IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Paul Dai
July 20, 2022

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection: SpaceX API and Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis (EDA): SQL and Data Visualization
  - Interactive Visual Analytics: Folium and Dashboard
  - Predictive Analysis: Classification
- Summary of all results
  - EDA result
  - Screenshot of folium and dashboard
  - Predictive Analytics result

Introduction

- Project background and context
  - 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. The goal of this project is to create a data pipeline and machine learning models to predict whether the first stage of SpaceX would land successfully or not.
- Problems you want to find answers
  1. What features will impact on the landing success rate of first stage?
  2. What the landing success rate of first stage is predicted in our model ?
  3. What the accuracy of our model?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Using SpaceX API and web scraping from Wikipedia to collect those raw data.
- Perform data wrangling
  - We deal with the missing values and one-hot encoding is applied to our features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - We standardize our data and split it in to train and test dataset, and use GridSearchCVto find the best classification model to predict the landing success rate.

Data Collection

- We collect our raw data with a series of processes:
  1. We request the data by SpaceX API and transform the json file into data frame.
  2. Second step, we do some necessary ETL job to clean our data.
  3. After that, we use BeautifulSoup to scrape Wikipedia for SpaceX launch records and convert those records into data frame.

Data Collection – SpaceX API

1. Collecting the raw data by SpaceX REST calls

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

2. Filter the data only with 'Falcon 9' launch

```
data_falcon9=launch_df[launch_df['BoosterVersion']=='Falcon 9']
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9.head()
```

3. Dealing with Missing Values

```
# Calculate the mean value of PayloadMass column
data_falcon9['PayloadMass']=data_falcon9['PayloadMass'].replace(np.nan, data_falcon9['PayloadMass'].mean())
print(df.isnull().sum())
data_falcon9
```

GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/1_Collecting%20the%20data.ipynb

Data Collection – SpaceX API

The information of the launch data

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 90 entries, 4 to 93
Data columns (total 17 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   FlightNumber    90 non-null     int64  
 1   Date            90 non-null     object 
 2   BoosterVersion  90 non-null     object 
 3   PayloadMass     90 non-null     float64
 4   Orbit           90 non-null     object 
 5   LaunchSite      90 non-null     object 
 6   Outcome         90 non-null     object 
 7   Flights         90 non-null     int64  
```

```
 8   GridFins        90 non-null     bool   
 9   Reused          90 non-null     bool   
 10  Legs            90 non-null     bool   
 11  LandingPad      64 non-null     object 
 12  Block           90 non-null     float64
 13  ReusedCount     90 non-null     int64  
 14  Serial          90 non-null     object 
 15  Longitude       90 non-null     float64
 16  Latitude        90 non-null     float64
dtypes: bool(3), float64(4), int64(3), object(7)
memory usage: 10.8+ KB
```

The examples of the launch data

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
4	1	2010-06-04	Falcon 9	6123.547647	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003	-80.577366	28.561857
5	2	2012-05-22	Falcon 9	525.000000	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0005	-80.577366	28.561857
6	3	2013-03-01	Falcon 9	677.000000	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0007	-80.577366	28.561857
7	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None	1.0	0	B1003	-120.610829	34.632093
8	5	2013-12-03	Falcon 9	3170.000000	GTO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B1004	-80.577366	28.561857

Data Collection - Scraping

1. Request Wiki and get the launch table

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
r=requests.get(static_url).text
soup=BeautifulSoup(r)
html_tables=[]
html_tables=soup.find_all('table')
first_launch_table = html_tables[2]
```

2. Iterate columns and extract data

```
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        row=rows.find_all('td')
        if flag:
            extracted_row += 1
            # Flight Number value
            datatimelist=date_time(row[0])                  # Flight Number value
            print(flight_number)
            launch_dict['Flight No.'].append(flight_number)

            date = datatimelist[0]
```

3. Convert to data frame

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

GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/2_Web%20Scraping.ipynb

Data Collection – Scraping Result

The information of the scraping result

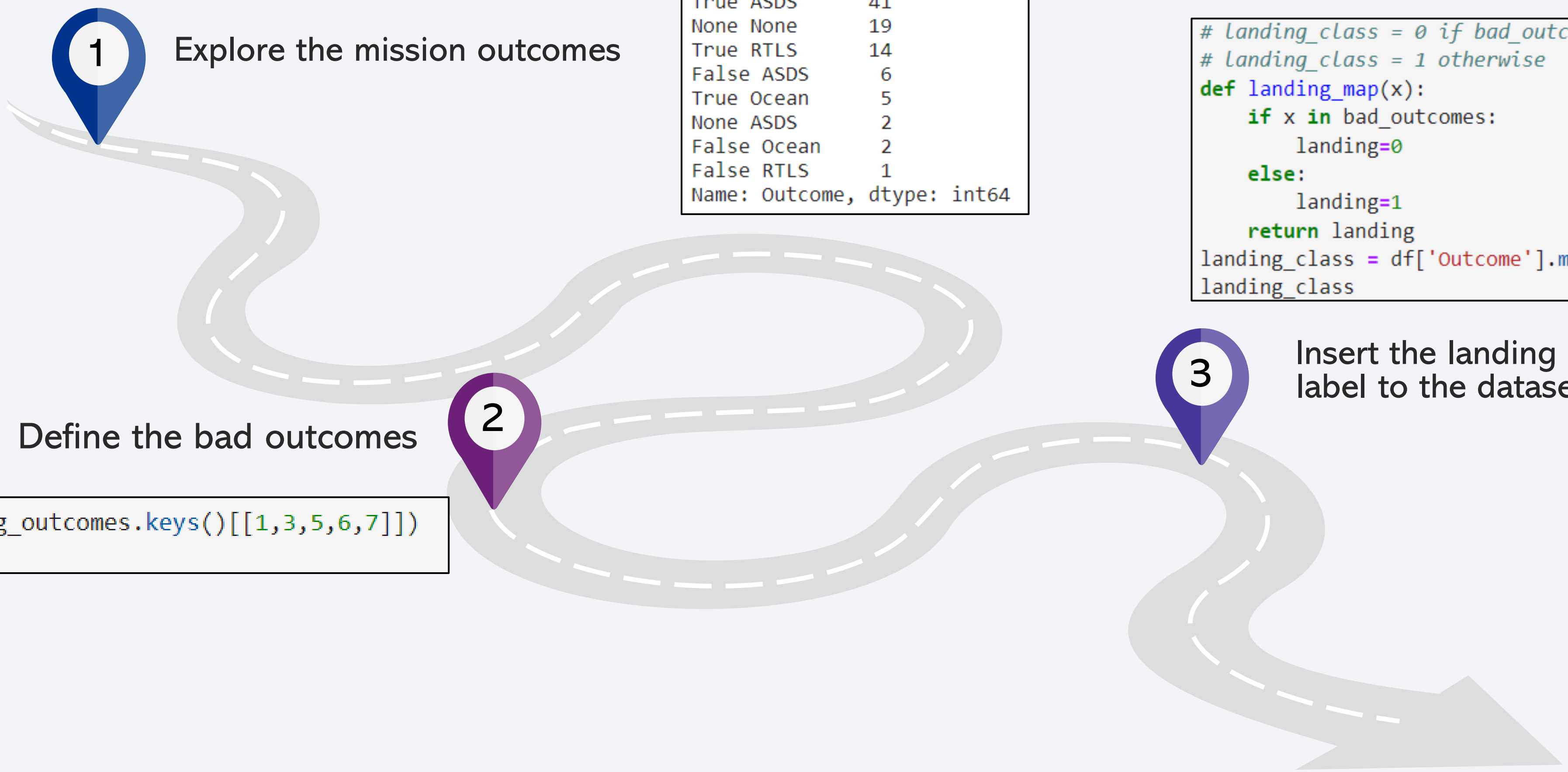
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 121 entries, 0 to 120
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Flight No.       121 non-null    object
 1   Launch site      121 non-null    object
 2   Payload          121 non-null    object
 3   Payload mass     121 non-null    object
 4   Orbit            121 non-null    object
 5   Customer         120 non-null    object
 6   Launch outcome   121 non-null    object
 7   Version Booster  121 non-null    object
 8   Booster landing  121 non-null    object
 9   Date             121 non-null    object
 10  Time             121 non-null    object
dtypes: object(11)
memory usage: 10.5+ KB
```

The examples of the scraping result

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	F9 v1.0B0003.1	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	NASA	Success	F9 v1.0B0005.1	No attempt\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success\n	F9 v1.0B0006.1	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success\n	F9 v1.0B0007.1	No attempt\n	1 March 2013	15:10

GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/2_Web%20Scraping.ipynb

Data Wrangling



1 Explore the mission 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
Name: Outcome, dtype: int64
```

2 Define the bad outcomes

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

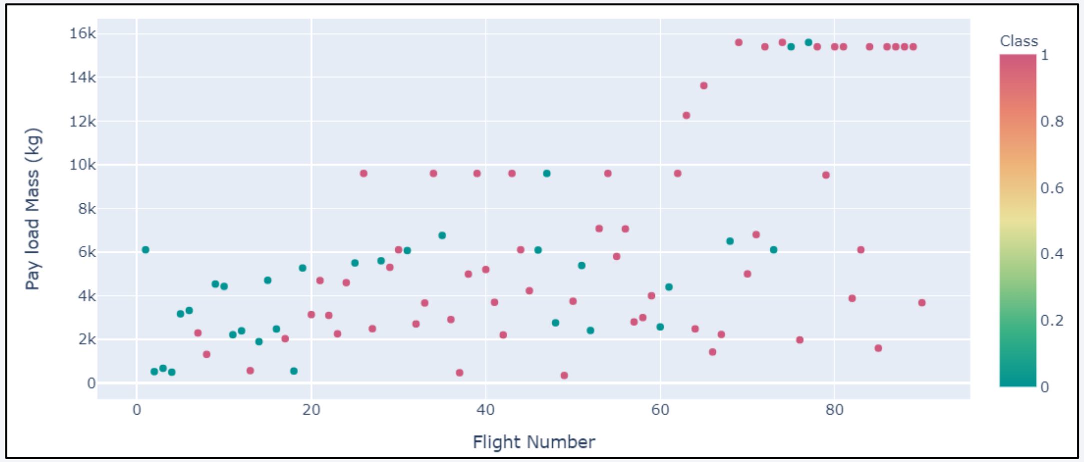
3 Insert the landing outcome label to the dataset

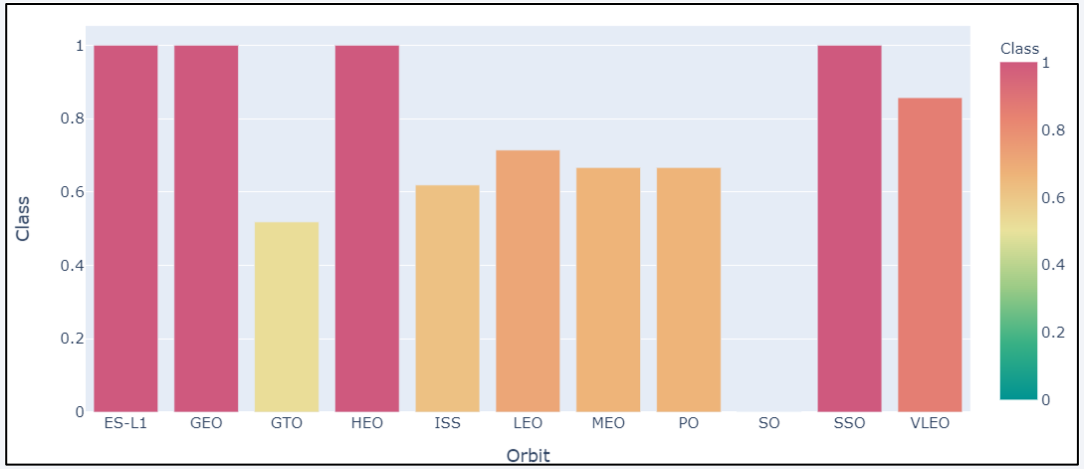
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
def landing_map(x):
    if x in bad_outcomes:
        landing=0
    else:
        landing=1
    return landing
landing_class = df['Outcome'].map(landing_map)
landing_class
```

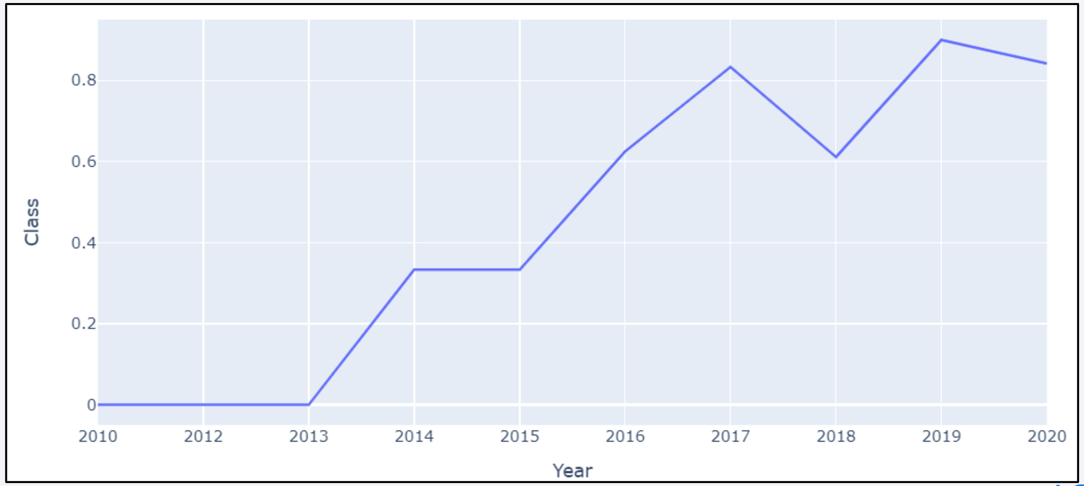
GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/3_Data%20wrangling.ipynb

EDA with Data Visualization

Before creating machine learning model, we used data visualization tools (plotly express) to do some EDA and tries to understand our features such as Orbit, Launch Site, Payload Mass and Launch Result.







GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/5_Exploring%20and%20Preparing%20Data.ipynb

EDA with SQL

- As we learned the relationship between our features, we could apply SQL to do some descriptive statistics:
  - Display the names of the unique launch sites in the space mission
  - Display average payload mass carried by booster version F9 v1.1
  - List the date when the first successfully landing outcome in ground pad was achieved.
  - List the total number of successful and failure mission outcomes
  - List the names of the booster versions which have carried the maximum payload mass.
  - Rank the count of successful landing outcomes between the date 04-06-2010 and 20-03-2017

GitHub Repo: [*https://github.com/Paul60209/Space-Y/blob/main/4_EDA%20with%20SQL.ipynb*](https://github.com/Paul60209/Space-Y/blob/main/4_EDA%20with%20SQL.ipynb)

Build an Interactive Map with Folium

- As SpaceX has launched rocket in different launch site, we were thinking if the launch location will impact on landing result. We use Folium to put our data on the map:
  - We marked each launch site on the map with `folium.map.Marker()`
  - We marked launch times on each launch site with `MarkerCluster().add_to()`
  - We added landing result on the tooltip of each launch site with `folium.Icon()`
  - We calculated the distance and drew a line between a launch site to its proximities with `folium.PolyLine()`

GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/6_Locations%20Analysis%20with%20Folium.ipynb

Build a Dashboard with Plotly Dash

- To help our shareholders to understand our features easily, we used plotly dash to build a interactive dashboard, here is the following steps:
  - First, we import our data as data frame.
  - Then, we used html() to create the layout of the dashboard.
  - We created interactive components (e.g. filters, figures) by using dcc package.
  - Next, we used plotly express to draw each charts.
  - Finally, the callback function would help us to communicate the parameters between dcc and our charts.

GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/7_spaceX_dash_app.py

Predictive Analysis (Classification)

- After we done our EDA, we could start to build our machine learning model and do our predictive analysis with below processes:

    1. Load our cleaned data as data frame.
    2. Check there is non-missing value and our features are already been encoded in one-hot encoding.
    3. Standardize our features to prevent features with wider ranges from dominating the distance metric.
    4. Split our dataset into two parts: train data and test data.
    5. Create several classification models and use GridSearchCV to find the best hyperparameters.
    6. Evaluate our models accuracy and determine our best model.

GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/8_Machine%20Learning%20Prediction.ipynb

Results

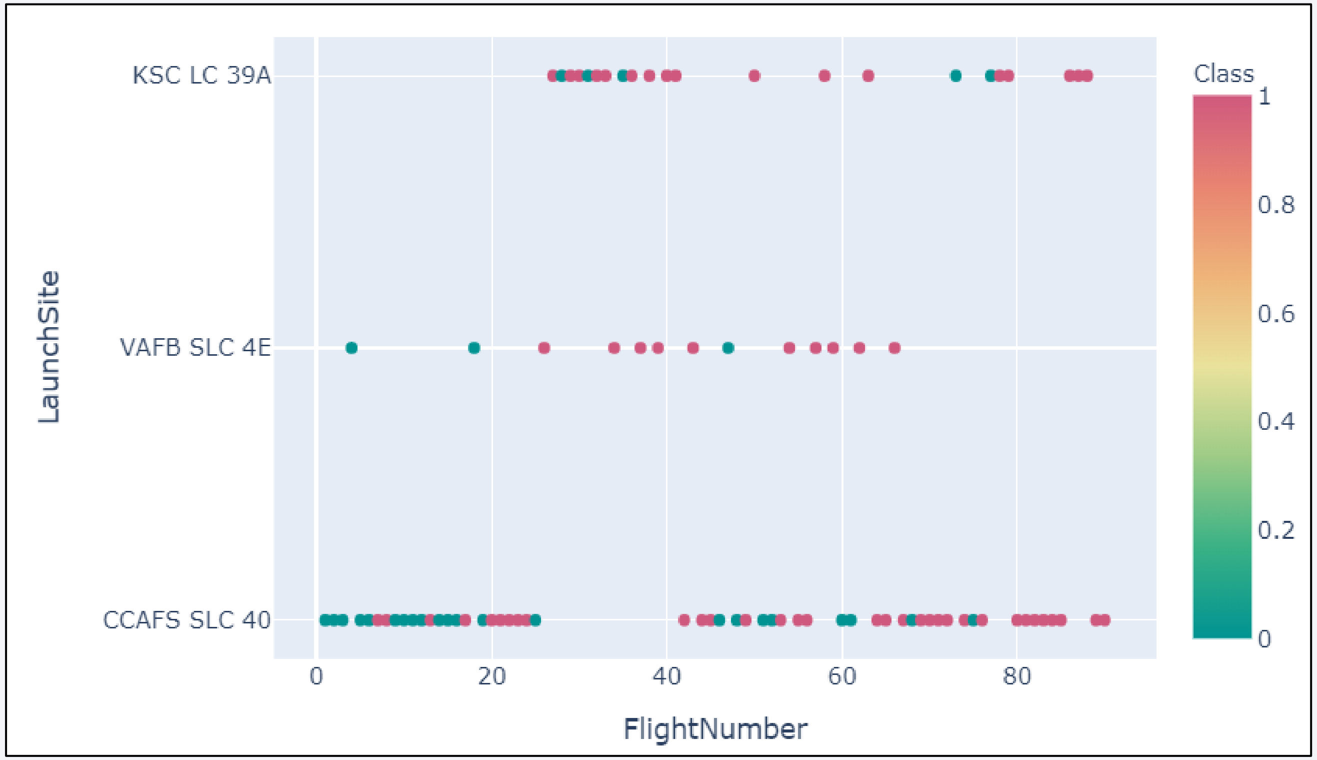
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

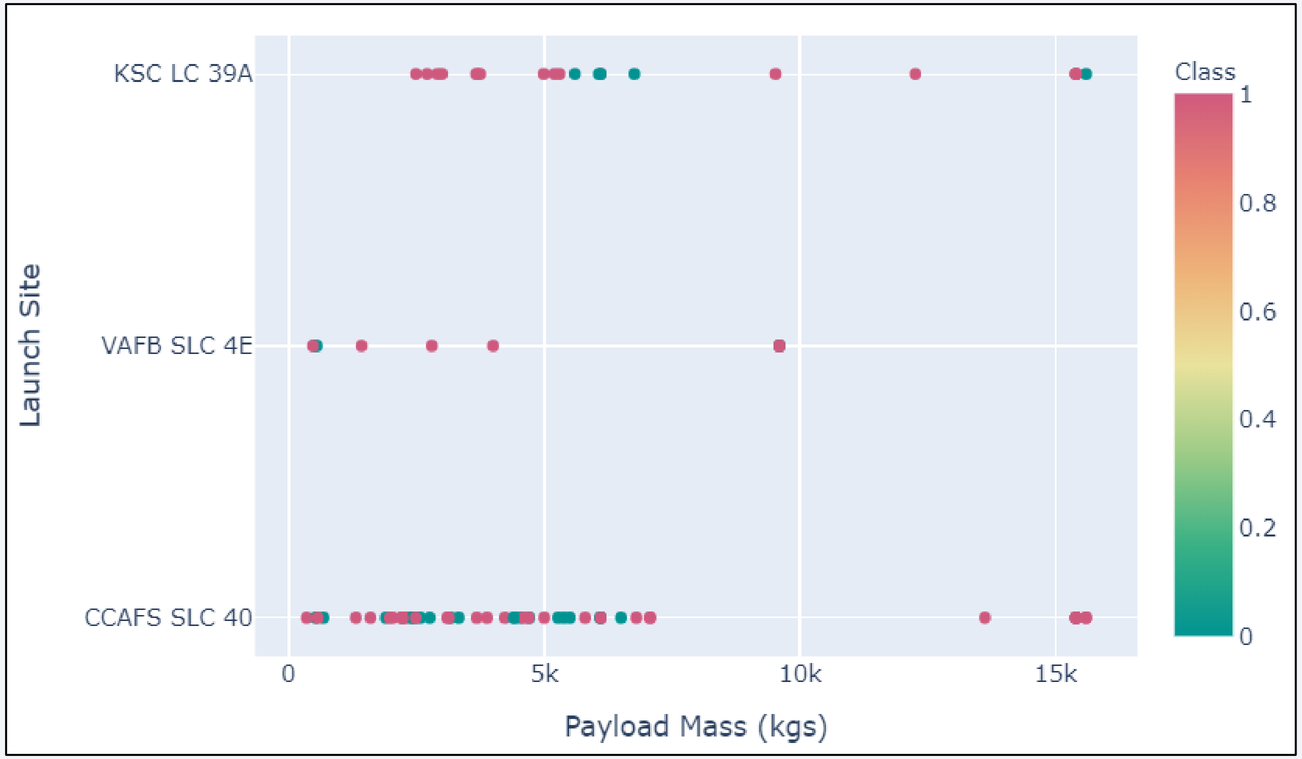
Flight Number vs. Launch Site

- The right scatter plot shows there is a significant positive relationship between the success rate and the flight number.
- Because the flight numbers are ordering by time, which means the landing success rate in SpaceX is improving.



GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/5_Exploring%20and%20Preparing%20Data.ipynb

Payload vs. Launch Site

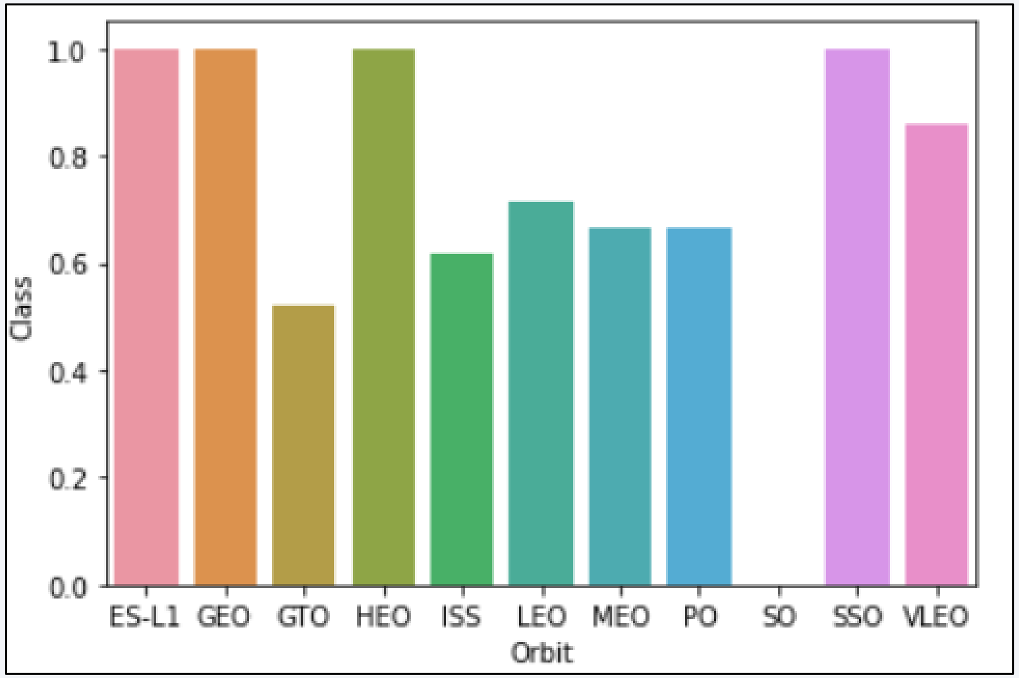


- The plot seems has greater success rate when is has higher payload in each launch site.
- If we focus on the launch site 'CCAFS SLC 40', we could see it has 100% success rate when payload over 10k(kgs).

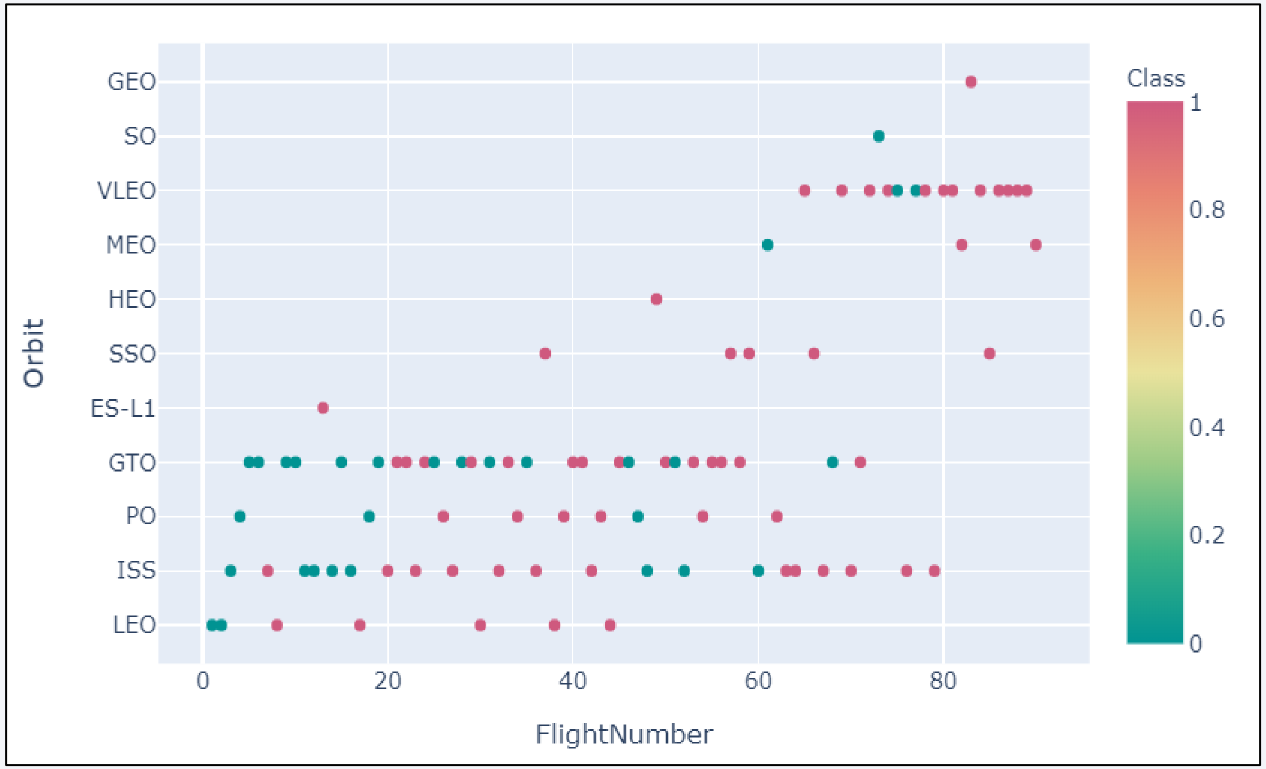
GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/5_Exploring%20and%20Preparing%20Data.ipynb

Success Rate vs. Orbit Type

- ES-LI, GEO, HEO and SSO both have 100% success rate.
- But the SO has 0% success rate (They only launched 1 time in SO orbit.).



Flight Number vs. Orbit Type



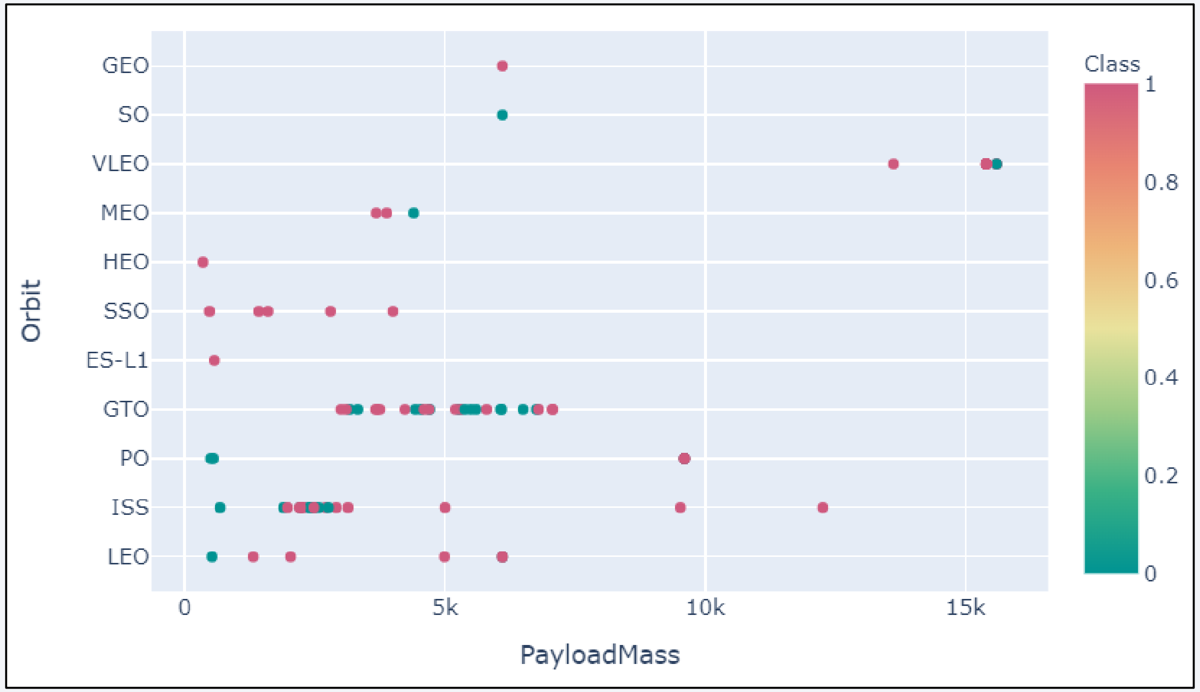
- The left figure shows Flight Number vs. Orbit Type. It's hard to find significant relationship between flight number orbit type.

GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/5_Exploring%20and%20Preparing%20Data.ipynb

Payload vs. Orbit Type

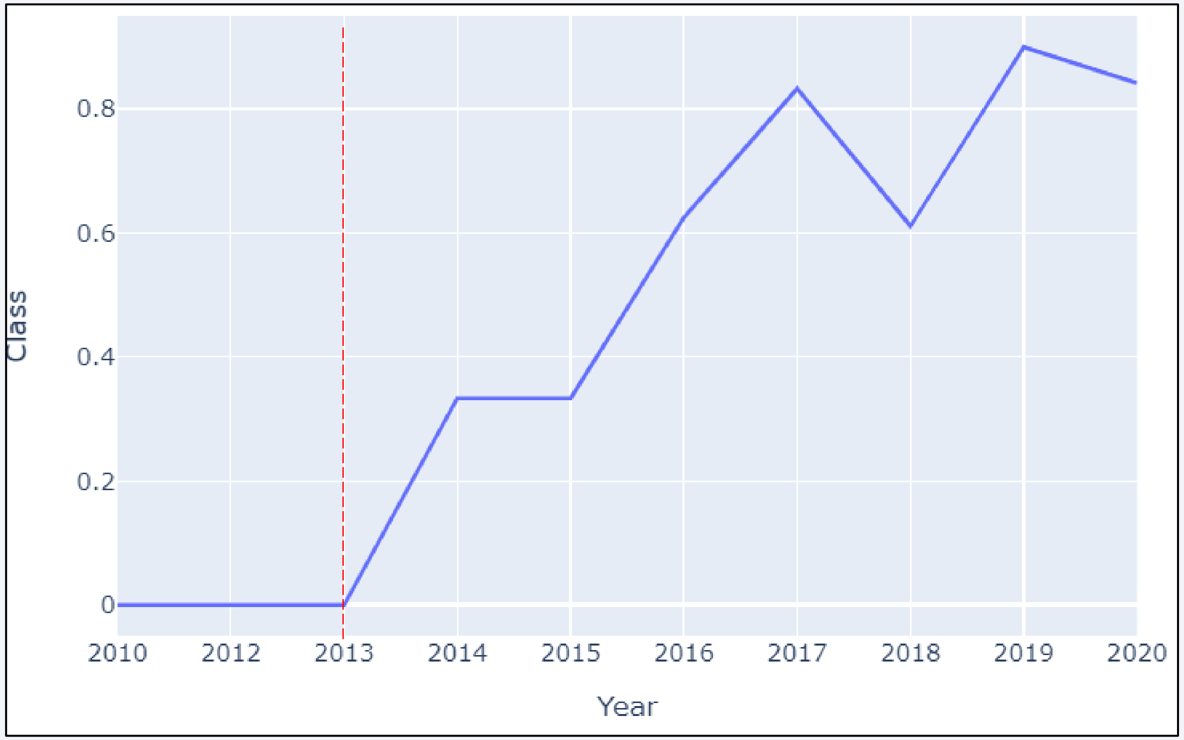
- With heavy payloads the successful landing or positive landing rate are more for PO, LEO and ISS.

- However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.



GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/5_Exploring%20and%20Preparing%20Data.ipynb

Launch Success Yearly Trend



- It's easy to observe that the success rate since 2013 kept increasing till 2020

GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/5_Exploring%20and%20Preparing%20Data.ipynb

All Launch Site Names

- We could use 'DISTINCT' or 'GROUP BY' keyword in SQL to find the names of the unique launch sites.
- In this time we used 'GROUP BY' with 'COUNT', it would show how many times launched in each site additionally.

```
%%sql
SELECT "Launch_Site", COUNT("Launch_Site")
FROM SPACEXTBL
GROUP BY "Launch_Site"
```

Launch_Site	COUNT("Launch_Site")
CCAFS LC-40	26
CCAFS SLC-40	34
KSC LC-39A	25
VAFB SLC-4E	16

GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/4_EDA%20with%20SQL.ipynb

Launch Site Names Begin with 'CCA'

- We use ‘WHERE’ and ‘LIKE’ to find the data which launch site start with ‘CAA’.
- And ‘LIMIT’ keyword could help us to control how many rows of data we select.

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0
08-12-2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677

GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/4_EDA%20with%20SQL.ipynb

Total Payload Mass

- The ‘SUM()’ function could add up chose column.
- Use ‘AS’ to give the calculated column a alias.

```
%%sql
SELECT SUM("PAYLOAD_MASS__KG_") AS 'TOTAL PAYLOAD'
FROM SPACEXTBL
WHERE "Customer" = "NASA (CRS)"
```

TOTAL PAYLOAD
45596

GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/4_EDA%20with%20SQL.ipynb

Average Payload Mass by F9 v1.1

- The ‘AVG()’ function could get the mean of the chose column.
- Use ‘AS’ to give the calculated column a alias.

```
%%sql
SELECT AVG("PAYLOAD_MASS__KG_") AS 'Avg. PAYLOAD'
FROM SPACEXTBL
WHERE "Booster_Version" like "F9 v1.1%"
```

Avg. PAYLOAD
2534.6666666666665

GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/4_EDA%20with%20SQL.ipynb

First Successful Ground Landing Date

- We used 'MIN()' function to find the first successful date, and the date is Jan. 5th 2017.

```
%%sql
SELECT MIN("Date") AS 'First Date'
FROM SPACEXTBL
WHERE "Landing _Outcome" like "Success (ground pad)"
```

First Date
01-05-2017

GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/4_EDA%20with%20SQL.ipynb

Successful Drone Ship Landing with Payload between 4000 and 6000

- We use ‘WHERE’ to find the results which is matched our criteria.
- When we have multiple criteria, we could use ‘AND’, ‘OR’ to connect our criteria in different logic.

```
%%sql
SELECT "Booster_Version"
FROM SPACEXTBL
WHERE "Landing _Outcome" like "Success (drone ship)"
AND "PAYLOAD_MASS__KG_" > 4000
AND "PAYLOAD_MASS__KG_" < 6000
```

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

GitHub Repo: *https://github.com/Paul60209/Space-Y/blob/main/4_EDA%20with%20SQL.ipynb*

Total Number of Successful and Failure Mission Outcomes

- We could use ‘COUNT()’ function to find Number of Outcome.
- Then use ‘GROUP BY’ to aggregate Outcome into sole type of outcome.

```
%%sql
SELECT "Mission_Outcome", COUNT("Mission_Outcome") AS Times
FROM SPACEXTBL
GROUP BY "Mission_Outcome"
```

Mission_Outcome	Times
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/4_EDA%20with%20SQL.ipynb

Boosters Carried Maximum Payload

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

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

- We use ‘subquery’ skill to find the max of payload first.
- We select our data again and insert the subquery in our WHERE clause.

GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/4_EDA%20with%20SQL.ipynb

2015 Launch Records

- First step, we use ‘SUBSTRING’ to get Month from Date.
- Find the Year in 2015 by using ‘SUBSTRING’ again.

```
%%sql
SELECT SUBSTRING("Date", 4, 2) as month
        ,"Landing _Outcome"
        ,"Booster_Version"
        ,"Launch_Site"
FROM SPACEXTBL
WHERE SUBSTRING("Date",7,4)='2015'
AND "Landing _Outcome" LIKE "%Failure (drone ship)%"
```

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

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

- We use 'GROUP BY' to aggregate the Date data, and 'ORDER BY', 'DESC' to show result in descending order.

```
%%sql
SELECT "Date", COUNT("Landing _Outcome") AS "Success_Times"
FROM SPACEXTBL
WHERE "Date" > "04-06-2010"
AND "Date" < "20-03-2017"
AND "Landing _Outcome" LIKE "%Success%"
GROUP BY "Date"
ORDER BY "Date" DESC
```

Date	Success_Times
19-02-2017	1
18-10-2020	1
18-08-2020	1
18-07-2016	1
18-04-2018	1
17-12-2019	1
16-11-2020	1
15-12-2017	1
15-11-2018	1
14-08-2017	1
14-08-2016	

GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/8_Machine%20Learning%20Prediction.ipynb

Section 3

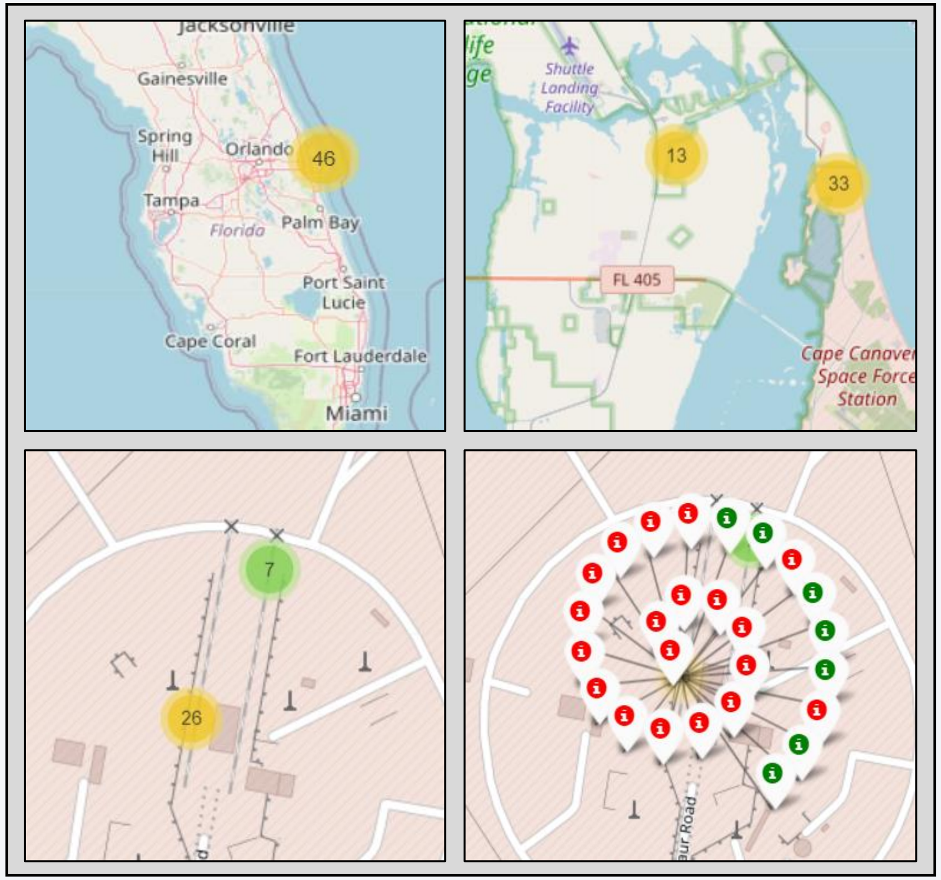
Launch Sites Proximities Analysis

All launch sites in global map

All SpaceX launch sites are in Florida and California in USA.

GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/6_Locations%20Analysis%20with%20Folium.ipynb

The success/failed launches for each



The number on the map are launch times and when we zoom in closer, the folium will separate the number in to two part.

If we click on the number, it would show the "i" icon in green and red, which means success or failed landing.

GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/6_Locations%20Analysis%20with%20Folium.ipynb

Launch Sites to its proximities

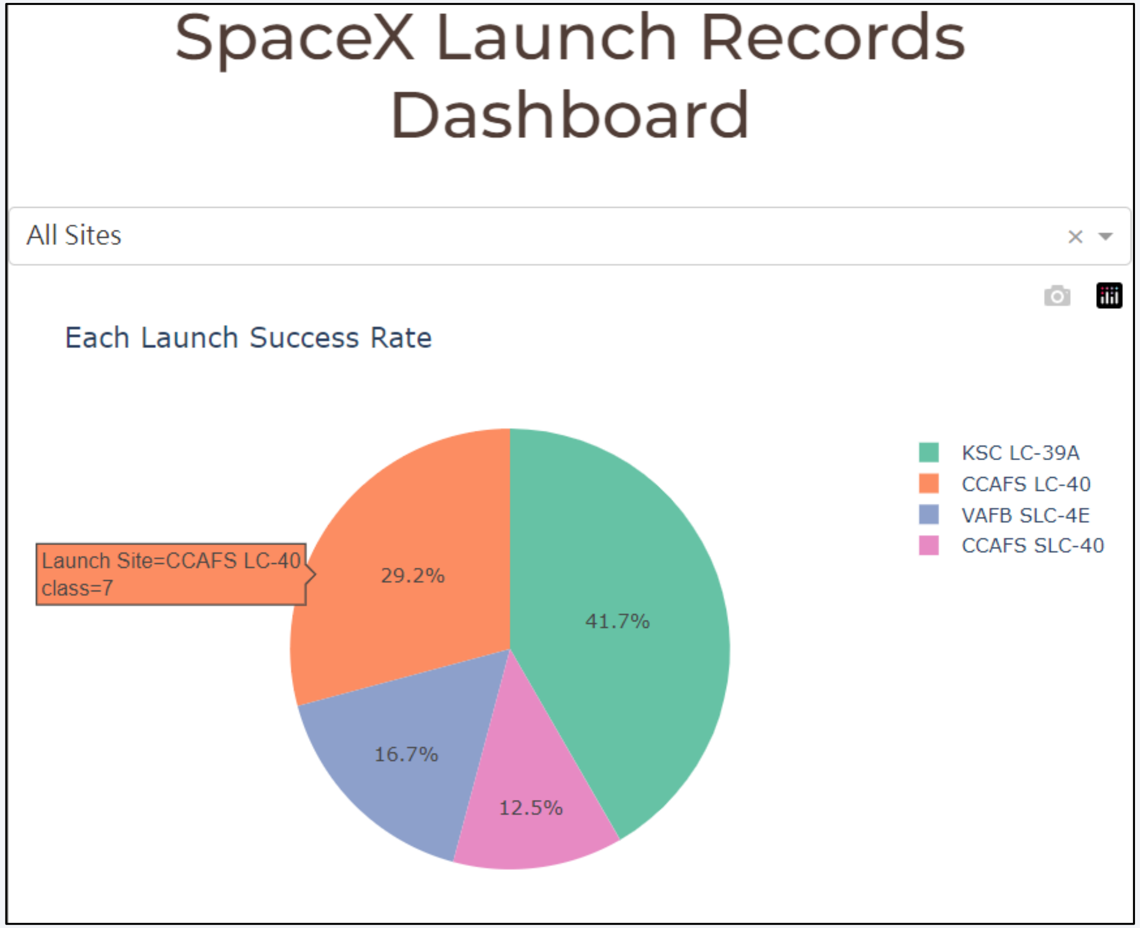
We also calculate the distances from launch site to its proximities(e.g. sea, railway, highway…etc)

GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/6_Locations%20Analysis%20with%20Folium.ipynb

Section 4

Build a Dashboard with Plotly Dash

Launch Success Rate

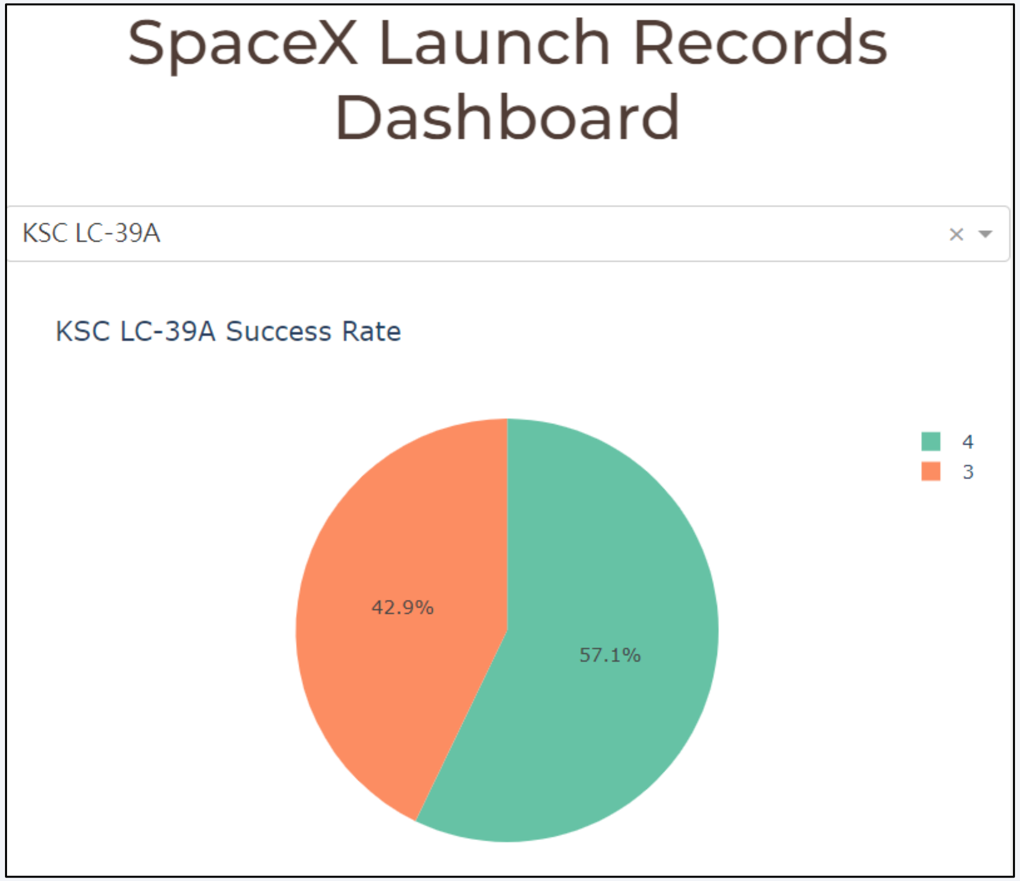


The filter's default value is 'All sites', it will shows the landing success rate of first stage in each launch site.

GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/7_spaceX_dash_app.py

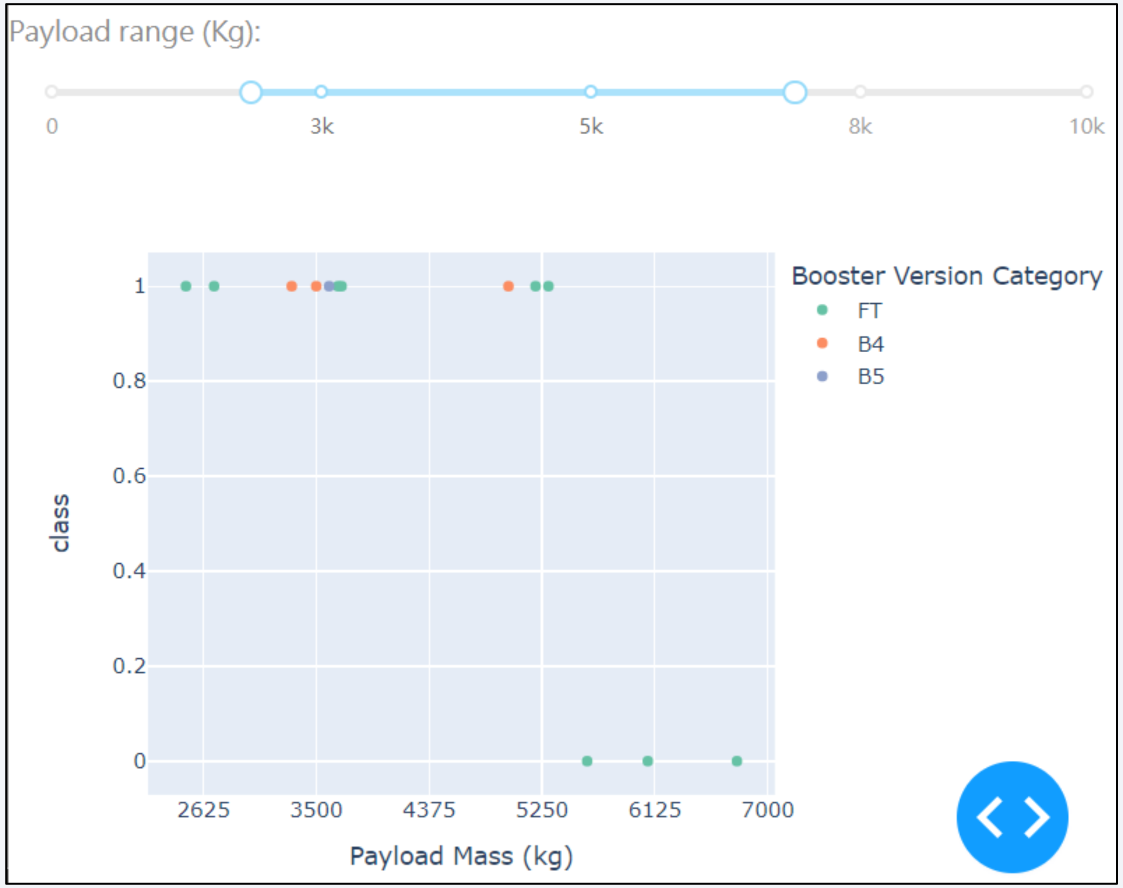
Success rate of specific launch site

When you choose a launch site (e.g. KSC KC-39A), it will shows how many times success/failed landing in the launch site.



GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/7_spaceX_dash_app.py

Relationship between payload and outcome



The figure shows the landing outcome in different payload mass, class=1 means success and class=0 means failed.

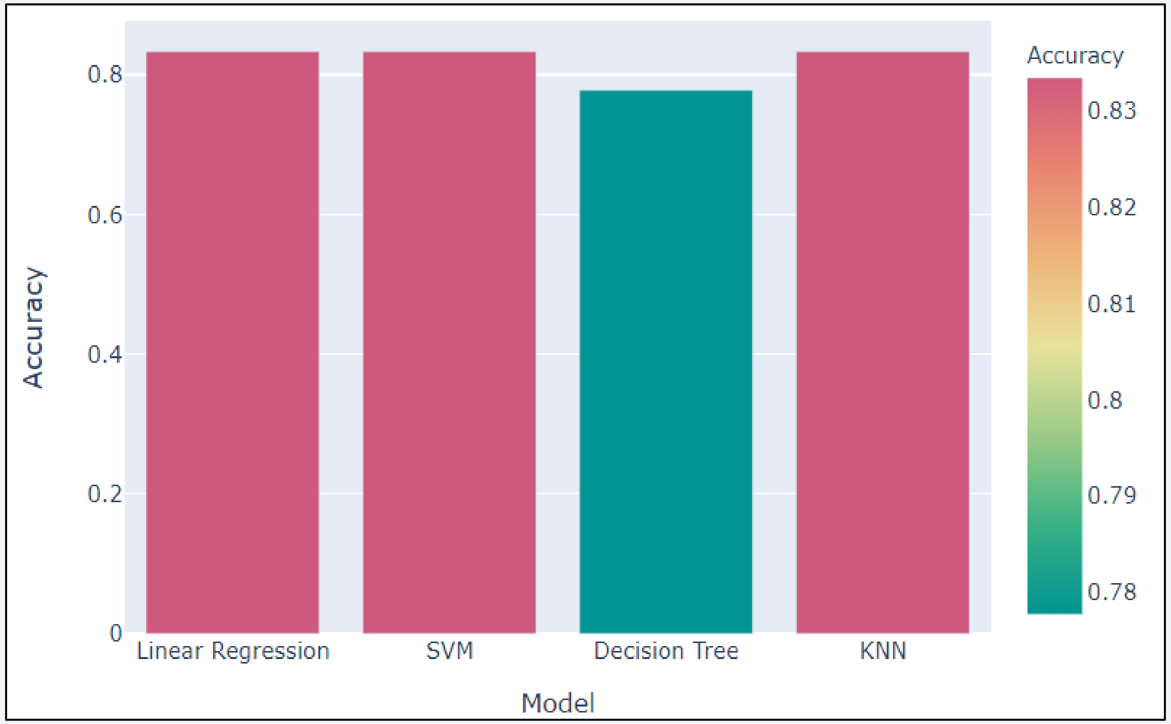
The above scope-bar could choose the specific range of payload mass.

GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/7_spaceX_dash_app.py

Section 5

Predictive Analysis (Classification)

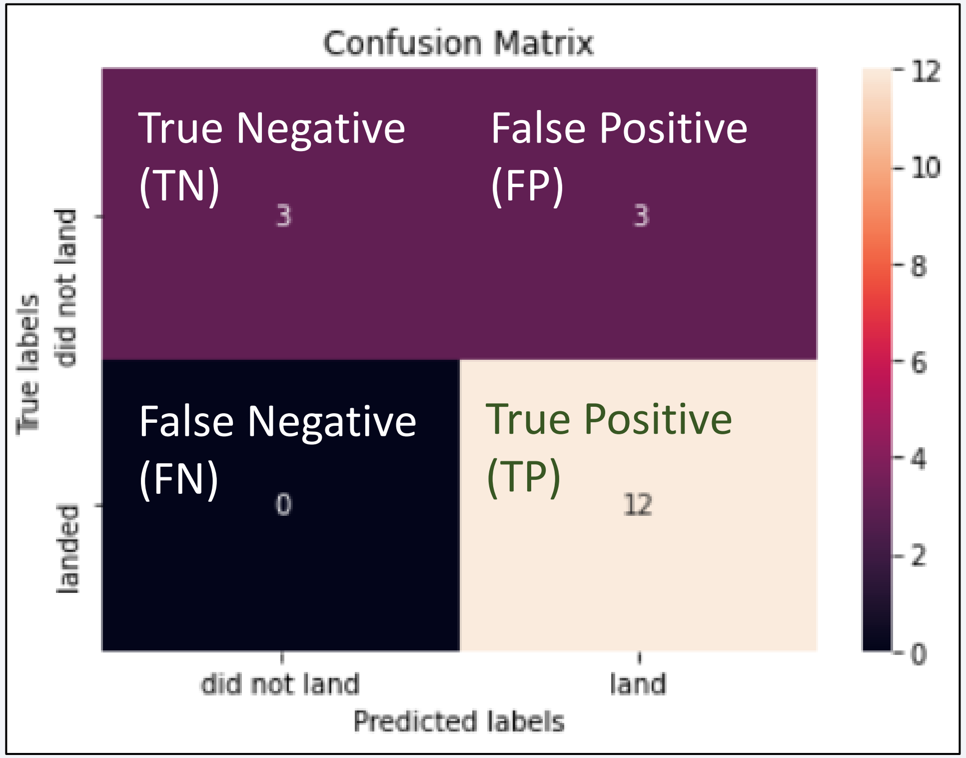
Classification Accuracy



The accuracies of Linear Regression, SVM and KNN are 83%, but the accuracy of decision tree is only 77%

GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/8_Machine%20Learning%20Prediction.ipynb

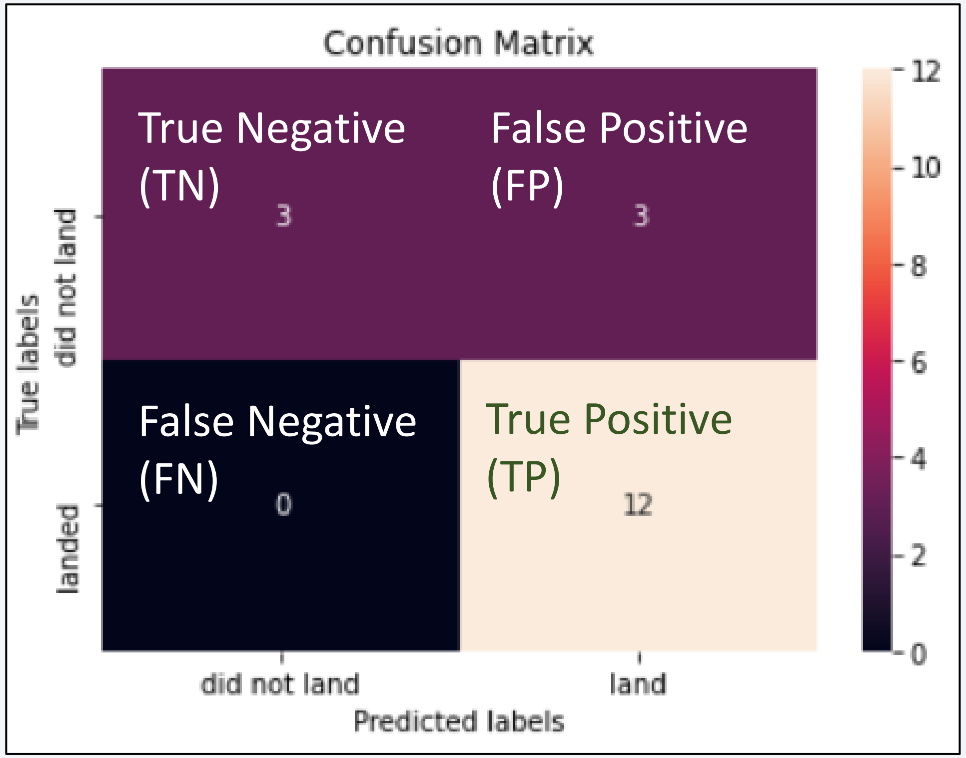
Confusion Matrix of linear regression



- Recall Rate/Sensitivity(召回率)=TP/(TP+FN)=100%
- Specificity=TN/(FP+TN)=50%
- Precision(準確率)=TP/(TP+FP)=80%
- Accuracy(正確率)=(TP+TN)/(P+N)=83%

GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/8_Machine%20Learning%20Prediction.ipynb

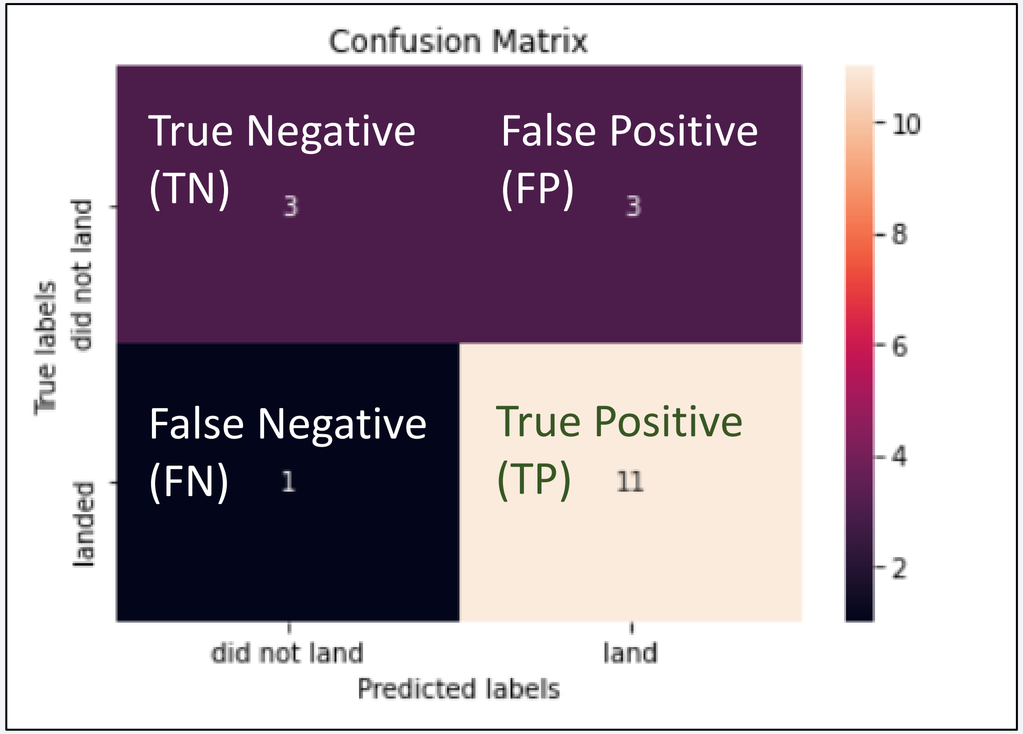
Confusion Matrix of SVM



- Recall Rate/Sensitivity(召回率)=TP/(TP+FN)=100%
- Specificity=TN/(FP+TN)=50%
- Precision(準確率)=TP/(TP+FP)=80%
- Accuracy(正確率)=(TP+TN)/(P+N)=83%

GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/8_Machine%20Learning%20Prediction.ipynb

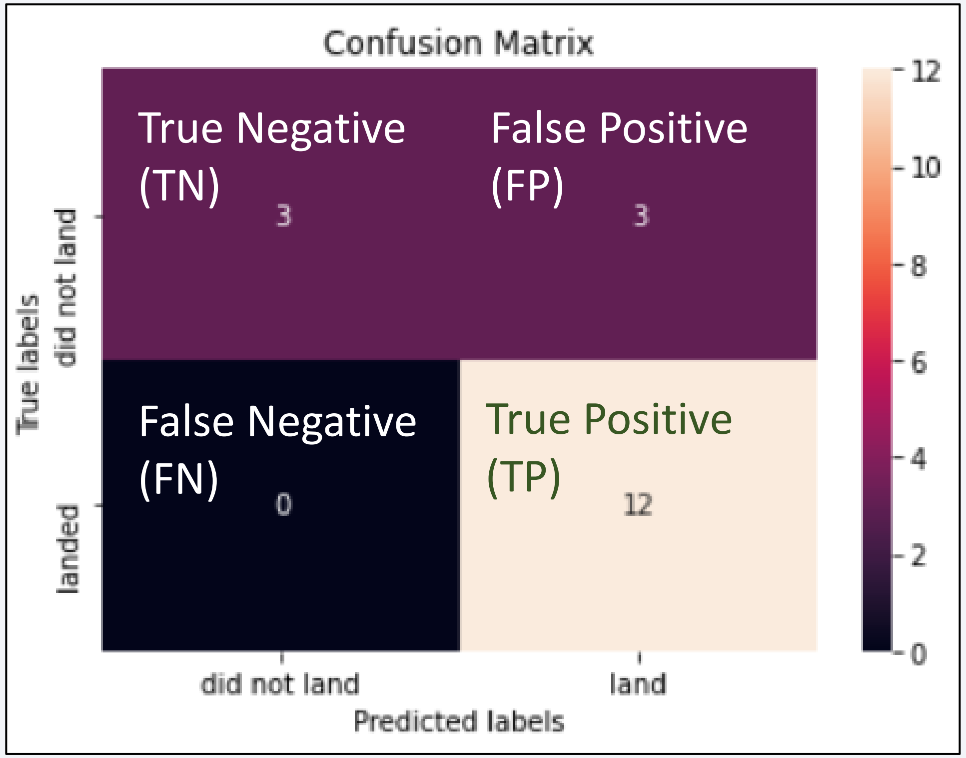
Confusion Matrix of decision tree



- Recall Rate/Sensitivity(召回率)=TP/(TP+FN)=91.6%
- Specificity=TN/(FP+TN)=50%
- Precision(準確率)=TP/(TP+FP)=78.6%
- Accuracy(正確率)=(TP+TN)/(P+N)=77.8%

GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/8_Machine%20Learning%20Prediction.ipynb

Confusion Matrix of KNN



- Recall Rate/Sensitivity(召回率)=TP/(TP+FN)=100%
- Specificity=TN/(FP+TN)=50%
- Precision(準確率)=TP/(TP+FP)=80%
- Accuracy(正確率)=(TP+TN)/(P+N)=83%

GitHub Repo: https://github.com/Paul60209/Space-Y/blob/main/8_Machine%20Learning%20Prediction.ipynb

Conclusions

- We found site with highest score which is KSC LC-39A
- The payload 0~5,000 kgs is more diverse than 6,000~10,000 kgs
- The landing success rate of first stage was increasing in 2013 to 2020.
- The Linear Regression, SVM and KNN are suitable model for this prediction.

Appendix

- All data, codes, directions, charts and figure could be found on my GitHub:

https://github.com/Paul60209/Space-Y

Thank you!