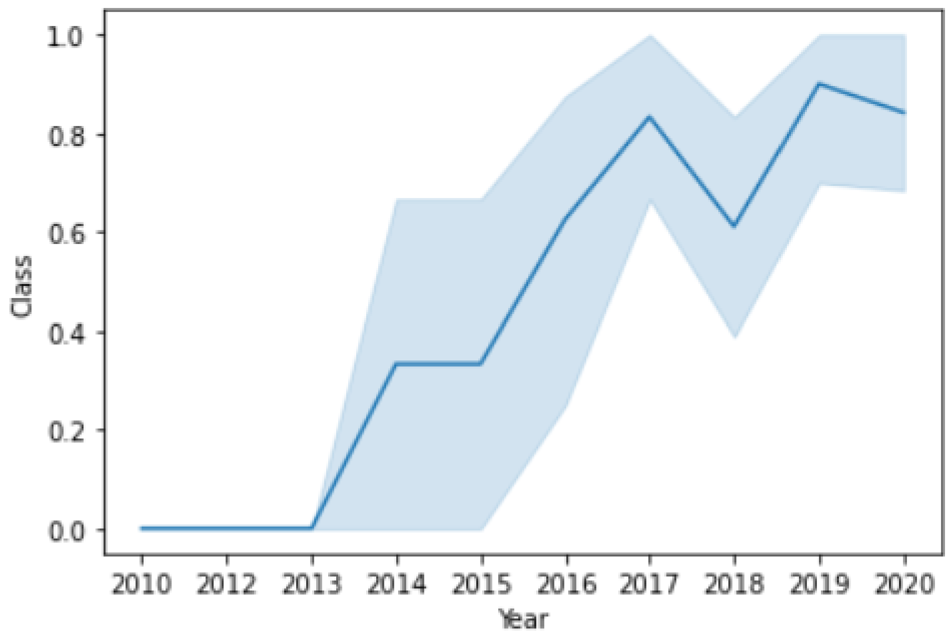
Payload mass tend to have no impacts on GTO. However, ISS show higher success rate for higher payload mass.



Launch success yearly trend

Show a line chart of yearly average success rate

The success rate keep increasing from 2013.



EDA with SQL

All launch site names

SQL QUERY

```
select DISTINCT Launch_Site
from tblSpaceX
```

Unique Launch Sites
CCAFS LC-40
CCAFS SLC-40
CCAFS SLC-40
KSC LC-39A
VAFB SLC-4E

QUERY EXPLAINATION

Using the word ***DISTINCT*** in the query means that it will only show Unique values in the ***Launch_Site*** column from ***tblSpaceX***

Launch site names begin with `CCA`

- SELECT TOP 5 FROM tblSpaceX WHERE Launch_Site LIKE ‘CCA%’

```
        Date Time..UTC. Booster_Version Launch_Site                                                     Payload PAYLOAD_MASS__KG_     Orbit
1 04-06-2010   18:45:00  F9 v1.0  B0003 CCAFS LC-40                              Dragon Spacecraft Qualification Unit                 0       LEO
2 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)
3 22-05-2012   07:44:00  F9 v1.0  B0005 CCAFS LC-40                                             Dragon demo flight C2               525 LEO (ISS)
4 08-10-2012   00:35:00  F9 v1.0  B0006 CCAFS LC-40                                                      SpaceX CRS-1               500 LEO (ISS)
5 01-03-2013   15:10:00  F9 v1.0  B0007 CCAFS LC-40                                                      SpaceX CRS-2               677 LEO (ISS)
         Customer Mission_Outcome     Landing._Outcome
1          SpaceX         Success Failure (parachute)
2 NASA (COTS) NRO         Success Failure (parachute)
3     NASA (COTS)         Success          No attempt
4      NASA (CRS)         Success          No attempt
5      NASA (CRS)         Success          No attempt
```

- Using TOP 5 for first 5 rows of records; LIKE ‘CCA%’ is for filtering character strings starts with ‘CCA’.

Total payload mass

- Select SUM(PAYLOAD_MASS_KG_) from tblSpaceX WHERE CUSTOMER = ‘NASA(CRS)’

	Total Payload Mass
0	45596

QUERY EXPLAINATION

Using the function ***SUM*** summates the total in the column ***PAYLOAD_MASS_KG_***

The ***WHERE*** clause filters the dataset to only perform calculations on ***Customer NASA (CRS)***

Average payload mass by F9 v1.1

```
select AVG(PAYLOAD_MASS_KG_) AveragePayloadMass from tblSpaceX
where Booster_Version = 'F9 v1.1'
```

	Average Payload Mass
0	2928

QUERY EXPLAINATION

Using the function ***AVG*** works out the average in the column ***PAYLOAD_MASS_KG_***

The ***WHERE*** clause filters the dataset to only perform calculations on ***Booster_version F9 v1.1***

First successful ground landing date

select MIN(Date) SLO from tblSpaceX where Landing_Outcome = "Success (drone ship)"

	Date which first Successful landing outcome in drone ship was acheived.
0	06-05-2016

QUERY EXPLAINATION

Using the function ***MIN*** works out the minimum date in the column ***Date***

The ***WHERE*** clause filters the dataset to only perform calculations on ***Landing_Outcome Success (drone ship)***

Successful drone ship landing with payload between 4000 and 6000

select Booster_Version from tblSpaceX where Landing_Outcome = 'Success (ground pad)'
AND Payload_MASS_KG_ > 4000 AND Payload_MASS_KG_ < 6000

	Date which first Successful landing outcome in drone ship was acheived.
0	F9 FT B1032.1
1	F9 B4 B1040.1
2	F9 B4 B1043.1

QUERY EXPLAINATION

Selecting only ***Booster_Version***

The ***WHERE*** clause filters the dataset to ***Landing_Outcome = Success (drone ship)***

The ***AND*** clause specifies additional filter conditions
Payload_MASS_KG_ > 4000 AND ***Payload_MASS_KG_ < 6000***

Total number of successful and failure mission outcomes

- SELECT (select(count(mission_outcome) from tblSpaceX WHERE Mission_Outcome LIKE ‘%success%’) as successful_missing_outcomes,

(Select count(mission_outcome) from tblSpaceX where Mission_outcome like ‘%Failure%’ as failture_mission_countcomes

	Successful_Mission_Outcomes	Failure_Mission_Outcomes
0	100	1

QUERY EXPLAINATION

a much harder query I must say, we used subqueries here to produce the results. The ***LIKE ‘%foo%’*** wildcard shows that in the record the ***foo*** phrase is in any part of the string in the records for example.

Boosters carried maximum payload

- SELECT DISTINCT booster_version, max(payload_mass_kg_) as Maximum_payload_mass

  FROM tblSpaceX GROUP BY booster_version ORDER BY Maximum_payload_mass DESC

	Booster_Version	Maximum Payload Mass
0	F9 B5 B1048.4	15600
1	F9 B5 B1048.5	15600
2	F9 B5 B1049.4	15600
3	F9 B5 B1049.5	15600
4	F9 B5 B1049.7	15600
...	...	...
92	F9 v1.1 B1003	500
93	F9 FT B1038.1	475
94	F9 B4 B1045.1	362
95	F9 v1.0 B0003	0
96	F9 v1.0 B0004	0

2015 launch records

```
SELECT DATENAME(month, DATEADD(month,
MONTH(CONVERT(date, Date, 105)), 0) - 1) AS Month,
Booster_Version, Launch_Site, Landing_Outcome
FROM     tblSpaceX
WHERE  (Landing_Outcome LIKE N'%Success%') AND
(YEAR(CONVERT(date, Date, 105)) = '2017')
```

Month	Booster_Version	Launch_Site	Landing_Outcome
January	F9 FT B1029.1	VAFB SLC-4E	Success (drone ship)
February	F9 FT B1031.1	KSC LC-39A	Success (ground pad)
March	F9 FT B1021.2	KSC LC-39A	Success (drone ship)
May	F9 FT B1032.1	KSC LC-39A	Success (ground pad)
June	F9 FT B1035.1	KSC LC-39A	Success (ground pad)
June	F9 FT B1029.2	KSC LC-39A	Success (drone ship)
June	F9 FT B1036.1	VAFB SLC-4E	Success (drone ship)
August	F9 B4 B1039.1	KSC LC-39A	Success (ground pad)
August	F9 FT B1038.1	VAFB SLC-4E	Success (drone ship)
eptember	F9 B4 B1040.1	KSC LC-39A	Success (ground pad)
October	F9 B4 B1041.1	VAFB SLC-4E	Success (drone ship)
October	F9 FT B1031.2	KSC LC-39A	Success (drone ship)
October	F9 B4 B1042.1	KSC LC-39A	Success (drone ship)
December	F9 FT B1035.2	CCAFS SLC-40	Success (ground pad)

Rank success count between 2010-06-04 and 2017-03-20

- Select count(Landing_Outcome) FROM tblSpaceX

WHERE (Landing_Outcome LIKE ‘%Success%’) AND (Date > ’2010-06-04’) AND (Date < ‘2017-03-20’)

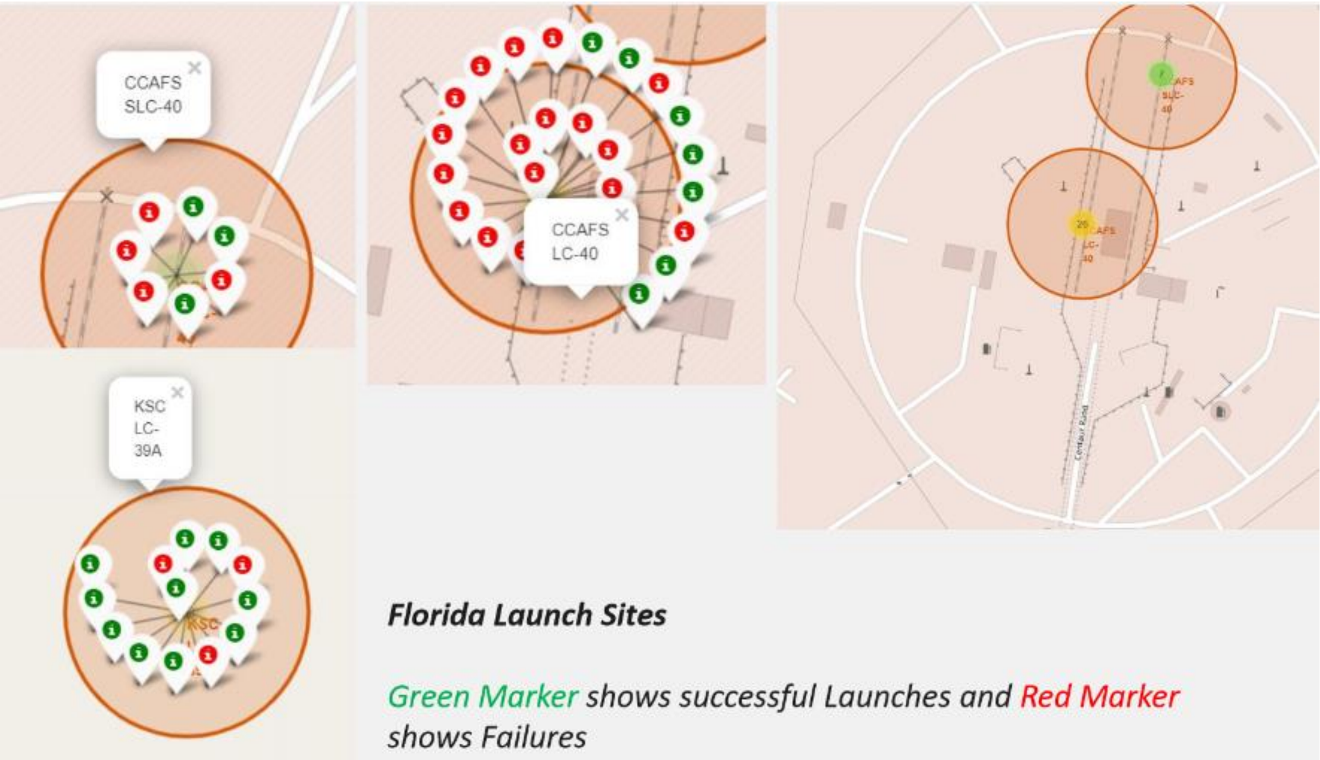
Between 2010-06-04 and 2017-03-20
34

Count is to records the number of records meet the requirements listed by WHERE.

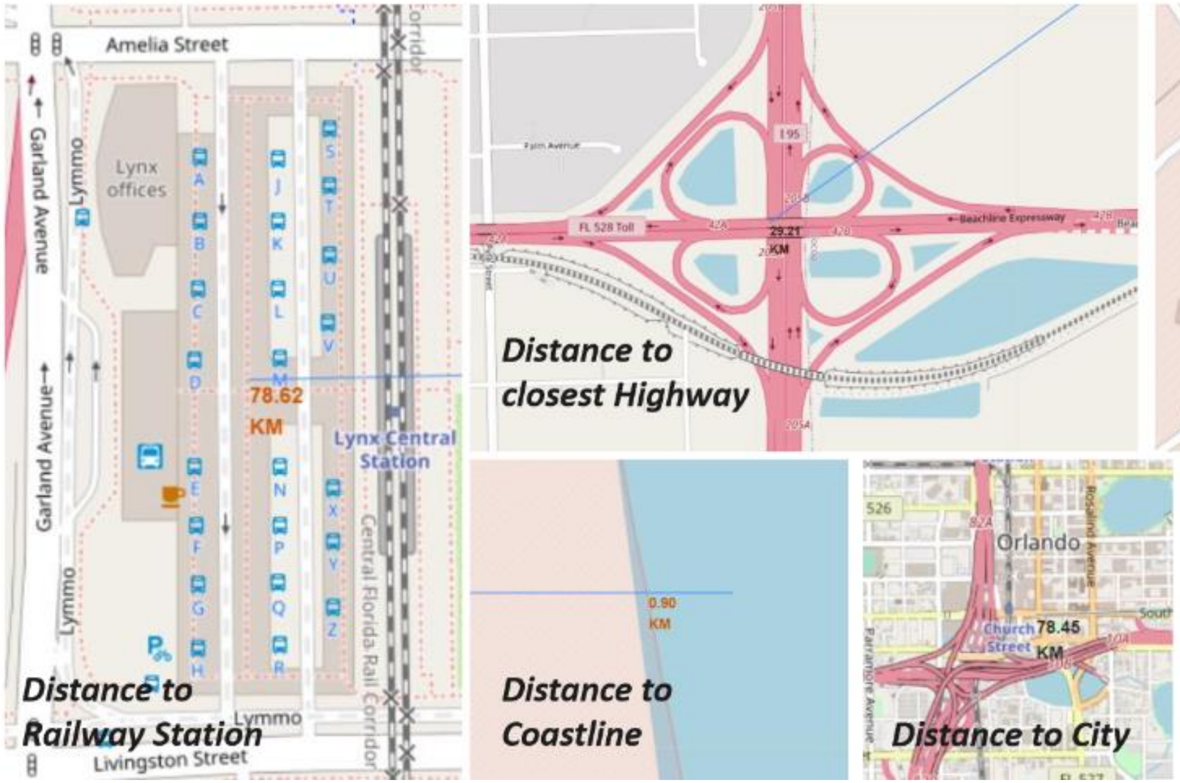
Interactive map with Folium

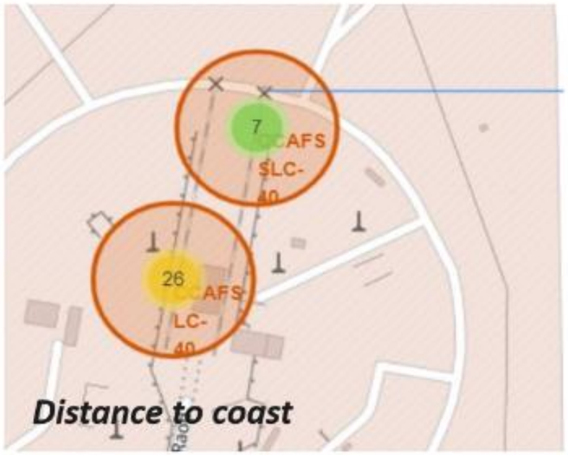
Launch sites locations

Launch sites labelled with outcomes



Distance between launch sites and landmarks



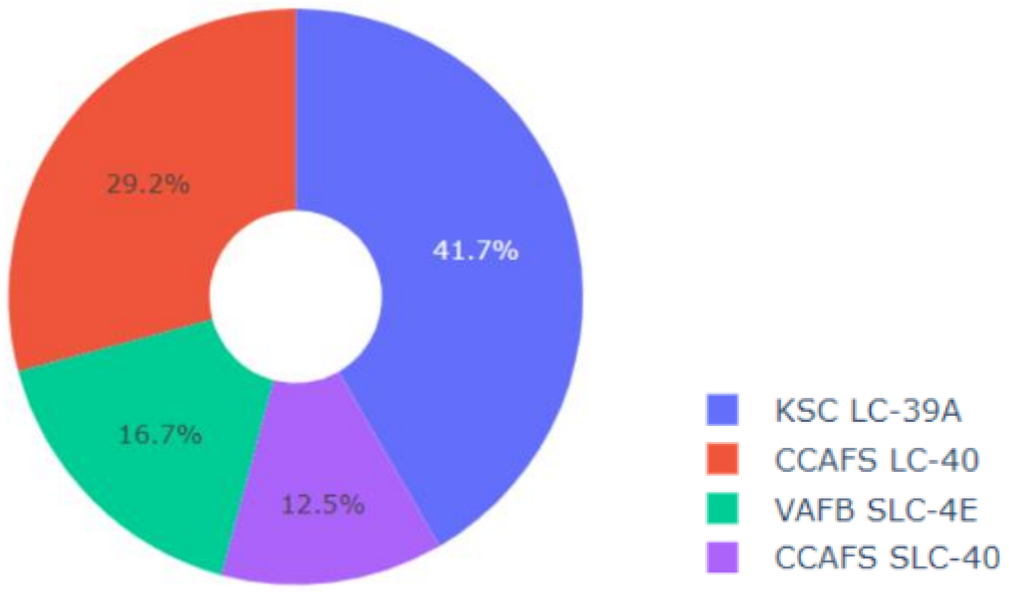


- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Build a Dashboard with Plotly Dash

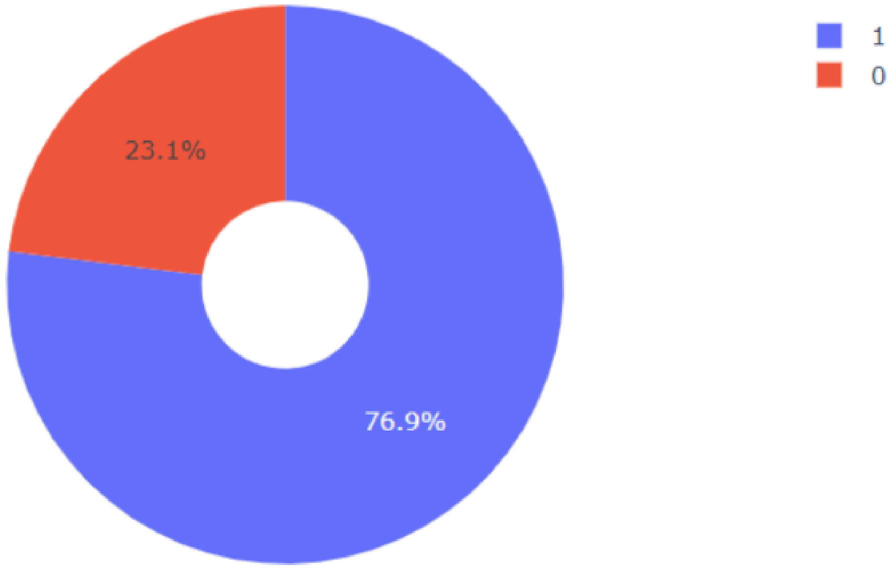
Proportion of success launches by sites

KSC LC-39A had the highest successful proportion from all sites.



launch site with highest launch success ratio

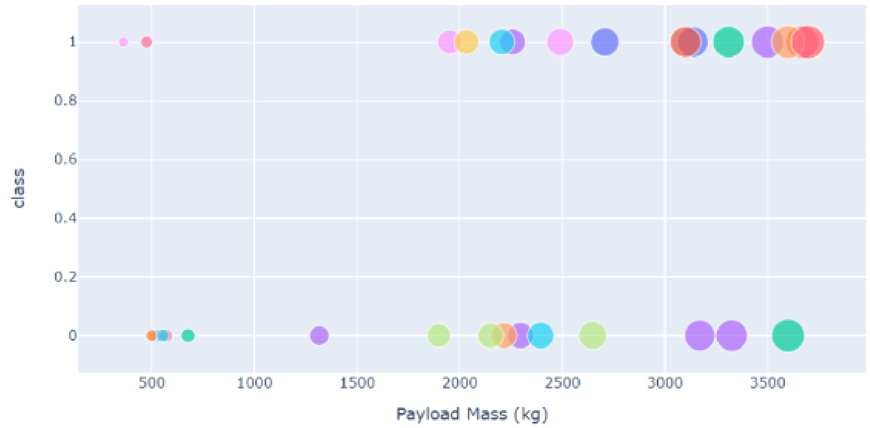
- Successful rate is up to 76.9%



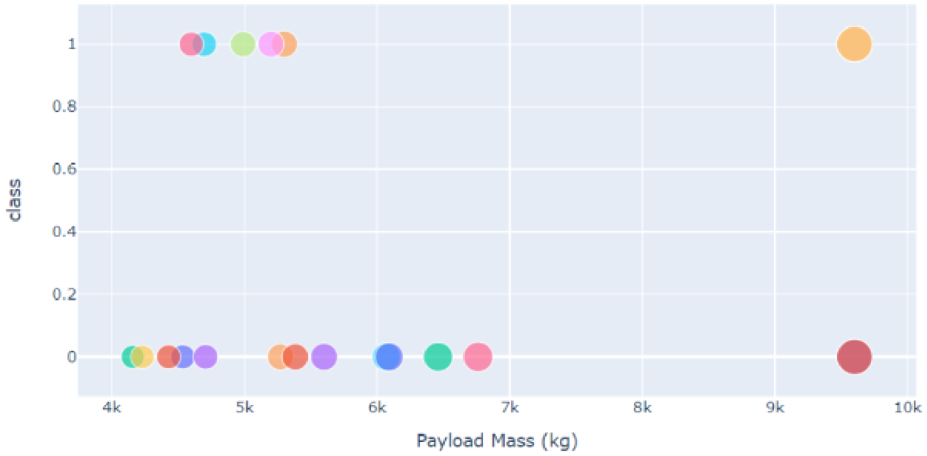
Payload vs. Launch Outcome

- Launches with medium payload tend to have higher successful rate

Low Weighted Payload 0kg – 4000kg



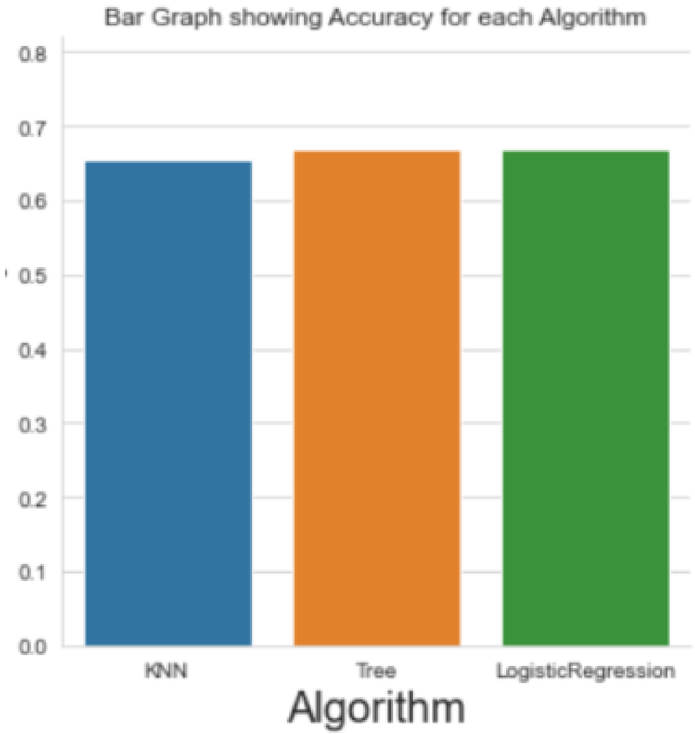
Heavy Weighted Payload 4000kg – 10000kg



Predictive analysis (Classification)

Classification Accuracy

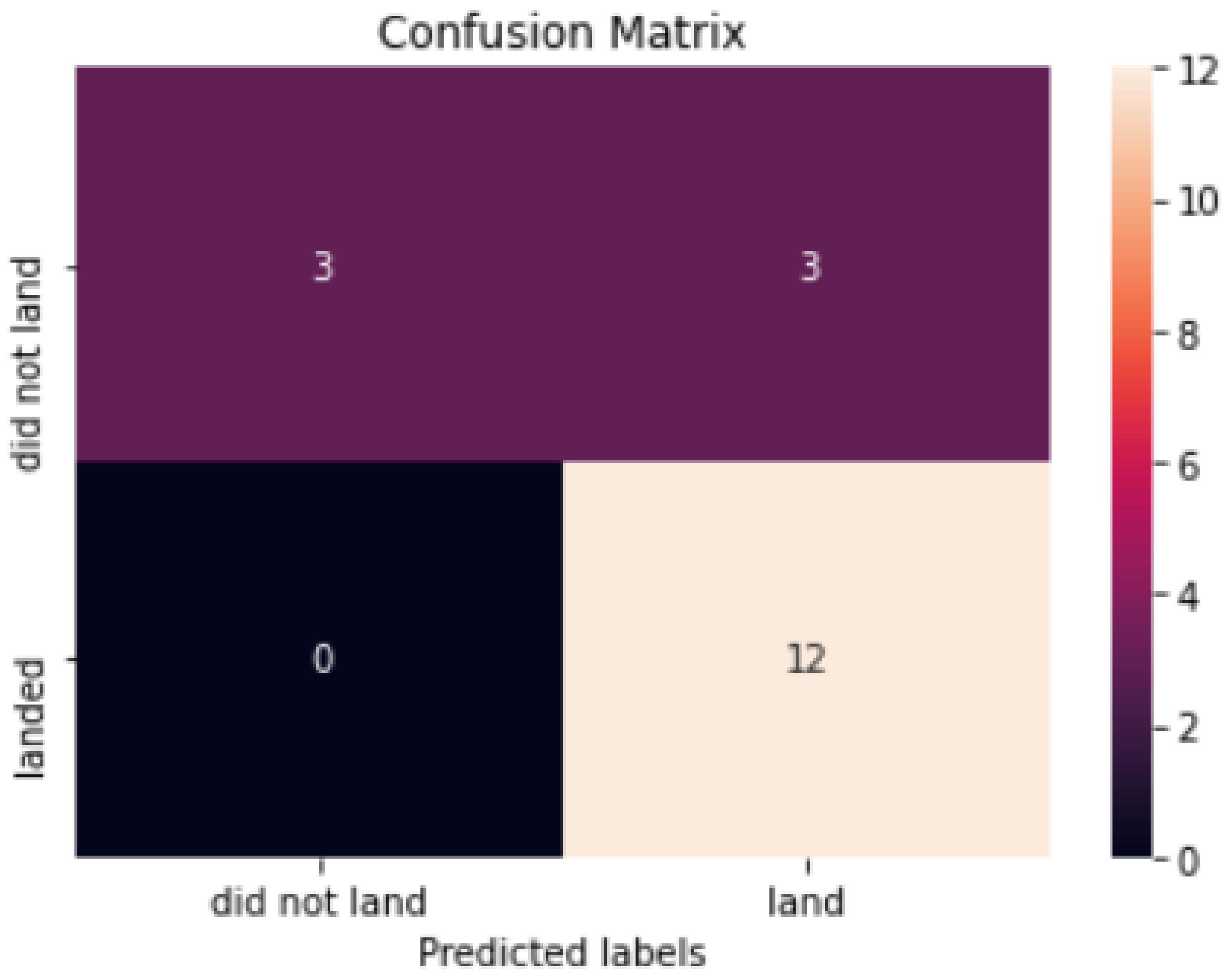
	Accuracy	Algorithm
0	0.653571	KNN
1	0.667857	Tree
2	0.667857	LogisticRegression



In general, the models show similar accuracy, but Tree algorithm is slightly better.

Confusion Matrix

Show the confusion matrix of the best performing model with explanation



CONCLUSION

- Machine learning models can make reasonable predictions for the launch outcomes
- Many factors influence the successful rates
- Orbits GEO,HEO, SSO and ES-L1 have the highest successful rates
- The overall successful rates increase through years

APPENDIX

- Include all python notebooks in the Github folder.