IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Jackeline Roberts
March 20th, 2021

Outline

Executive Summary

- Summary of methodologies

  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Dara 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

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

- Problems you want to find answers

  - What factors determine if the rocket will land successfully?
  - The interaction amongst various features that determine the success rate of a successful landing.
  - What operating conditions needs to be in place to ensure a successful landing program.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX API and web scraping from Wikipedia
- Perform data wrangling
  - 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

- The data was collected using various methods

  - Data collection was done using get request to the SpaceX API.

  - Next, we decoded the response content as a Jason using .json() function call and turn it into a pandas dataframe using .jason_normalize()

  - We the cleaned the data, checked for missing values and fill in missing values where necessary.

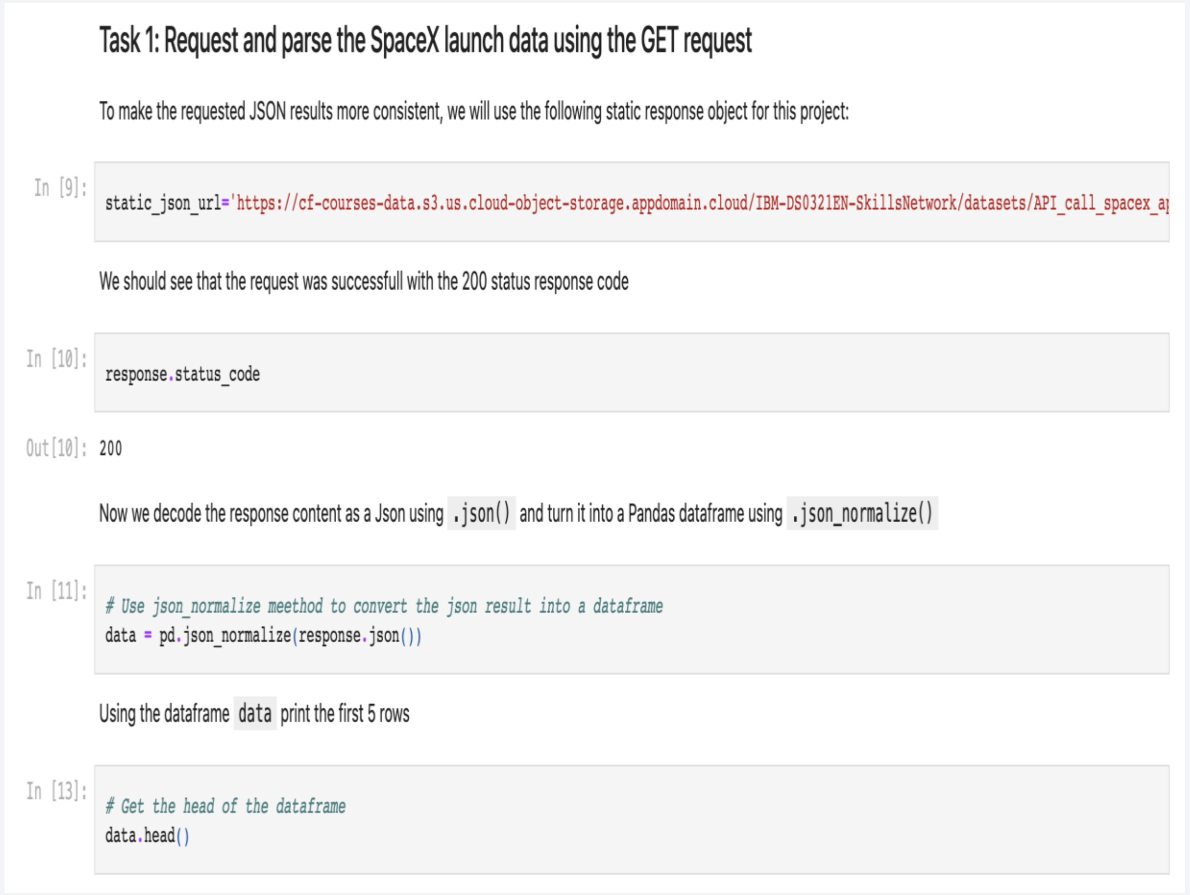
  - In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeuatifulSoup.

  - The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis.

Data Collection – SpaceX API

- We used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.
- The link to the notebook is:

https://github.com/jackelineRoberts/SpaceX-Falcon-9-first-stage-Landing-Prediction---Lab-1-Collecting-the-data.git



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

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

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

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

```
In [10]: response.status_code

Out[10]: 200
```

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

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

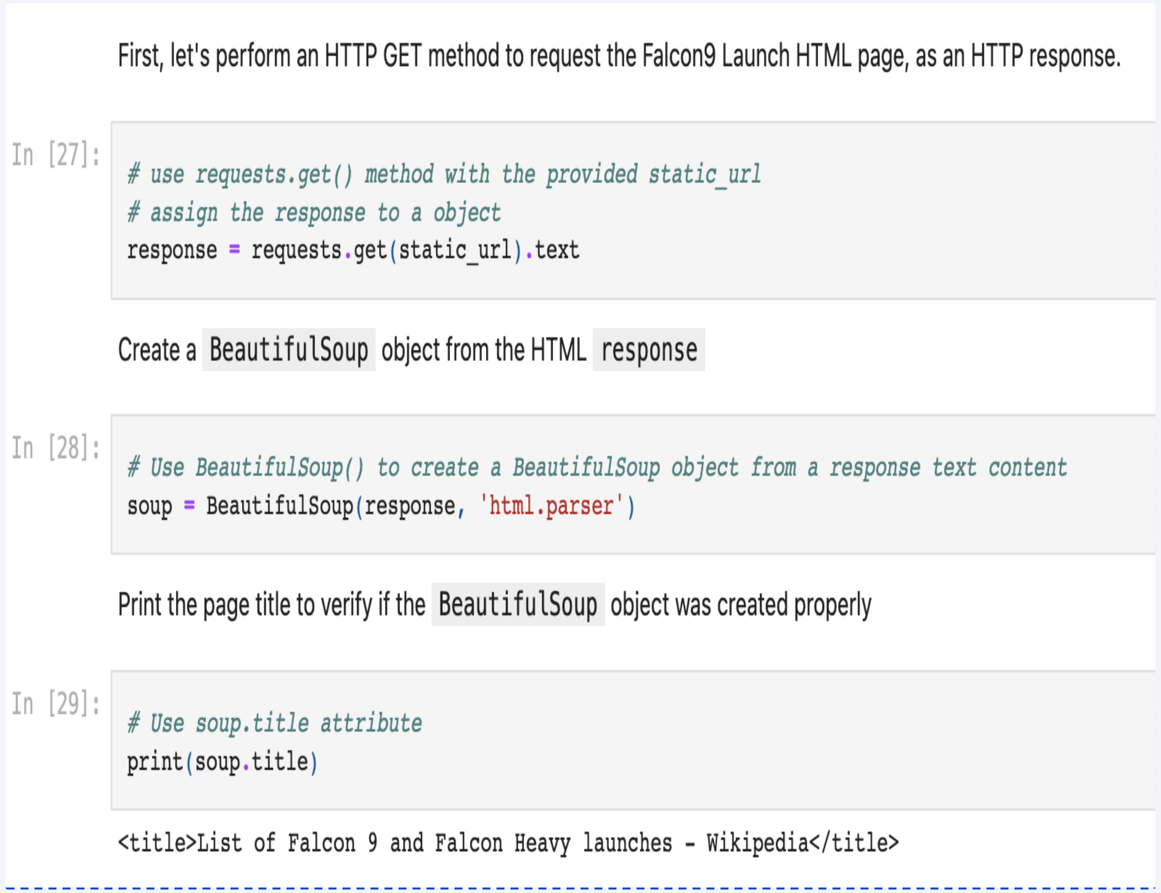
Using the dataframe `data` print the first 5 rows

```
In [13]: # Get the head of the dataframe
         data.head()
```

Data Collection - Scraping

- We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup
- We parsed the table and converted it into a pandas dataframe.
- The link to the notebook is:

https://github.com/jackelineRoberts/Space-X-Falcon-9-First-Stage-Landing-Prediction.git



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

```
In [27]: # use requests.get() method with the provided static_url
         # assign the response to a object
         response = requests.get(static_url).text
```

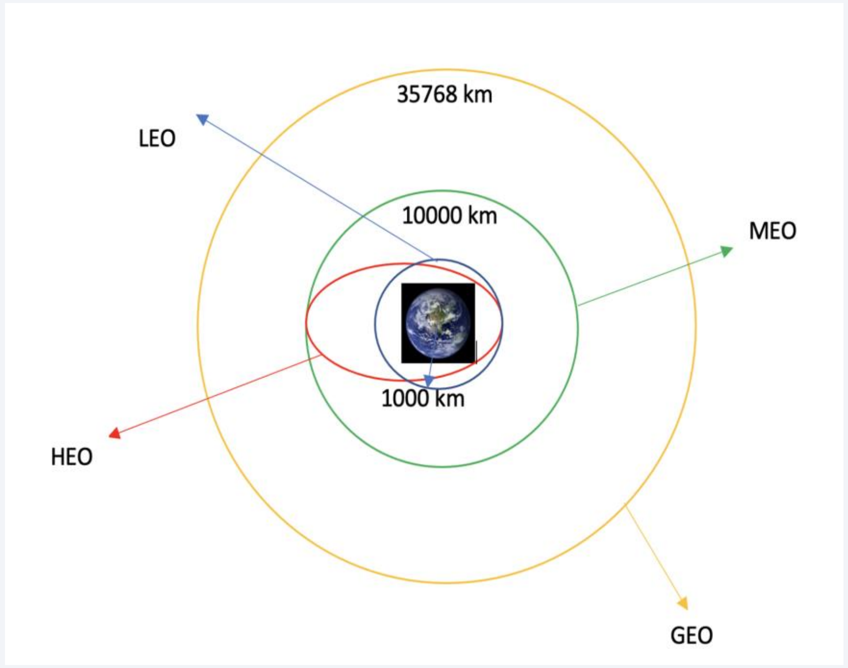
Create a `BeautifulSoup` object from the HTML `response`

```
In [28]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
         soup = BeautifulSoup(response, 'html.parser')
```

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

```
In [29]: # Use soup.title attribute
         print(soup.title)

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

Data Wrangling



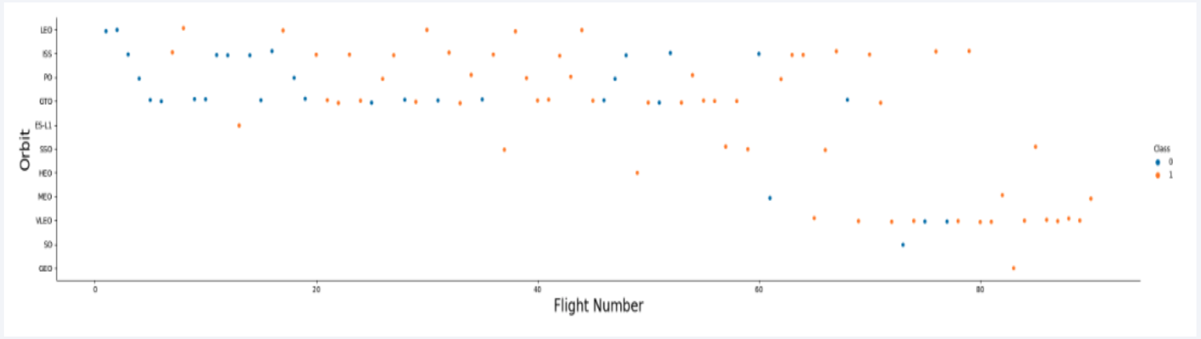
- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits.
- We created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is:

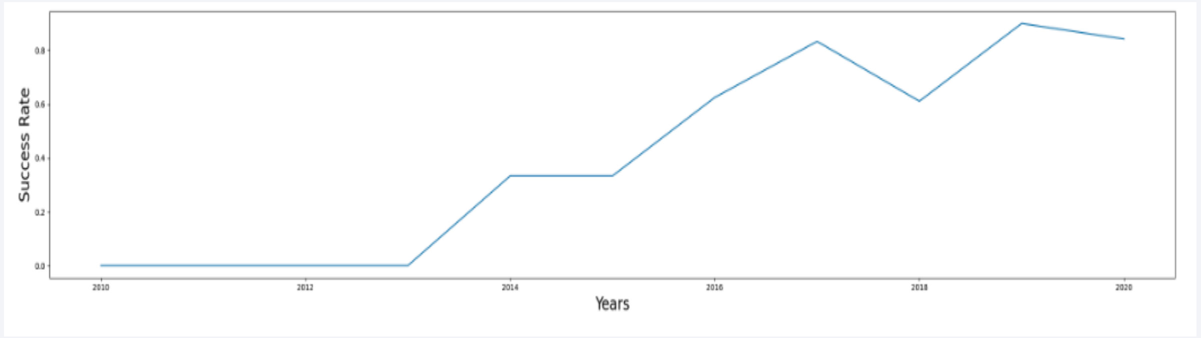
https://github.com/jackelineRoberts/Peer-graded-Assignment--Lab-2.git

EDA with Data Visualization

- Scatter plot with Payload vs Orbit and Class as hue: shows the relationship between two variables (Payload and Orbit) and how they vary by category (Class).
- Line chart with extracted year vs success rate: shows how success rates for a particular event have changed over time, and can help identify trends or factors that may have influenced those changes.
- The link to the notebook is:

https://github.com/jackelineRoberts/SpaceX-Falcon-9-First-Stage-Landing-Prediction---Assignment-Exploring-and-Preparing-Data.git





EDA with SQL

- We loaded the SpaceX dataset into a PostgreSQL database without leaving the jupyter notebook.
- We applied EDA with SQL to get insight from the data. We wrote queries to find out for instance:

- The names of unique launch sites in the mission.
- The total payload mass carried by boosters launched by NASA (CRS)
- The average payload mass carried by booster version F9 v1.1
- The total number of successful and failure mission outcomes
- The failed landing outcomes in drone ship, their booster version and launch site names.

- The link to the notebook is: https://github.com/jackelineRoberts/Assignment-SQL-Notebook-for-Peer-Assignment.git

Build an Interactive Map with Folium

- We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1 .i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the distances between a launch site to its proximities. We answered some question for instance:

- Are launch sites near railways, highways and coastlines.
- Do launch sites keep certain distance away from cities.

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- The link to the notebook is: https://github.com/jackelineRoberts/SpaceX-Falcon-9-First-Stage-Landing-Prediction---Assignment-Exploring-and-Preparing-Data.git

Predictive Analysis (Classification)

- We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.
- The link to the notebook is: https://github.com/jackelineRoberts/Machine-Learning-Prediction.git

Results

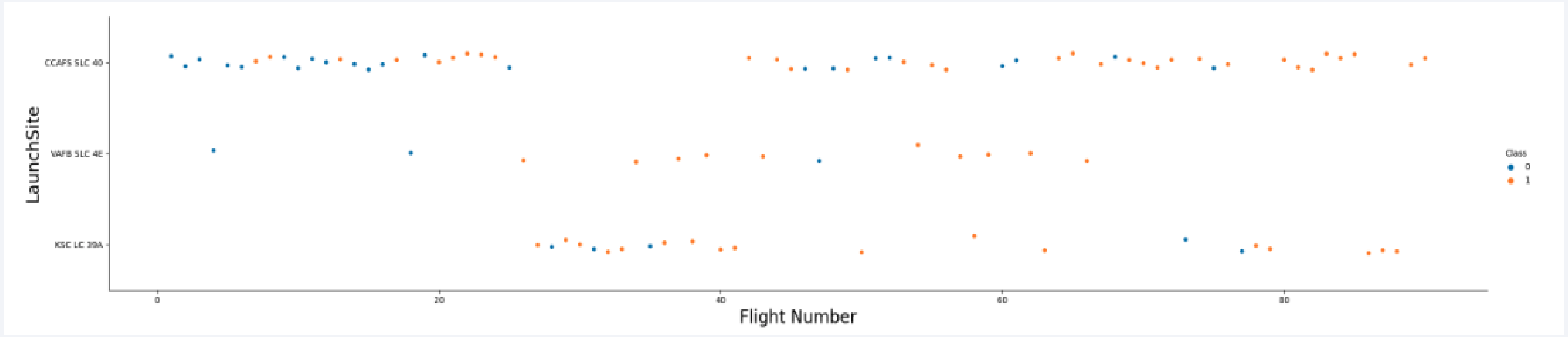
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

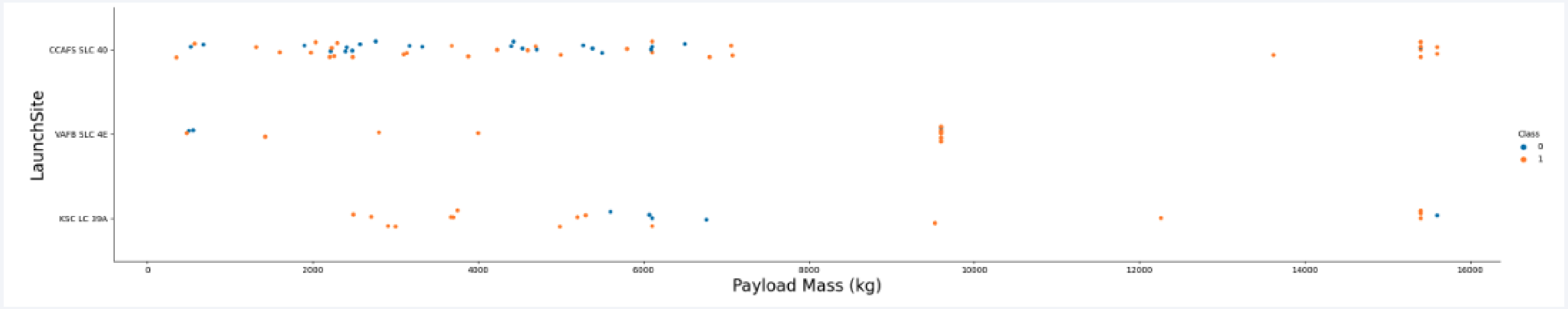
Flight Number vs. Launch Site

- From the plot, we found that the larger the flight amount at a launch site, the greater the success rate at a launch site.



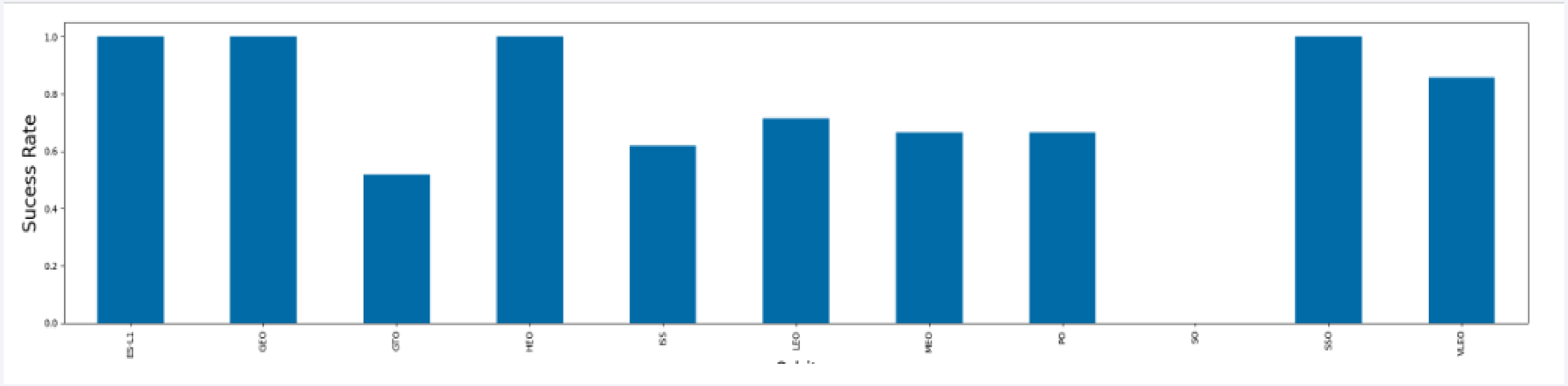
Payload vs. Launch Site

- The greater the payload mass for launch site CCAFS SLC 40 the higher the success rate for rocket.



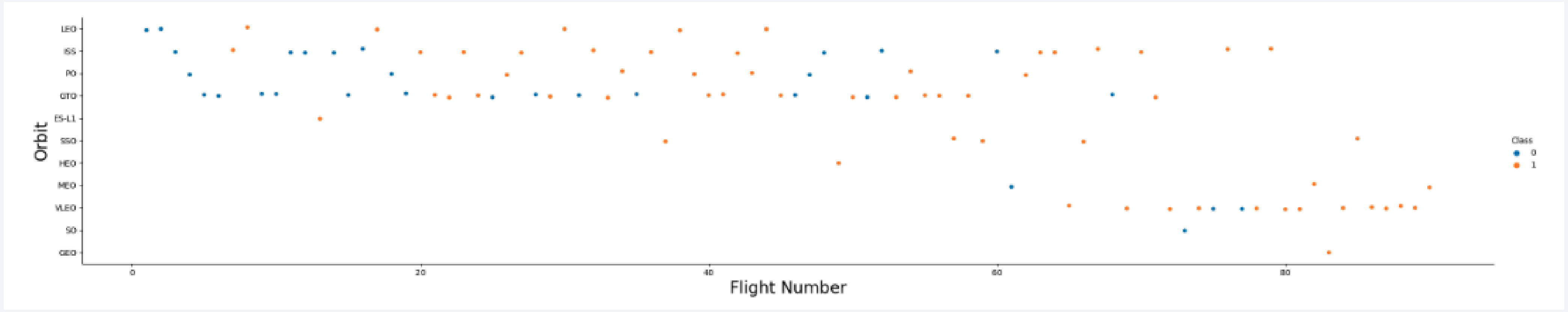
Success Rate vs. Orbit Type

- From the plot, we can see that ES-L1, GEO, HEO, SSO, VLEO had the most success rate.



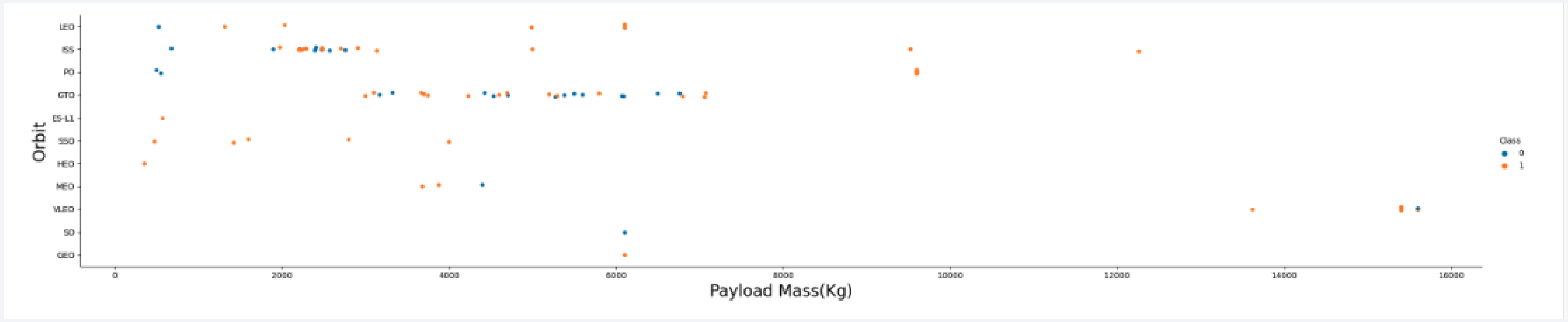
Flight Number vs. Orbit Type

- The plot below shows the Flight Number vs. Orbit type. We observe that in the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.



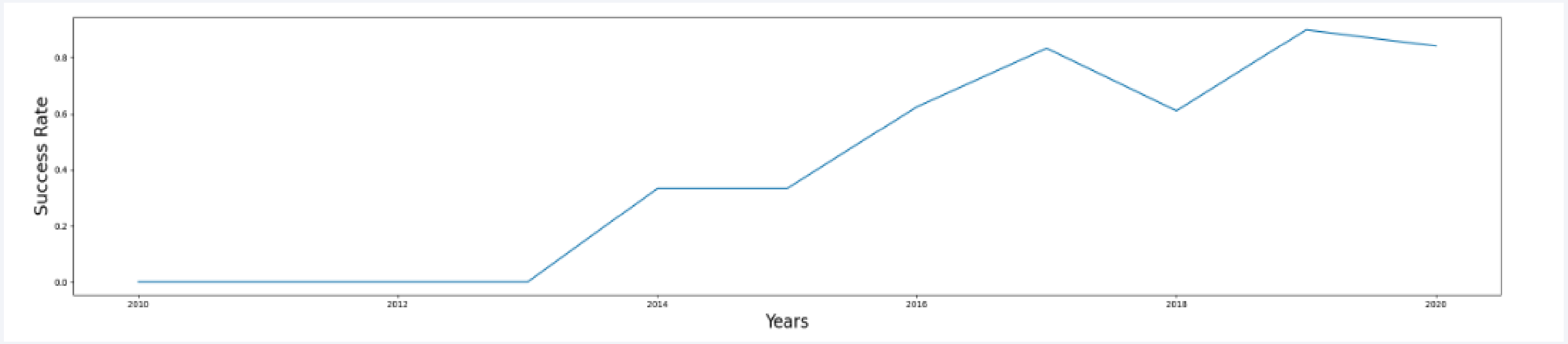
Payload vs. Orbit Type

- We can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.,



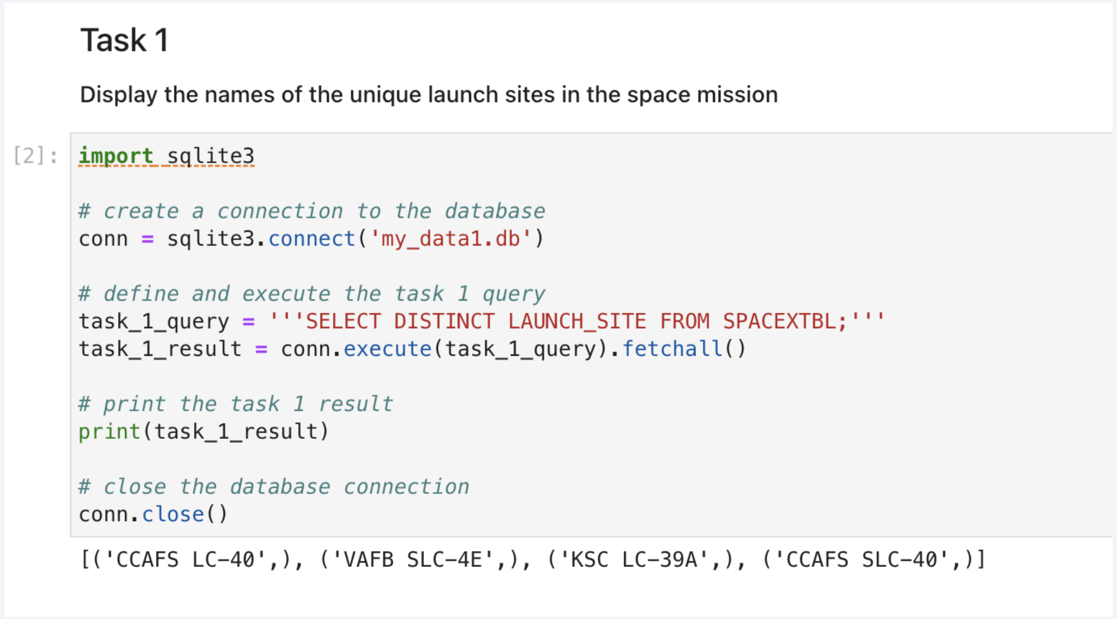
Launch Success Yearly Trend

- From the plot, we can observe that success rate since 2023 kept on increasing till 2020.



All Launch Site Names

- We used the key word DISTINCT to show only unique launch sites from the SpaceX data.



Task 1

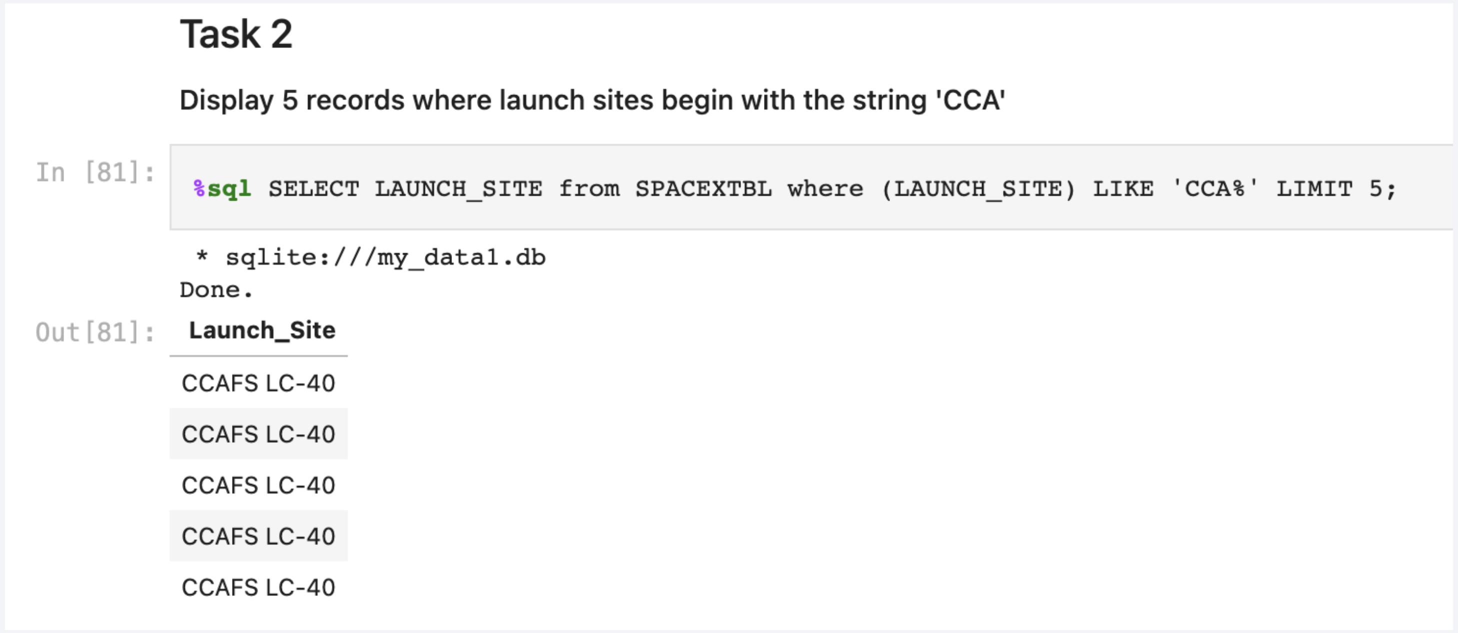
Display the names of the unique launch sites in the space mission

```
[2]: import sqlite3

     # create a connection to the database
     conn = sqlite3.connect('my_data1.db')

     # define and execute the task 1 query
     task_1_query = '''SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL;'''
     task_1_result = conn.execute(task_1_query).fetchall()

     # print the task 1 result
     print(task_1_result)

     # close the database connection
     conn.close()
```

```
[('CCAFS LC-40',), ('VAFB SLC-4E',), ('KSC LC-39A',), ('CCAFS SLC-40',)]
```

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`



Task 2

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

```
In [81]: %sql SELECT LAUNCH_SITE from SPACEXTBL where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5;
```

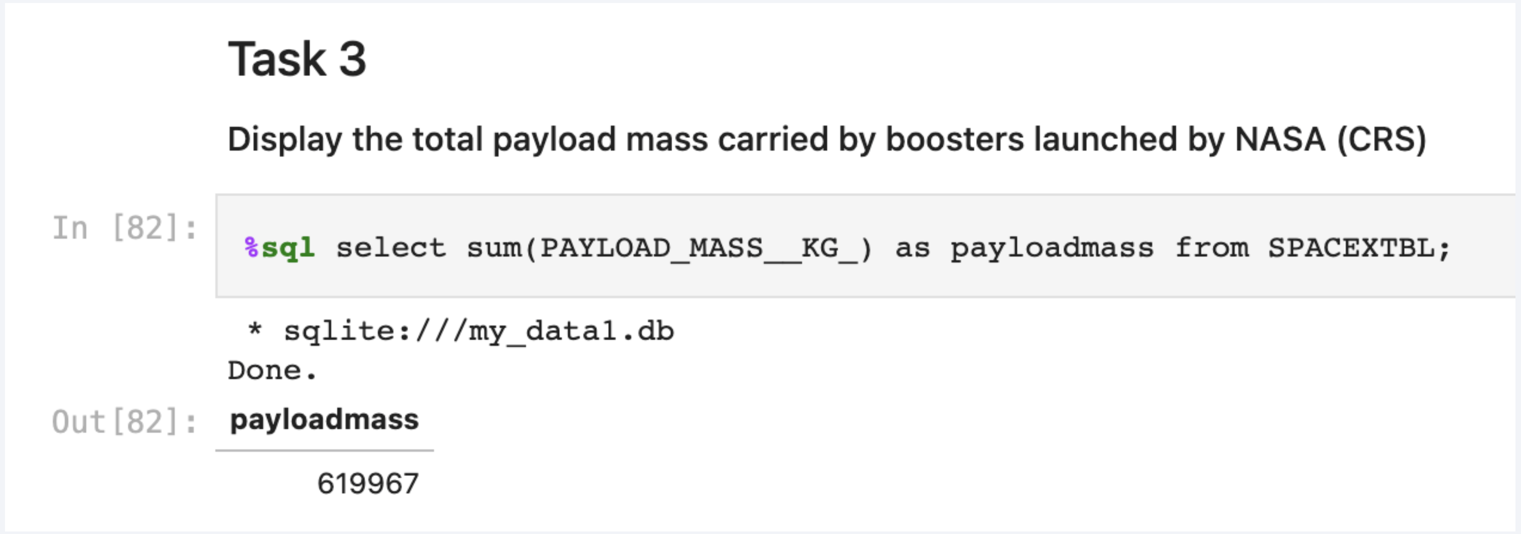
```
 * sqlite:///my_data1.db
Done.
```

Out[81]:

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Total Payload Mass

- We calculate the total payload carried by boosters from NASA



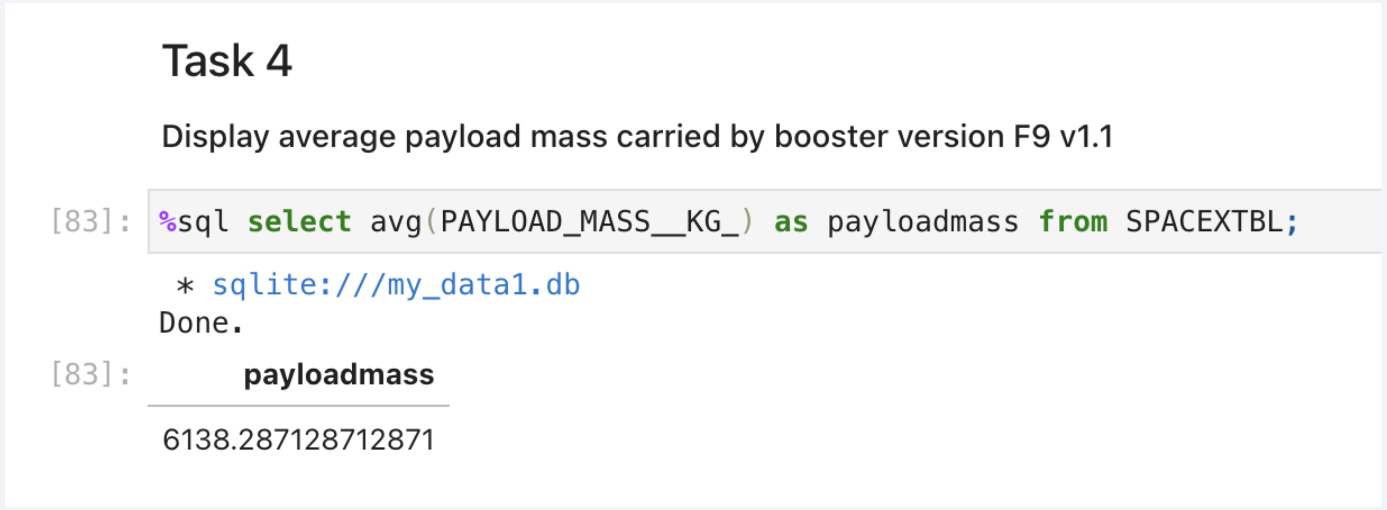
Task 3

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

```
In [82]: %sql select sum(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL;

 * sqlite:///my_data1.db
Done.
Out[82]: payloadmass
              619967
```

Average Payload Mass by F9 v1.1

- We calculate the average payload mass carried by booster version F9 v1.1



Task 4

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

```
[83]: %sql select avg(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL;
```

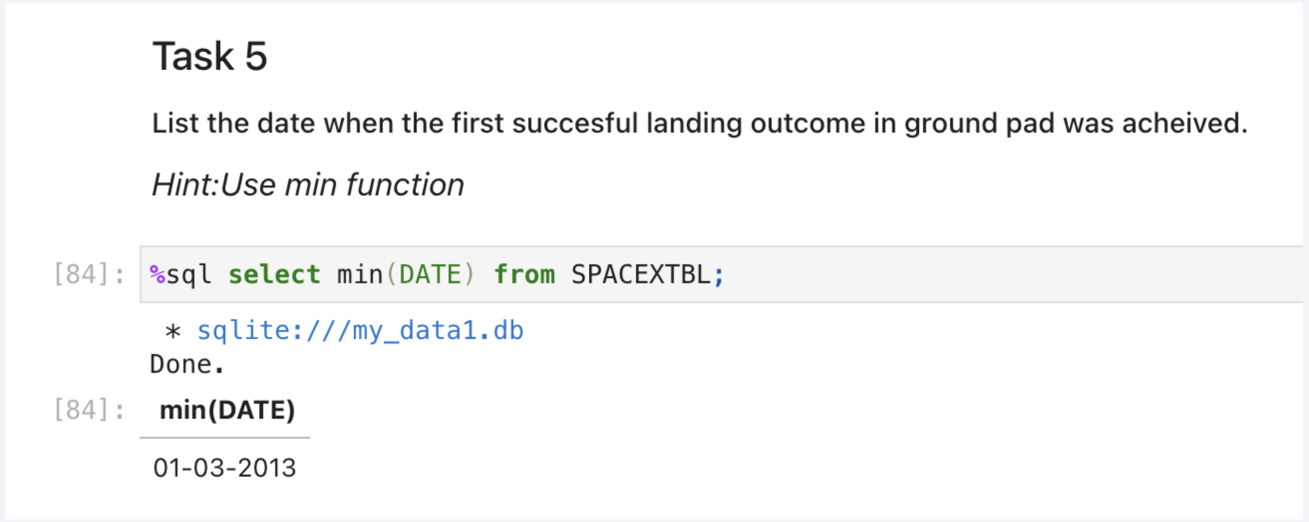
```
 * sqlite:///my_data1.db
Done.
```

[83]:

payloadmass
6138.287128712871

First Successful Ground Landing Date

- We find the dates of the first successful landing outcome on ground pad



Task 5

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

Hint:Use min function

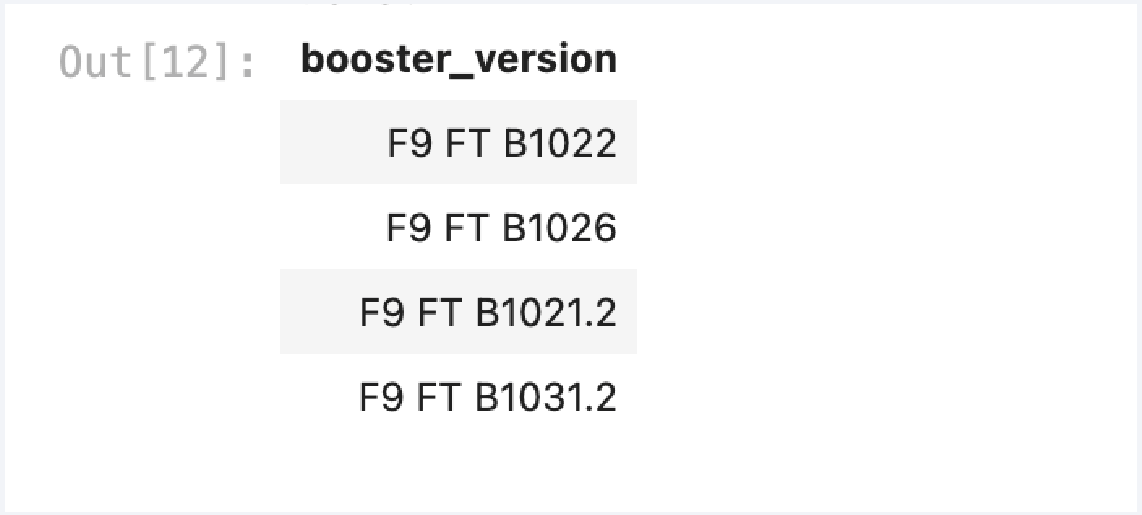
```
[84]: %sql select min(DATE) from SPACEXTBL;

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

[84]:

min(DATE)
01-03-2013

Successful Drone Ship Landing with Payload between 4000 and 6000

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

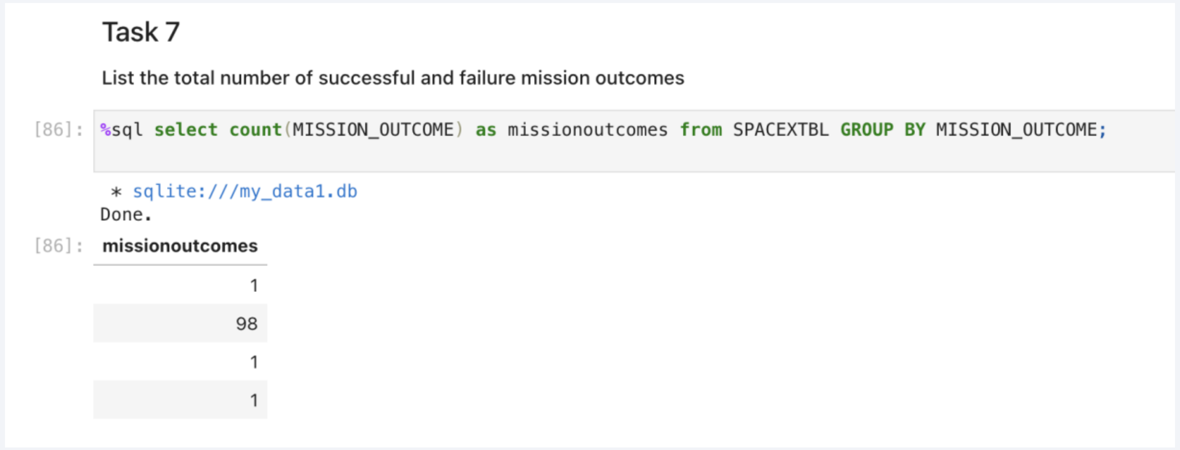


```
Out[12]:
```

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

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes



Task 7

List the total number of successful and failure mission outcomes

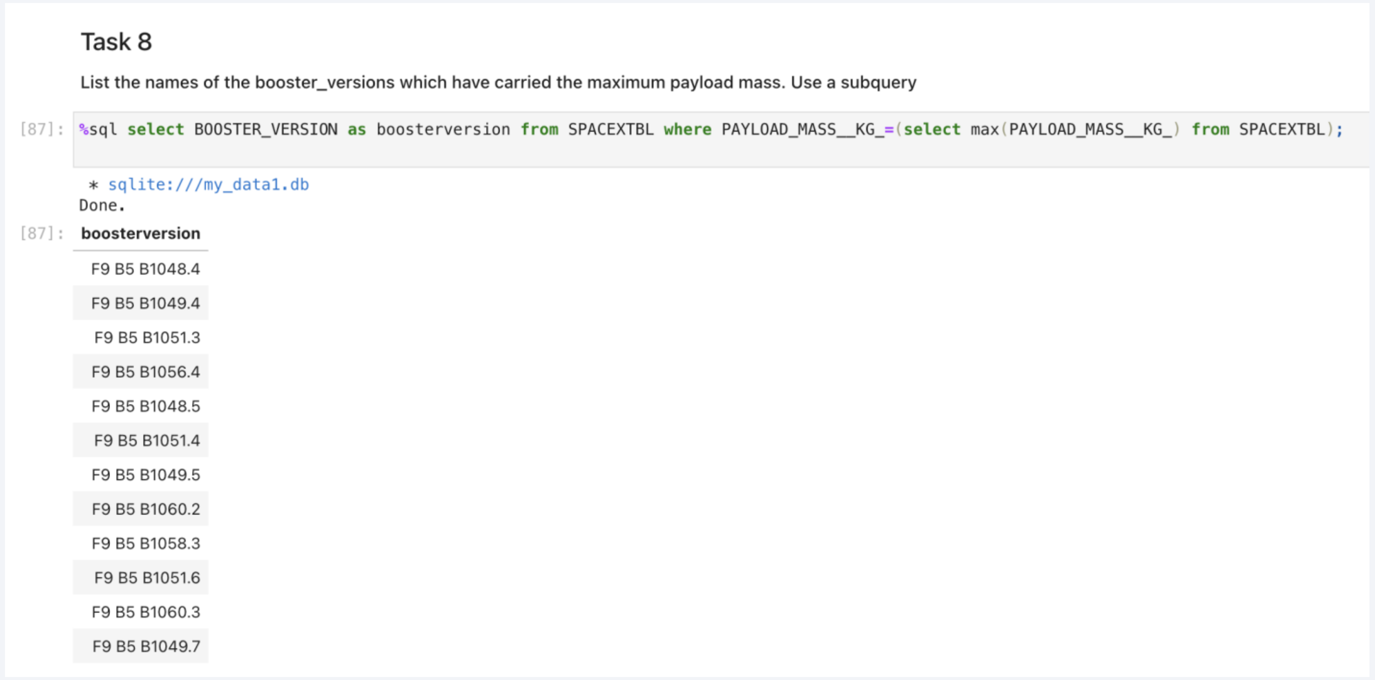
```
[86]: %sql select count(MISSION_OUTCOME) as missionoutcomes from SPACEXTBL GROUP BY MISSION_OUTCOME;

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

[86]:

missionoutcomes
1
98
1
1

Boosters Carried Maximum Payload

- We list the names of the booster 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

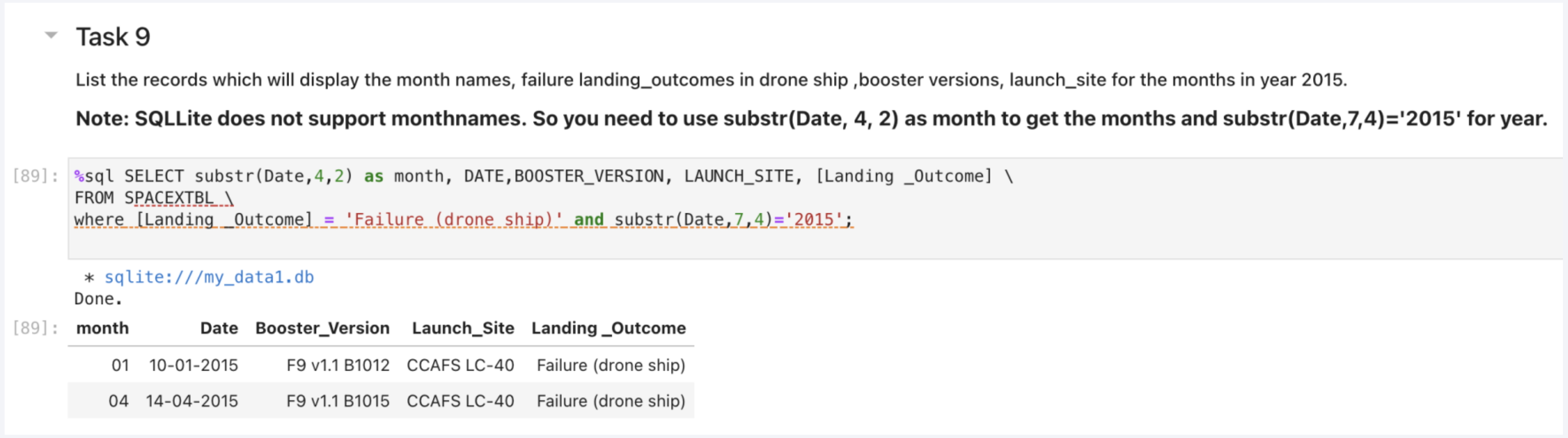
```
[87]: %sql select BOOSTER_VERSION as boosterversion from SPACEXTBL where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACEXTBL);

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

[87]:

boosterversion
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

2015 Launch Records

- We list the 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.

```
[89]: %sql SELECT substr(Date,4,2) as month, DATE,BOOSTER_VERSION, LAUNCH_SITE, [Landing _Outcome] \
      FROM SPACEXTBL \
      where [Landing _Outcome] = 'Failure (drone ship)' and substr(Date,7,4)='2015';
```

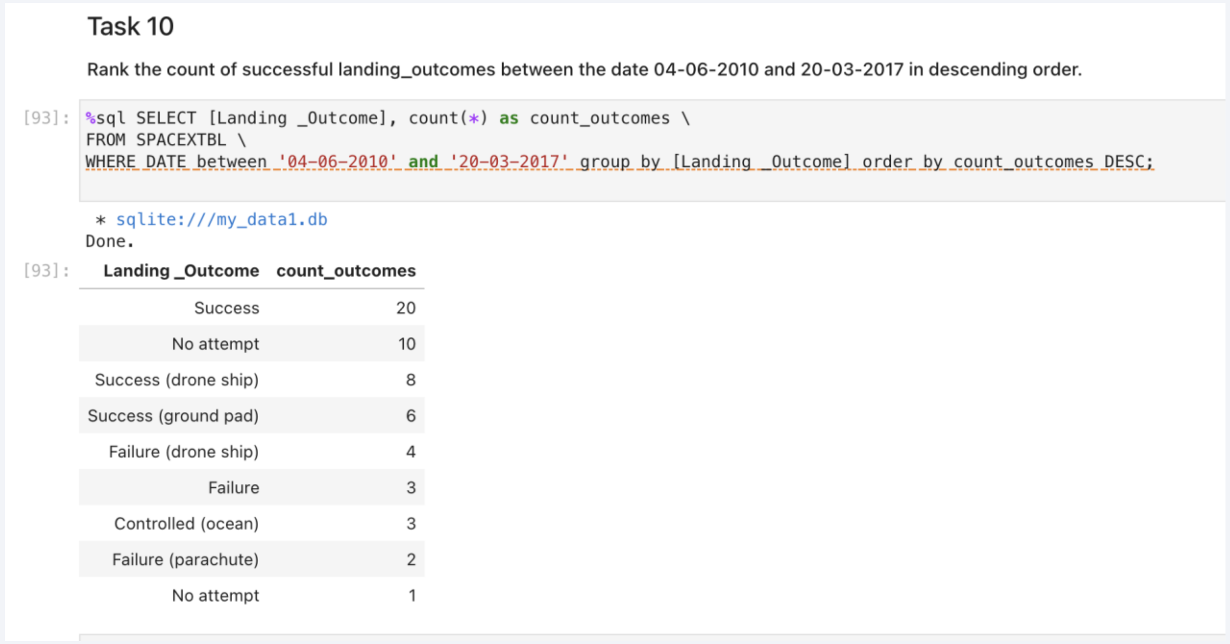
```
 * sqlite:///my_data1.db
Done.
```

[89]:

month	Date	Booster_Version	Launch_Site	Landing _Outcome
01	10-01-2015	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
04	14-04-2015	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order



Task 10

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

```
[93]: %sql SELECT [Landing _Outcome], count(*) as count_outcomes \
FROM SPACEXTBL \
WHERE DATE between '04-06-2010' and '20-03-2017' group by [Landing _Outcome] order by count_outcomes DESC;
```

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

[93]:

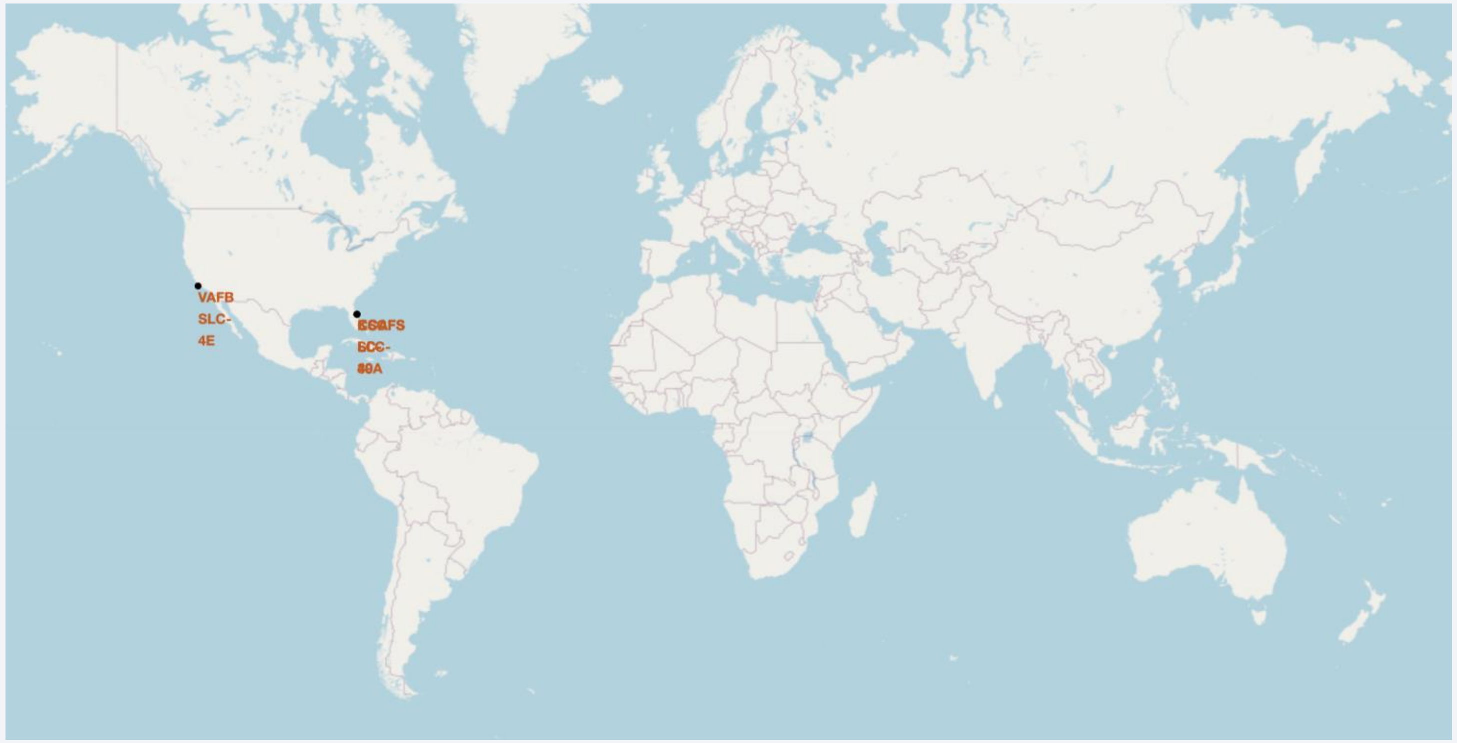
Landing _Outcome	count_outcomes
Success	20
No attempt	10
Success (drone ship)	8
Success (ground pad)	6
Failure (drone ship)	4
Failure	3
Controlled (ocean)	3
Failure (parachute)	2
No attempt	1

Section 3

Launch Sites Proximities Analysis

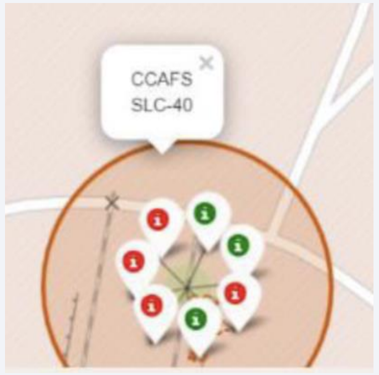
All launch sites global map markers

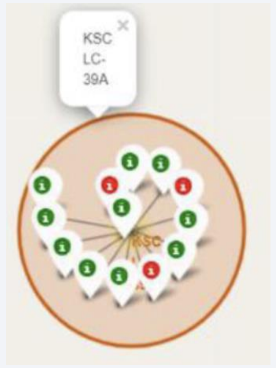
- We can see that the SpaceX launches are in the United States of America coasts. Florida and California

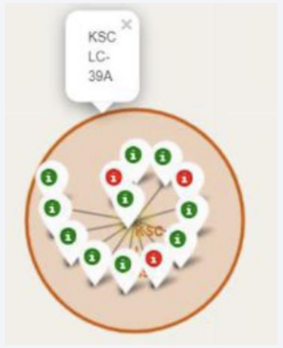


Markers showing launch sites with color labels

- Florida launch sites (Green marker shows successful launches and red marker shows failures.

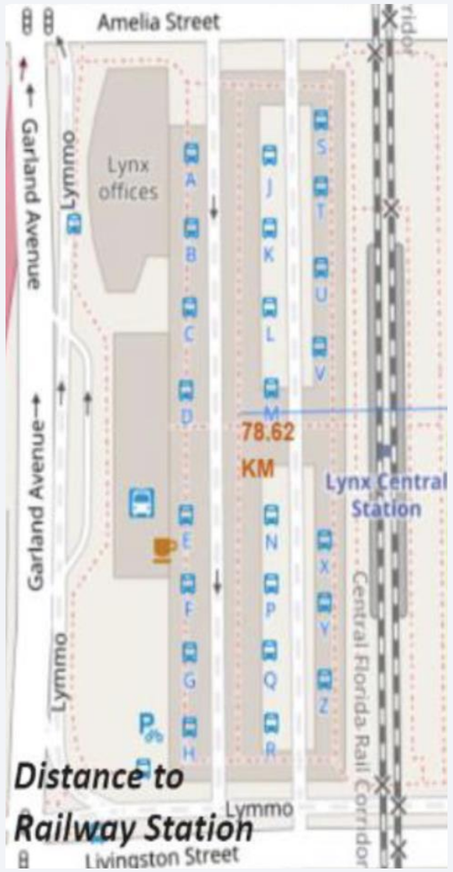


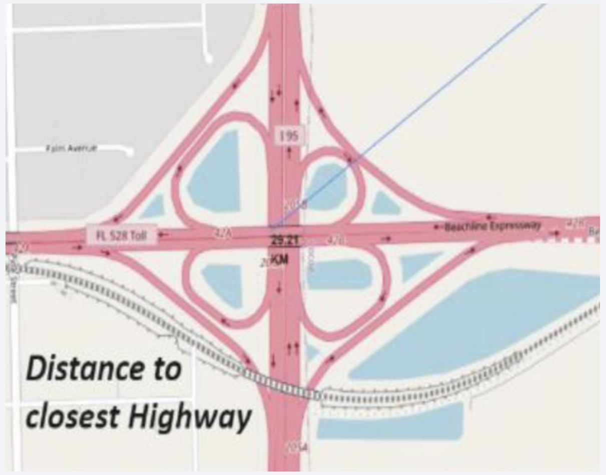




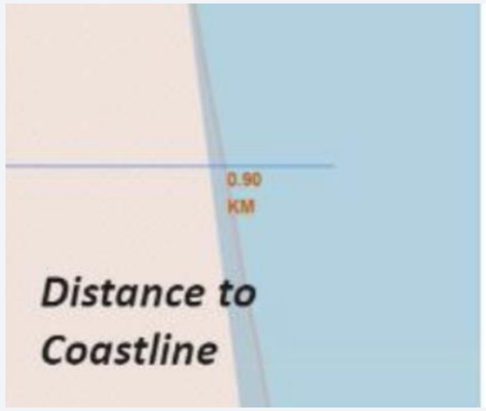
- California launch sites.

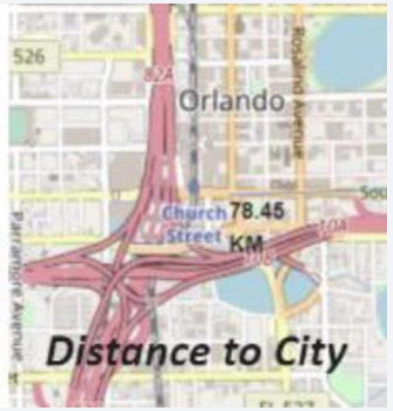
Launch site distance to landmarks





7
CCAFS SLC-40
26
CCAFS LC-40
Distance to coast

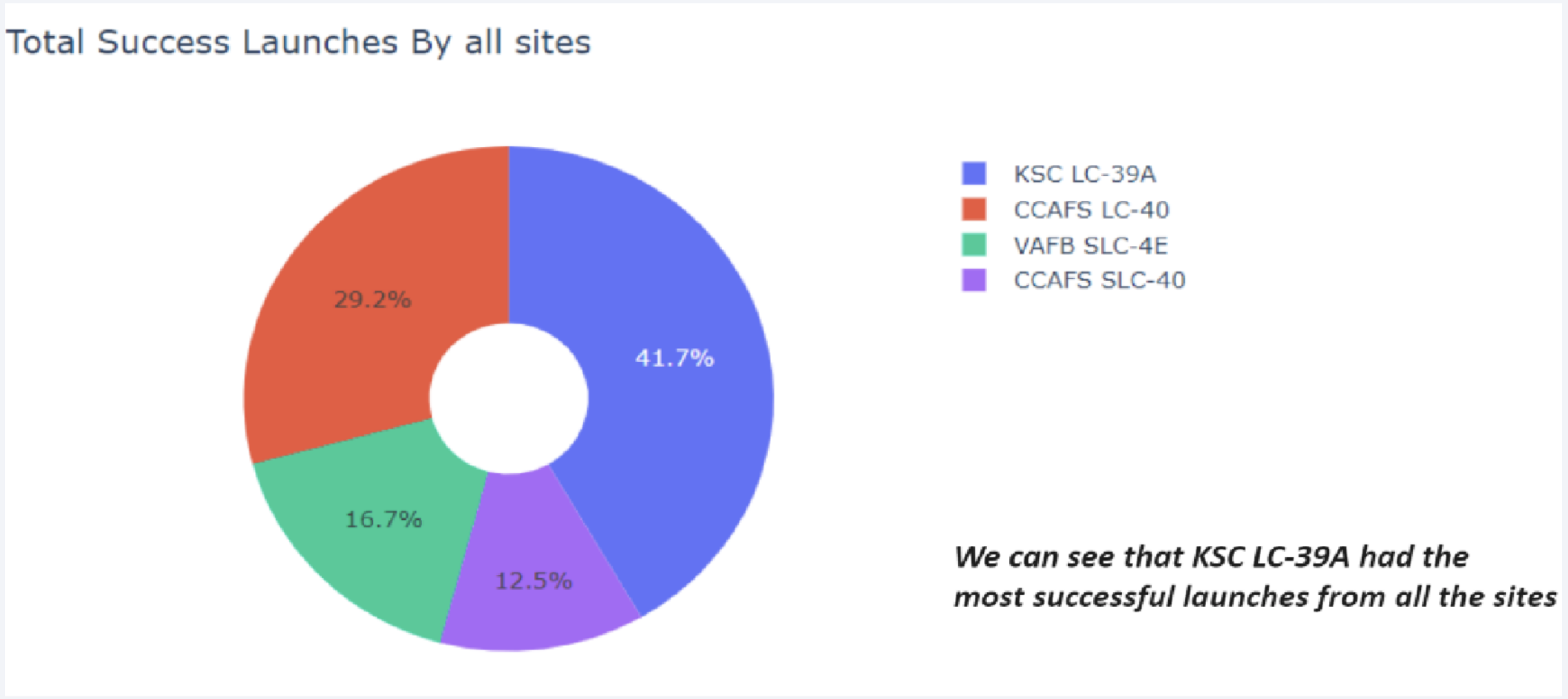




Section 4

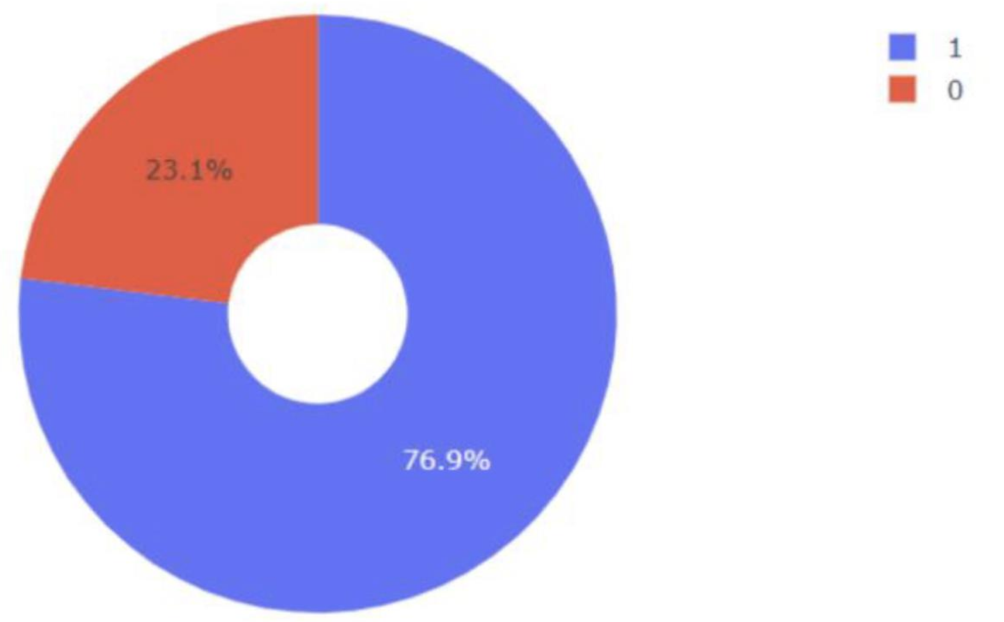
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch



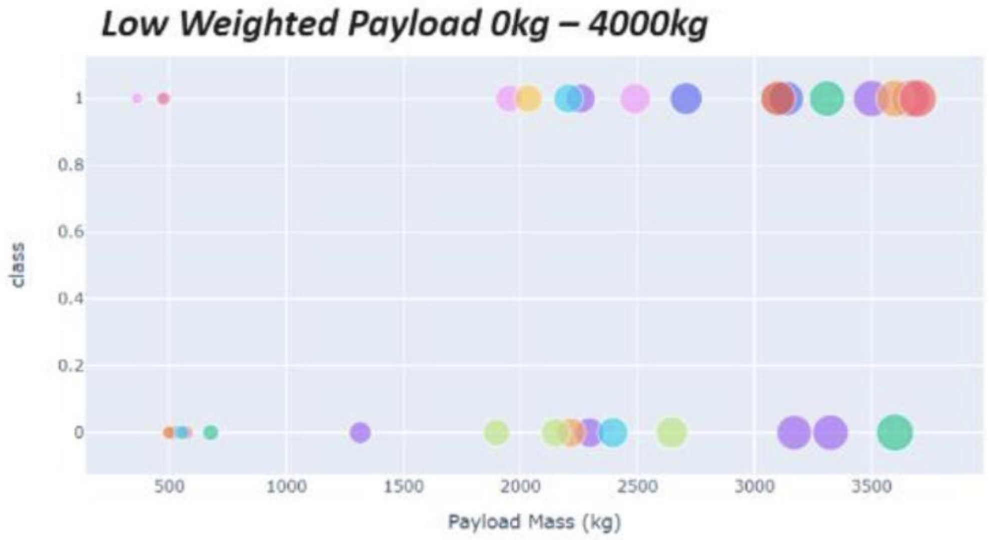
We can see that KSC LC-39A had the most successful launches from all the sites

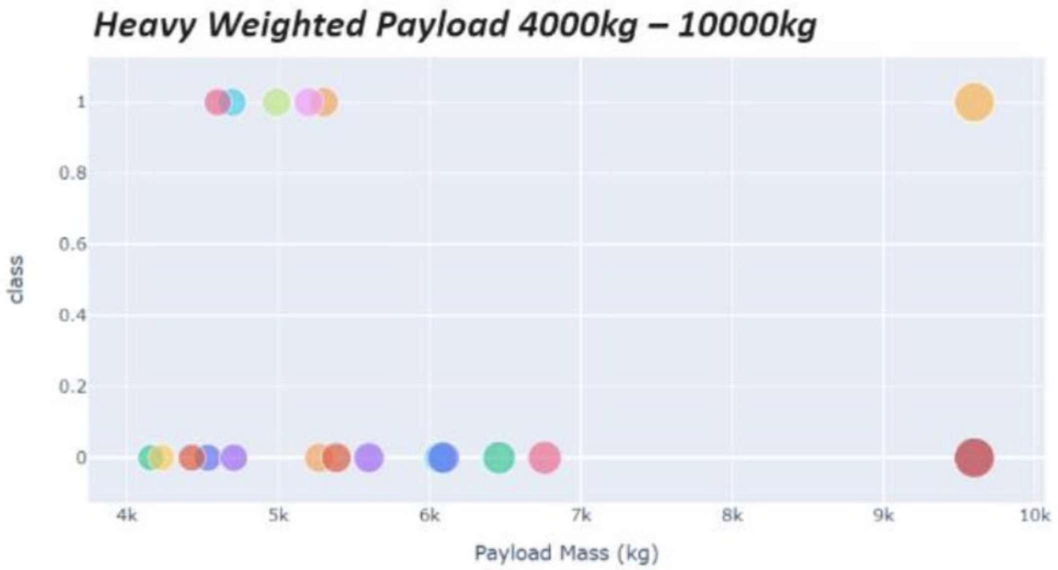
Pie chart showing the launch site with the highest launch success ratio



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

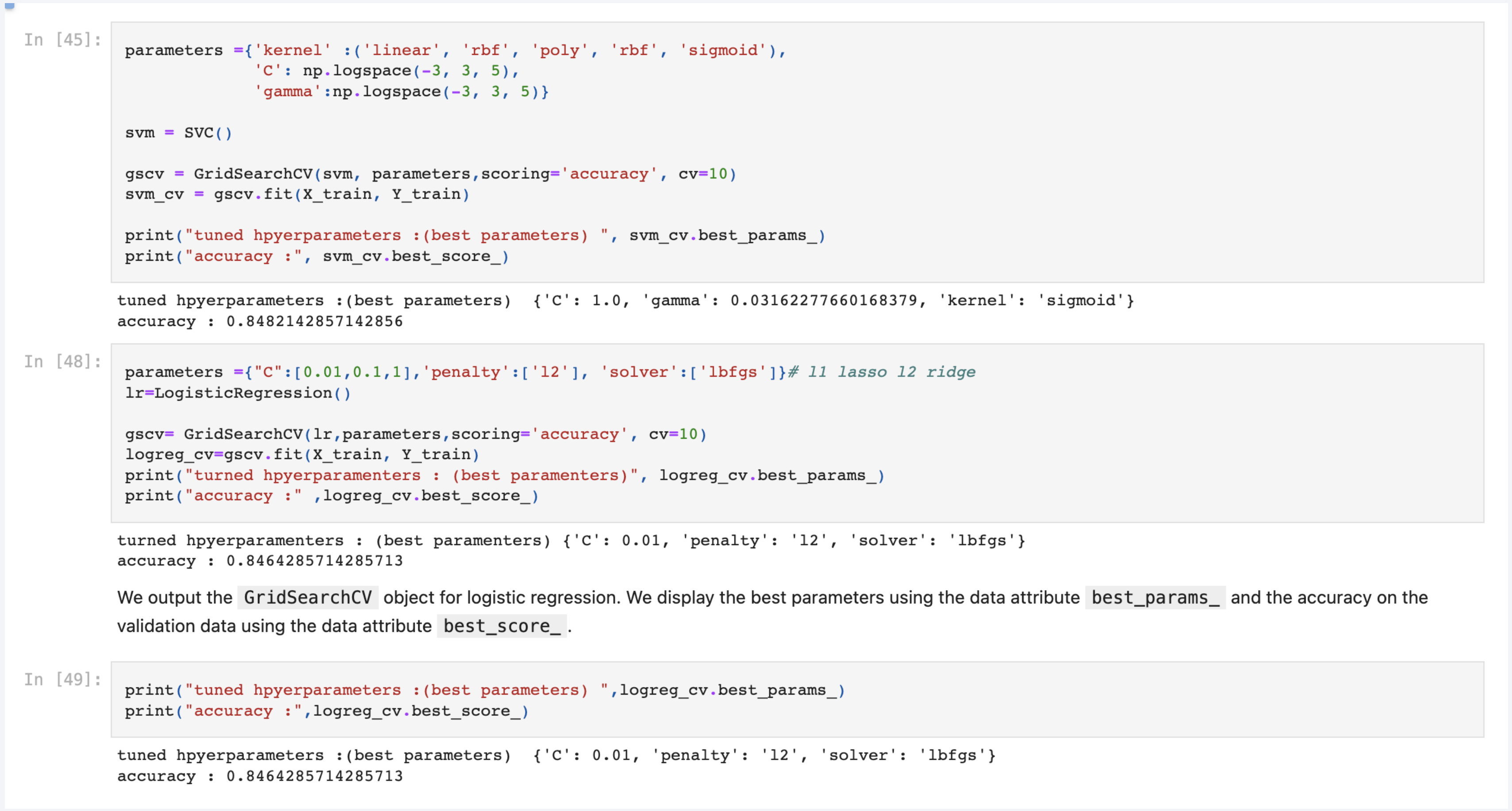




Section 5

Predictive Analysis (Classification)

Classification Accuracy



```
In [45]: parameters ={'kernel' :('linear', 'rbf', 'poly', 'rbf', 'sigmoid'),
                      'C': np.logspace(-3, 3, 5),
                      'gamma':np.logspace(-3, 3, 5)}

         svm = SVC()

         gscv = GridSearchCV(svm, parameters,scoring='accuracy', cv=10)
         svm_cv = gscv.fit(X_train, Y_train)

         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
```

```
In [48]: parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
         lr=LogisticRegression()

         gscv= GridSearchCV(lr,parameters,scoring='accuracy', cv=10)
         logreg_cv=gscv.fit(X_train, Y_train)
         print("turned hpyerparamenters : (best paramenters)", logreg_cv.best_params_)
         print("accuracy :" ,logreg_cv.best_score_)
```

```
turned hpyerparamenters : (best paramenters) {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8464285714285713
```

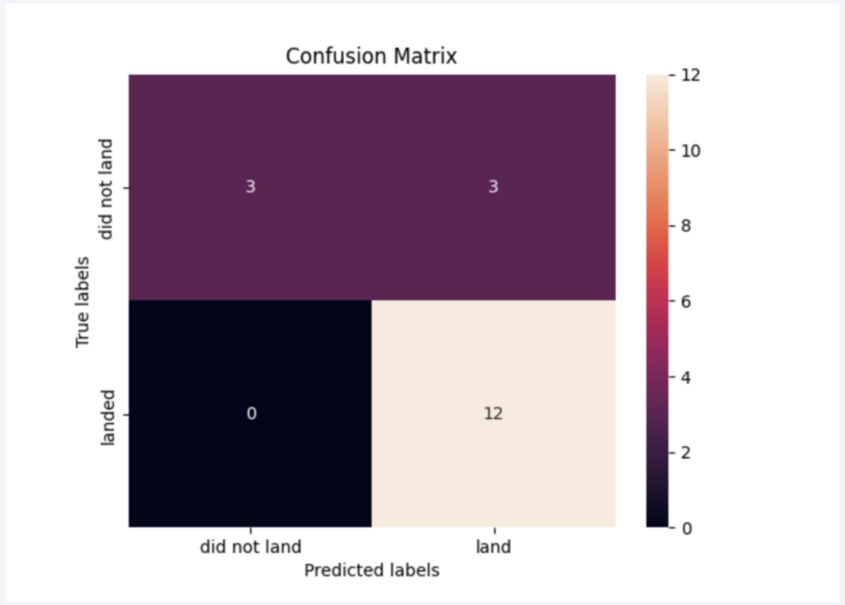
We output the `GridSearchCV` object for logistic regression. We display the best parameters using the data attribute `best_params_` and the accuracy on the validation data using the data attribute `best_score_`.

```
In [49]: print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
         print("accuracy :",logreg_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8464285714285713
```

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.



Conclusions

We can conclude that:

- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase in 2013 till 2020.
- Orbits EA-L1, GEO, HEO, SSO, VLEO had the most success rate.
- KSC LC-39A had the most successful launches of any sites.
- The Decision tree classifier is the best machine learning algorithm for this task.

Thank you!