IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Santiago Bonilla Cárdenas
May 12th of 2023

Outline

Executive Summary

- Summary of methodologies
- This project uses a wide array of techniques to provide valuable insights on the variables that determine the success of Stage-1 landing of SpaceX Falcon 9 rocket launches. With a variety of Python libraries and methods we collect, wrangle, prepare and visualize Falcon 9 launch data from 2010 to 2020, which allows us to then build several classification models to predict whether Stage-1 of a rocket mission will successfully land, thus significantly reducing launch costs.
- Summary of findings
- We find that, as the number of flights grows at a launch site, the success rate of said launch site will increase; at the same time, the success rate of all launch sites increases from 2013 onward. On the other hand, KSC LC-39A is the most successful launch site from all sites in the database, and the highest success rate orbits for launching rockets are ES-L1, GEO, HEO, SSO, and VLEO.

Introduction

- Background:
- According to SpaceX official information, Falcon 9 rocket launches cost about 62 million dollars, which turns out to be significantly cheaper than launch costs of its competitors, which are above 165 million dollars each. The reason is that the first stage of Falcon 9 is reusable, provided that it lands successfully. Hence, one of the most important pieces of information for SpaceX competitors to estimate launch costs is whether a rocket mission will successfully recuperate its first stage or not.
- Problems to find answers to:
- ¿What are the main factors affecting the landing outcome of the rocket?
- ¿Which predictive models can better help us to estimate the landing success of a rocket?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:

The data we use for this project was obtained by running a series of requests to a SpaceX API, and by performing a web scraping process on the Wikipedia page for SpaceX.

- Perform data wrangling

Using many Python libraries and methods, we create the dummy feature “Class”, which has a value of 1 if the landing was successful, and 0 otherwise. Similarly, we use one-hot encoding to obtain many binary variables representing various 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

We use several *scikit learn* libraries for classification models, which we train with a subset of our data, and use test data to obtain their predictions. We optimize their parameters using *GridSearchCV.*

Data Collection

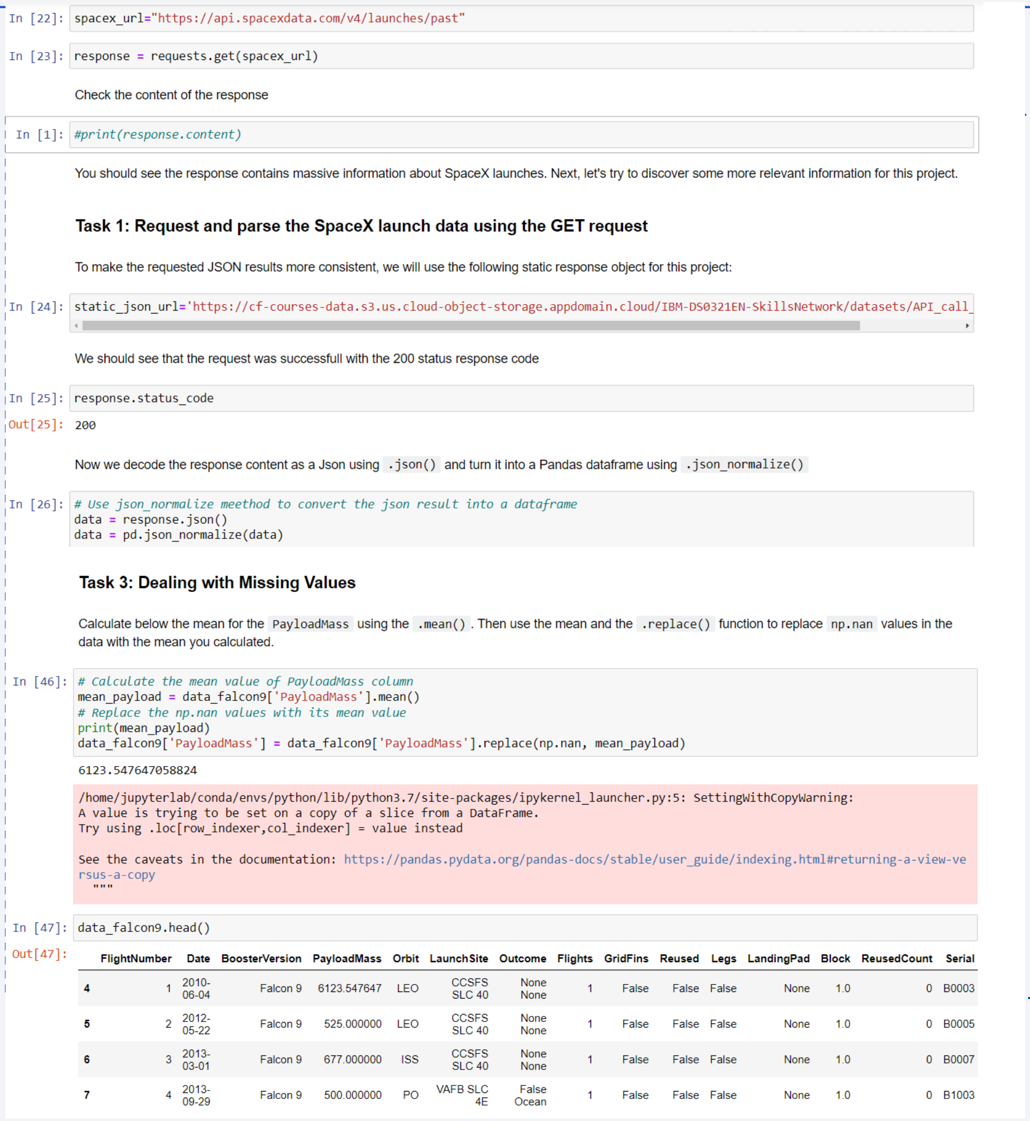
- We collect our data using several methods.
- First, we use the SpaceX API, to which we send a request and get its response.
- Then, we decode the response content as a Json file and turn it into a Pandas dataframe.
- After this, we perform data cleaning and wrangling, checking for missing values and filling them with the mean, in the specific case of the “Payload mass” feature.
- We also use web scraping on the Wikipedia page for Falcon 9 launch records, using the Beautiful Soup library, thus extracting the launch records as an HTML table, to then parse the table and turn it to a pandas dataframe.

Data Collection – SpaceX API

- Here we can see part of the code we use to run the request to the SpaceX API, to then convert the Json response to a Pandas dataframe, as well as dealing with missing values.

- Link to the GitHub repository for the full Jupyter notebook:

- https://github.com/sbch86/my_ibm_captsone_project/blob/main/COMPLETED_jupyter-labs-spacex-data-collection-api.ipynb



```
In [22]: spacex_url="https://api.spacexdata.com/v4/launches/past"

In [23]: response = requests.get(spacex_url)
```

Check the content of the response

```
In [1]: #print(response.content)
```

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

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

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

```
In [24]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_
```

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

```
In [25]: response.status_code
Out[25]: 200
```

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

```
In [26]: # Use json_normalize meethod to convert the json result into a dataframe
         data = response.json()
         data = pd.json_normalize(data)
```

Task 3: Dealing with Missing Values

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace `np.nan` values in the data with the mean you calculated.

```
In [46]: # Calculate the mean value of PayloadMass column
         mean_payload = data_falcon9['PayloadMass'].mean()
         # Replace the np.nan values with its mean value
         print(mean_payload)
         data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, mean_payload)

         6123.547647058824

         /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/ipykernel_launcher.py:5: SettingWithCopyWarning:
         A value is trying to be set on a copy of a slice from a DataFrame.
         Try using .loc[row_indexer,col_indexer] = value instead

         See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
         rsus-a-copy
           """

In [47]: data_falcon9.head()
Out[47]:
```

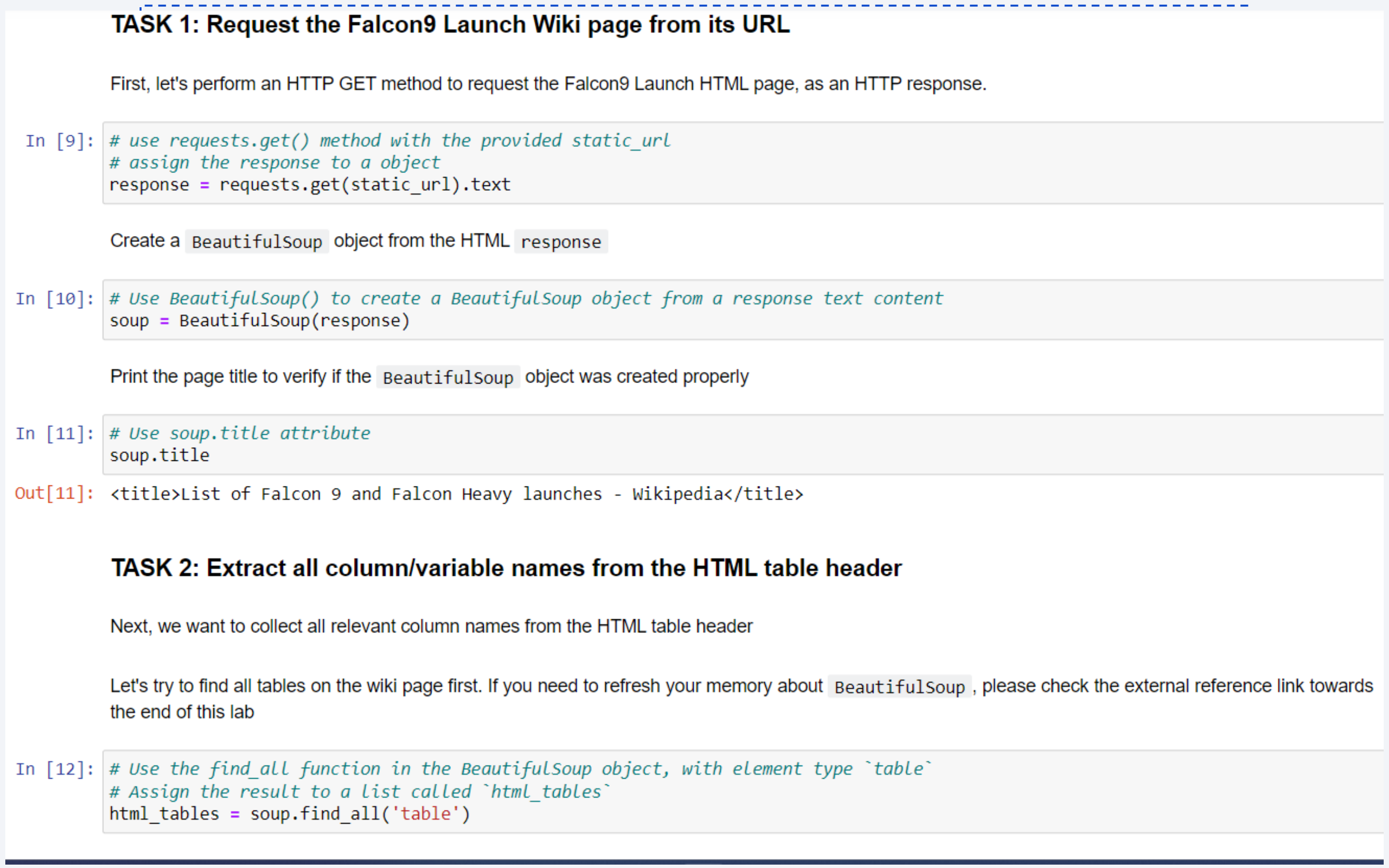
	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
4	1	2010-06-04	Falcon 9	6123.547647	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003
5	2	2012-05-22	Falcon 9	525.000000	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0005
6	3	2013-03-01	Falcon 9	677.000000	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0007
7	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None	1.0	0	B1003

Data Collection - Scraping

- We use Beautiful Soup to web scrape data from the Wike page of SpaceX.

- Link to the GitHub repository for the full Jupyter notebook:

- https://github.com/sbch86/my_ibm_captsone_project/blob/main/COMPLETED_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 HTTP response.

```
In [9]: # 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`

```
In [10]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
         soup = BeautifulSoup(response)
```

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

```
In [11]: # Use soup.title attribute
         soup.title

Out[11]: <title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

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

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

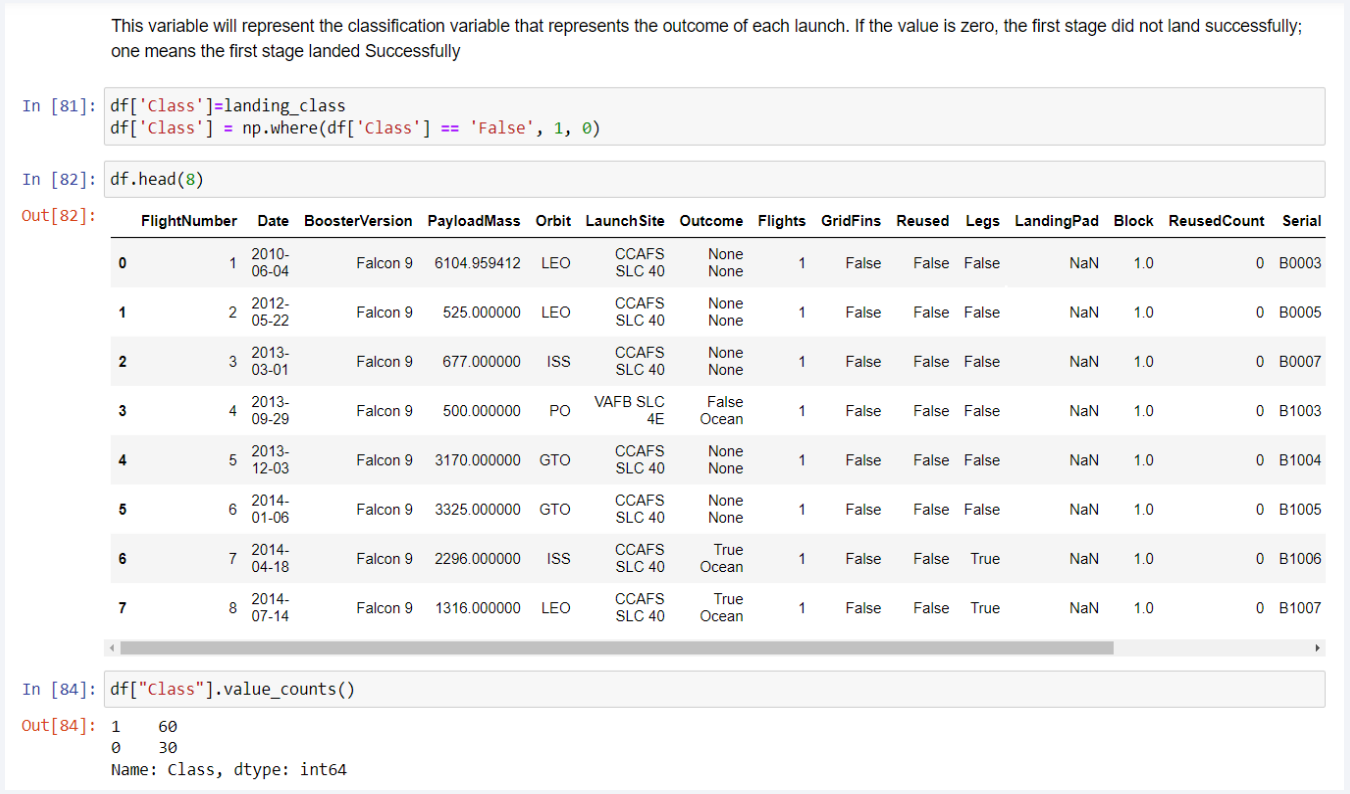
Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

```
In [12]: # Use the find_all function in the BeautifulSoup object, with element type `table`
         # Assign the result to a list called `html_tables`
         html_tables = soup.find_all('table')
```

Data Wrangling

- Describe how data were processed
- In this part we use the categorical feature “landing_outcomes” in order to create the new variable “Class”, corresponding to 1 if the landing outcome was successful, and zero otherwise.

- Link to the GitHub repository for the full Jupyter notebook:
- https://github.com/sbch86/my_ibm_captsone_project/blob/main/COMPLETED_labs-jupyter-spacex-data_wrangling_jupyterlite.jupyterlite.ipynb



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

```
In [81]: df['Class']=landing_class
         df['Class'] = np.where(df['Class'] == 'False', 1, 0)
```

```
In [82]: df.head(8)
```

Out[82]:

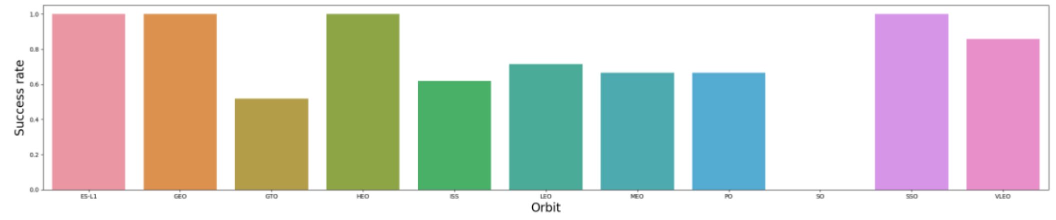
	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004
5	6	2014-01-06	Falcon 9	3325.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1005
6	7	2014-04-18	Falcon 9	2296.000000	ISS	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1006
7	8	2014-07-14	Falcon 9	1316.000000	LEO	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1007

```
In [84]: df["Class"].value_counts()

Out[84]: 1    60
         0    30
         Name: Class, dtype: int64
```

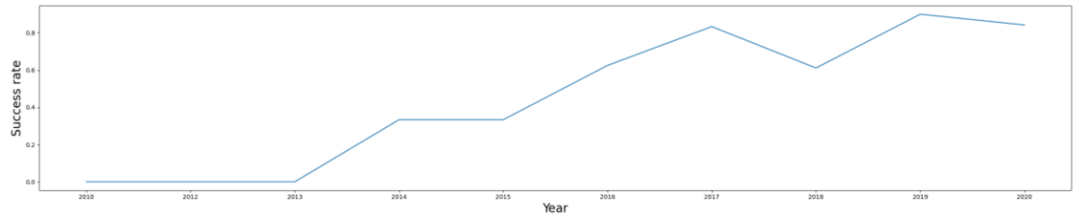
EDA with Data Visualization

- We explore the relationship among several launch features, such as Launchsite vs. Flight number, Launchsite vs. Payload mass, Orbit vs. Success rate, Orbit type vs. Flight number and Payload mass. Finally, we chart a line plot showing the Success rate trend throughout the years.
- Link to the GitHub repository for the full Jupyter notebook:
- https://github.com/sbch86/my_ibm_captsone_project/blob/main/COMPLETED_jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

```
In [20]: sns.barplot(data=success_rate, x='Orbit', y='Class')
         plt.xlabel("Orbit",fontsize=20)
         plt.ylabel("Success rate",fontsize=20)
         plt.show()
```



```
In [26]: # Plot a line chart with x axis to be the extracted year and y axis to be the success rate
         success_rate = df.groupby(['Date'])['Class'].mean().reset_index()
         sns.lineplot(data=success_rate, x='Date', y='Class')
         plt.xlabel("Year",fontsize=20)
         plt.ylabel("Success rate",fontsize=20)
         plt.show()
```



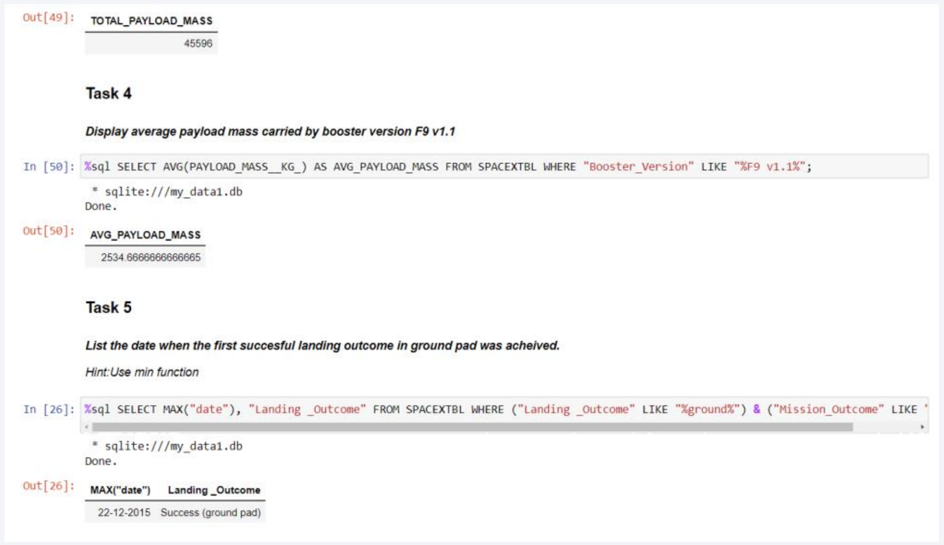
you can observe that the sucess rate since 2013 kept increasing till 2020

EDA with SQL

- In this part, we use SQL alchemy to perform various SQL queries from our Jupyter notebook, as follows:

- *Display the names of the unique launch sites in the space mission*
- *Display 5 records where launch sites begin with the string 'CCA'*
- *Display the total payload mass carried by boosters launched by NASA (CRS)*
- *Display average payload mass carried by booster version F9 v1.1*
- *List the date when the first successful landing outcome in ground pad was achieved.*
- *List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000*
- *List the total number of successful and failure mission outcomes*
- *List the names of the booster versions which have carried the maximum payload mass*
- *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*
- *Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.*

- https://github.com/sbch86/my_ibm_captsone_project/blob/main/COMPLETED_jupyter-labs-eda-sql-coursera_sqllite%20(1).ipynb



```
Out[49]:  TOTAL_PAYLOAD_MASS
                       45596
```

Task 4

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

```
In [50]: %sql SELECT AVG(PAYLOAD_MASS__KG_) AS AVG_PAYLOAD_MASS FROM SPACEXTBL WHERE "Booster_Version" LIKE "%F9 v1.1%";

          * sqlite:///my_data1.db
         Done.

Out[50]:  AVG_PAYLOAD_MASS
            2534.6666666666665
```

Task 5

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

Hint:Use min function

```
In [26]: %sql SELECT MAX("date"), "Landing _Outcome" FROM SPACEXTBL WHERE ("Landing _Outcome" LIKE "%ground%") & ("Mission_Outcome" LIKE '

          * sqlite:///my_data1.db
         Done.

Out[26]:  MAX("date")   Landing _Outcome
           22-12-2015   Success (ground pad)
```

Build an Interactive Map with Folium

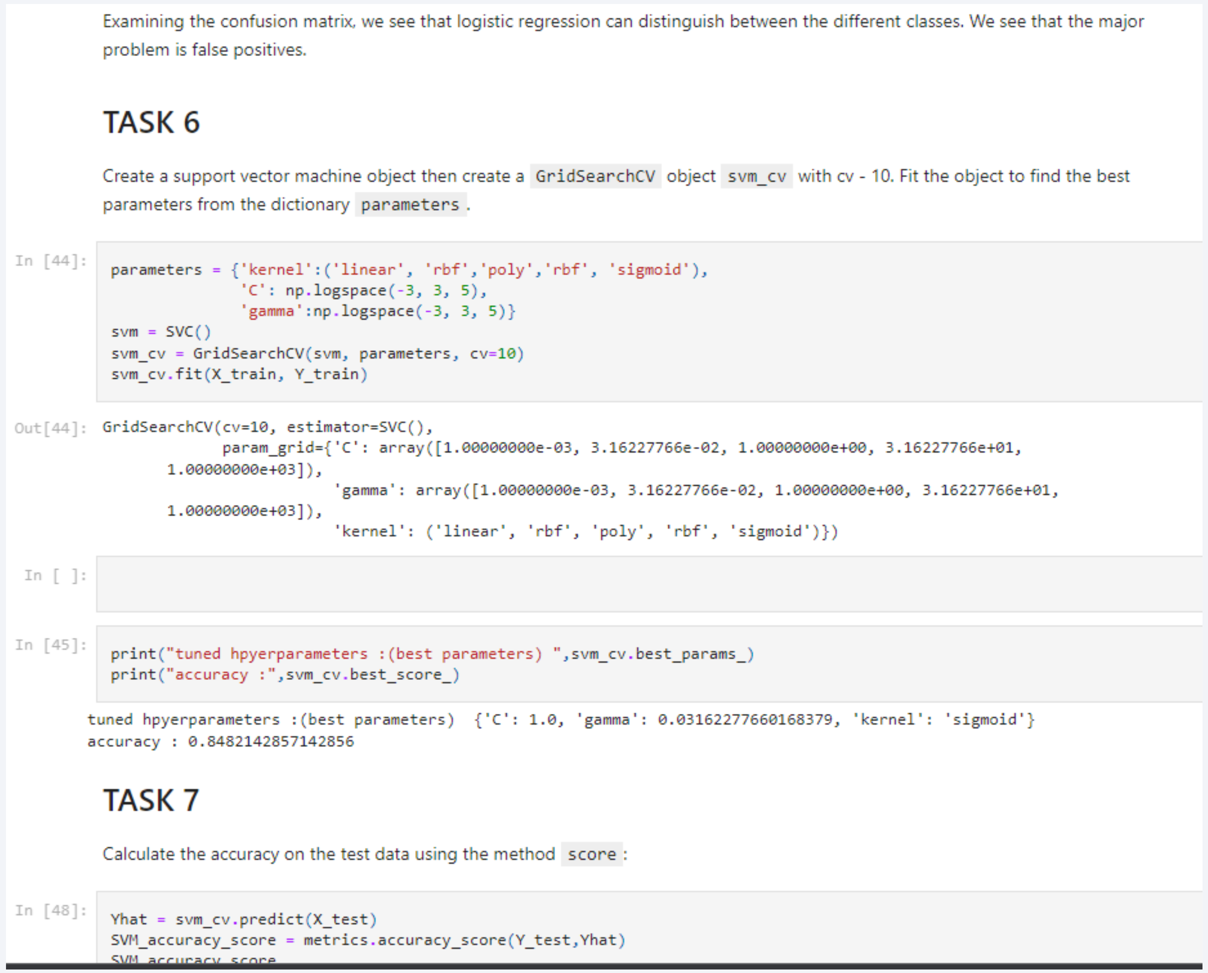
- In this part, we use Folium library to display a map with all launch sites and added objects like markers and circles to show the success or failure of launches for each site.
- Using marker clusters, we identify which launch sites have relatively high and low success rate.
- We calculate the distances between a launch site to any of its proximities, in order to answer questions like:
  - Are launch sites near railways, highways and coastlines.
  - Do launch sites keep certain distance away from cities.
- https://github.com/sbch86/my_ibm_captsone_project/blob/main/COMPLETED_lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- We build an interactive dashboard using the Plotly dash library.
- We plot pie charts showing total launches at specific sites.
- We build scatter plots showing the relationship between Landing Outcome and Payload Mass (Kg) for the different booster versions.

Predictive Analysis (Classification)

- In this part we use Numpy, Pandas and Scikit learn, to transform the data, split it into training and testing, and then build machine learning classification models, tuning them with different hyperparameters using GridSearchCV to determine which of these generate the most accurate results.
- We use the accuracy score as the metric for our model in order to find the best performing model.
- https://github.com/sbch86/my_ibm_capstone_project/blob/main/COMPLETED_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite%20(1).ipynb



Examining the confusion matrix, we see that logistic regression can distinguish between the different classes. We see that the major problem is false positives.

TASK 6

Create a support vector machine object then create a `GridSearchCV` object `svm_cv` with cv - 10. Fit the object to find the best parameters from the dictionary `parameters`.

```
In [44]:
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
svm_cv = GridSearchCV(svm, parameters, cv=10)
svm_cv.fit(X_train, Y_train)
```

```
Out[44]: GridSearchCV(cv=10, estimator=SVC(),
                      param_grid={'C': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.16227766e+01,
                1.00000000e+03]),
                                  'gamma': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.16227766e+01,
                1.00000000e+03]),
                                  'kernel': ('linear', 'rbf', 'poly', 'rbf', 'sigmoid')})
```

```
In [ ]:
```

```
In [45]:
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8482142857142856
```

TASK 7

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

```
In [48]:
Yhat = svm_cv.predict(X_test)
SVM_accuracy_score = metrics.accuracy_score(Y_test,Yhat)
SVM_accuracy_score
```

Results

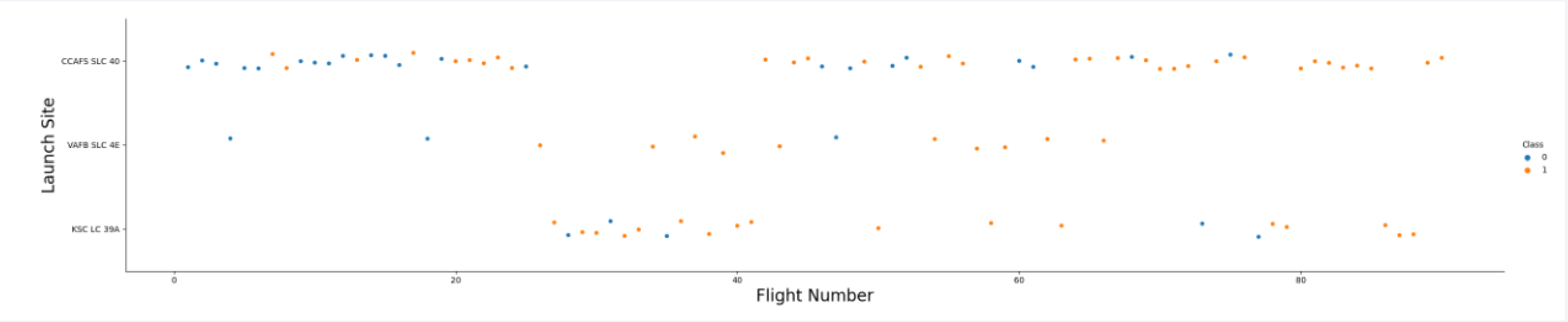
- We find that the highest success rate orbits for launching rockets are ES-L1, GEO, HEO, SSO, and VLEO. Also, success rates have been growing from 2013 to 2020.
- With respect to our prediction models, we find that Logistic regression, Support vector machine, Decision trees and K-nearest neighbors perform equally well on the test data subset in terms of accuracy score, once their hyperparameters have been optimized.

Section 2

Insights drawn from EDA

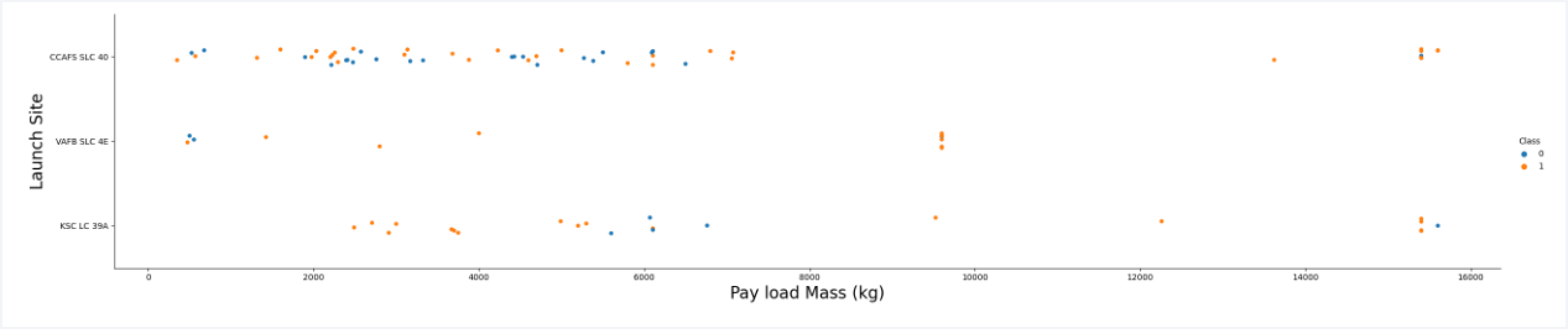
Flight Number vs. Launch Site

- From the below figure, we can conclude that the higher the number of flights in a launch site, the better it will perform in terms of successful landings.



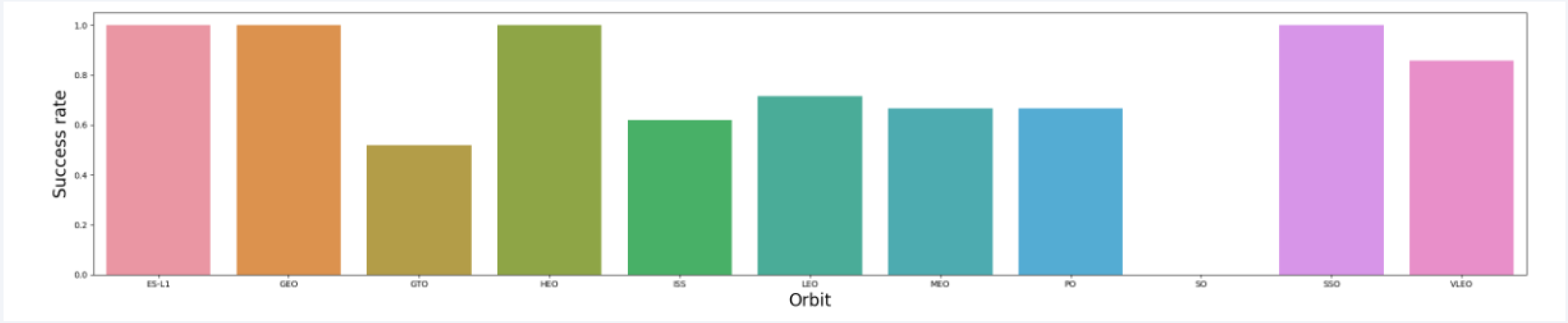
Payload vs. Launch Site

- According to the below figure, launch site CCAFS SLC 40 has not been particularly successful with lighter rockets (mass under 8000 kgs). However, for heavier payloads, its success rate increases.



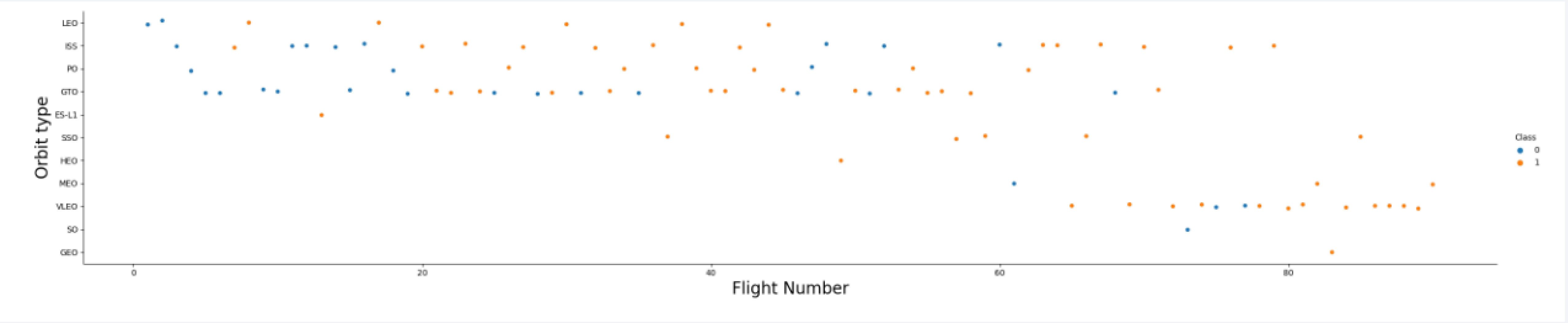
Success Rate vs. Orbit Type

- From the figure below, we can determine that the most successful orbits are ES-L1, GEO, HEO, SSO, VLEO.



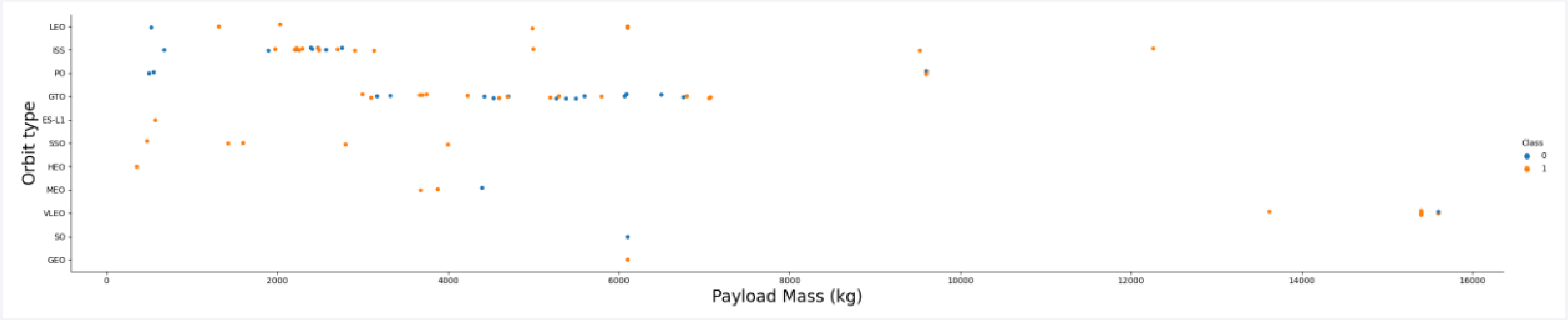
Flight Number vs. Orbit Type

- We can conclude that, in the case of LEO orbit, success is related to a higher number of flights. However, in the GTO orbit there appears to be no relationship between flight number and the orbit, as even when flight number rises, we still find unsuccessful landings.



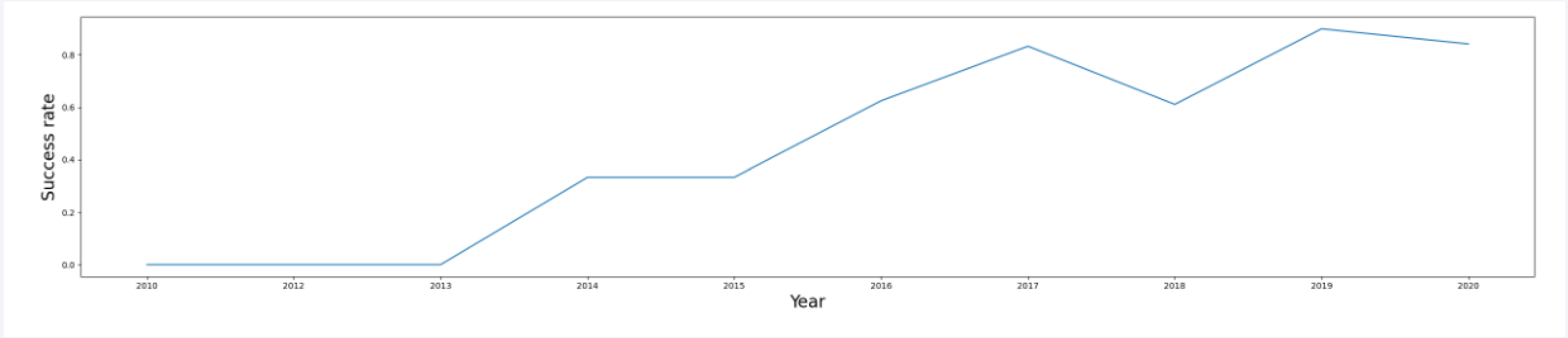
Payload vs. Orbit Type

- For lighter payload masses, SSO orbit is the most successful. However, for heavier masses, VLEO and ISS tend to be more successful in terms of landing outcome.



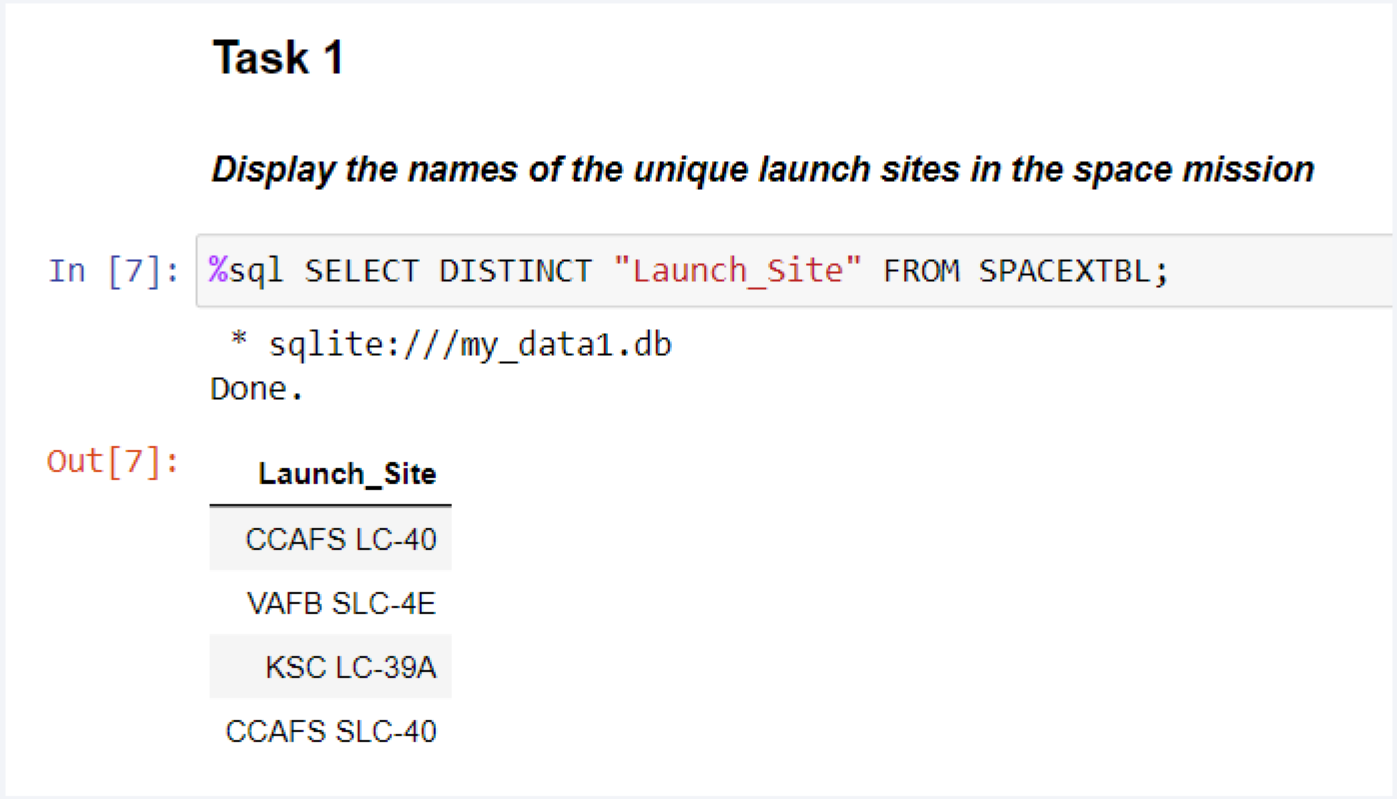
Launch Success Yearly Trend

- The figure below shows that average success rate has been steadily growing from 2013 to 2020.



All Launch Site Names

- The only launch sites in our database are the 4 shown below. We use the DISTINCT command.



Task 1

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

```
In [7]: %sql SELECT DISTINCT "Launch_Site" FROM SPACEXTBL;
```

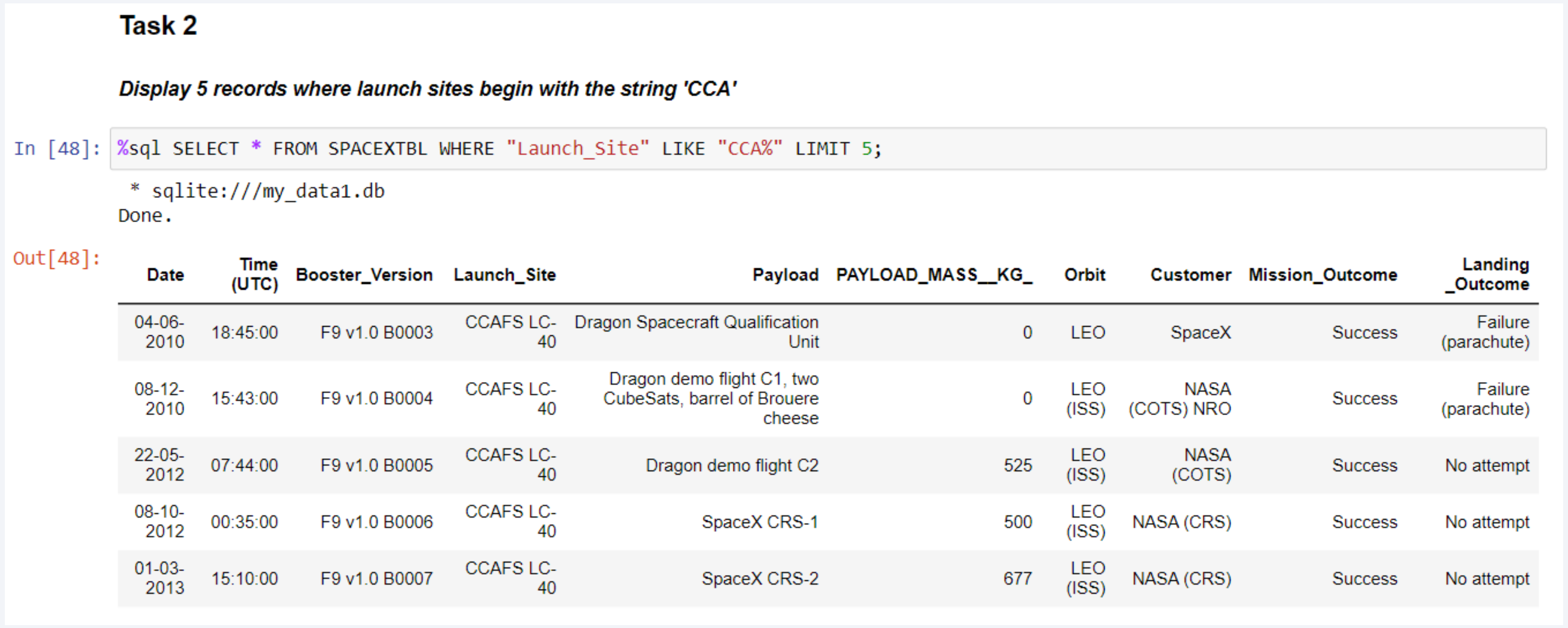
```
 * sqlite:///my_data1.db
Done.
```

Out[7]:

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

Launch Site Names Begin with 'CCA'

- To find those specific records, we use the LIKE command together with “CCA%” key, using LIMIT 5 too.



Task 2

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

```
In [48]: %sql SELECT * FROM SPACEXTBL WHERE "Launch_Site" LIKE "CCA%" LIMIT 5;
```

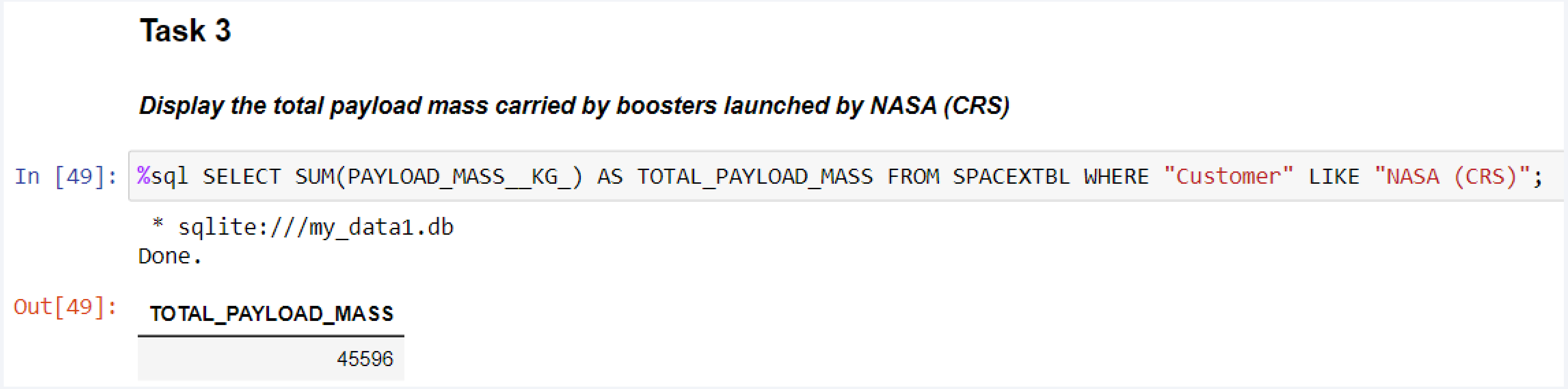
```
 * sqlite:///my_data1.db
Done.
```

Out[48]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- We use the SUM function, filtering for the NASA customer.



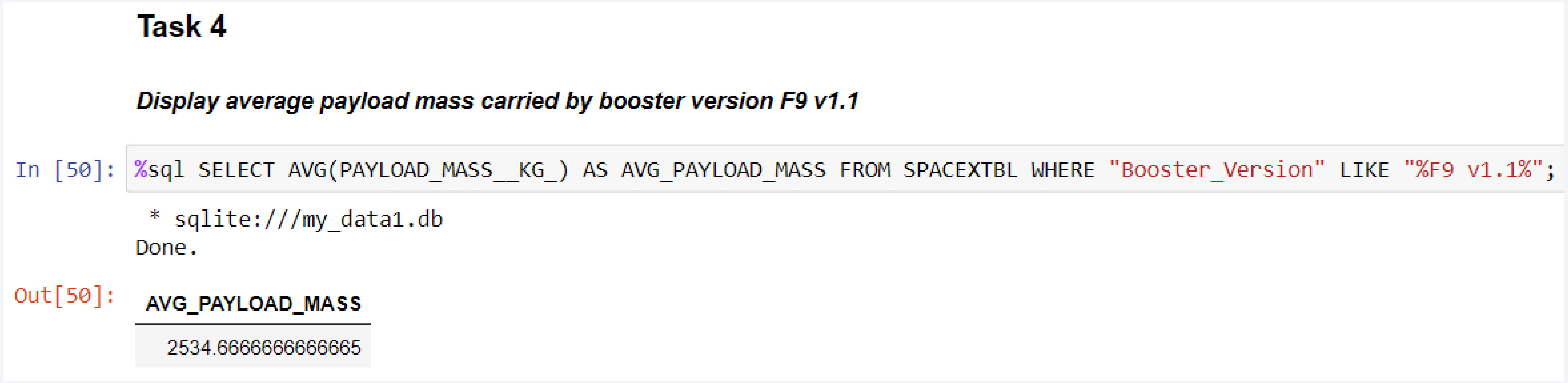
Task 3

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

```
In [49]: %sql SELECT SUM(PAYLOAD_MASS__KG_) AS TOTAL_PAYLOAD_MASS FROM SPACEXTBL WHERE "Customer" LIKE "NASA (CRS)";

 * sqlite:///my_data1.db
Done.

Out[49]:
TOTAL_PAYLOAD_MASS
             45596
```

Average Payload Mass by F9 v1.1

- In this case, we filter for the required booster version, and use AVG function.



Task 4

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

```
In [50]: %sql SELECT AVG(PAYLOAD_MASS__KG_) AS AVG_PAYLOAD_MASS FROM SPACEXTBL WHERE "Booster_Version" LIKE "%F9 v1.1%";
```

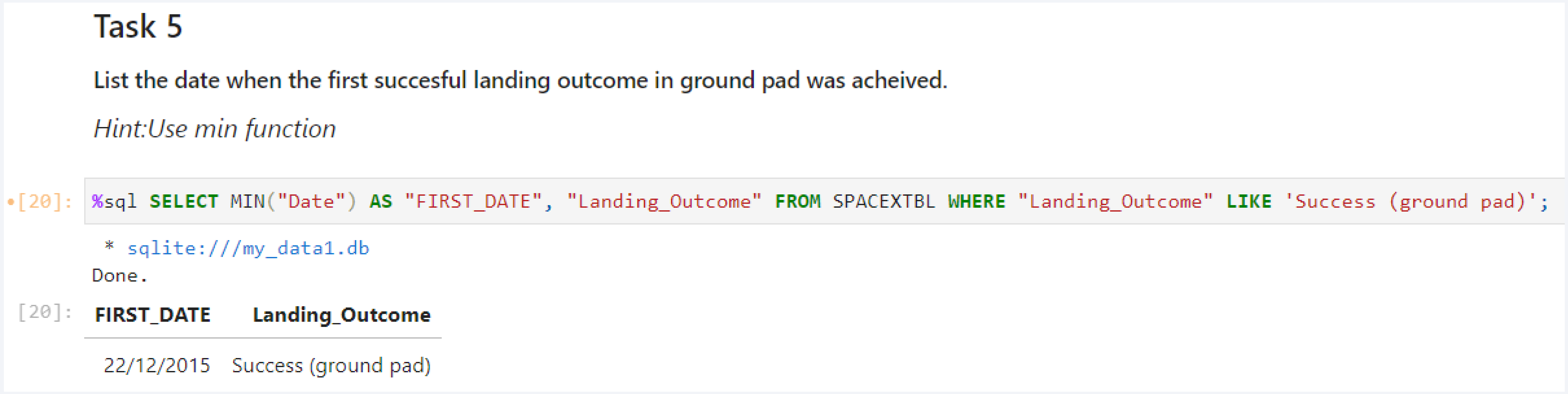
```
 * sqlite:///my_data1.db
Done.
```

Out[50]:

AVG_PAYLOAD_MASS
2534.6666666666665

First Successful Ground Landing Date

- The date of first successful landing on ground pad was 22 of December 2015.



Task 5

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

Hint:Use min function

```
[20]: %sql SELECT MIN("Date") AS "FIRST_DATE", "Landing_Outcome" FROM SPACEXTBL WHERE "Landing_Outcome" LIKE 'Success (ground pad)';
```

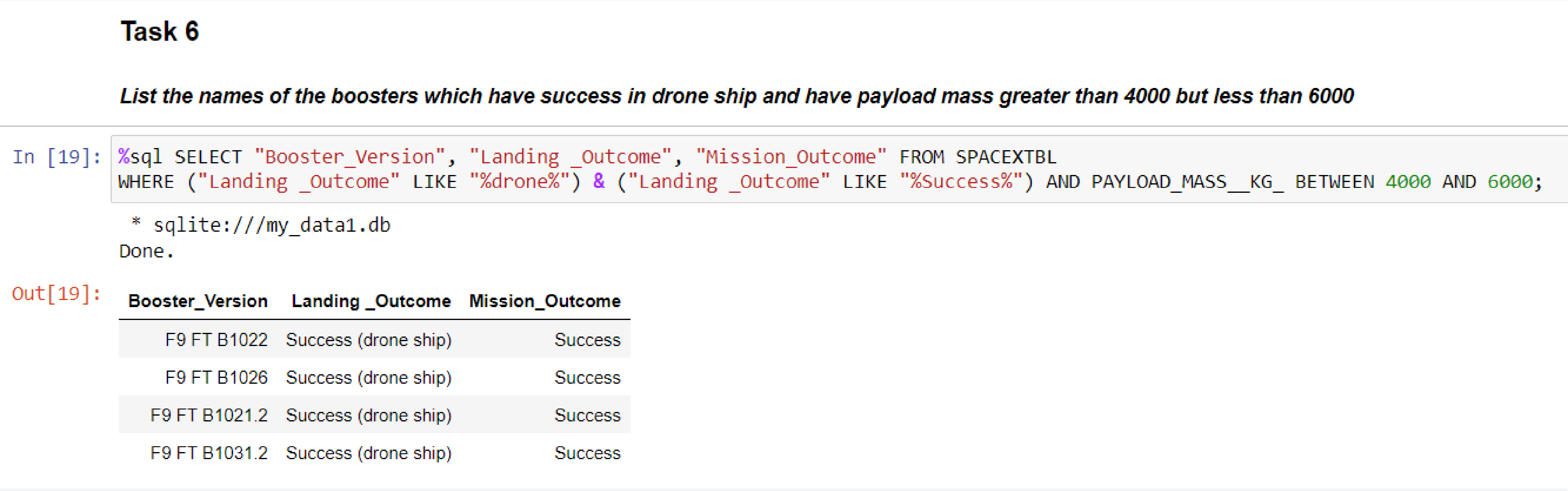
```
 * sqlite:///my_data1.db
Done.
```

[20]:

FIRST_DATE	Landing_Outcome
22/12/2015	Success (ground pad)

Successful Drone Ship Landing with Payload between 4000 and 6000

- These are the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000



Task 6

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

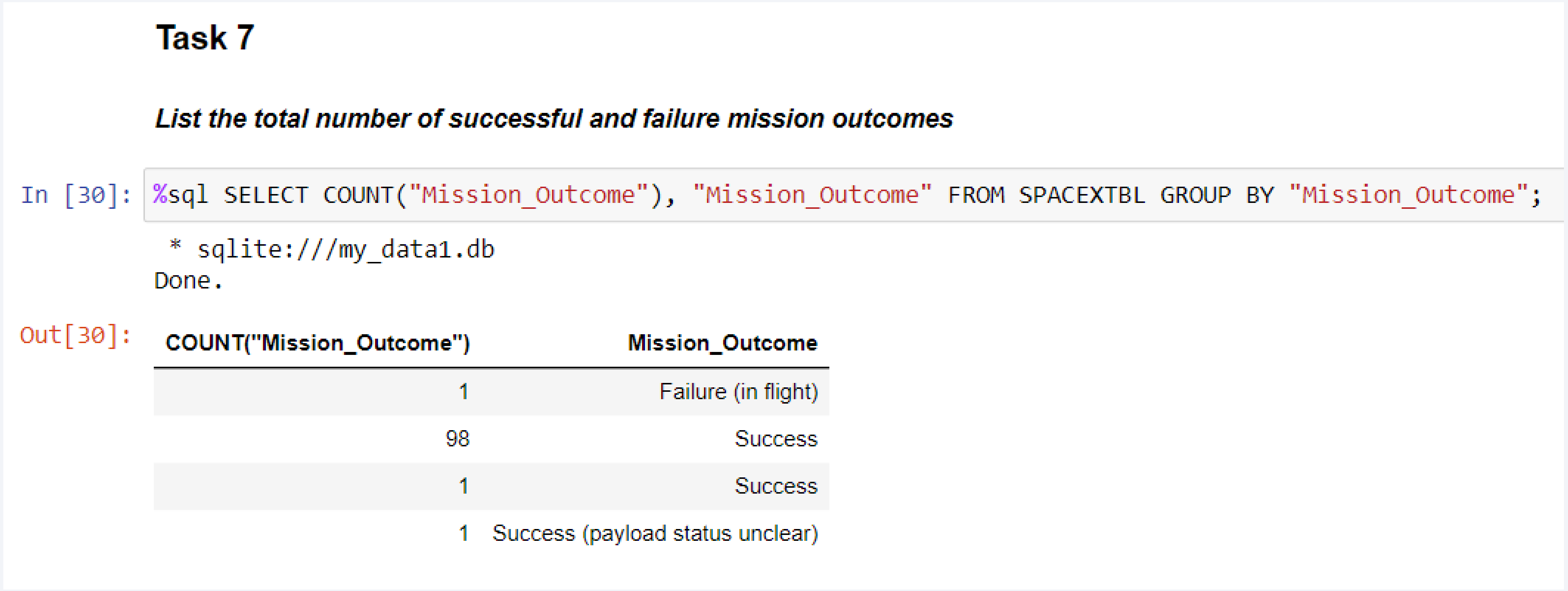
```
In [19]: %sql SELECT "Booster_Version", "Landing _Outcome", "Mission_Outcome" FROM SPACEXTBL
         WHERE ("Landing _Outcome" LIKE "%drone%") & ("Landing _Outcome" LIKE "%Success%") AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000;

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

Out[19]:

Booster_Version	Landing _Outcome	Mission_Outcome
F9 FT B1022	Success (drone ship)	Success
F9 FT B1026	Success (drone ship)	Success
F9 FT B1021.2	Success (drone ship)	Success
F9 FT B1031.2	Success (drone ship)	Success

Total Number of Successful and Failure Mission Outcomes

- Here we see the total number of successful and failure mission outcomes: 100 total successes, and 1 failure in flight.



Task 7

List the total number of successful and failure mission outcomes

```
In [30]: %sql SELECT COUNT("Mission_Outcome"), "Mission_Outcome" FROM SPACEXTBL GROUP BY "Mission_Outcome";
```

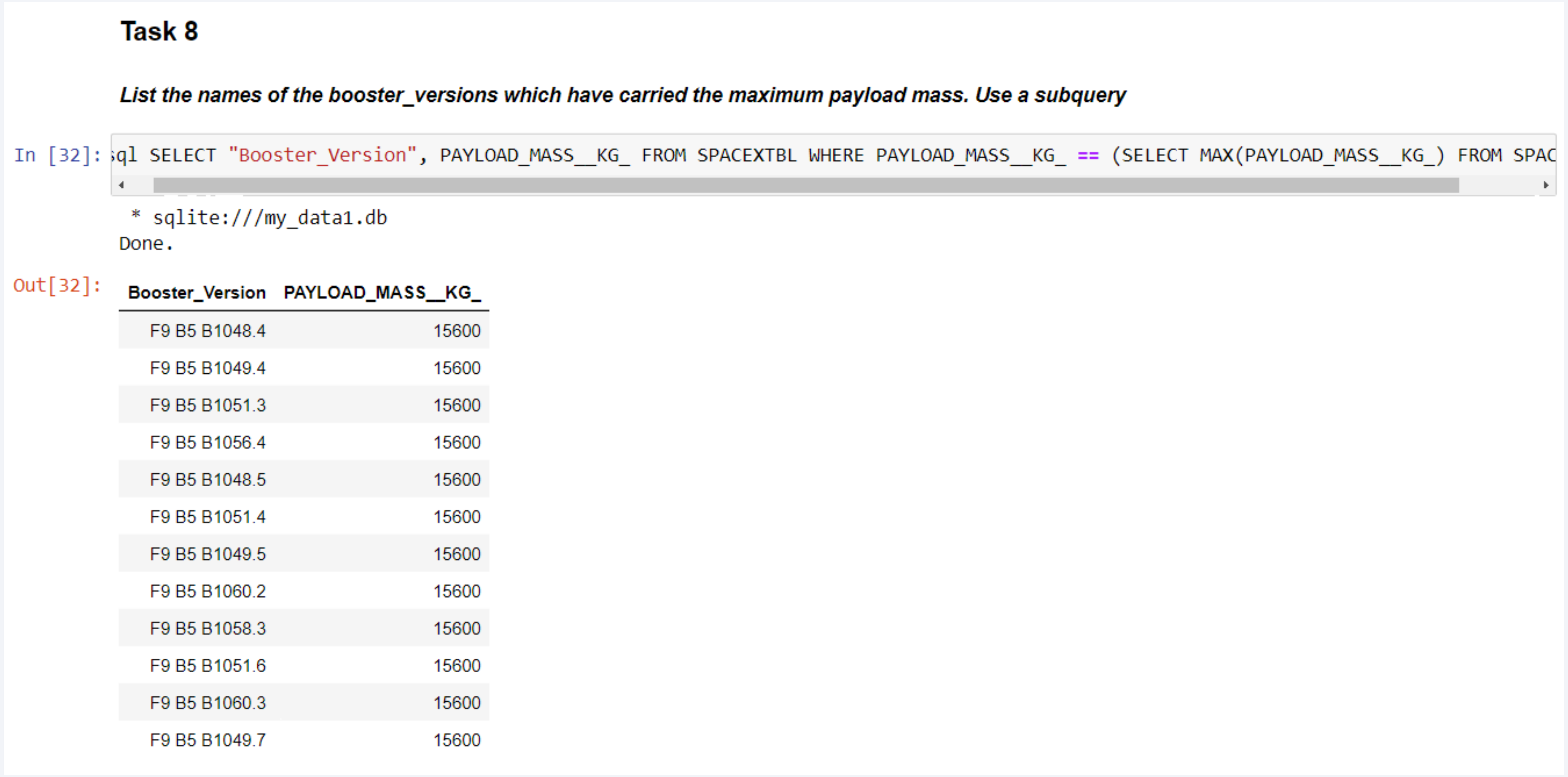
```
 * sqlite:///my_data1.db
Done.
```

Out[30]:

COUNT("Mission_Outcome")	Mission_Outcome
1	Failure (in flight)
98	Success
1	Success
1	Success (payload status unclear)

Boosters Carried Maximum Payload

- These are the boosters which have carried the maximum payload mass.



Task 8

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

```
In [32]: sql SELECT "Booster_Version", PAYLOAD_MASS__KG_ FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ == (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPAC
```

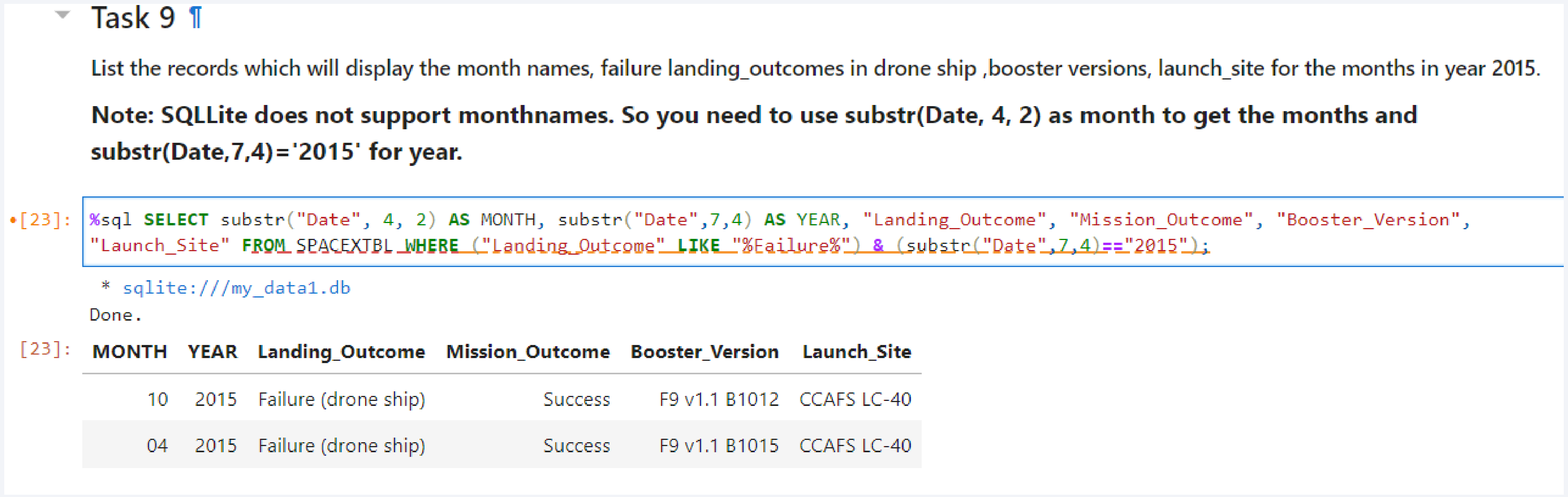
```
 * sqlite:///my_data1.db
Done.
```

Out[32]:

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

2015 Launch Records

- This is the list of failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015



Task 9 ¶

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

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

```
•[23]: %sql SELECT substr("Date", 4, 2) AS MONTH, substr("Date",7,4) AS YEAR, "Landing_Outcome", "Mission_Outcome", "Booster_Version",
"Launch_Site" FROM SPACEXTBL WHERE ("Landing_Outcome" LIKE "%Failure%") & (substr("Date",7,4)=="2015");
```

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

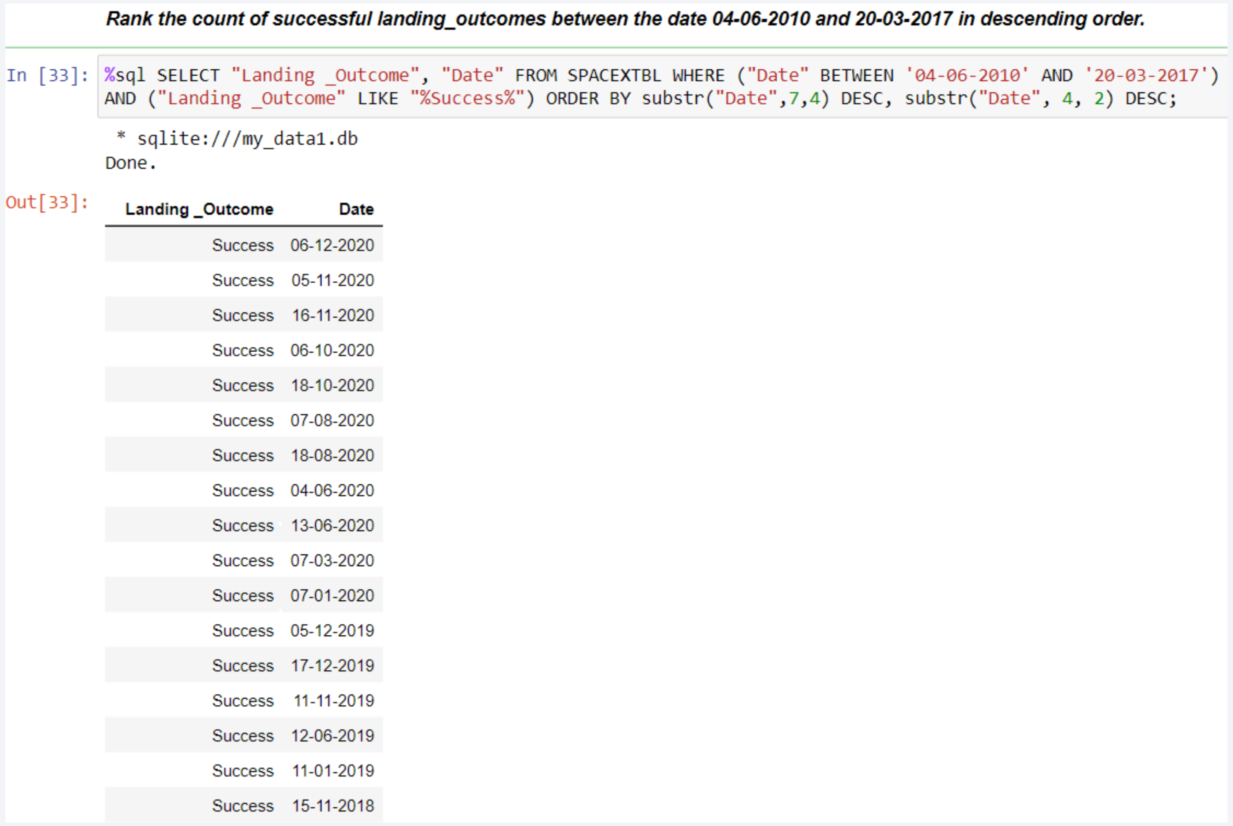
[23]:

MONTH	YEAR	Landing_Outcome	Mission_Outcome	Booster_Version	Launch_Site
10	2015	Failure (drone ship)	Success	F9 v1.1 B1012	CCAFS LC-40
04	2015	Failure (drone ship)	Success	F9 v1.1 B1015	CCAFS LC-40

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

- This is 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.
- Note: the whole output

Didn’t fit in the screenshot



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

```
In [33]: %sql SELECT "Landing _Outcome", "Date" FROM SPACEXTBL WHERE ("Date" BETWEEN '04-06-2010' AND '20-03-2017')
         AND ("Landing _Outcome" LIKE "%Success%") ORDER BY substr("Date",7,4) DESC, substr("Date", 4, 2) DESC;

          * sqlite:///my_data1.db
         Done.
Out[33]:
```

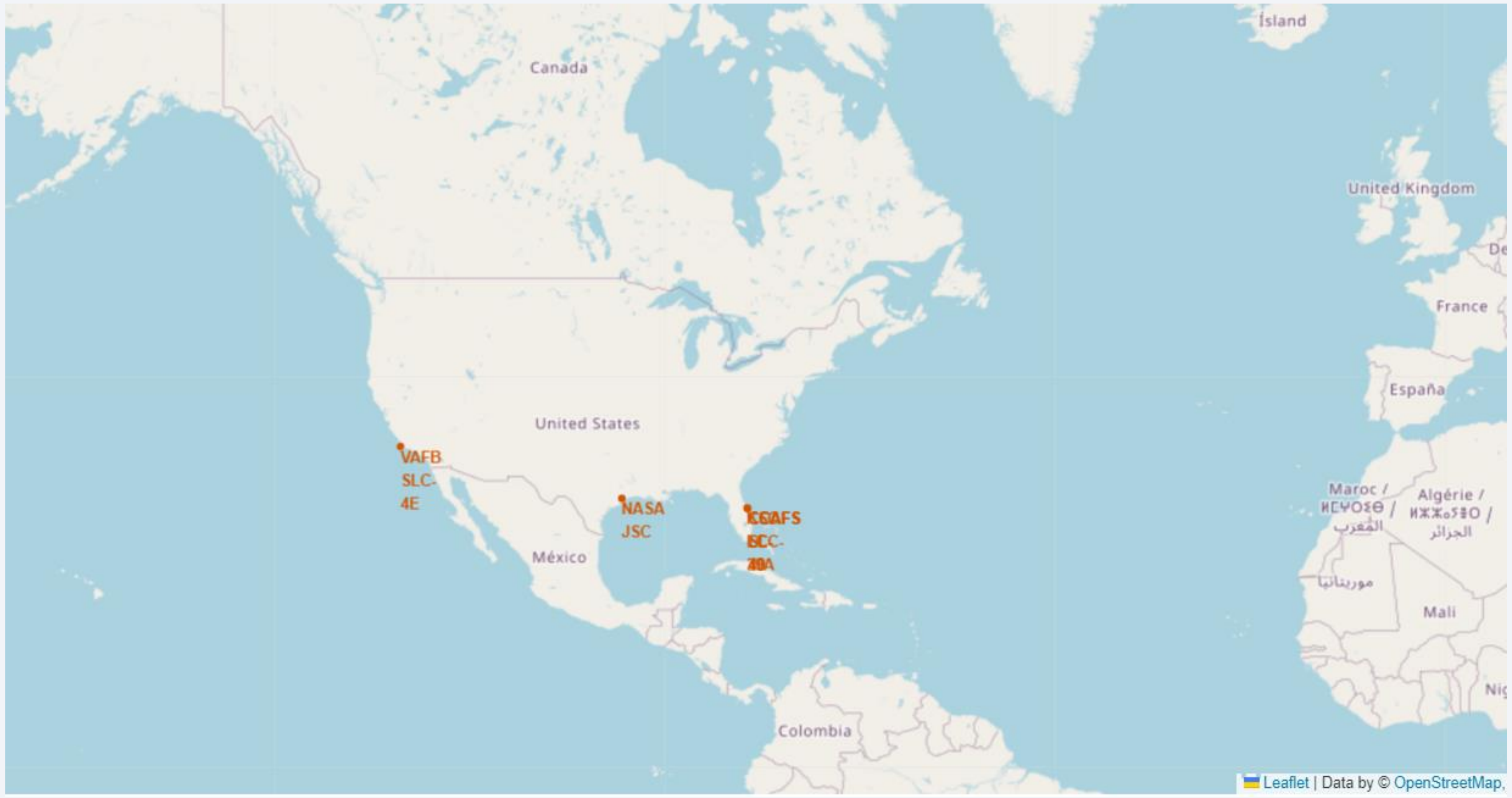
Landing _Outcome	Date
Success	06-12-2020
Success	05-11-2020
Success	16-11-2020
Success	06-10-2020
Success	18-10-2020
Success	07-08-2020
Success	18-08-2020
Success	04-06-2020
Success	13-06-2020
Success	07-03-2020
Success	07-01-2020
Success	05-12-2019
Success	17-12-2019
Success	11-11-2019
Success	12-06-2019
Success	11-01-2019
Success	15-11-2018

Section 3

Launch Sites Proximities Analysis

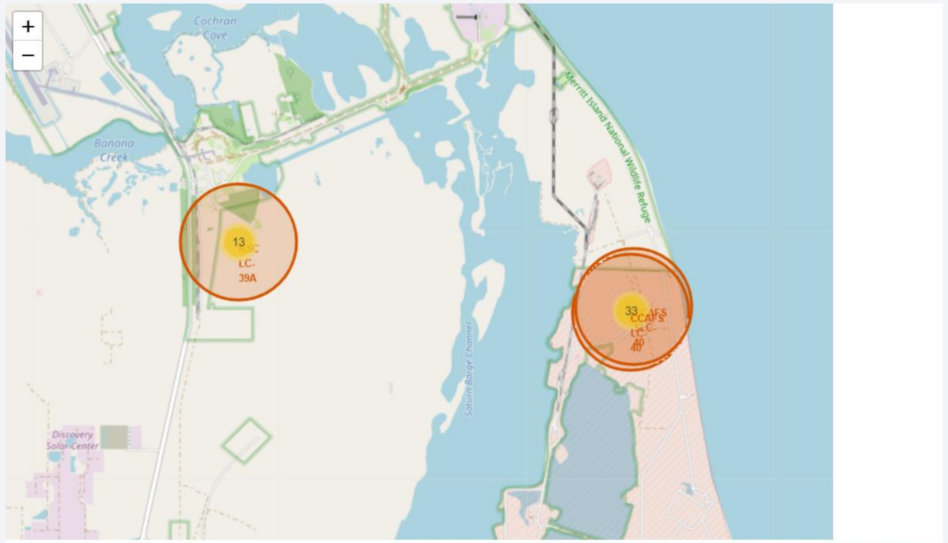
Global map of all launch sites with markers

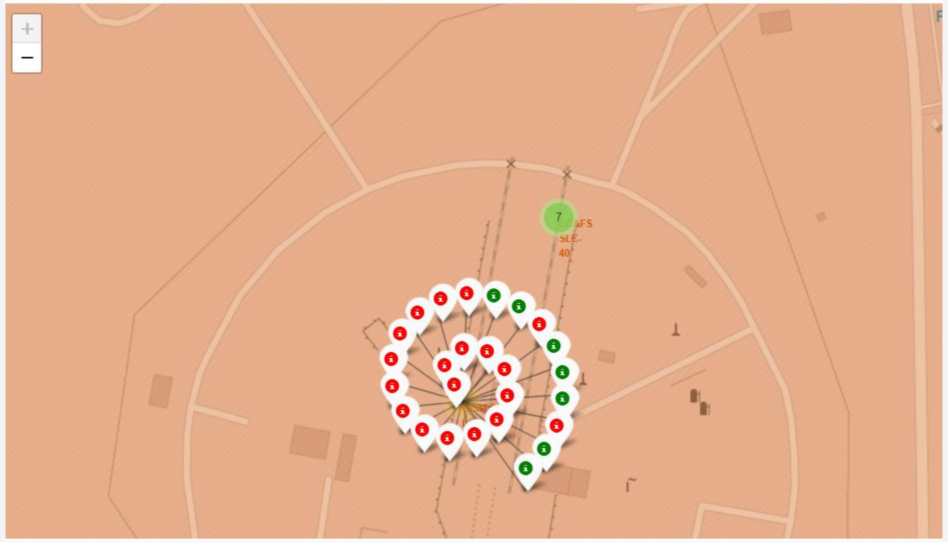
- Here we see all launch sites plus the NASA headquarters in Houston.



Markers showing launch sites and their successes/failures

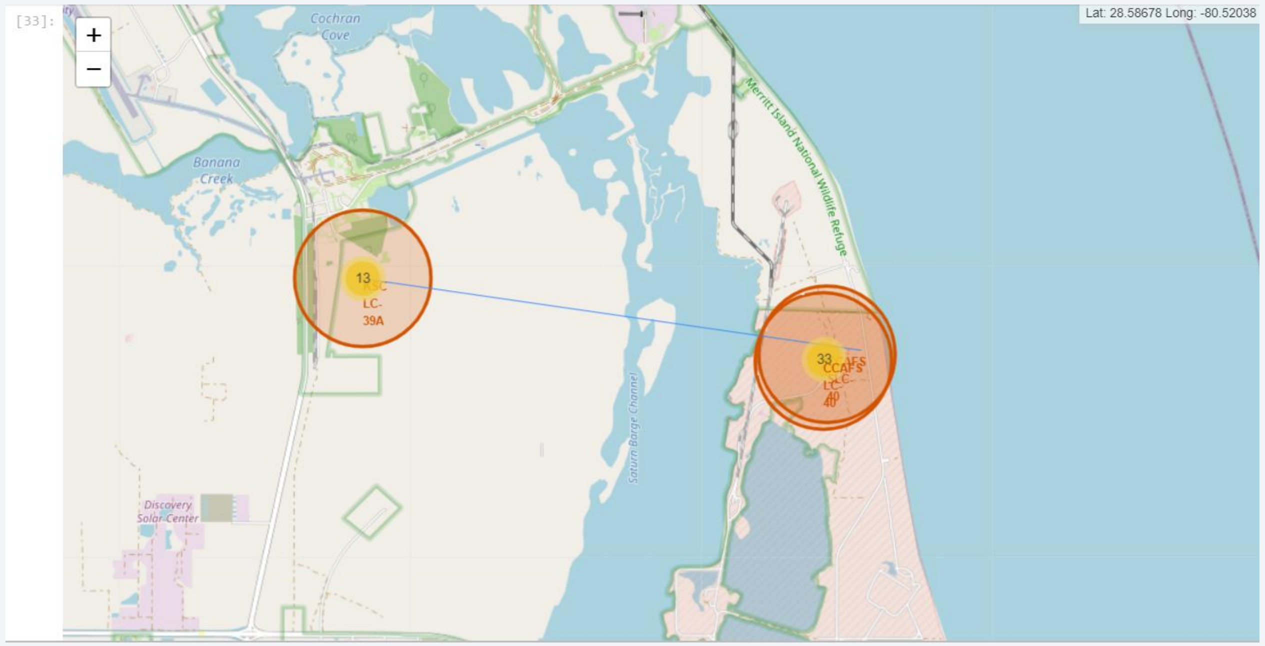
- This is the map that also shows successes (green) and failures

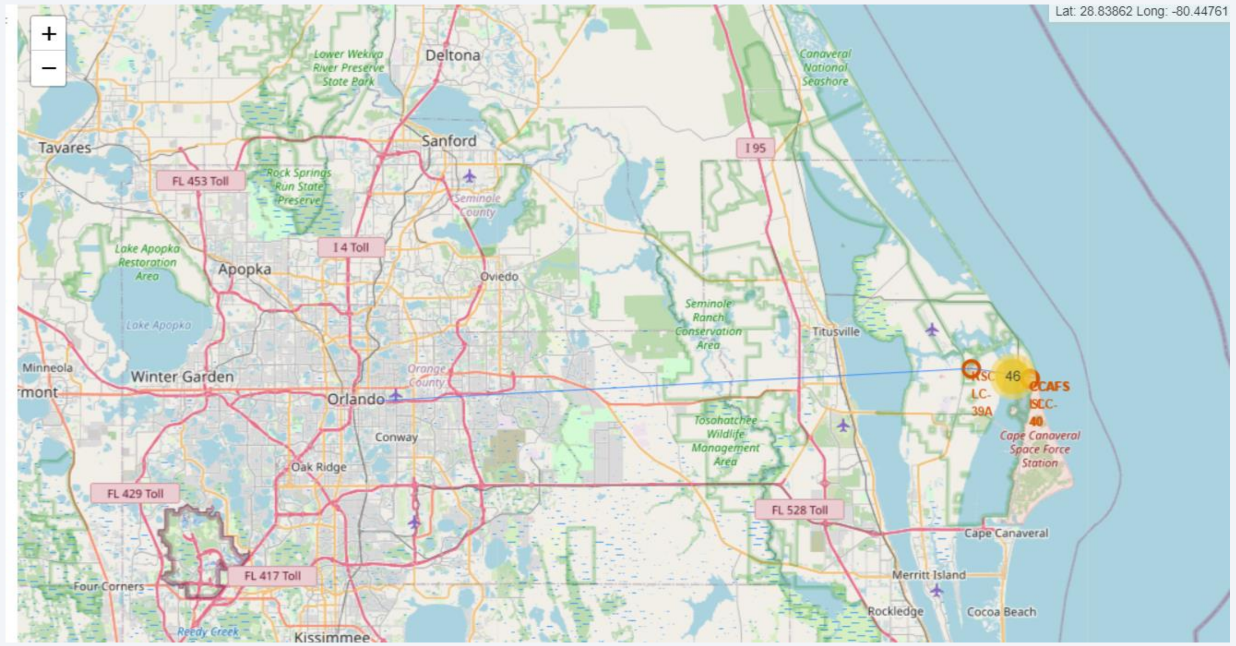


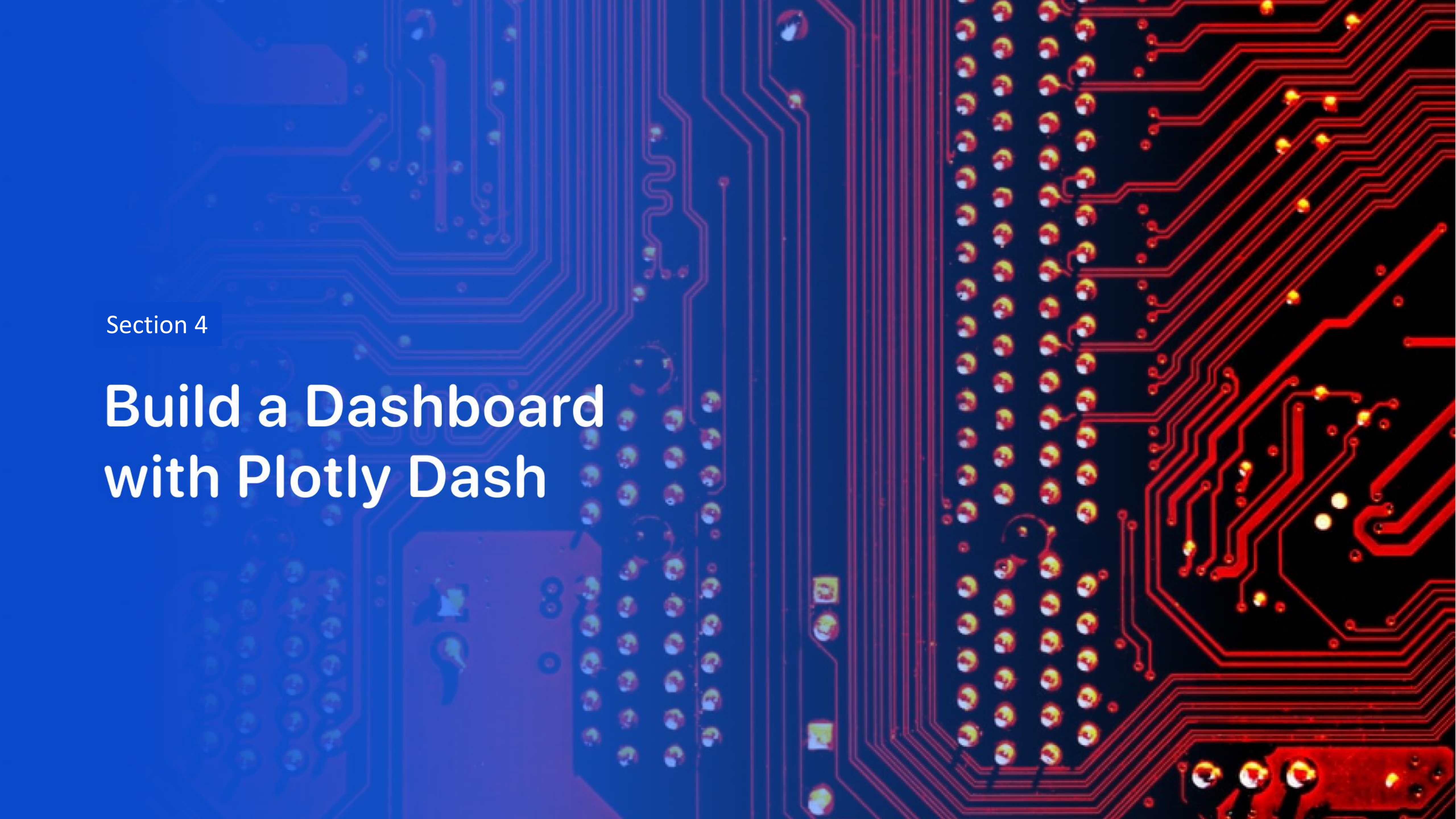


Launch Site distance to landmarks

- This map shows the distance between two launch sites, and between KSC LC39A and Orlando.





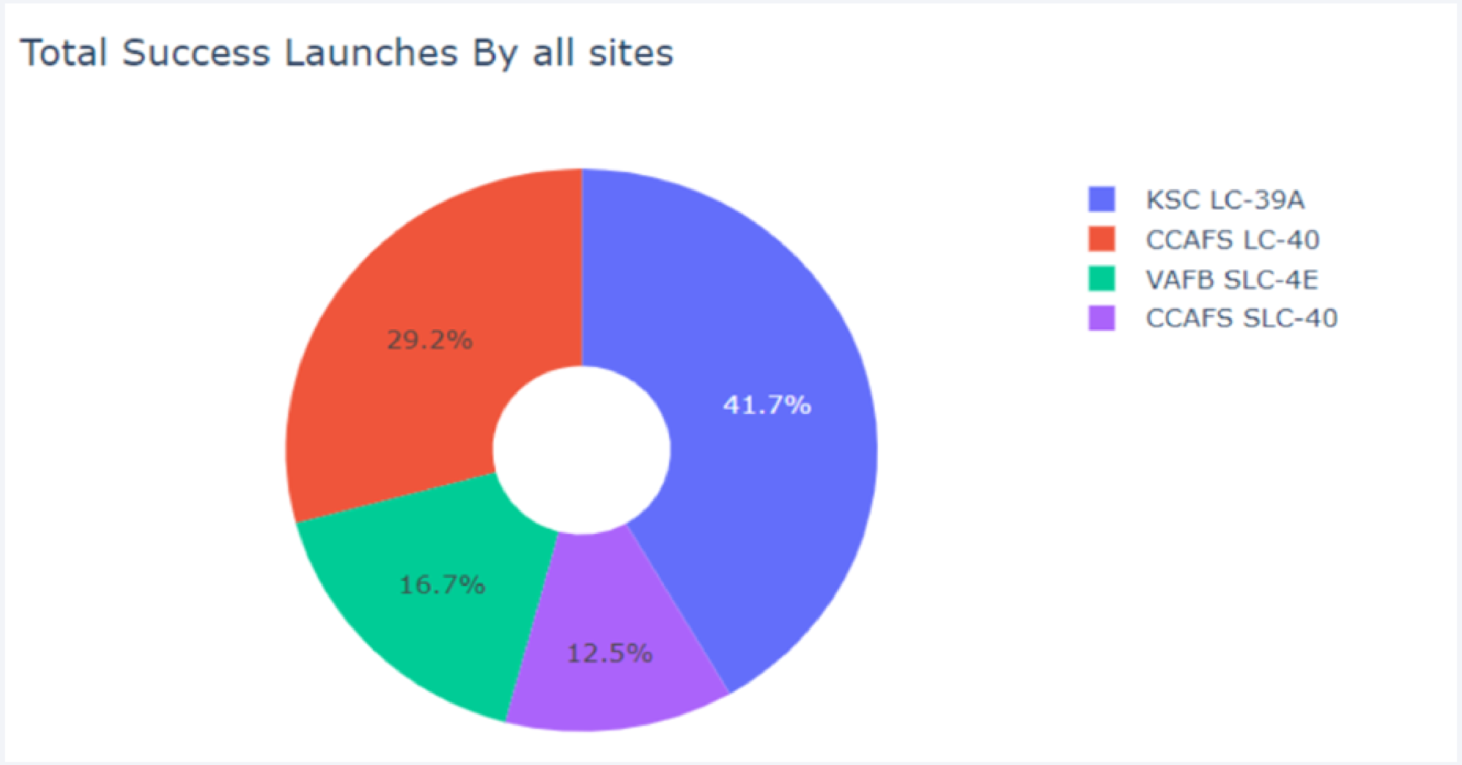


Section 4

Build a Dashboard with Plotly Dash

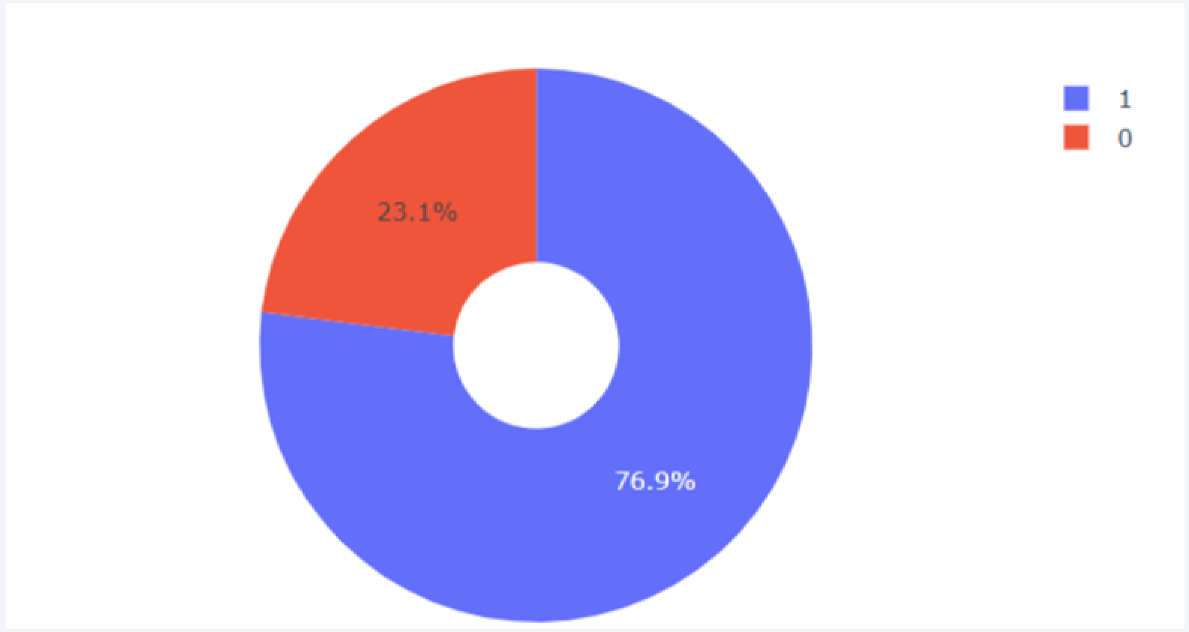
Pie chart: success rate achieved by each launch site

- We observe that KSC LC39A is the most successful site.



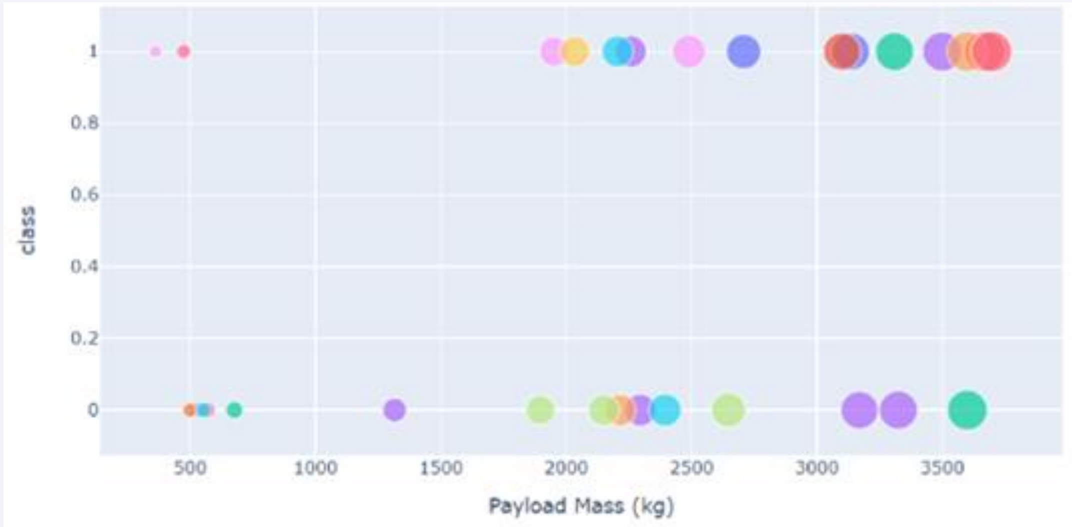
Pie chart: most successful launch site

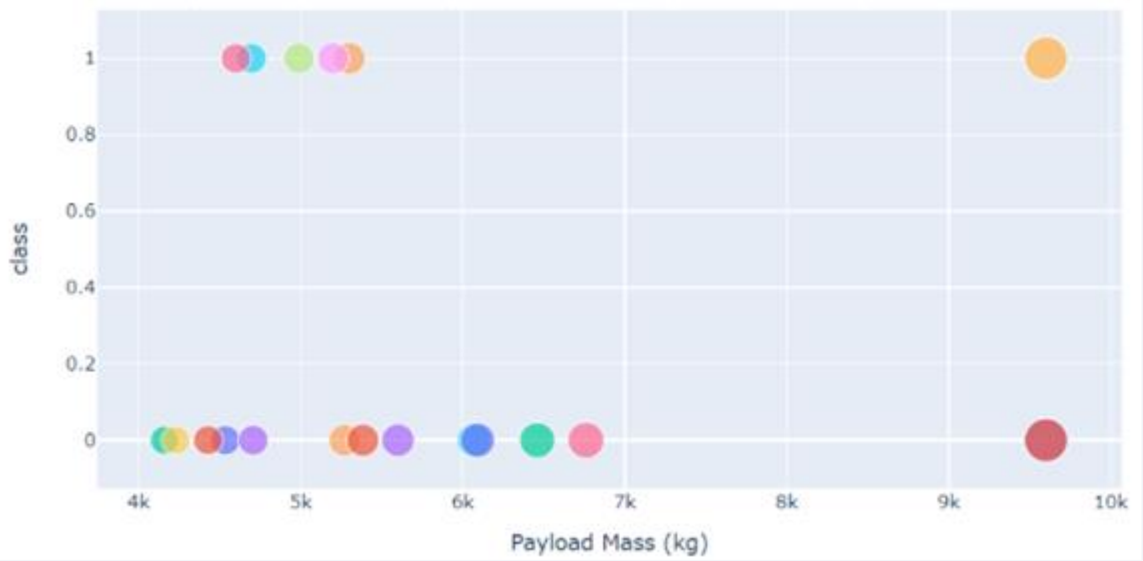
- We observe that KSC LC39A has a success rate of 76.9%



Scatter plot: Payload Mass vs Launch Outcome for all sites; different payloads selected with the slider

- The left figure shows the scatter plot for light Payload masses (0 – 4000kgs), and the right one for heavy Payload Masses (4000 – 10000kgs)



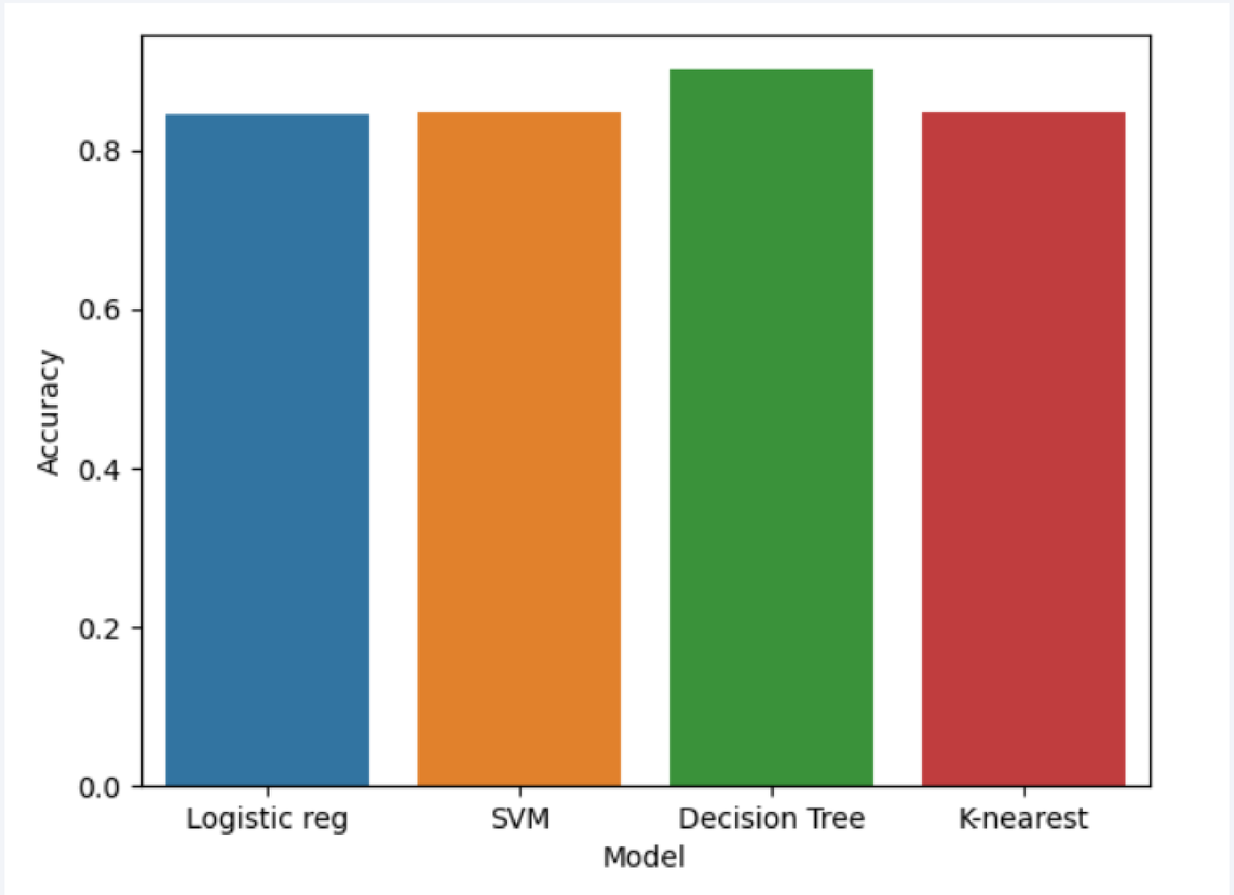


Section 5

Predictive Analysis (Classification)

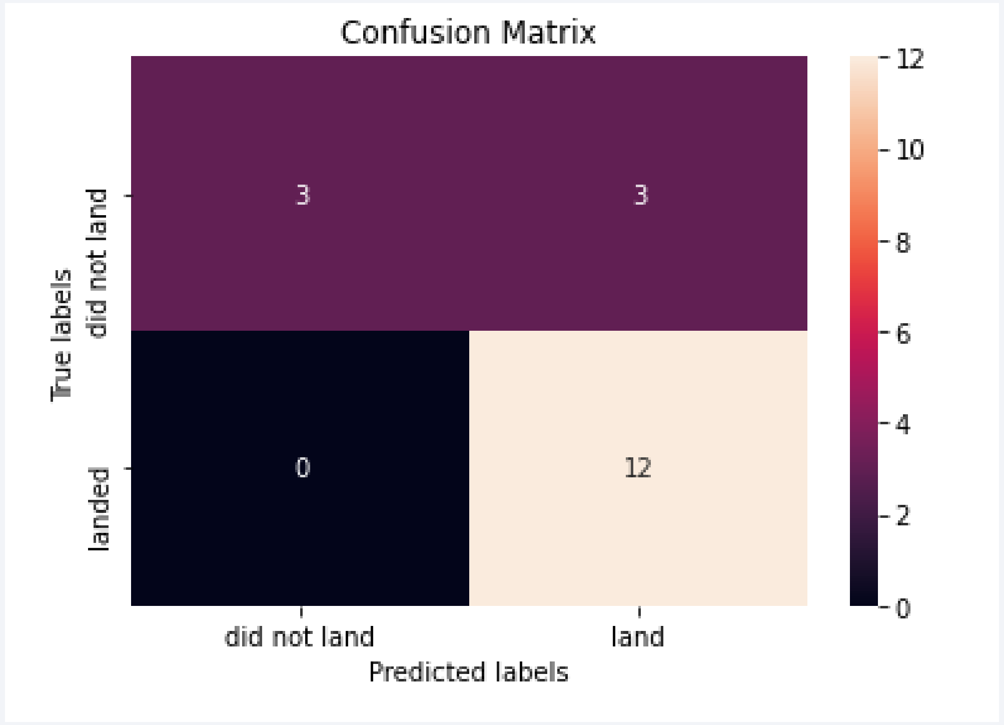
Classification Accuracy

- The model with the highest accuracy in the train sample was the Decision Tree.



Confusion Matrix

- This is the confusion matrix of the Decision Tree. We can see that there are no false negatives. However, there are false positives (3/12)= 25%



Conclusions

- We have performed a comprehensive analysis of the SpaceX data, which we obtain by a request to SpaceX API, and web scraping, and after cleaning and wrangling the data, we proceed to visualize and construct several insightful queries. Finally, we build machine learning models to predict the success of rocket landings.
- We find that, as the number of flights grows at a launch site, the success rate of said launch site will increase.
- The success rate of all launch sites increases from 2013 onward. On the other hand, KSC LC-39A is the most successful launch site from all sites in the database, and the highest success rate orbits for launching rockets are ES-L1, GEO, HEO, SSO, and VLEO.
- Finally, the best performing model to predict the success of a Stage-1 landing is the Decision Tree classifier.

Thank you!