Data Science Capstone project

Tung Guangwei Kenny

29 August 2021

Outline

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

Executive Summary

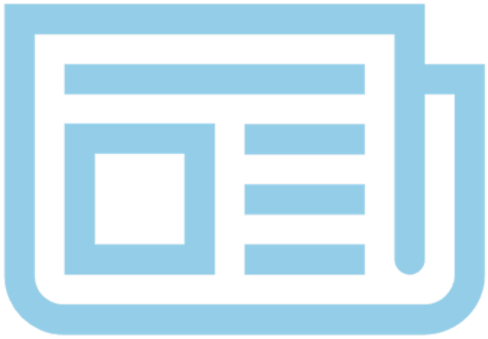
- A study of Falcon 9's historical launches had been conducted to identify the success rate of the first stage launch landing successfully.
- Data were obtained from web sources by API and web crawling. Exploratory data analysis is performed, and thereafter perform data wrangling to prepare for modeling
- A good accuracy model was obtained, however due to the size of the dataset and interpretation of the classification model, there is opportunities for improvements to provide a better accuracy and precision.

Introduction

- As a new data scientist in a new startup to compete with SpaceX, the paper follows the Data Science methodology to study the launch data from SpaceX to make informed bids against SpaceX for a rocket launch.

- The objective is to study and come up with an effective model to predict the likelihood of the Falcon 9's first stage rocket landing successfully, and determine the price of each launch, as this contributes to the large cost savings if the first stage can be reused.

Methodology



- Data collection methodology:
  - The data are collected via Wikipedia via web crawling and the use of REST API to obtain the data of previous SpaceX launches.
- Perform data wrangling
  - The data were then explored using pandas library to check for completeness, encoding the variables using dummy variables for the purpose of statistics modeling.
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Exploratory data analysis is performed next with the use of SQL and visualization tools (Seaborn) to identify any patterns and have an broad understand of the data at hand.
- Perform interactive visual analytics using Folium and Plotly Dash
  - The launch sites and the mission outcomes are plotted using Folium and Plotly Dash to find any geographical insights of Falcon 9's launches.
- Perform predictive analysis using classification models
  - Finally, 4 classification models are created and evaluated to find the best performing model to predict the mission outcome of a launch.

Data Collection

Data collection – SpaceX API

One of the data sources is from the following API endpoint: **api.spacexdata.com/v4/launches/past**

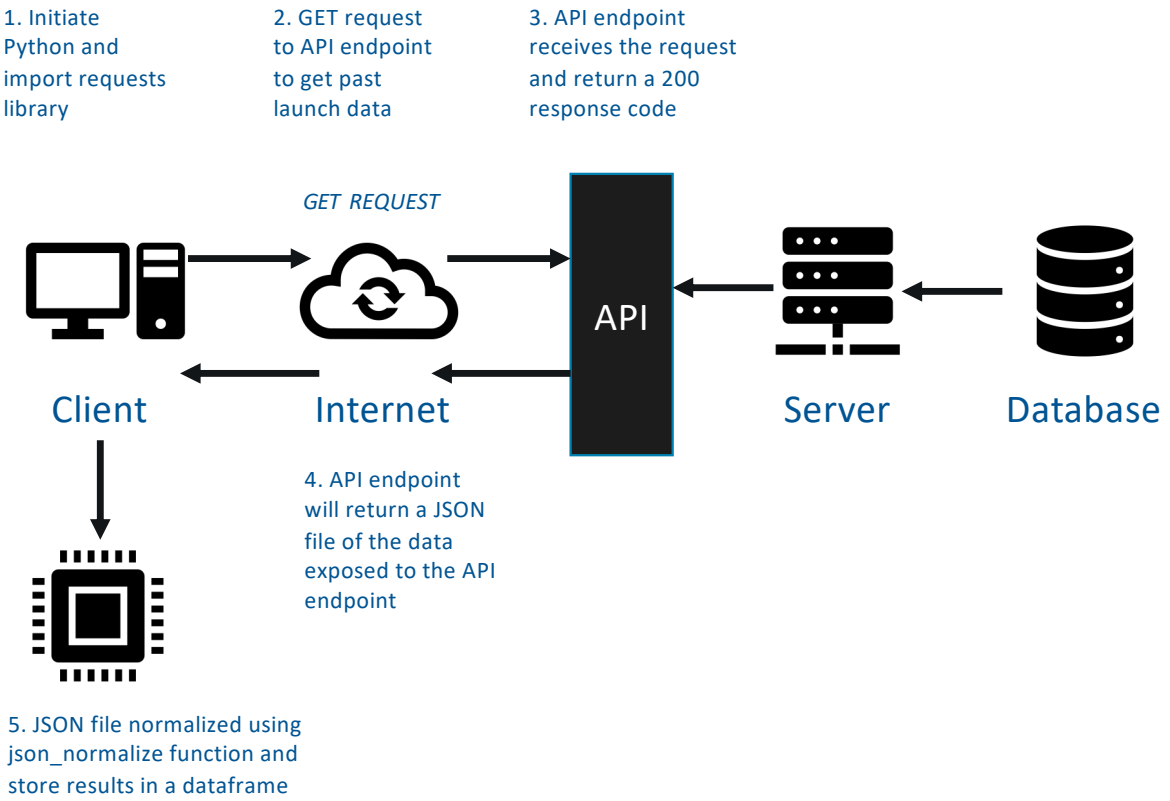
The API will give us data about past launches, including information about the rocket used, payload delivered, launch specifications, landing specifications, and landing outcome

Libraries used in python: **requests**

By calling a GET requests from the API endpoint, the data will be returned in JSON format.

The results are a list of JSON objects for each launch.

The JSON result is converted to a DataFrame as a flat table through the use of the **json_normalize** function.



Github url : *https://github.com/jidokasg/IBM-Watson-Studio-captstone/blob/master/capstone-week1-bs4.ipynb*

Data collection – Web scraping

Another data sources is from Wikipedia:

https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches

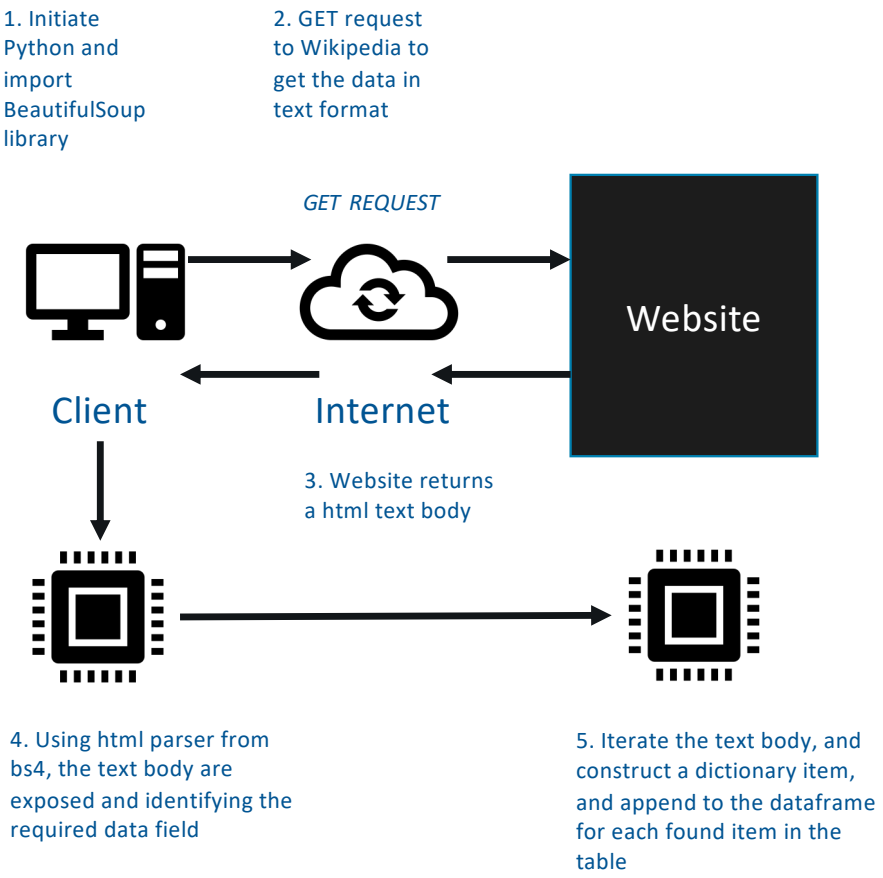
Wikipedia is also another good source of information.

Libraries used in python: **BeautifulSoup**

Historical launch data are crawled from Wikipedia through the use of BS4.

A html text body is returned.

The data is later structured into a dataframe by iterating the body of text, identify the corresponding indexes to the data required and parsing it to a dataframe.

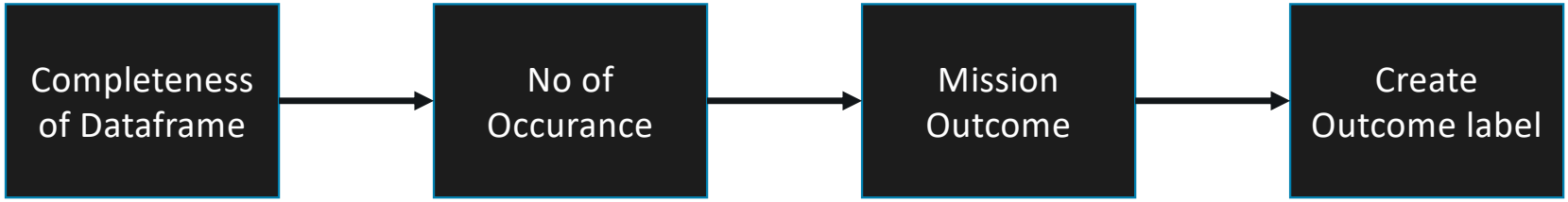


Github url : *https://github.com/jidokasg/IBM-Watson-Studio-captstone/blob/master/capstone-week1-bs4.ipynb*

Data Wrangling

Data wrangling

- Data Wrangling is conducted next, to identify what are the data fields available, the shape and size, data completeness and determining the the training labels for the upcoming model.
- After understanding the data, a target variable / label is created for the outcome of the launches for onward data modeling.



40.63% of the [Landing Pad] column are null values.

The rest of the data fields are complete with data.

Columns are also identified whether they are numerical or categorical

The count of launches at each Launch Site are calculated to identify the distribution of data

The number and occurrence of each orbit are also calculated

There are various permutations of what constitute to a unsuccessful launch outcome.

The categorical values of these bad outcomes are identified as 'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'.

The objective is to predict whether the launch is successful or not successful, therefore an outcome label 'Class' is created in binary terms, 1 or 0.

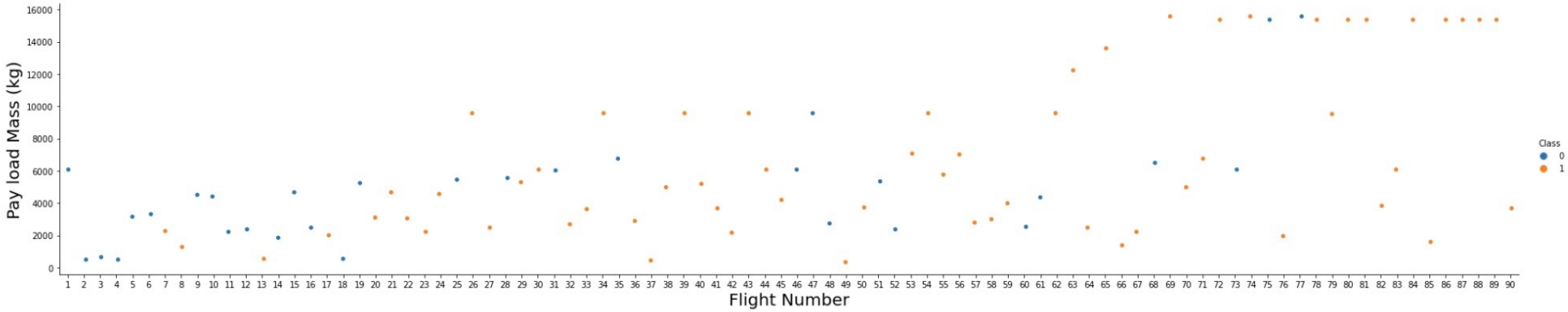
Those with bad_outcomes are labeled with 0 and successful launches as 1.

Github url : https://github.com/jidokasg/IBM-Watson-Studio-captstone/blob/master/capstone-week1-eda.ipynb

EDA with Visualization

Payload vs. Launch Site

Scatter plot : Flight number and Payload relationship with launch outcome, where [Class] 0 = unsuccessful and [Class] 1 = successful



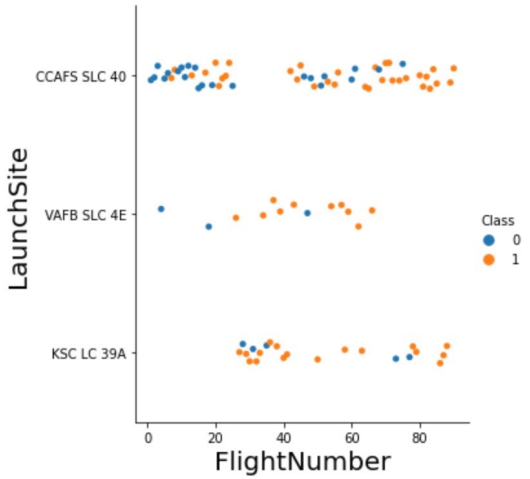
A scatter plot is used to identify if there're any relationship between the flight number, payload against the successful (or unsuccessful) outcome of a first stage launch.

As the number of flights increases, the first stage has a higher likelihood to land successfully.
it also seems the more massive the payload, the less likely the first stage will return.

Github url : https://github.com/jidokasg/IBM-Watson-Studio-captstone/blob/master/captstone-week2-visualization.ipynb

Flight Number vs. Launch Site

Scatter plot : Flight number and Launch Site



The success rate differs across sites. CCAFS SLC 40 has a success rate of 60%* and as the number of flights increases, the more likely the the first stage launch is successful.

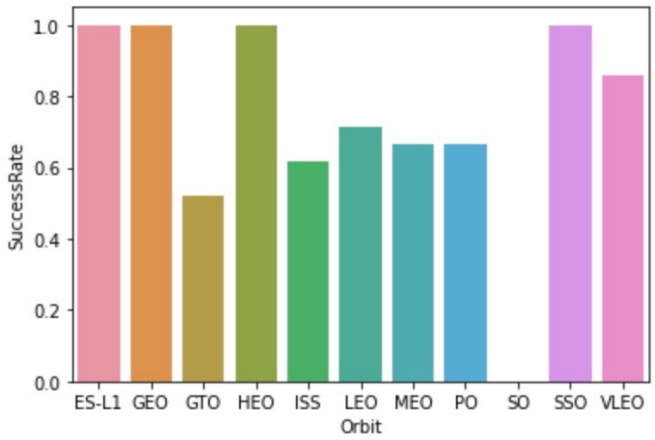
CAFB SLC 4E and KSC LC 39A has a higher success rate of 77%*.

Github url : https://github.com/jidokasg/IBM-Watson-Studio-captstone/blob/master/captstone-week2-visualization.ipynb

**Refer to Snip 1 in Appendices - Annex A*

Success rate vs. Orbit type

Bar Chart : Success Rate and Orbit



The launches at ES-LI, GEO, HEO and SSO** sees the highest success rate, while GTO has the lowest success rate.

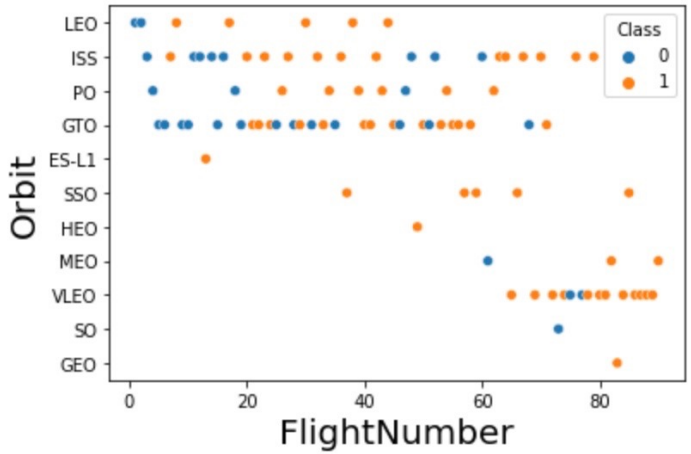
Next we will dive deeper into the specific launches to identify any patterns.

Github url : https://github.com/jidokasg/IBM-Watson-Studio-captstone/blob/master/captstone-week2-visualization.ipynb

***Refer to Appendices - Annex B : Orbit Types*

Flight Number vs. Orbit type

Scatter plot : Flight number and Orbit



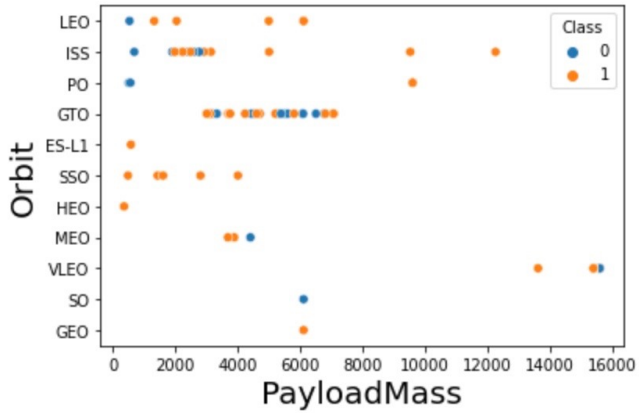
In the LEO orbit, the success rate appears related to the number of flights, whereas for GTO, such relationship is not present

For ES-L1, GEO, HEO, there're only 1 flight each, thus the high success rate may not be a good reflection until there is a larger sample size

Github url : https://github.com/jidokasg/IBM-Watson-Studio-captstone/blob/master/captstone-week2-visualization.ipynb

Payload vs. Orbit type

Scatter plot : Payload and Orbit



There is also positive influence of heavier payloads to success rate for LEO and ISS orbits.

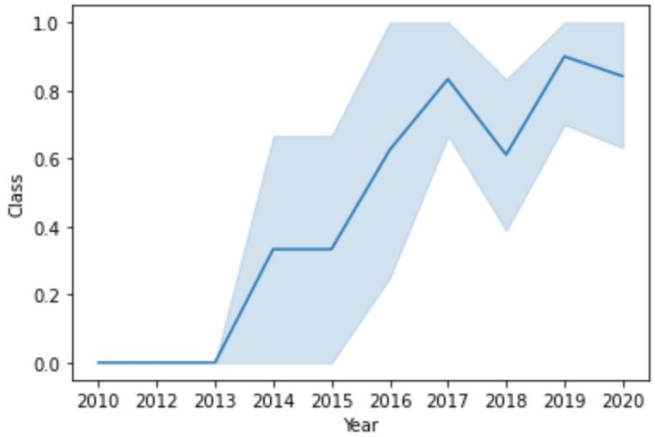
In comparison, heavier payloads appears to have negative influence on GTO orbit.

SSO Orbits have 5 launches at lower payload mass, with all launches being successful.

Github url : https://github.com/jidokasg/IBM-Watson-Studio-captstone/blob/master/captstone-week2-visualization.ipynb

Launch success yearly trend

Line Chart : Trend of Success rate by year



Starting from 2013, there is an improvement in the success rate and continue to move in an upward trend till 2020.

This can be expected when more knowledge and insights are gained from each launches, making subsequent launches a higher probability of succeeding.

Github url : https://github.com/jidokasg/IBM-Watson-Studio-captstone/blob/master/captstone-week2-visualization.ipynb

Data Preparation

Preparation of data for modeling:

Feature Engineering:

The following data fields are identified as useful variables for predicting the outcome of a first stage launch success rate:

FlightNumber, PayloadMass, Orbit, LaunchSite, Flights, GridFins, Reused, Legs, LandingPad, Block, ReusedCount, Serial

One Hot Encoding:

Non binary categorical data fields – Orbit, LaunchSite, LandingPad, Serial converted to numeric by creating dummy variables via one hot encoding.

For consistency of the numeric values, all the numeric values will be cast as float data type.

Github url : https://github.com/jidokasg/IBM-Watson-Studio-captstone/blob/master/captstone-week2-visualization.ipynb

EDA with SQL

All launch site names

- SELECT DISTINCT(launch_site) FROM SPACEXTBL

- Using DISTINCT function to get the names of the different launch sites from the data table. This will return the unique values of column launch site.

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

```
In [7]: %sql SELECT DISTINCT(launch_site) FROM SPACEXTBL
```

```
 * ibm_db_sa://wjb48437:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

Out[7]:

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

Github url : https://github.com/jidokasg/IBM-Watson-Studio-captstone/blob/master/capstone-week2-edasql.ipynb

Launch site names begin with `CCA`

- SELECT * FROM SPACEXTBL WHERE launch_site like 'CCA%' LIMIT 5

- Query the sites that starts with the text ‘CCA’ using wildcard and LIKE function. Thereafter return only 5 results using the LIMIT function

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

In [14]:
```
%sql SELECT * FROM SPACEXTBL WHERE launch_site like 'CCA%' LIMIT 5
```

```
 * ibm_db_sa://wjb48437:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

Out[14]:

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	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)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Github url : https://github.com/jidokasg/IBM-Watson-Studio-captstone/blob/master/capstone-week2-edasql.ipynb

Total payload mass

- SELECT SUM(payload_mass__kg_) FROM SPACEXTBL WHERE customer = 'NASA (CRS)’
- Sums up all the values in payload_mass__kg_ where the customer name is ’NASA (CRS)’

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

```
In [26]: %sql SELECT SUM(payload_mass__kg_) FROM SPACEXTBL WHERE customer = 'NASA (CRS)'
 * ibm_db_sa://wjb48437:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu01qde00.databases.appdomain.cloud:31321/bludb
Done.
Out[26]:
```

1
45596

Github url : https://github.com/jidokasg/IBM-Watson-Studio-captstone/blob/master/capstone-week2-edasql.ipynb

Average payload mass by F9 v1.1

- SELECT AVG(payload_mass__kg_) FROM SPACEXTBL WHERE booster_version like 'F9 v1.1%'

- Aggregate the values in payload_mass__kg_ where the booster version starts with F9 v1.1. Using wildcard and LIKE function to query all associated data

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

```
In [27]: %sql SELECT AVG(payload_mass__kg_) FROM SPACEXTBL WHERE booster_version like 'F9 v1.1%'
 * ibm_db_sa://wjb48437:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
Out[27]:
```

1
2534

Github url : https://github.com/jidokasg/IBM-Watson-Studio-captstone/blob/master/capstone-week2-edasql.ipynb

First successful ground landing date

- SELECT MIN(DATE) FROM SPACEXTBL WHERE landing__outcome = 'Success (ground pad)’
- Filtering all records where landing__outcome is ‘Success (ground pad), and finding the earliest date with MIN function on the date field

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

```
In [29]: %sql SELECT MIN(DATE) FROM SPACEXTBL WHERE landing__outcome = 'Success (ground pad)'
 * ibm_db_sa://wjb48437:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu01qde00.databases.appdomain.cloud:31321/bludb
Done.
```

Out[29]:

1
2015-12-22

Github url : https://github.com/jidokasg/IBM-Watson-Studio-captstone/blob/master/capstone-week2-edasql.ipynb

Successful drone ship landing with payload between 4000 and 6000

- SELECT DISTINCT(booster_version) FROM SPACEXTBL WHERE landing__outcome = 'Success (drone ship)' AND (payload_mass__kg_ BETWEEN 4000 AND 6000)
- Query all distinct booster version that meets the criteria of landing__outcome as ‘Success (drone ship)’ with a payload between 4000 and 6000

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

In [32]:
```
%sql sELECT DISTINCT(booster_version) FROM SPACEXTBL WHERE landing__outcome = 'Success (drone ship)' AND (payload_mass__kg_ BE
TWEEN 4000 AND 6000)
```

```
 * ibm_db_sa://wjb48437:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

Out[32]:

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

Github url : https://github.com/jidokasg/IBM-Watson-Studio-captstone/blob/master/capstone-week2-edasql.ipynb

Total number of successful and failure mission outcomes

- SELECT mission_outcome ,COUNT(mission_outcome) FROM SPACEXTBL GROUP BY mission_outcome

- Count the number of occurrence by misson_outcome (i.e. success and failure) by using GROUP BY to aggregate the data (akin Pivot Tbl)

List the total number of successful and failure mission outcomes

In [41]:
```
%sql SELECT mission_outcome ,COUNT(mission_outcome) FROM SPACEXTBL GROUP BY mission_outcome
```

```
 * ibm_db_sa://wjb48437:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

Out[41]:

mission_outcome	2
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

Github url : https://github.com/jidokasg/IBM-Watson-Studio-captstone/blob/master/capstone-week2-edasql.ipynb

Boosters carried maximum **payload**

- SELECT DISTINCT(booster_version) FROM SPACEXTBL WHERE payload_mass__kg_ = (SELECT MAX(payload_mass__kg_) FROM SPACEXTBL)

- Perform a sub-query to get the max payload mass, and then using that to find the booster version with the maximum payload found

List the names of the booste

```
In [46]: %sql SELECT DISTINCT(b
          * ibm_db_sa://wjb4843
         Done.
```

Out[46]:

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

Github url : https://github.com/jidokasg/IBM-Watson-Studio-captstone/blob/master/capstone-week2-edasql.ipynb

2015 launch records

- SELECT landing__outcome, booster_version, launch_site from SPACEXTBL where landing__outcome like '%Fail%' AND YEAR(DATE) = '2015’
- Querying the booster version, launch site and landing outcome in 2015 by filtering the date and if the landing__outcome contains the word ‘Fail’ using LIKE and Wildcard(%)

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for the in year 2015

In [52]:
```
%sql SELECT landing__outcome, booster_version, launch_site from SPACEXTBL where landing__outcome like '%Fail%' AND YEAR(DATE)
= '2015'
```

```
 * ibm_db_sa://wjb48437:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

Out[52]:

landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

Github url : https://github.com/jidokasg/IBM-Watson-Studio-captstone/blob/master/capstone-week2-edasql.ipynb

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

- SELECT landing__outcome, COUNT(*), RANK() OVER (Order BY COUNT(*) DESC) as ranking FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY landing__outcome

- Rank landing outcomes by highest count to lowest count (using DESC) between 4th June 2010 and 20th March 2017.

Rank the count of landing outcomes (suc order

```
In [61]: %sql SELECT landing__outcome, COU
         0-06-04' AND '2017-03-20' GROUP B
  * ibm_db_sa://wjb48437:***@ba99a
Done.
```

Out[61]:

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

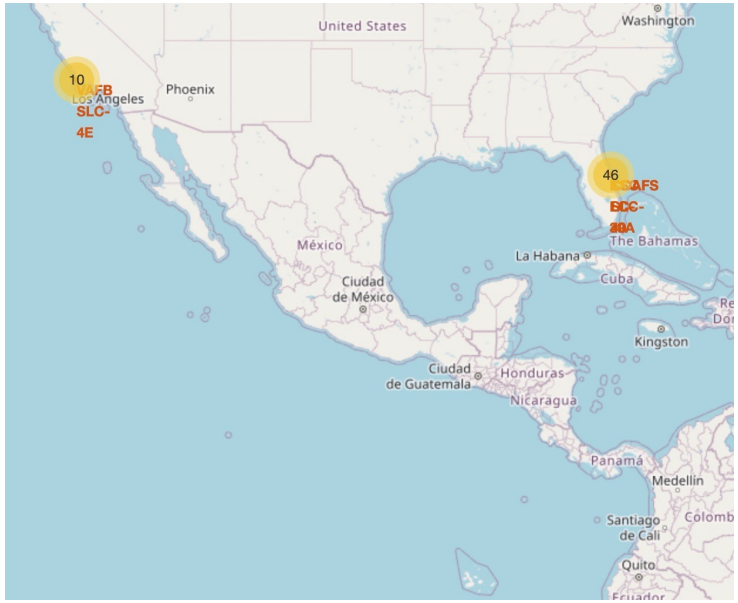
Github url : https://github.com/jidokasg/IBM-Watson-Studio-captstone/blob/master/capstone-week2-edasql.ipynb

Interactive map with Folium

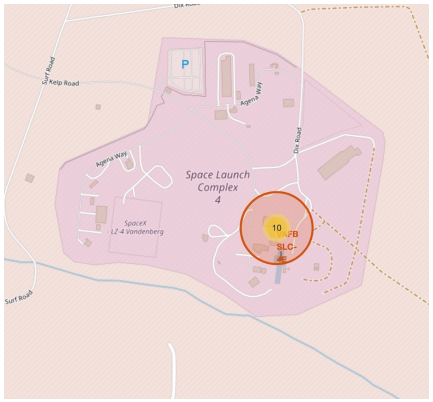
Launch Sites

With the coordinates of the 4 launch sites, the location are plotted on the map using Folium.
The circle markers are used to display the location and the marker label to show the site name.
It appears that the launch sites are situated very near coastal areas.

Global View of all launch sites

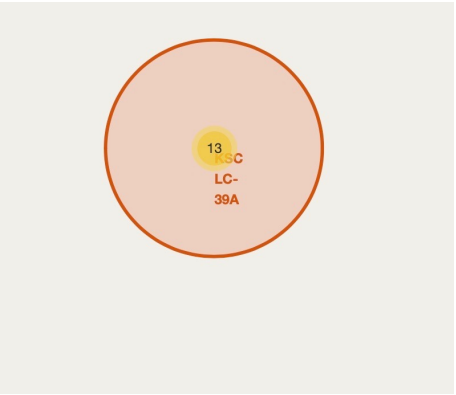


Site View 1 – VAFB SLC-4E



Site View 2 – CCAFS SLC-40 &
CCAFS LC-40

Site View 3 – KSC LC-39A



Github url : https://github.com/jidokasg/IBM-Watson-Studio-captstone/blob/master/capstone-week3-folium.ipynb

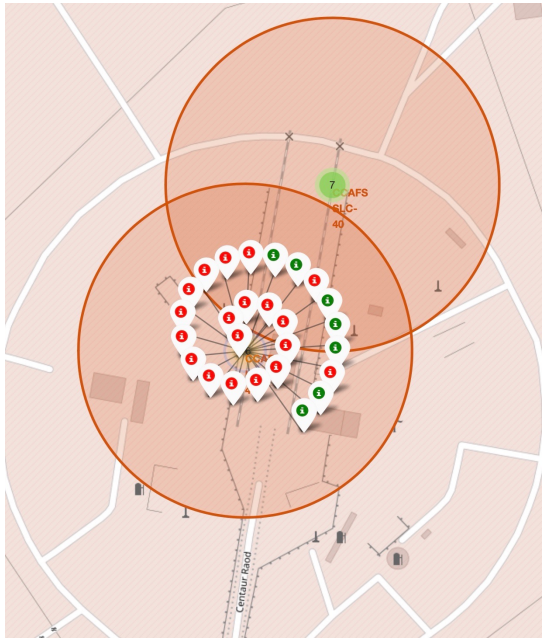
View at IBM Watson Studio : https://eu-de.dataplatform.cloud.ibm.com/analytics/notebooks/v2/16a446e3-8ff7-49c6-8038-db357315df46/view?access_token=7c1101932ce9442df954da2db1f6cfb7b28963122707a1a86dbe8475a6e4bbd9

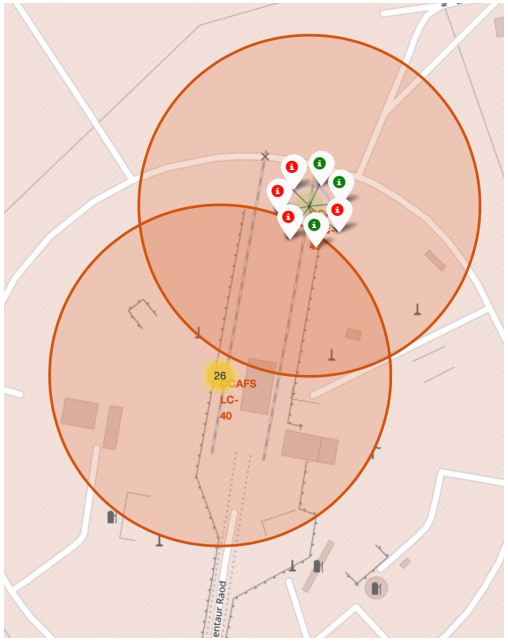
Success Outcome at each launch sites

A color coding field is created for the outcome (‘class’) of the launch, with **green** being successful and **red** being unsuccessful.
This will allow us to use the marker_color data field as a variable for the map creation instead of manually specifying the color in each element creation.

A marker cluster is added to each launch site, reflected the total number of flights made, and the outcome of these launches.

	Launch Site	Lat	Long	class	marker_color
46	KSC LC-39A	28.573255	-80.646895	1	green
47	KSC LC-39A	28.573255	-80.646895	1	green
48	KSC LC-39A	28.573255	-80.646895	1	green
49	CCAFS SLC-40	28.563197	-80.576820	1	green
50	CCAFS SLC-40	28.563197	-80.576820	1	green
51	CCAFS SLC-40	28.563197	-80.576820	0	red
52	CCAFS SLC-40	28.563197	-80.576820	0	red
53	CCAFS SLC-40	28.563197	-80.576820	0	red
54	CCAFS SLC-40	28.563197	-80.576820	1	green
55	CCAFS SLC-40	28.563197	-80.576820	0	red





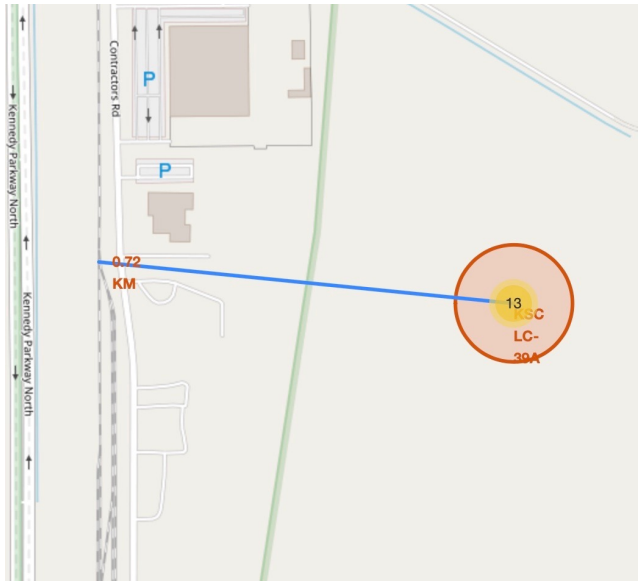
Github url : https://github.com/jidokasg/IBM-Watson-Studio-captstone/blob/master/capstone-week3-folium.ipynb

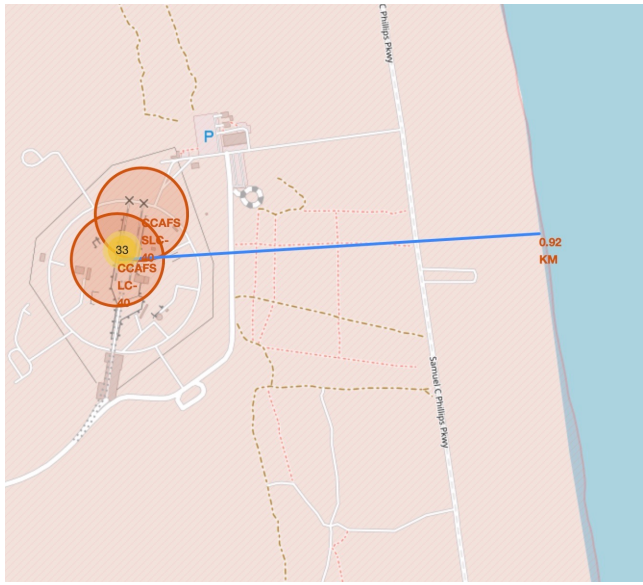
View at IBM Watson Studio : https://eu-de.dataplatform.cloud.ibm.com/analytics/notebooks/v2/16a446e3-8ff7-49c6-8038-db357315df46/view?access_token=7c1101932ce9442df954da2db1f6cfb7b28963122707a1a86dbe8475a6e4bbd9

Geographical insights of each location

Polylines are created based on 2 provided coordinates to show the distance between different point of interest.
Apart from KSC LC-39A which is pretty near railway (within 1km distance), the remaining 3 sites are near to coastal area approximately from 1 to 1.5km away.

It is likely due to the fact that rocket launches are a great noise and air pollution, and any failure in the launch would result in impact to the proximity, thus located far from city district.



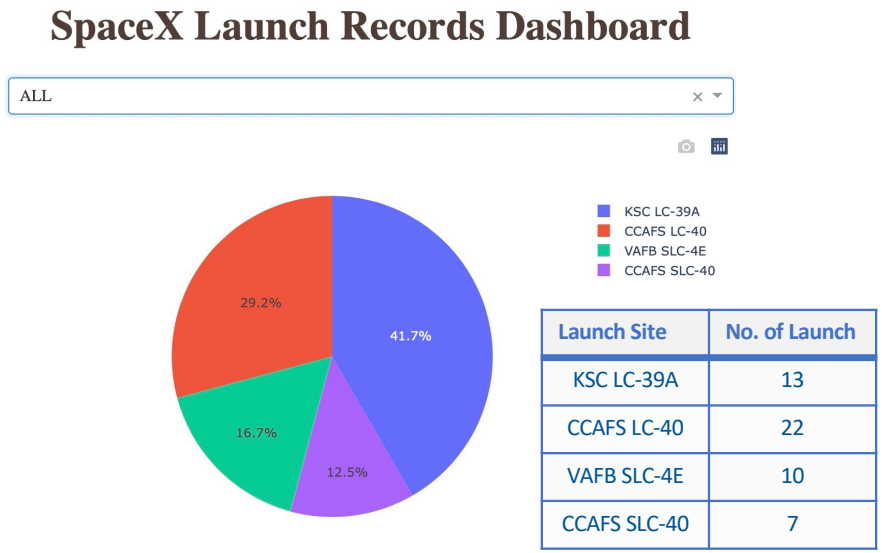


Github url : https://github.com/jidokasg/IBM-Watson-Studio-captstone/blob/master/capstone-week3-folium.ipynb

View at IBM Watson Studio : https://eu-de.dataplatform.cloud.ibm.com/analytics/notebooks/v2/16a446e3-8ff7-49c6-8038-db357315df46/view?access_token=7c1101932ce9442df954da2db1f6cfb7b28963122707a1a86dbe8475a6e4bbd9

Build a Dashboard with Plotly Dash

Success rate of launch sites

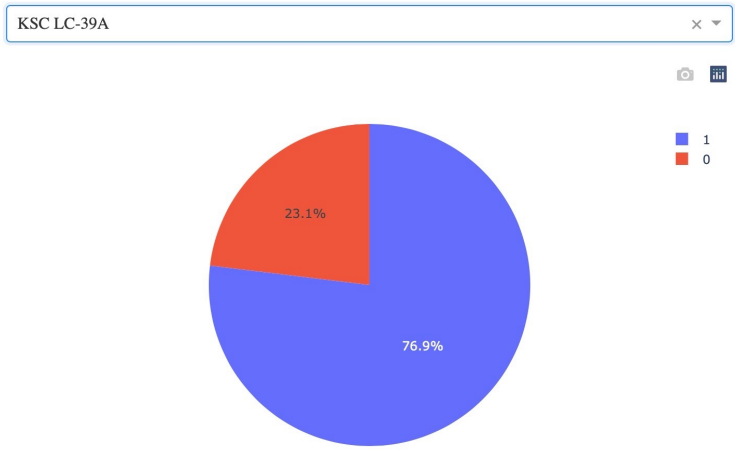


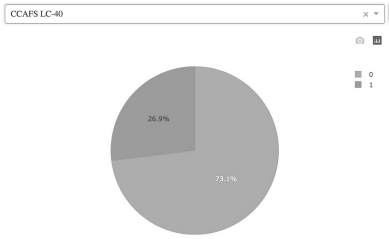
Launch Site	No. of Launch
KSC LC-39A	13
CCAFS LC-40	22
VAFB SLC-4E	10
CCAFS SLC-40	7

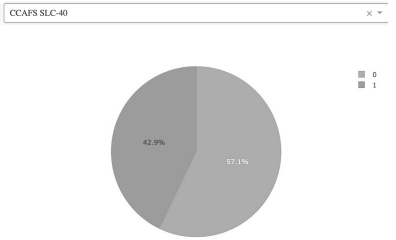
The above pie chart shows the ratio of successful launches at different sites
It is noted that KSC LC-39A has the highest number of success launches.

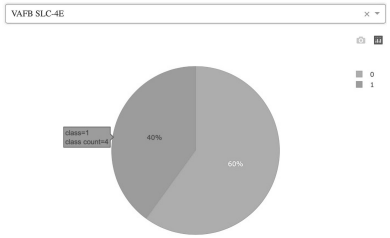
Next we will look deeper into each of the sites to see the success rate of each sites and whether the success rate are a good representation

Launch Site with highest success rate







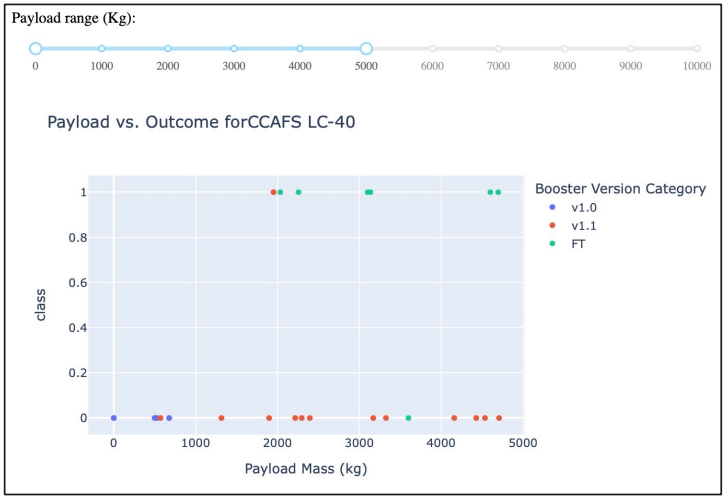


Launch Site	No. of Launch
KSC LC-39A	13
CCAFS LC-40	22
VAFB SLC-4E	10
CCAFS SLC-40	7

Looking at each launch site, only KSC KC-39A has a higher success rate at 76.9%, compared to the rest of the sites where the success rate are below 50%.

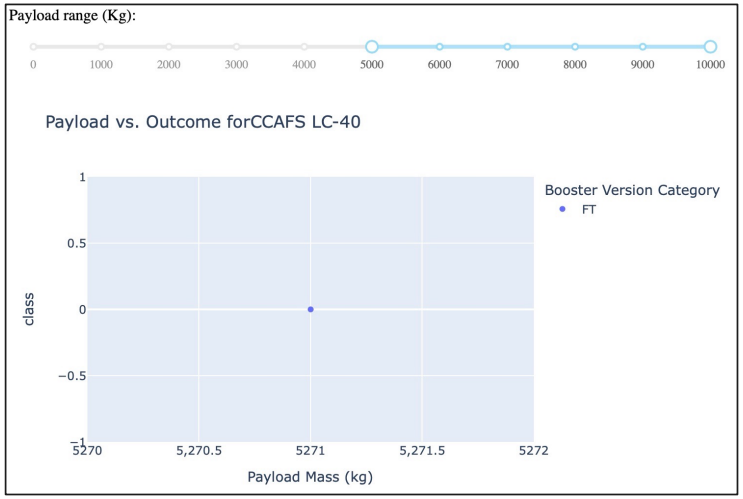
KSC KC-39A also has a decent number of launches (13) which further supports a good success rate for KSC LC-39A

Payload vs Launch Outcome



Most of the launches at site CCAFS LC-40 are within the payload mass of 5000kg.

As seen from the scatter plot above, there is no influence on the payload against the success rate of the launch.



There's only 1 launch for payload mass above 5000kg.

The outcome of the launch is unsuccessful as well.

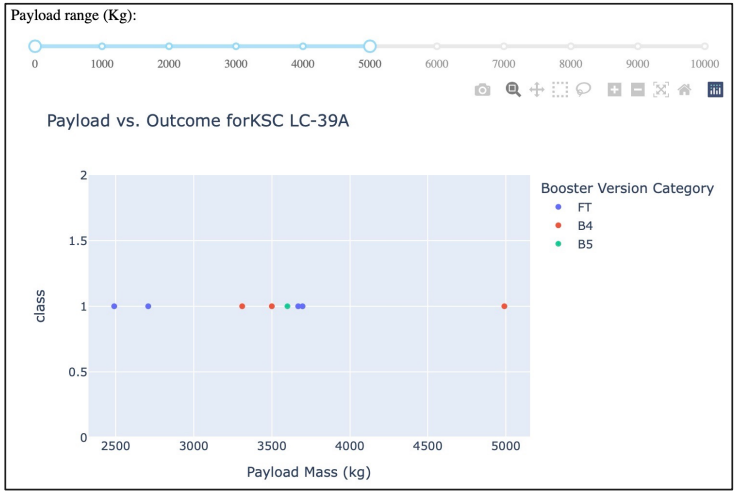
Payload vs Launch Outcome

Looking at VAFB SLC-4E, there are only 4 launches and only the lowest payload launch was successful.

Similarly for a larger payload, we see that the payload above 9000kg also had 1 successful launch.

The findings above also show no relationship between the payload and the launch outcome.

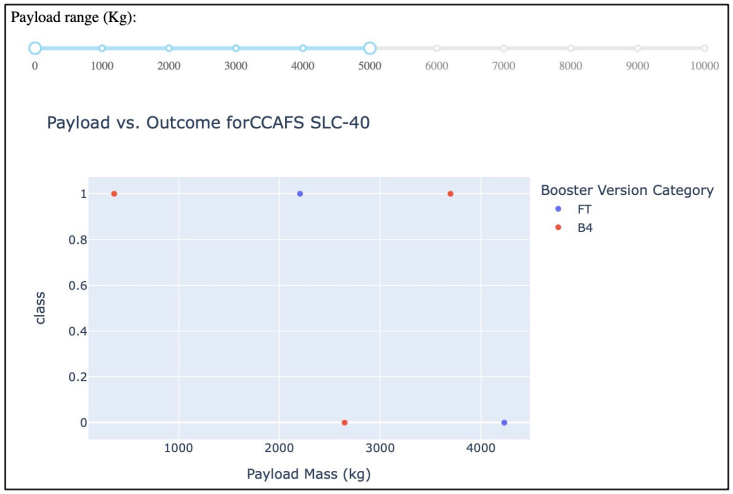
Payload vs Launch Outcome



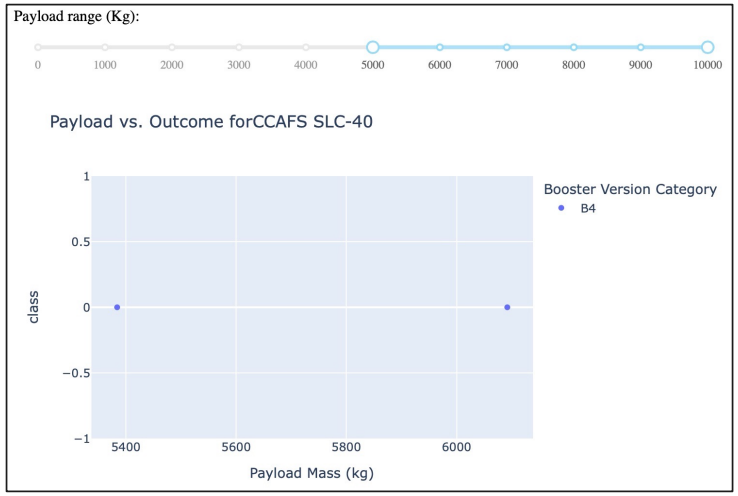
For the highest success rate launch site at KSC LC-39A, we see that all the launches within 5000kg payload are successful.

For payload above 5000kg, it appears that higher payload as a negative influence to the mission outcome.

Payload vs Launch Outcome



Similarly for CCAFS SLC-40, it appears that the payload has a negative impact to the mission outcome, particularly for Booster Version FT.

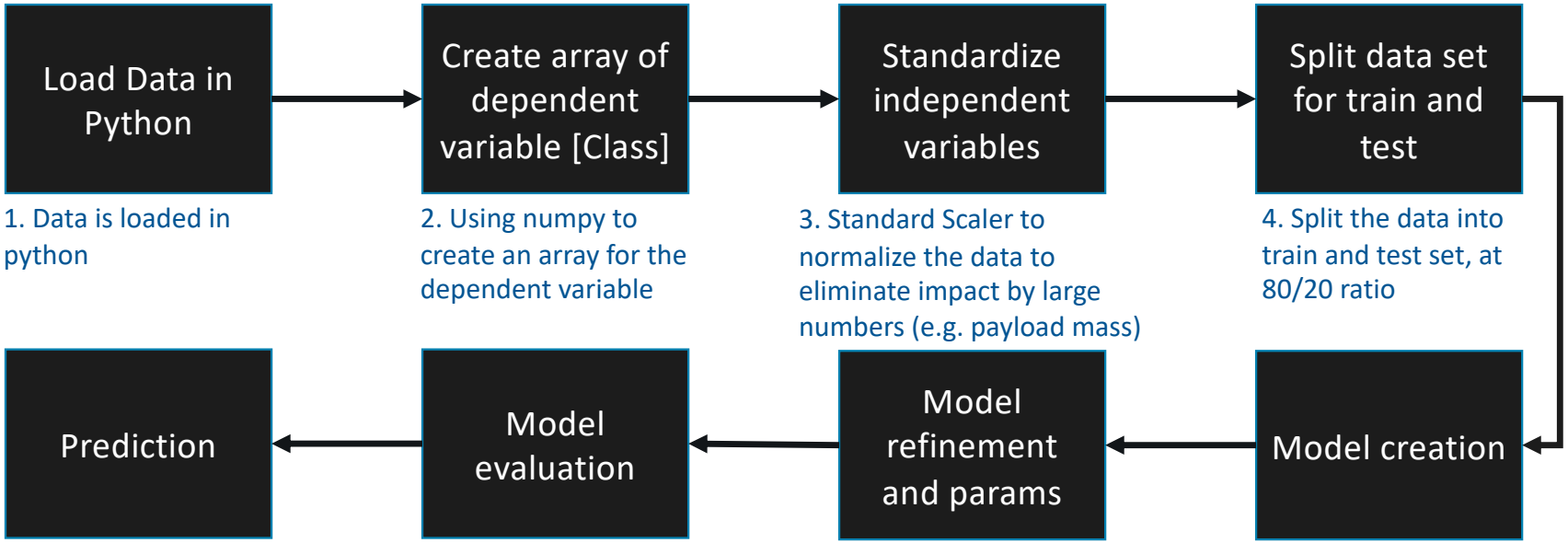


For higher payload above 5000kg, the mission outcome were unsuccessful as shown in the scatter plot.

Predictive analysis (Classification)

Predictive Analysis Classification (Workflow)

The dataset is prepared and ready to be loaded into Python for creation of classification model, finding the best parameters, model evaluation and performing the prediction.

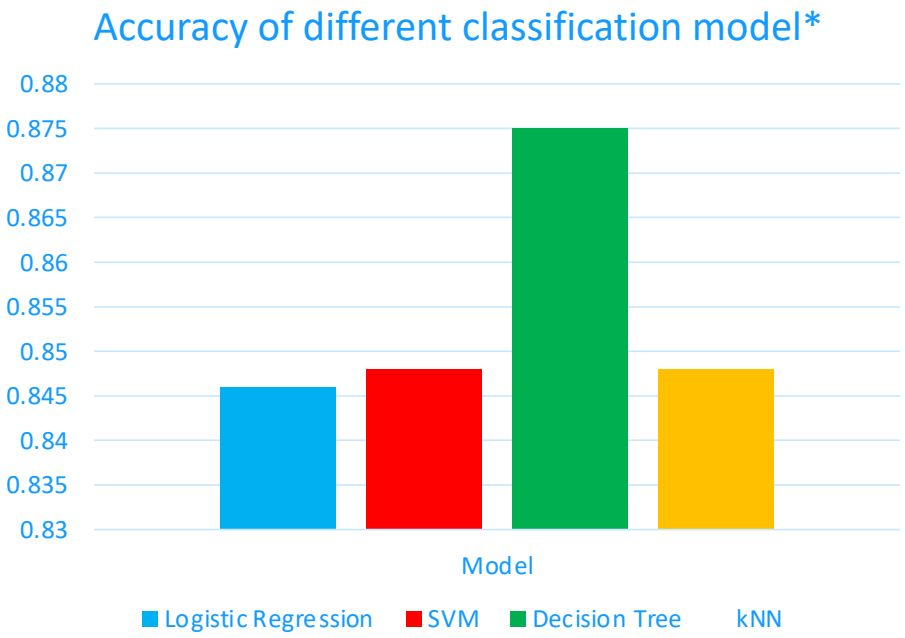


Classification Accuracy

By using a GridSearch across all the models, the best parameters identified and to create the model using the prepared dataset

As we can see from the bar chart on the left, Decision Tree is the best performing model with the highest accuracy at 87.5%

Next, we shall look into the confusion matrix of the Decision Tree model for the prediction.



Github url : https://github.com/jidokasg/IBM-Watson-Studio-captstone/blob/master/capstone-week4-predictive.ipynb

**Refer to Appendices - Annex C : Model Evaluations*

Confusion Matrix

We can see that the model predicted the True positives (landed and land) accurately.

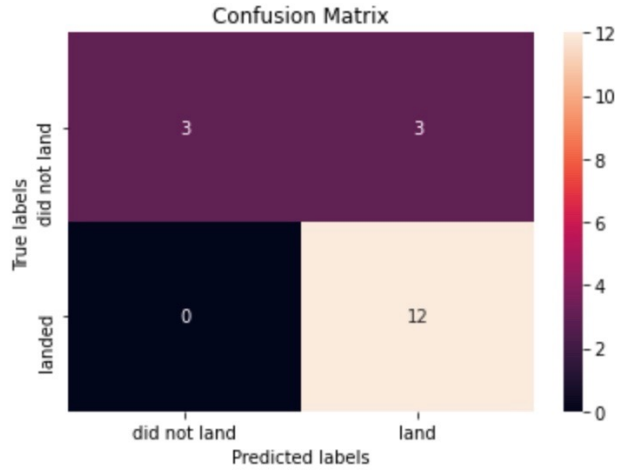
However, for negative outcome prediction, we see a False Positive of 50%.

In this scenario, although with a good prediction accuracy, the impact of False Positive will incur additional cost to the company may under budget based on the results.

```
In [54]: tree_cv.score(X_test, Y_test)
Out[54]: 0.8333333333333334
```

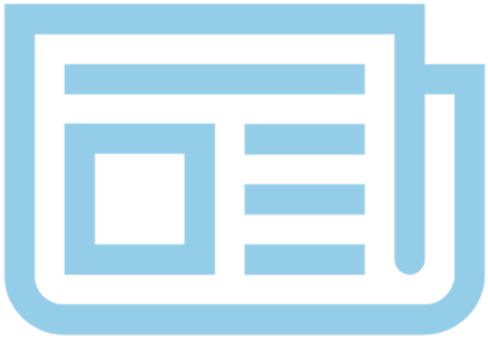
We can plot the confusion matrix

```
In [55]: yhat = svm_cv.predict(X_test)
         plot_confusion_matrix(Y_test,yhat)
```



Github url : https://github.com/jidokasg/IBM-Watson-Studio-captstone/blob/master/capstone-week4-predictive.ipynb

Results



- Through exploratory data analysis, we are able to identify trends of the success landing rate of the launch. Across the years, we saw an upward trend on the success of the first stage launch outcome. We also saw the influences of the orbit and payload to the mission outcome.
- Through visualization, we are also able to get a geographical view of the launch sites and their activities. Most launch sites are close to costal areas away from the city central
- 4 Classification models were tested. Of the models tested, Decision Tree was the best performing model.

Conclusion

- We see higher likelihood of successful landing of the first stage launch. This can be attributed to getting more knowledge and experiences as number of flights increases.
- It also appears that launching to certain orbits grants a higher success rate of it landing back successfully.
- Through visualization, we also gain more knowledge on the geographical location of each of the launch sites, which are all away from the city central and near to coastal area.
- We find the Decision Tree to be the best performing model to predict the successful landing of the launch, at an accuracy rate of 75%.
- However, the high accuracy is largely based on the True negative, and there is an error rate of 50% for False Positive. This error will end up incurring most cost for the organization.
- In ANNEX D, we saw from the data set that there are other features such as Serial Number which may not have any effect on the results.
- The process can be reiterated by calculating the feature importance in the dataset and re-training the model for higher accuracy. This is due to the fact that Decision Tree is largely based on the biasness of the data, and with the small dataset available, any data bias will affect the accuracy of the model significantly

Appendix

APPENDIX

ANNEX A - Python Snippets

Snip 1: Average success rate by Launch Site

```
df_ccafs = df[df['LaunchSite'] == 'CCAFS SLC 40']['Class'].mean()
print(df_ccafs)

df_vafb = df[df['LaunchSite'] == 'VAFB SLC 4E']['Class'].mean()
print(df_vafb)

df_ksc = df[df['LaunchSite'] == 'KSC LC 39A']['Class'].mean()
print(df_ksc)
```

```
0.6
0.7692307692307693
0.7727272727272727
```

APPENDIX

ANNEX B – Orbit Types

•**LEO**: Low Earth orbit (LEO)is an Earth-centred orbit with an altitude of 2,000 km (1,200 mi) or less (approximately one-third of the radius of Earth),[1] or with at least 11.25 periods per day (an orbital period of 128 minutes or less) and an eccentricity less than 0.25.[2] Most of the manmade objects in outer space are in LEO \[1].
•**VLEO**: Very Low Earth Orbits (VLEO) can be defined as the orbits with a mean altitude below 450 km. Operating in these orbits can provide a number of benefits to Earth observation spacecraft as the spacecraft operates closer to the observation\[2].
•**GTO** A geosynchronous orbit is a high Earth orbit that allows satellites to match Earth's rotation. Located at 22,236 miles (35,786 kilometers) above Earth's equator, this position is a valuable spot for monitoring weather, communications and surveillance. Because the satellite orbits at the same speed that the Earth is turning, the satellite seems to stay in place over a single longitude, though it may drift north to south," NASA wrote on its Earth Observatory website \[3].
•**SSO (or SO)**: It is a Sun-synchronous orbit also called a heliosynchronous orbit is a nearly polar orbit around a planet, in which the satellite passes over any given point of the planet's surface at the same local mean solar time [4] .
•**ES-L1** :At the Lagrange points the gravitational forces of the two large bodies cancel out in such a way that a small object placed in orbit there is in equilibrium relative to the center of mass of the large bodies. L1 is one such point between the sun and the earth \[5] .
•**HEO** A highly elliptical orbit, is an elliptic orbit with high eccentricity, usually referring to one around Earth \[6].
•**ISS** A modular space station (habitable artificial satellite) in low Earth orbit. It is a multinational collaborative project between five participating space agencies: NASA (United States), Roscosmos (Russia), JAXA (Japan), ESA (Europe), and CSA (Canada) \[7]
•**MEO** Geocentric orbits ranging in altitude from 2,000 km (1,200 mi) to just below geosynchronous orbit at 35,786 kilometers (22,236 mi). Also known as an intermediate circular orbit. These are "most commonly at 20,200 kilometers (12,600 mi), or 20,650 kilometers (12,830 mi), with an orbital period of 12 hours \[8]
•**HEO** Geocentric orbits above the altitude of geosynchronous orbit (35,786 km or 22,236 mi) \[9]
•**GEO** It is a circular geosynchronous orbit 35,786 kilometres (22,236 miles) above Earth's equator and following the direction of Earth's rotation \[10]
•**PO** It is one type of satellites in which a satellite passes above or nearly above both poles of the body being orbited (usually a planet such as the Earth \[11]

APPENDIX

ANNEX C – Model Evaluation

Snip 2 – Logistic Regression Python Code

```
In [42]: parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
         lr=LogisticRegression()

         logreg_cv = GridSearchCV(lr, parameters, cv=10)

         logreg_cv.fit(X_train, Y_train)

Out[42]: GridSearchCV(cv=10, estimator=LogisticRegression(),
                      param_grid={'C': [0.01, 0.1, 1], 'penalty': ['l2'],
                                  'solver': ['lbfgs']})
```

We output the `GridSearchCV` object for logistic regression. We display the best parameters using the data attribute `best_params\_` and the accuracy on the validation data using the data attribute `best_score\_`.

```
In [43]: 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
```

Snip 3 – Logistic Regression Confusion Matrix

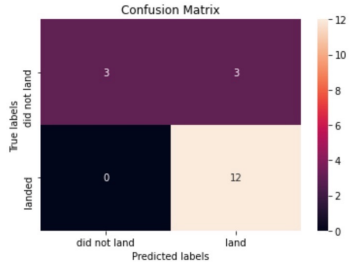
Calculate the accuracy on the test data using the method `score`:

```
In [44]: logreg_cv.score(X_test, Y_test)

Out[44]: 0.8333333333333334
```

Lets look at the confusion matrix:

```
In [45]: yhat=logreg_cv.predict(X_test)
         plot_confusion_matrix(Y_test,yhat)
```



APPENDIX

ANNEX C – Model Evaluation

Snip 5 – Support Vector Machine (SVM) Python Code

```
In [46]: parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
                       'C': np.logspace(-3, 3, 5),
                       'gamma':np.logspace(-3, 3, 5)}
         svm = SVC()

In [47]: svm_cv = GridSearchCV(svm, parameters, cv=10)

         svm_cv.fit(X_train, Y_train)

Out[47]: GridSearchCV(cv=10, estimator=SVC(),
                      param_grid={'C': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.16227766e+01,
                1.00000000e+03]),
                                  'gamma': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.16227766e+01,
                1.00000000e+03]),
                                  'kernel': ('linear', 'rbf', 'poly', 'rbf', 'sigmoid')})

In [48]: 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
```

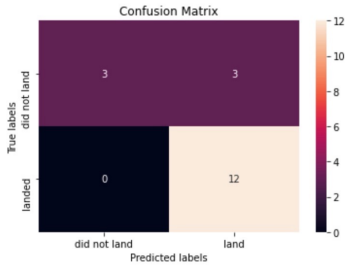
Snip 6 – SVM Confusion Matrix

```
In [49]: svm_cv.score(X_test, Y_test)

Out[49]: 0.8333333333333334
```

We can plot the confusion matrix

```
In [50]: yhat=svm_cv.predict(X_test)
         plot_confusion_matrix(Y_test,yhat)
```



APPENDIX

ANNEX C – Model Evaluation

Snip 7 – Decision Tree Python Code

```
In [51]: parameters = {'criterion': ['gini', 'entropy'],
              'splitter': ['best', 'random'],
              'max_depth': [2*n for n in range(1,10)],
              'max_features': ['auto', 'sqrt'],
              'min_samples_leaf': [1, 2, 4],
              'min_samples_split': [2, 5, 10]}

         tree = DecisionTreeClassifier()

In [52]: tree_cv = GridSearchCV(tree, parameters, cv=10)

         tree_cv.fit(X_train, Y_train)

Out[52]: GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
                       param_grid={'criterion': ['gini', 'entropy'],
                                   'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18],
                                   'max_features': ['auto', 'sqrt'],
                                   'min_samples_leaf': [1, 2, 4],
                                   'min_samples_split': [2, 5, 10],
                                   'splitter': ['best', 'random']})

In [53]: print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
         print("accuracy :",tree_cv.best_score_)

         tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 4, 'max_features': 'auto', 'min_samples_
         leaf': 4, 'min_samples_split': 10, 'splitter': 'random'}
         accuracy : 0.875
```

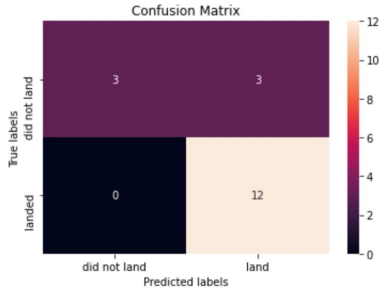
Snip 8 – Logistic Regression Confusion Matrix

```
In [54]: tree_cv.score(X_test, Y_test)

Out[54]: 0.8333333333333334
```

We can plot the confusion matrix

```
In [55]: yhat = svm_cv.predict(X_test)
         plot_confusion_matrix(Y_test,yhat)
```



APPENDIX

ANNEX C – Model Evaluation

Snip 9 – Logistic Regression Python Code

```
In [56]: parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                       'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
                       'p': [1,2]}

         KNN = KNeighborsClassifier()

In [57]: knn_cv = GridSearchCV(KNN, parameters, cv=10)

         knn_cv.fit(X_train, Y_train)

Out[57]: GridSearchCV(cv=10, estimator=KNeighborsClassifier(),
                      param_grid={'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
                                  'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                                  'p': [1, 2]})

In [58]: 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
```

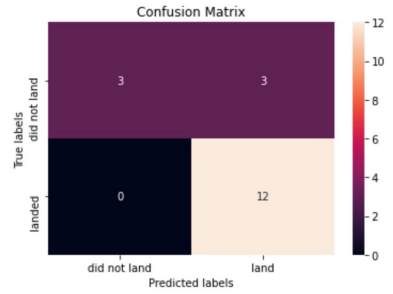
Snip 10 – Logistic Regression Confusion Matrix

```
In [59]: knn_cv.score(X_test, Y_test)

Out[59]: 0.8333333333333334
```

We can plot the confusion matrix

```
In [60]: yhat = knn_cv.predict(X_test)
         plot_confusion_matrix(Y_test,yhat)
```



APPENDIX

ANNEX D – Data Features

```
In [35]: #X = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain

         # If you were unable to complete the previous lab correctly you can uncomment

         dftest = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdo

         dftest.info()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 90 entries, 0 to 89
         Data columns (total 83 columns):
          #   Column                                Non-Null Count  Dtype
         ---  ------                                --------------  -----
          0   FlightNumber                          90 non-null     float64
          1   PayloadMass                           90 non-null     float64
          2   Flights                               90 non-null     float64
          3   Block                                 90 non-null     float64
          4   ReusedCount                           90 non-null     float64
          5   Orbit_ES-L1                           90 non-null     float64
          6   Orbit_GEO                             90 non-null     float64
          7   Orbit_GTO                             90 non-null     float64
          8   Orbit_HEO                             90 non-null     float64
          9   Orbit_ISS                             90 non-null     float64
          10  Orbit_LEO                             90 non-null     float64
          11  Orbit_MEO                             90 non-null     float64
          12  Orbit_PO                              90 non-null     float64
          13  Orbit_SO                              90 non-null     float64
          14  Orbit_SSO                             90 non-null     float64
          15  Orbit_VLEO                            90 non-null     float64
          16  LaunchSite_CCAFS SLC 40               90 non-null     float64
          17  LaunchSite_KSC LC 39A                 90 non-null     float64
          18  LaunchSite_VAFB SLC 4E                90 non-null     float64
          19  LandingPad_5e9e3032383ecb267a34e7c7    90 non-null     float64
          20  LandingPad_5e9e3032383ecb554034e7c9    90 non-null     float64
          21  LandingPad_5e9e3032383ecb6bb234e7ca    90 non-null     float64
          22  LandingPad_5e9e3032383ecb761634e7cb    90 non-null     float64
          23  LandingPad_5e9e3033383ecbb9e534e7cc    90 non-null     float64
          24  Serial_B0003                          90 non-null     float64
          25  Serial_B0005                          90 non-null     float64
          26  Serial_B0007                          90 non-null     float64
          27  Serial_B1003                          90 non-null     float64
          28  Serial_B1004                          90 non-null     float64
          29  Serial_B1005                          90 non-null     float64
          30  Serial_B1006                          90 non-null     float64
          31  Serial_B1007                          90 non-null     float64
          32  Serial_B1008                          90 non-null     float64
          33  Serial_B1010                          90 non-null     float64
          34  Serial_B1011                          90 non-null     float64
          35  Serial_B1012                          90 non-null     float64
```