

Data Science and Machine Learning Capstone Project

ZHANG MINGWEI

16/10/2024

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusions

Introduction

Just recent news reminds me I used to completed this SpaceX Projects few years ago, companies have been in fierce competition to make space travel more affordable. One of the leading successes in this field is SpaceX, which has significantly reduced the cost of rocket launches by reusing the first stage of its boosters which is already come true.

This study aims to explore the factors that influence the successful landing of the first stage, as well as determine how these factors affect the cost of each launch and SpaceX's decision on whether to reuse the booster

Data Collection

Data Collection via Web Scraping

Create a `BeautifulSoup` object from the HTML `response`

```
[11]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html_data.text, 'html.parser')
```

Print the page title to verify if the `BeautifulSoup` object was created properly

```
[12]: # Use soup.title attribute
soup.title
```

```
[12]: <title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

Data Collection via API connection

```
In [6]: spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
In [7]: response = requests.get(spacex_url)
```

Check the content of the response

```
In [8]: print(response.content)
```

```
b' [{"fairings": {"reused": false, "recovery_attempt": false, "recovered": false, "ships": []}, "links": {"patch": {"small": "https://images2.imgbox.com/94/f2/NN6Ph45r_o.png", "large": "https://images2.imgbox.com/5b/02/QcxHub5V_o.png"}, "reddit": {"campaign": null, "launch": null, "media": null, "recovery": null}, "flickr": {"small": [], "original": []}, "presskit": null, "webcast": "https://www.youtube.com/watch?v=0a_00nJ_Y88", "youtube_id": "0a_00nJ_Y88", "article": "https://www.space.com/2196-spacex-inaugural-falcon-1-rocket-lost-launch.html", "wikipedia": "https://en.wikipedia.org/wiki/DemoSat"}, "static_fire_date_utc": "2006-03-17T00:00:00.000Z", "static_fire_date_unix": 1142553600, "net": false, "window": 0, "rocket": "5e9d0d95eda69955f709d1eb", "success": false, "failures": [{"time": 33, "altitude": null, "reason": "merlin engine failure"}], "details": "Engine failure at 33 seconds and loss of vehicle", "crew": [], "ships": [], "capsule": {"payloads": ["5eb0e4b5b6c3bb0006eeb1e1"], "launchpad": "5e9e4502f5090995de566f86", "flight_number": 1, "name": "FalconSat", "date_utc": "2006-03-24T22:30:00.000Z", "date_unix": 1143239400, "date_local": "2006-03-25T10:30:00+12:00", "date_precision": "hour", "upcoming": false, "cores": [{"core": "5e9e289df35918033d3b2623", "flight": 1, "gridfins": false, "legs": false, "reused": false, "landing_attempt": false, "landing_success": null, "landing_type": null, "landpad": null}], "auto_update": true, "tbd": false, "launch_library_id": null, "id": "5eb87cd9ffd86e000604b32a"}, {"fairings": {"reused": false, "recovery_attempt": false, "recovered": false, "ships": []}, "links": {"patch": {"small": "https://images2.imgbox.com/5b/02/QcxHub5V_o.png", "large": "https://images2.imgbox.com/5b/02/QcxHub5V_o.png"}, "reddit": {"campaign": null, "launch": null, "media": null, "recovery": null}, "flickr": {"small": [], "original": []}, "presskit": null, "webcast": "https://www.youtube.com/watch?v=0a_00nJ_Y88", "youtube_id": "0a_00nJ_Y88", "article": "https://www.space.com/2196-spacex-inaugural-falcon-1-rocket-lost-launch.html", "wikipedia": "https://en.wikipedia.org/wiki/DemoSat"}, "static_fire_date_utc": "2006-03-17T00:00:00.000Z", "static_fire_date_unix": 1142553600, "net": false, "window": 0, "rocket": "5e9d0d95eda69955f709d1eb", "success": false, "failures": [{"time": 33, "altitude": null, "reason": "merlin engine failure"}], "details": "Engine failure at 33 seconds and loss of vehicle", "crew": [], "ships": [], "capsule": {"payloads": ["5eb0e4b5b6c3bb0006eeb1e1"], "launchpad": "5e9e4502f5090995de566f86", "flight_number": 1, "name": "FalconSat", "date_utc": "2006-03-24T22:30:00.000Z", "date_unix": 1143239400, "date_local": "2006-03-25T10:30:00+12:00", "date_precision": "hour", "upcoming": false, "cores": [{"core": "5e9e289df35918033d3b2623", "flight": 1, "gridfins": false, "legs": false, "reused": false, "landing_attempt": false, "landing_success": null, "landing_type": null, "landpad": null}], "auto_update": true, "tbd": false, "launch_library_id": null, "id": "5eb87cd9ffd86e000604b32a"}]
```

Data Wrangling

• Data Cleansing

- Step1: Parse those data into table from HTML
- Step2: using loop function to get those useful information.
- Step3: Store those data into dictionary
- Step4: Frame data from dictionary

1

try to find all tables on the wiki page first. If you need to refresh your memory about BeautifulSoup, please check the external reference link towards the end of this lab

```
[16]: # Use the find_all function in the BeautifulSoup object, with element type 'table'
# Assign the result to a list called 'html_tables'
html_tables=soup.find_all('table')
```

Starting from the third table is our target table contains the actual launch records.

```
[17]: # Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

```
<table class="wikitable plainrowheaders collapsible" style="width: 100%;">
<tbody><tr>
<th scope="col">Flight No.
</th>
<th scope="col">Date and<br/>time ( <a href="/wiki/Coordinated_Universal_Time" title="Coordinated Universal Time">UTC</a>)
</th>
<th scope="col"><a href="/wiki/List_of_Falcon_9_first-stage_boosters" title="List of Falcon 9 first-stage boosters">Version,<br/>Booster</a> <sup class="reference" id
```

2

```
[22]: 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
# Append the Non-empty column name ('if name is not None and len(name) > 0') into a
for i in first_launch_table.find_all('th'):
    col_name = extract_column_from_header(i)
    if col_name is not None and len(col_name)>0:
        column_names.append(col_name)

column_names
```

```
[22]: ['Flight No.',
      'Date and time ( )',
      'Launch site',
      'Payload',
      'Payload mass',
      'Orbit',
      'Customer',
      'Launch outcome']
```

3

```
[24]: 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']=[]
```

4

dataframe from it.

```
[26]: df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

We can now export it to a **CSV** for the next section, but to make the answers consistent and in case you have difficulties finishing this lab.

Following labs will be using a provided dataset to make each lab independent.

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

EDA - SQL

Display the names of the unique launch sites in the space mission

```
[9]: %sql Select distinct Launch_Site from SPACEXTABLE
```

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

Done.

```
[9]: Launch_Site
```

CCAFS LC-40

VAFB SLC-4E

KSC LC-39A

CCAFS SLC-40

Display 5 records where launch sites begin with the string 'KSC'

Display 5 records where launch sites begin with the string KSC

```
[12]: %sql Select * from SPACEXTABLE where Launch_Site = 'KSC LC-39A' LIMIT 5
```

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

Done.

```
[12]:
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS_KG
------	------------	-----------------	-------------	---------	-----------------

2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	249
------------	----------	---------------	------------	---------------	-----

2017-03-16	6:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	560
------------	---------	-------------	------------	-------------	-----

2017-03-30	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	530
------------	----------	---------------	------------	--------	-----

2017-05-01	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	530
------------	----------	---------------	------------	---------	-----

2017-05-15	23:21:00	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	607
------------	----------	-------------	------------	---------------	-----

EDA - SQL

Display the total payload mass carried by boosters launched by NASA (CRS)

```
(CRS)
[13]: %sql select distinct Customer from SPACEXTABLE
* sqlite:///my_data1.db
Done.
```

Customer
SpaceX
NASA (COTS) NRO
NASA (COTS)
NASA (CRS)
MDA
SES
Thaicom
Orbcomm
AsiaSat
U.S. Air Force NASA NOAA
ABS Eutelsat
Turkmenistan National Space Agency

Display average payload mass carried by booster version F9 V1.1

Display average payload mass carried by booster version F9 v1.1

```
[15]: %sql select Avg(PAYLOAD_MASS_KG_) from SPACEXTABLE where Booster_Ver
* sqlite:///my_data1.db
Done.
```

Avg(PAYLOAD_MASS_KG_)
2928.4

EDA - SQL

List the date where the succesful landing outcome in drone ship was acheived

```
[17]: %sql select Date From SPACEXTABLE where landing_Outcome = 'Success (d
```

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

Done.

```
[17]:
```

Date

2016-04-08

2016-05-06

2016-05-27

2016-08-14

2017-01-14

2017-03-30

2017-06-23

2017-06-25

2017-08-24

List the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000

```
[18]: %sql select distinct landing_Outcome from SPACEXTABLE
```

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

Done.

```
[18]:
```

Landing_Outcome

Failure (parachute)

No attempt

Uncontrolled (ocean)

Controlled (ocean)

Failure (drone ship)

Precluded (drone ship)

Success (ground pad)

Success (drone ship)

Success

Failure

No attempt

EDA - SQL

List the total number of successful and failure mission outcomes

```
[20]: %sql Select Mission_Outcome, Count(1) From SPACEXTABLE GROUP BY Mission_Outcome
```

* sqlite:///my_data1.db
Done.

```
[20]:
```

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

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
[21]: %sql select Booster_version From SPACEXTABLE WHERE PAYLOAD_MASS_KG = (select max(PAYLOAD_MASS_KG) From SPACEXTABLE)
```

* sqlite:///my_data1.db
Done.

```
[21]:
```

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

EDA - SQL

List the records which will display the month names, succesful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017

```
[23]: %sql select substr(date,6,2) as 'Month', Customer, Landing_Outcome, B
```

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

Done.

```
[23]:
```

Month	Customer	Landing_Outcome	Booster_Version	Launch_Site
06	SpaceX	Failure (parachute)	F9 v1.0 B0003	CCAFS LC-40
12	NASA (COTS) NRO	Failure (parachute)	F9 v1.0 B0004	CCAFS LC-40

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

```
[28]: (select landing_Outcome, COUNT(1) AS 'counter' FROM SPACEXTABLE where
```

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

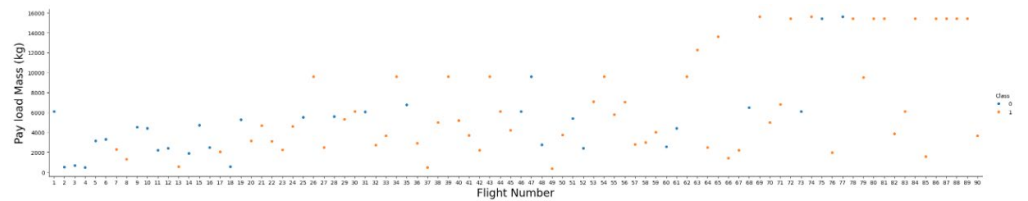
Done.

```
[28]:
```

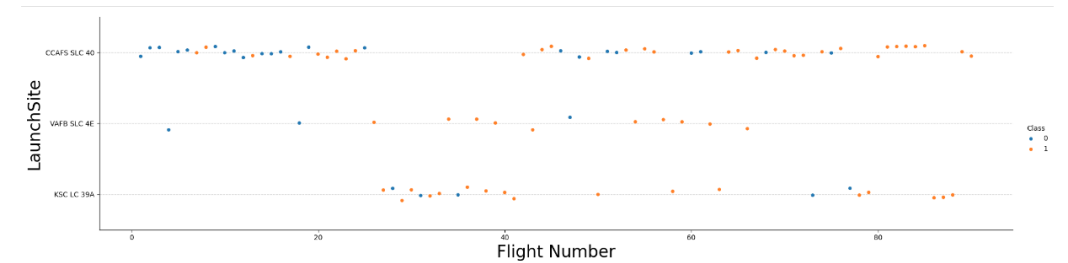
rank	landing_Outcome	counter
1	No attempt	10
2	Failure (drone ship)	5
2	Success (drone ship)	5
4	Controlled (ocean)	3
4	Success (ground pad)	3
6	Failure (parachute)	2
6	Uncontrolled (ocean)	2
8	Precluded (drone ship)	1

EDA – Data Visualization

Scatter Plot – Flight number Vs PayloadMass



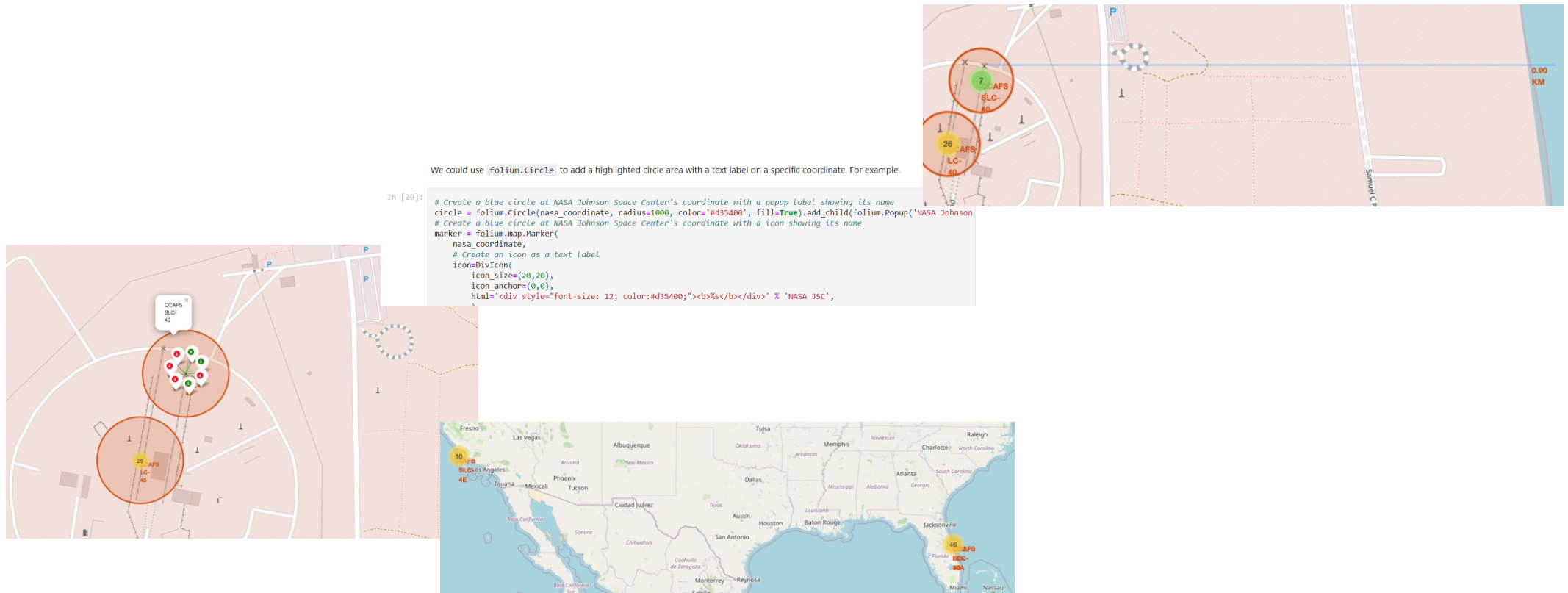
Scatter Plot – FlightNumber Vs LaunchSite



EDA - Interactive Visual Analytics – Folium

Interactive Map with Folium:

- Label and text on the map
- Land marking with colour
- Show distance



EDA - Interactive Visual Analytics -Plotly Dashboard

1/2

Can't run properly in Skills Network, so have to run in local machine:

Install PowerShell and Python3.12 version

Dashboard with Select launch site

- Pie chat
- Range of payload mass]
- Scater plot class Vs. Payload mass by Booster version

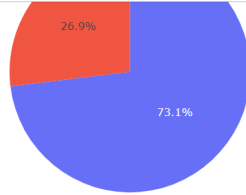
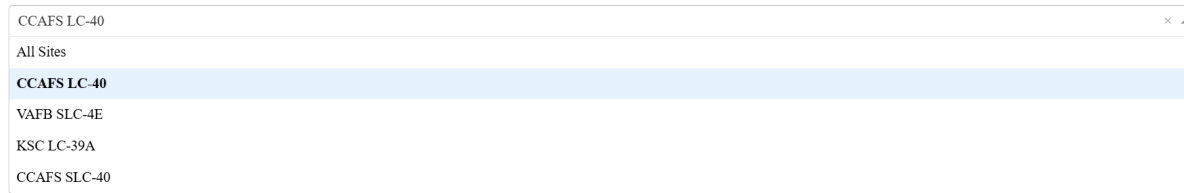
The screenshot displays a local development environment with three main components:

- Left Panel:** A sidebar with instructions for running the application. It includes steps like "Observe the port number (8050) shown in the terminal," "In the left Navigation Pane click on Others and click Launch Application option under it. Enter the application port number as 8050. Click Your Application .", and "You should see a nearly blank web page indicating a successfully running dash app. Next, let's fill the skeleton app with required".
- Center Panel:** A web browser window showing the "SpaceX Launch Records Dashboard". The dashboard has a title bar and a large empty plot area.
- Right Panel:** A Windows PowerShell terminal window showing the command prompt and the output of the application. The output includes a warning about the deprecated 'dash_core_components' package and a list of HTTP requests and responses from the application.

EDA - Interactive Visual Analytics -Plotly

Dashboard 2/2

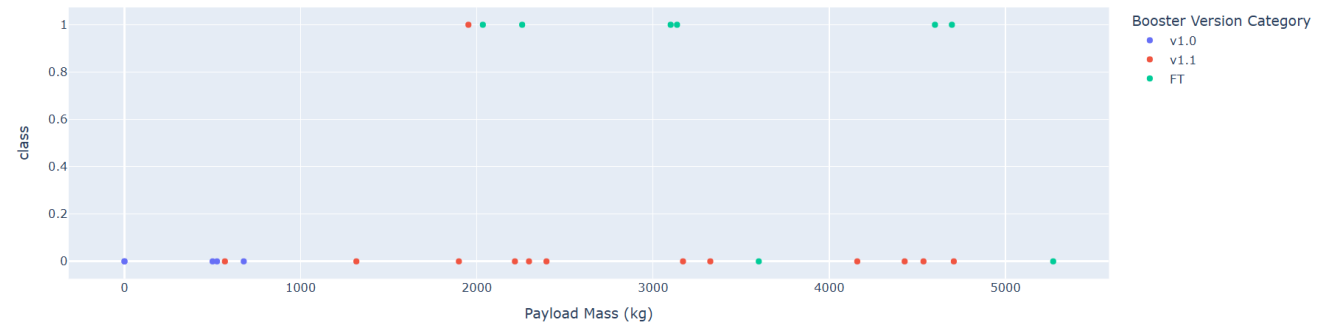
SpaceX Launch Records Dashboard



Payload range (Kg):



Correlation Between Payload and Success for Site CCAFS LC-40



Predictive Analysis 1/2

All test accuracy are same: 0.8334

```
[36]: print(log_score, svm_score, tree_score, knn_score)
0.8333333333333334 0.8333333333333334 0.8333333333333334 0.8333333333333334
```

```
]: print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("accuracy :",logreg_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters) {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8464285714285713
```

```
[23]: print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters) {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8482142857142856
```

```
]: print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters) {'criterion': 'entropy', 'max_depth': 4, 'max_features': 'sqrt', 'min_samples_split': 10, 'splitter': 'best'}
accuracy : 0.875
```

```
[33]: print("tuned hpyerparameters :(best parameters) ",knn_cv.best_params_)
print("accuracy :",knn_cv.best_score_)
```

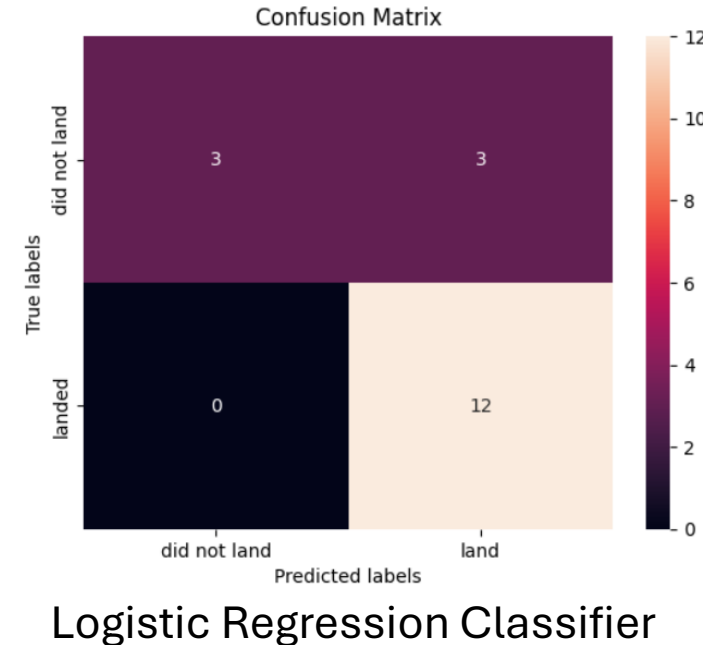
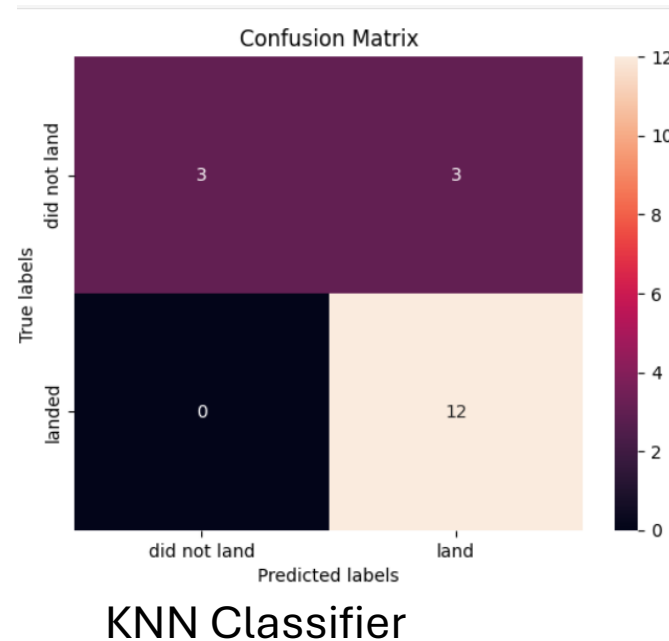
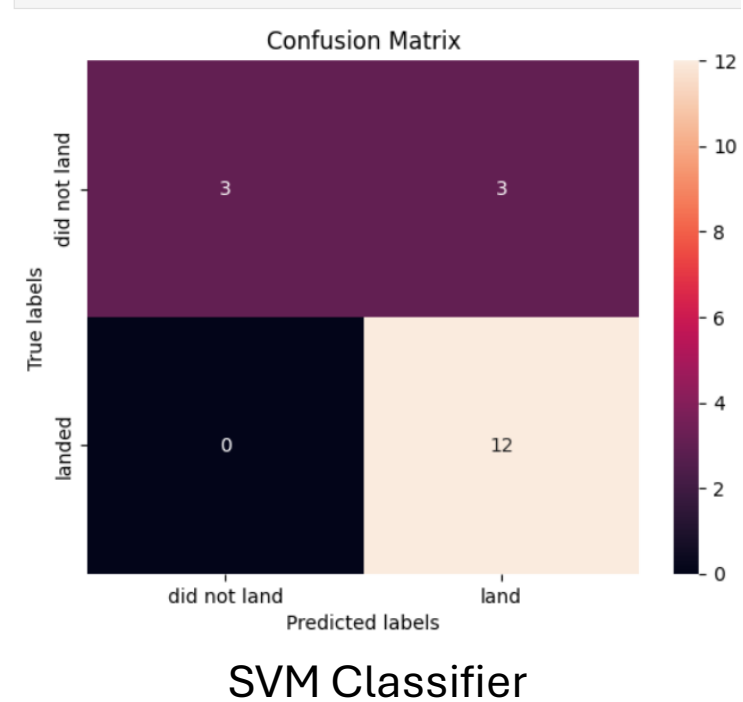
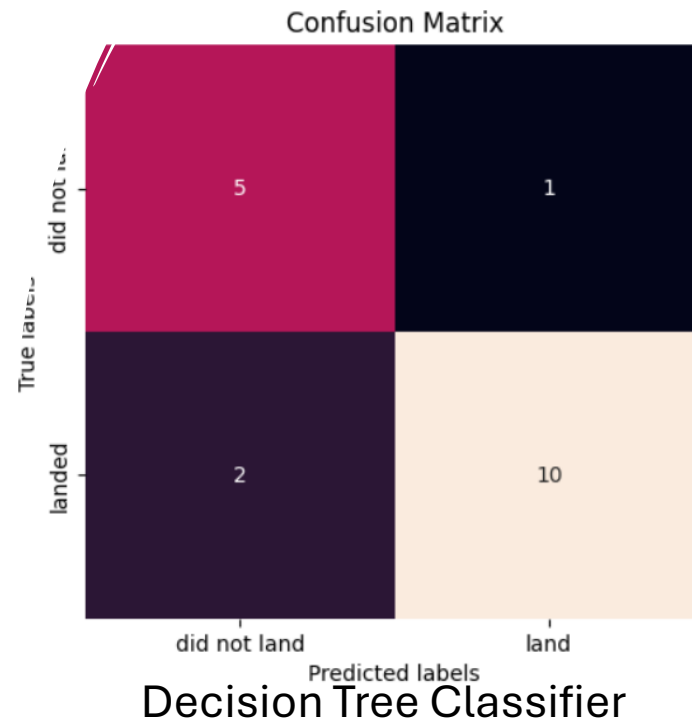
```
tuned hpyerparameters :(best parameters) {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
accuracy : 0.8482142857142858
```

Classifier Train Accuracy:

- KNN: 0.8482
- Decision Tree: 0.8750
- SVM: 0.8482
- Logreg: 0.8464

Predictive Analysis 2/2

- All models logistic regression, support vector machine, Decision tree classifier and K-nearest neighbors



Conclusion

- Success rate increased over more years, and also lower payload mass has higher success rate.
- The lowest success rate is SO type
- If payload range less than 5000kg, it shows best performance
- Classification method show same result for predicting.
- KSC LC-39A is good launch site
- Based on launch site position in the map (Folium Analysis), launch site will be built closed to water, it is kind of protect debris. And also far away from city. and it is closed to railways and highways, it will be easier for fuel truck