Winning Space Race with Data Science

Mayank Bisht
26 May 2023

Outline

Executive Summary

- **Summary of methodologies**

  -SpaceX Data Collection using SpaceX API

  -SpaceX Data Collection with Web Scraping

  -SpaceX Data Wrangling

  -SpaceX Exploratory Data Analysis using SQL

  -Space-X EDA DataViz Using Python Pandas and Matplotlib

  -Space-X Launch Sites Analysis with Folium-Interactive Visual Analytics and PlotyDash

  -SpaceX Machine Learning Landing Prediction
- **Summary of all results**

  -EDA results

  -Interactive Visual Analytics and Dashboards

  -Predictive Analysis(Classification)

Introduction

- **Project background and context**

SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

•Problems you want to find answers

In this capstone, we will predict if the Falcon 9 first stage will land successfully using data from Falcon 9 rocket launches advertised on its website.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:

  Describes how data sets were collected

- Perform data wrangling

  Describes how data were processed

- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and PlotlyDash
- Perform predictive analysis using classification models

  How to build, tune, evaluate classification models

Data Collection

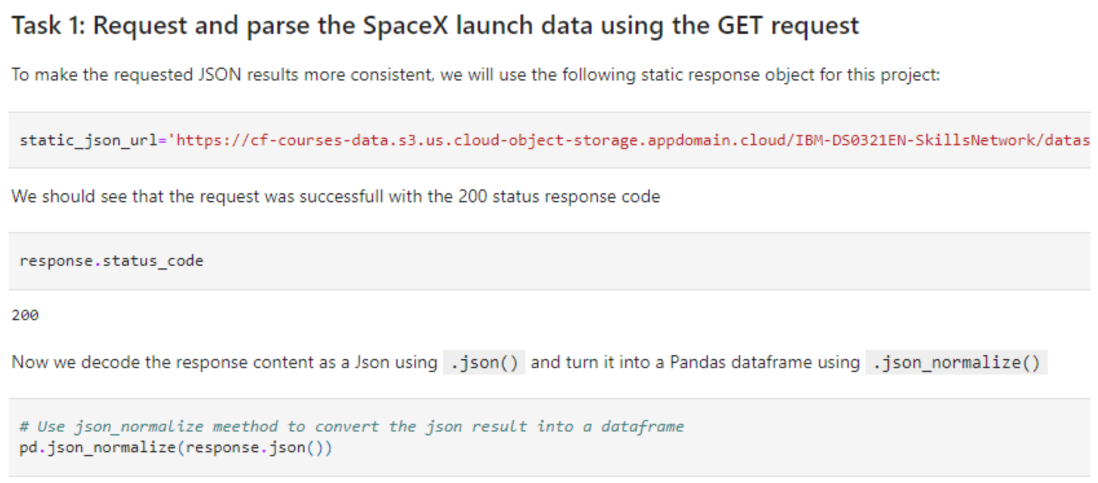
Description of how SpaceX Falcon9 data was collected.

- Data was first collected using SpaceX API (a RESTful API) by making a get request to the SpaceX API. This was done by first defining a series helper functions that would help in the use of the API to extract information using identification numbers in the launch data and then requesting rocket launch data from the SpaceX API url.
- Finally, to make the requested JSON results more consistent, the SpaceX launch data was requested and parsed using the GET request and then decoded the response content as a Json result which was then converted into a Pandas data frame.
- Also performed web scraping to collect Falcon 9 historical launch records from a Wikipedia page titled List of Falcon 9 and Falcon Heavy launches of the launch records are stored in a HTML. Using BeautifulSoup and request Libraries, I extract the Falcon 9 launch HTML table records from the Wikipedia page, Parsed the table and converted it into a Pandas data frame

Data Collection –SpaceX API

- Data collected using SpaceX API (a RESTful API) by making a get request to the SpaceX API then requested and parsed the SpaceX launch data using the GET request and decoded the response content as a Json result which was then converted into a Pandas data frame

- Here is the GitHub URL of the completed SpaceX API calls notebook (https://github.com/Mayank-Bisht-2000/IBM-Capstone-Project/blob/master/jupyter-labs-spacex-data-collection-api.ipynb)



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:

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datas
```

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

```
response.status_code
```

```
200
```

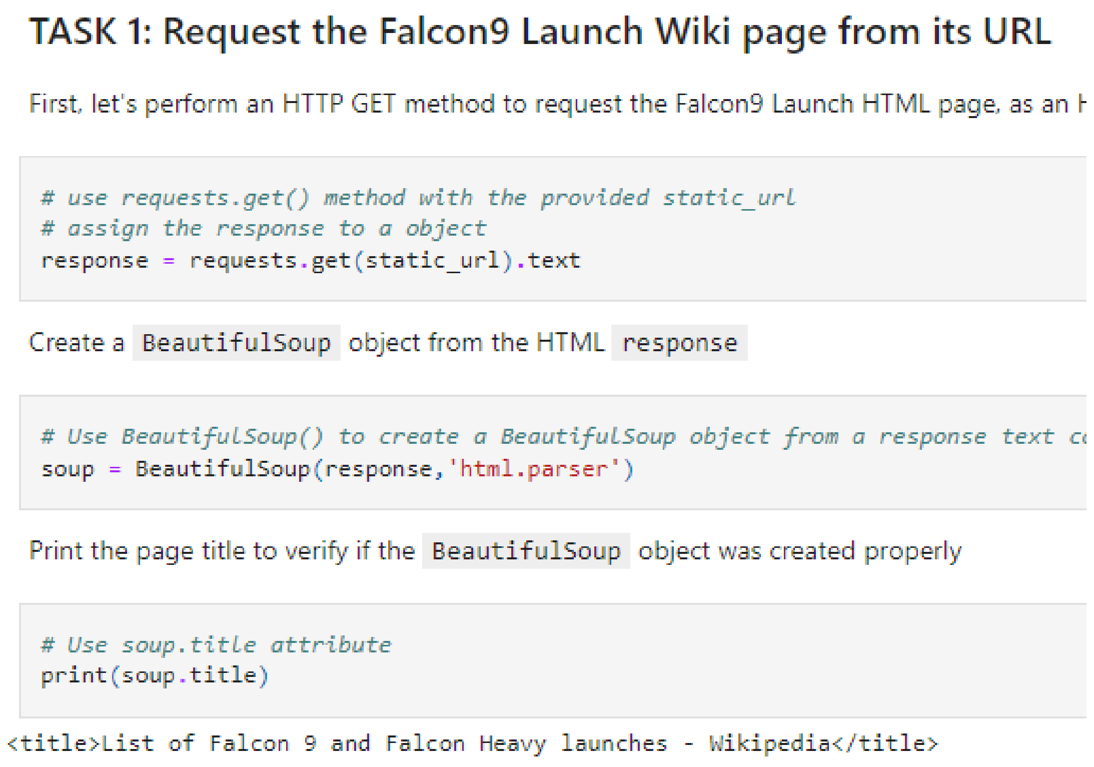
Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

```
# Use json_normalize meethod to convert the json result into a dataframe
pd.json_normalize(response.json())
```

Data Collection -Web Scraping

- Performed web scraping to collect Falcon 9 historical launch records from a Wikipedia using BeautifulSoup and request, to extract the Falcon 9 launch records from HTML table of the Wikipedia page, then created a data frame by parsing the launch HTML.

- Here is the GitHub URL of the completed web scraping notebook. (https://github.com/Mayank-Bisht-2000/IBM-Capstone-Project/blob/master/jupyter-labs-webscraping.ipynb)



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 H

```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url).text
```

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

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text co
soup = BeautifulSoup(response,'html.parser')
```

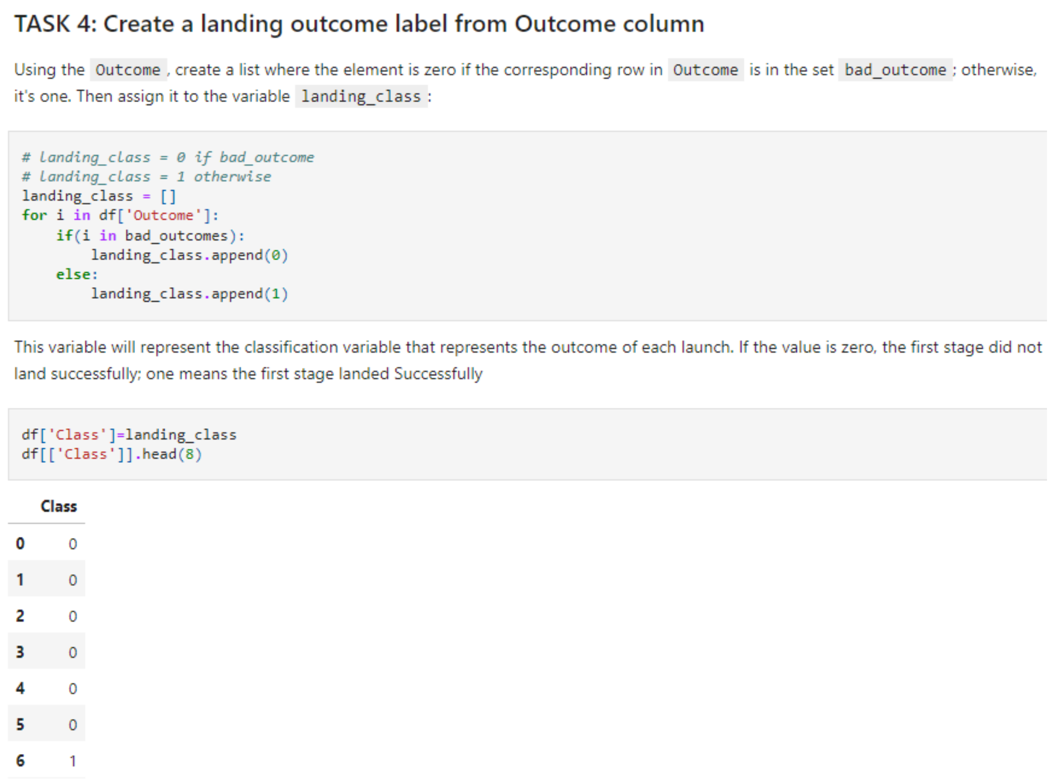
Print the page title to verify if the `BeautifulSoup` object was created properly

```
# Use soup.title attribute
print(soup.title)
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

Data Wrangling

- After obtaining and creating a Pandas DF from the collected data, data was filtered using the Booster Version column to only keep the Falcon 9 launches, then dealt with the missing data values in the Landing Pad and Payload Mass columns. For the Payload Mass, missing data values were replaced using mean value of column.
- Also performed some Exploratory Data Analysis (EDA) to find some patterns in the data and determine what would be the label for training supervised models
- Here is the GitHub URL of the completed data wrangling related notebooks.(https://github.com/Mayank-Bisht-2000/IBM-Capstone-Project/blob/master/IBM-DS0321EN-SkillsNetwork_labs_module_1_L3_labs-jupyter-spacex-data_wrangling_jupyterlite.jupyterlite.ipynb)



TASK 4: Create a landing outcome label from Outcome column

Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome`; otherwise, it's one. Then assign it to the variable `landing_class`:

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for i in df['Outcome']:
    if(i in bad_outcomes):
        landing_class.append(0)
    else:
        landing_class.append(1)
```

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

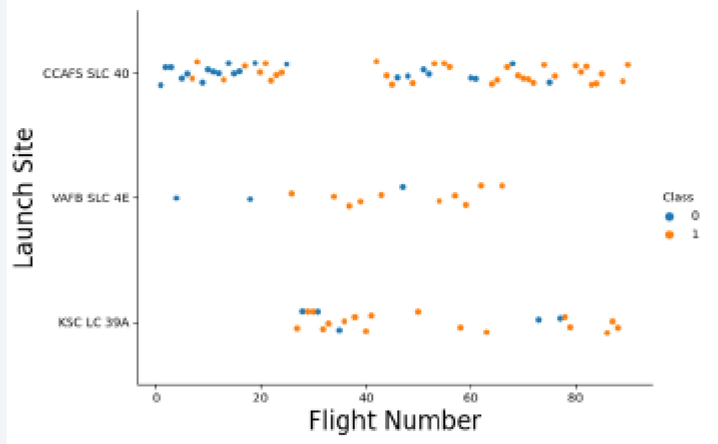
```
df['Class']=landing_class
df[['Class']].head(8)
```

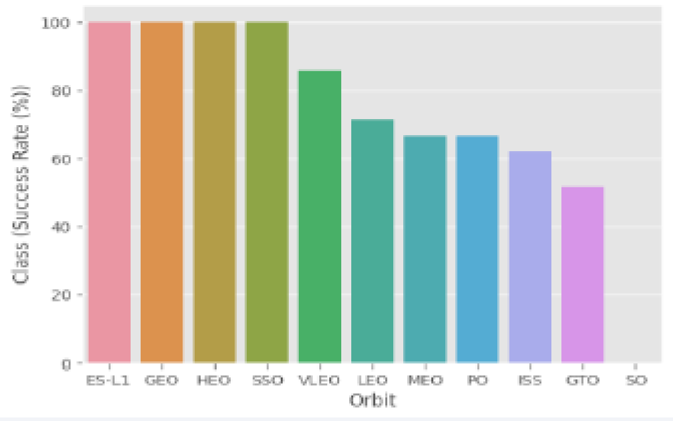
	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1

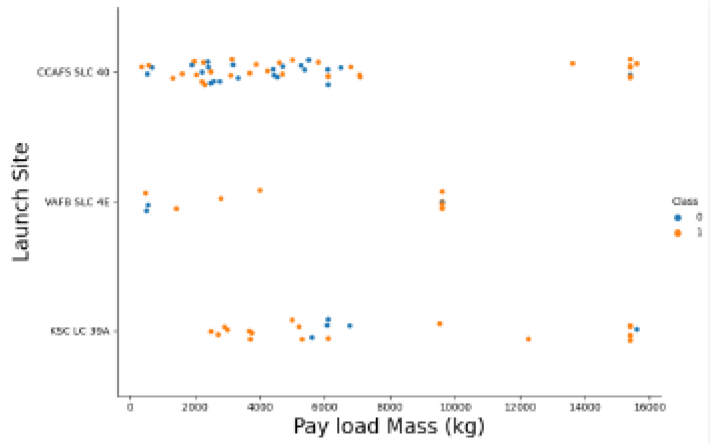
EDA with Data Visualization

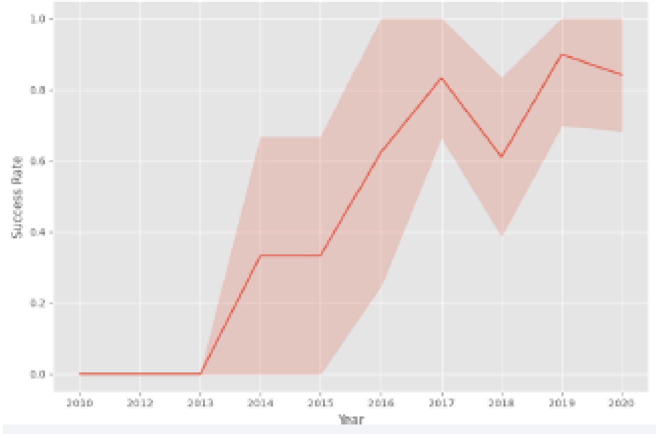
- Performed data Analysis and Feature Engineering using Pandas and Matplotlib.i.e.
    - Exploratory Data Analysis
    - Preparing Data Feature Engineering
- Used scatter plots to Visualize the relationship between Flight Number and Launch Site, Payload and Launch Site, Flight Number and Orbit type, Payload and Orbit type.
- Used Bar chart to Visualize the relationship between success rate of each orbit type
- Line plot to Visualize the launch success yearly trend.
- Here is the GitHub URL of your completed EDA with data visualization notebook, (https://github.com/Mayank-Bisht-2000/IBM-Capstone-Project/blob/master/IBM-DS0321EN-SkillsNetwork_labs_module_2_jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with Data Visualization









EDA with SQL

- The following SQL queries were performed for EDA
- Display the names of the unique launch sites in the space mission

```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
```

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

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

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

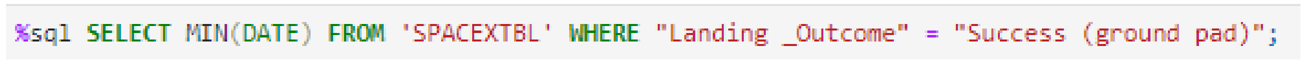
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) as "Total Payload Mass(Kgs)", Customer FROM 'SPACEXTBL' WHERE Customer = 'NASA (CRS)';
```

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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) as "Payload Mass Kgs", Customer, Booster_Version FROM 'SPACEXTBL' WHERE Booster_Version LIKE 'F9 v1.1%';
```

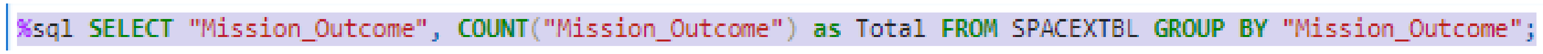
EDA with SQL (Cont….)

- List the date when the first successful landing outcome in ground pad was achieved



```
%sql SELECT MIN(DATE) FROM 'SPACEXTBL' WHERE "Landing _Outcome" = "Success (ground pad)";
```

- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000 (%sqlSELECT DISTINCT Booster_Version, Payload FROM SPACEXTBL WHERE "Landing _Outcome" = "Success (drone ship)" AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;)



```
%sql SELECT "Mission_Outcome", COUNT("Mission_Outcome") as Total FROM SPACEXTBL GROUP BY "Mission_Outcome";
```

- List the total number of successful and failure mission outcomes
- Here is the GitHub URL of your completed EDA with SQL notebook. (https://github.com/Mayank-Bisht-2000/IBM-Capstone-Project/blob/master/jupyter-labs-eda-sql-coursera_sqllite.ipynb)

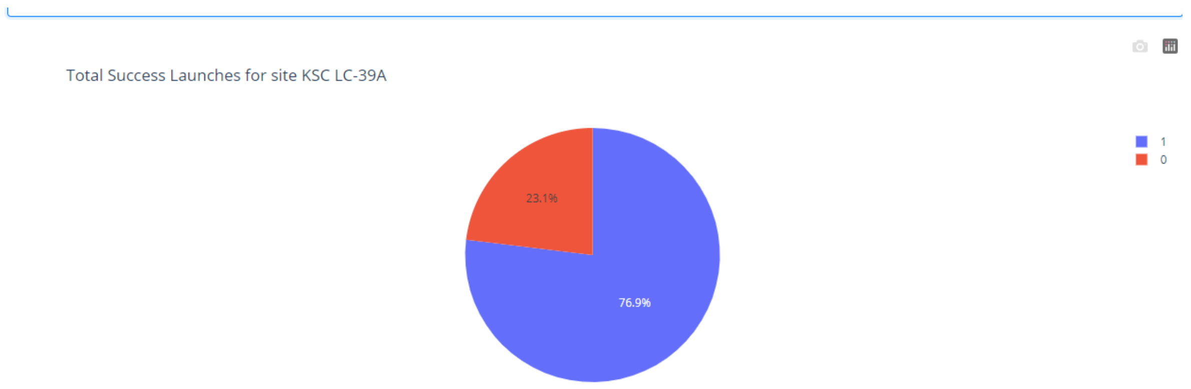
Build an Interactive Map with Folium

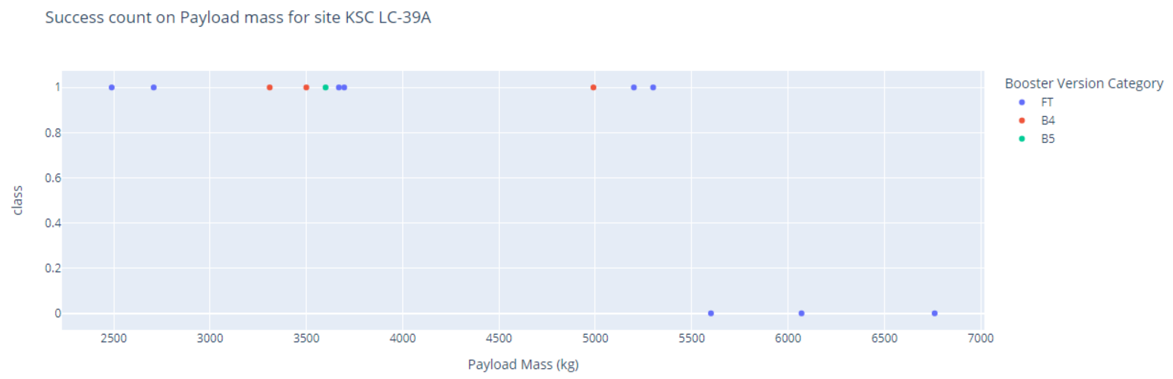
- Created folium map to marked all the launch sites, and created map objects such as markers, circles, lines to mark the success or failure of launches for each launch site.
- Created a launch set outcomes (failure=0 or success=1).
- Here is the GitHub URL of the completed interactive map with Folium map, as an external reference and peer-review purpose (https://github.com/Mayank-Bisht-2000/IBM-Capstone-Project/blob/master/IBM-DS0321EN-SkillsNetwork_labs_module_3_lab_jupyter_launch_site_location.jupyterlite.ipynb)

Build a Dashboard with Plotly Dash

- Built an interactive dashboard application with Plotly dash by:
- Adding a Launch Site Drop-down Input Component
- Adding a callback function to render success-pie-chart based on selected site dropdown
- Adding a Range Slider to Select Payload
- Adding a callback function to render the success-payload-scatter-chart scatter plot
- Here is the GitHub URL of your completed Plotly Dash lab (https://github.com/Mayank-Bisht-2000/IBM-Capstone-Project/blob/master/spacex_dash_app.py)

SpaceX Dash App





Predictive Analysis (Classification)

- Summary of how I built, evaluated, improved, and found the best performing classification model
- After loading the data as a Pandas Dataframe, I set out to perform exploratory Data Analysis and determine Training Labels by;

1. creating a NumPy array from the column Class in data, by applying the method to_numpy() then assigned it to the variable Y as the outcome variable.
2. Then standardized the feature dataset (x) by transforming it using preprocessing. StandardScaler() function from Sklearn.
3. After which the data was split into training and testing sets using the function train_test_split from sklearn.model_selection with the test_size parameter set to 0.2 and random_state to 2.

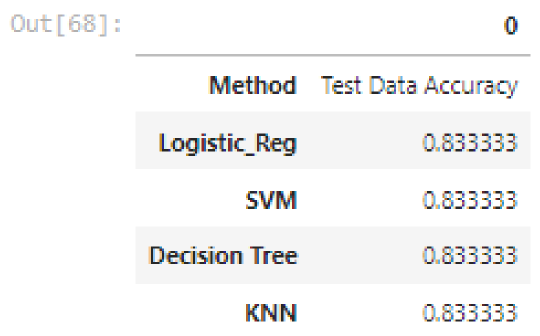
Predictive Analysis (Classification)

- In order to find the best ML model/ method that would performs best using the test data between SVM, Classification Trees, k nearest neighbors and Logistic Regression;

1. First created an object for each of the algorithms then created a GridSearchCV object and assigned them a set of parameters for each model.
2. For each of the models under evaluation, the GridsearchCV object was created with cv=10, then fit the training data into the GridSearch object for each to Find best Hyperparameter.
3. After fitting the training set, we output GridSearchCV object for each of the models, then displayed the best parameters using the data attribute best_params_ and the accuracy on the validation data using the data attribute best_score_.
4. Finally using the method score to calculate the accuracy on the test data for each model and plotted a confussion matrix for each using the test and predicted outcomes.

Predictive Analysis (Classification)

- The table below shows the test data accuracy score for each of the methods comparing them to show which performed best using the test data between SVM, Classification Trees, k nearest neighbors and Logistic Regression;



```
Out[68]:                          0
                 Method  Test Data Accuracy
           Logistic_Reg            0.833333
                    SVM            0.833333
          Decision Tree            0.833333
                    KNN            0.833333
```

- GitHub URL of the completed predictive analysis lab (https://github.com/Mayank-Bisht-2000/IBM-Capstone-Project/blob/master/IBM-DS0321EN-SkillsNetwork_labs_module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb)

Results

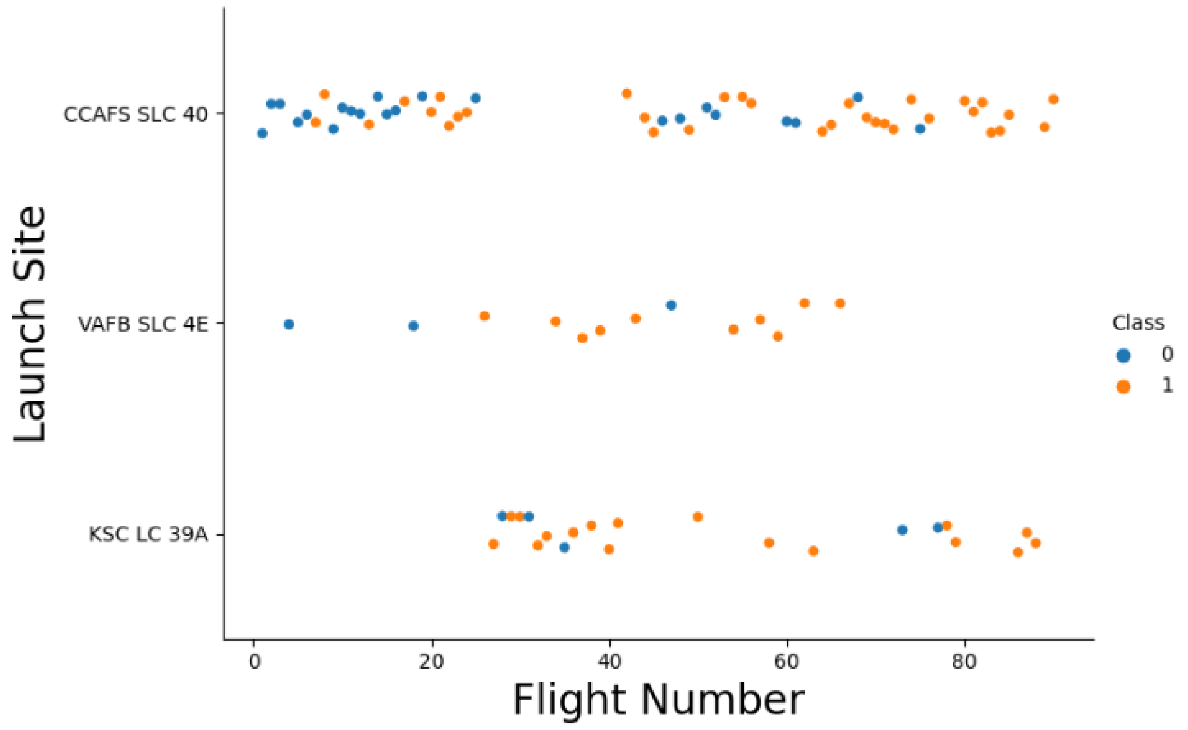
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

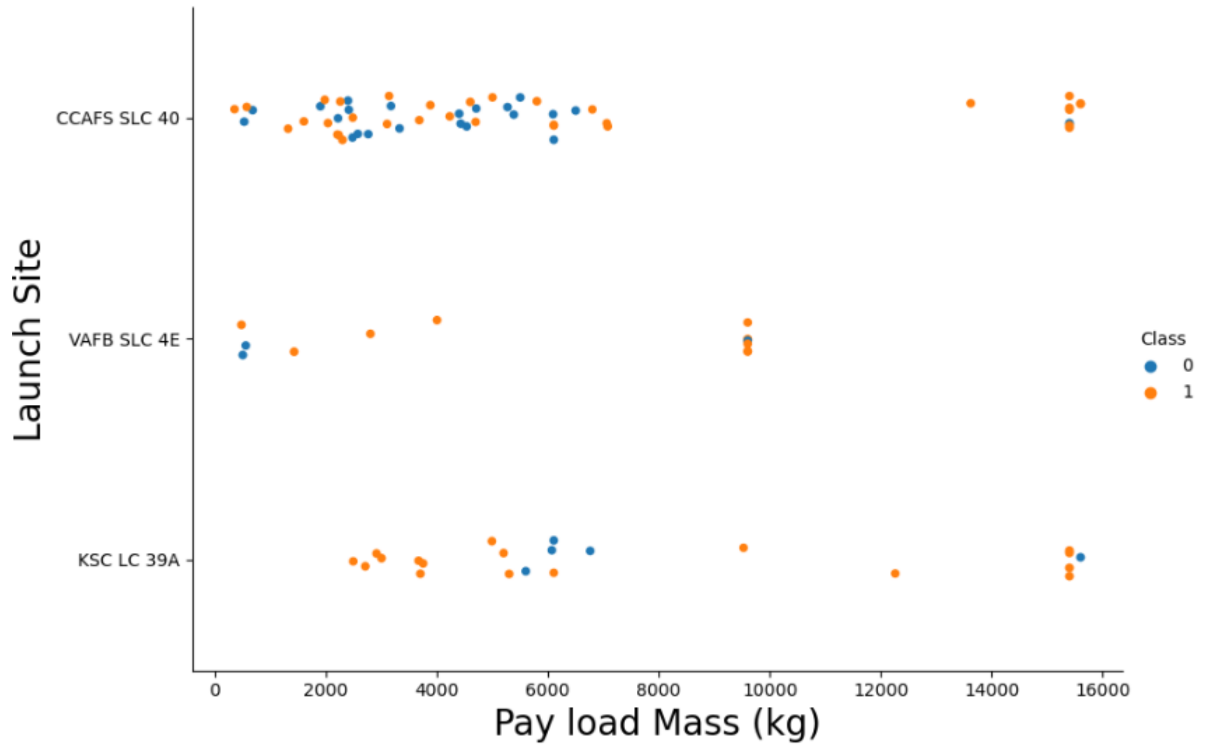
Flight Number vs. Launch Site

A scatter plot of Flight Number vs. Launch Site



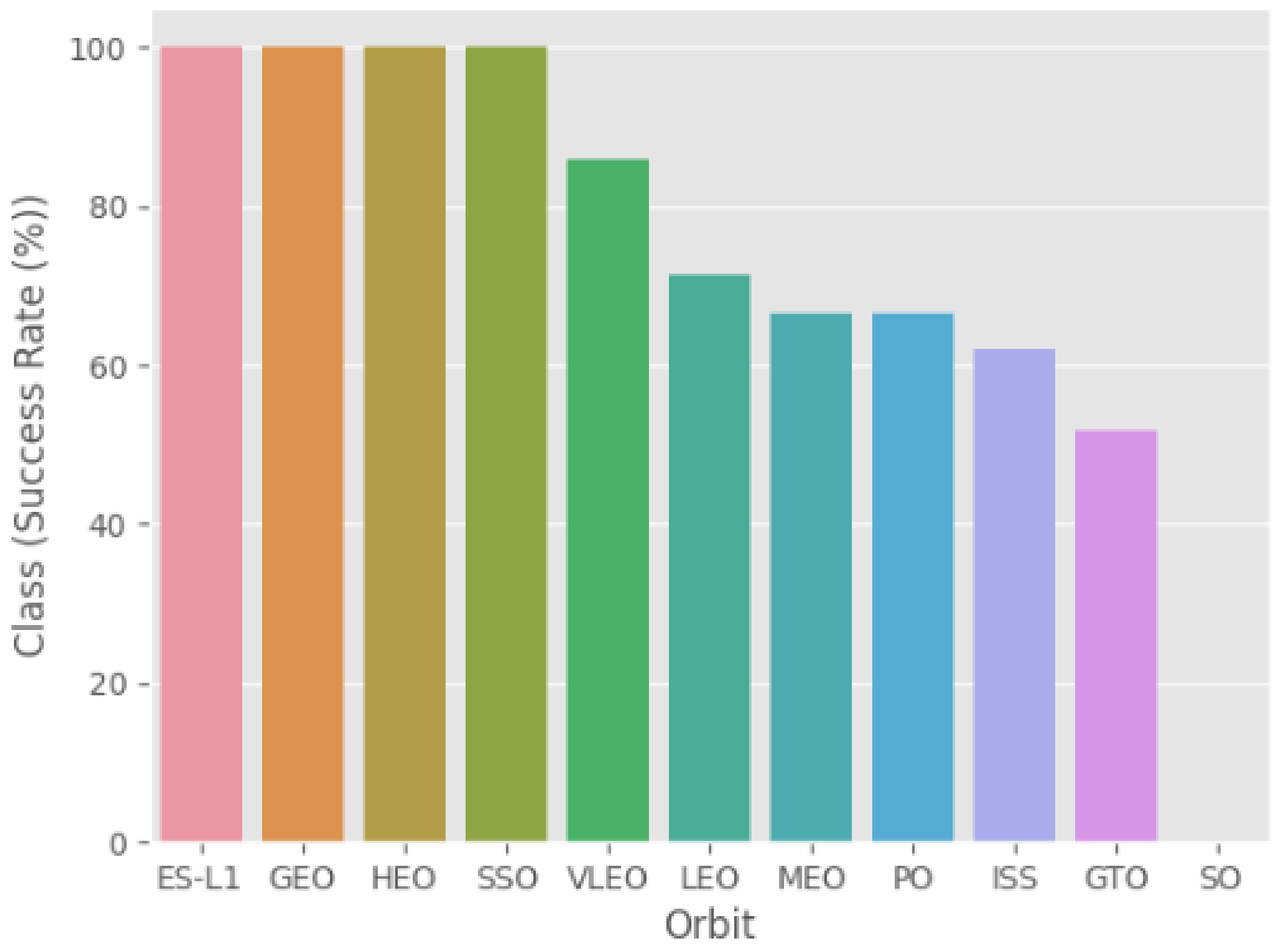
Payload vs. Launch Site

A scatter plot of Payload vs. Launch Site



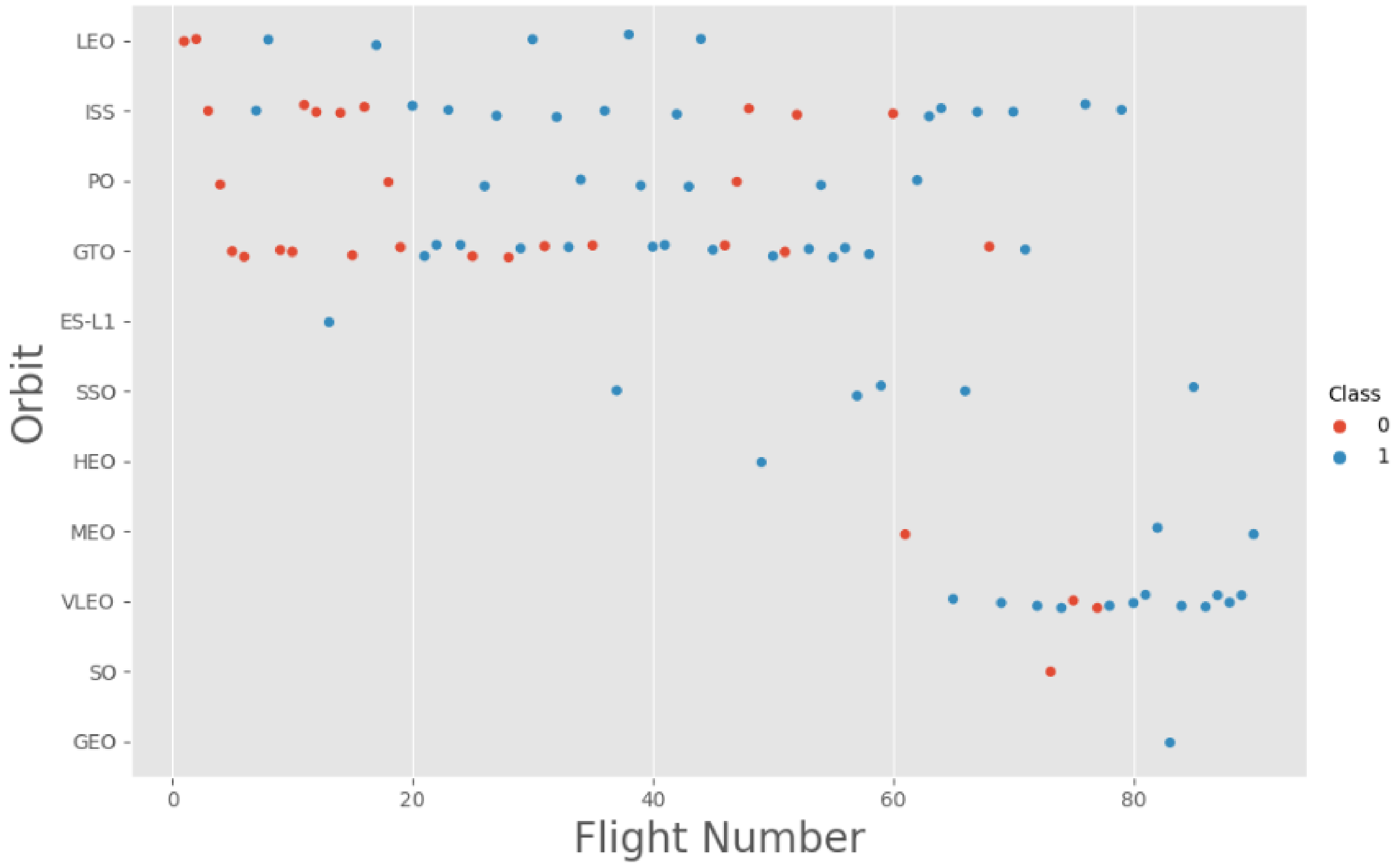
Success Rate vs. Orbit Type

- Show a bar chart for the success rate of each orbit type



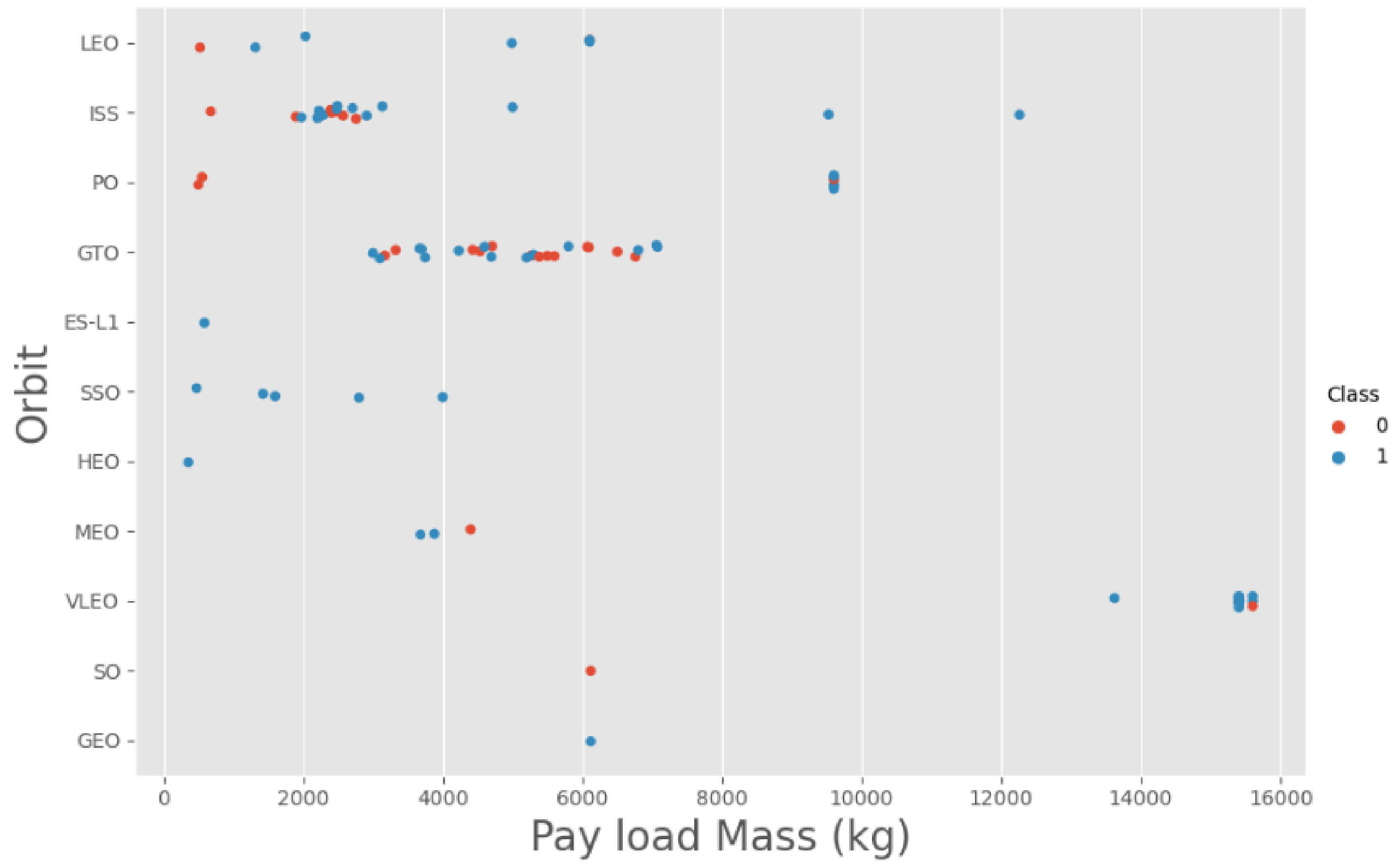
Flight Number vs. Orbit Type

- A scatter point of Flight number vs. Orbit type



Payload vs. Orbit Type

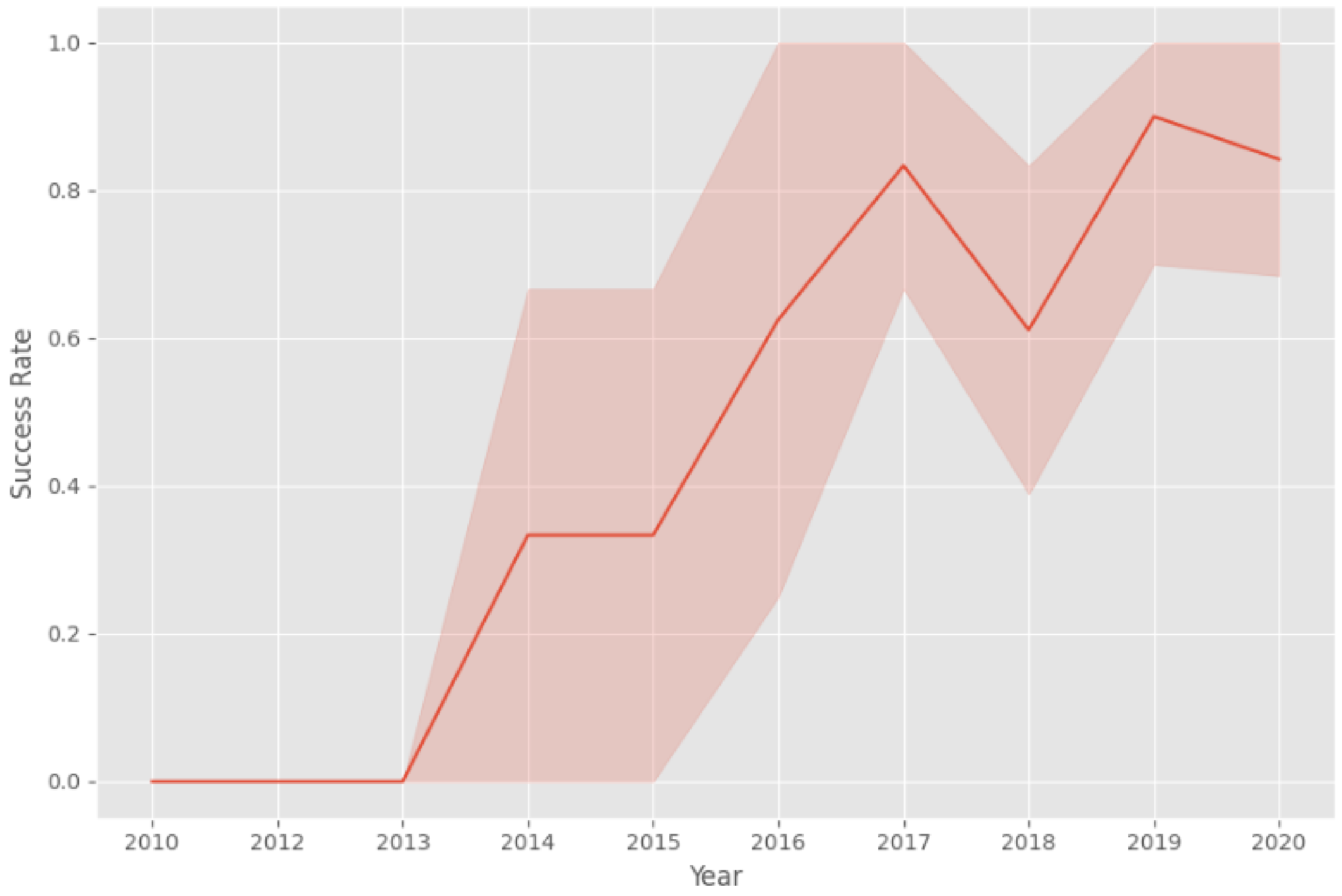
- With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.
- •However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) both have near equal chances



Launch Success Yearly Trend

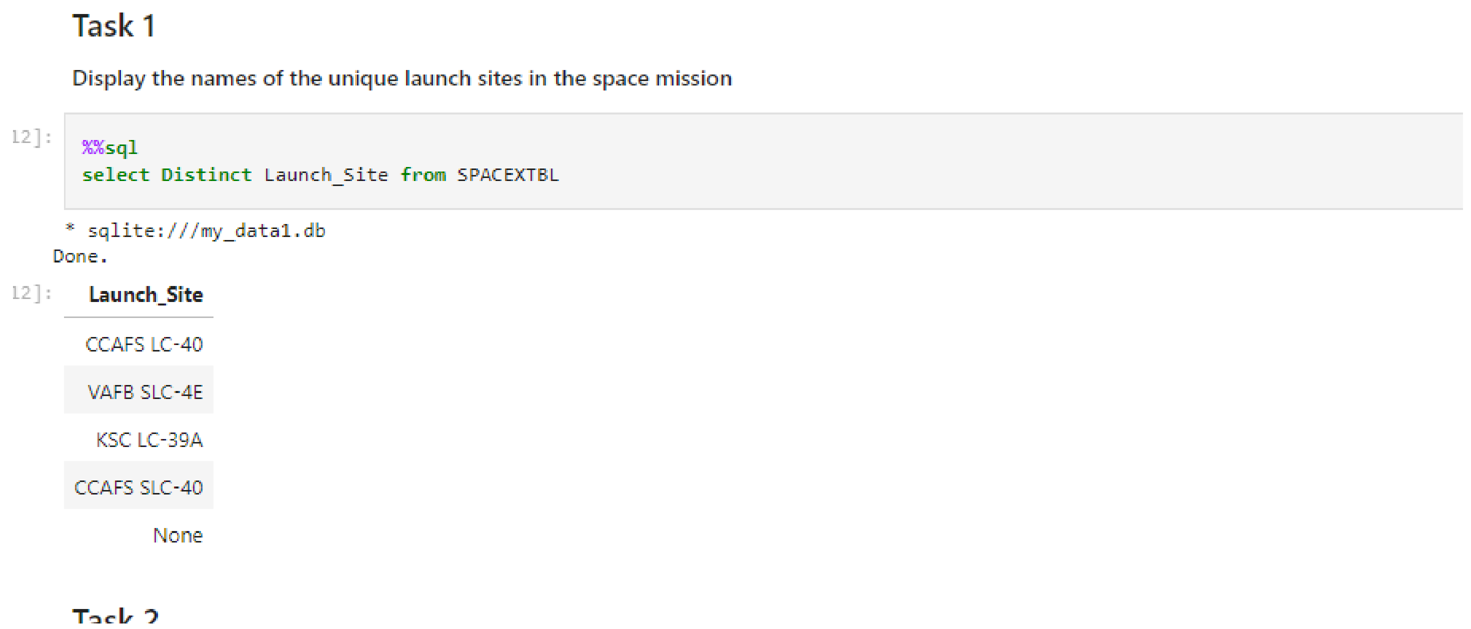
- Since 2013, the success rate kept going up till 2020

A line chart of yearly average success rate



All Launch Site Names

- Find the names of the unique launch sites
- Used ‘SELECT DISTINCT’ statement to return only the unique launch sites from the ‘LAUNCH_SITE’ column of the SPACEXTBL table



Task 1

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

```
12]: %%sql
     select Distinct Launch_Site from SPACEXTBL
```

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

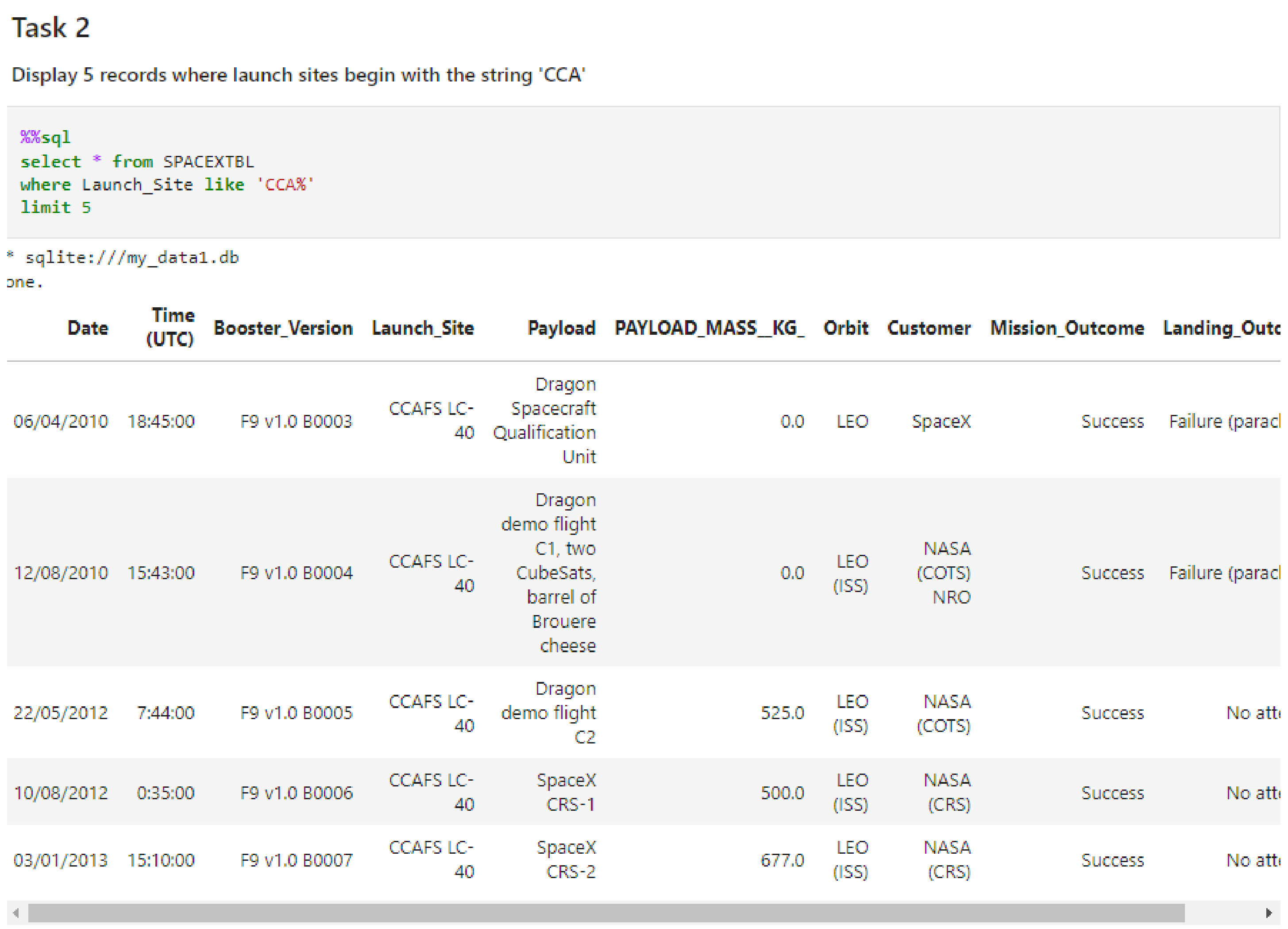
12]:

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

Task 2

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`
- Used ‘LIKE’ command with ‘%’ wildcard in ‘WHERE’ clause to select and display a table of all records where launch sites begin with the string 'CCA'



Task 2

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

```
%%sql
select * from SPACEXTBL
where Launch_Site like 'CCA%'
limit 5
```

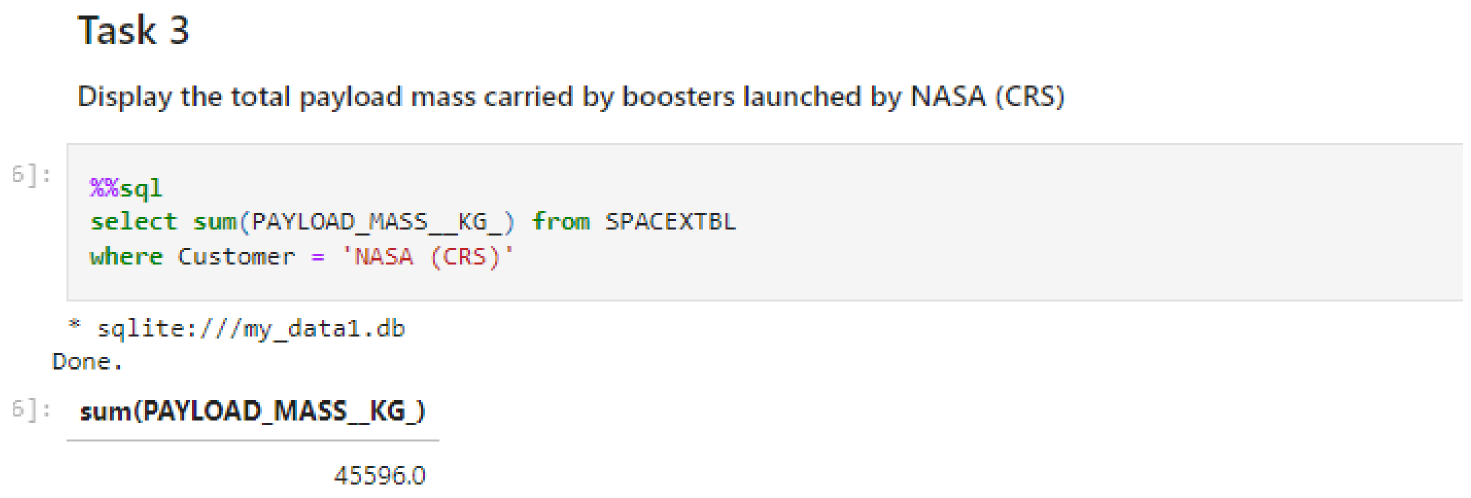
```
* sqlite:///my_data1.db
one.
```

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

Total Payload Mass

- Calculate and Display the total payload carried by boosters from NASA

- Used the 'SUM()' function to return and dispaly the total sum of 'PAYLOAD_MASS_KG' column for Customer 'NASA(CRS)'



Task 3

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

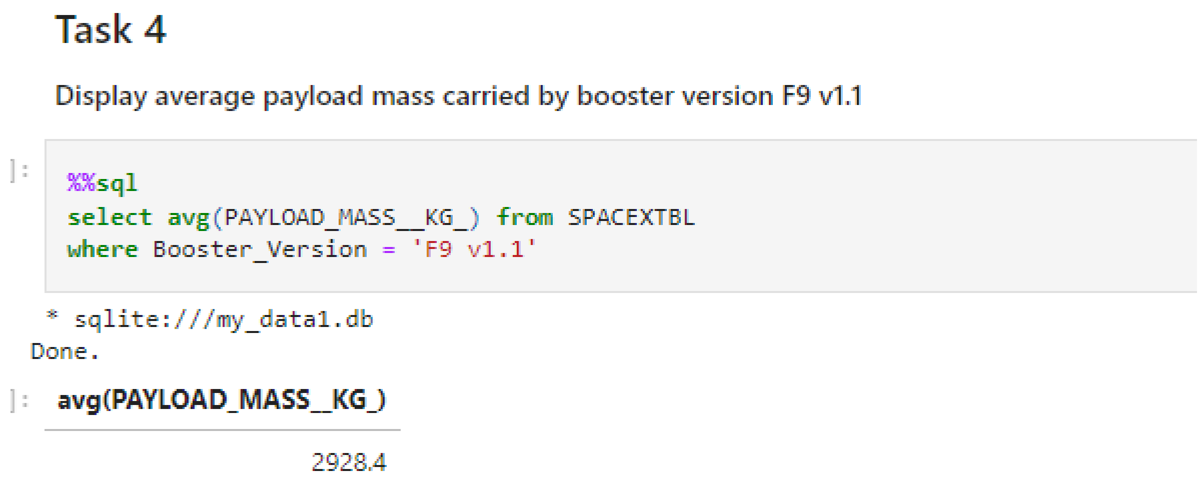
```
6]:  %%sql
     select sum(PAYLOAD_MASS__KG_) from SPACEXTBL
     where Customer = 'NASA (CRS)'

      * sqlite:///my_data1.db
     Done.

6]:  sum(PAYLOAD_MASS__KG_)
                    45596.0
```

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1
- Used the ‘AVG()’ function to return and display the average payload mass carried by booster version F9 v1.1



Task 4

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

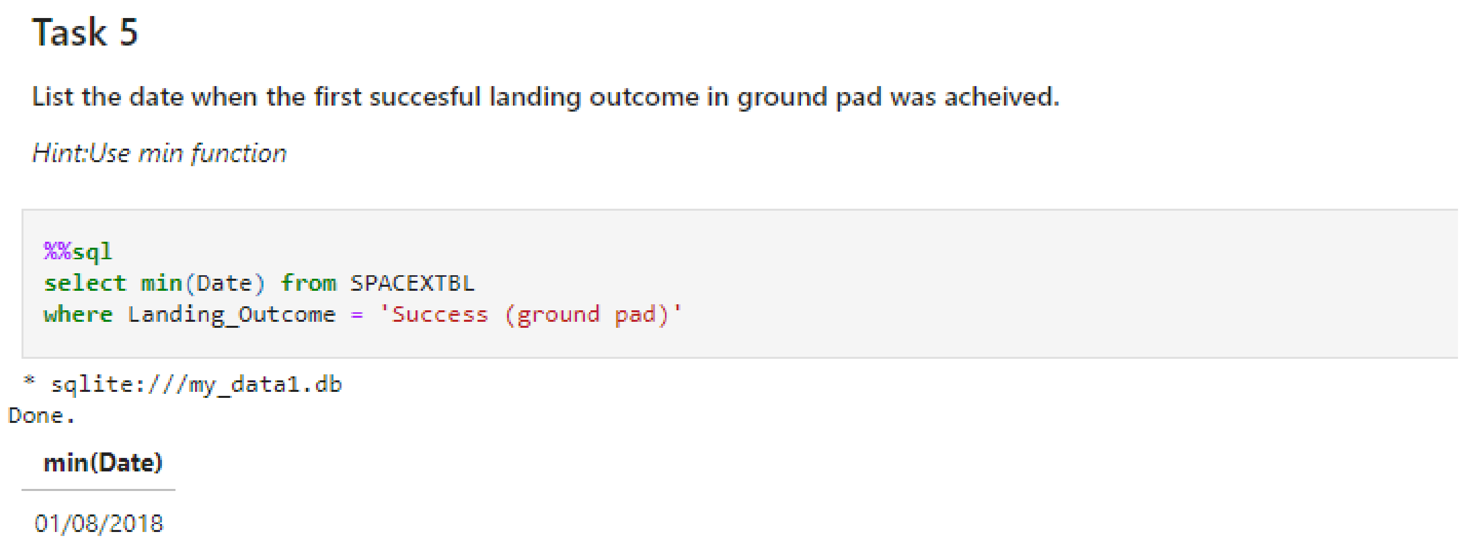
```
]: %%sql
   select avg(PAYLOAD_MASS__KG_) from SPACEXTBL
   where Booster_Version = 'F9 v1.1'

 * sqlite:///my_data1.db
Done.

]: avg(PAYLOAD_MASS__KG_)
                   2928.4
```

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad
- Used the 'MIN()' function to return and display the first (oldest) date when first successful landing outcome on ground pad 'Success (ground pad)'happened



Task 5

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

Hint:Use min function

```
%%sql
select min(Date) from SPACEXTBL
where Landing_Outcome = 'Success (ground pad)'
```

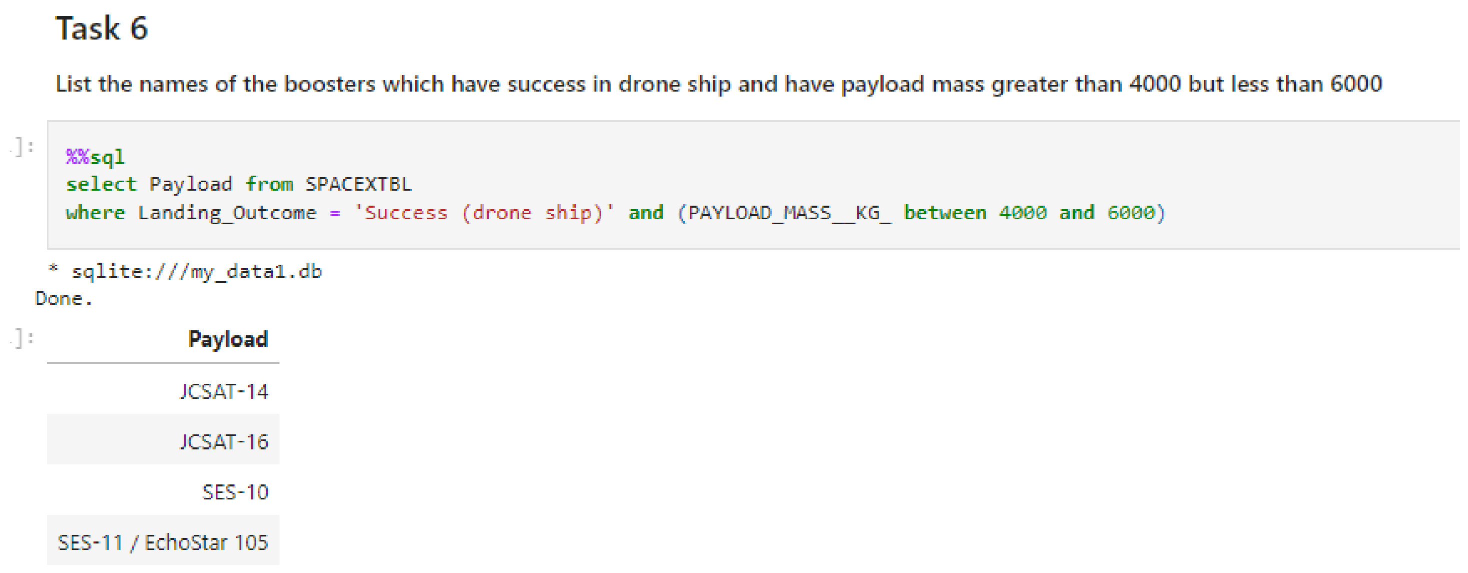
```
 * sqlite:///my_data1.db
Done.
```

min(Date)
01/08/2018

Successful Drone Ship Landing with Payload between 4000 and 6000

- List of Boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

- Used 'Select Distinct' statement to return and list the 'unique' names of boosters with operators >4000 and <6000 to only list booster with payloads between 4000-6000 with landing outcome of 'Success (drone ship)'.



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

```
%%sql
select Payload from SPACEXTBL
where Landing_Outcome = 'Success (drone ship)' and (PAYLOAD_MASS__KG_ between 4000 and 6000)
```

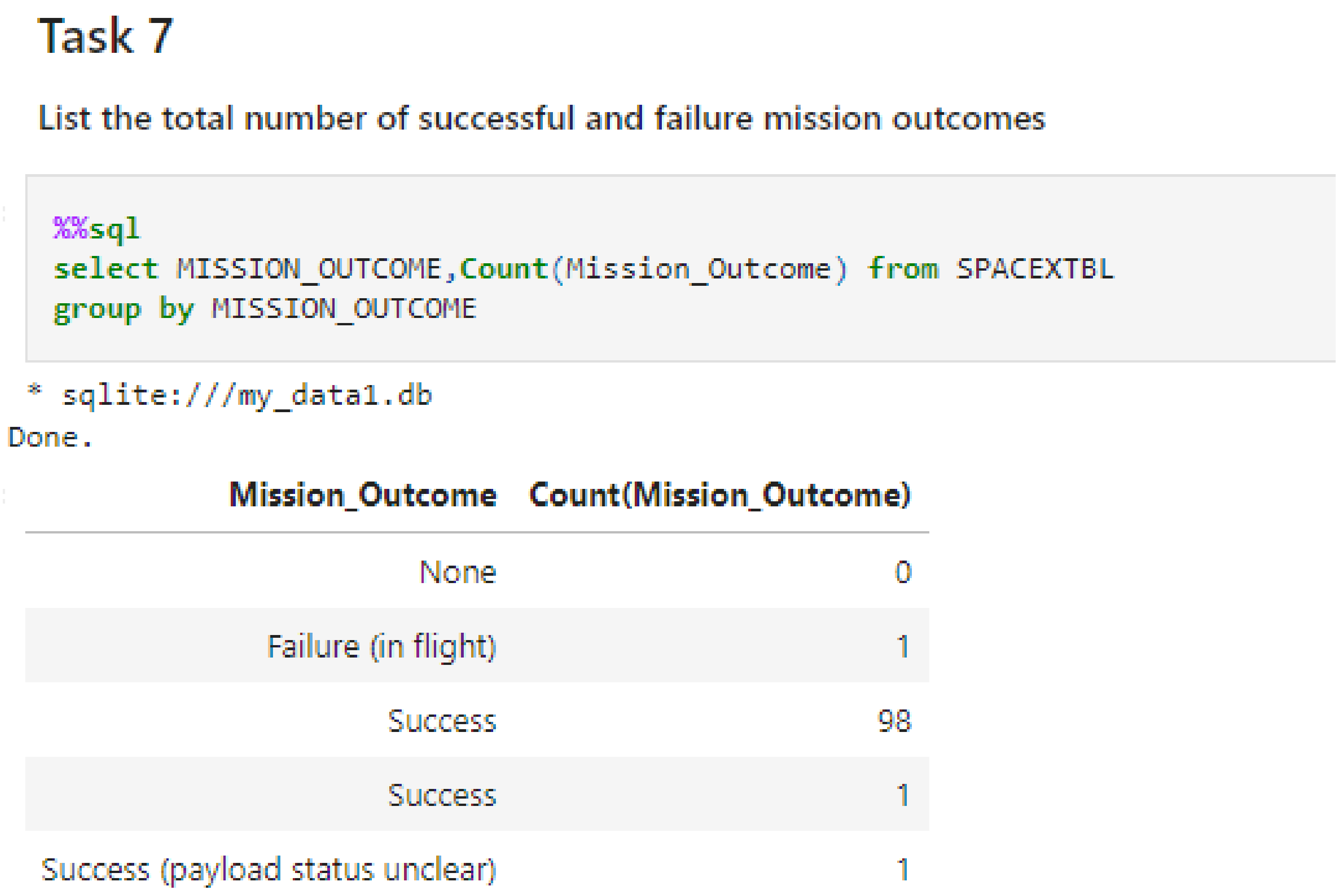
```
 * sqlite:///my_data1.db
Done.
```

Payload
JCSAT-14
JCSAT-16
SES-10
SES-11 / EchoStar 105

Total Number of Successful and FailureMission Outcomes

- Calculate the total number of successful and failure mission outcomes

- Used the ‘COUNT()’ together with the ‘GROUP BY’ statement to return total number of missions outcomes



Task 7

List the total number of successful and failure mission outcomes

```
%%sql
select MISSION_OUTCOME,Count(Mission_Outcome) from SPACEXTBL
group by MISSION_OUTCOME
```

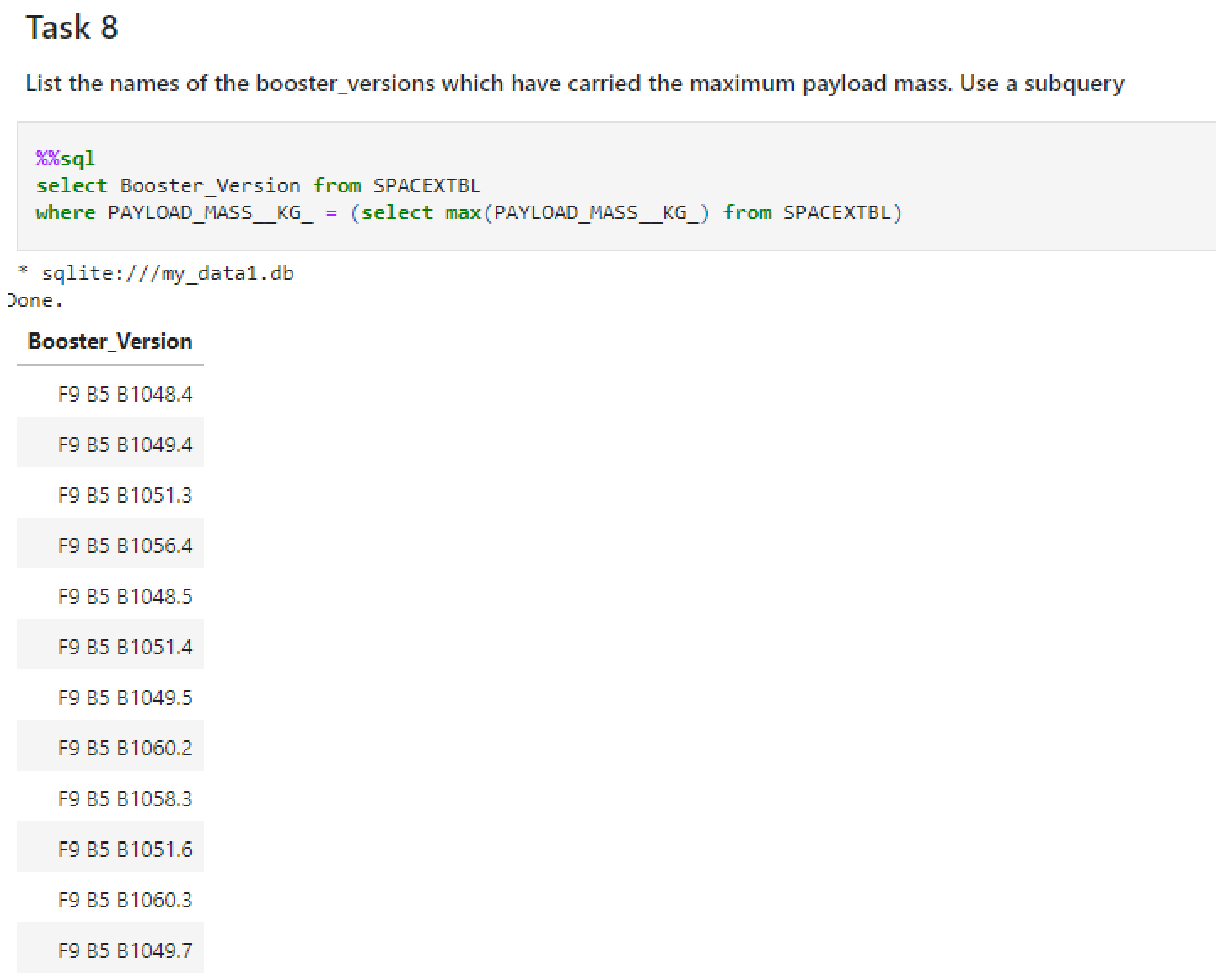
```
 * sqlite:///my_data1.db
Done.
```

Mission_Outcome	Count(Mission_Outcome)
None	0
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

- List of the boosters which have carried the maximum payload mass

- Using a Subquery to return and pass the Max payload and used it list all the boosters that have carried the Max payload of 15600kgs



Task 8

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

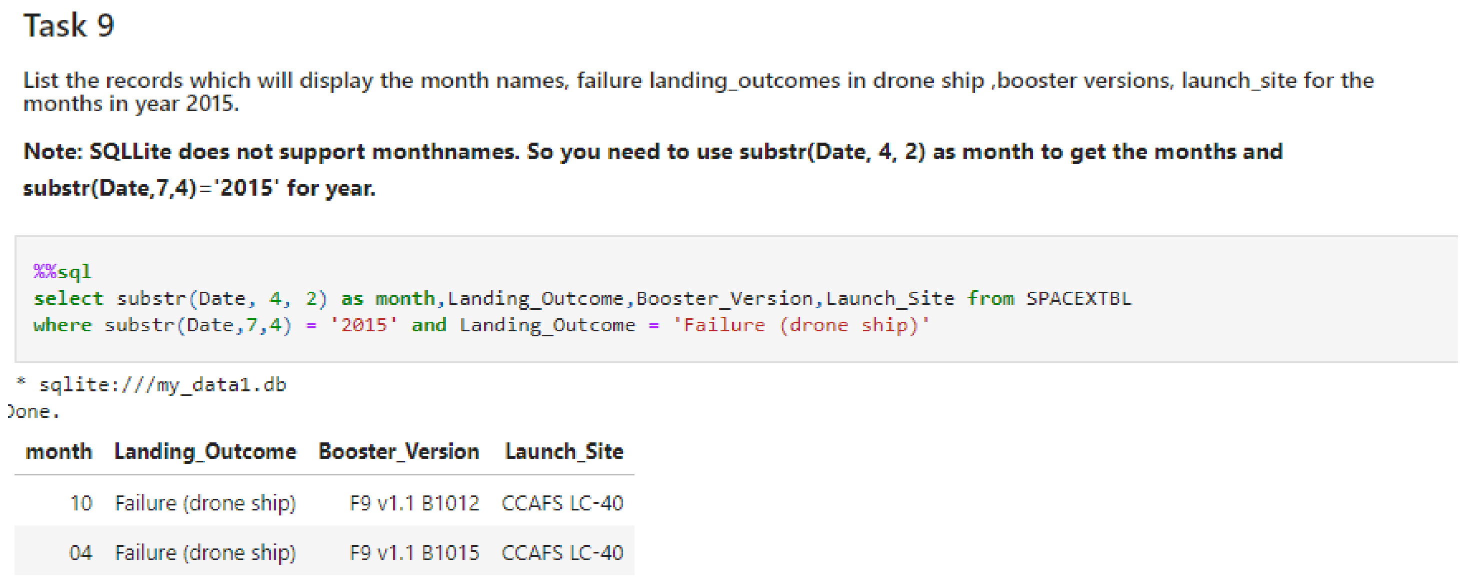
```
%%sql
select Booster_Version from SPACEXTBL
where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTBL)
```

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

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

- List of failed landing outcomes in drone ship, with their booster versions, and launch site names in 2015
- Used the ‘subsrt()’ in the select statement to get the month and year from the date column where substr(Date,7,4)='2015' for year and Landing_outcome was ‘Failure (drone ship’) and return the records matching the filter.



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.

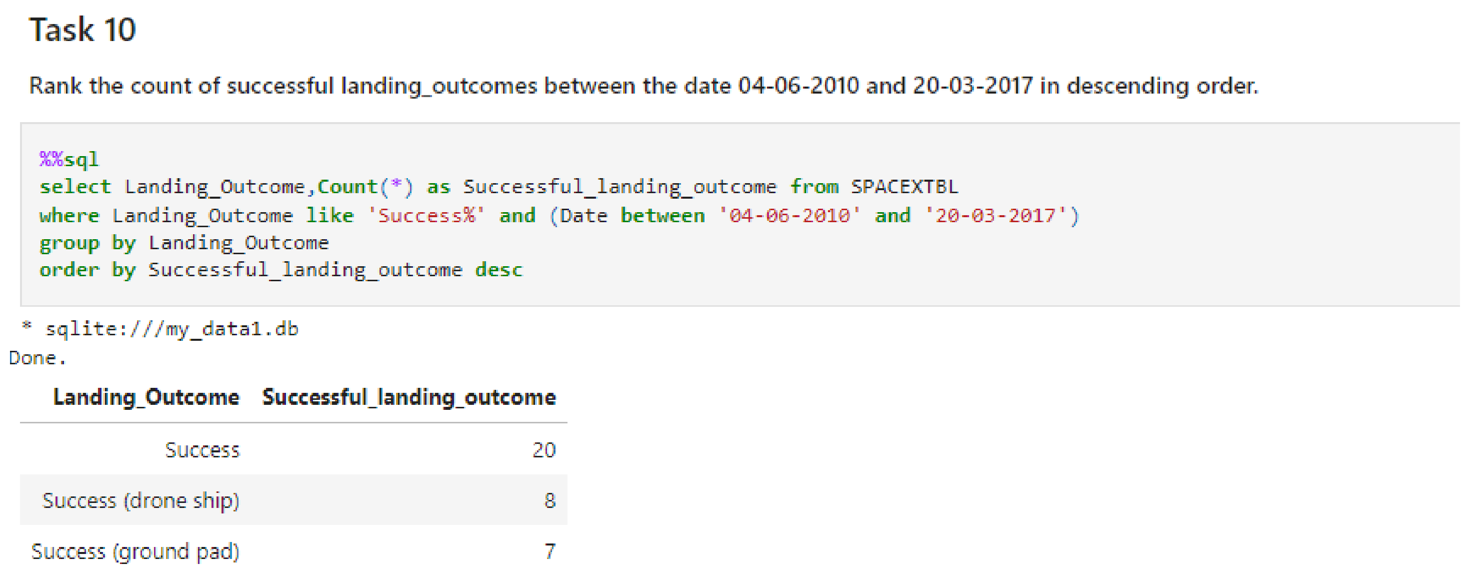
```
%%sql
select substr(Date, 4, 2) as month,Landing_Outcome,Booster_Version,Launch_Site from SPACEXTBL
where substr(Date,7,4) = '2015' and Landing_Outcome = 'Failure (drone ship)'
```

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

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

- 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



Task 10

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

```
%%sql
select Landing_Outcome,Count(*) as Successful_landing_outcome from SPACEXTBL
where Landing_Outcome like 'Success%' and (Date between '04-06-2010' and '20-03-2017')
group by Landing_Outcome
order by Successful_landing_outcome desc
```

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

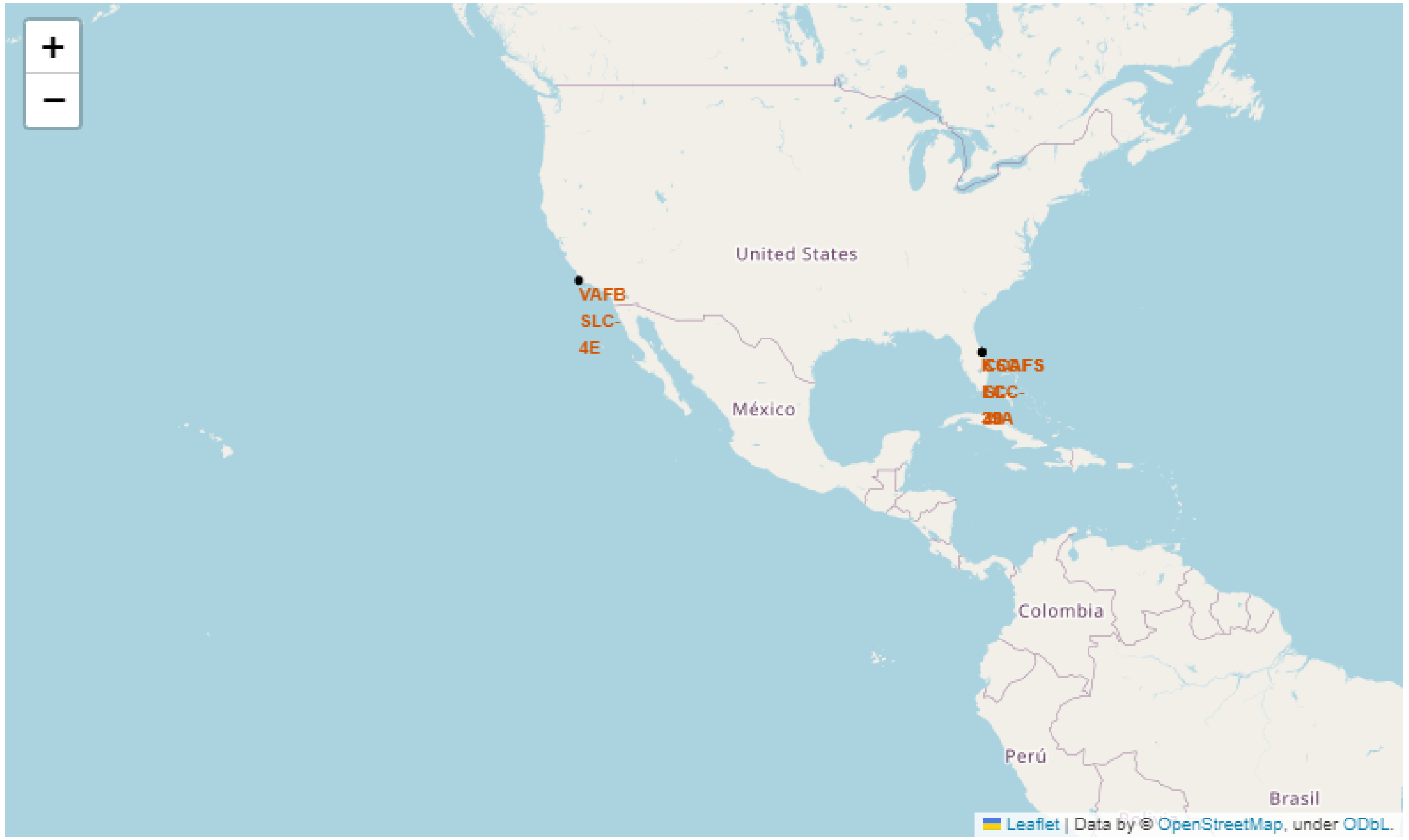
Landing_Outcome	Successful_landing_outcome
Success	20
Success (drone ship)	8
Success (ground pad)	7

Section 4

Launch Sites Proximities Analysis

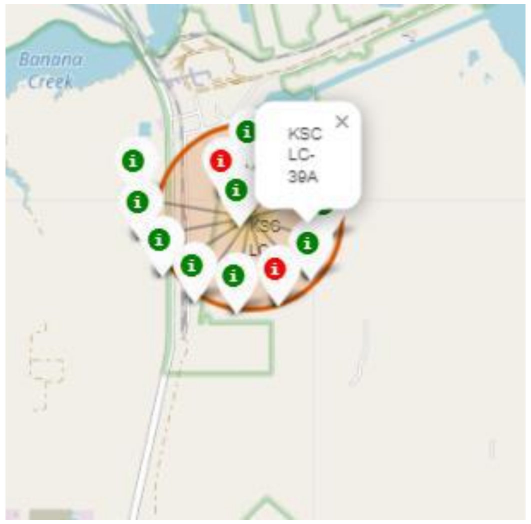
Markers of all launch sites on global map

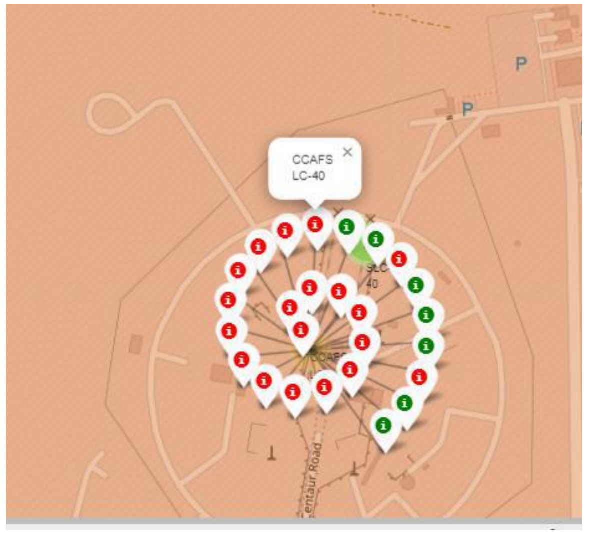
- All launch sites are in proximity to the Equator, (located southwards of the US map). Also, all the launch sites are in very close proximity to the coast

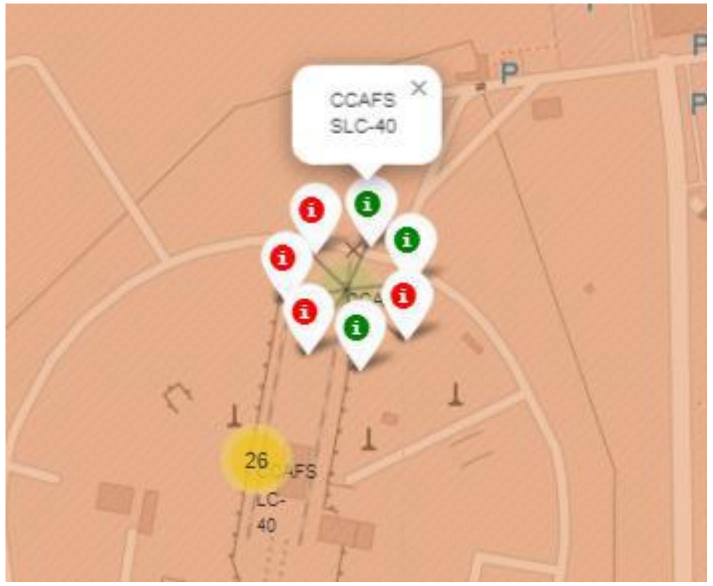


Launch outcomes for each site on the map With Color Markers

Florida Sites





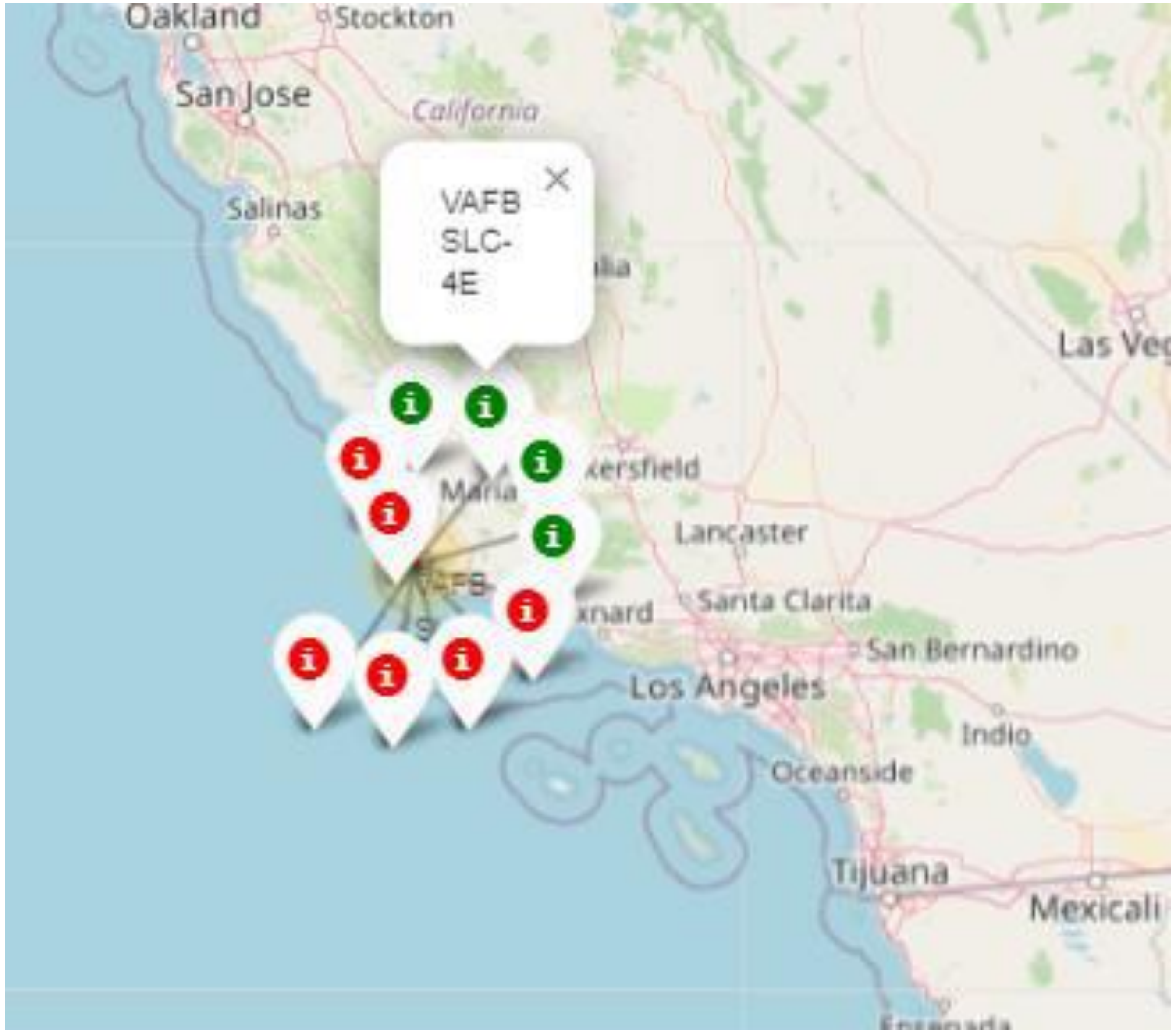


- In the Eastern coast (Florida) Launch site KSC LC-39A has relatively high success rates compared to CCAFS SLC-40 & CCAFS LC-40.

West Coast/ California

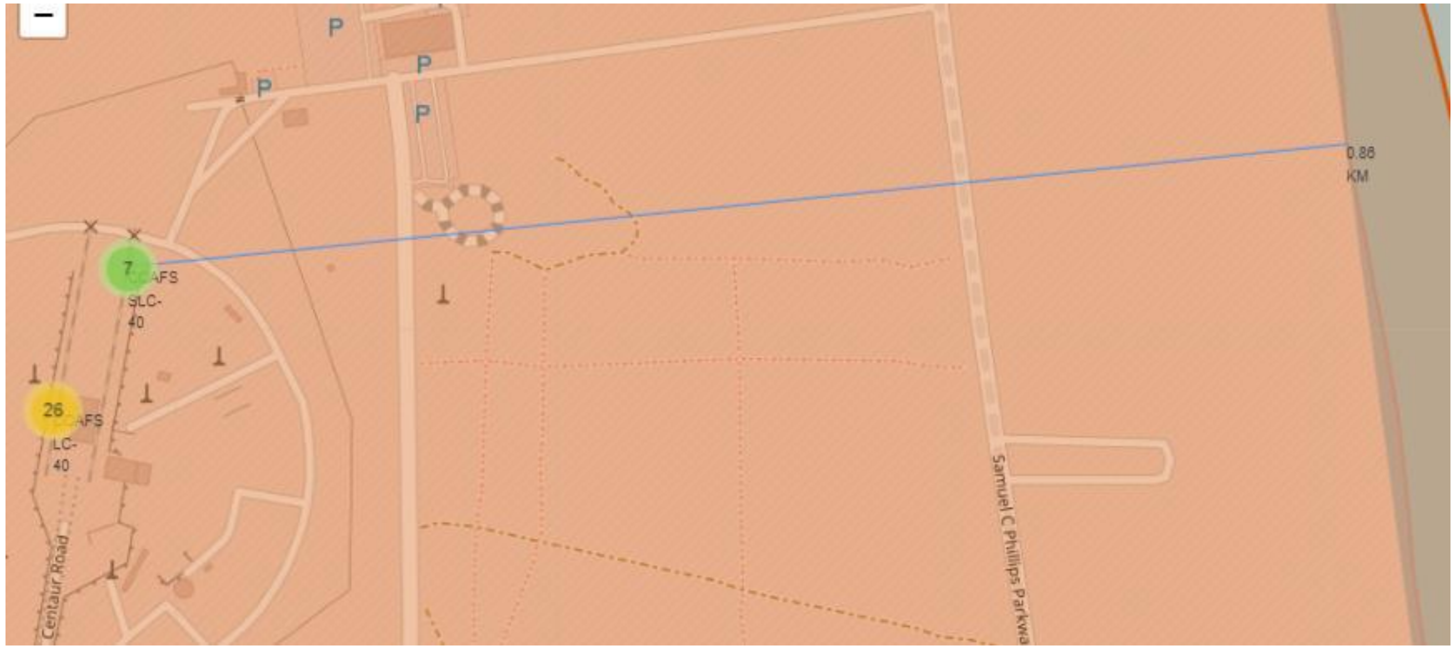
Launch outcomes for each site on the map With Color Markers

- In the West Coast (California) Launch site VAFB SLC-4E has relatively lower success rates 4/10 compared to KSC LC-39A launch site in the Eastern Coast of Florida.



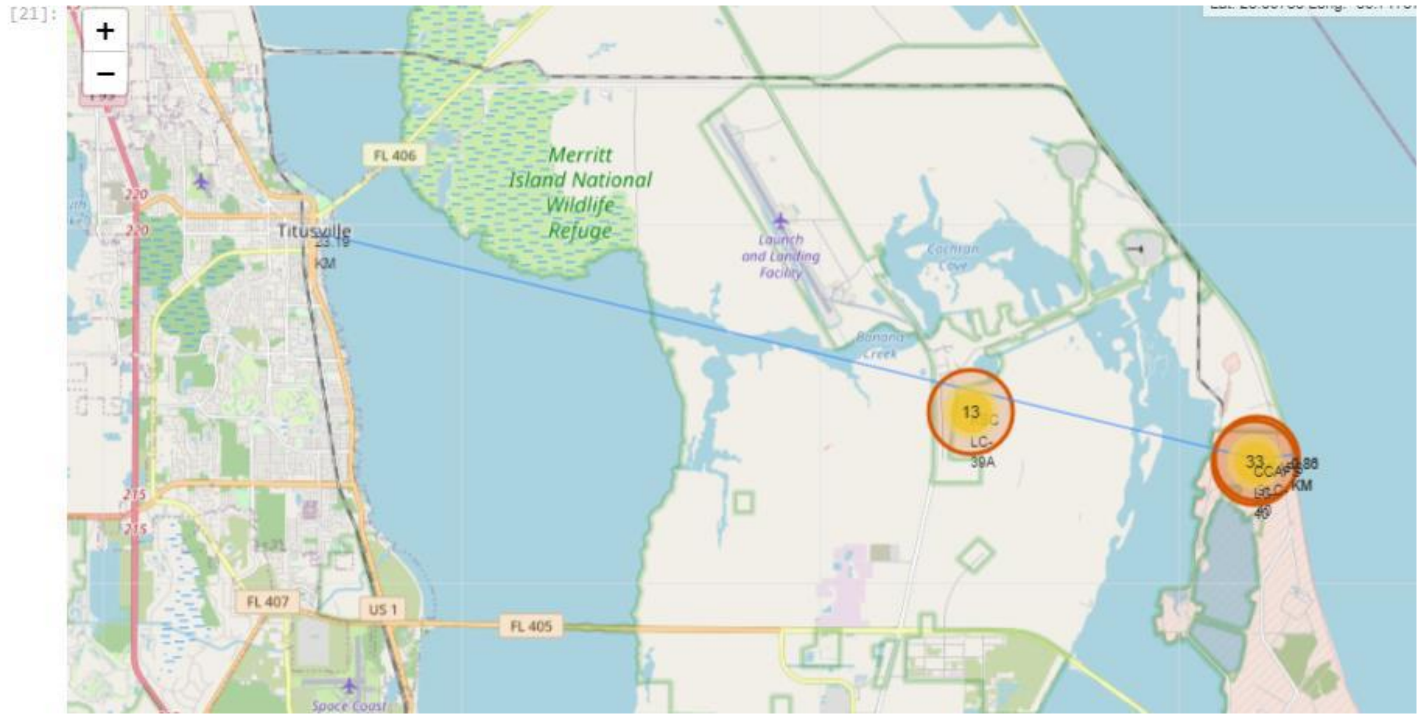
Distances between a launch site to its proximities

- Launch site CCAFS SLC-40proximity to coastline is 0.86km



Distances between a launch site to its proximities

- Launch site CCAFS SLC-40 closest to highway (Washington Avenue) is 23.19km

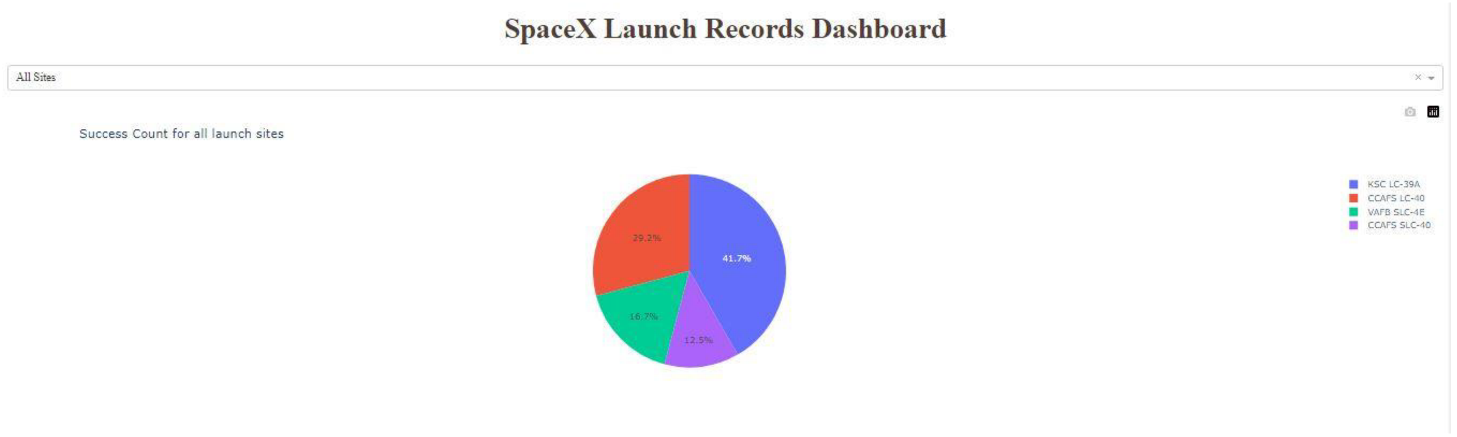


Section 5

Build a Dashboard with Plotly Dash

Pie-Chart for launch success count for all sites

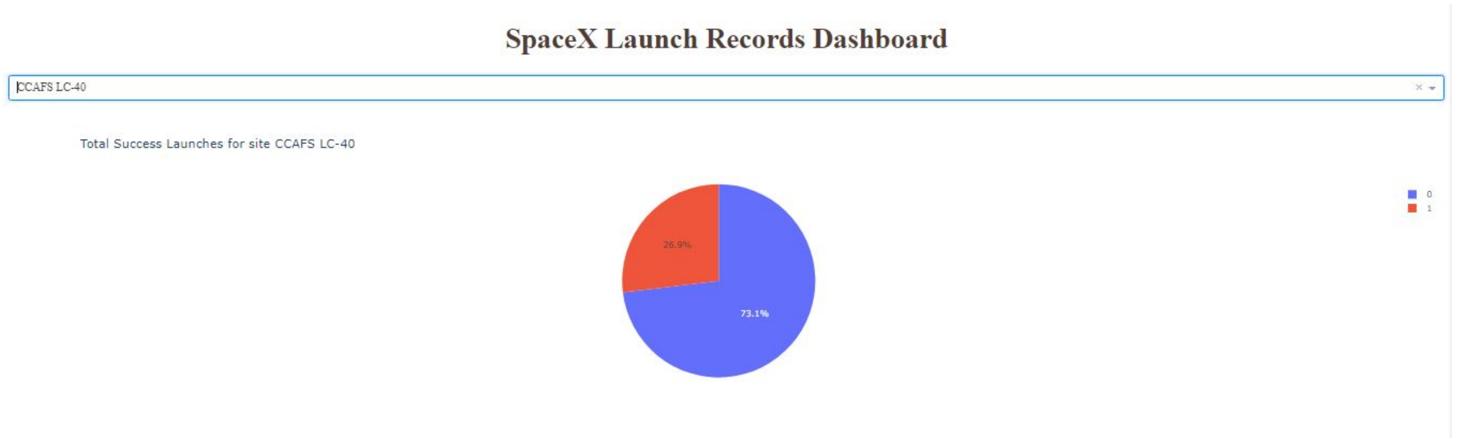
- Launch site KSC LC-39A has the highest launch success rate at 42% followed by CCAFS LC-40 at 29%, VAFB SLC-4E at 17% and lastly launch site CCAFS SLC-40 with a success rate of 13%


SpaceX Launch Records Dashboard

All Sites

Pie chart for the launch site with 2nd highest launch success ratio

- Launch site CCAFS LC-40 had the 2ndhighest success ratio of 73% success against 27% failed launches



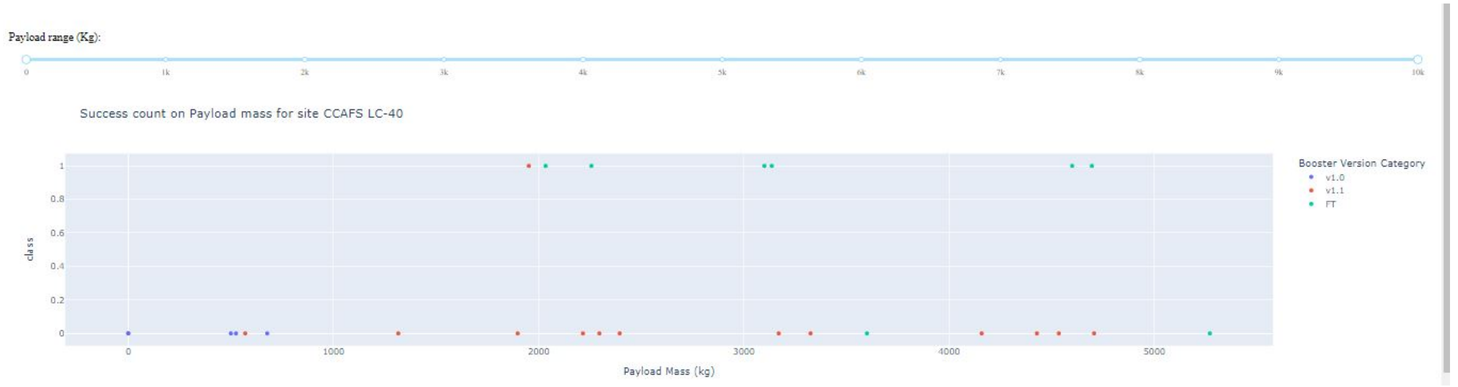
SpaceX Launch Records Dashboard

CCAFS LC-40

Total Success Launches for site CCAFS LC-40

Payload vs. Launch Outcome scatter plot for all sites

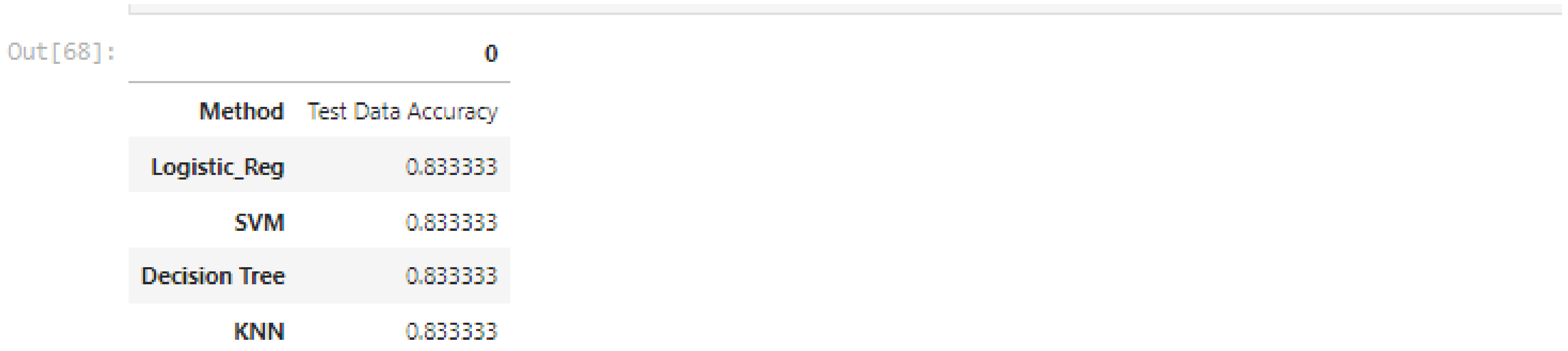
- For Launch site CCAFS LC-40 the booster version FT has the largest success rate from a payload mass of >2000kg



Section 6

Predictive Analysis (Classification)

Classification Models Accuracy



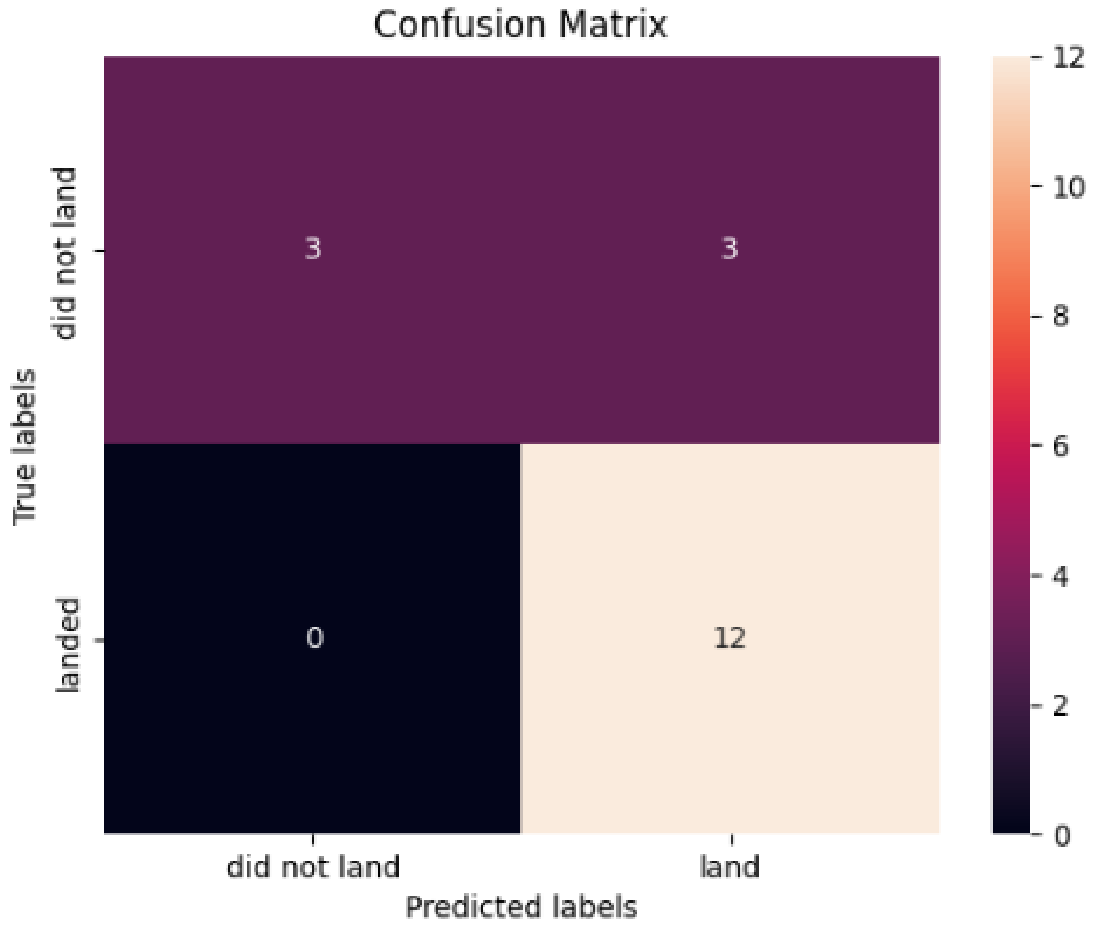
Out[68]:

	0
Method	Test Data Accuracy
Logistic_Reg	0.833333
SVM	0.833333
Decision Tree	0.833333
KNN	0.833333

All the methods perform equally on the test data: i.e. They all have the same accuracy of 0.833333 on the test Data

Confusion Matrix

- All the 4-classification model had the same confusion matrixes and were able equally distinguish between the different classes. The major problem is false positives for all the models.



Conclusions

- Different launch sites have different success rates. CCAFS LC-40, has a success rate of 60 %, while KSC LC-39A and VAFB SLC 4E has a success rate of 77%.
- We can deduce that, as the flight number increases in each of the 3 launch sites, so does the success rate. For instance, the success rate for the VAFB SLC 4E launch site is 100% after the Flight number 50. Both KSC LC 39A and CCAFS SLC 40 have a 100% success rates after 80th flight
- If you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launch site, there are no rockets launched for heavy payload mass(greater than 10000).
- Orbits ES-L1, GEO, HEO & SSO have the highest success rates at 100%, with SO orbit having the lowest success rate at ~50%. Orbit SO has 0% success rate.
- LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit

Conclusions Cont.….

- With heavy payloads the successful landing or positive landing rate are more for Polar LEO and ISS. However, for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here
- And finally the success rate since 2013 kept increasing till 2020.

Thank you!