IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Ali Mohammed A Asiri
26-08-2022

Outline

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

Executive Summary

- Summary of methodologies
  - 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 from Machine Learning Lab.

Introduction

SpaceX : is a revolutionary company who has disrupt the space industry by offering a rocket launches specifically in Falcon 9, as low as 62 million dollars while the other providers cost upward of 165 million dollar for each one. Most of this saving thanks to SpaceX astounding idea to reuse the first stage of the launch by re-land the rocket to be used on the next mission. Repeating this process will make the price down even further. As a data scientist of a startup rivaling SpaceX, the main goal of this project is to create the machine learning pipeline to predict the landing outcome of the first stage in the further. This project is crucial in identifying the right price to bid against SpaceX for a rocket launch.

The problems included:

- Identifying all factors that influence the landing outcome.
- The relationship between each variables and how it is affecting the outcome.
- The best condition needed to increase the probability of successful landing.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - I was collected data by using SpaceX RESR API and Web Scrapping from Wikipedia
- Perform data wrangling
  - I was processed the data using one-hot encoding for 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

- Data collection is the process of gathering and measuring information on targeted variables in an established system, which then enables one to answer relevant questions and evaluate outcomes. As mentioned, the dataset was collected by REST API and Web Scrapping from Wikipedia.

- For REST API, its started by using the get request. Then, we decoded the response content as Json and turn it into a pandas dataframe using json_normalize(). We then cleaned the data, checked for missing values and fill with whatever needed.

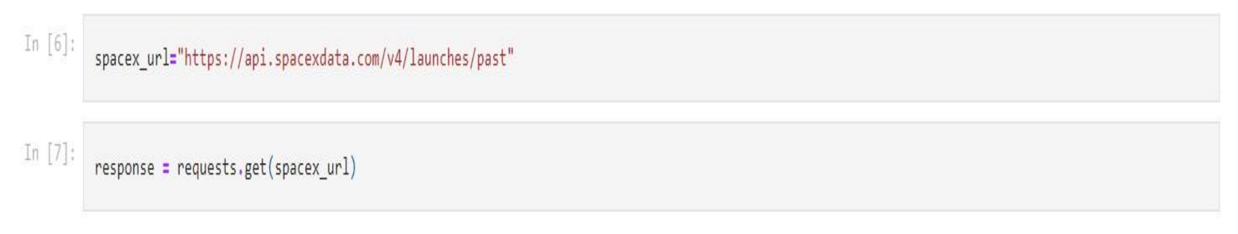
- For web scrapping, we will use the BeautifulSoup to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for further analysis

Data Collection – SpaceX API

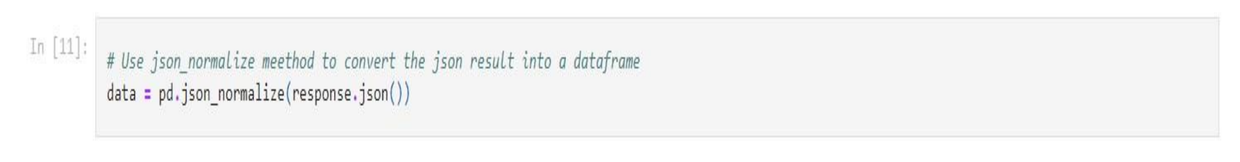
Get request for rocket launch data using API

Convert json result to dataframe by use json_normalize

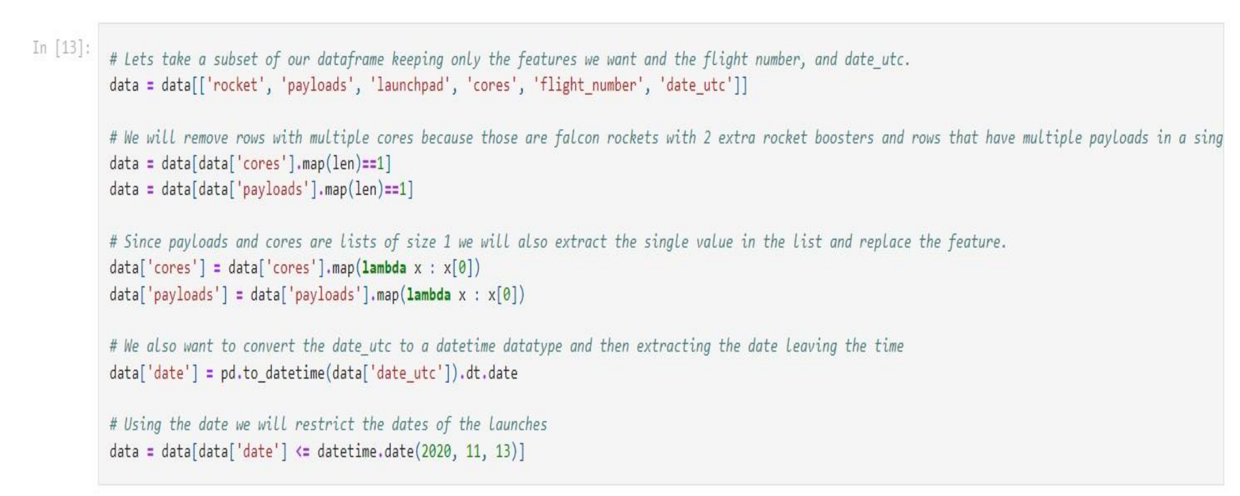
Filling the missing value and performed data cleaning



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

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



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



```
In [13]: # Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
         data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

         # We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that have multiple payloads in a sing
         data = data[data['cores'].map(len)==1]
         data = data[data['payloads'].map(len)==1]

         # Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the feature.
         data['cores'] = data['cores'].map(lambda x : x[0])
         data['payloads'] = data['payloads'].map(lambda x : x[0])

         # We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
         data['date'] = pd.to_datetime(data['date_utc']).dt.date

         # Using the date we will restrict the dates of the launches
         data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

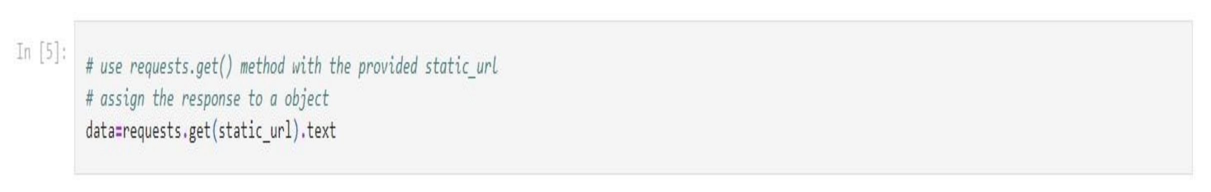
https://github.com/ams2021/Data-Science-and-Machine-Learning-Capstone-Project-SpaceX-/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Data Collection - Scraping

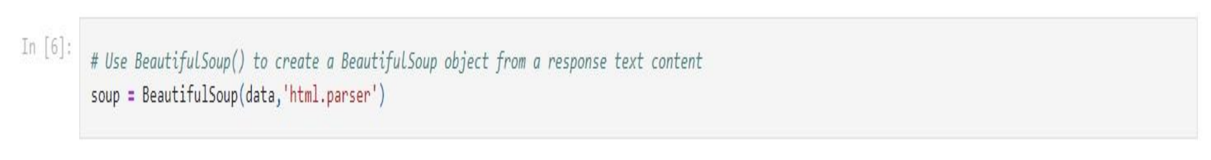
Request the Falcon9 launch wiki page from url

Create a BeautifulSoup from the html response

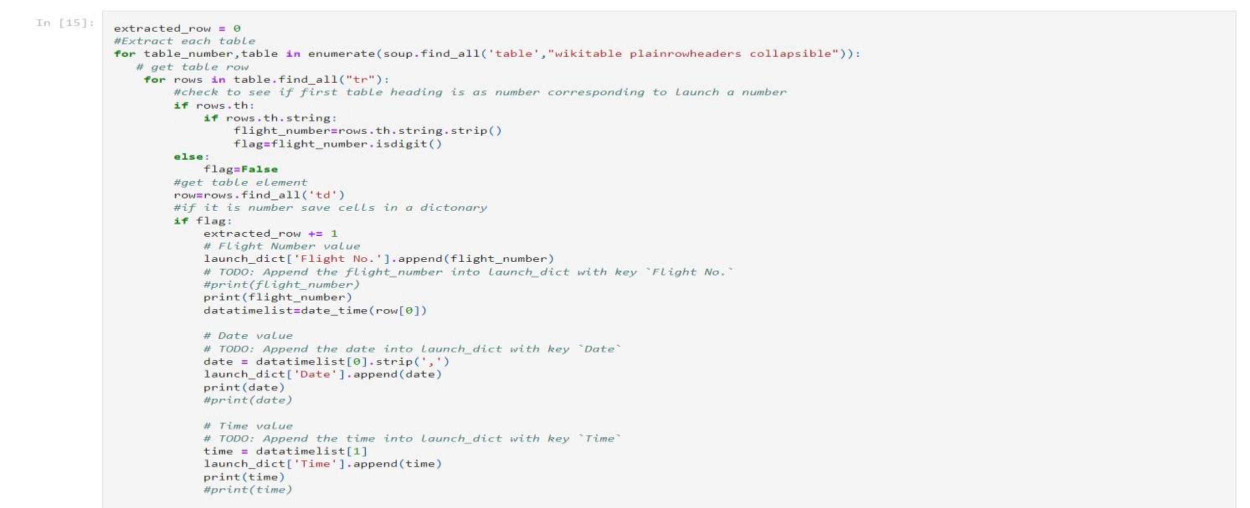
Extract all column names from the html header



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



```
In [6]:
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(data,'html.parser')
```



```
In [15]:
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
  # get table row
  for rows in table.find_all("tr"):
    #check to see if first table heading is as number corresponding to launch a number
    if rows.th:
      if rows.th.string:
        flight_number=rows.th.string.strip()
        flag=flight_number.isdigit()
    else:
      flag=False
    #get table element
    row=rows.find_all('td')
    #if it is number save cells in a dictonary
    if flag:
      extracted_row += 1
      # Flight Number value
      launch_dict['Flight No.'].append(flight_number)
      # TODO: Append the flight_number into launch_dict with key `Flight No.`
      #print(flight_number)
      print(flight_number)
      datatimelist=date_time(row[0])

      # Date value
      # TODO: Append the date into launch_dict with key `Date`
      date = datatimelist[0].strip(',')
      launch_dict['Date'].append(date)
      print(date)
      #print(date)

      # Time value
      # TODO: Append the time into launch_dict with key `Time`
      time = datatimelist[1]
      launch_dict['Time'].append(time)
      print(time)
      #print(time)
```

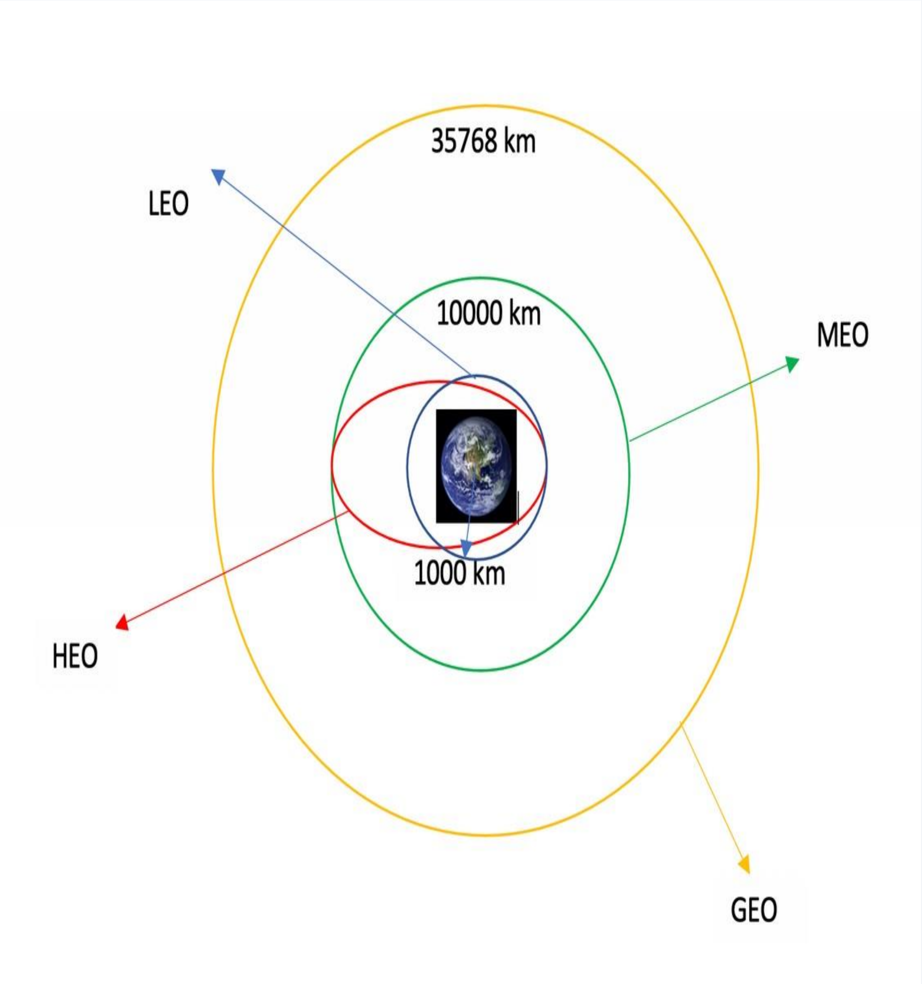
https://github.com/ams2021/Data-Science-and-Machine-Learning-Capstone-Project-SpaceX-/blob/main/jupyter-labs-webscraping.ipynb

Data Wrangling

Data Wrangling is the process of cleaning and unifying messy and complex data sets for easy access and Exploratory Data Analysis (EDA).

We will first calculate the number of launches on each site, then calculate the number and occurrence of mission outcome per orbit type.

We then create a landing outcome label from the outcome column. This will make it easier for further analysis, visualization, and ML. Lastly

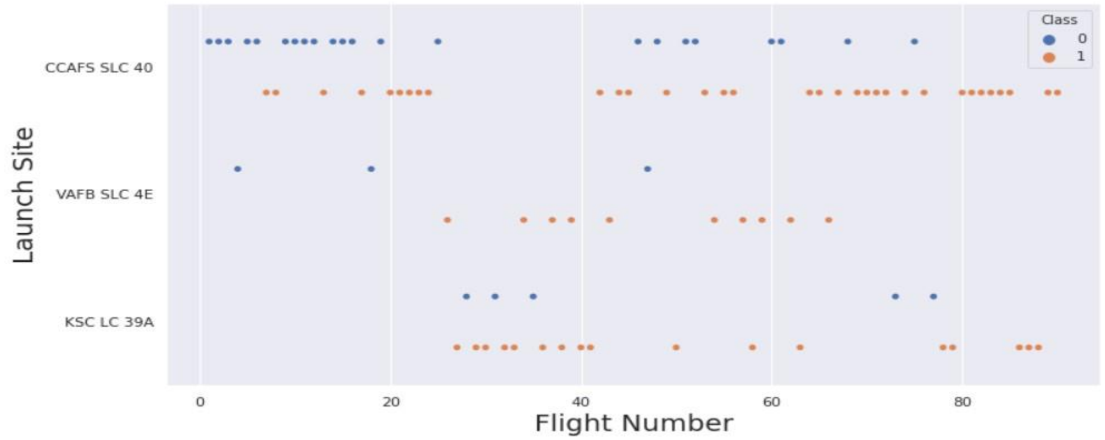


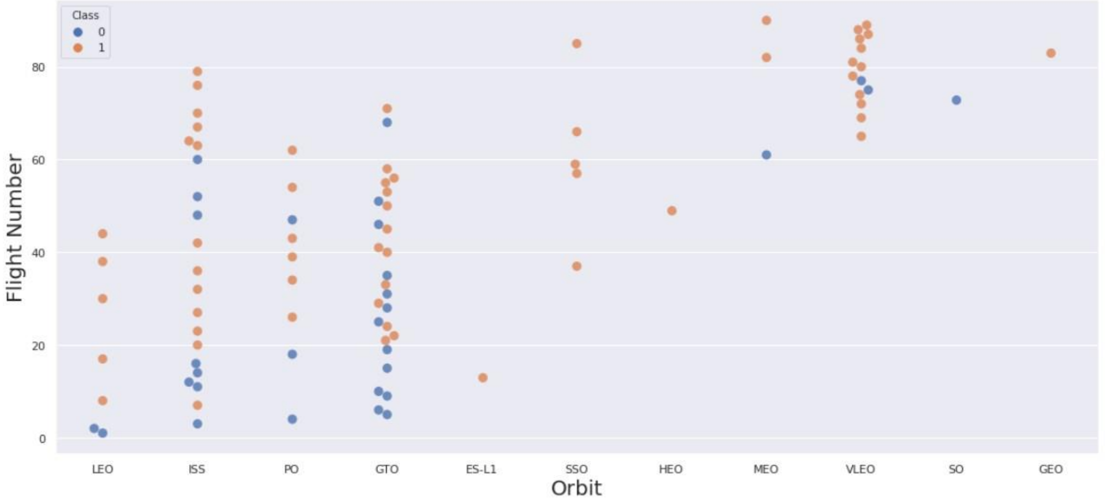
https://github.com/ams2021/Data-Science-and-Machine-Learning-Capstone-Project-SpaceX-/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- We first started by using scatter graph to find the relationship between the attributes such as between:
  - Payload and Flight Number.
  - Flight Number and Launch Site.
  - Payload and Launch Site.
  - Flight Number and Orbit Type.
  - Payload and Orbit Type.

Scatter plots show dependency of attributes on each other. Once a pattern is determined from the graphs. It’s very easy to see which factors affecting the most to the success of the landing outcomes.





https://github.com/ams2021/Data-Science-and-Machine-Learning-Capstone-Project-SpaceX-/blob/main/jupyter-labs-eda-dataviz.ipynb

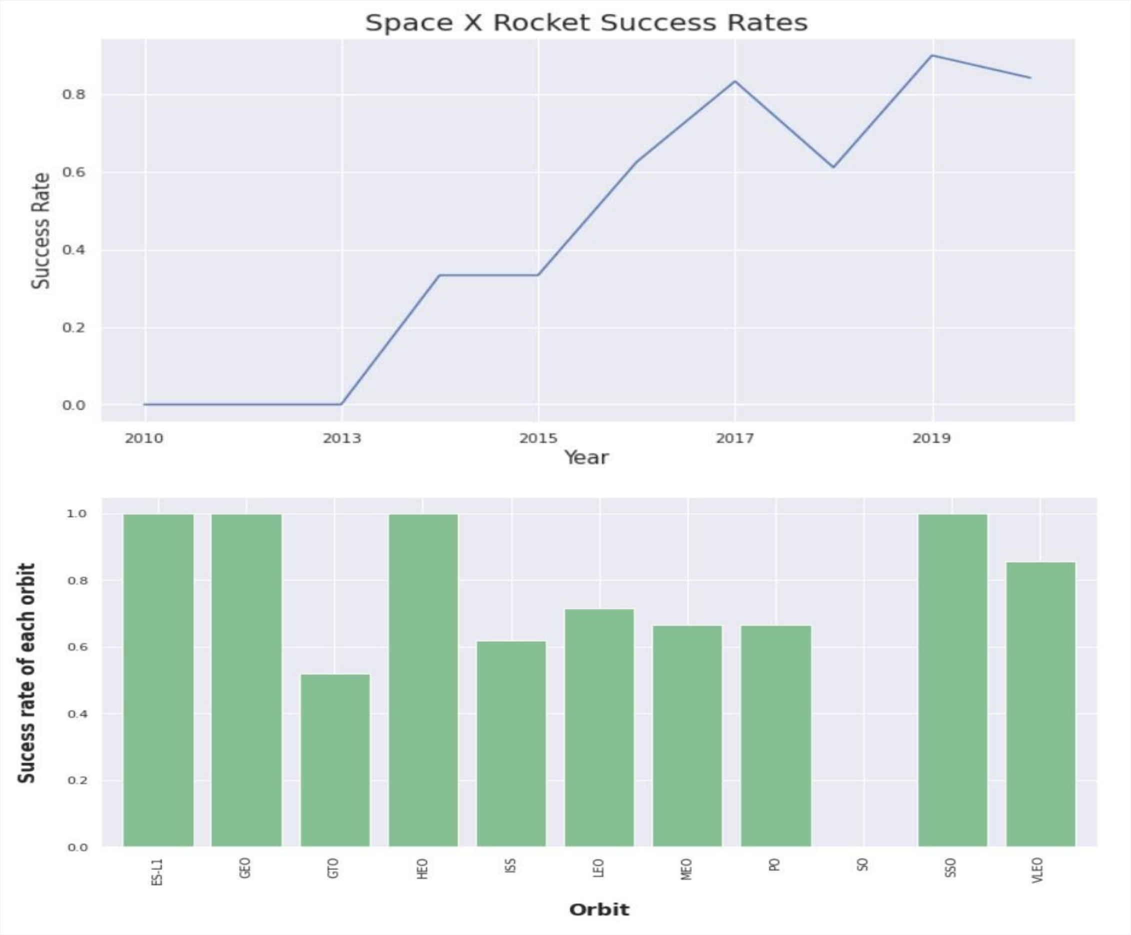
EDA with Data Visualization

- Once we get a hint of the relationships using scatter plot. We will then use further visualization tools such as bar graph and line plots graph for further analysis.

- Bar graphs is one of the easiest way to interpret the relationship between the attributes. In this case, we will use the bar graph to determine which orbits have the highest probability of success.

- We then use the line graph to show a trends or pattern of the attribute over time which in this case, is used for see the launch success yearly trend.

- We then use Feature Engineering to be used in success prediction in the future module by created the dummy variables to categorical columns.



https://github.com/ams2021/Data-Science-and-Machine-Learning-Capstone-Project-SpaceX-/blob/main/jupyter-labs-eda-dataviz.ipynb

EDA with SQL

https://github.com/ams2021/Data-Science-and-Machine-Learning-Capstone-Project-SpaceX-/blob/main/jupyter-labs-eda-sql-edx_sqllite.ipynb

- **Using SQL, we had performed many queries to get better understanding of the dataset, Ex: -**
  - Display the names of the unique launch sites in the space mission.
  - Display 5 records where launch sites begin with the string ‘KSC.
  - Display the total payload mass carried by boosters launched by NASA (CRS)
  - Display average payload mass carried by booster version F9 v1.1
  - Listing the date where the successful landing outcome in drone ship was acheived.
  - Listing the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000
  - Listing the total number of successful and failure mission outcomes.
  - Listing the names of the booster_versions which have carried the maximum payload mass.
  - Listing the records which will display the month names, successful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017
  - Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

Build an Interactive Map with Folium

- To visualize the launch data into an interactive map. We took the latitude and longitude coordinates at each launch site and added a circle marker around each launch site with a label of the name of the launch site.
- We then assigned the dataframe launch_outcomes(failure,success) to classes 0 and 1 with Red and Green markers on the map in MarkerCluster().
- We then used the Haversine's formula to calculated the distance of the launch sites to various landmark to find answer to the questions of:
  - How close the launch sites with railways, highways and coastlines?
  - How close the launch sites with nearby cities?

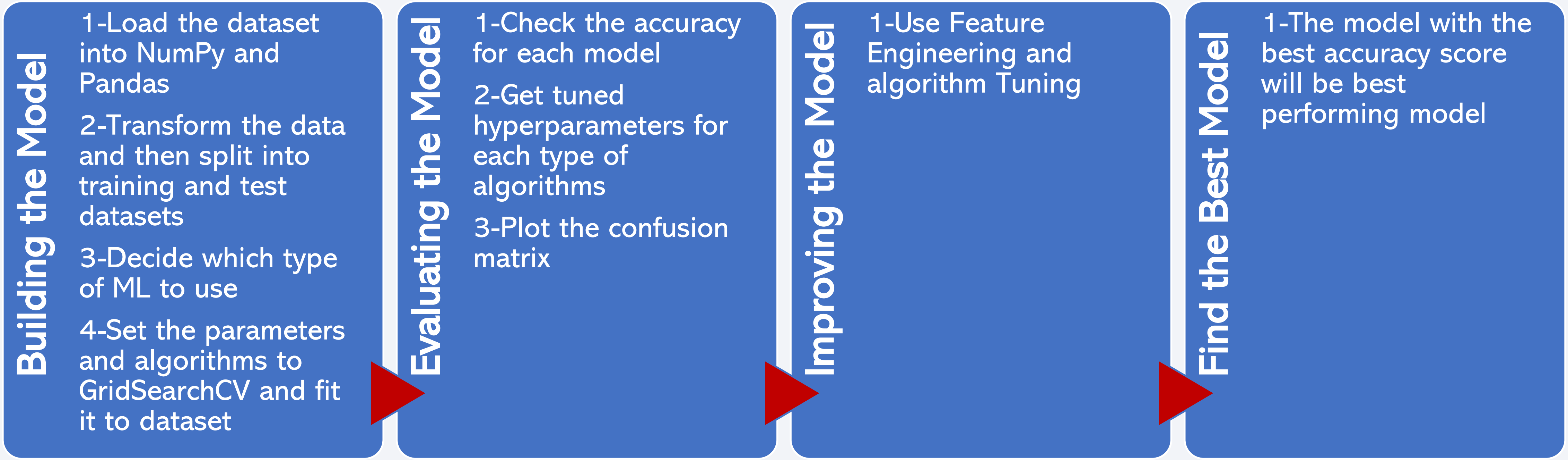
https://github.com/ams2021/Data-Science-and-Machine-Learning-Capstone-Project-SpaceX-/blob/main/lab_jupyter_launch_site_location.ipynb -

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash which allowing the user to play around with the data as they need.
- We plotted pie charts showing the total launches by a certain sites.
- We then plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.

https://github.com/ams2021/Data-Science-and-Machine-Learning-Capstone-Project-SpaceX-/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)



Building the Model

1-Load the dataset into NumPy and Pandas

2-Transform the data and then split into training and test datasets

3-Decide which type of ML to use

4-Set the parameters and algorithms to GridSearchCV and fit it to dataset

Evaluating the Model

1-Check the accuracy for each model

2-Get tuned hyperparameters for each type of algorithms

3-Plot the confusion matrix

Improving the Model

1-Use Feature Engineering and algorithm Tuning

Find the Best Model

1-The model with the best accuracy score will be best performing model

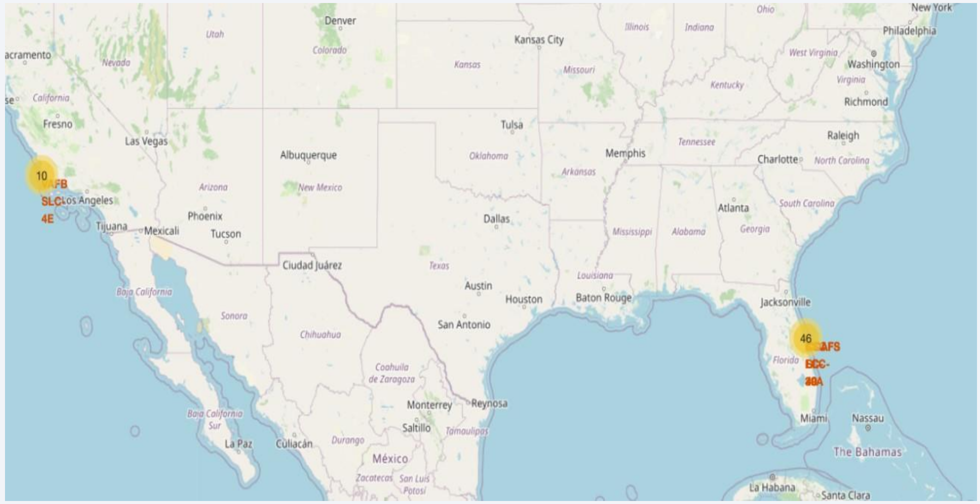
https://github.com/ams2021/Data-Science-and-Machine-Learning-Capstone-Project-SpaceX-/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

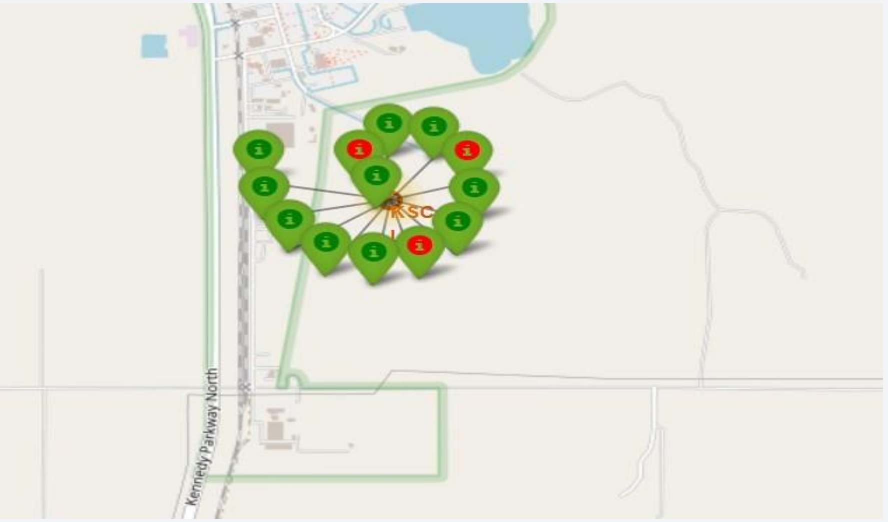
Results

- Exploratory data analysis results
  - In SpaceX uses 4 different Launch sites.
  - The successful Landing outcome in drone ship was achieved in 06-05-2016.
  - The average payload of F9 v1.1 booster is 2928.4 kg.
  - The Boosters which have success in ground pad and have payload mass between 4000 and 6000 are 3.
  - The total number of successful mission outcomes is 100 and the failure mission outcome is only one.
  - The total a count of successful landing outcomes between the date 04-06-2010 and 20-03-2017 are 34.

Results

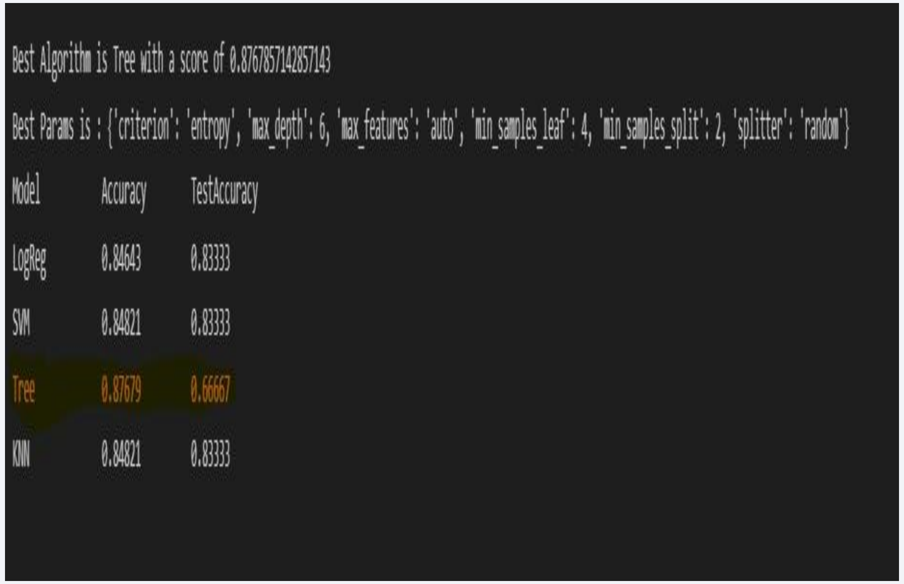
- Interactive analytics demo in screenshots
  - Using interactive analytics was possible to identify that launch sites use to be in safety places, near sea(very close proximity to the coast), and also all launch sites are in the north of the Equator line.
  - Most launches happens at east cost launch sites.
  - launch sites have relatively high success rates is KSC LC-39A.



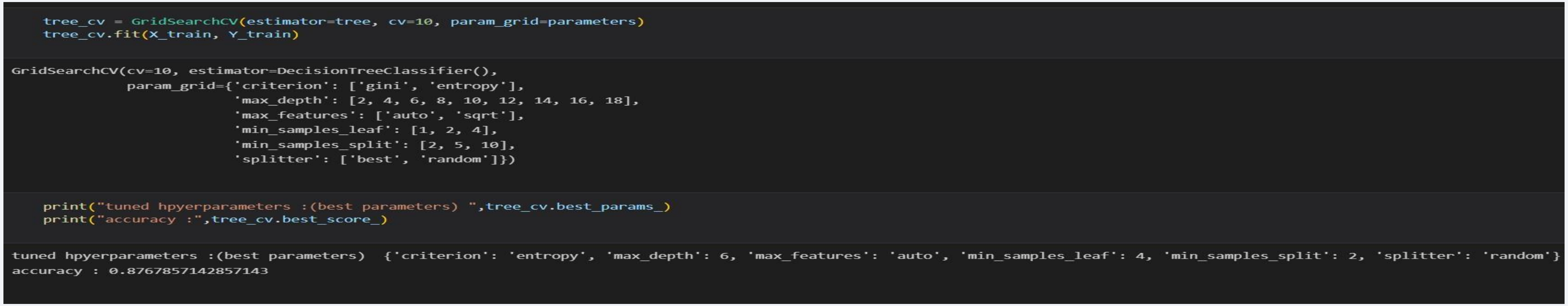


Results

- Predictive analysis results
  - Predictive Analysis showed that Decision Tree Classifier is the best model to predict successful landings, having accuracy over 87%.



```
Best Algorithm is Tree with a score of 0.8767857142857143

Best Params is : {'criterion': 'entropy', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 2, 'splitter': 'random'}

Model      Accuracy   TestAccuracy

LogReg     0.84643    0.83333

SVM        0.84821    0.83333

Tree       0.87679    0.66667

KNN        0.84821    0.83333
```



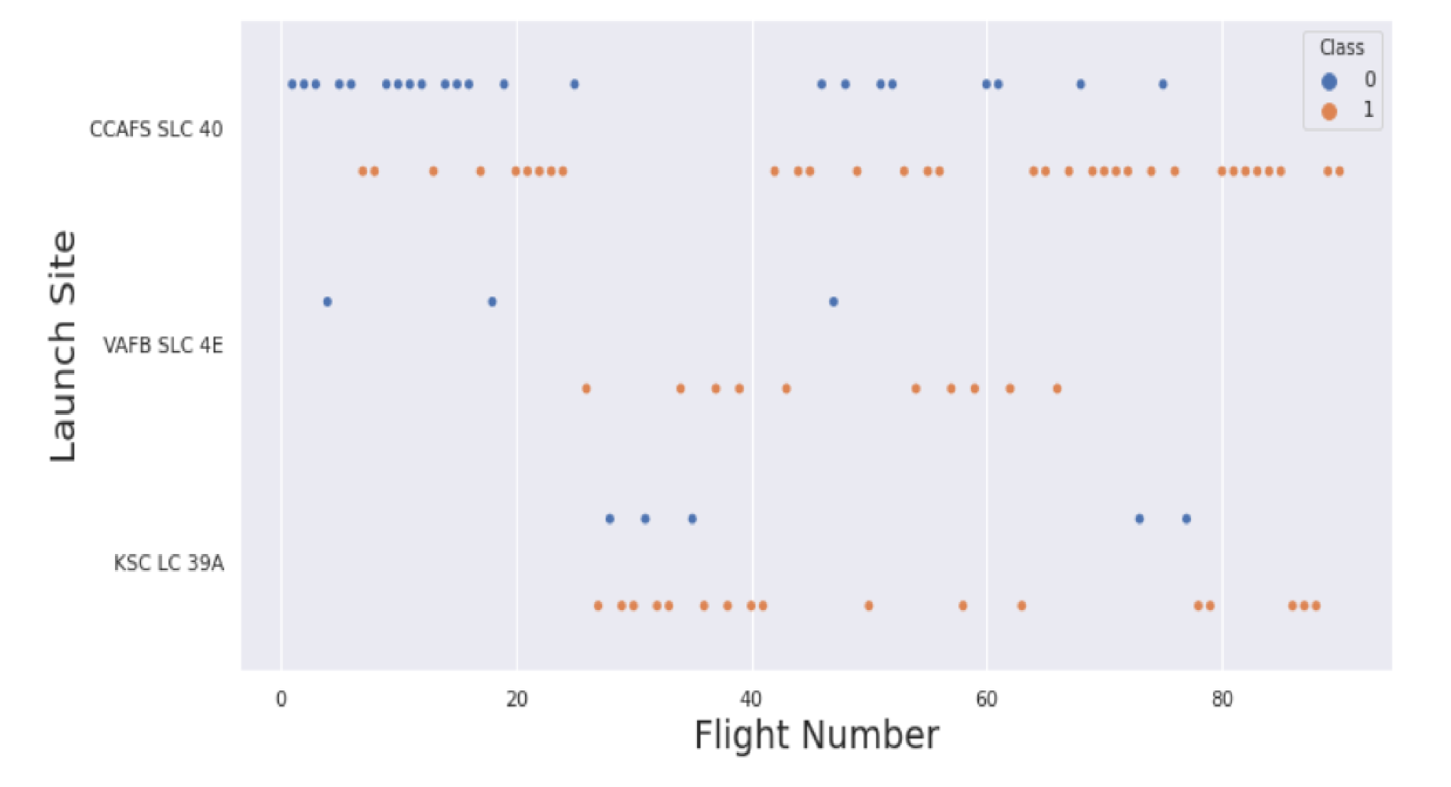
```
    tree_cv = GridSearchCV(estimator=tree, cv=10, param_grid=parameters)
    tree_cv.fit(X_train, Y_train)

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

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

tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 2, 'splitter': 'random'}
accuracy : 0.8767857142857143
```

Section 2

Insights drawn from EDA

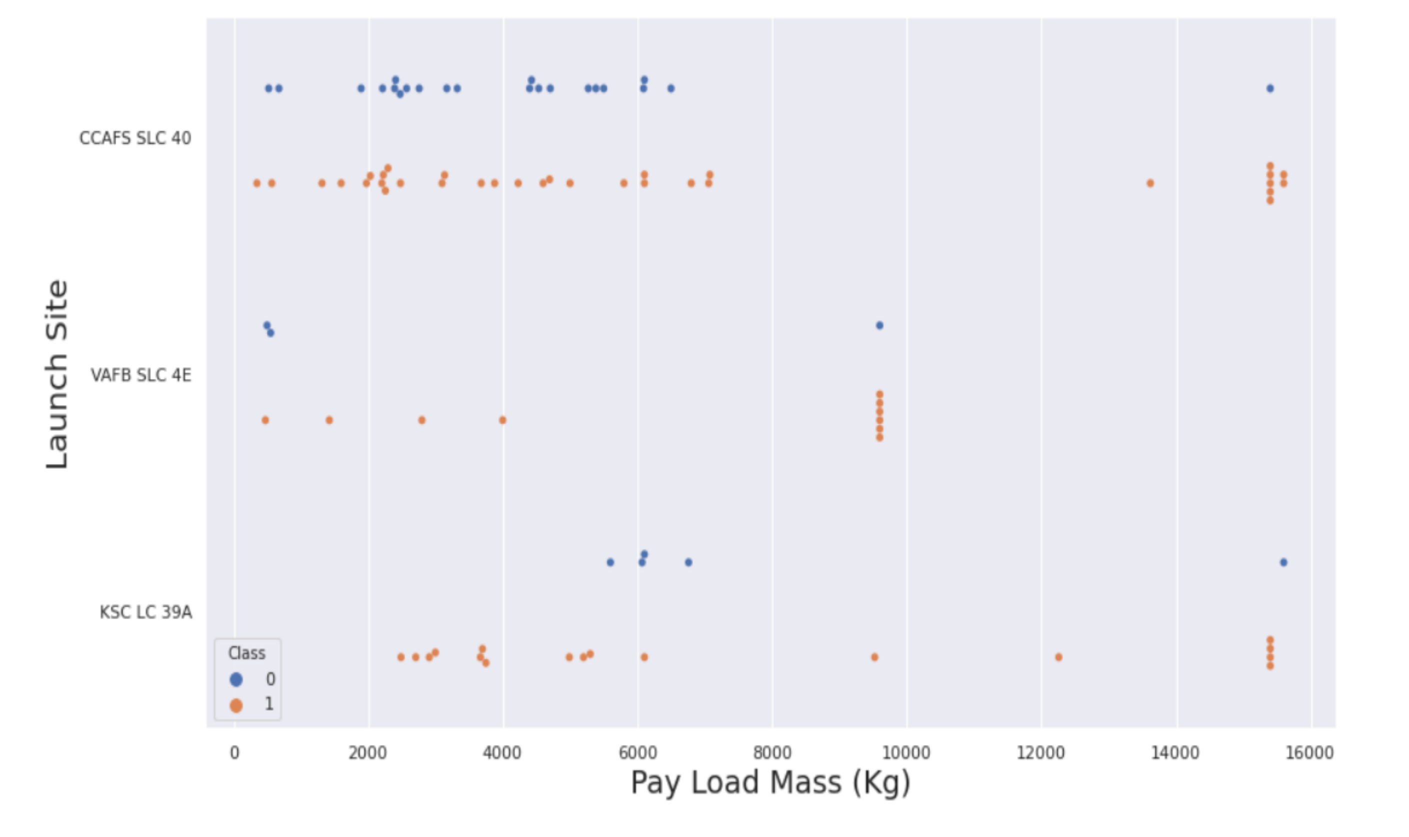
Flight Number vs. Launch Site

- This scatter plot shows that the larger the flights amount of the launch site, the greater the success rate will be.
- However, site CCAFS SLC40 shows the least pattern of this.
- According to the plot above, it’s possible to verify that the best launch site nowadays is CCAF5 SLC 40, where most of recent launches were successful.



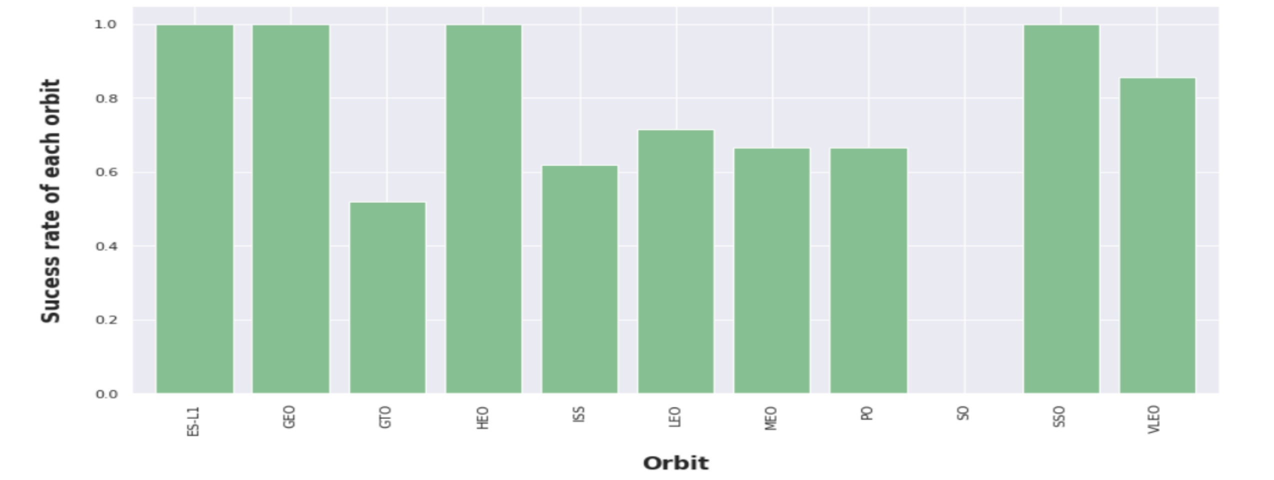
Payload vs. Launch Site

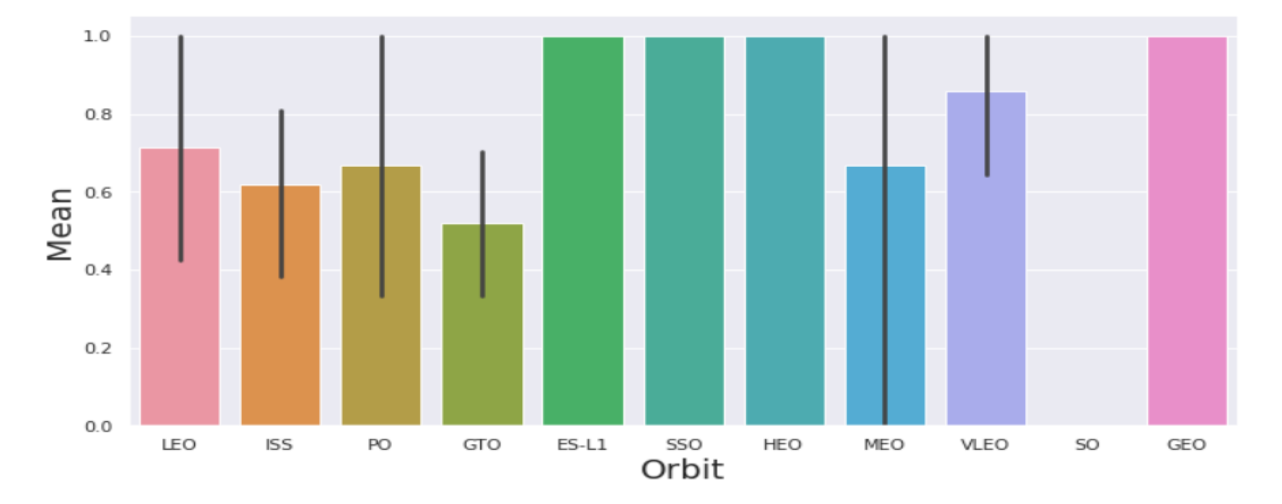
- This scatter plot shows once the pay load mass is greater than 7000kg, the probability of the success rate will be highly increased.
- However, there is no clear pattern to say the launch site is dependent to the pay load mass for the success rate.



Success Rate vs. Orbit Type

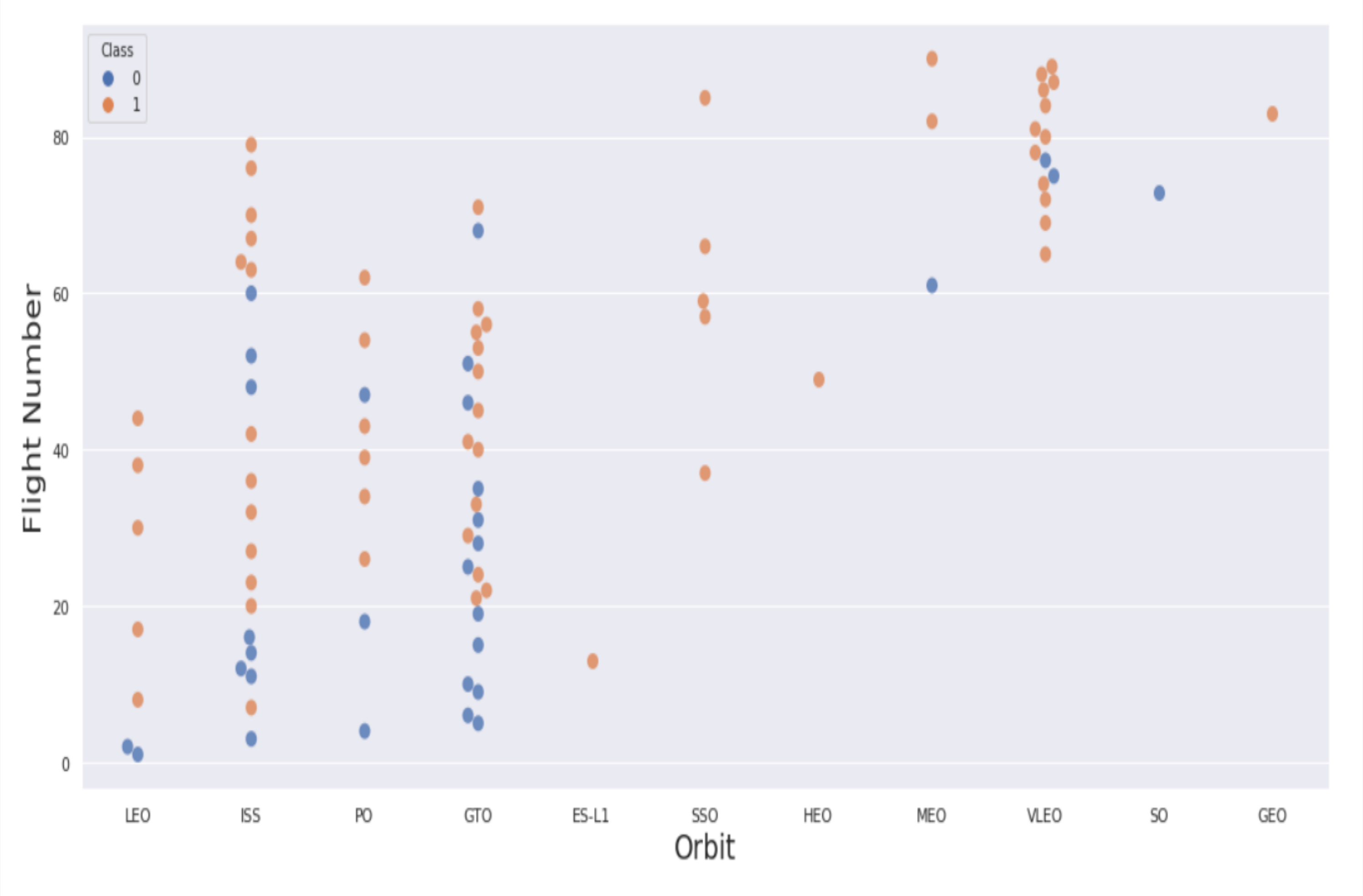
- This figure depicted the possibility of the orbits to influences the landing outcomes as some orbits has 100% success rate such as SSO, HEO, GEO AND ES-L1 while SO orbit produced 0% rate of success.
- However, deeper analysis show that some of this orbits has only 1 occurrence such as GEO, SO, HEO and ES-L1 which mean this data need more dataset to see pattern or trend before we draw any conclusion.





Flight Number vs. Orbit Type

- This scatter plot shows that generally, the larger the flight number on each orbits, the greater the success rate (especially LEO orbit) except for GTO orbit which depicts no relationship between both attributes.
- Orbit that only has 1 occurrence should also be excluded from above statement as it's needed more dataset.

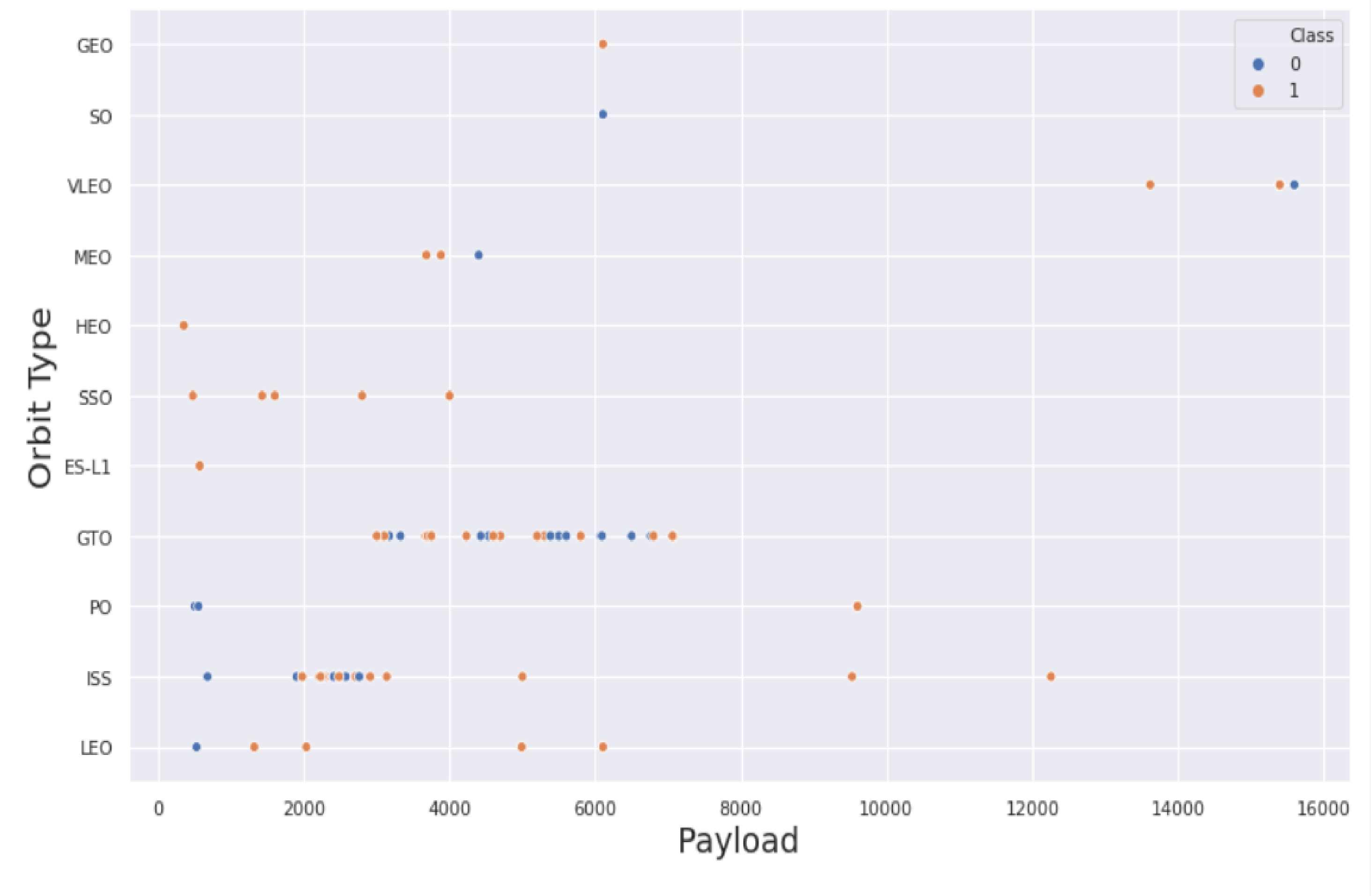


Payload vs. Orbit Type

- Heavier payload has positive impact on LEO, ISS and PO orbit. However, it has negative impact on MEO and VLEO orbit.

- ISS orbit has the widest range of payload and a good rate of success;

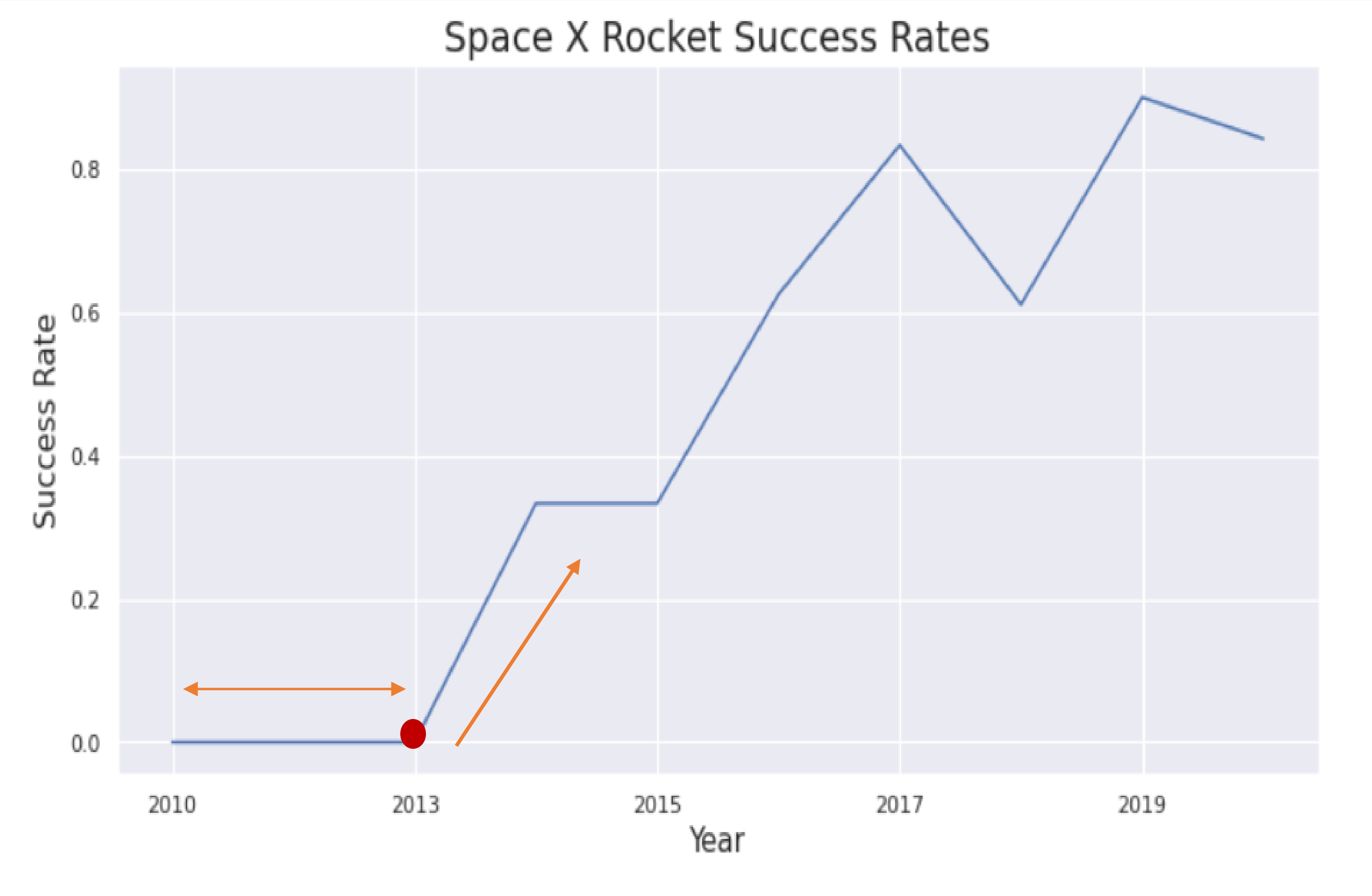
- There are few launches to the orbits SO and GEO.



Launch Success Yearly Trend

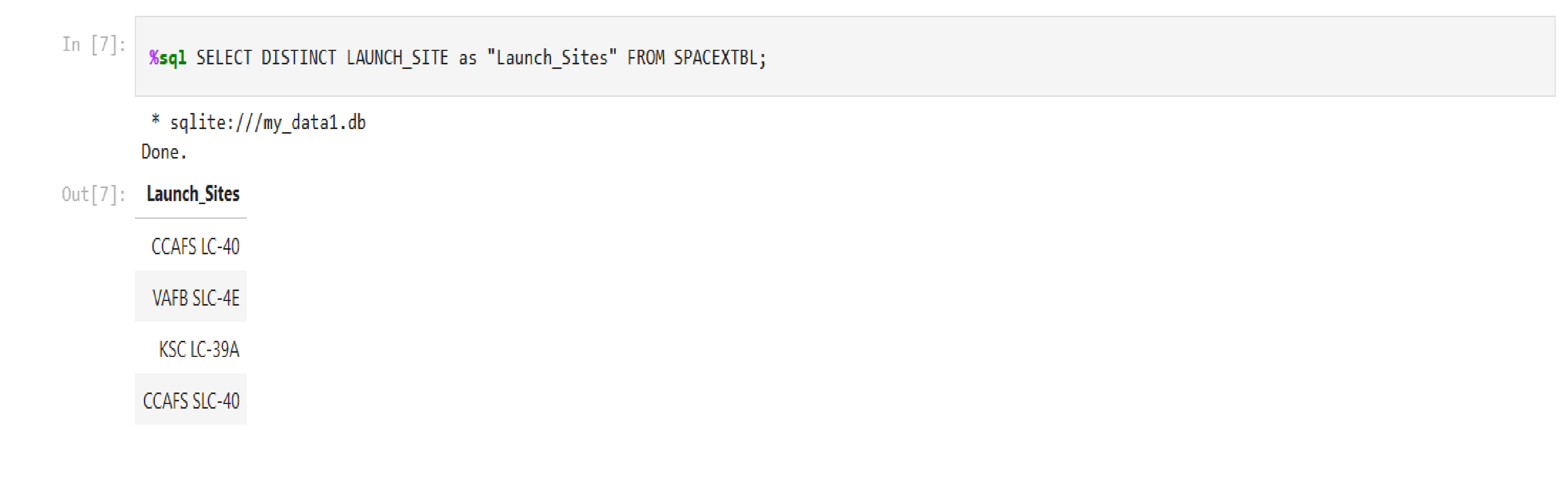
- This figures clearly depicted and increasing trend from the year 2013 until 2020.If this trend continue for the next year onward. The success rate will steadily increase until reaching100% success rate.

- It seems that the first three years were a period of adjusts and improvement of technology.



All Launch Site Names

- According to data, there are four launch sites.
- They are obtained by selecting unique occurrences of “launch_site” values from the dataset by using key word DISTINCT



In [7]:

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

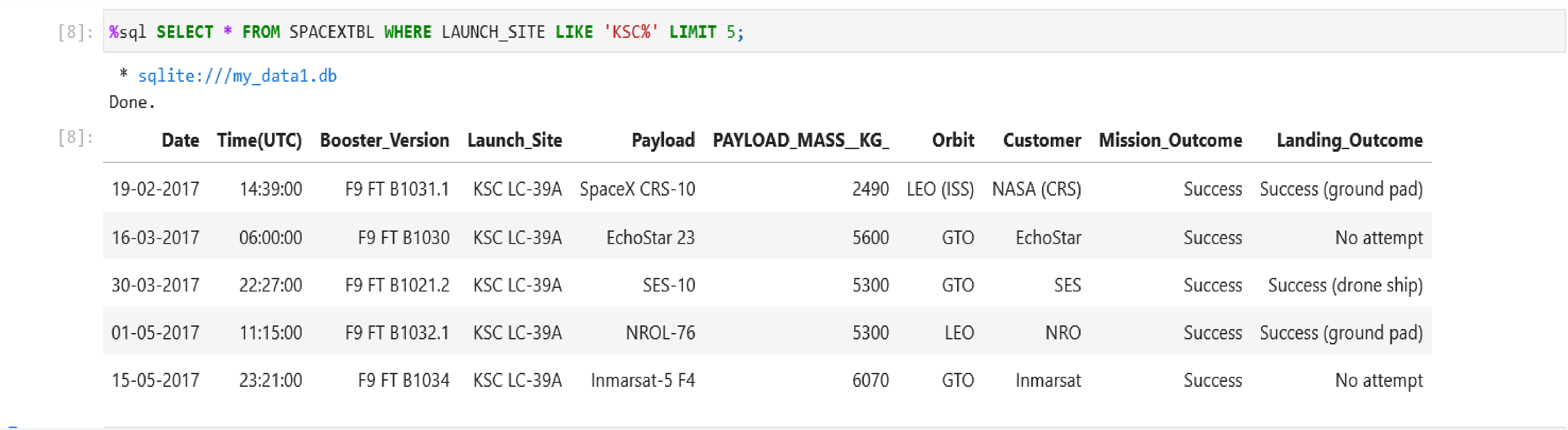
```
 * sqlite:///my_data1.db
Done.
```

Out[7]:

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

Launch Site Names Begin with 'KSC'

- We used query to displayed 5 records where launch sites' names start with `KSC` by used key word LIKE 'KSC%'.



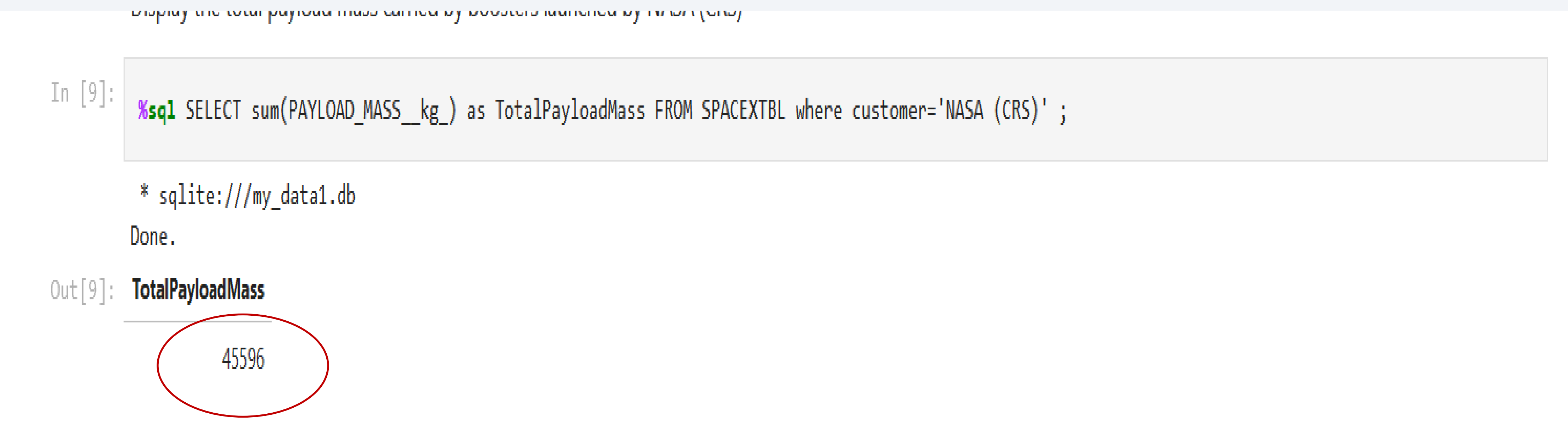
```
[8]: %sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'KSC%' LIMIT 5;

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

[8]:

Date	Time(UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
19-02-2017	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
16-03-2017	06:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	5600	GTO	EchoStar	Success	No attempt
30-03-2017	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
01-05-2017	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300	LEO	NRO	Success	Success (ground pad)
15-05-2017	23:21:00	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070	GTO	Inmarsat	Success	No attempt

Total Payload Mass

- We calculated the total payload carried by boosters from NASA as 45596 using the query below.



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

```
In [9]:  %sql SELECT sum(PAYLOAD_MASS__kg_) as TotalPayloadMass FROM SPACEXTBL where customer='NASA (CRS)' ;
```

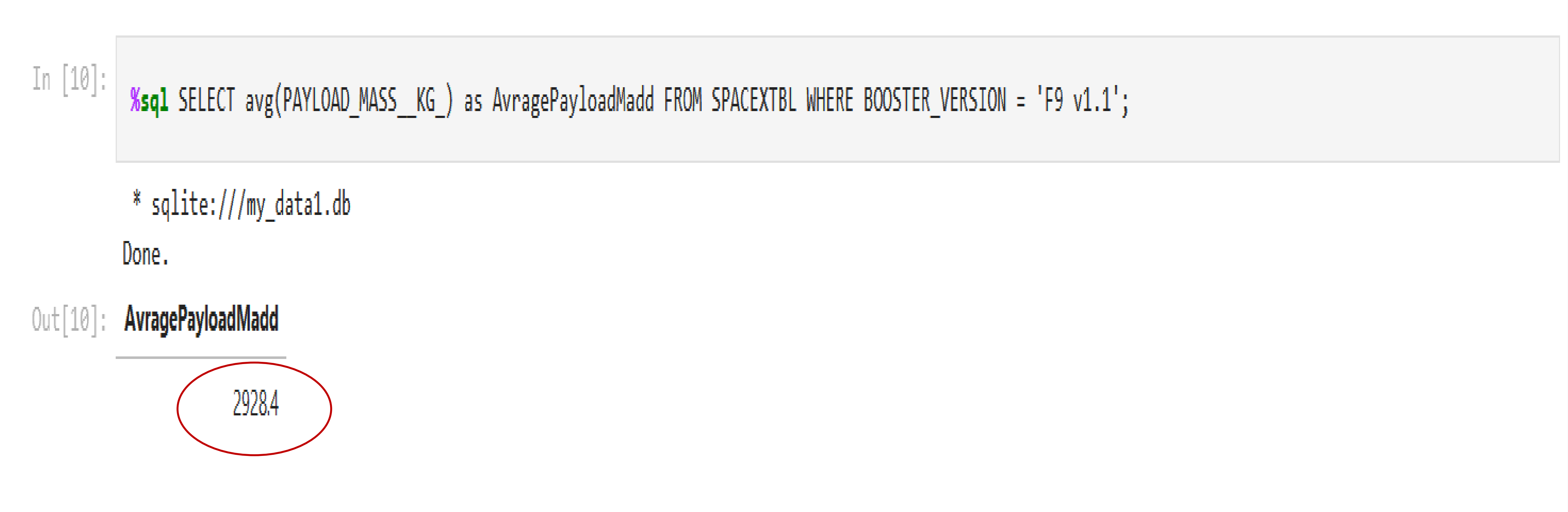
```
 * sqlite:///my_data1.db
Done.
```

Out[9]:

TotalPayloadMass
45596

Average Payload Mass by F9 v1.1

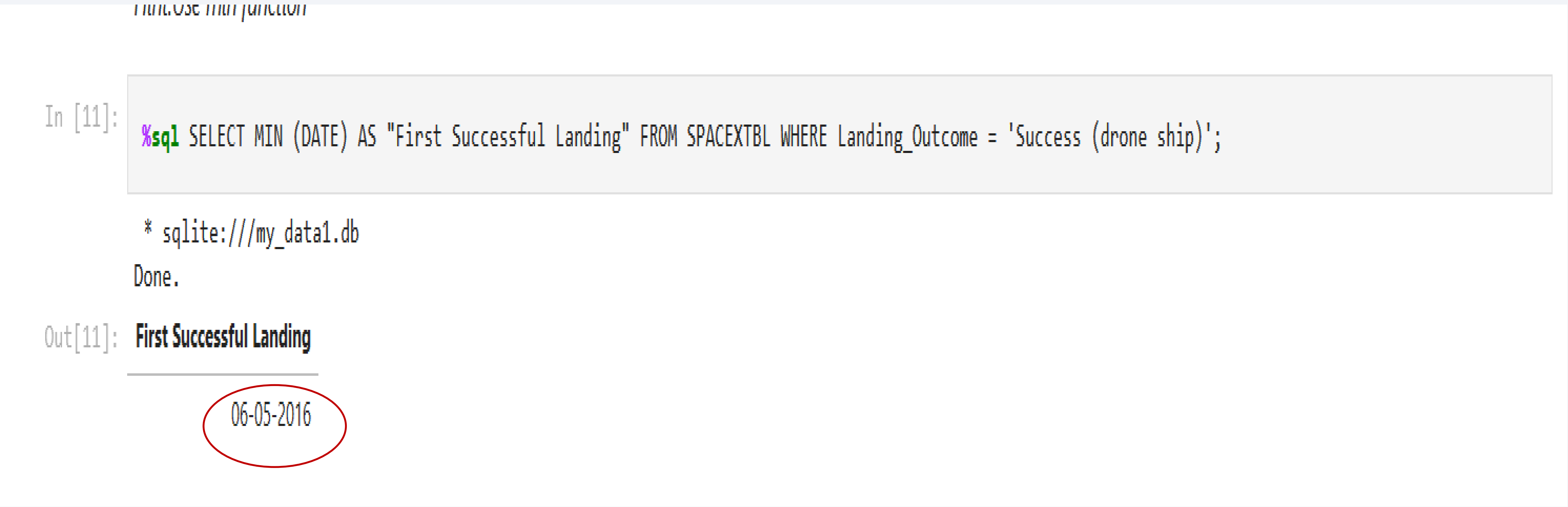
- We calculated the average payload mass carried by booster version F9 v1.1 as **2928.4**



```
In [10]:  %sql SELECT avg(PAYLOAD_MASS__KG_) as AvragePayloadMadd FROM SPACEXTBL WHERE BOOSTER_VERSION = 'F9 v1.1';

 * sqlite:///my_data1.db
Done.

Out[10]:  AvragePayloadMadd
              2928.4
```

First Successful Ground Landing Date

- We filtering data by successful landing outcome on Drone Ship and getting the minimum value for date it’s possible to identify the first occurrence, that happened on 06-05-2016.

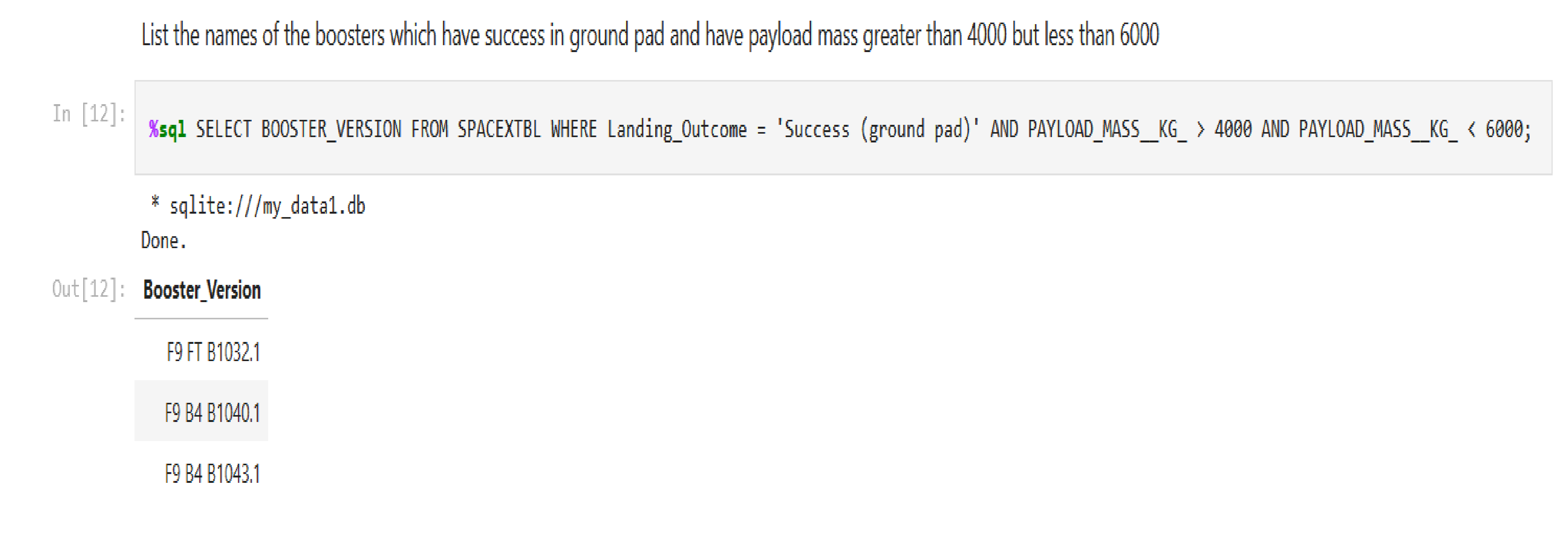


Hint:Use min function

```
In [11]: %sql SELECT MIN (DATE) AS "First Successful Landing" FROM SPACEXTBL WHERE Landing_Outcome = 'Success (drone ship)';

 * sqlite:///my_data1.db
Done.

Out[11]: First Successful Landing
              06-05-2016
```

Successful Ground Pad Landing with Payload between 4000 and 6000

- We used the WHERE clause to filter for boosters which have successfully landed on ground pad and applied the AND condition to determine successful landing with payload mass greater than 4000 but less than 6000.



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

```
In [12]: %sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE Landing_Outcome = 'Success (ground pad)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```

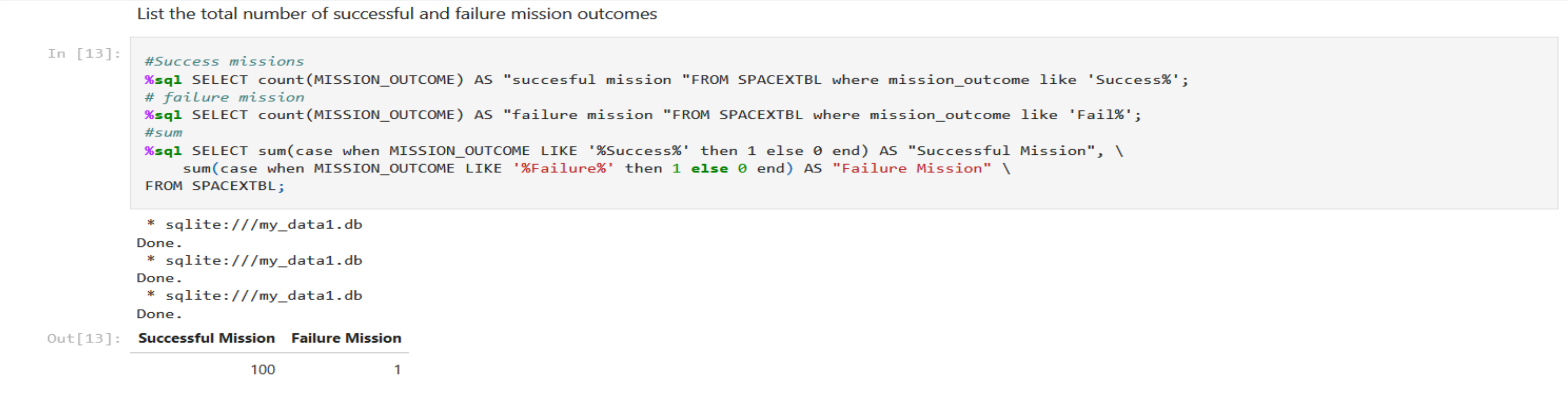
```
 * sqlite:///my_data1.db
Done.
```

Out[12]:

Booster_Version
F9 FT B1032.1
F9 B4 B1040.1
F9 B4 B1043.1

Total Number of Successful and Failure Mission Outcomes

- We used a key word (count, like '%') to filter for WHERE Mission Outcome was a success or a failure.
- Total number of success missions = 100.
- Total number of failure mission = 1



List the total number of successful and failure mission outcomes

```
In [13]:
#Success missions
%sql SELECT count(MISSION_OUTCOME) AS "succesful mission "FROM SPACEXTBL where mission_outcome like 'Success%';
# failure mission
%sql SELECT count(MISSION_OUTCOME) AS "failure mission "FROM SPACEXTBL where mission_outcome like 'Fail%';
#sum
%sql SELECT sum(case when MISSION_OUTCOME LIKE '%Success%' then 1 else 0 end) AS "Successful Mission", \
    sum(case when MISSION_OUTCOME LIKE '%Failure%' then 1 else 0 end) AS "Failure Mission" \
FROM SPACEXTBL;
```

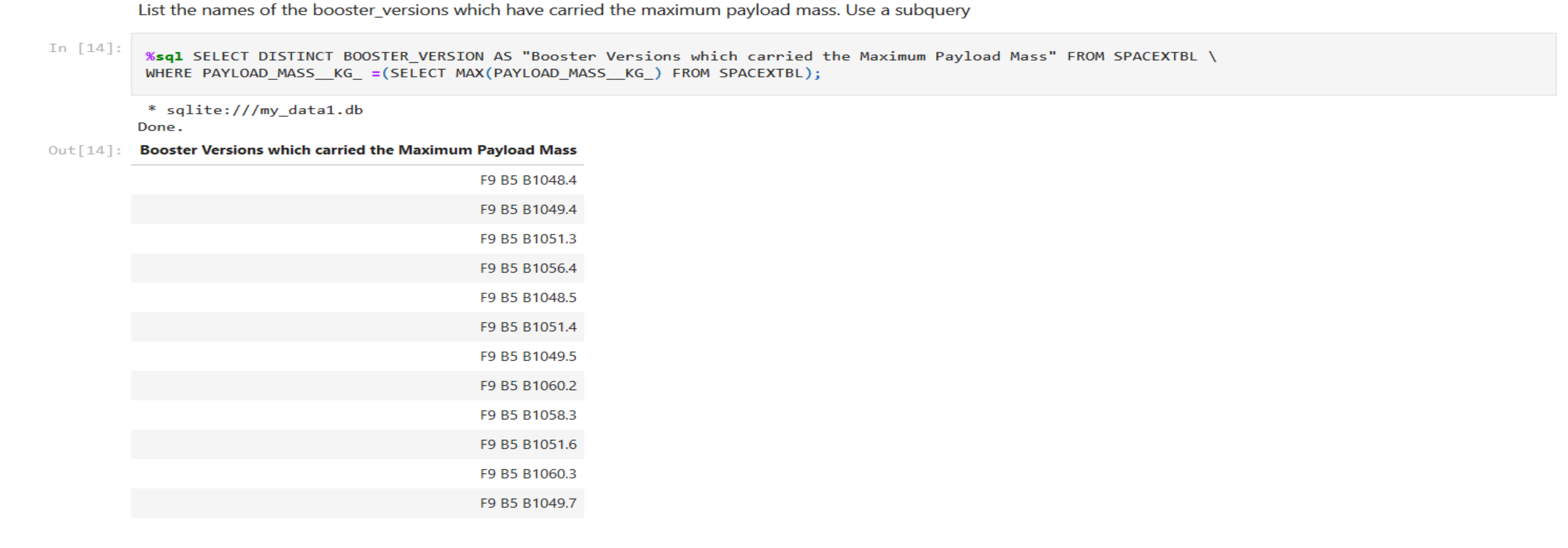
```
 * sqlite:///my_data1.db
Done.
 * sqlite:///my_data1.db
Done.
 * sqlite:///my_data1.db
Done.
```

Out[13]:

Successful Mission	Failure Mission
100	1

Boosters Carried Maximum Payload

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



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

In [14]:

```
%sql SELECT DISTINCT BOOSTER_VERSION AS "Booster Versions which carried the Maximum Payload Mass" FROM SPACEXTBL \
WHERE PAYLOAD_MASS__KG_ =(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

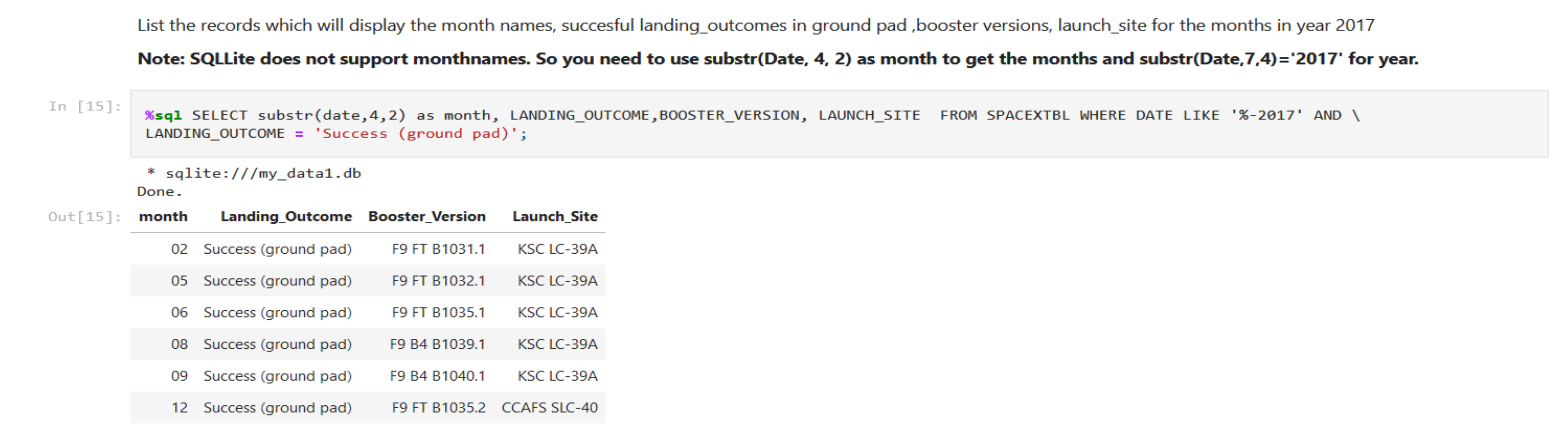
```
 * sqlite:///my_data1.db
Done.
```

Out[14]:

Booster Versions which carried the Maximum Payload Mass
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

2017 Launch Records

- We used subtsr to display the month number for successful landing outcomes in ground pad ,booster versions, launch_site for the months in year 2017
- The result bellow the month names from (February to December 2017)



List the records which will display the month names, succesful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017

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)='2017' for year.

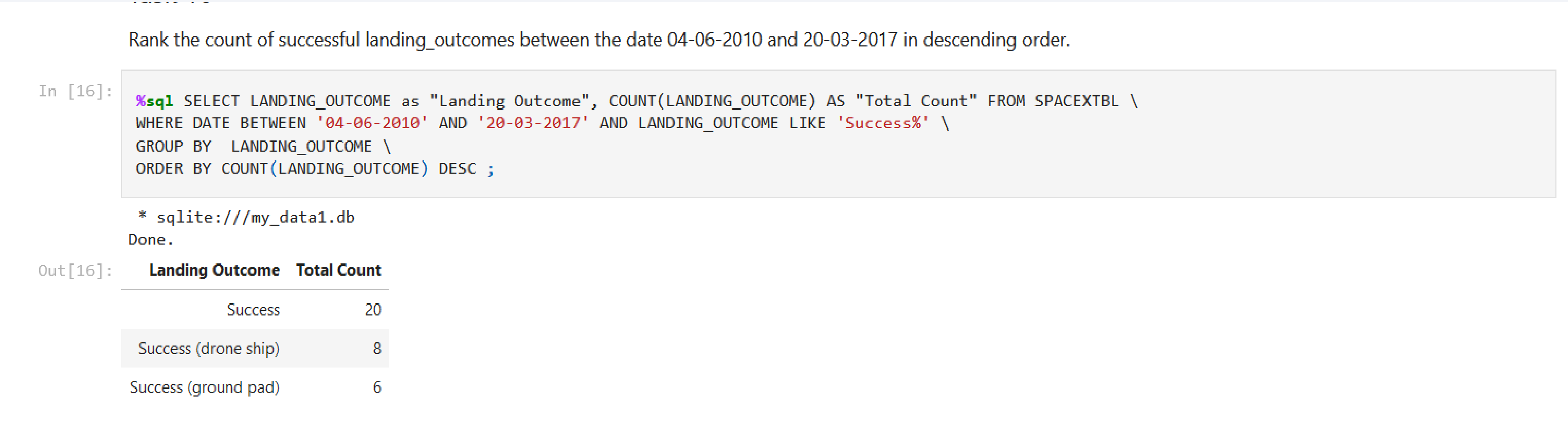
```
In [15]: %sql SELECT substr(date,4,2) as month, LANDING_OUTCOME,BOOSTER_VERSION, LAUNCH_SITE  FROM SPACEXTBL WHERE DATE LIKE '%-2017' AND \
         LANDING_OUTCOME = 'Success (ground pad)';

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

Out[15]:

month	Landing_Outcome	Booster_Version	Launch_Site
02	Success (ground pad)	F9 FT B1031.1	KSC LC-39A
05	Success (ground pad)	F9 FT B1032.1	KSC LC-39A
06	Success (ground pad)	F9 FT B1035.1	KSC LC-39A
08	Success (ground pad)	F9 B4 B1039.1	KSC LC-39A
09	Success (ground pad)	F9 B4 B1040.1	KSC LC-39A
12	Success (ground pad)	F9 FT B1035.2	CCAFS SLC-40

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

- We selected Landing outcomes and the COUNT of landing outcomes from the data and used the WHERE clause to filter for landing outcomes BETWEEN 04-06-2010 to 20-03-2017. and applied the GROUP BY clause to group the landing outcomes and the ORDER BY clause to order the grouped landing outcome in descending order.



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

```
In [16]: %sql SELECT LANDING_OUTCOME as "Landing Outcome", COUNT(LANDING_OUTCOME) AS "Total Count" FROM SPACEXTBL \
         WHERE DATE BETWEEN '04-06-2010' AND '20-03-2017' AND LANDING_OUTCOME LIKE 'Success%' \
         GROUP BY  LANDING_OUTCOME \
         ORDER BY COUNT(LANDING_OUTCOME) DESC ;
```

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

Out[16]:

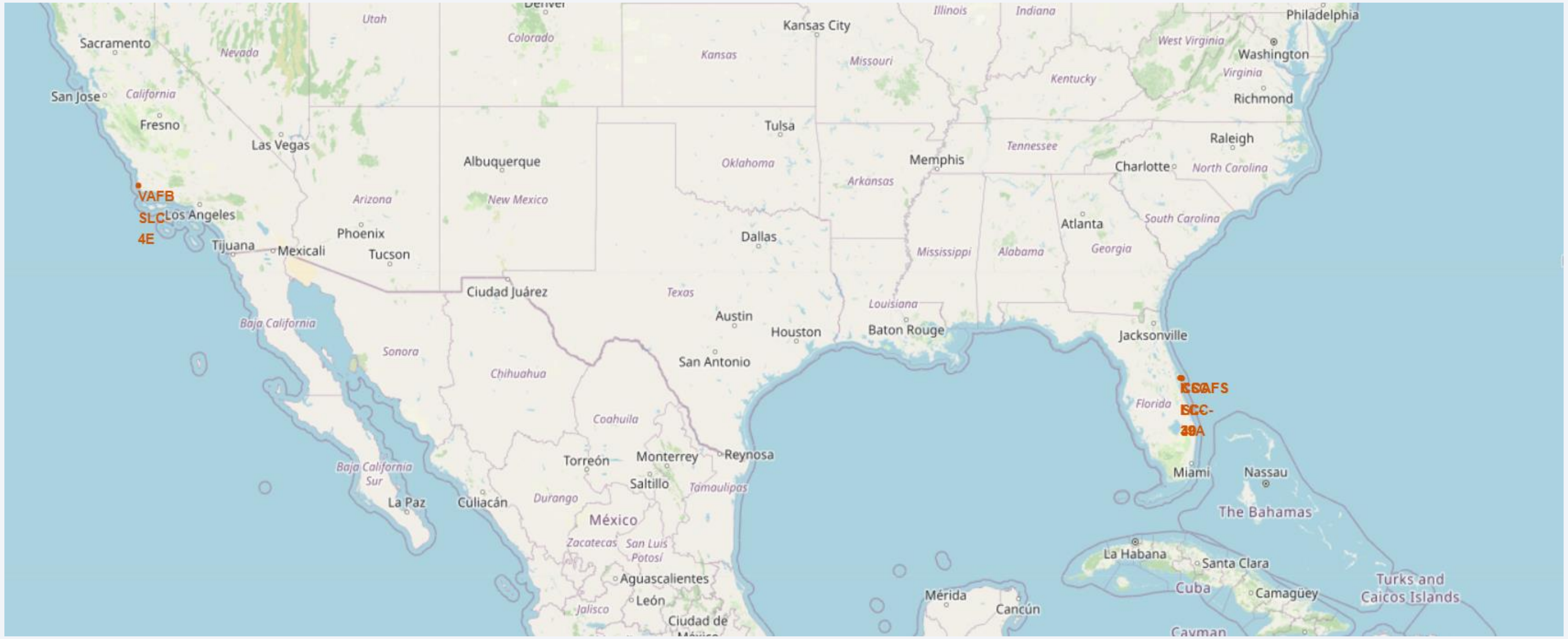
Landing Outcome	Total Count
Success	20
Success (drone ship)	8
Success (ground pad)	6

Section 3

Launch Sites Proximities Analysis

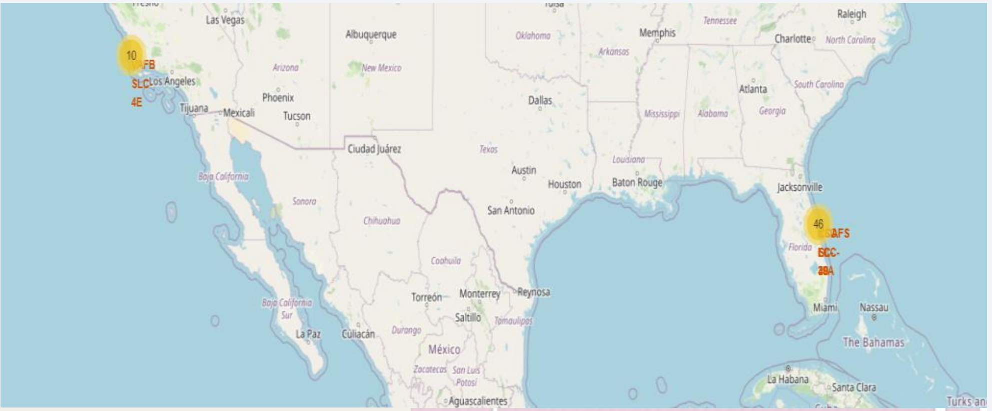
Mark all launch sites on a Map

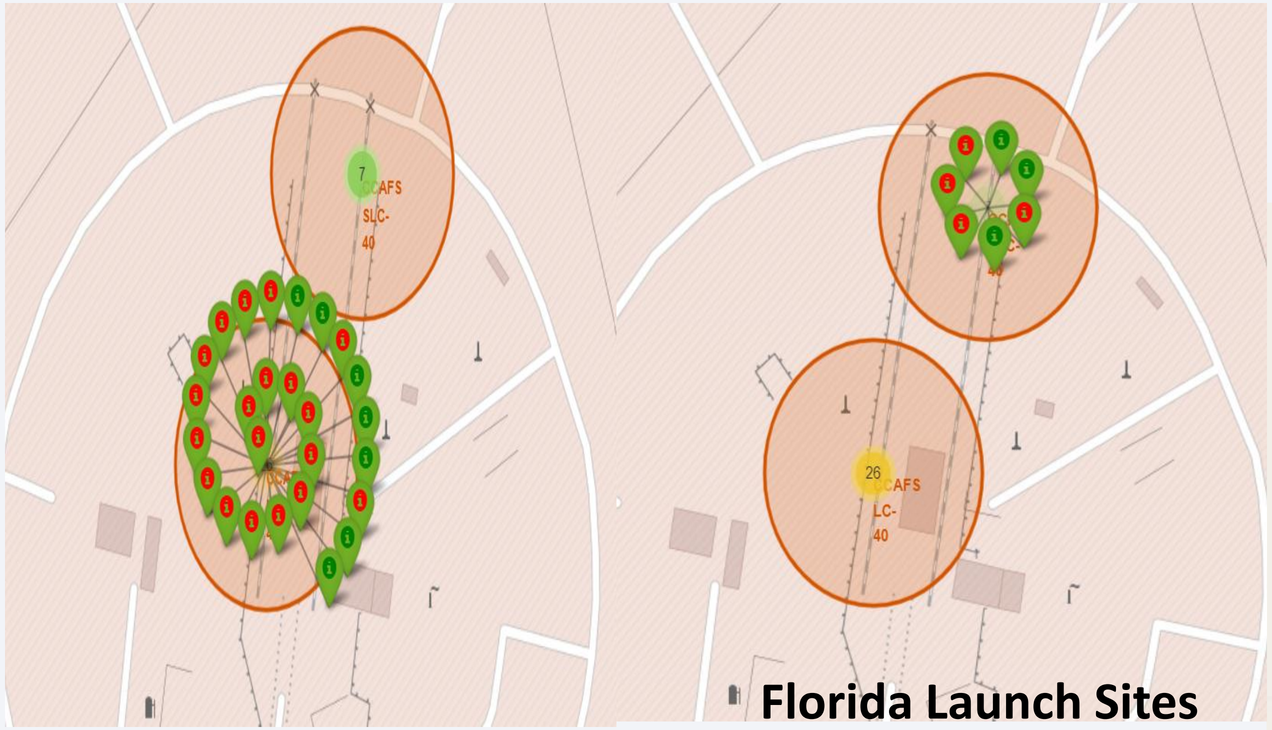
- We can see that all the SpaceX launch sites are located inside the United States.
- Launch sites are near sea, probably by safety, but not too far from roads and railroads.
- all launch sites are in the north of the Equator line.



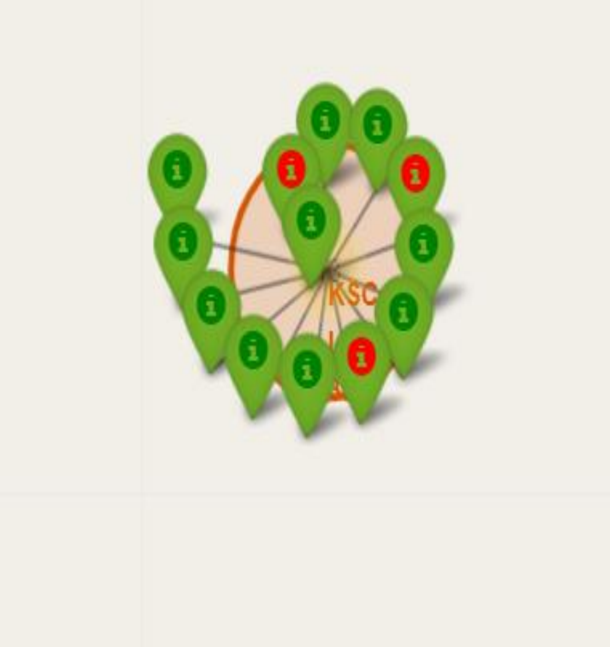
Markers showing launch sites with color labels

- Green Marker shows successful Launches and Red Marker shows Failures.





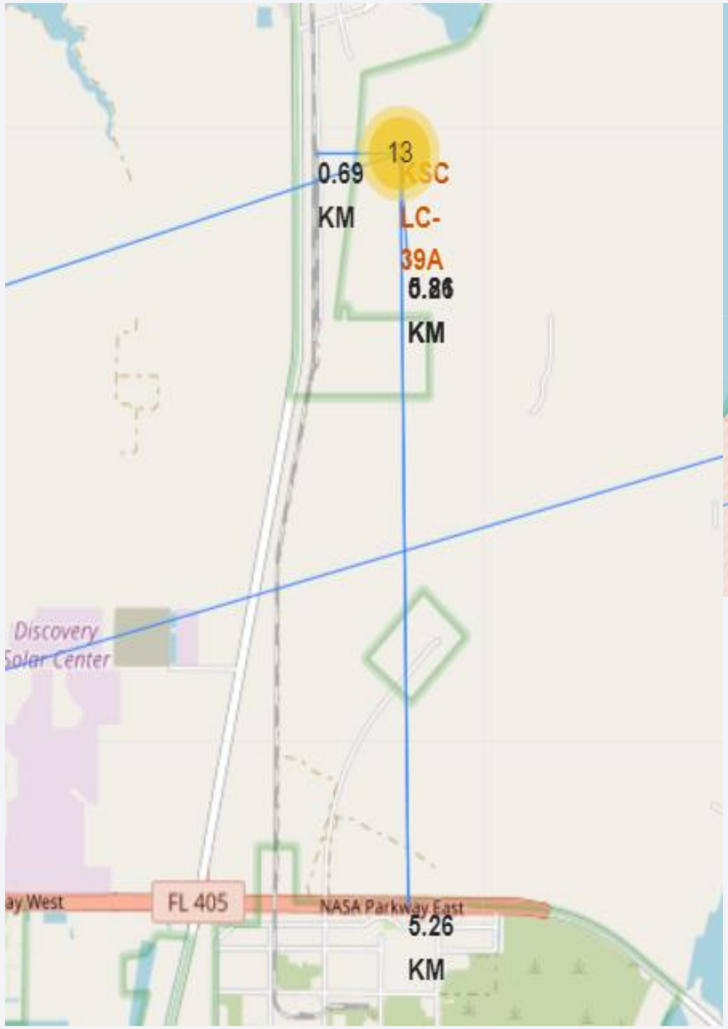
Florida Launch Sites

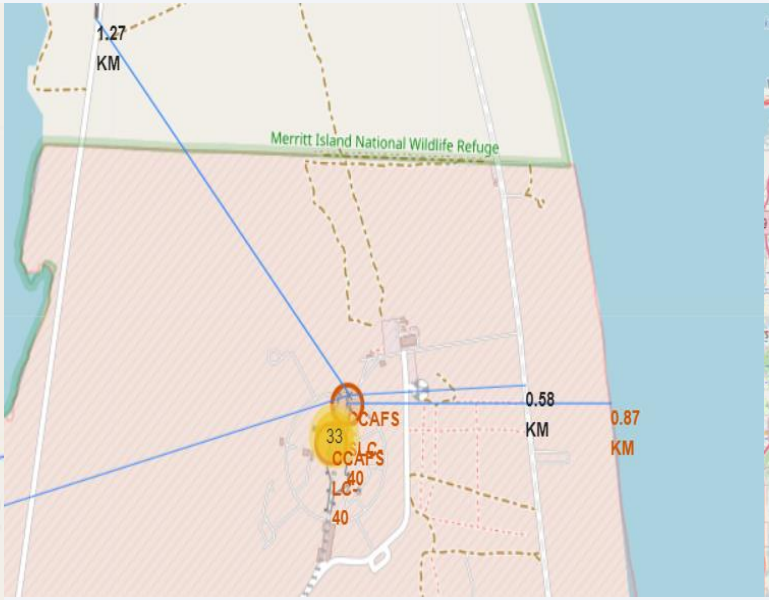


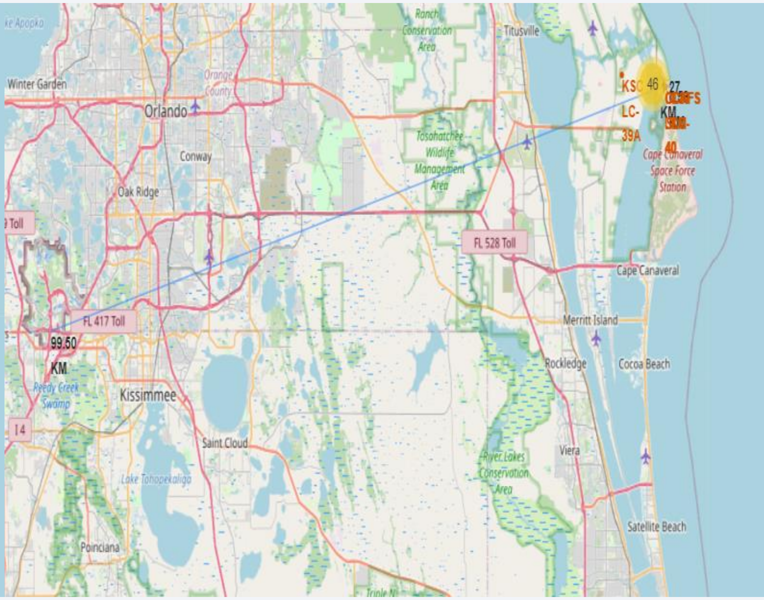
VAFB
Dix Road
Launch Tower

California Launch Site

Calculate Launch Sites Distance





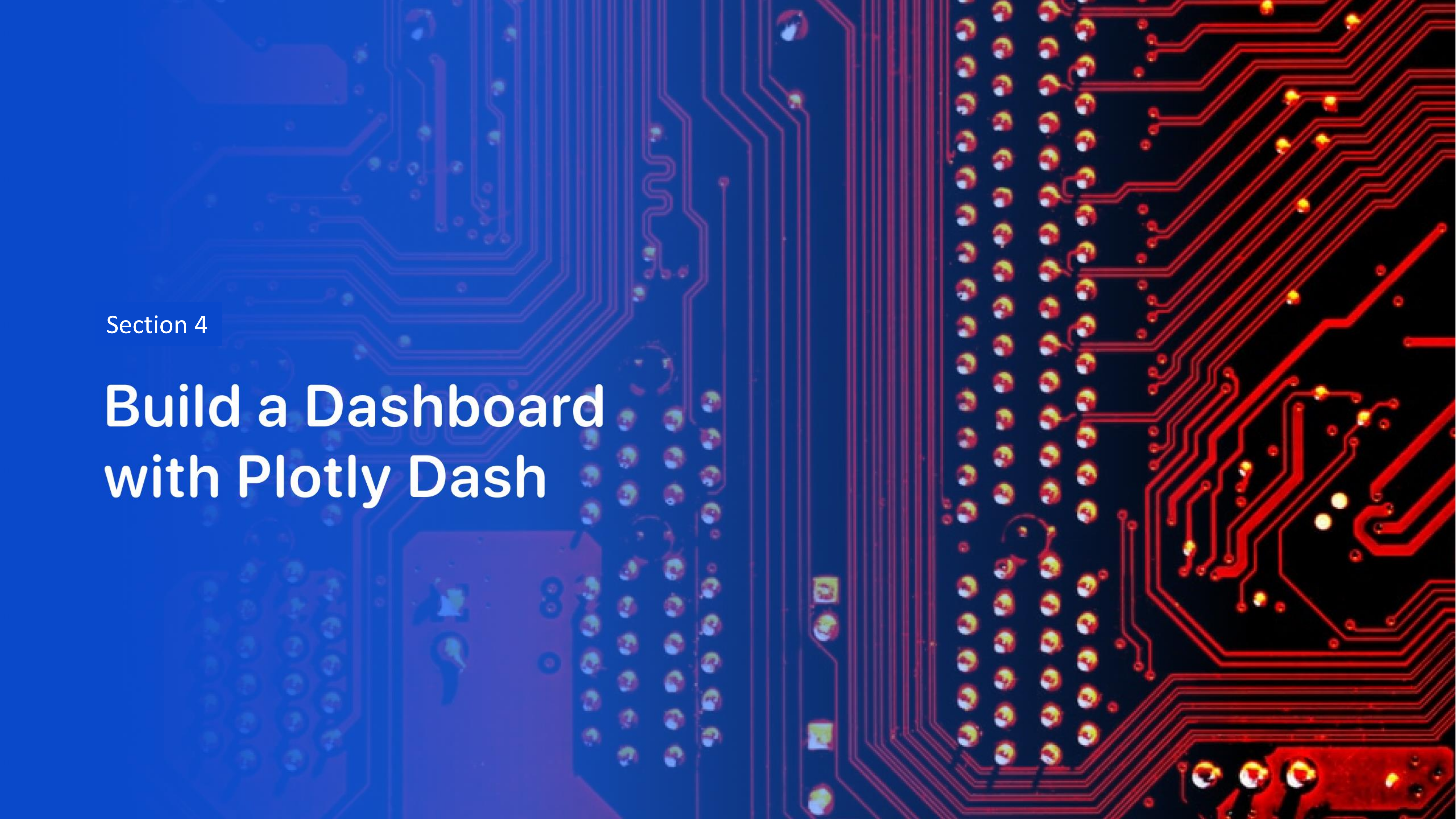


*Are launch sites in close proximity to railways? NO
*Are launch sites in close proximity to highways? NO
*Are launch sites in close proximity to coastline? YES
*Do launch sites keep certain distance away from cities? YES

- Launch site KSC LC-39A has good logistics aspects, being near railroad and Highroad and relatively far from inhabited areas.

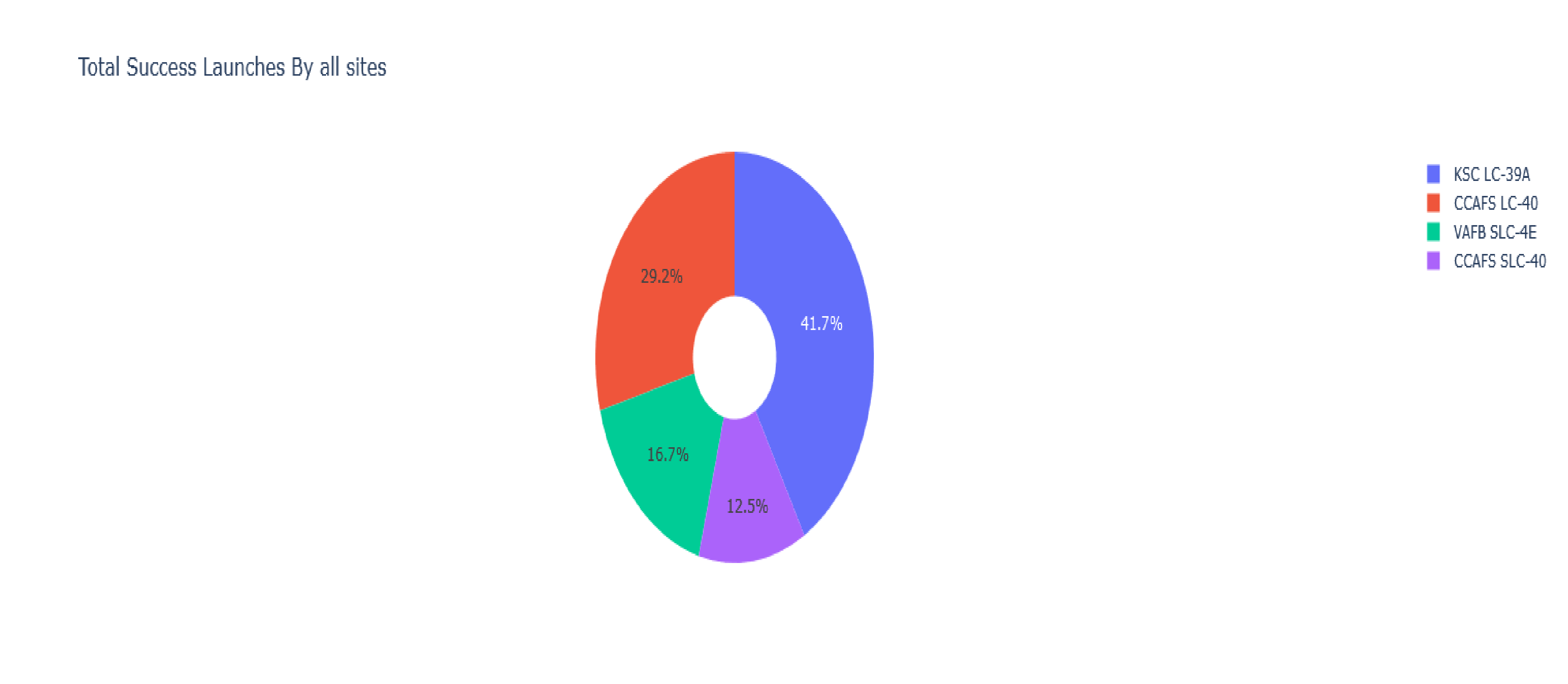
Section 4

Build a Dashboard with Plotly Dash



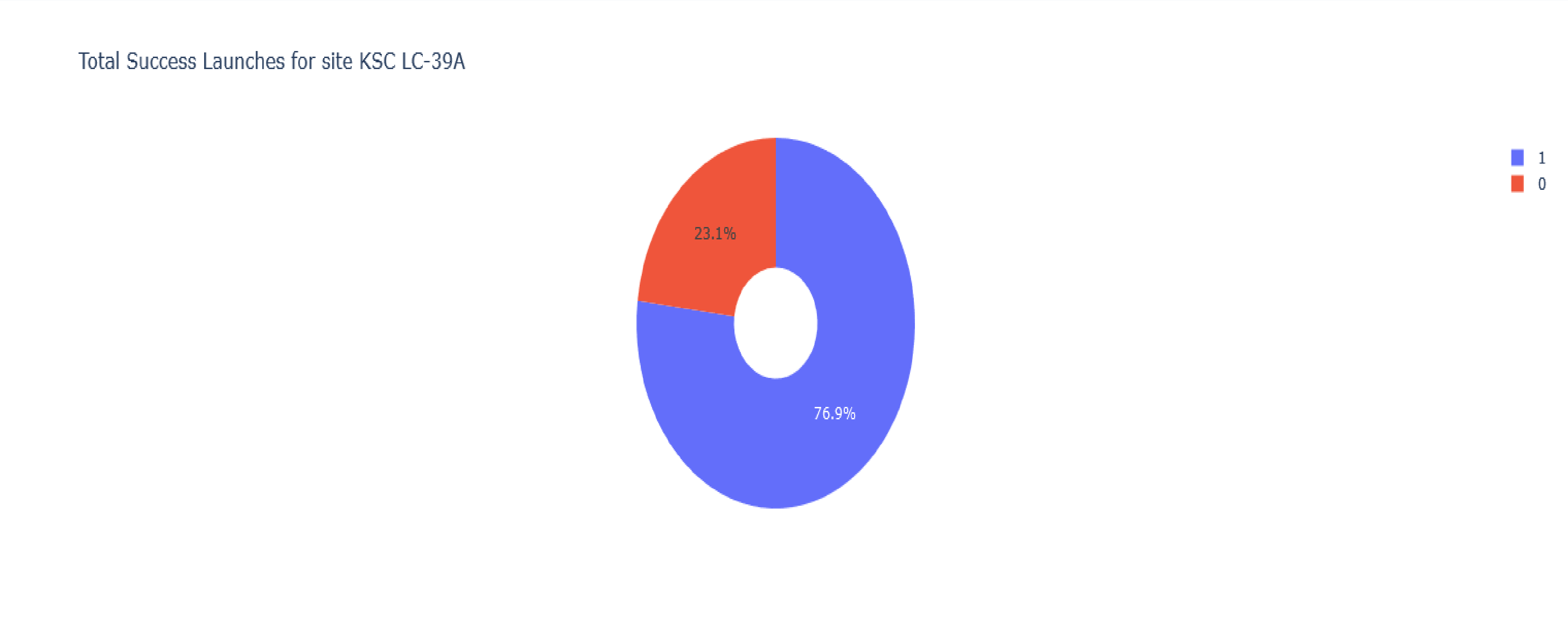
Successful Launches by Site

- The KSC LC-39A had the most successful launches from all the sites its take 41,7%.



The highest launch-success ratio: KSC LC-39A

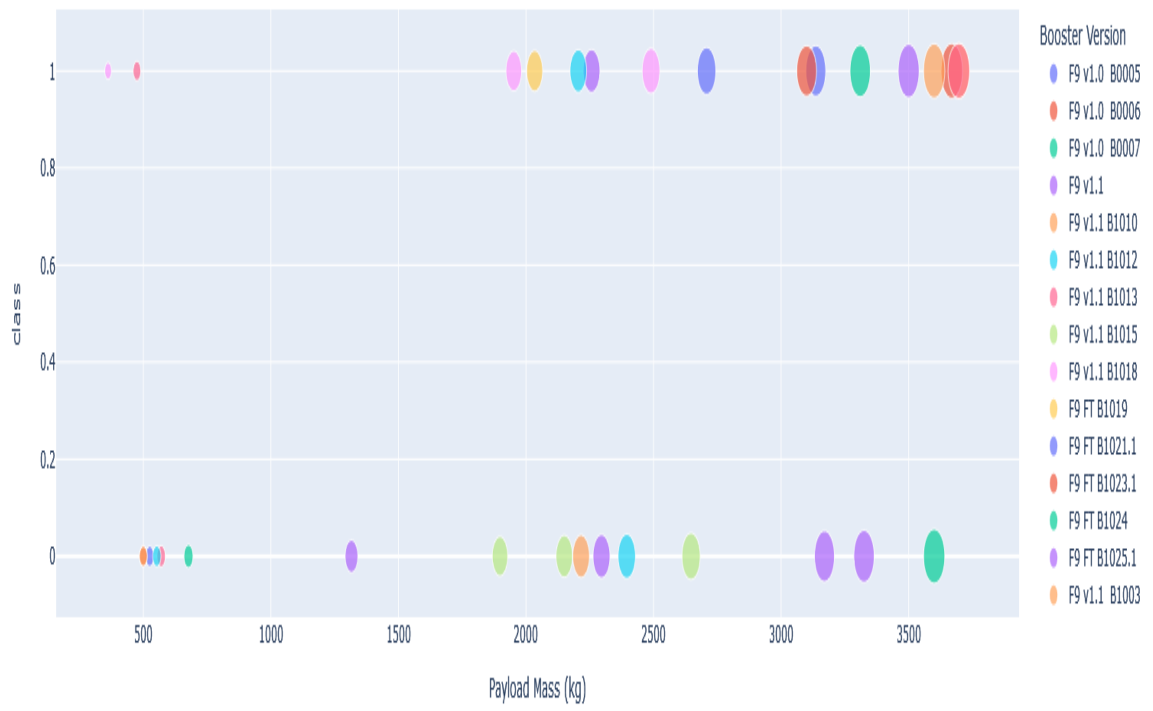
- KSC LC-39A achieved a 76,9% success rate while getting a 23,1% failure rate.



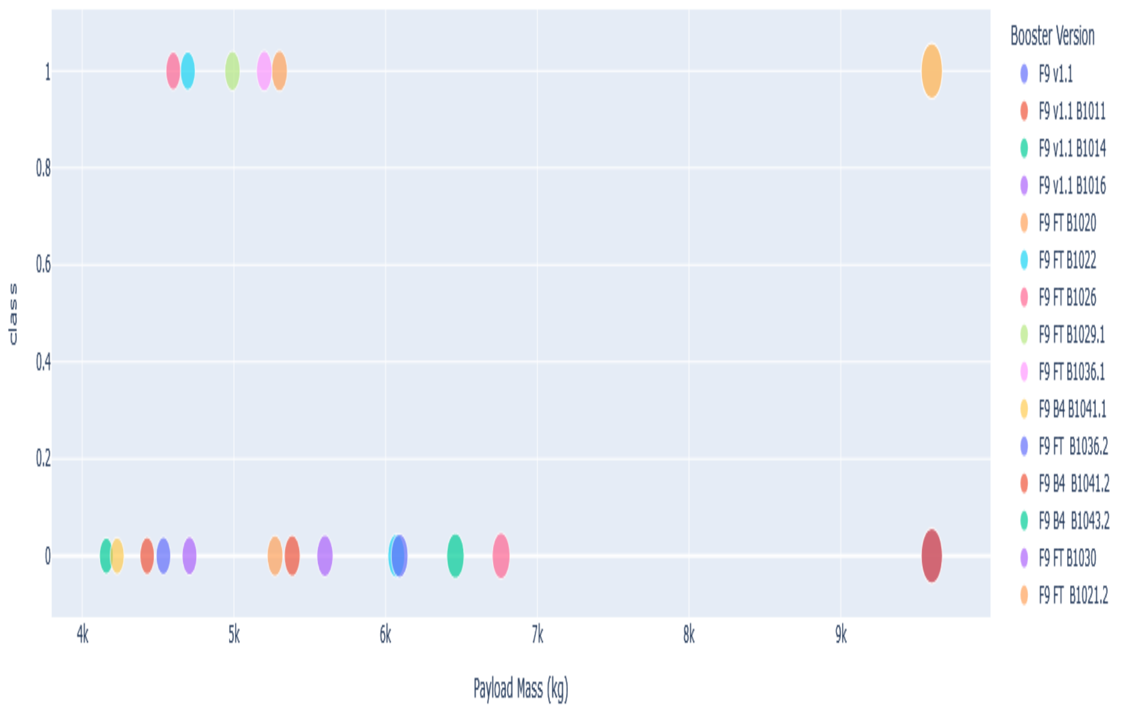
Payload vs Launch Outcome Scatter Plot

- We can see that all the success rate for low weight payload is higher than heavy weighted payload.

Low weight payload (0-4000)



heavy weight payload (4000-10000)

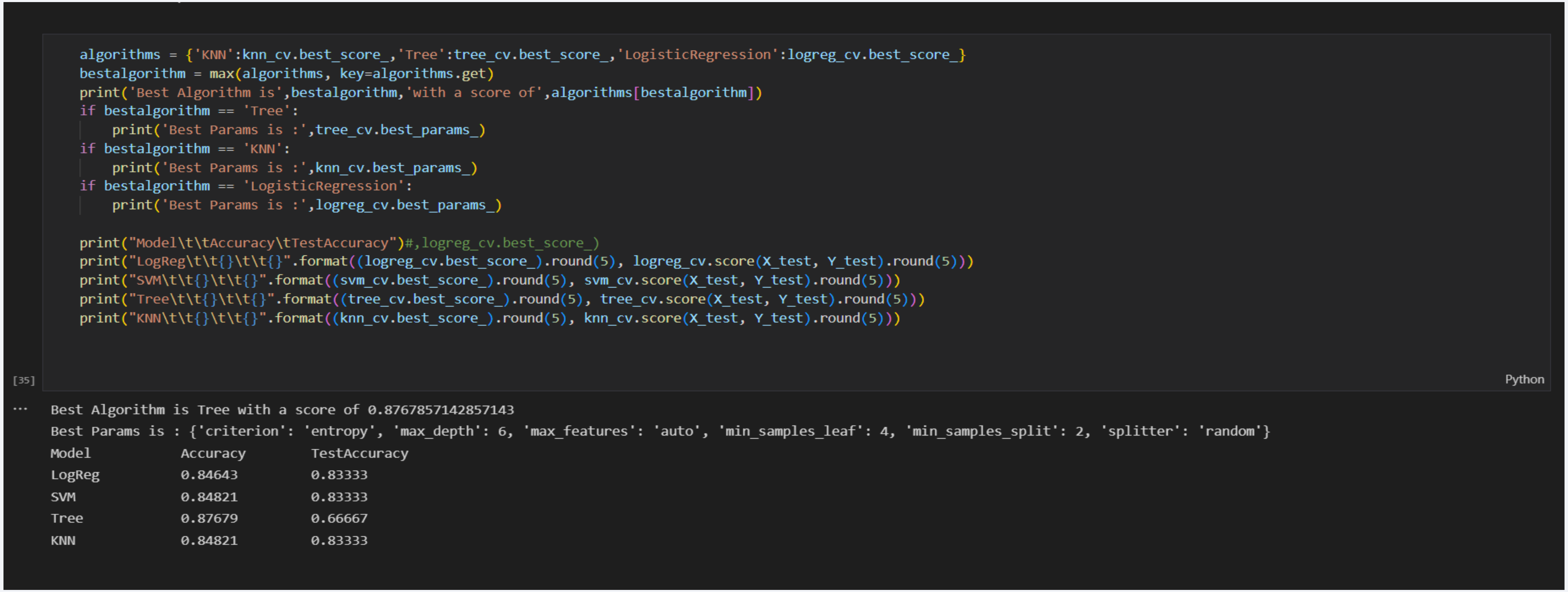


Section 5

Predictive Analysis (Classification)

Classification Accuracy

- As we can see, by using the code as below: we could identify that the best algorithm to be the Tree Algorithm which have the highest classification accuracy.

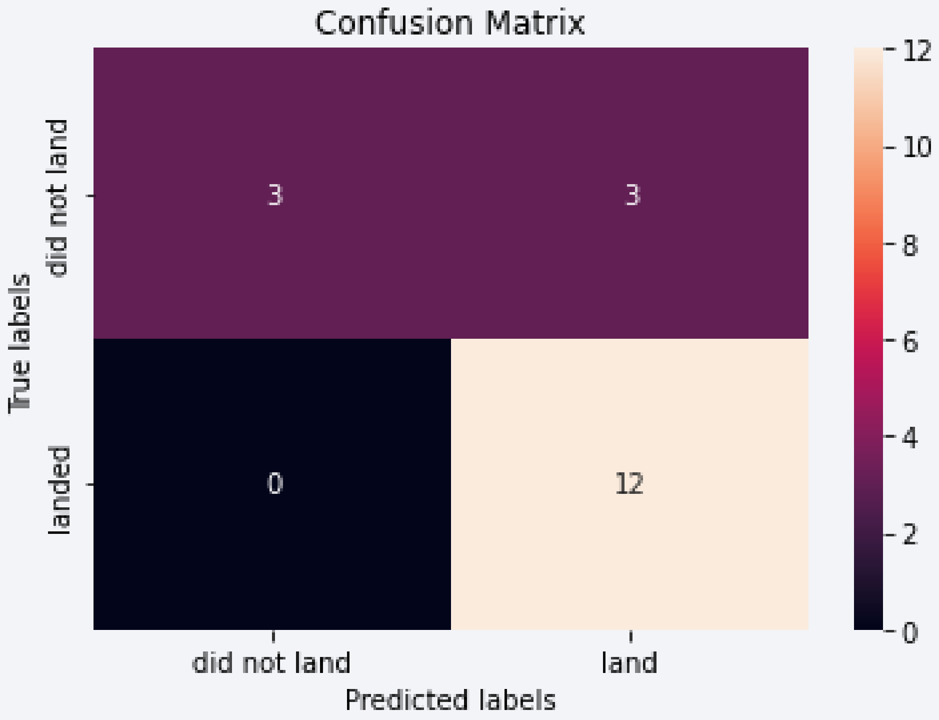


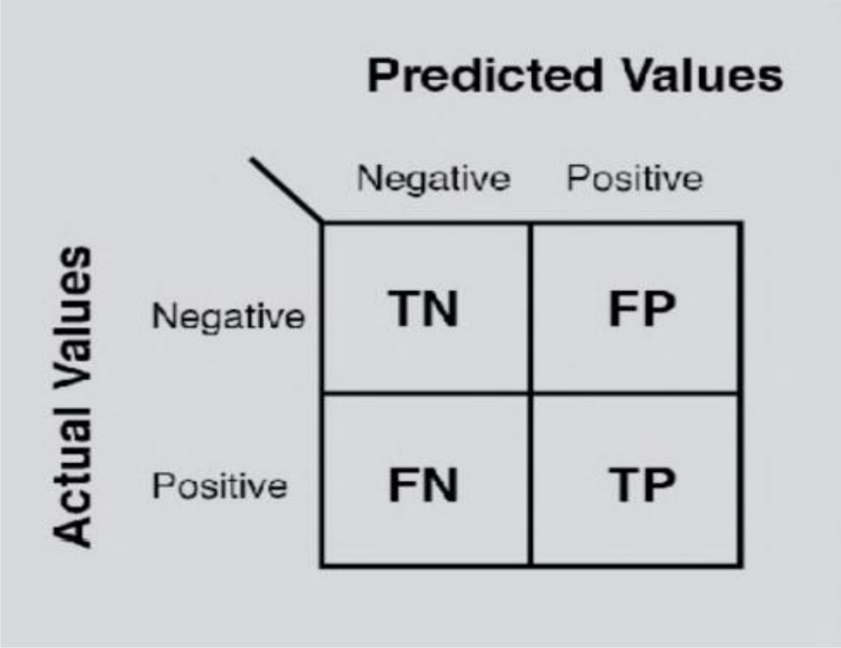
```python
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)

print("Model\t\tAccuracy\tTestAccuracy")#,logreg_cv.best_score_)
print("LogReg\t\t{}\t\t{}".format((logreg_cv.best_score_).round(5), logreg_cv.score(X_test, Y_test).round(5)))
print("SVM\t\t{}\t\t{}".format((svm_cv.best_score_).round(5), svm_cv.score(X_test, Y_test).round(5)))
print("Tree\t\t{}\t\t{}".format((tree_cv.best_score_).round(5), tree_cv.score(X_test, Y_test).round(5)))
print("KNN\t\t{}\t\t{}".format((knn_cv.best_score_).round(5), knn_cv.score(X_test, Y_test).round(5)))
```

```
Best Algorithm is Tree with a score of 0.8767857142857143
Best Params is : {'criterion': 'entropy', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 2, 'splitter': 'random'}
Model           Accuracy        TestAccuracy
LogReg          0.84643         0.83333
SVM             0.84821         0.83333
Tree            0.87679         0.66667
KNN             0.84821         0.83333
```

Confusion Matrix

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





Predicted Values

Actual Values

		Predicted Values	
		Negative	Positive
Actual Values	Negative	TN	FP
	Positive	FN	TP

Conclusions

- We can conclude that:
  - The Tree Classifier Algorithm is the best Machine Learning approach for this dataset.
  - The low weighted payloads (which define as 4000kg and below) performed better than the heavy weighted payloads.
  - Starting from the year 2013, the success rate for SpaceX launches is increased, directly proportional time in years to 2020, which it will eventually perfect the launches in the future.
  - KSC LC-39A have the most successful launches of any sites; 76.9%
  - SSO orbit have the most success rate; 100% and more than 1 occurrence.

Appendix

- Folium didn’t show maps on Github, so I took screenshots.

Thank you!

Ali Mohammed A Asiri
26-08-2022