IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Wendhy Novizar Rigo
July 29th 2023

Outline

- Executive Summary
- Introduction
- Methodology
- Results
  - Visualization – Charts
  - Dashboard
- Discussion
  - Findings & Implications
- Conclusion
- Appendix

Executive Summary

- Is Space Travel Possible and Profitable?
- Using Data Science to Win Space Travel
  - Detailed Review of Predictive Analysis
  - Based on Falcon 9 SpaceX Launch Dataset
  - Machine Learning
- Methods Used
  - Data Collection and Wrangling with Python
  - Exploratory Data Analysis with SQL
  - Data Visualization with Folium
- Machine Learning with Predictive Analysis
  - Determining the Failure Rate
  - Using Logistic Regression, RBF, and Sigmoid
- Results
  - Exploratory Data Analysis
  - Interactive Map
  - Predictive Analysis

Introduction

- Background
  - Possibility of Space Travel
  - Profitability of Space Travel
  - Failure Rate of Space Travel
- Problem-identifying Questions
  - What factors determines a successful launch
  - Are interactions amongst features could determine a successful launch
  - What operation conditions needs to be in place for a successful launch
-

Section 1

Methodology

Methodology

- Data Collection
  - Connect to SpaceX API
  - Web Scraping from Wikipedia
- Data Wrangling
  - One-hot encoding for each category
- Exploratory Data Analysis
  - SQL
  - Python
- Interactive Visuals with Folium & Plotly
- Machine Learning
  - Predictive Analytics using Classification Models

Data Collection

- Various methods of Data Collection
  - Get Request function to the SpaceX API
  - Normalize the Data with json_normalize and convert to Python Dataframe
  - Data Cleaning by checking missing values and fill in the missing values
  - Web scraping from Wikipedia Falcon 9 Launch records with BeautifulSoup
  - Extract scrapped launch records to HTML table, to be parsed and converted to Python Dataframe

Data Collection – SpaceX API

- We used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.
- The link to the notebook is here

1. Get request for rocket launch data using API

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

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

2. Use json_normalize method to convert json result to dataframe

In [12]:
```
# Use json_normalize method to convert the json result into a dataframe

# decode response content as json
static_json_df = res.json()
```

In [13]:
```
# apply json_normalize
data = pd.json_normalize(static_json_df)
```

3. We then performed data cleaning and filling in the missing values

In [30]:
```
rows = data_falcon9['PayloadMass'].values.tolist()[0]

df_rows = pd.DataFrame(rows)
df_rows = df_rows.replace(np.nan, PayloadMass)

data_falcon9['PayloadMass'][0] = df_rows.values
data_falcon9
```

Data Collection - Web Scraping

- Web scrapping Wikipedia Falcon 9 Launch records
- Using BeautifulSoup and html.parser
- Download the notebook here

1. Apply HTTP Get method to request the Falcon 9 rocket launch page

In [4]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

In [5]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
html_data = requests.get(static_url)
html_data.status_code
```

Out[5]:
```
200
```

2. Create a BeautifulSoup object from the HTML response

In [6]:
```
# 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

In [7]:
```
# Use soup.title attribute
soup.title
```

Out[7]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

3. Extract all column names from the HTML table header

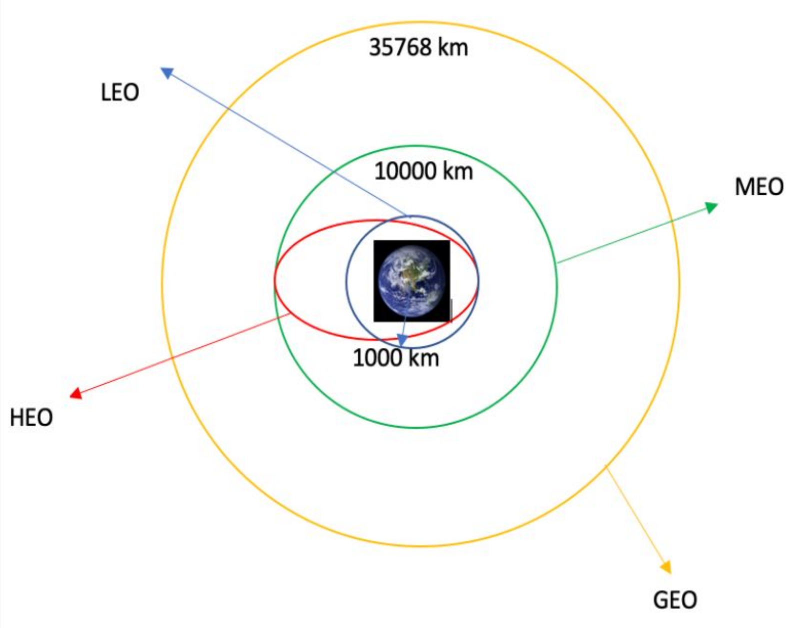
In [10]:
```
column_names = []

# 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 a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names

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

4. Create a dataframe by parsing the launch HTML tables
5. Export data to csv

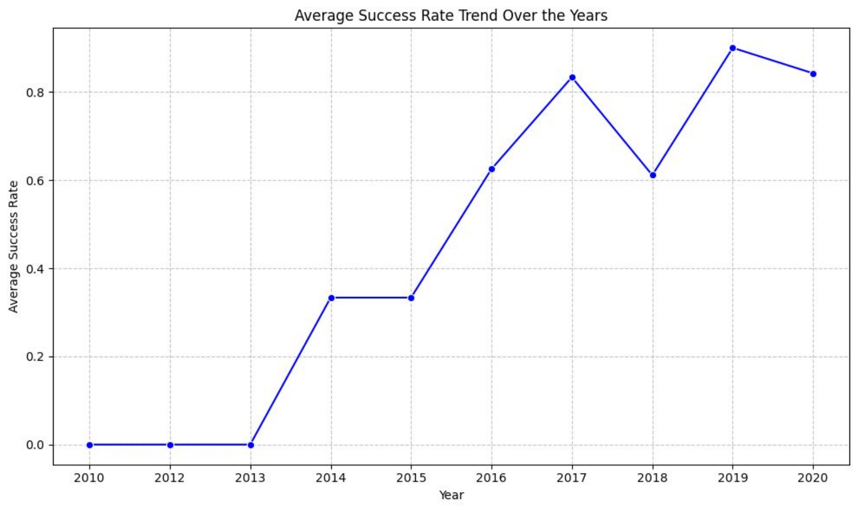
Data Wrangling

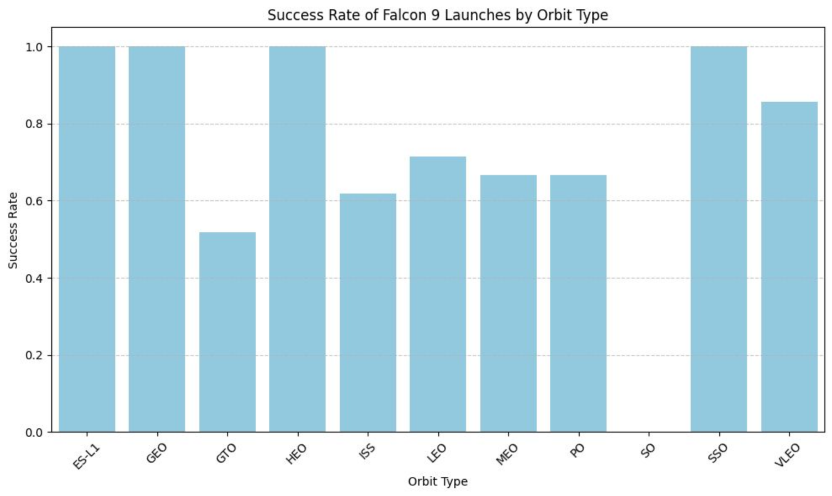


- Determine training labels after EDA
- Calculate number of launches against various data and occurrence of each orbits
- Create landing outcome label from outcome column to be exported to csv
- Download the notebook here
- Download the CSV here

EDA with Data Visualization

- Visualizing the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend.





- The link to the notebook is [here]

EDA with SQL

- Loaded the SpaceX dataset into a PostgreSQL database without leaving the jupyter notebook.
- Analyze the Data with SQL to get insight from the data. Some queries including
  - The names of unique launch sites in the space mission.
  - The total payload mass carried by boosters launched by NASA (CRS)
  - The average payload mass carried by booster version F9 v1.1
  - The total number of successful and failure mission outcomes
  - The failed landing outcomes in drone ship, their booster version and launch site names.
- Download the notebook here

Build an Interactive Map with Folium

- Marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- Assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, identifying which launch sites have relatively high success rate.
- Calculated the distances between a launch site to its proximities, answering questions below for example (download the notebook here:
    - Are launch sites near railways, highways and coastlines.
    - Do launch sites keep certain distance away from cities.

Build a Dashboard with Plotly Dash

- Built interactive dashboard with Plotly dash
- Plotted pie charts showing the total launches by a certain sites
- Plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- Download the program here

Predictive Analysis (Classification Models)

- Loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- Built different machine learning models and tune different hyperparameters using GridSearchCV.
- Using accuracy as the metric for our model, it improved the model using feature engineering and algorithm tuning.
- Determined the best performing classification model, which is Sigmoid
- The link to the notebook is here

Results

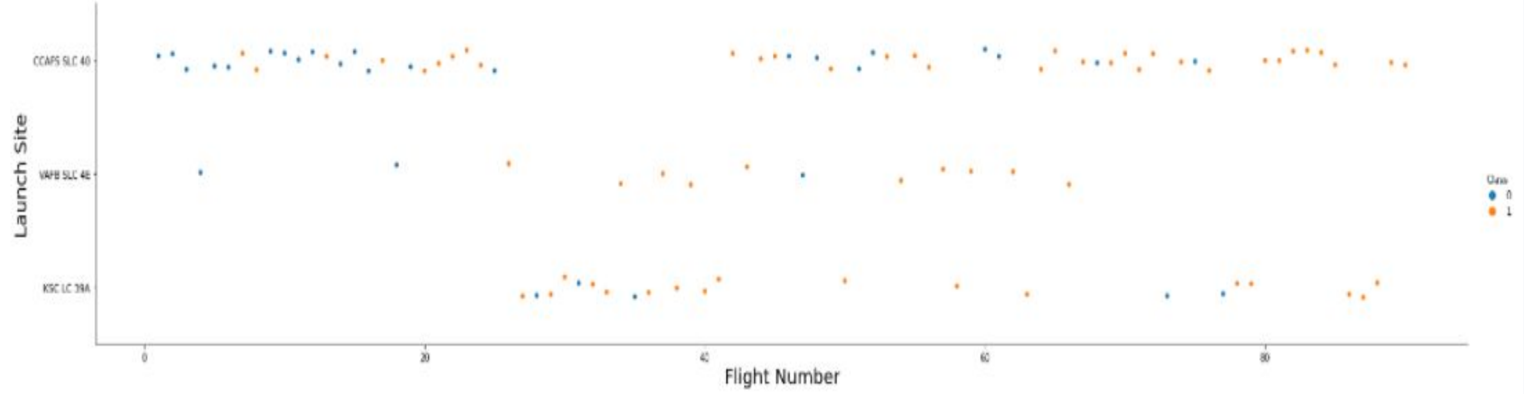
- Exploratory Data Analysis Results
- Interactive Analytics Demo Shots
- Predictive Analysis Results

Section 2

Insights drawn from EDA

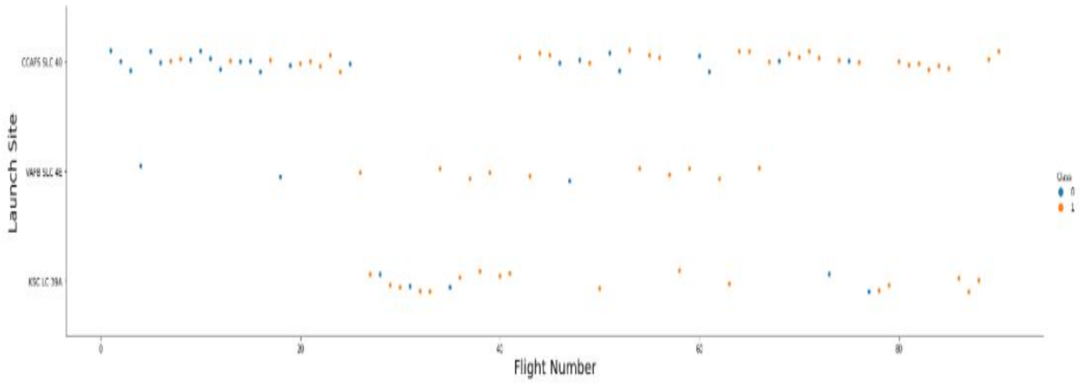
Flight Number vs. Launch Site

- The larger the flight amount at a launch site, the greater the success rate at a launch site.



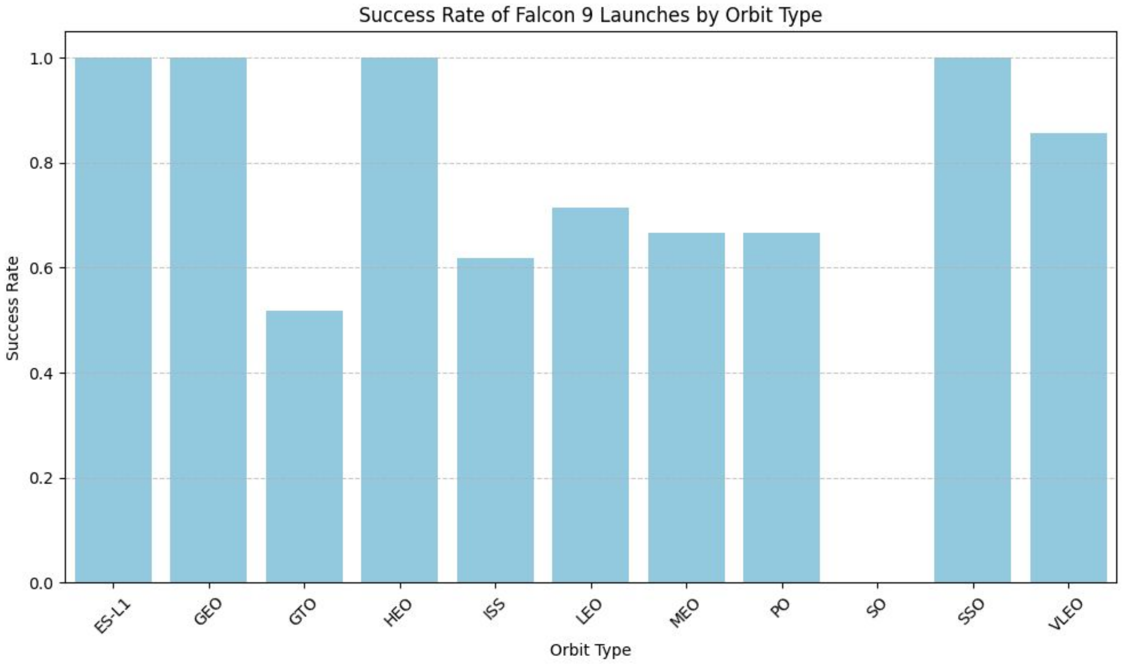
Payload vs. Launch Site

At launch site CCAFS SLC40, the greater the payload mass the higher successful launch rate



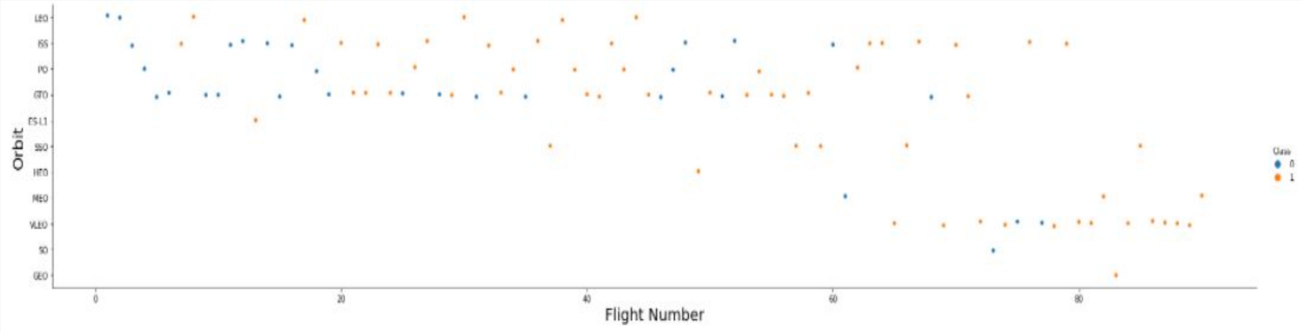
Success Rate vs. Orbit Type

- From the plot, ES-L1, GEO, HEO, SSO, VLEO had the most success rate.



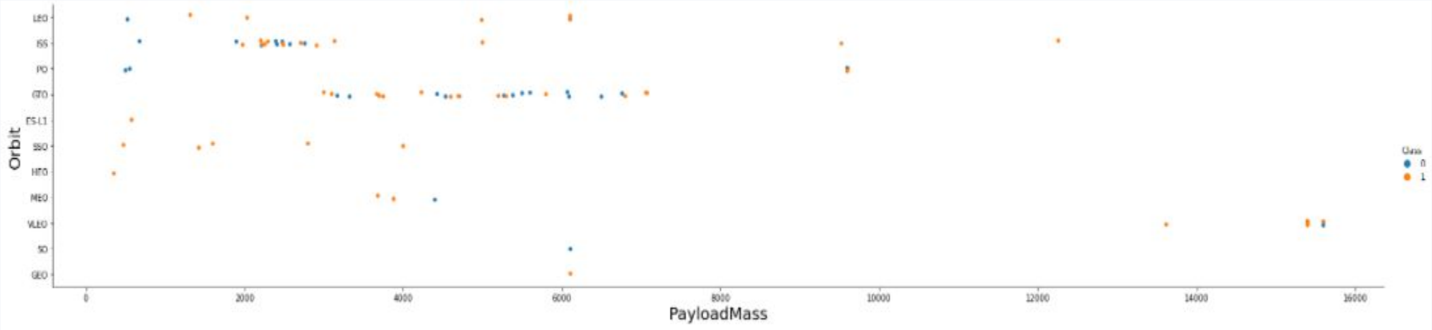
Flight Number vs. Orbit Type

- The plot below shows the Flight Number vs. Orbit type. That in the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.



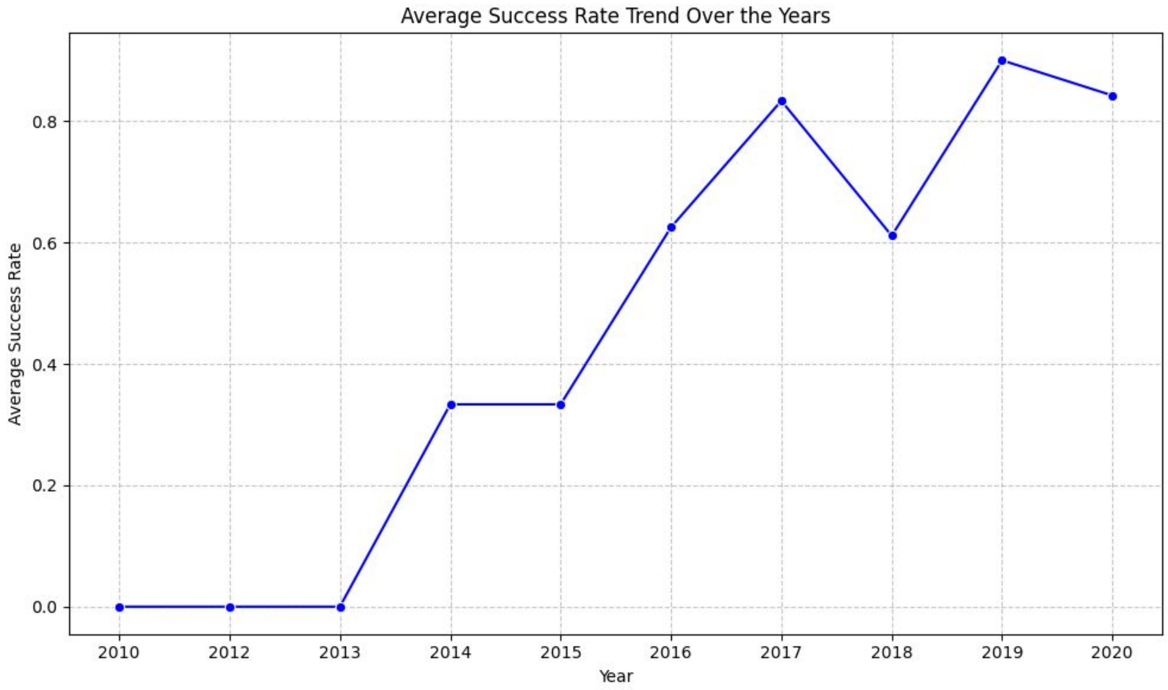
Payload vs. Orbit Type

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



Launch Success Yearly Trend

- Success rate has been increasing steadily since 2013 kept on increasing till 2020.



All Launch Site Names

- Using **DISTINCT** SQL query to show only unique launch sites from the SpaceX data.

Task 1

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

In [9]:

```
%%sql
SELECT DISTINCT "Launch_Site"
FROM SPACEXTBLS
```

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

Out[9]:

Launch_Site
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40
None

Launch Site Names Begin with 'CCA'

Task 2

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

In [10]:

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

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

Out[10]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outc
06/04/2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX	Success	Failure (paracl
12/08/2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0.0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (paracl
22/05/2012	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525.0	LEO (ISS)	NASA (COTS)	Success	No att
10/08/2012	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	NASA (CRS)	Success	No att
03/01/2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	NASA (CRS)	Success	No att

- Filter only launch sites that begin with `CCA`
-

Total Payload Mass

- Calculated the total payload carried by boosters from NASA as 45596 using the query below

Task 3

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

In [12]:
```
%%sql
SELECT SUM(PAYLOAD_MASS__KG_) AS 'Total_Payload_NASA_CRS'
FROM SPACEXTBLS
WHERE "Customer" = 'NASA (CRS)'
```

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

Out[12]:

Total_Payload_NASA_CRS
45596.0

Average Payload Mass by F9 v1.1

- Calculated the average payload mass carried by booster version F9 v1.1 as 2928.4

Task 4

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

In [15]:

```sql
%%sql
SELECT AVG(PAYLOAD_MASS__KG_) AS 'Mean_Payload_F9_v1.1_Booster'
FROM SPACEXTBLS
WHERE "Booster_Version" = 'F9 v1.1'
```

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

Out[15]:

Mean_Payload_F9_v1.1_Booster
2928.4

First Successful Ground Landing Date

- The first successful landing was in December 12nd 2015

Task 5

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

Hint:Use min function

In [14]:

```
%%sql
SELECT MIN(DATE((substr(Date, 7, 4) || '-' || substr(Date, 4, 2) || '-' || substr(Date, 1, 2)))) AS Date_First_Sucessful_La
FROM SPACEXTBLS
WHERE "Landing_Outcome" = 'Success (ground pad)'
```

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

Out[14]:

Date_First_Sucessful_Landing_Ground_Pad
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- **Used WHERE** clause to filter for boosters which have successfully landed on drone ship and applied the **AND** condition to determine successful landing with payload mass greater than 4000 but less than 6000

Task 6

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

In [16]:

```
%%sql
SELECT "Booster_Version"
FROM SPACEXTBLs
WHERE "Landing_Outcome" = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000
```

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

Out[16]:

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

Total Number of Successful and Failure Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

In [18]:
```
%%sql
SELECT "Mission_Outcome", COUNT(*)
FROM SPACEXTBLS
GROUP BY "Mission_Outcome"
```

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

Out[18]:

Mission_Outcome	COUNT(*)
None	898
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

- Using count to check for **WHERE** MissionOutcome was a success or a failure.

Boosters Carried Maximum Payload

- Determined the booster that have carried the maximum payload using a subquery in the **WHERE** clause and the **MAX()** function.

Task 8

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

In [19]:

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

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

Out[19]:

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600.0
F9 B5 B1049.4	15600.0
F9 B5 B1051.3	15600.0
F9 B5 B1056.4	15600.0
F9 B5 B1048.5	15600.0
F9 B5 B1051.4	15600.0
F9 B5 B1049.5	15600.0
F9 B5 B1060.2	15600.0
F9 B5 B1058.3	15600.0
F9 B5 B1051.6	15600.0
F9 B5 B1060.3	15600.0
F9 B5 B1049.7	15600.0

2015 Launch Records

- Using substr to circumvent the limitation of SQLLite, to check Failure in year 2015

Task 9

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

In [22]:

```
%%sql
SELECT substr(Date, 4, 2) AS Month, "Landing_Outcome", "Booster_Version", "Launch_Site"
FROM SPACEXTBLS
WHERE substr(Date, 7, 4) = '2015' AND "Landing_Outcome" = 'Failure (drone ship)'
```

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

Out[22]:

Month	Landing_Outcome	Booster_Version	Launch_Site
10	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

Task 10

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.

In [23]:

```
%%sql
SELECT "Landing_Outcome", COUNT(*)
FROM SPACEXTBLS
GROUP BY "Landing_Outcome"
HAVING DATE((substr(Date, 7, 4) || '-' || substr(Date, 4, 2) || '-' || substr(Date, 1, 2))) BETWEEN '2010-06-04' AND '2017-
ORDER BY COUNT(*) DESC
```

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

Out[23]:

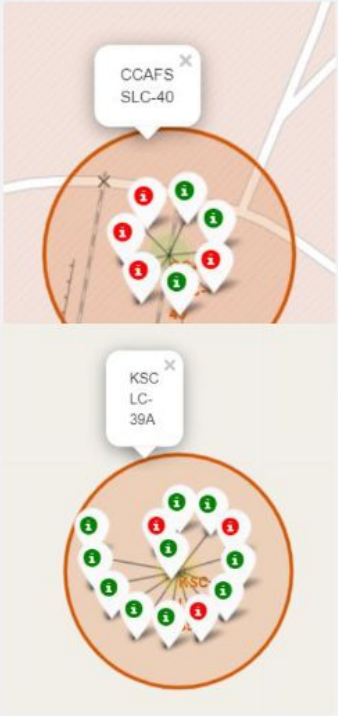
Landing_Outcome	COUNT(*)
No attempt	21
Success (drone ship)	14
Success (ground pad)	9
Failure (drone ship)	5
Controlled (ocean)	5
Uncontrolled (ocean)	2
Precluded (drone ship)	1

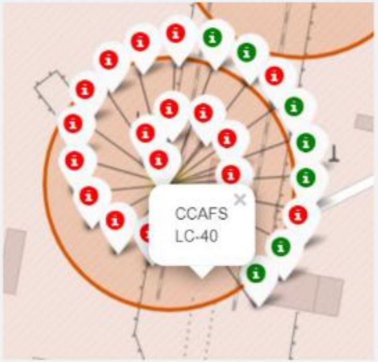
Section 4

Launch Sites Proximities Analysis

All launch sites global map markers

Markers showing launch sites with color labels





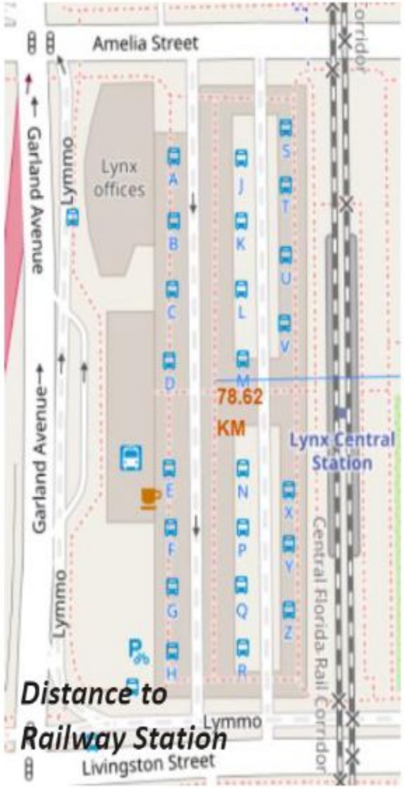
Florida Launch Sites

Green Marker shows successful Launches and Red Marker shows Failures

California Launch Site

37

Launch Site distance to landmarks

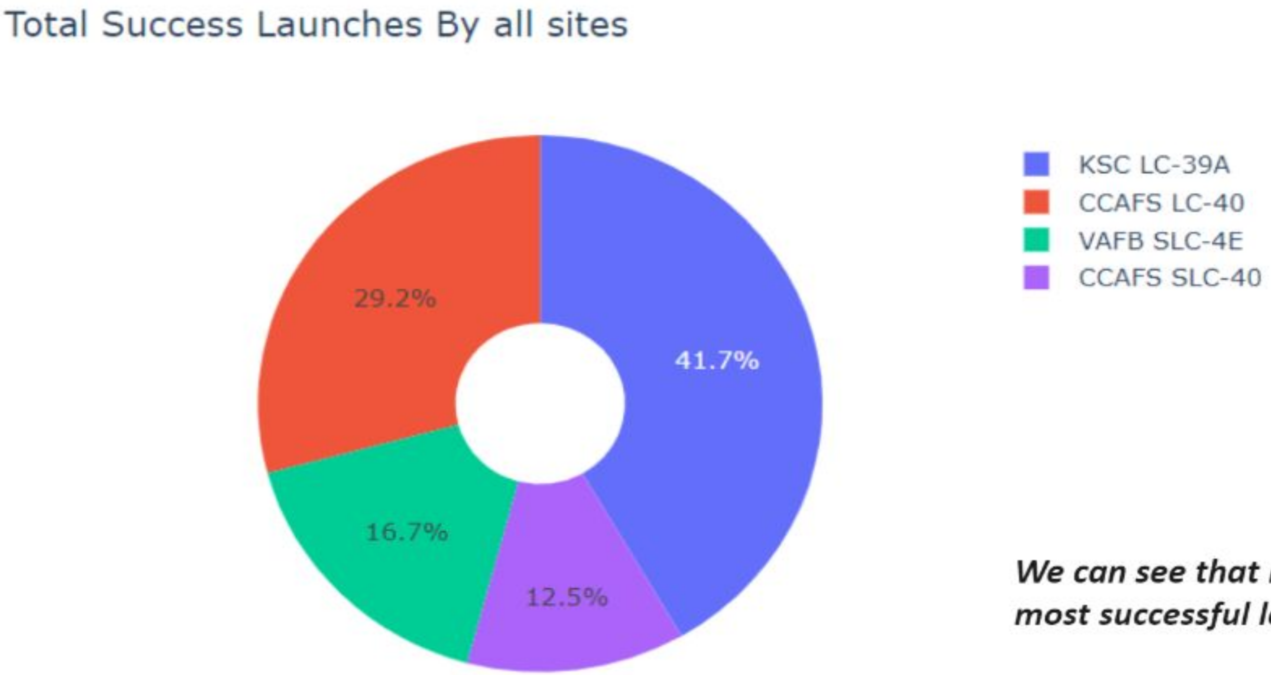


- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Section 5

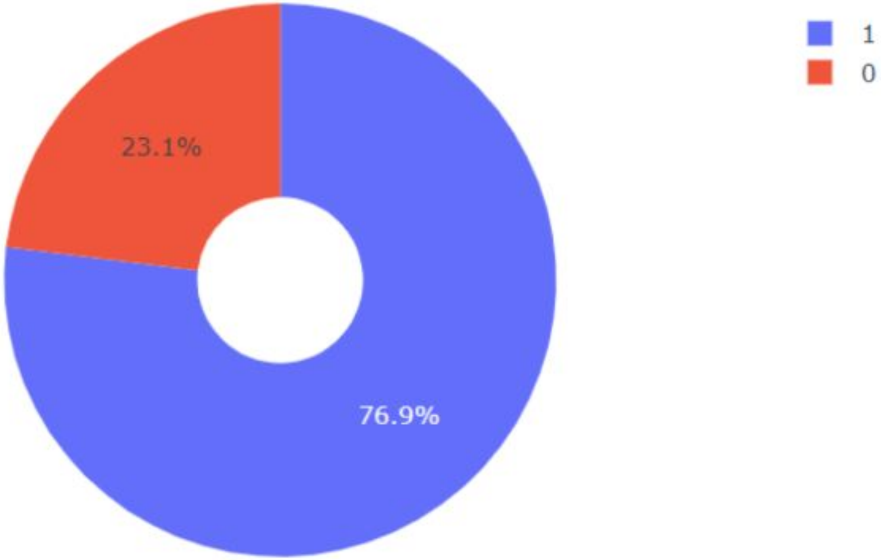
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



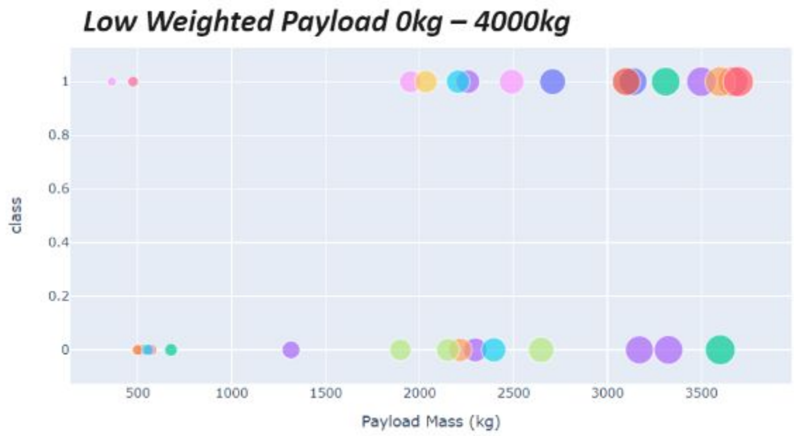
We can see that KSC LC-39A had the most successful launches from all the sites

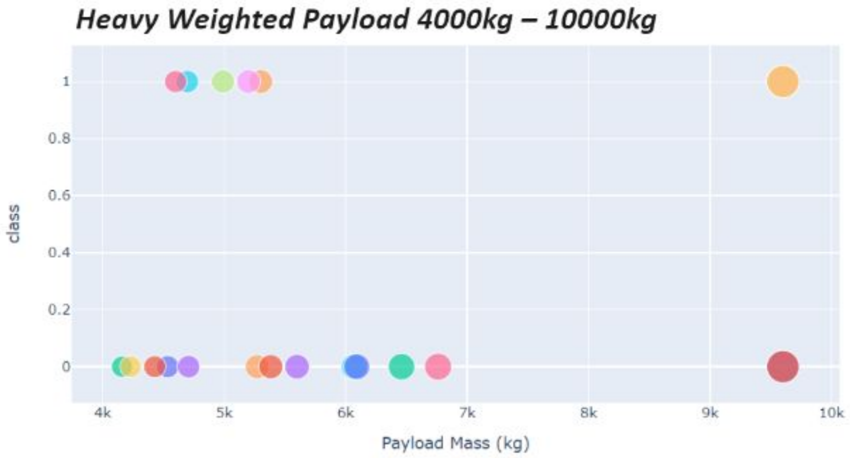
Pie chart showing the Launch site with the highest launch success ratio



KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider





We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 6

Predictive Analysis (Classification)

Classification Accuracy

The decision tree classifier is the model with the highest classification accuracy

TASK 8

Create a decision tree classifier object then create a `GridSearchCV` object `tree_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

In [17]:

```
# define hyperparameters to tune
parameters_tree = {'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]}

# define the model
tree = DecisionTreeClassifier(random_state = 12345)

# define the grid search object
grid_search_tree = GridSearchCV(
    estimator = tree,
    param_grid = parameters_tree,
    scoring = 'accuracy',
    cv = 10
)
# execute search
tree_cv = grid_search_tree.fit(X_train, Y_train)
```

In [18]:

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

Confusion Matrix

The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives, which is defined by unsuccessful landing marked as successful landing by the classifier.

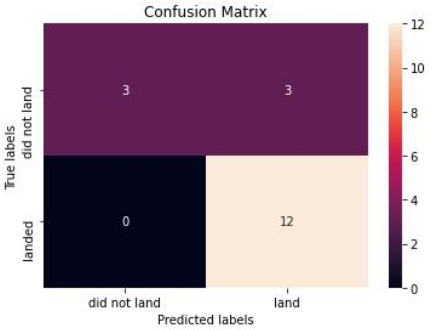
Calculate the accuracy of tree_cv on the test data using the method `score` :

```
In [19]: print('Accuracy on test data is: {:.3f}'.format(tree_cv.score(X_test, Y_test)))
```

```
Accuracy on test data is: 0.833
```

We can plot the confusion matrix

```
In [20]: yhat_tree = tree_cv.predict(X_test)
         plot_confusion_matrix(Y_test, yhat_tree)
```



Conclusions

We can conclude that:

- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase in 2013 till 2020.
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- KSC LC-39A had the most successful launches of any sites.
- The Decision tree classifier is the best machine learning algorithm for this task.

Thank you!