IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Azy.m>
<20220523>

Outline

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

Executive Summary

Summary of methodologies

- Data Collection through API
- Data Collection with Web Scraping
- Data Wrangling
- Exploratory Data Analysis with SQL
- Exploratory Data Analysis with Data Visualization
- Interactive Visual Analytics with Folium
- Machine Learning Prediction

Summary of all results

- Exploratory Data Analysis result
- Interactive analytics in screenshots
- Predictive Analytics result

Introduction

- Project background and context
- The purpose of this project was to determine if the first stage Falcon 9 will be successful using data analysis.
- Falcon 9 is one of the SpaceX projects with the cost of 62 million as they advertise on their website.
- Others estimate the cost as of 162 million. It just because SpaceX reuse the first stage over and over.
- Therefore in this project if we determine the first stage will land then we can determine its cost.
- Problems you want to find answers
- Factor determines that the successful landing.
- Features and variable interaction for successful landing rate.
- Operation condition for successful landing.

Section 1

Methodology

Methodology

Executive Summary

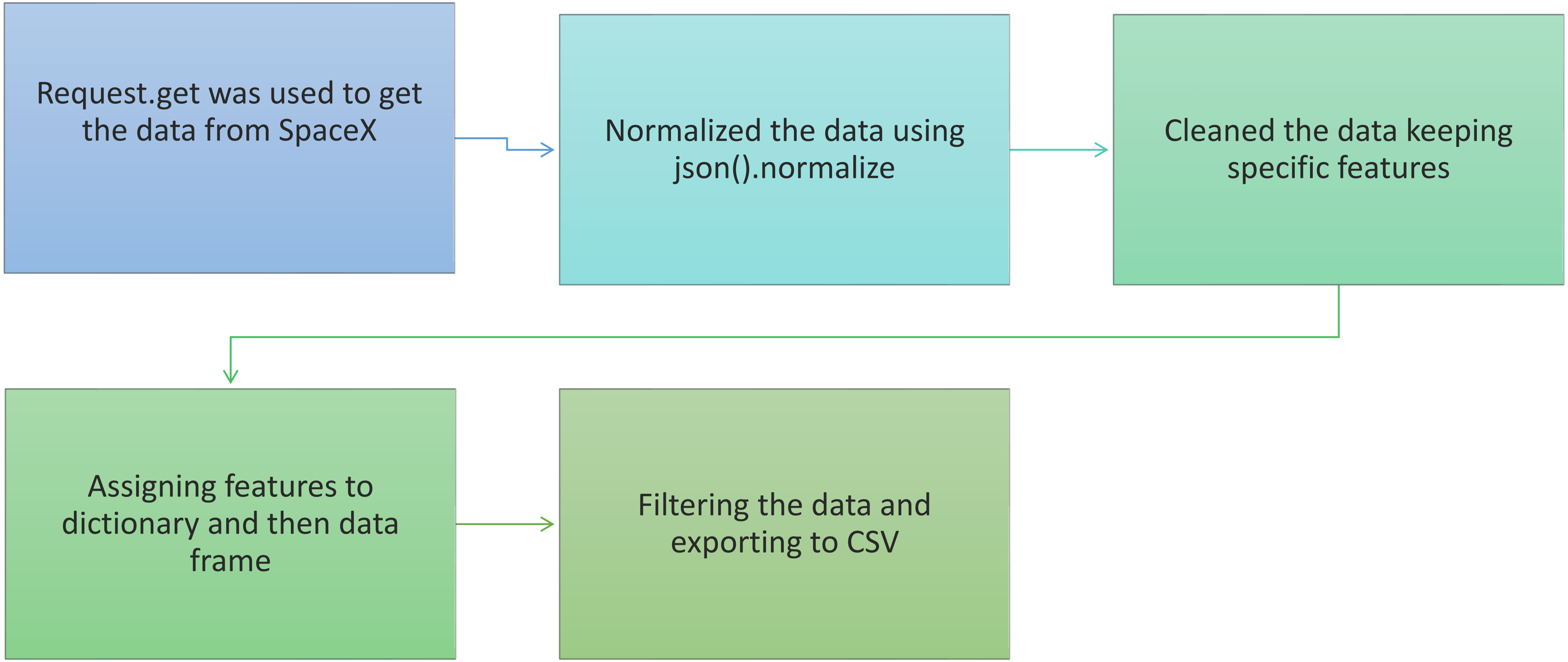
- Data collection methodology:
    - The Required data was collected using API and webscrabing of SpaceX and Wikipedia
- Perform data wrangling
    - To determine the rate of success; one-hot encoding was applied to categorical features.

- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
    - How to build, tune, evaluate classification models

Data Collection

- Describe how data sets were collected.
- Request.get was used to get the data from SpaceX
- Normalized the data using json().normalize
- Cleaned the data keeping specific features
- Assigning features to dictionary and then data frame
- Filtering the data and exporting to CSV
- You need to present your data collection process use key phrases and flowcharts

Data Collection continued



Request.get was used to get the data from SpaceX

Normalized the data using json().normalize

Cleaned the data keeping specific features

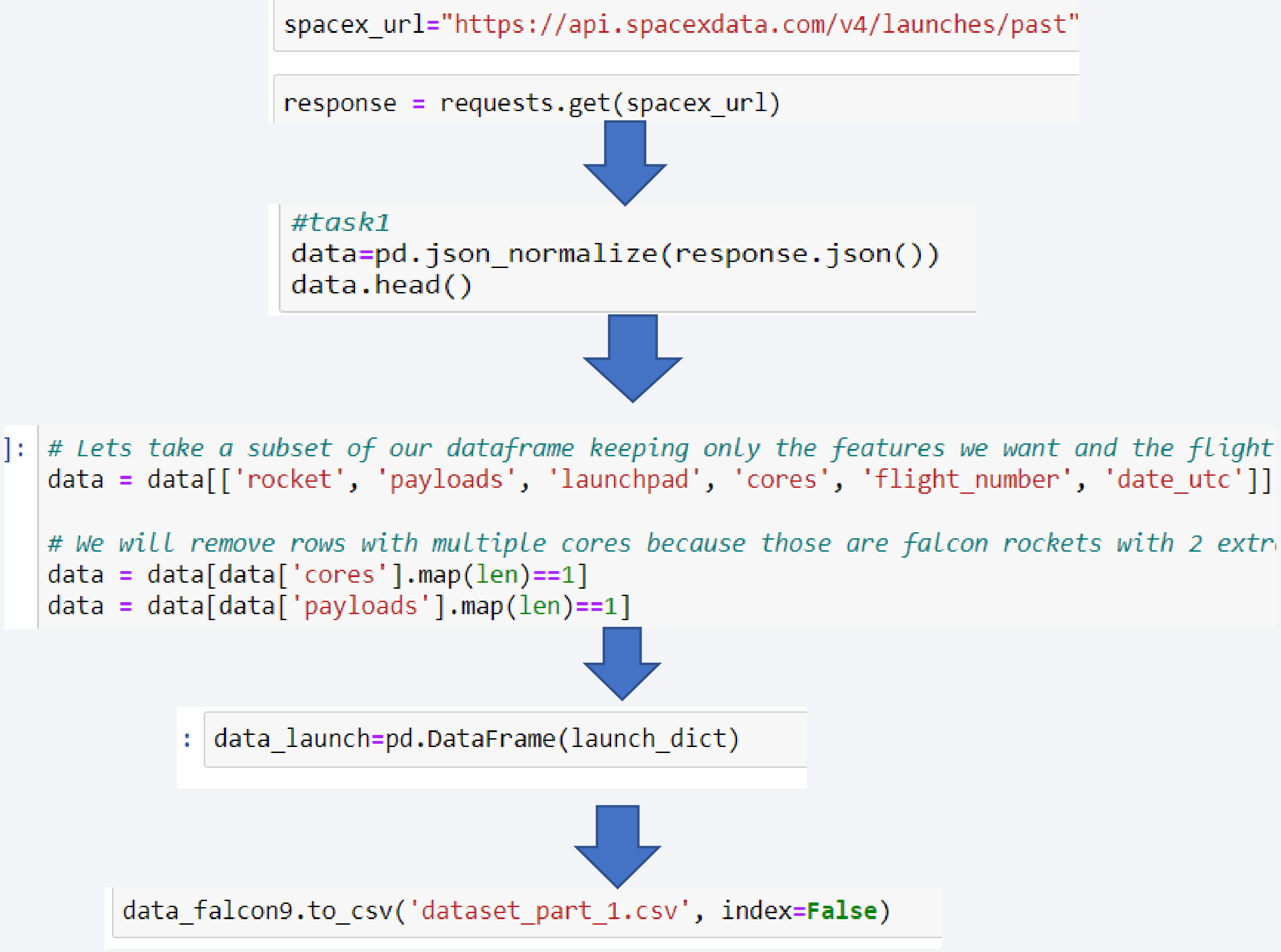
Assigning features to dictionary and then data frame

Filtering the data and exporting to CSV

Data Collection – SpaceX API

- Paths codes represents how data was collected using python method

- The SpaceX API calls notebook GitHub URL is as follow: (https://github.com/azy78/DataCollectionFalcon9-20220522/blob/main/Collecting%20the%20data-Redo-20220522.ipynb), as an external reference.



```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
response = requests.get(spacex_url)
```

```
#task1
data=pd.json_normalize(response.json())
data.head()
```

```
]: # Lets take a subset of our dataframe keeping only the features we want and the flight
   data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

   # We will remove rows with multiple cores because those are falcon rockets with 2 extr
   data = data[data['cores'].map(len)==1]
   data = data[data['payloads'].map(len)==1]
```

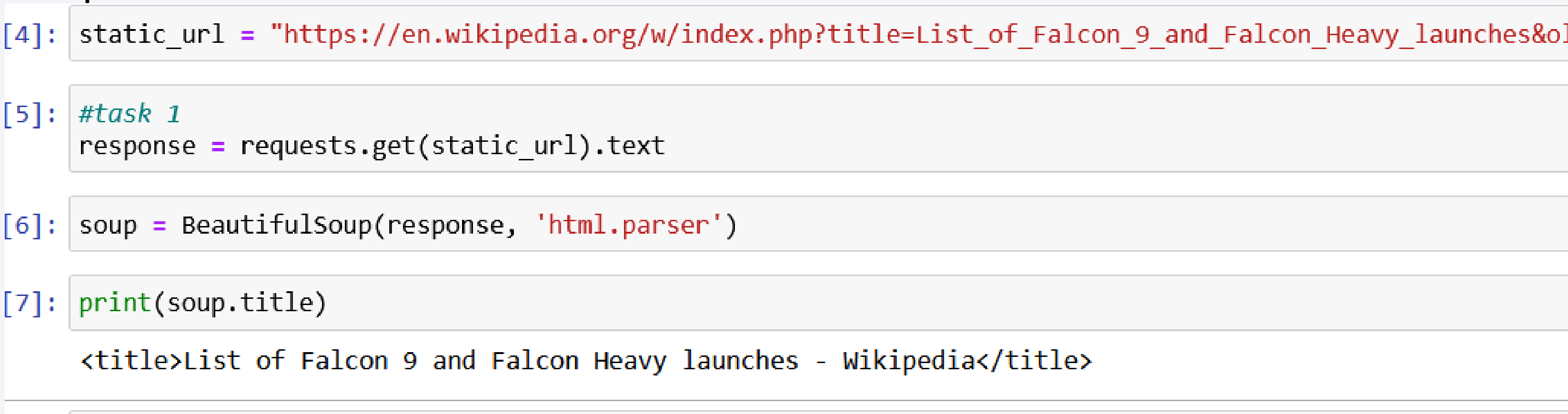
```
: data_launch=pd.DataFrame(launch_dict)
```

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection - Scraping

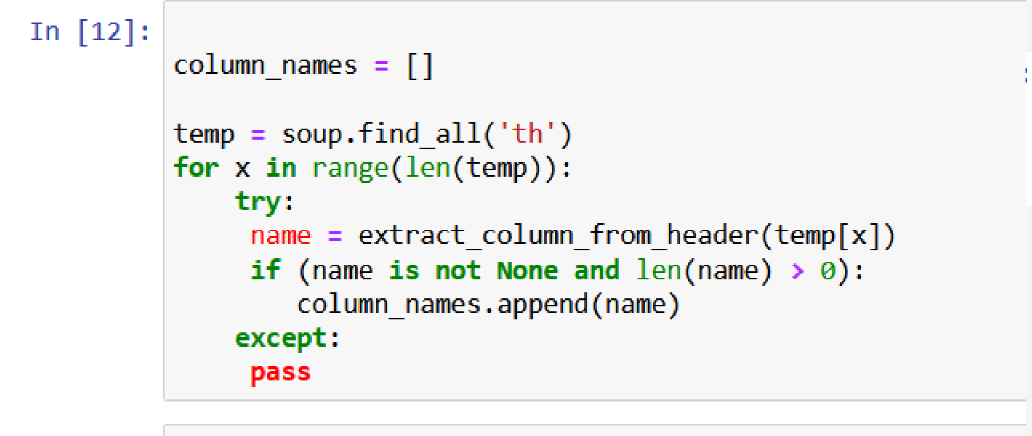
- The get request was used to the SpaceX API to collect data, then requested data was cleaned & formatted & was exported to CSV.
- The SpaceX data collection notebook GitHub URL is as follow: (https://github.com/azy78/Data Scrapping-2/blob/main/DataScrapping-20220522.ipynb), as an external reference.

Step 1:



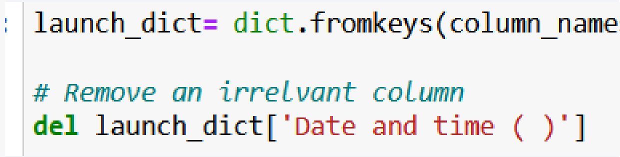
```
[4]: static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&ol

[5]: #task 1
     response = requests.get(static_url).text

[6]: soup = BeautifulSoup(response, 'html.parser')

[7]: print(soup.title)

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

Step2:



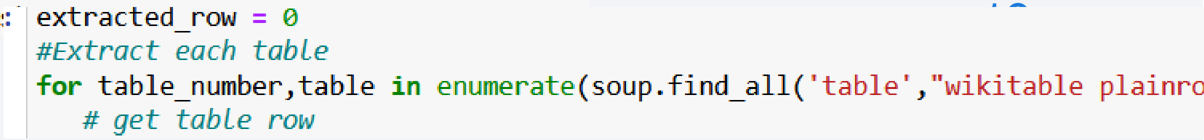
```
In [12]:
         column_names = []

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

In [13]: print(column_name:
```

Step 3:



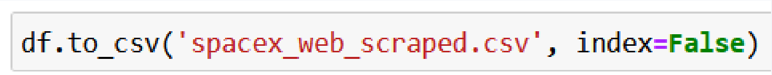
```
launch_dict= dict.fromkeys(column_name

# Remove an irrelvant column
del launch_dict['Date and time ( )']
```

Step 4:



```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainro
   # get table row
```

Step 5:



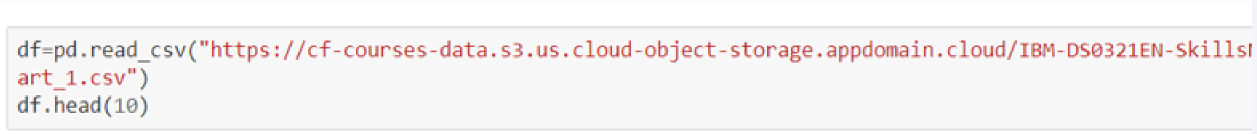
```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Data Wrangling

First data was analyzed. Then number of launches on each site was calculated (each launch with dedicated orbit). Then number of occurrence on each orbit was calculated. Then the number and occurrence of mission outcome per orbit type was calculated. At the end landing outcome labels from Outcome column was calculated and the data was exported to csv.

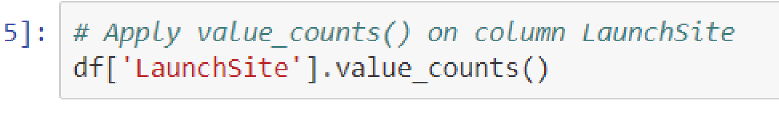
The Data wrangling notebook GitHub URL is as follow: (https://github.com/azy78/DataWrangling20220518/blob/main/DataWrangling-20220518.ipynb) as an external reference.

Step 1



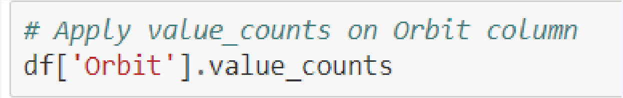
```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsI
art_1.csv")
df.head(10)
```

Step 2



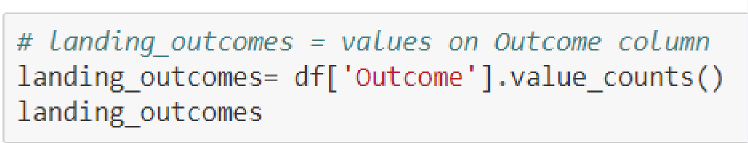
```
5]: # Apply value_counts() on column LaunchSite
    df['LaunchSite'].value_counts()
```

Step 3



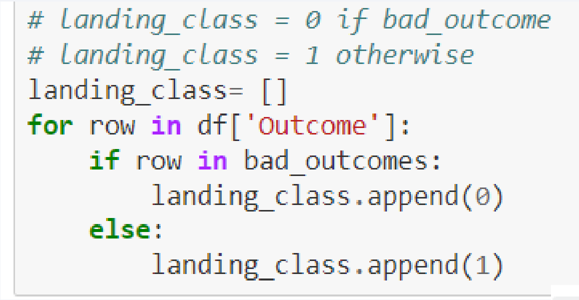
```
# Apply value_counts on Orbit column
df['Orbit'].value_counts
```

Step 4



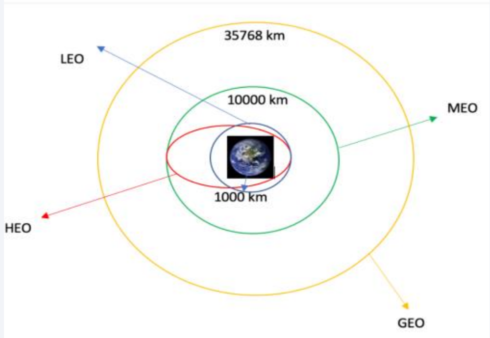
```
# landing_outcomes = values on Outcome column
landing_outcomes= df['Outcome'].value_counts()
landing_outcomes
```

Step 5

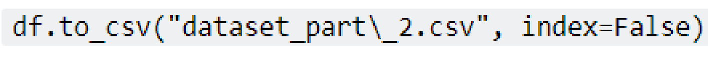


```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class= []
for row in df['Outcome']:
    if row in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

SpaceX, orbits launch sites



Step 6



```
df.to_csv("dataset_part\_2.csv", index=False)
```

EDA with Data Visualization

- The relationship between Flight Number and Launch Site was visualized (figure 1).
- The relationship between success rate of each orbit type was visualized using bar graph (figure 2).
- Visualize the launch success yearly trend (figure 3)

Figure 1

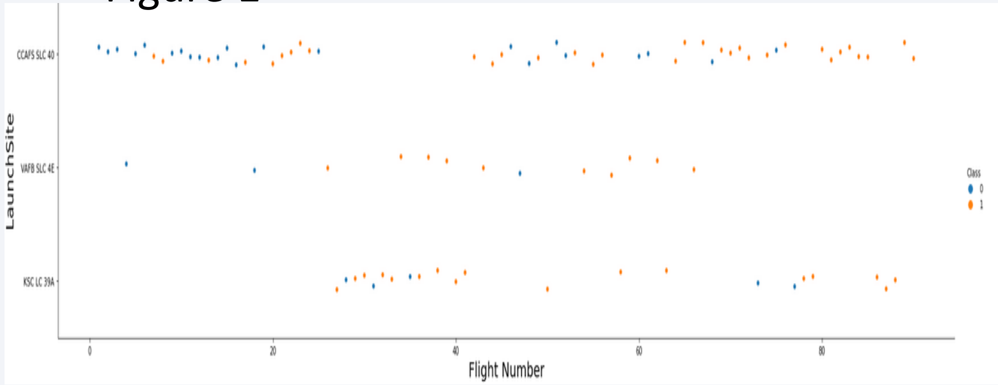
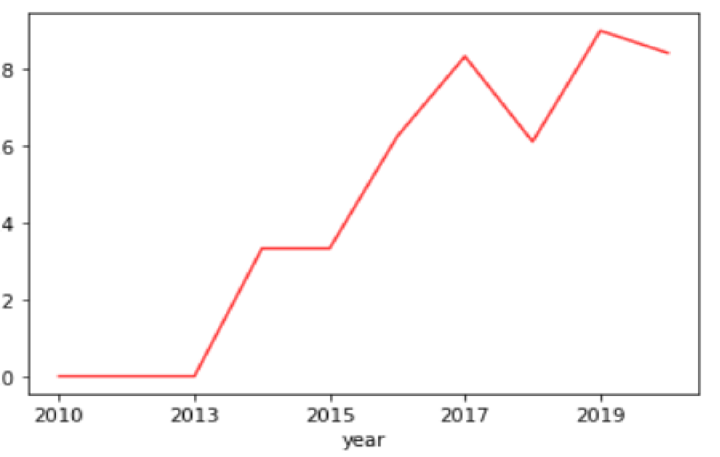


Figure 3

```
: <AxesSubplot:xlabel='year'>
```



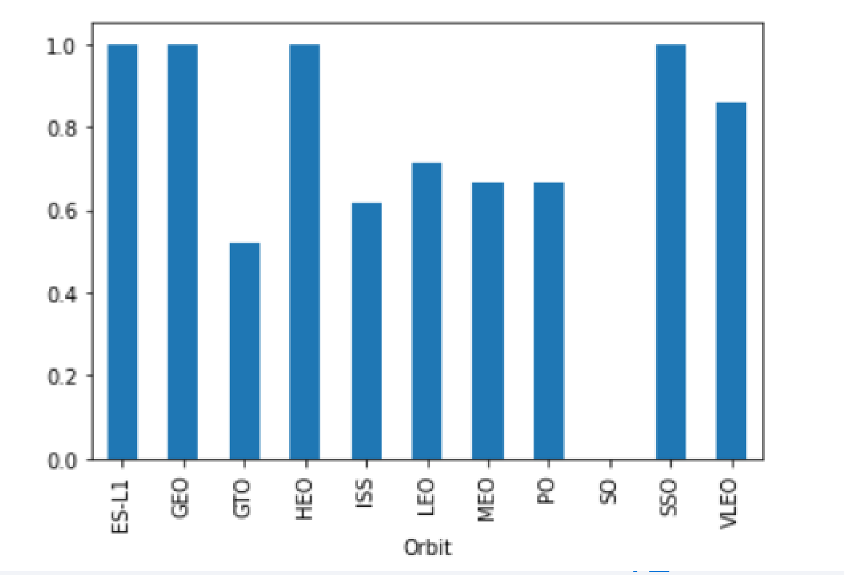
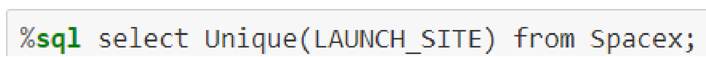


Figure 2

- The EDA notebook GitHub URL is as follow: (https://github.com/azy78/EDA-Data-vitulization-202205022-3/blob/main/EDA%20with%20Data%20Visualization-20220522.ipynb) as an external reference.

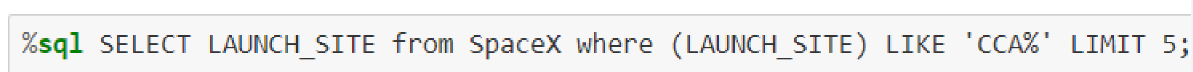
EDA with SQL

- The name of unique launch sites in the space mission:



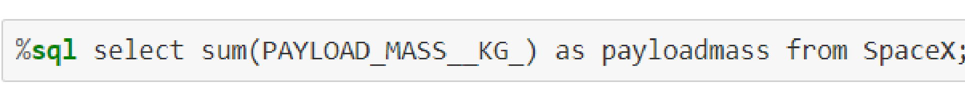
```
%sql select Unique(LAUNCH_SITE) from Spacex;
```

- Records where launch sites found using string ‘CCA:



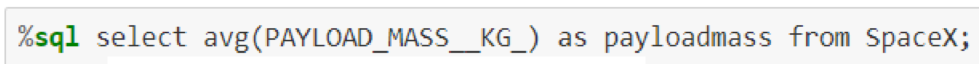
```
%sql SELECT LAUNCH_SITE from SpaceX where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5;
```

- The total payload mass carried by boosters launched by NASA (CRS) was found:



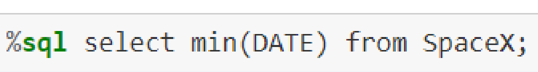
```
%sql select sum(PAYLOAD_MASS__KG_) as payloadmass from SpaceX;
```

- The average payload mass carried by booster was found:



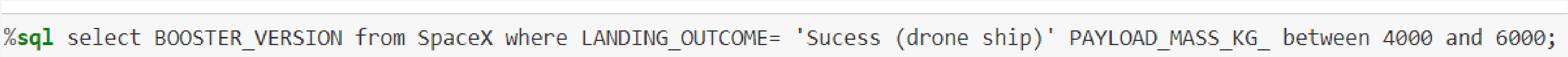
```
%sql select avg(PAYLOAD_MASS__KG_) as payloadmass from SpaceX;
```

- The date of the first successful landing was found using min function:



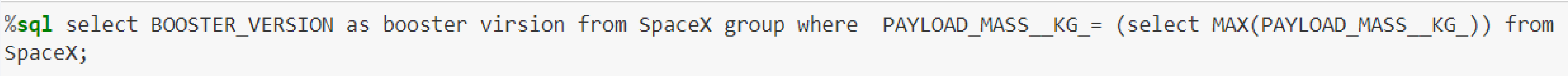
```
%sql select min(DATE) from SpaceX;
```

- Names of the boosters with success in drone ship & with payload mass 4000> & < 6000:



```
%sql select BOOSTER_VERSION from SpaceX where LANDING_OUTCOME= 'Sucess (drone ship)' PAYLOAD_MASS_KG_ between 4000 and 6000;
```

- Total number of success and failure landing was found:



```
%sql select BOOSTER_VERSION as booster virsion from SpaceX group where  PAYLOAD_MASS__KG_= (select MAX(PAYLOAD_MASS__KG_)) from
SpaceX;
```

- The EDA with SQL notebook GitHub URL is as follow: (https://github.com/azy78/EDA-with-SQL-20220522Lab/blob/main/lab2-20220518.ipynb) as an external reference.

Build an Interactive Map with Folium

- All the launching site on map was marked with blue circles, using Folium. Circle and Folium.Marker.
- The success/failed launches for each site on the map was marked using Marker Cluster object.
- The distances between a launch site to its proximities was calculated by adding Mouse Position & using a Marker object & by drawing a polyline.

- The Build an Interactive Map with Folium notebook GitHub URL is as follow: (https://github.com/azy78/Build-an-Interactive-Map-with-Folium/blob/main/Launch%20Sites%20Locations%20Analysis%20with%20Folium-Lab20220521.ipynb) as an external reference.

Build a Dashboard with Plotly Dash

- An interactive dashboard with Plotly dash was made.
- Pie charts showing the total launches by a certain sites was built.
- A scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version was plotted.
- The Dash with Plotly notebook GitHub URL is as follow: (https://github.com/azy78/Spacex-Dash/blob/main/spacex_dash_app.py)

Predictive Analysis (Classification)

- Data was loaded into data sets and was virtualized and was early analyzed.

  The data set was split into training set and test set & picked ML algorithm (Building the model).

- Models accuracy was determined and confusion matrix was made (Model evaluation).
- The models was improved by tuning the algorithm and variables (Model improvement).
- The GitHub URL of the completed predictive analysis lab is as follow:

(https://github.com/azy78/MachinLearning-20220519/blob/main/Machine%20Learning%20PredictionRedo-lab20220521(1).ipynb)

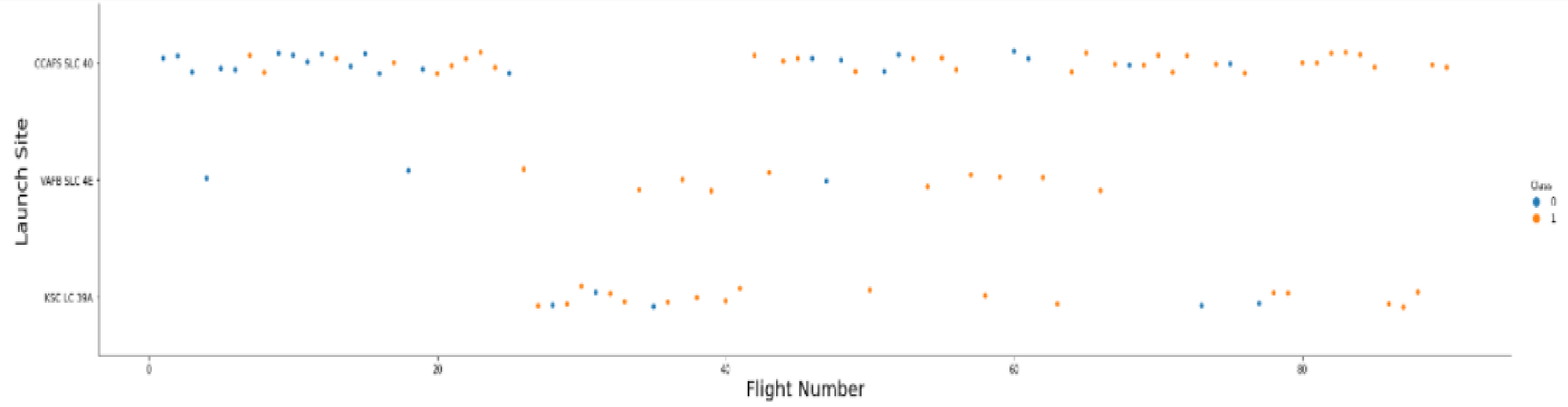
Results

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

Section 2

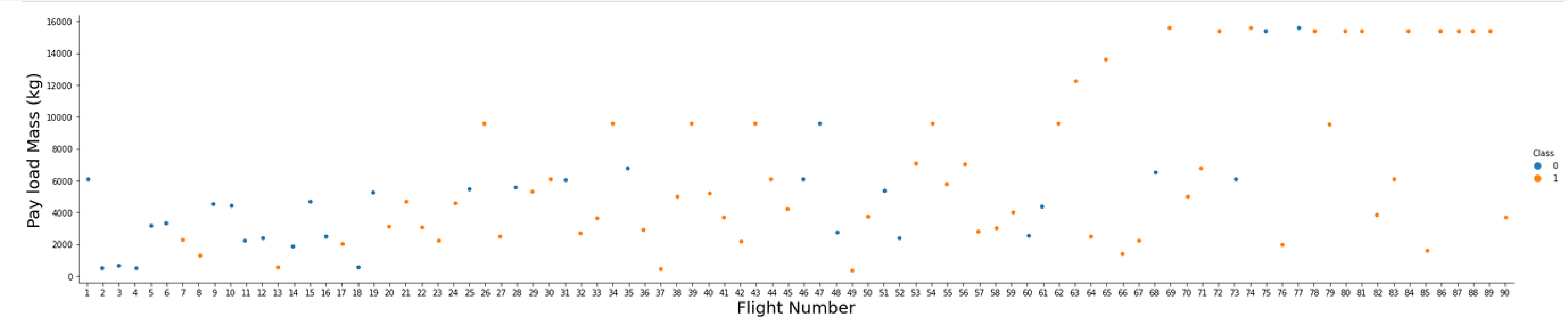
Insights drawn from EDA

Flight Number vs. Launch Site



The scattered plot shows the launch site vs flight number of Falcon 9 SpaceX , the class 0 (blue dots) shows the fail launch and class 1(orange dots) shows successful launches. The site CCAFS SLC 40 has the largest amount of launching.

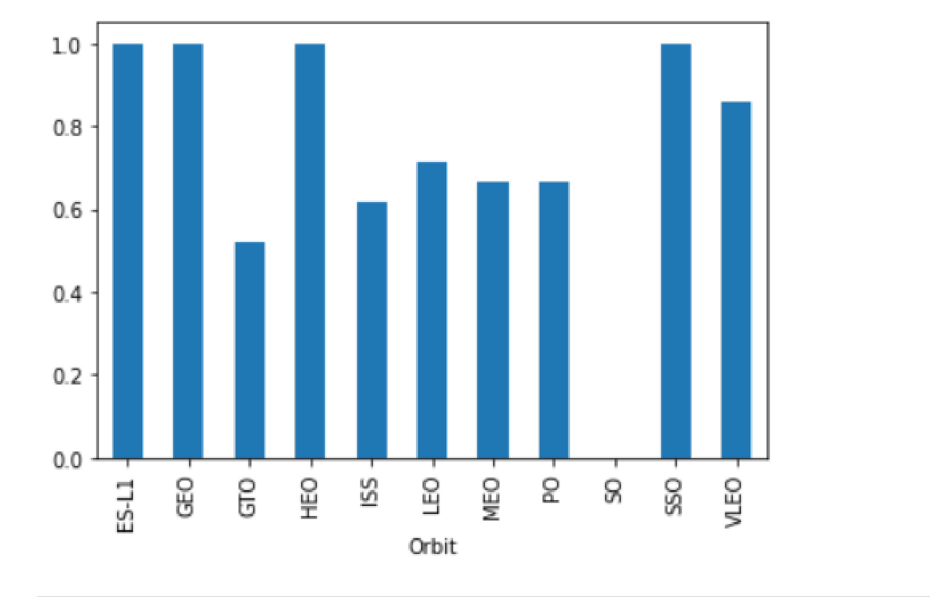
Payload vs. Launch Site



The scattered plot shows the relation between the Pay load Mass (Kg) to flight number. It shows the higher the mass the higher the rate of failing.

Success Rate vs. Orbit Type

- The bar chart shows the success rate of each orbit type.
- ES-L1, GEO, HEO and SSO have the most successful rate out of all orbits.

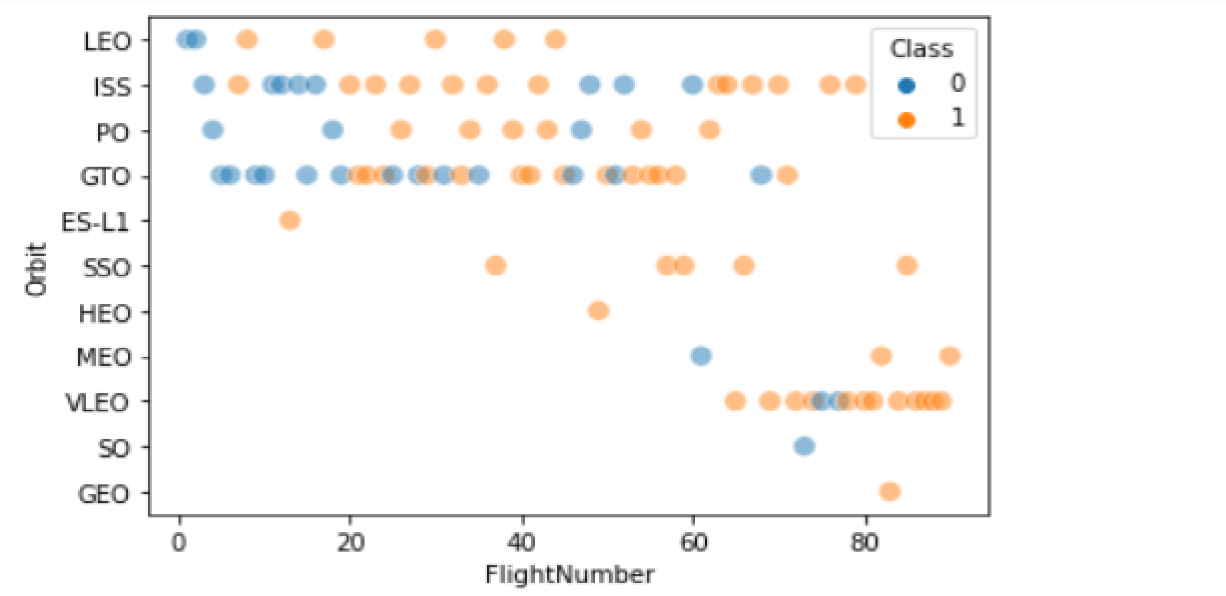


Flight Number vs. Orbit Type

- The plot below shows the Flight Number vs. Orbit type.

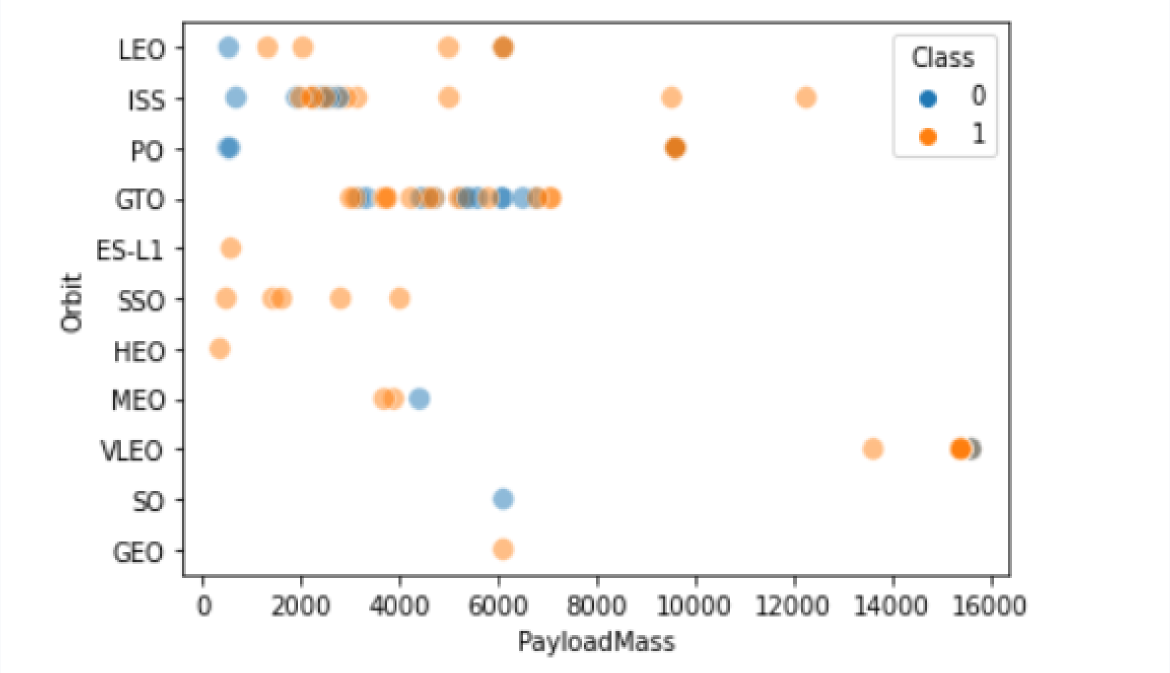
- The graph shows that more flights was done

In orbits GTO, PO, ISS, LEO, either successful or failed.



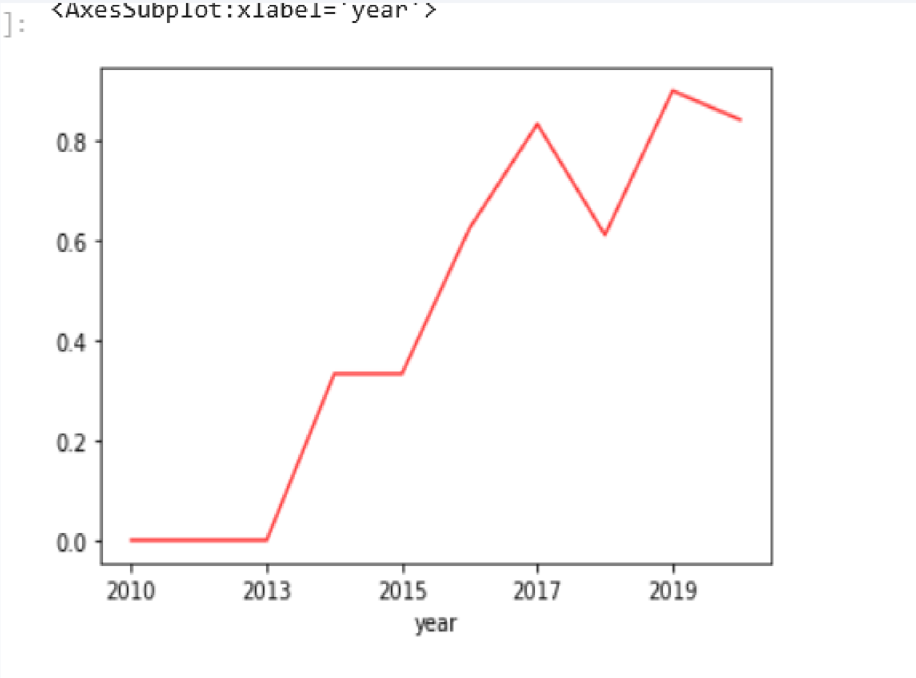
Payload vs. Orbit Type

- The plot shows the correlation between the orbit and payloadmass.
- The lower the mass and the smaller the orbit has more flights around them.
- GTO and ISS orbits have more consistent trails.



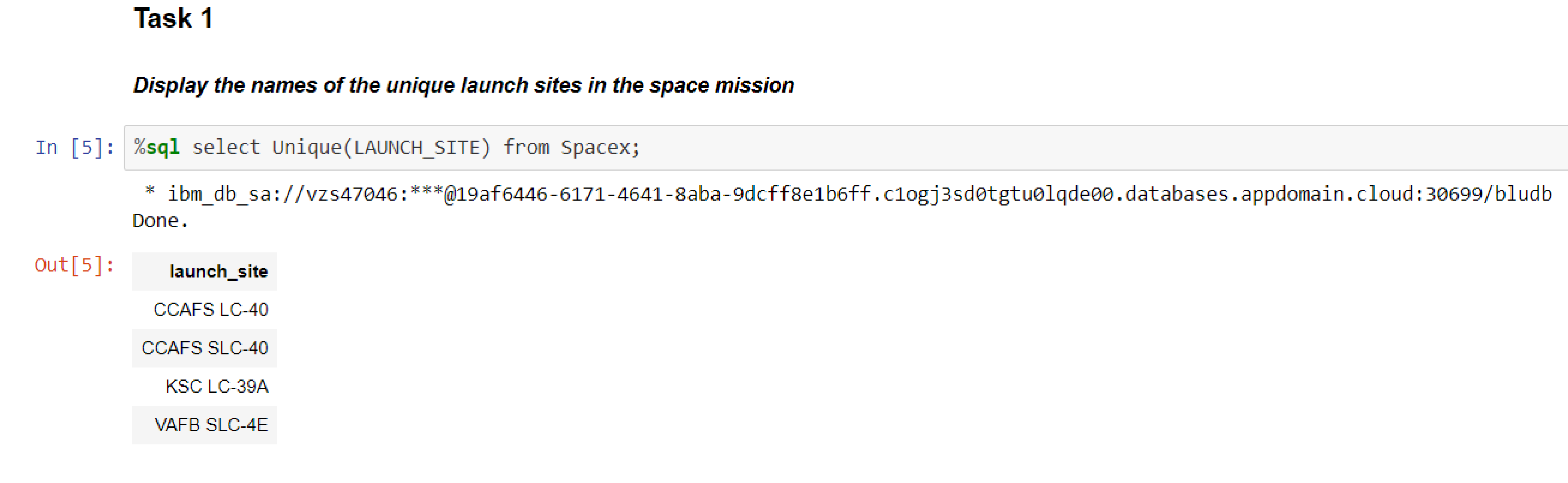
Launch Success Yearly Trend

- The plot shows the yearly average success rate of Falcon 9.
- The graph shows over years with improvement & tunning the conditions, it has more success rates.



```
]:  <AxesSubplot:xlabel='year'>
```

All Launch Site Names



Task 1

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

```
In [5]: %sql select Unique(LAUNCH_SITE) from Spacex;
```

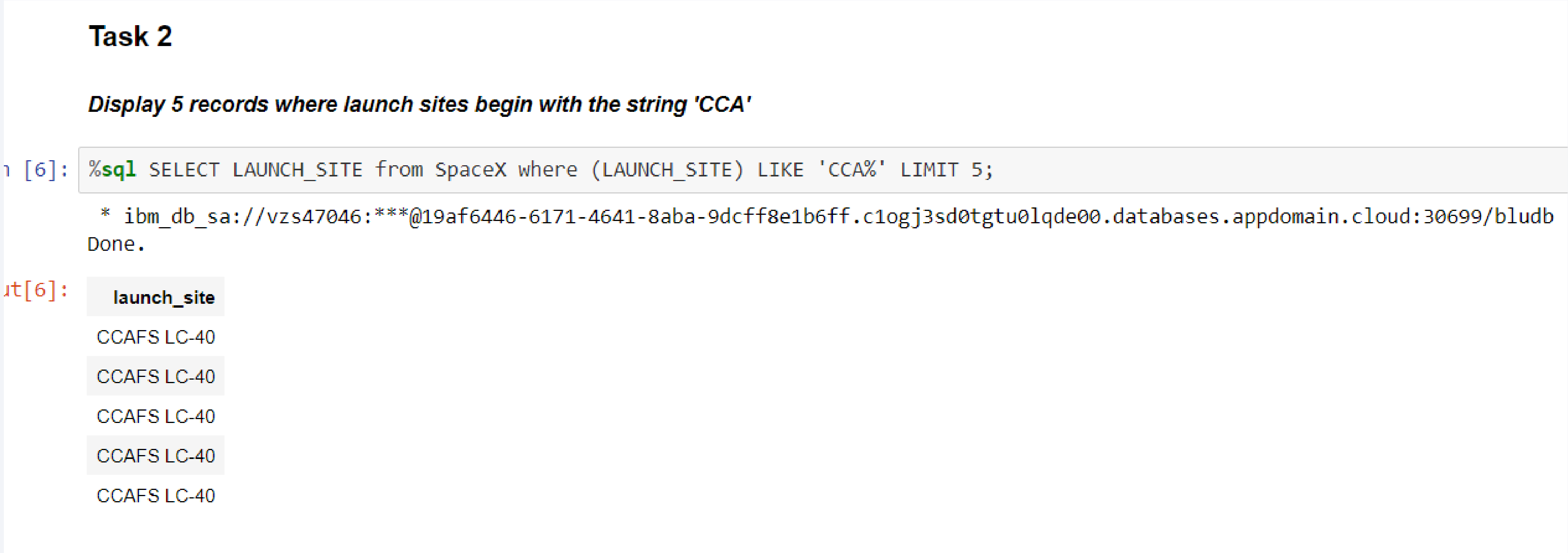
```
 * ibm_db_sa://vzs47046:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30699/bludb
Done.
```

Out[5]:

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

Four names of unique launch sites in the spaceX mission was found using % sql select method from EDA SQL lab.

Launch Site Names Begin with 'CCA'



Task 2

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

```
n [6]: %sql SELECT LAUNCH_SITE from SpaceX where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5;
```

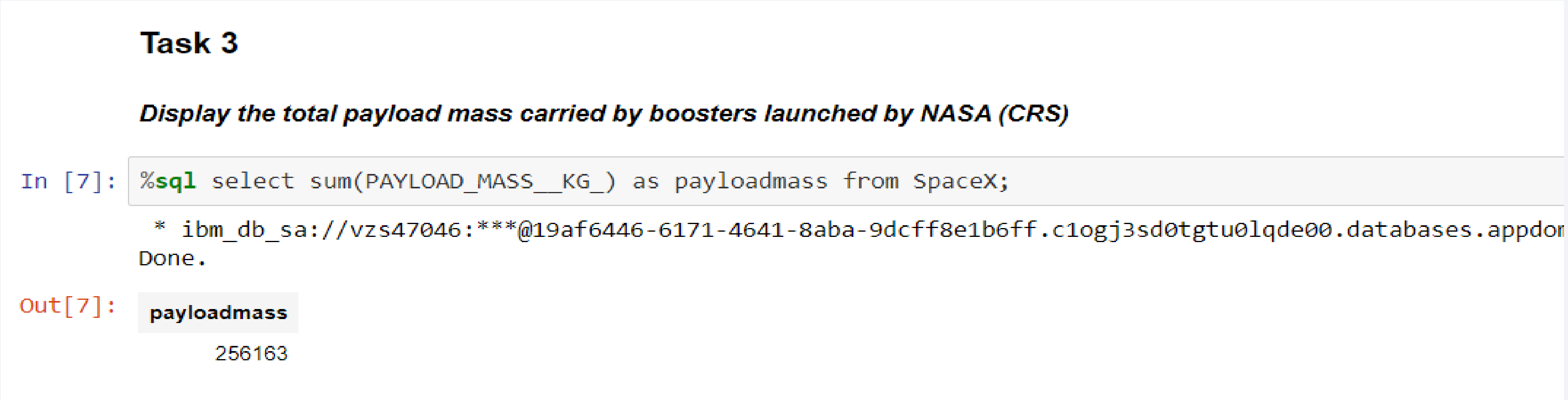
```
 * ibm_db_sa://vzs47046:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30699/bludb
Done.
```

ut[6]:

launch_site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

- The screen shot above shows 5 records where launch sites begin with the string CCA. This was found using % sql select method.

Total Payload Mass



Task 3

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

```
In [7]: %sql select sum(PAYLOAD_MASS__KG_) as payloadmass from SpaceX;
```

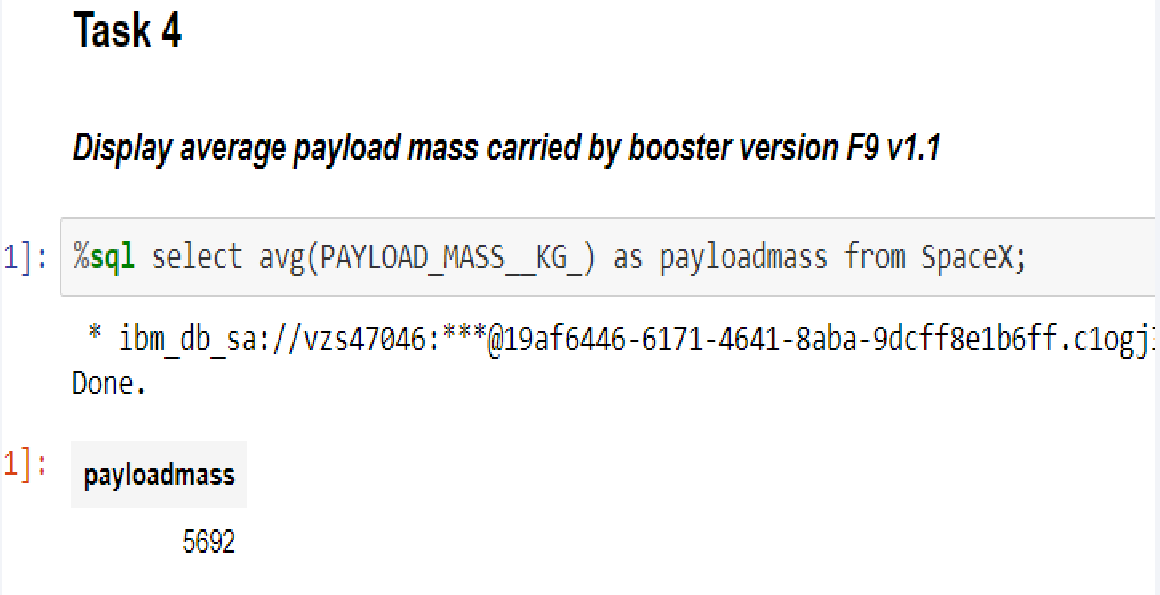
```
 * ibm_db_sa://vzs47046:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdom
Done.
```

Out[7]:

payloadmass
256163

- The screen shot above shows the total payload carried by boosters from NASA. The total payload mass was 256,163 kg. This was found using %sql select method.

Average Payload Mass by F9 v1.1



Task 4

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

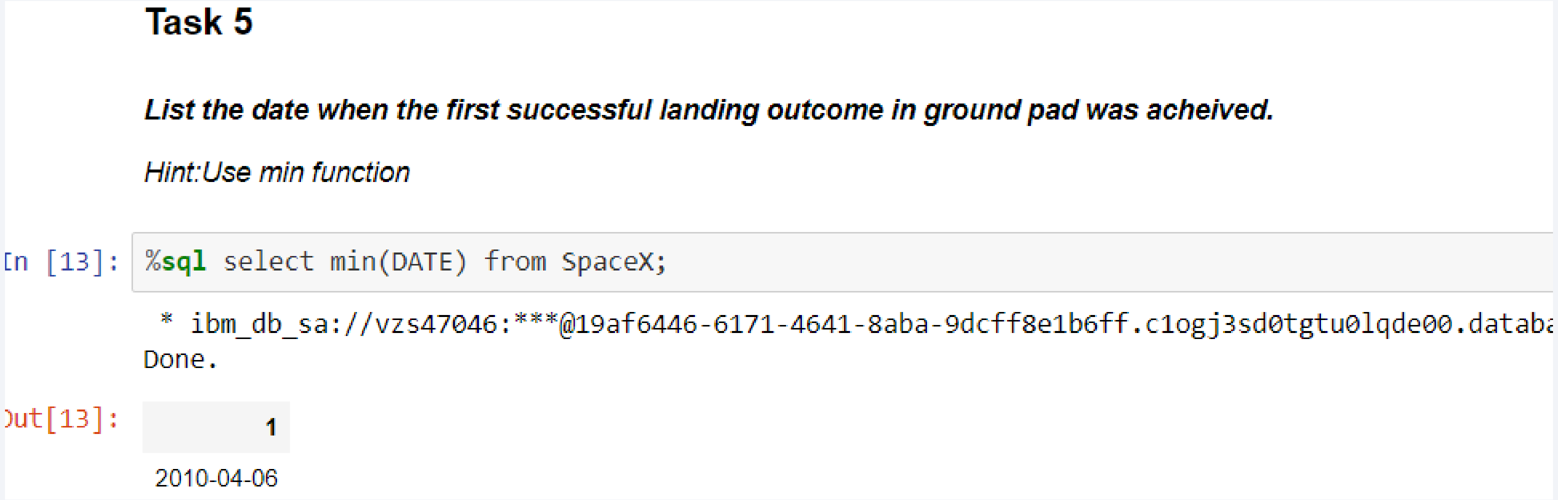
```
1]: %sql select avg(PAYLOAD_MASS__KG_) as payloadmass from SpaceX;

 * ibm_db_sa://vzs47046:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj
Done.

1]:
```

payloadmass
5692

- The screen shot shows the calculation of the average payload mass carried by booster version F9 v1.1. The average mass was 5692 kg. This was found using % sql select method.

First Successful Ground Landing Date

- The screen shots shows the date of the first successful landing outcome on ground pad. This was found using min function and %sql select method.



Task 5

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

Hint:Use min function

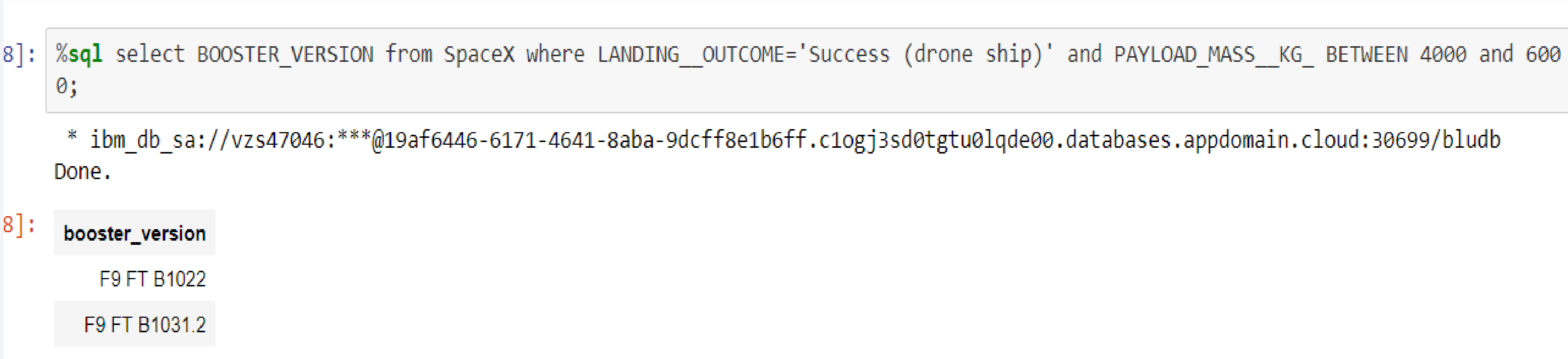
```
In [13]: %sql select min(DATE) from SpaceX;
```

```
 * ibm_db_sa://vzs47046:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databa
Done.
```

Out[13]:

1
2010-04-06

Successful Drone Ship Landing with Payload between 4000 and 6000

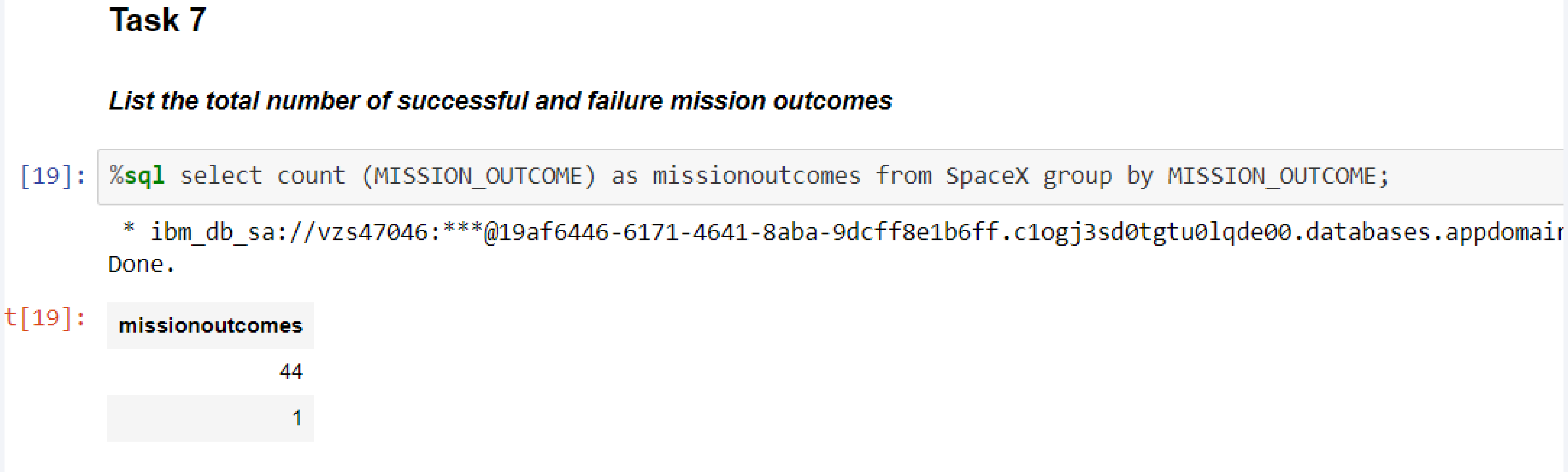


```
8]: %sql select BOOSTER_VERSION from SpaceX where LANDING__OUTCOME='Success (drone ship)' and PAYLOAD_MASS__KG_ BETWEEN 4000 and 600
    0;

     * ibm_db_sa://vzs47046:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30699/bludb
    Done.

8]:
```

booster_version
F9 FT B1022
F9 FT B1031.2

- The screen shots shows the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000. Thus was found using % sql select booster_version method.

Total Number of Successful and Failure Mission Outcomes



Task 7

List the total number of successful and failure mission outcomes

```
[19]: %sql select count (MISSION_OUTCOME) as missionoutcomes from SpaceX group by MISSION_OUTCOME;

 * ibm_db_sa://vzs47046:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomair
Done.
```

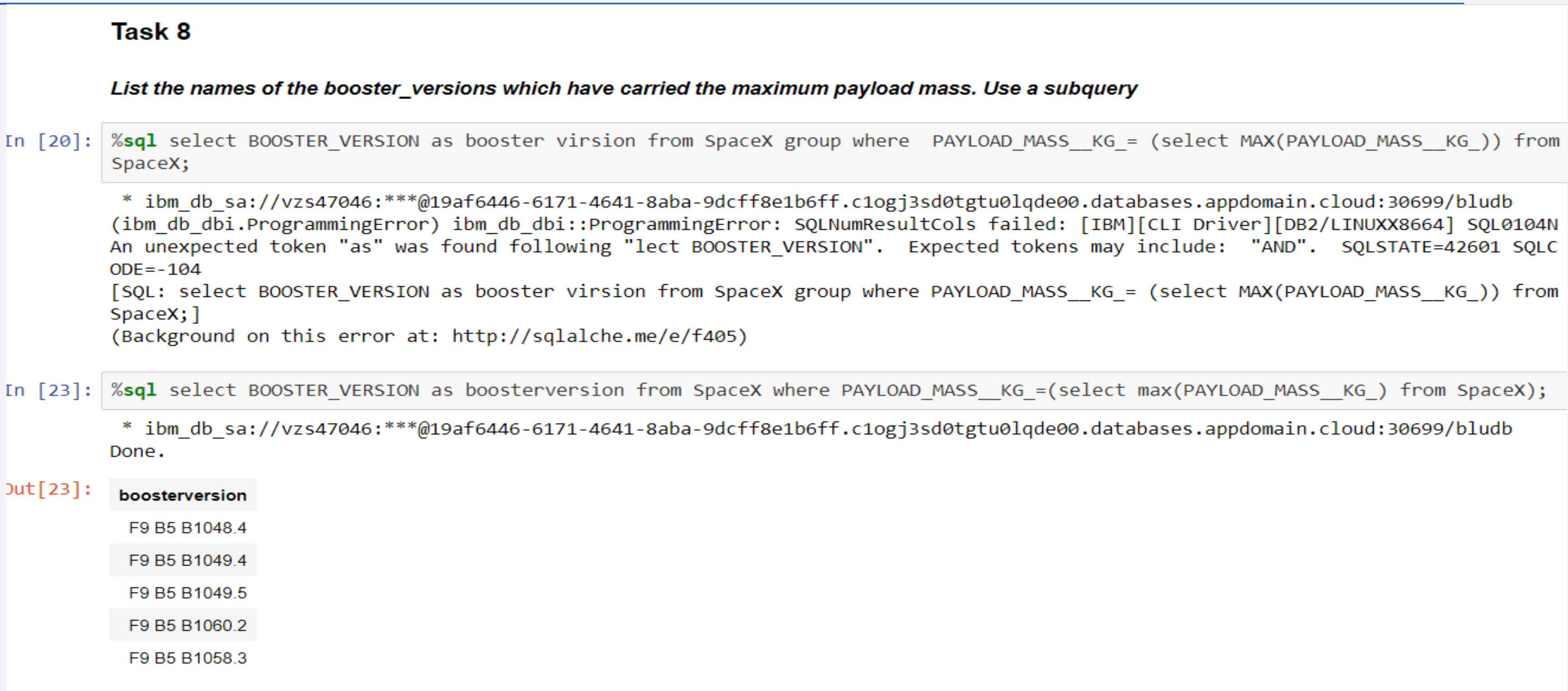
t[19]:

missionoutcomes
44
1

- The screen shot shows the total number of successful and failure mission outcomes.
- The outcome shows 44 attempt successful and 1 fail trial.

Boosters Carried Maximum Payload

- The screen shot shows names of the booster which have carried the maximum payload mass
- The results was found using %sql select booster_version.



Task 8

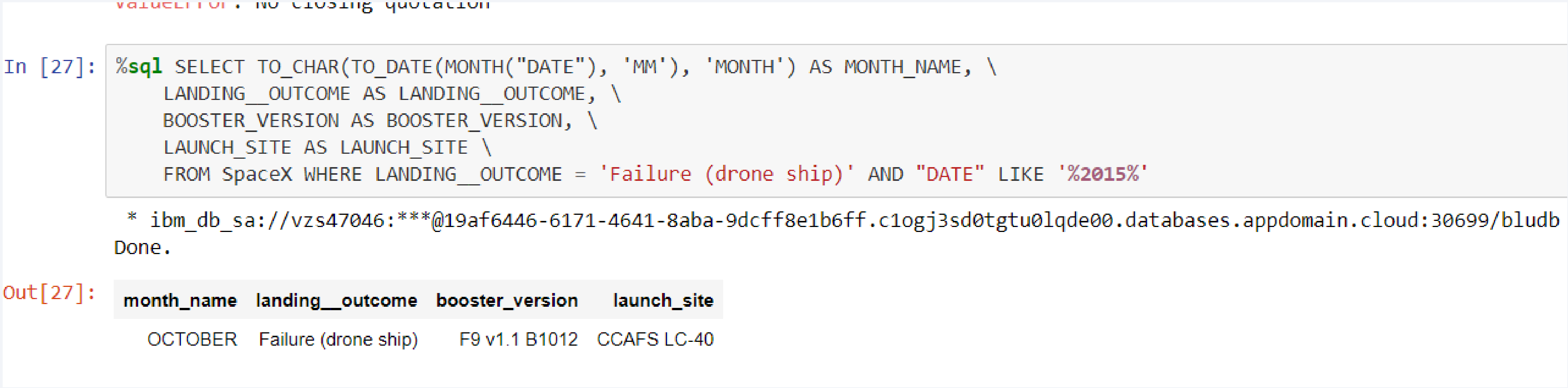
List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
In [20]: %sql select BOOSTER_VERSION as booster virsion from SpaceX group where  PAYLOAD_MASS__KG_= (select MAX(PAYLOAD_MASS__KG_)) from
         SpaceX;

          * ibm_db_sa://vzs47046:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30699/bludb
         (ibm_db_dbi.ProgrammingError) ibm_db_dbi::ProgrammingError: SQLNumResultCols failed: [IBM][CLI Driver][DB2/LINUXX8664] SQL0104N
         An unexpected token "as" was found following "lect BOOSTER_VERSION".  Expected tokens may include:  "AND".  SQLSTATE=42601 SQLC
         ODE=-104
         [SQL: select BOOSTER_VERSION as booster virsion from SpaceX group where PAYLOAD_MASS__KG_= (select MAX(PAYLOAD_MASS__KG_)) from
         SpaceX;]
         (Background on this error at: http://sqlalche.me/e/f405)

In [23]: %sql select BOOSTER_VERSION as boosterversion from SpaceX where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SpaceX);

          * ibm_db_sa://vzs47046:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30699/bludb
         Done.
```

Out[23]:

boosterversion
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3

2015 Launch Records



```
ValueError: No closing quotation
```

```
In [27]: %sql SELECT TO_CHAR(TO_DATE(MONTH("DATE"), 'MM'), 'MONTH') AS MONTH_NAME, \
             LANDING__OUTCOME AS LANDING__OUTCOME, \
             BOOSTER_VERSION AS BOOSTER_VERSION, \
             LAUNCH_SITE AS LAUNCH_SITE \
             FROM SpaceX WHERE LANDING__OUTCOME = 'Failure (drone ship)' AND "DATE" LIKE '%2015%'
```

```
 * ibm_db_sa://vzs47046:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30699/bludb
Done.
```

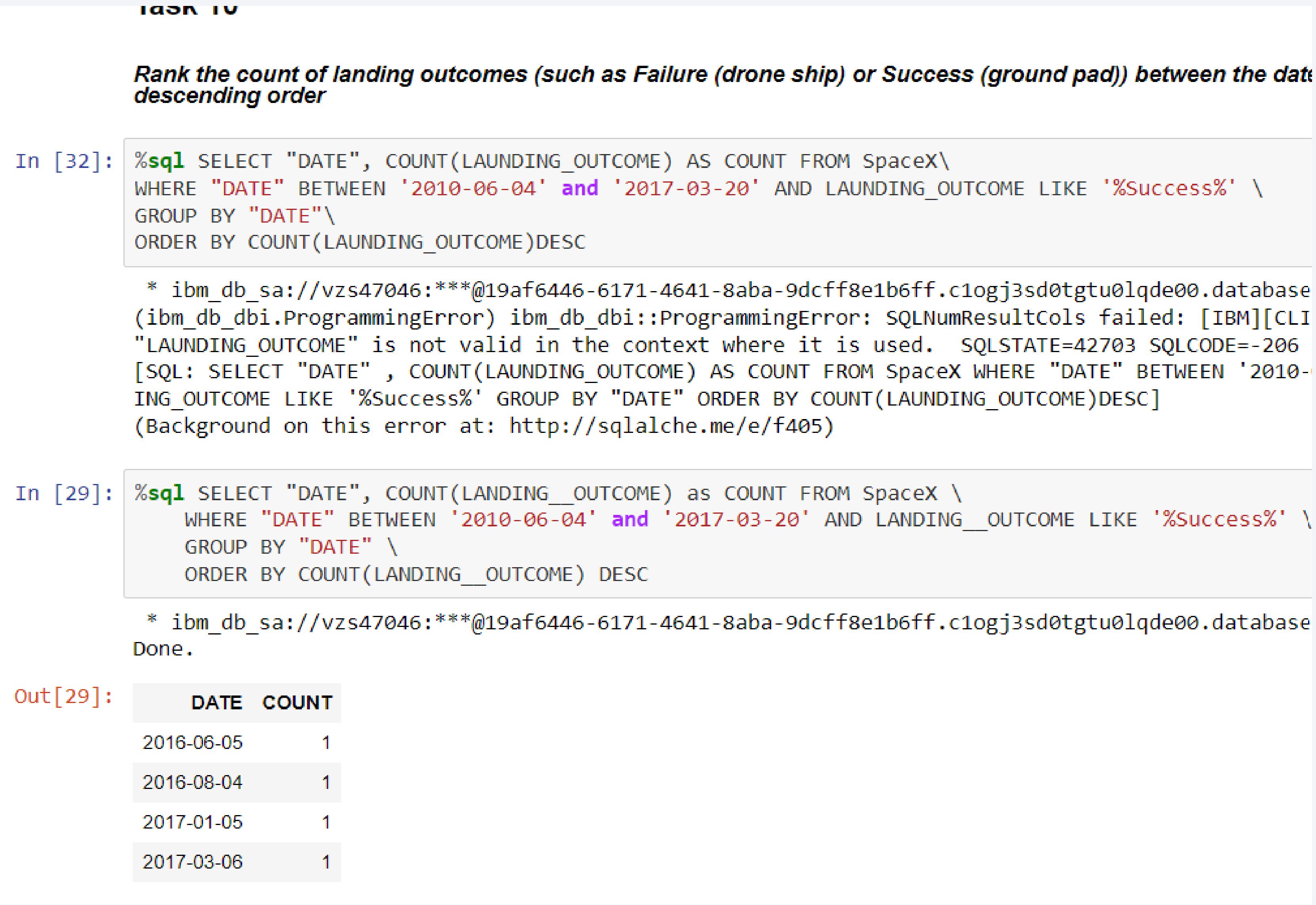
Out[27]:

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

- The screen shot shows failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015.
- The result was found using % sql select method.

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

- The screen shot shows 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.
- It was found using %sql select method.



Task 10

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the dat descending order

In [32]:
```
%sql SELECT "DATE", COUNT(LAUNDING_OUTCOME) AS COUNT FROM SpaceX\
WHERE "DATE" BETWEEN '2010-06-04' and '2017-03-20' AND LAUNDING_OUTCOME LIKE '%Success%' \
GROUP BY "DATE"\
ORDER BY COUNT(LAUNDING_OUTCOME)DESC
```

```
 * ibm_db_sa://vzs47046:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.database
(ibm_db_dbi.ProgrammingError) ibm_db_dbi::ProgrammingError: SQLNumResultCols failed: [IBM][CLI
"LAUNDING_OUTCOME" is not valid in the context where it is used.  SQLSTATE=42703 SQLCODE=-206
[SQL: SELECT "DATE" , COUNT(LAUNDING_OUTCOME) AS COUNT FROM SpaceX WHERE "DATE" BETWEEN '2010-
ING_OUTCOME LIKE '%Success%' GROUP BY "DATE" ORDER BY COUNT(LAUNDING_OUTCOME)DESC]
(Background on this error at: http://sqlalche.me/e/f405)
```

In [29]:
```
%sql SELECT "DATE", COUNT(LANDING__OUTCOME) as COUNT FROM SpaceX \
    WHERE "DATE" BETWEEN '2010-06-04' and '2017-03-20' AND LANDING__OUTCOME LIKE '%Success%' \
    GROUP BY "DATE" \
    ORDER BY COUNT(LANDING__OUTCOME) DESC
```

```
 * ibm_db_sa://vzs47046:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.database
Done.
```

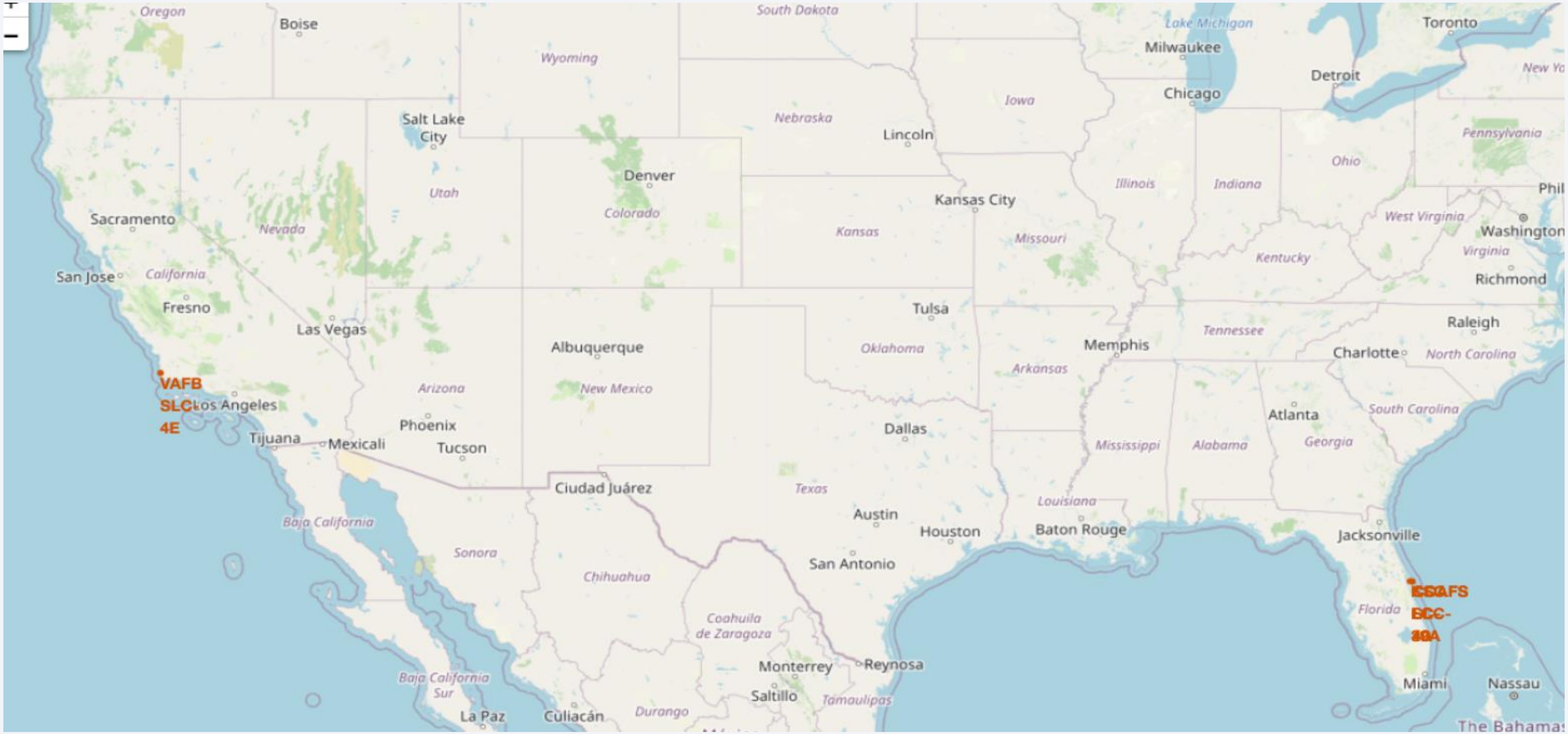
Out[29]:

DATE	COUNT
2016-06-05	1
2016-08-04	1
2017-01-05	1
2017-03-06	1

Section 3

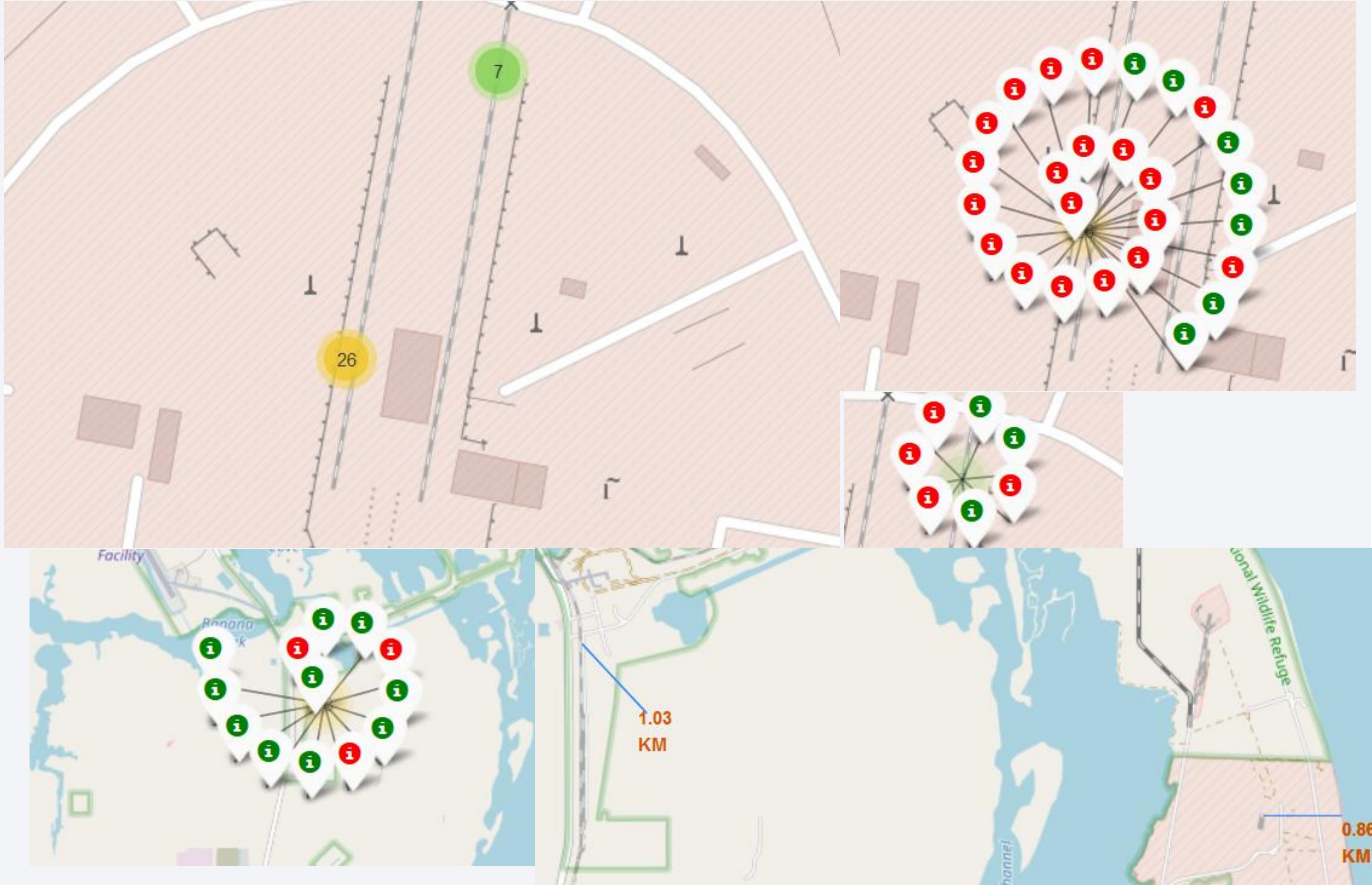
Launch Sites Proximities Analysis

All launch sites global map markers



Markers showing launch sites with color labels

- Using mouse position two sites were explored.
- The green color shows the successful landing and the red color shows the fail landing.

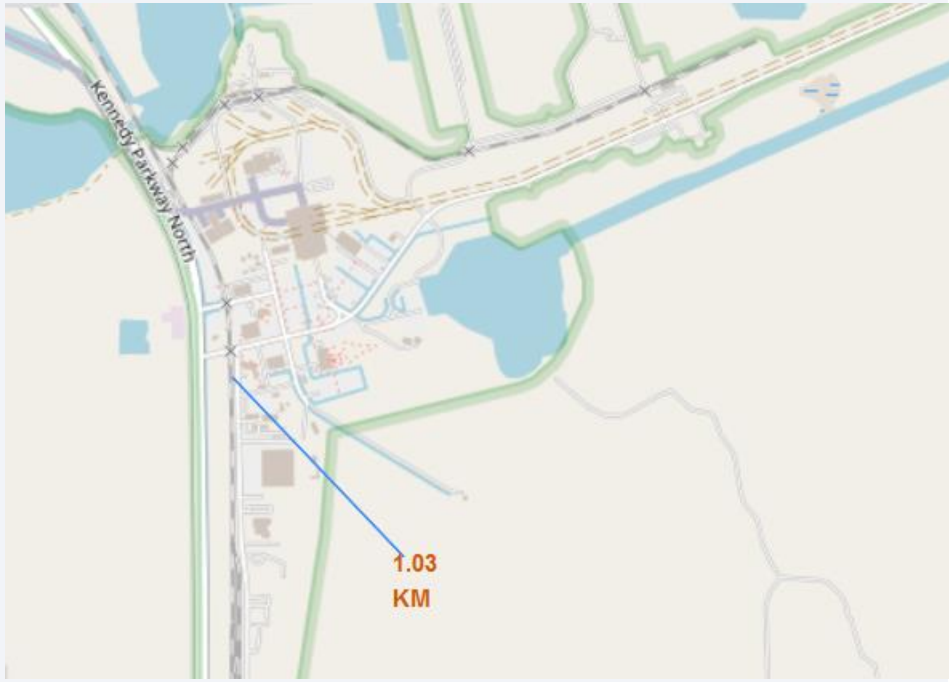


Launch Site distance to landmarks

- The screen shot shows

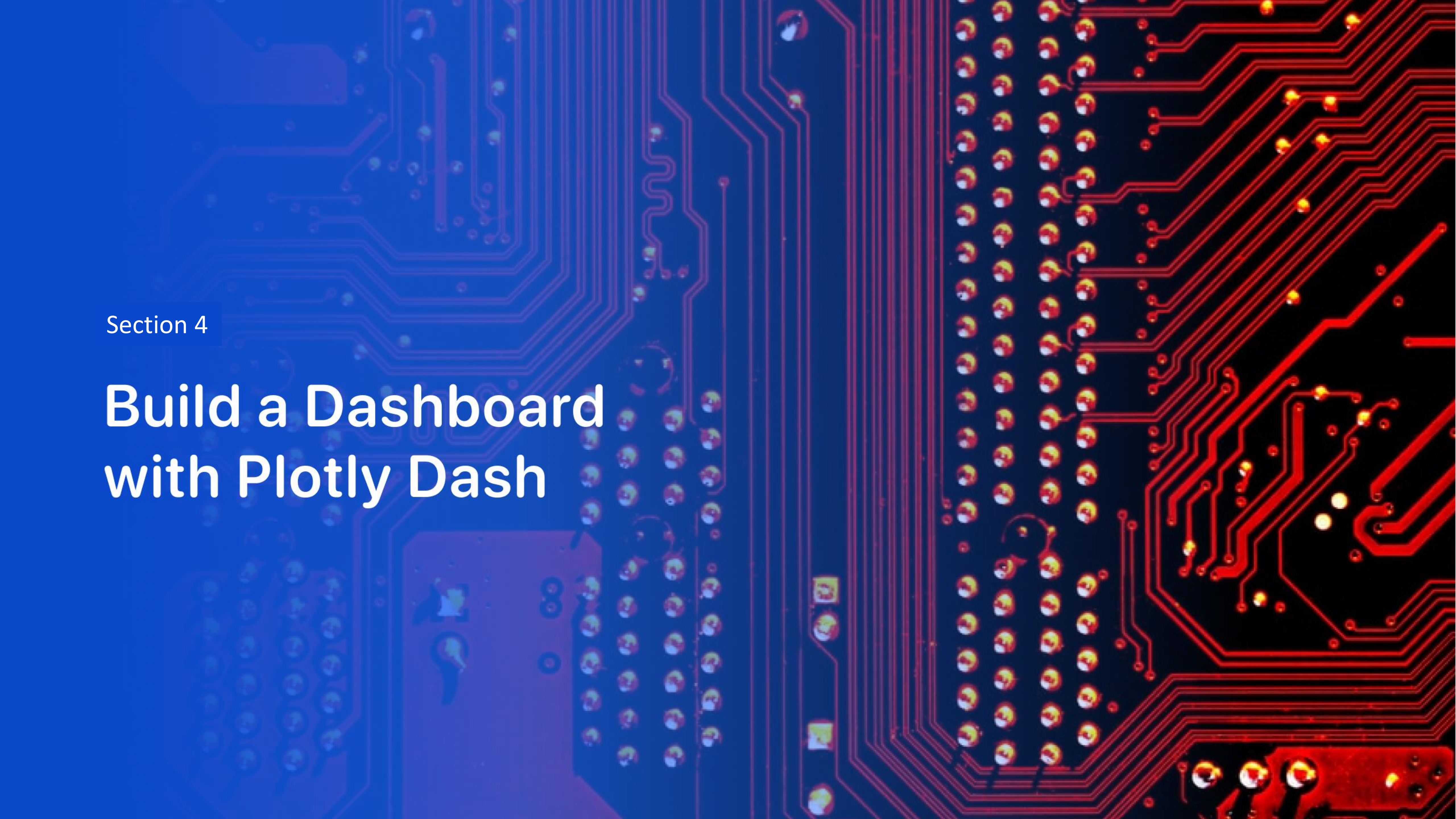
The distance of the roads and the railways to the site.

Centaur Road
oad

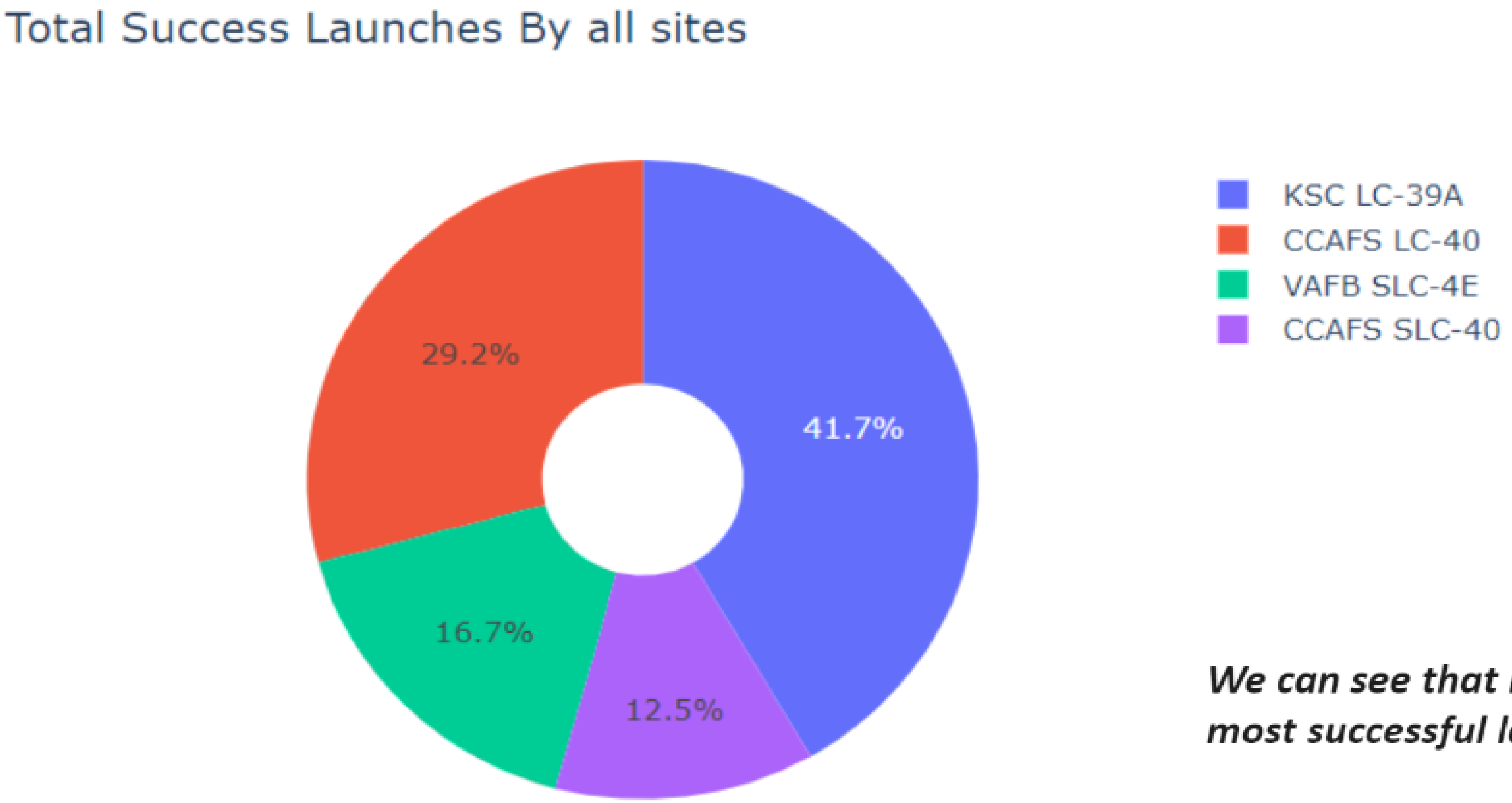


Section 4

Build a Dashboard with Plotly Dash

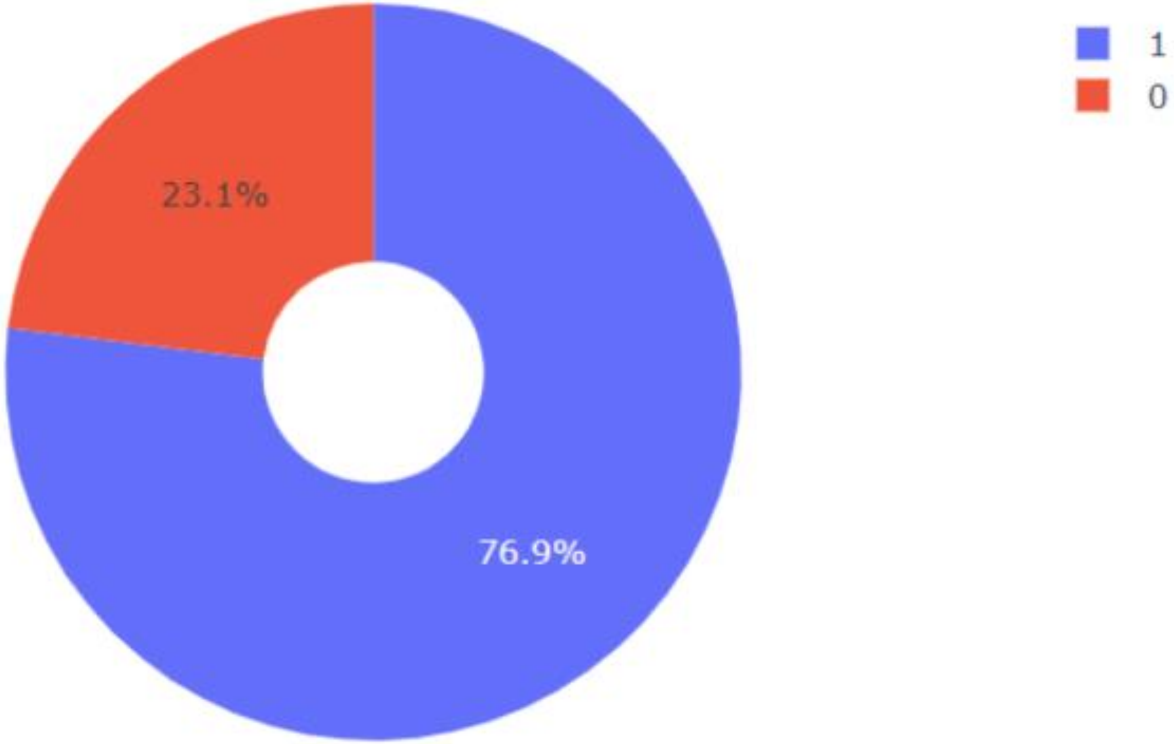


Pie chart showing the success percentage achieved by each launch site



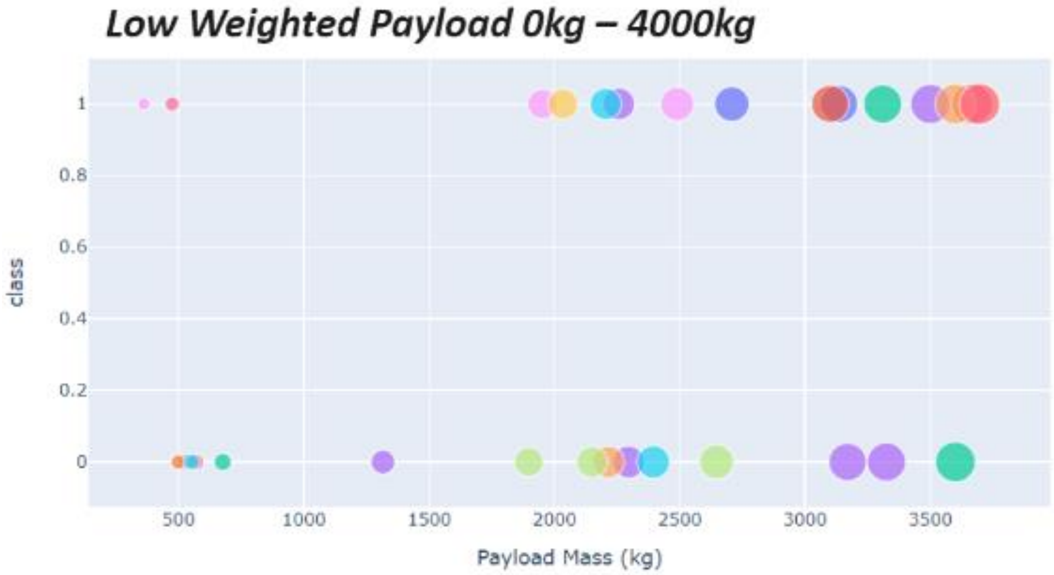
We can see that KSC LC-39A had the most successful launches from all the sites

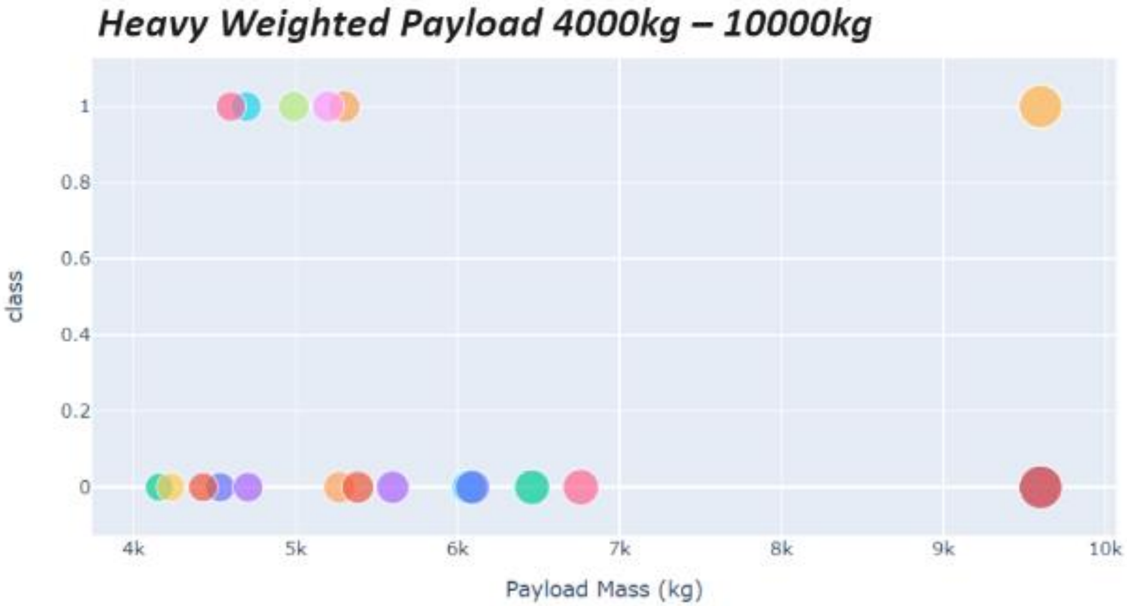
Pie chart showing the Launch site with the highest launch success ratio



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.


Low Weighted Payload 0kg – 4000kg


Heavy Weighted Payload 4000kg – 10000kg

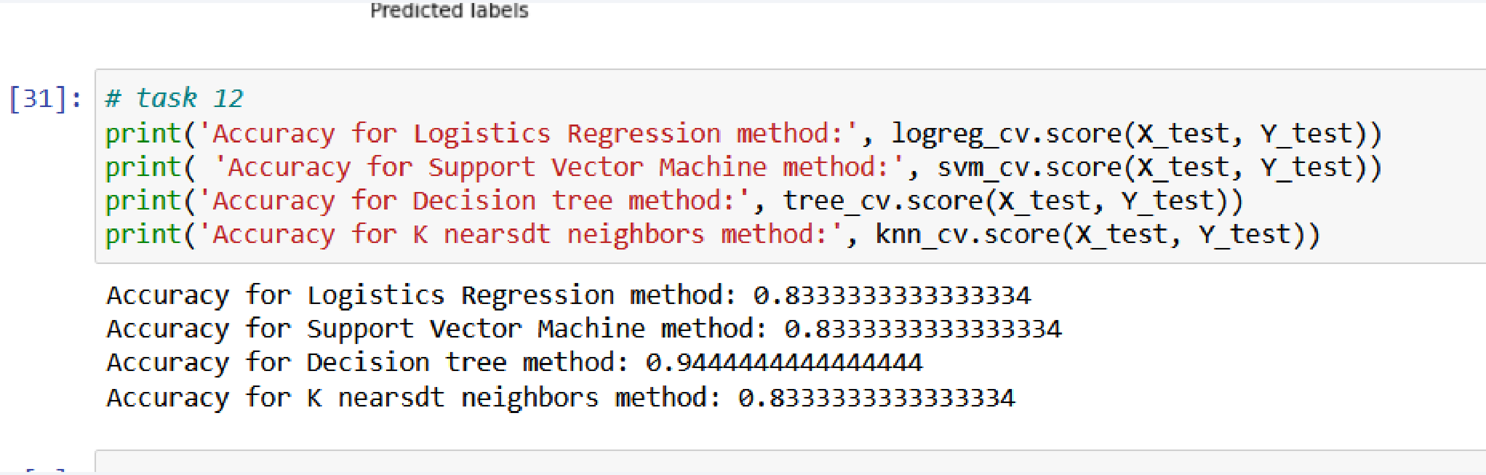
We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- As the printed accuracy of the four models shows the decision tree has the highest accuracy.

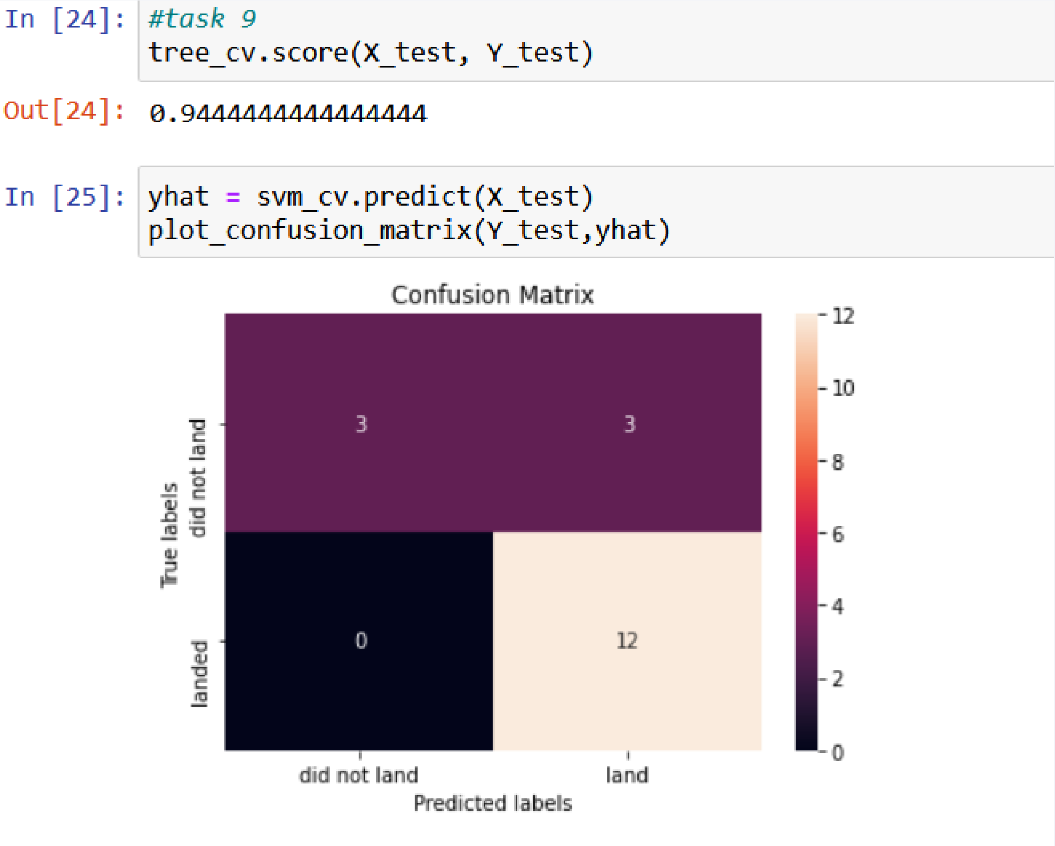


Predicted labels

```
[31]: # task 12
      print('Accuracy for Logistics Regression method:', logreg_cv.score(X_test, Y_test))
      print( 'Accuracy for Support Vector Machine method:', svm_cv.score(X_test, Y_test))
      print('Accuracy for Decision tree method:', tree_cv.score(X_test, Y_test))
      print('Accuracy for K nearsdt neighbors method:', knn_cv.score(X_test, Y_test))

      Accuracy for Logistics Regression method: 0.8333333333333334
      Accuracy for Support Vector Machine method: 0.8333333333333334
      Accuracy for Decision tree method: 0.9444444444444444
      Accuracy for K nearsdt neighbors method: 0.8333333333333334
```

Confusion Matrix

- The screen shot shows the confusion matrix of decision tree model. The accuracy Is over %90. The major problem is the false Positives. The unsuccessful landing marked as successful landing by the classifier.



```
In [24]: #task 9
         tree_cv.score(X_test, Y_test)

Out[24]: 0.9444444444444444

In [25]: yhat = svm_cv.predict(X_test)
         plot_confusion_matrix(Y_test,yhat)
```

Conclusions

- The Decision tree classifier is the best machine learning algorithm.
- The higher the flight numbers at a launch site, the greater the success rate.
- Launch success rate improved from in 2010 till 2019.
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- The lower the mass load the better was the performance.

Thank you!