IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Thomas Freeman
24th April 2022

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection via API
  - Data Collection using Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with Data Visualization
  - Exploratory Data Analysis with SQL
  - Interactive Maps with Folium
  - Dashboard Analytics with Plotly Dash
  - Predictive Analytics via Classification Machine Learning
- Summary of all results
  - Exploratory data analysis results
  - Interactive analytics demo in screenshots
  - Predictive analysis results

Introduction

- Project background and context

Space X is a multi-billion-dollar aerospace company founded by Elon Musk in 2002 to revitalize public interest in the exploration of space, manufacturing rockets affordably. Such affordability is a result of unique design features that enable Space X's falcon 9 rockets to reuse their first stage. Understanding whether the recycling of the first stage lands successfully can help to determine the cost of launch. Hence in the knowledge of such cost-related-benefit, this gives a company the opportunity to be competitive.

Therefore, the objective of this project is to determine the price of each launch via the use of data gathering, data visualization and machine learning to predict landing success.

- Problems you want to find answers
  - What variables influence the success of a rocket landing?
  - What are the ideal requirements necessary to enable a successful landing program?
  - What is the most appropriate classification model and it's optimal hyperparameters?

Section 1

Methodology

Methodology

Executive Summary

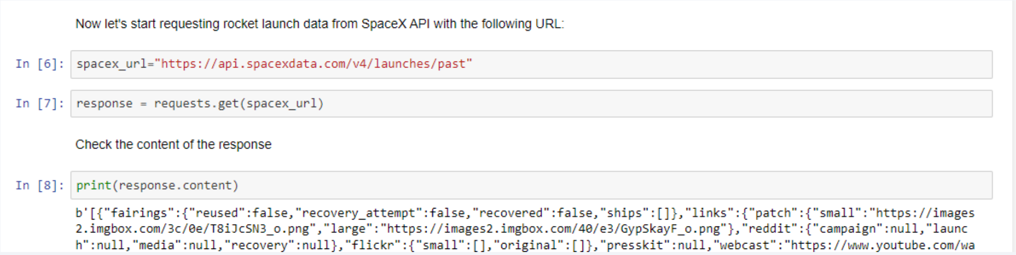
- Data collection methodology:
  - The data collection method for the project used both the SpaceX API, and web scraping from Wikipedia.
- Perform data wrangling
  - To understand variables in greater detail the method value_counts() with the creation of a dataframe to display successes and failures as a binary outcome and show the overall success rate.
- Perform exploratory data analysis (EDA) using visualization and SQL
  - One-hot encoding was applied to categorical attributes.
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- The SpaceX Data used with the project has been via multiple methods
  - Data was collected by means of a get request to the SpaceX API
  - Then to make the requested JSON results more consistent, a static response object was used
  - Subsequently, the response content was decoded as a Json using the .json() function and transformed into a Pandas dataframe using .json_normalize()
  - The data was then filtered to keep Falcon 9 launches only; the sequential flight number was reset, and missing Pay Load Mass values were identified and replaced with its mean column value in order to clean the data
  - Additionally, web scraping using BeautifulSoup was implemented on a Wikipedia webpage holding details of Falcon 9 launches
  - Finally, the extracted HTML table launch records were parsed and converted into a Pandas dataframe

Data Collection – SpaceX API

- A get request was implemented to the SpaceX API to retrieve the data. Subsequently cleansing and wrangling was applied to the data to, filter, re-format, identify and replace missing values with a calculated mean.
- The link to the Notebook can be see below: https://github.com/FreeTheFreeman/IBMAppliedDataScienceCapstone/blob/master/SpaceX%20Lab:%20Data%20Collection%20%26%20Wrangling.ipynb



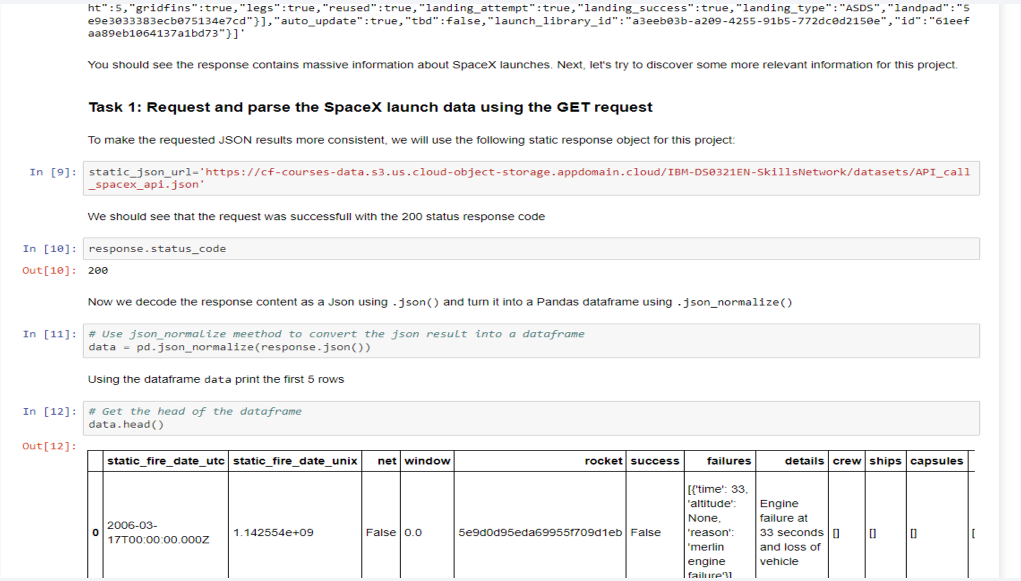
Now let's start requesting rocket launch data from SpaceX API with the following URL:

```
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/3c/0e/T8iJcSN3_o.png","large":"https://images2.imgbox.com/40/e3/GypSkayF_o.png"},"reddit":{"campaign":null,"launch":null,"media":null,"recovery":null},"flickr":{"small":[],"original":[]},"presskit":null,"webcast":"https://www.youtube.com/wa



ht":5,"gridfins":true,"legs":true,"reused":true,"landing_attempt":true,"landing_success":true,"landing_type":"ASDS","landpad":"5e9e3033383ecb075134e7cd"}],"auto_update":true,"tbd":false,"launch_library_id":"a3eeb03b-a209-4255-91b5-772dc0d2150e","id":"61eefaa89eb1064137a1bd73"}]'

You should see the response contains massive information about SpaceX launches. Next, let's try to discover some more relevant information for this project.

Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

```
In [9]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

We should see that the request was successfull with the 200 status response code

```
In [10]: response.status_code
Out[10]: 200
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

```
In [11]: # Use json_normalize meethod to convert the json result into a dataframe
         data = pd.json_normalize(response.json())
```

Using the dataframe `data` print the first 5 rows

```
In [12]: # Get the head of the dataframe
         data.head()
```

Out[12]:

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships	capsules
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 33, 'altitude': None, 'reason': 'merlin engine failure'}]	Engine failure at 33 seconds and loss of vehicle	[]	[]	[]

Data Collection - Scraping

- An additional method of data collection was used, specifically Web Scraping. Falcon 9 launch records were scraped from Wikipedia via BeautifulSoup. The table was then parsed and converted into a pandas dataframe.
- The link to the Notebook can be see below: https://github.com/FreeTheFreeman/IBMAppliedDataScienceCapstone/blob/master/SpaceX%20Lab:%20Web%20Scraping.ipynb

To keep the lab tasks consistent, you will be asked to scrape the data from a snapshot of the `List of Falcon 9 and Falcon Heavy launches` Wikipage updated on `9th June 2021`

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

Next, request the HTML page from the above URL and get a `response` object

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

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

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

In [6]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(data,"html5lib")
```

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

In [7]:
```
# Use soup.title attribute
print (soup.title.get_text())
```

```
List of Falcon 9 and Falcon Heavy launches - Wikipedia
```

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's 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

In [8]:
```
# 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')
print(html_tables)
```

```
[<table class="multicol" role="presentation" style="border-collapse: collapse; padding: 0; border: 0; background:transparent; w
idth:100%;">

<tbody><tr>
<td style="text-align: left; vertical-align: top;">
<h3><span class="mw-headline" id="Rocket_configurations">Rocket configurations</span></h3>
<div class="chart noresize" style="margin-top:1em;max-width:420px;">
<div style="position:relative;min-height:320px;min-width:420px;max-width:420px;">
```

```
   <th scope="col"><a href="/wiki/Falcon_9_first-stage_landing_tests" title="Falcon 9 first-stage landing tests">Booster<br/
   >landing</a>
   </th></tr>
```

Next, we just need to iterate through the `<th>` elements and apply the provided `extract_column_from_header()` to extract column name one by one

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

names = first_launch_table.find_all('th')
for name in names:
    header = extract_column_from_header(name)
    if header is not None and len(str(header)) > 0:
        column_names.append(header)
    else:
        del header
```

Check the extracted column names

In [11]:
```
print(column_names)
```

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

Data Wrangling

- Firstly, missing values were identified, and the percentage of missing values was calculated for each attribute, and numerical and categorical columns were identified.
- Subsequently, the method value_counts() was used on the LaunchSite column, Orbit column, and Outcome column to determine the number of launches on each site, the occurrence of each orbit, and the number of landing_outcomes respectively.
- A Class dataframe was determined from an outcome array, created using an np.where clause that used a list of bad outcomes to attribute a binary outcome.
- Finally, the overall success rate was determined to be 66% via the .mean() function.
- The link to the Notebook can be see below: https://github.com/FreeTheFreeman/IBMAppliedDataScienceCapstone/blob/master/SpaceX%20Lab:%20Exploratory%20Data%20Analysis.ipynb

EDA with Data Visualization

- For the exploration of the data, visualizations of the relationships between Flight Number and Launch Site, Payload Mass and Launch Site, Flight Number and Orbit Type, and Payload Mass and Orbit Type were displayed as scatter charts.
- Furthermore, the Success Rate for each orbit type was displayed as a bar chart, and a yearly time series of success rates displayed a trend by means of a line graph.
- Feature engineering was also implemented to specific columns of the dataframe via the use of OneHotEncoding in the form of the get_dummies function. Then all numeric columns were cast to the type float64.
- The link to the Notebook can be see below: https://github.com/FreeTheFreeman/IBMAppliedDataScienceCapstone/blob/master/SpaceX%20Lab:%20Exploratory%20Data%20Analysis.ipynb

EDA with SQL

- The SQL queries performed displayed:
  - the names of the unique launch sites
  - 5 records where the launch sites began with 'CCA'
  - the total payload mass carried by boosters launched by NASA (CRS)
  - the average payload mass carried by booster version F9 v1.1
- The SQL queries also listed:
  - the date when the first successful landing outcome happed in ground pad
  - the names of the boosters which had success in drone ship for payload mass greater than 4000 but less than 6000
  - the total number of successful and failed mission outcomes
  - the names of the booster_versions which carried the maximum payload mass
  - the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
- Furthermore, the SQL query ranked the count of landing outcomes between the date 2010-06-04 and 2017-03-20, in descending order
- The link to the Notebook can be see below: https://github.com/FreeTheFreeman/IBMAppliedDataScienceCapstone/blob/master/SpaceX%20Lab:%20EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

- The interactive map using folium marked all launch sites on a map and added success and failed launches for each site with the use of markers and circles.
- The class was used to determine the color of a marker for each launch depending on whether the launch was successful or not, such that green was assigned to class 1 for a success, and red to class 0 to represent a failed launch.
- The distances between a launch site to its proximities were calculated, specifically the proximity of a specific launch site to the nearest coastline, nearest railway, nearest highway road and nearest city.
- This was all in order to determine the closeness of proximity to these locations and to understand whether launch site keep a specific distance away from cities.
- The link to the Notebook can be see below: https://github.com/FreeTheFreeman/IBMAppliedDataScienceCapstone/blob/master/SpaceX%20Lab:%20Visualisations%20with%20Folium.ipynb

Build a Dashboard with Plotly Dash

- The interactive dashboard was built using Plotly Dash
- A pie chart is displayed as the first chart on the dashboard showing the proportion of successful launches by site. This chart can be further interrogated to show the proportion of successes and failures per site via dropdown selection.
- Another chart on the interactive dashboard was a scatterplot to show the Outcome versus Payload Mass for each Booster version. This included an interactive range slider to enable the changing of the Payload Mass axis.
- The link to the Notebook can be see below: https://github.com/FreeTheFreeman/IBMAppliedDataScienceCapstone/blob/master/SpaceX%20Lab:%20Interactive%20Visualisations%20with%20Dash.ipynb

Predictive Analysis (Classification)

- Firstly, a NumPy array from the column Class was created, by applying the method to_numpy() and assigning it to a variable
- The data was then transformed by standardising it using StandardScalar() and subsequently split into Training and Testing Data sets via the use of test_train_split()
- Then multiple machine learning models were constructed, including Logistic Regression, SVM, Decision Tree, and KNN; and each model was tuned using GridSearchCV for parameter optimisation.
- Each model's performance was assessed using the Accuracy metric on the Test data to inform which of the constructed and tuned models performed the best
- Hence subsequently we found the best forming tuned machine learning model
- The link to the Notebook can be see below: https://github.com/FreeTheFreeman/IBMAppliedDataScienceCapstone/blob/master/SpaceX%20Lab:%20Machine%20Learning.ipynb

Results

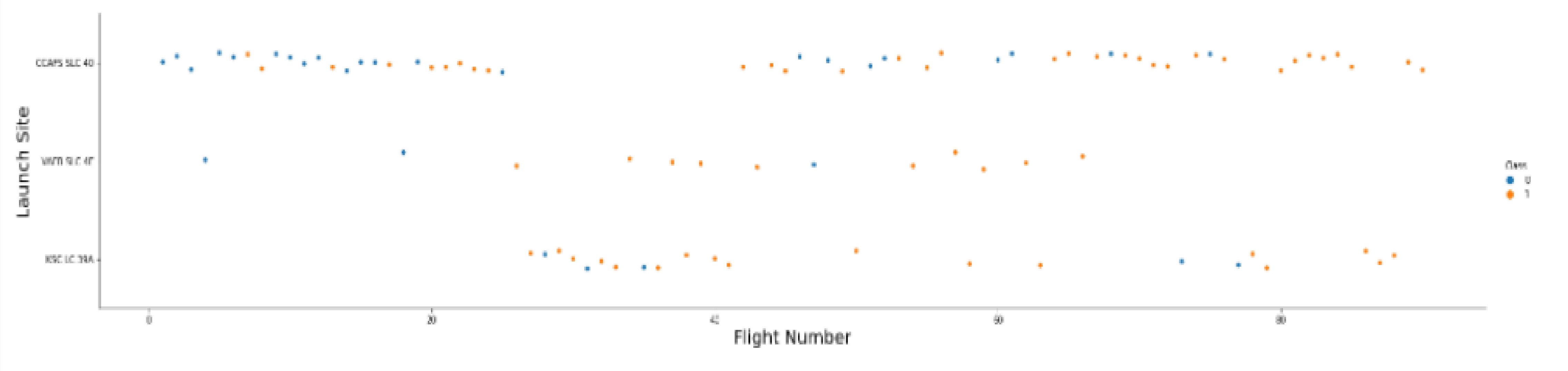
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

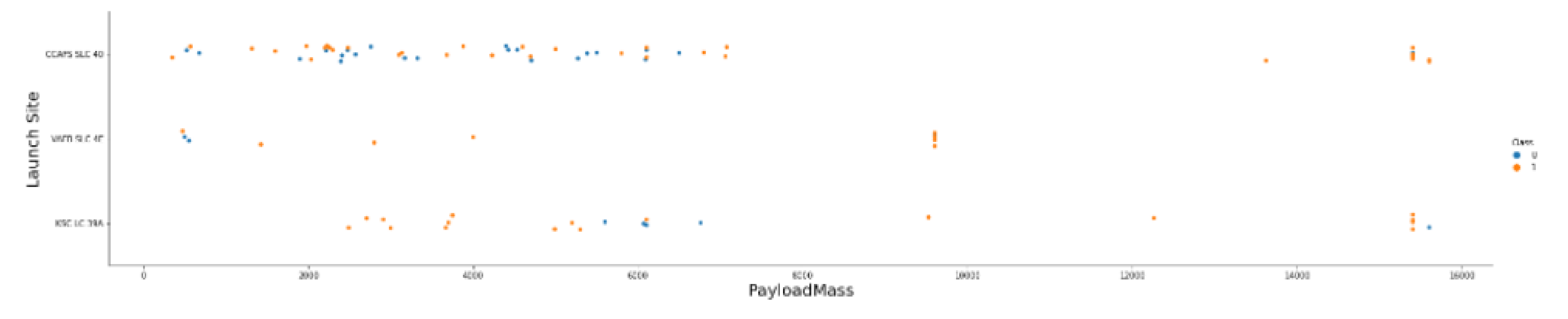
Flight Number vs. Launch Site

- As shown in the screenshot below, 0 represents a failed launch and 1 represents a successful launch.
- Generally, across all the launch sites as the flight number increases the number of successful launches increase.
- Furthermore, the site with the greatest number of launches is CCAFS SLC-40 and the site with the smallest number of launches is KSC LC-39A.



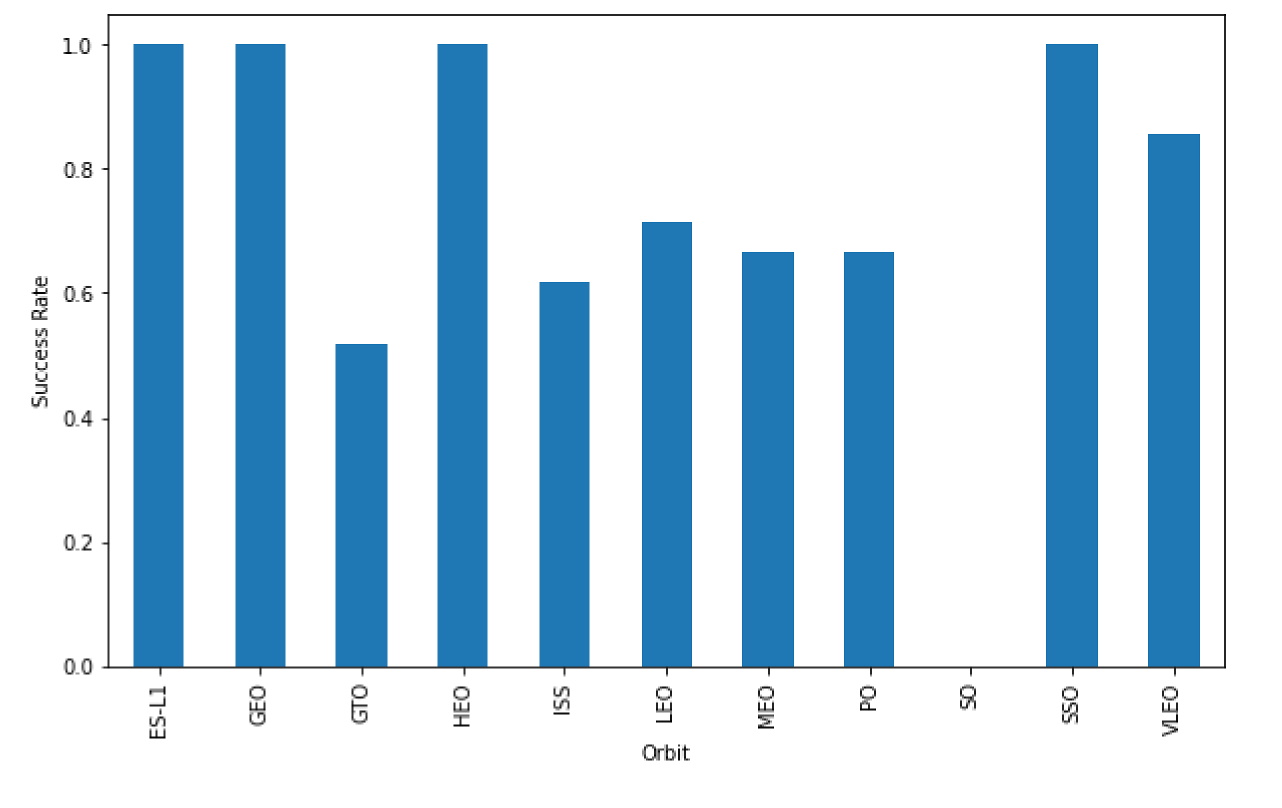
Payload vs. Launch Site

- As shown in the screenshot below, 0 represents a failed launch and 1 represents a successful launch.
- Generally, across all the launch sites as the Payload Mass increases the number of successful launches increase.
- Furthermore, a significant majority of launches across all sights is successful for Payload Masses greater than 9000kg.



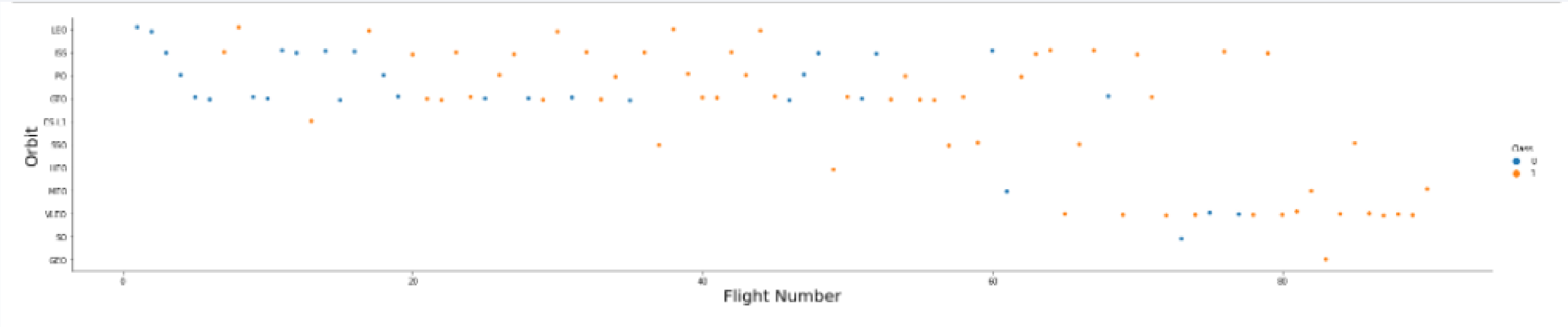
Success Rate vs. Orbit Type

- As seen in the bar graph to the right of the side, the Orbit with the least successful orbit is GTO, whereas the orbits with the most success are ES-L1, GEO, HEO and SSO with 100% success rate.



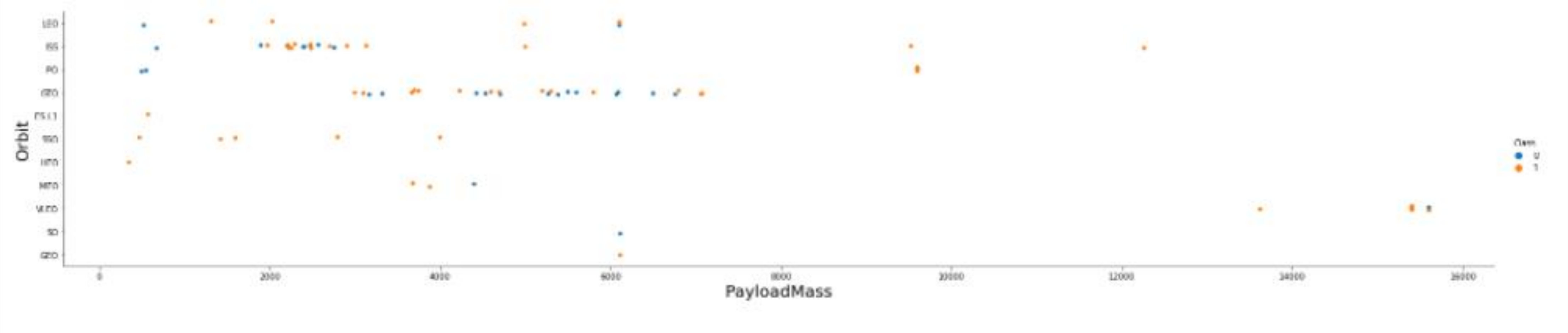
Flight Number vs. Orbit Type

- As shown in the screenshot below, 0 represents a failed launch and 1 represents a successful launch.
- As evident to in the graph, Success appears related to the number of flights for the LEO orbit. Whereas there seems to be no distinguishable relationship between flight number and success when in GTO orbit.



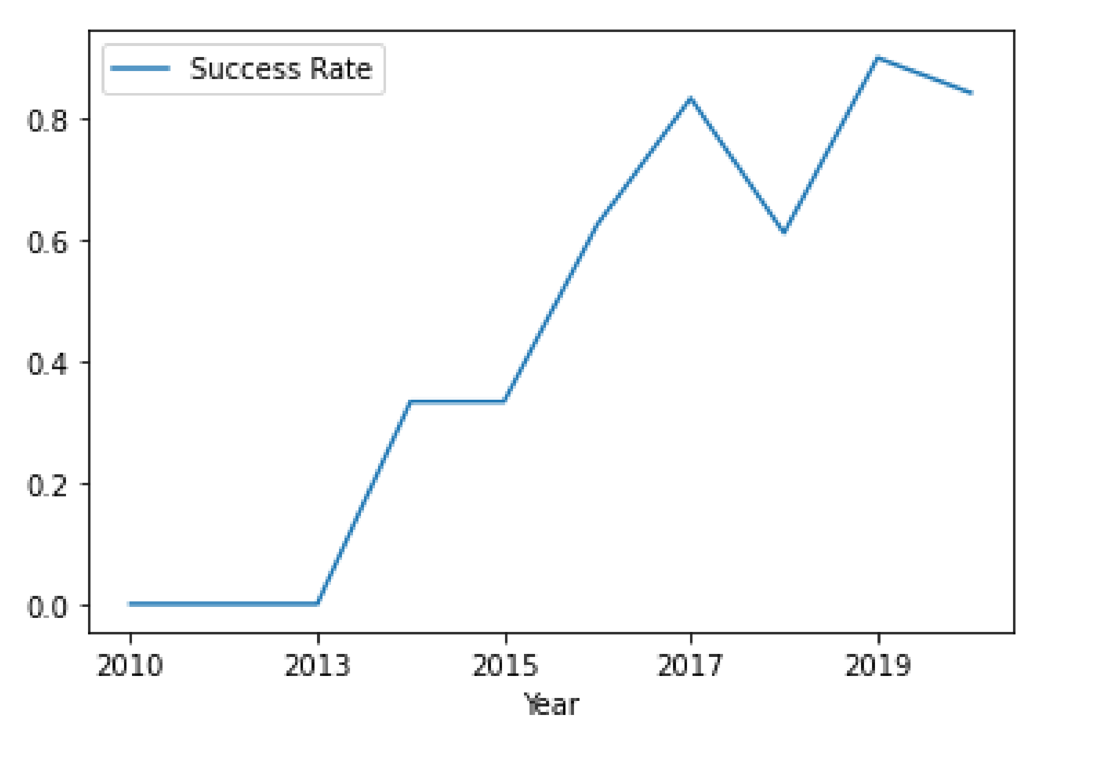
Payload vs. Orbit Type

- As shown in the screenshot below, 0 represents a failed launch and 1 represents a successful launch.
- As evident to in the graph, there is no evident relationship between successes as the Payload Mass increases for the GTO orbit.
- For orbits LEO, ISS and Polar success increases with Payload Mass.



Launch Success Yearly Trend

- The adjacent line graph shows for the first 3 years from 2010 there were no successes, yet the success rate has generally increased from 2013 onwards.



All Launch Site Names

- The below screenshot displays the SQL code used to determine the unique Launch Sites used for the space missions and display the results as a table

Task 1

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

In [4]:

```
%%sql

SELECT DISTINCT
launch_site
From SpaceX;
```

```
 * ibm_db_sa://pym48863:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/bludb
Done.
```

Out[4]:

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

Launch Site Names Begin with 'CCA'

- The below screenshot displays the SQL code used to show 5 records where the Launch Site begins with the character string ‘CCA’ and displays the results as a table

Task 2

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

In [5]:

```
%%sql

SELECT
*
FROM SpaceX
where launch_site like 'CCA%'
LIMIT 5;
```

```
 * ibm_db_sa://pym48863:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/bludb
Done.
```

Out[5]:

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

Total Payload Mass

- The below screenshot displays the SQL code used to determine the Total Payload Mass launched by NASA (CRS) via the ‘sum()’ aggregate function and displays the result

Task 3

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

In [6]:
```
%%sql

SELECT
sum(payload_mass__kg_) as Total_PayLoad_Mass
FROM SpaceX
WHERE customer = 'NASA (CRS)'
```

```
 * ibm_db_sa://pym48863:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/bludb
Done.
```

Out[6]:

total_payload_mass
45596

Average Payload Mass by F9 v1.1

- The below screenshot displays the SQL code used to determine the Average Payload Mass launched for booster version F9 v1.1 via the 'avg()' aggregate function and displays the result

Task 4

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

In [7]:
```
%%sql

SELECT
avg(payload_mass__kg_) as Average_Payload_Mass
FROM SpaceX
WHERE booster_version like 'F9 v1.1%'
```

```
 * ibm_db_sa://pym48863:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/bludb
Done.
```

Out[7]:

average_payload_mass
2534

First Successful Ground Landing Date

- The below screenshot displays the SQL code used to determine the date of the first successful ground landing via the use of the 'min()' aggregate function, and displays the result

Task 5

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

Hint:Use min function

In [8]:
```
%%sql

SELECT
min(Date) as First_Success_Landing
FROM SpaceX
WHERE landing__outcome = 'Success (ground pad)'
```

```
 * ibm_db_sa://pym48863:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/bludb
Done.
```

Out[8]:

first_success_landing
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- The below screenshot displays the SQL code used to determine the booster versions successfully landing on the drone ship with payloads between 4000 kg and 6000 kg, with use of a WHERE clause, and displays the result

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 [9]:
```
%%sql

SELECT
booster_version
FROM SpaceX
WHERE landing__outcome = 'Success (drone ship)'
    AND (payload_mass__kg_ > 4000 AND payload_mass__kg_ < 6000);
```

```
 * ibm_db_sa://pym48863:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/bludb
Done.
```

Out[9]:

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

Total Number of Successful and Failure Mission Outcomes

- The below screenshot displays the SQL code used to determine the total number of successful and failure mission outcomes with use of a ‘count()’ aggregate function and GROUP BY clause, and displays the result

Task 7

List the total number of successful and failure mission outcomes

In [10]:
```
%%sql

SELECT
mission_outcome,
count(*) as outcome_count
FROM SpaceX
GROUP BY mission_outcome;
```

```
 * ibm_db_sa://pym48863:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/bludb
Done.
```

Out[10]:

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

Boosters Carried Maximum Payload

- The below screenshot displays the SQL code used to determine a list of the names of the Booster Versions which have carried the maximum payload mass, via the use of a subquery and ‘max()’ aggregate function within the WHERE clause, and displays the result

Task 8

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

In [11]:
```
%%sql

SELECT
booster_version
FROM SpaceX
WHERE payload_mass__kg_ = (SELECT max(payload_mass__kg_) FROM SpaceX);
```

```
 * ibm_db_sa://pym48863:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/bludb
Done.
```

Out[11]:

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
F9 B5 B1060.3
F9 B5 B1049.7

2015 Launch Records

- The below screenshot displays the SQL code used to display the landing outcome, booster version and launch sites for failed drone ship landings in 2015, by use of a WHERE clause, and displays the result

Task 9

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

In [12]:

```
%%sql

SELECT
landing__outcome,
booster_version,
launch_site
FROM SpaceX
WHERE landing__outcome = 'Failure (drone ship)' AND DATE >= '2015-01-01' AND DATE <= '2015-12-31';
```

```
 * ibm_db_sa://pym48863:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/bludb
Done.
```

Out[12]:

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

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

- The adjacent screenshot displays the SQL code used to show the ranking of the count of landing outcomes between the dates 2010-06-04 and 2017-03-20, in descending order, by use of a ‘count()’ aggregate function and WHERE, GROUP BY, and ORDER BY clauses, and displays the result

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 [13]:

```
%%sql

SELECT
landing__outcome,
count(*) as landing_outcome_count
FROM SpaceX
WHERE DATE >= '2010-06-04' AND DATE <= '2017-03-20'
GROUP BY landing__outcome
ORDER BY landing_outcome_count desc;
```

```
 * ibm_db_sa://pym48863:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/bludb
Done.
```

Out[13]:

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

Section 3

Launch Sites Proximities Analysis

SpaceX Launch Sites

- As evident from the adjacent map, it is seen that launch sites are on the coast of the United States, specifically in states California and Florida.

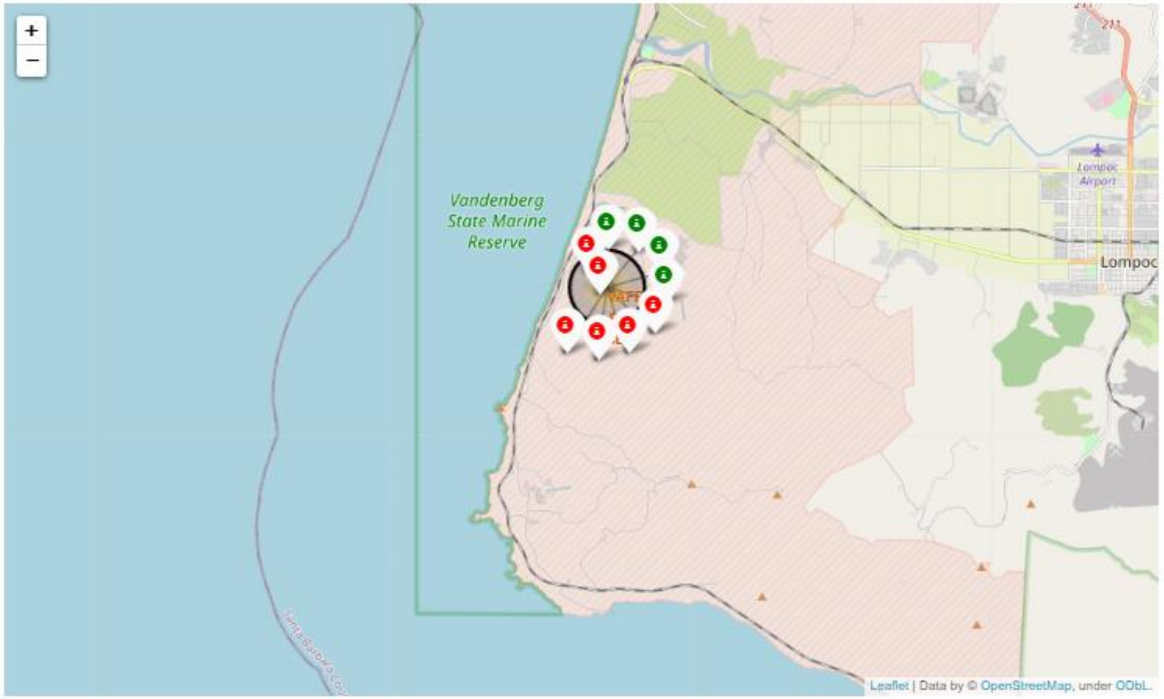
VAFB SLC-4E Launch Site Successes & Failures

- The neighboring screenshot shows the California launch site and distinguishes between the successful and failed launches with **green markers** and **red markers**, respectively.
- Out of the 10 launches at the California Launch site, 4 were successful and 6 failed.

In [13]:

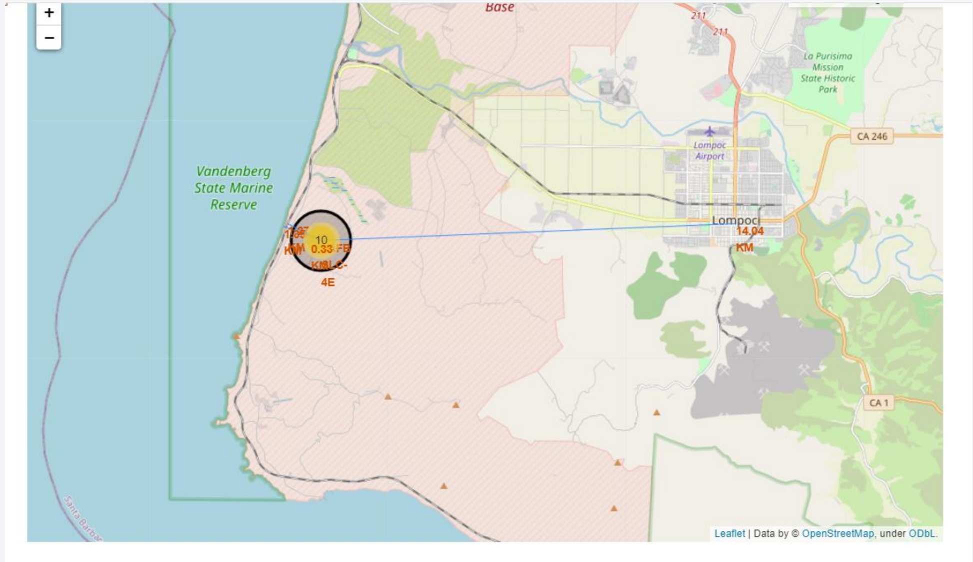
```
# Add marker_cluster to current site_map
site_map.add_child(marker_cluster)

# for each row in spacex_df data frame
# create a Marker object with its coordinate
# and customize the Marker's icon property to indicate if this launch was successed or failed,
# e.g., icon=folium.Icon(color='white', icon_color=row['marker_color']
for index, record in spacex_df.iterrows():
    coordinate = [record['Lat'], record['Long']]
    marker = folium.Marker(coordinate,
                  icon=folium.Icon(color='white', icon_color=record['marker_color']))
    marker_cluster.add_child(marker)
site_map
```

Out[13]:



VAFB SLC-4E Launch Site Proximity to Geographical Markers

- The top adjacent picture displays the distance in kilometers to the closest city which, as shown, is Lompoc and is 14km away from the VAFB SLC Launch site
- The bottom adjacent picture displace the distance in kilometers to the closest highway, railway and coast from the VADB SLC Launch site. These points of interest are 0.76km, 1.27km and 1.35km away from the California Launch site, respectively.

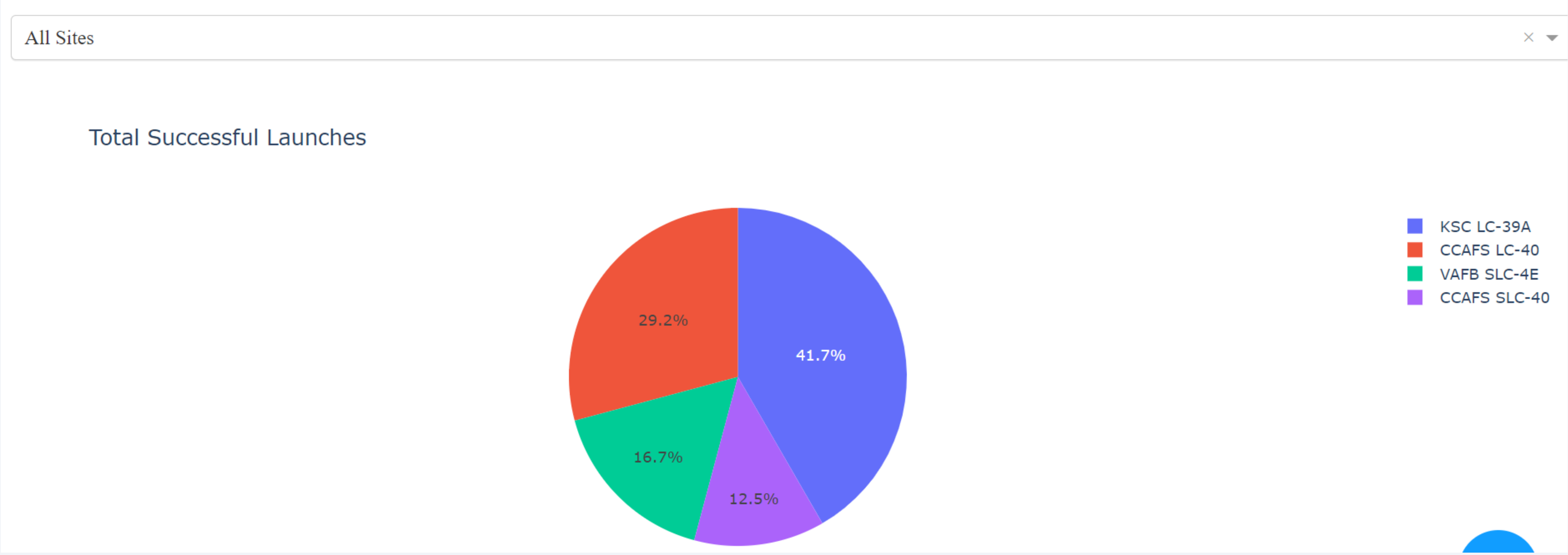


Section 4

Build a Dashboard with Plotly Dash

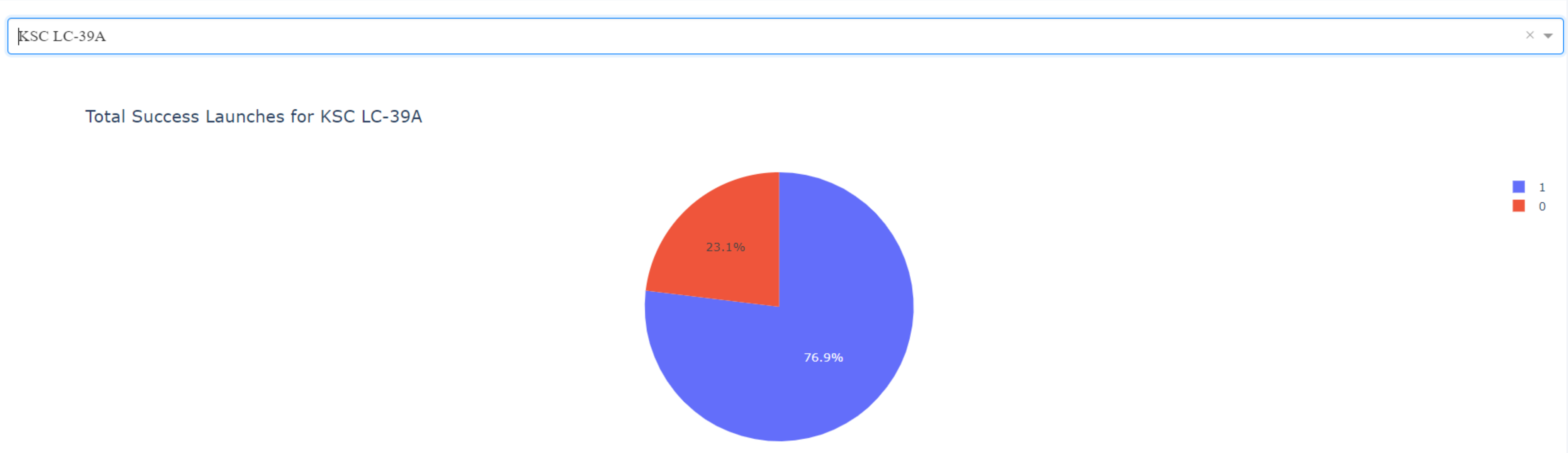
Launch Successes by Site Pie Chart

- The below screenshot displays the proportion of total success by launch site. Specifically, distinguishing that the KSC LC-39A Launch site has experienced the most successes at 41.7% of all successes and that the CCAFS SLC-40 launch site has experienced the least number of successes with 12.5% of total successes



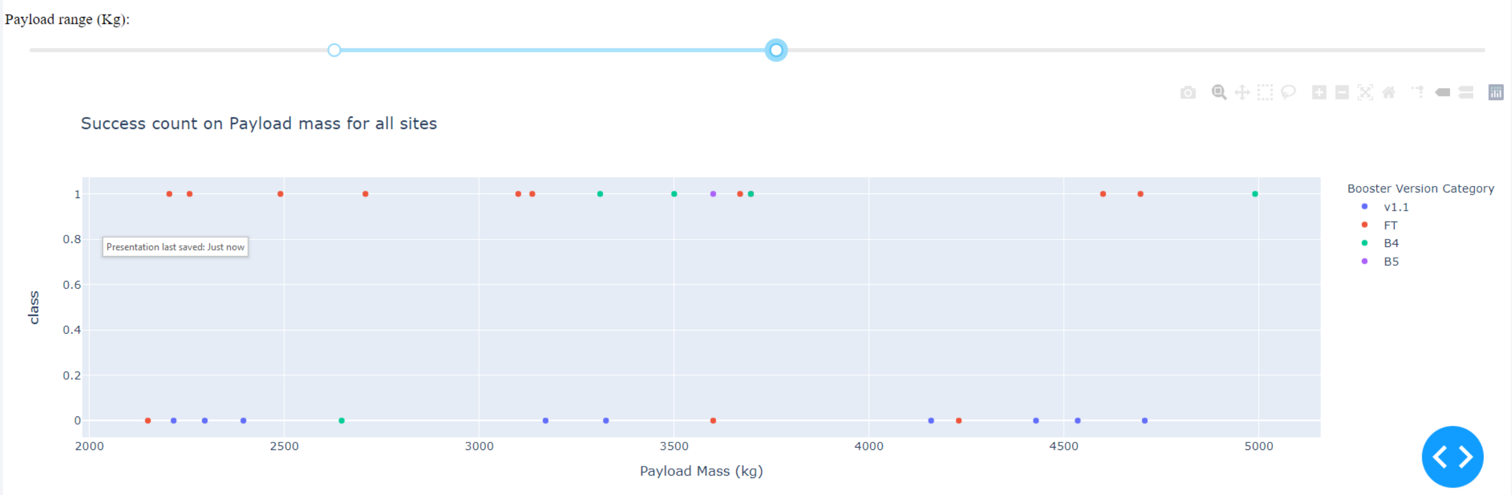
KSC LC-39A Success/Failure Rate Pie Chart

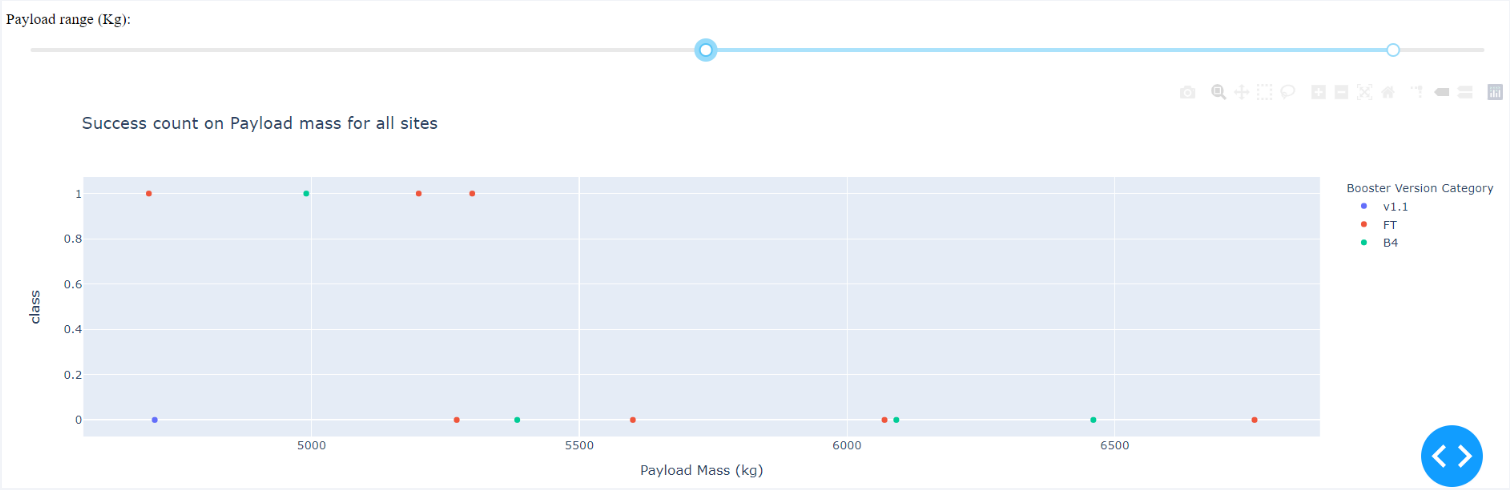
- The below screenshot displays the proportion of successes and failures for all launches at the KSC LC-39A Launch site. Specifically, out of all launches at this Launch site 76.9% were successful and 23.1% failed, where 1 represents a success and 0 represents a failure.



Multi-Range Payload vs. Launch Outcome Scatter Plot

- The two screenshots adjacent show different payload ranges, the top screenshot showing a range of 2000kg to 5250kg and the bottom screenshot of 4750kg to 7000kg.
- Booster Version FT has the most successes across both ranges.
- The bottom screenshot depicts that heavier payload masses are less successful than the lighter payload masses depicted in the top screenshot.





Section 5

Predictive Analysis (Classification)

Classification Accuracy

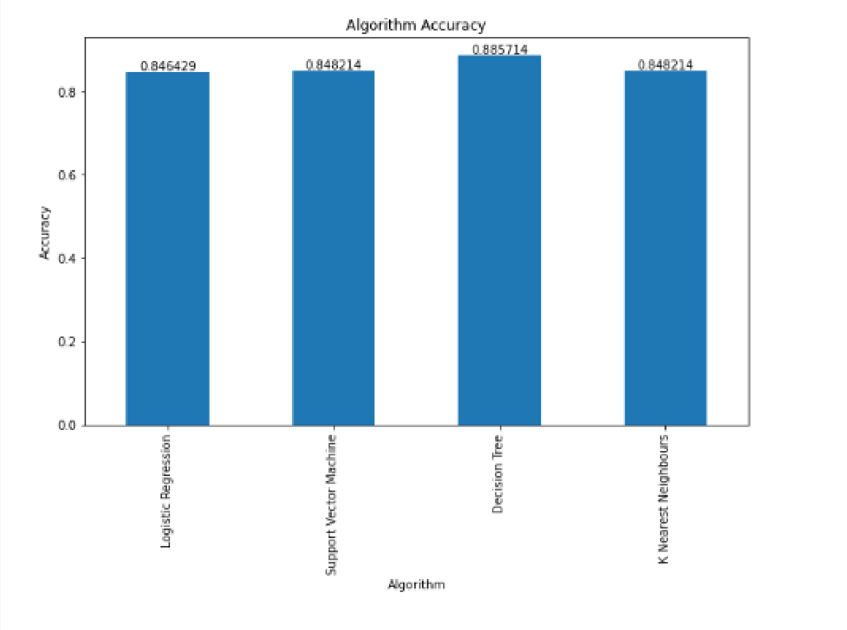
- The adjacent chart displays an accuracy result for each of the four algorithms applied to the SpaceX data

- The model with the highest classification accuracy is the Decision Tree Model, specifically with the below Hyper-Parameters

    - {'criterion': 'entropy', 'max_depth': 4, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 10, 'splitter': 'random'}

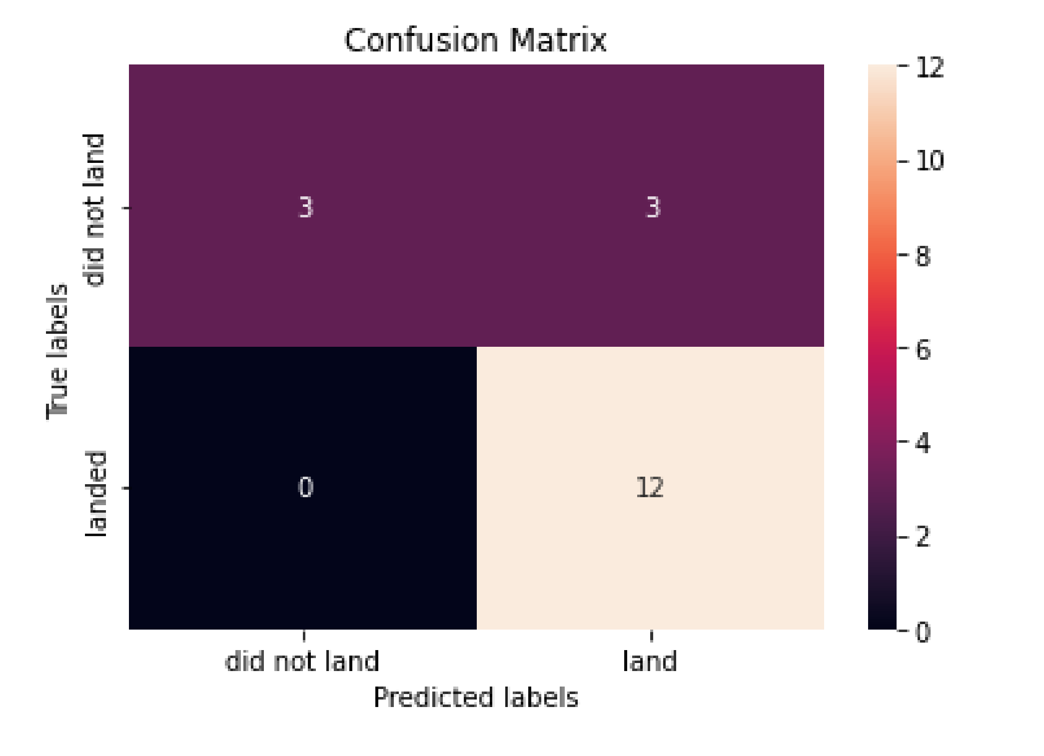
```
ML = {'LogisticRegression':logreg_cv.best_score_, 'SupportVectorMachine':svm_cv.best_score_,'DecisionTree':tree_cv.best_score_,'KNearestNeighbours':

bestML = max(ML, key=ML.get)

print('The best Machine Learning algorithm is',bestML,'with a score of', ML[bestML])

if bestML == 'LogisticRegression':
    print('with the tuned hyperparameters of:',logreg_cv.best_params_)
if bestML == 'SupportVectorMachine':
    print('with the tuned hyperparameters of:',svm_cv.best_params_)
if bestML == 'DecisionTree':
    print('with the tuned hyperparameters of:',tree_cv.best_params_)
if bestML == 'KNearestNeighbours':
    print('with the tuned hyperparameters of:',knn_cv.best_params_)
```

```
The best Machine Learning algorithm is DecisionTree with a score of 0.8767857142857143
with the tuned hyperparameters of: {'criterion': 'entropy', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 10,
'splitter': 'random'}
```



Confusion Matrix

- The adjacent confusion matrix is the confusion matrix of the aforementioned Decision Tree model, which was the best performing model.
- The Confusion Matrix states that the Decision Tree model has produced a three type 1 errors (False Positive) and no type 2 errors (False Negative), with recall of 1.0 and precision of 0.8.



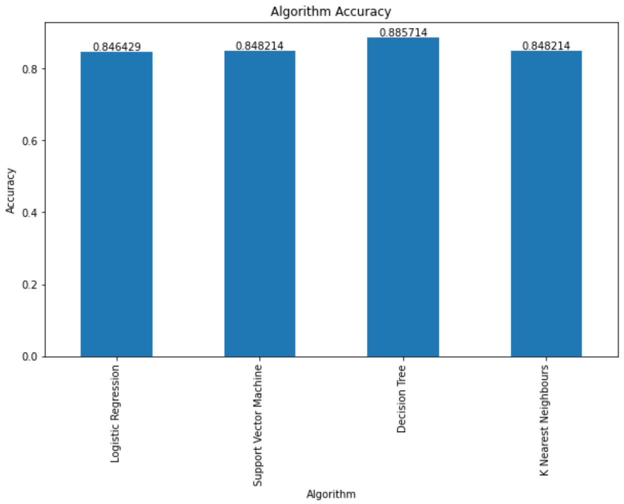
Conclusions

- From the analysis and investigations, it can be concluded that:
  - The larger the flight number at a launch site, the greater the success rate at that launch site
  - The most successful launch site of all the sites was, KSC LC-39A
  - Success rate generally increases over time, specifically from 2013 to 2020
  - The orbits with the highest success rate are ES-L1, GEO, HEO and SSO
  - The Decision Tree Model was the best machine learning classification model to predict future outcomes, specifically with tuned hyper parameters of {'criterion': 'entropy', 'max_depth': 4, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 10, 'splitter': 'random'}
- A future recommendation would be to apply other machine learning models to understand if there are better predictive models than the those used in this project

Appendix

- Code written to show the Accuracy of each classification Algorithm used within the Machine Learning section

```
In [31]: success_data = pd.DataFrame(columns=['Algorithm', 'Accuracy'])

         success_data = success_data.append({'Algorithm': 'Logistic Regression', 'Accuracy': logreg_cv.best_score_}, ignore_index=True)
         success_data = success_data.append({'Algorithm': 'Support Vector Machine', 'Accuracy': svm_cv.best_score_}, ignore_index=True)
         success_data = success_data.append({'Algorithm': 'Decision Tree', 'Accuracy': tree_cv.best_score_}, ignore_index=True)
         success_data = success_data.append({'Algorithm': 'K Nearest Neighbours', 'Accuracy': knn_cv.best_score_}, ignore_index=True)

         success_data.set_index('Algorithm', inplace=True)

         ax = success_data.plot(kind='bar', figsize=(10, 6), title='Algorithm Accuracy', xlabel='Algorithm', ylabel='Accuracy', legend=Fa
         lse)
         ax.bar_label(ax.containers[0], label_type='edge')

Out[31]: [Text(0, 0, '0.846429'),
          Text(0, 0, '0.848214'),
          Text(0, 0, '0.885714'),
          Text(0, 0, '0.848214')]
```



Thank you!