IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Ratheesh K
23-Aug-2022

Outline

Executive Summary

- Summary of methodologies
  - Data Collection via API, Web Scraping
  - Exploratory Data Analysis (EDA) with Data Visualization
  - EDA with SQL
  - Interactive Map with Folium
  - Dashboards with Plotly Dash
  - Predictive Analysis
- Summary of all results
  - Exploratory Data Analysis results
  - Interactive maps and dashboard
  - Predictive results

Introduction

- Project background and context

The aim of this project is to predict if the Falcon 9 first stage will successfully land. SpaceX says on its website that the Falcon 9 rocket launch cost 62 million dollars. Other providers cost upward of 165 million dollars each. The price difference is explained by the fact that SpaceX can reuse the first stage. By determining if the stage will land, we can determine the cost of a launch. This information is interesting for another company if it wants to compete with SpaceX for a rocket launch.

- Problems you want to find answers

  - What are the main characteristics of a successful or failed landing?
  - What are the effects of each relationship of the rocket variables on the success or failure of a landing?
  - What are the conditions which will allow space X to achieve the best landing success rate?

Section 1

Methodology

Methodology

Executive Summary

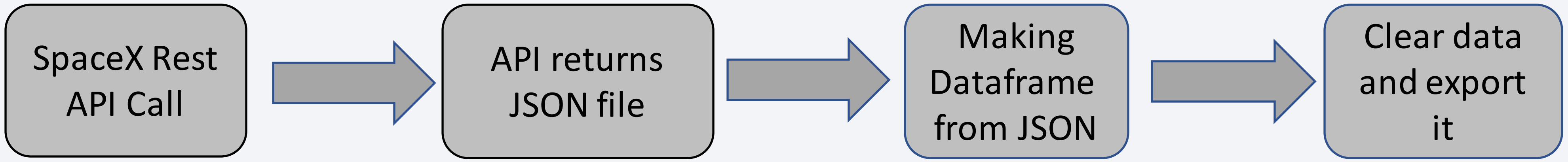
- Data collection methodology:
    - SpaceX REST API
    - Web Scraping from Wikipedia
- Perform data wrangling
    - Dropping unnecessary columns
    - One Hot Encoding for Classification models
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
    - How to build, tune, evaluate classification models

Data Collection

- Datasets are collected from Rest SpaceX API and webscrapping Wikipedia

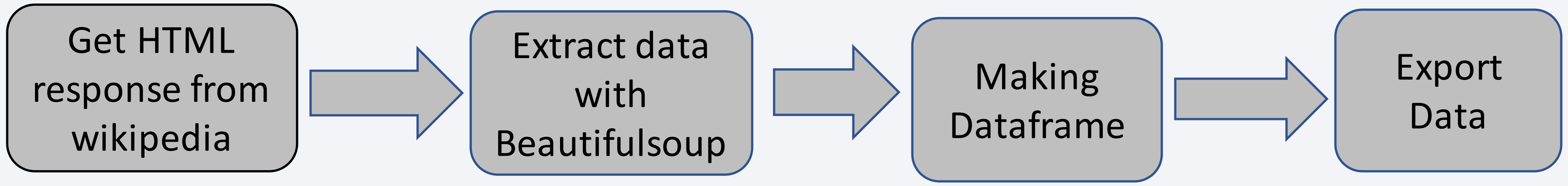
  > The information obtained by the API are rocket launches, payload information.

    > The SpaceX REST API URL is api.spacexdata.com/v4/



- The information obtained by the webscraping of wikipedia are launches, landing, payload information

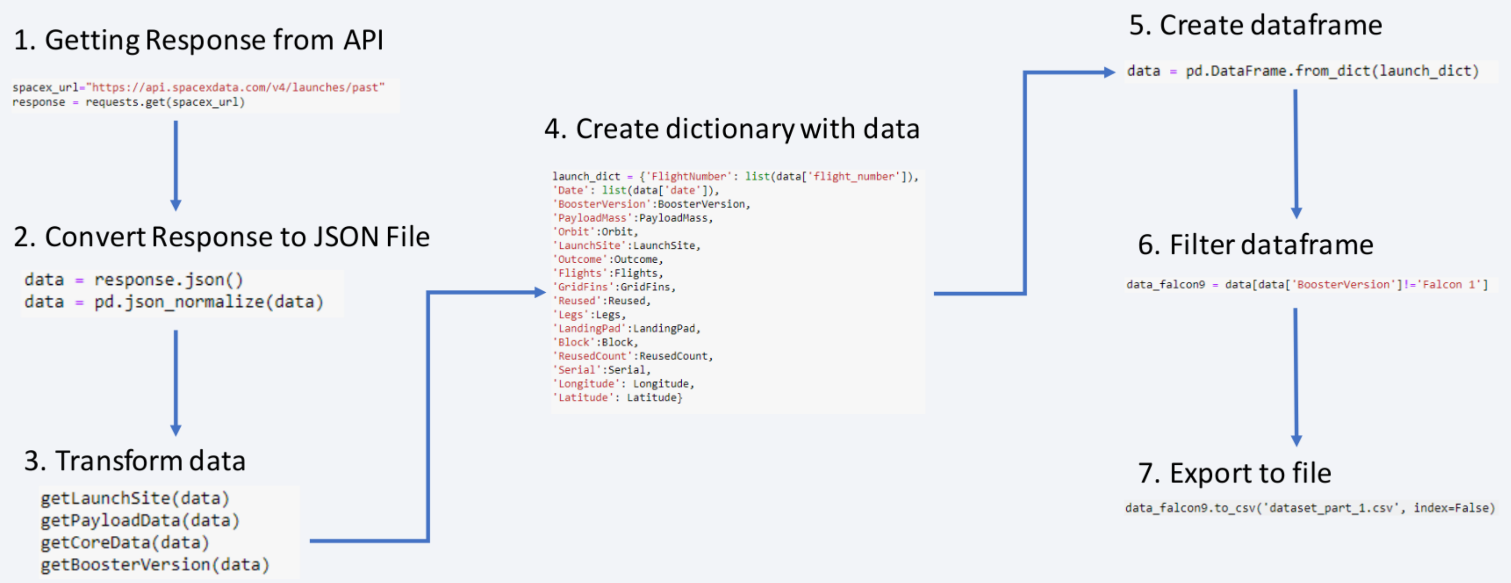
  > URL is https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922


Get HTML response from wikipedia
Extract data with Beautifulsoup
Making Dataframe
Export Data

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

[Link to the note book]()



1. Getting Response from API

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url)
```

2. Convert Response to JSON File

```
data = response.json()
data = pd.json_normalize(data)
```

3. Transform data

```
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
getBoosterVersion(data)
```

4. Create dictionary with data

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

5. Create dataframe

```
data = pd.DataFrame.from_dict(launch_dict)
```

6. Filter dataframe

```
data_falcon9 = data[data['BoosterVersion']!='Falcon 1']
```

7. Export to file

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

Data Collection - Scraping

1. Getting Response from HTML

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

2. Create BeautifulSoup Object

```
soup = BeautifulSoup(response.text, "html5lib")
```

3. Find all tables

```
html_tables = soup.findAll('table')
```

4. Get column names

```
for th in first_launch_table.find_all('th'):
    name = extract_column_from_header(th)
    if name is not None and len(name) > 0 :
        column_names.append(name)
```

5. Create dictionary

```
launch_dict= dict.fromkeys(column_names)

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

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

6. Add data to keys

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is a
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.stri
                flag=flight_number.isdigit()
```

See notebook for the rest of code

7. Create dataframe from dictionary

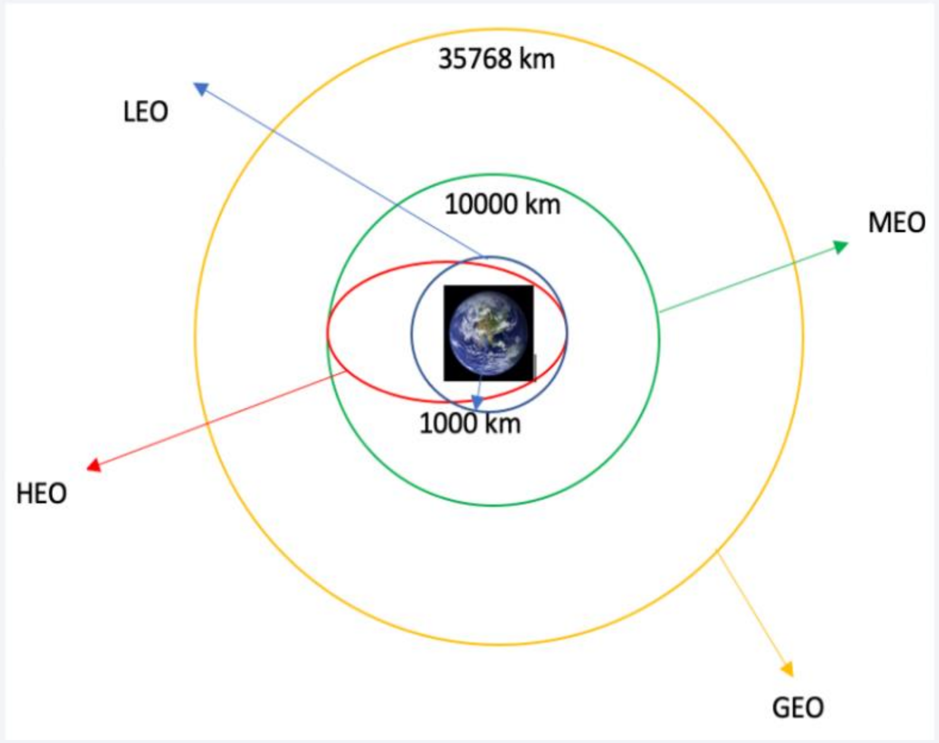
```
df=pd.DataFrame(launch_dict)
```

8. Export to file

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

Link to the notebook

Data Wrangling

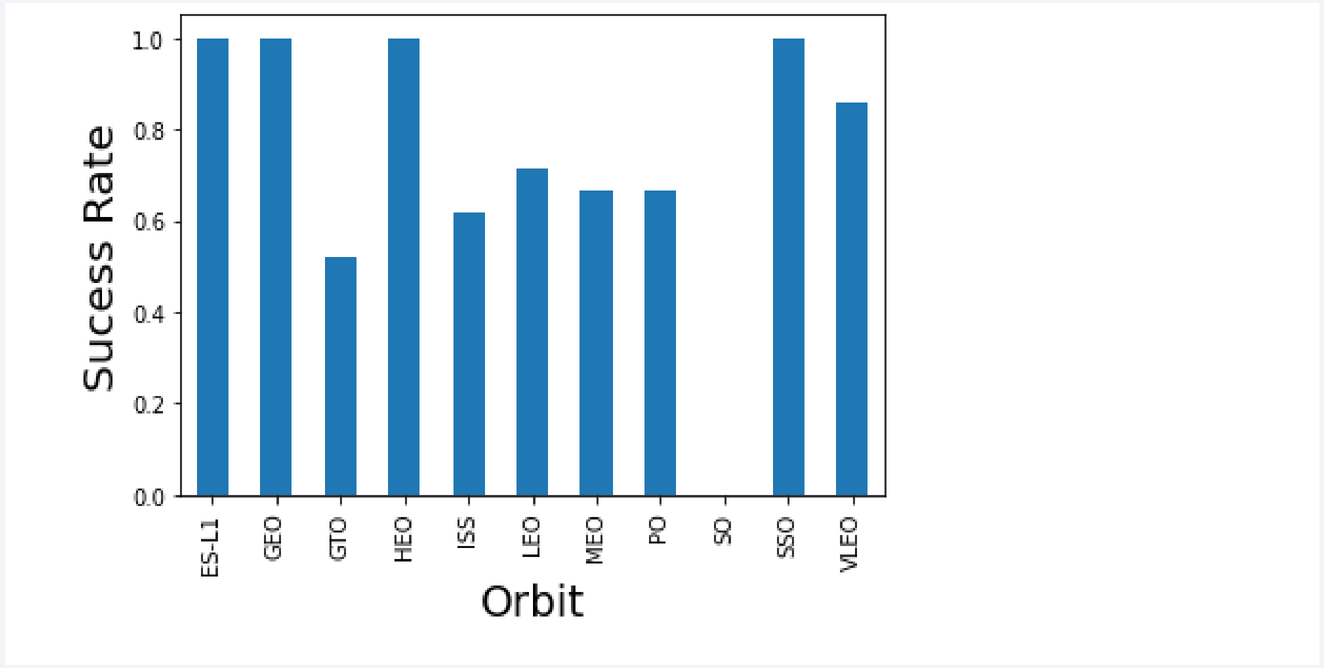


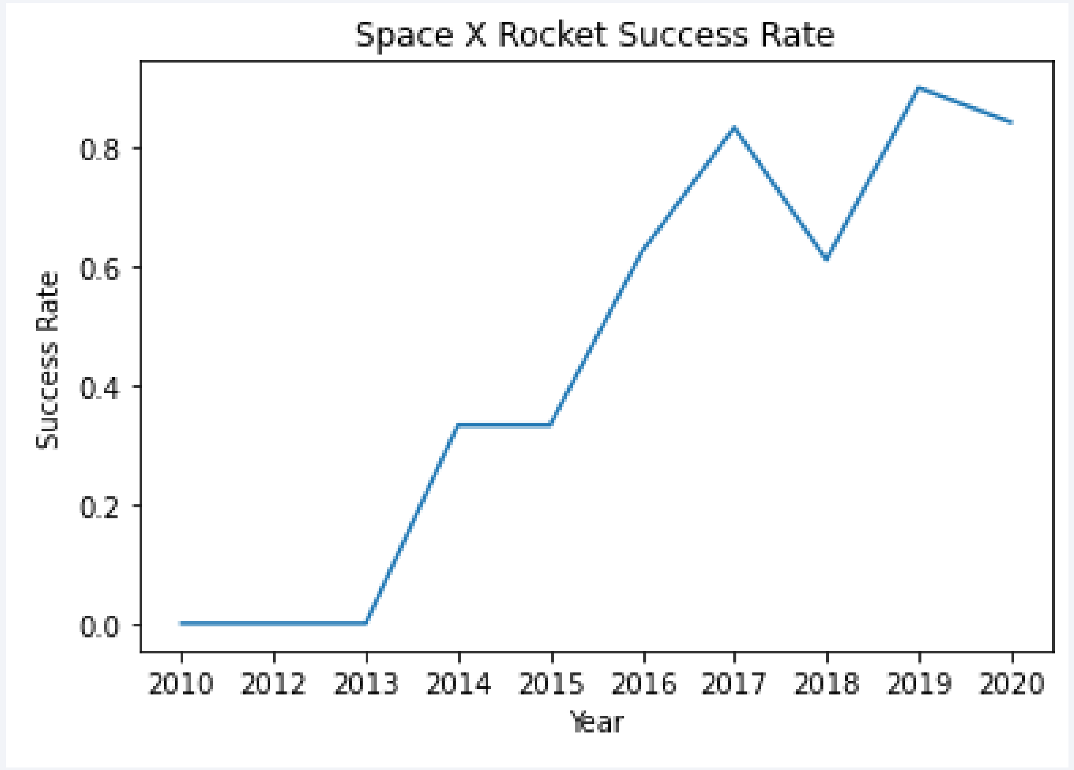
- We perform exploratory data analysis and determined the training labels
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the result to csv

Link to the notebook

EDA with Data Visualization

- We explored the data by visualizing the relationship between flight number and launch site, payload and launch site, success rate of each orbit type





Link to the notebook

EDA with SQL

- We loaded the SpaceX dataset into a Postger SQL database without leaving the jupyter notebook.
- We applied EDA with SQL to get insight from the data. We wrote queries to find out from instance:
  - The name of unique launch sites in the space mission.
  - The total payload mass carried by booster 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 outcome in drone ship, there booster version and launch site names.

Link to the notebook

Build an Interactive Map with Folium

- We marked all launch sites, and added map objects such as markers , circles , line to mark the success or failure if launches for each site on the folium map.
- We assigned the feature launch outcome (failure or success) to class 0 and 1. ie., 0 for failure and 1 for success
- Using the color-labeled marker cluster, we identified which launch site have relatively high success rate.
- We calculated the distance between a launch site to its proximities. We answered some questions. 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 creating sites
- We plotted scatter graph showing the relationship with Outcome and payload Mass (kg) for the different booster version.

Predictive Analysis (Classification)

- We loaded the data using numpy and pandas , transforming the data , split our data into training and testing .
- We built different machine learning models and turn 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.

[Link to the notebook]

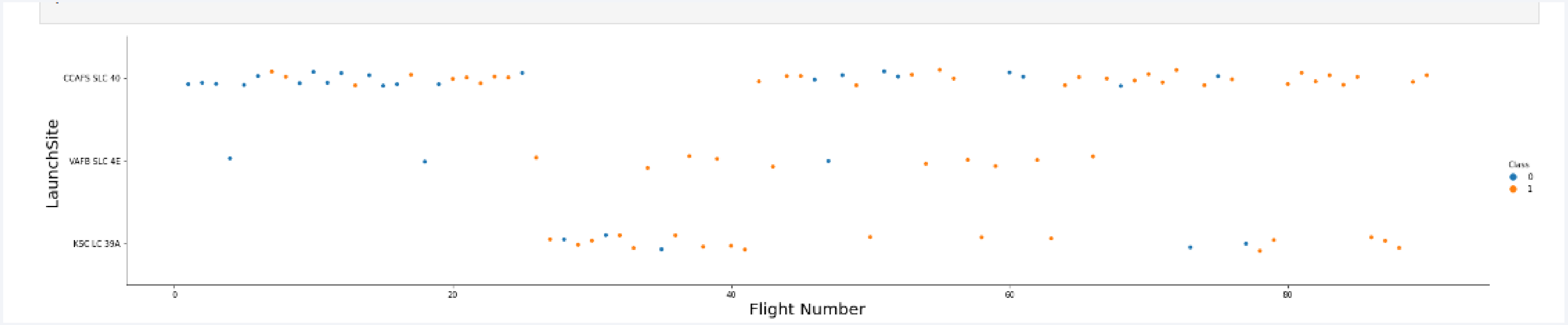
Results

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

Section 2

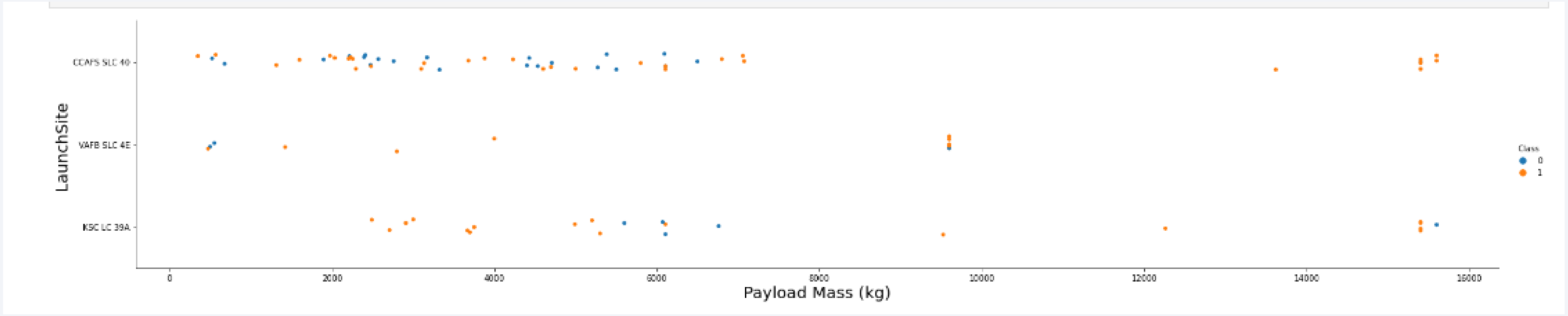
Insights drawn from EDA

Flight Number vs. Launch Site



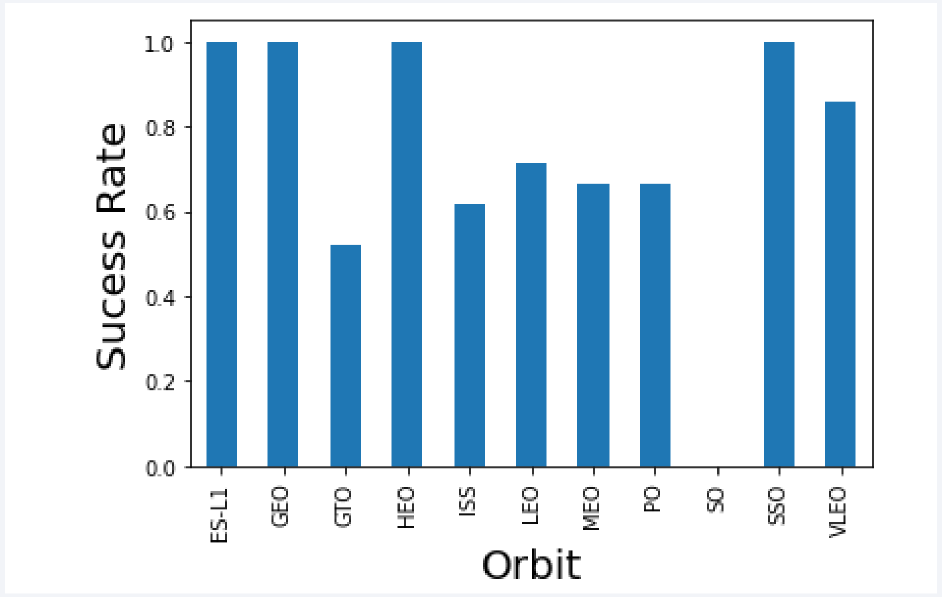
We observe that , for each site, the success rate is increasing

Payload vs. Launch Site



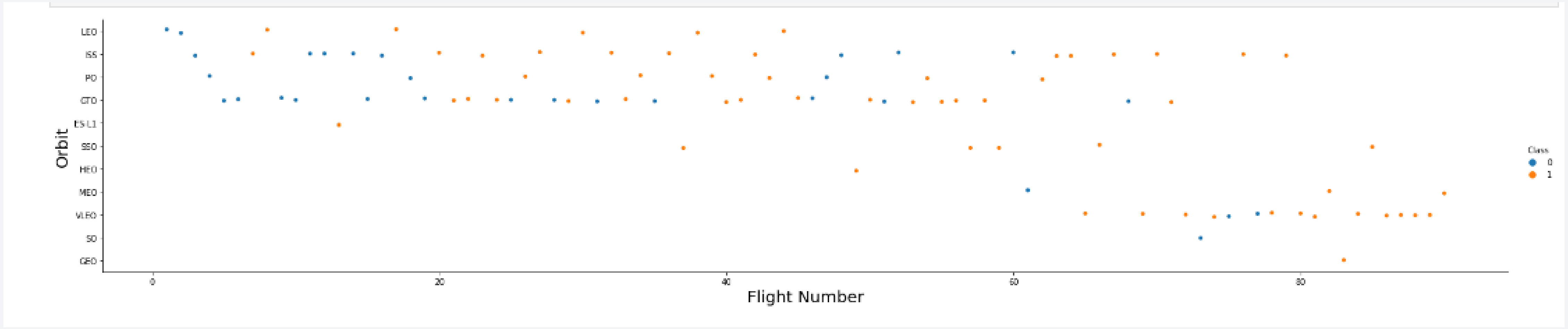
Depending on the launch site, a heavier payload may be a consideration for a successful landing. On the other hand, a too heavy payload can make a landing fail.

Success Rate vs. Orbit Type



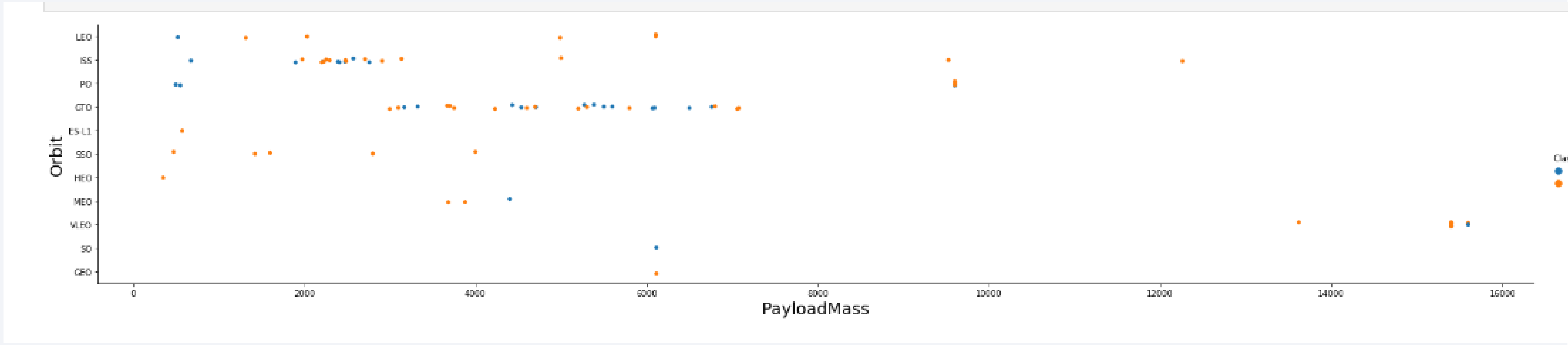
With this plot, we can see success rate for different orbit types. We note that ES-L1, GEO, HEO, SSO have the best success rate.

Flight Number vs. Orbit Type



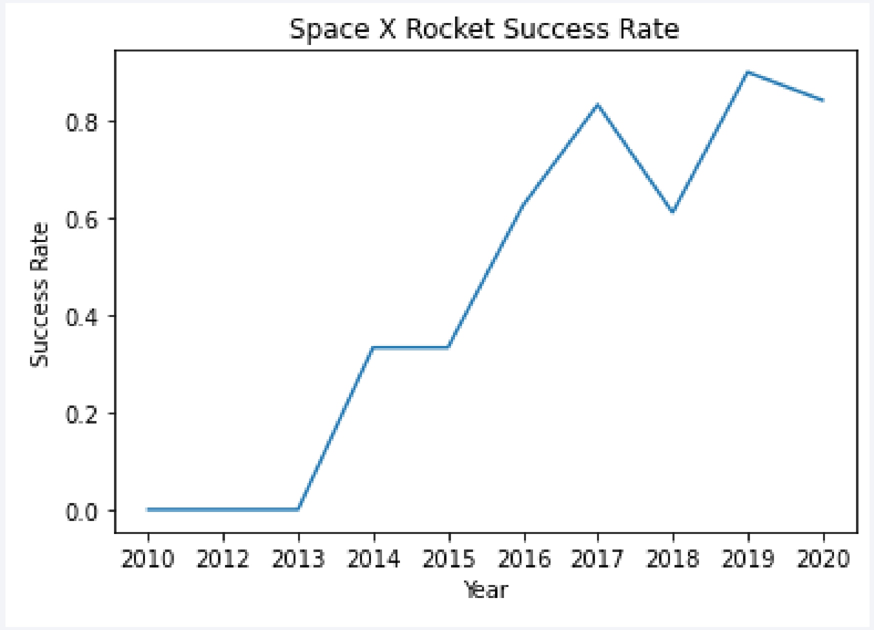
We notice that success rate increase with the number of flights for the LEO orbit . For some orbits like GTO, there is no relationship between the success rate and the number of flights. But we can suppose the high success rate of some orbit like SSO or HEO is due to the knowledge learned during former launches for other orbits.

Payload vs. Orbit Type



The weight of the payload can have a great influence on the success rate of the launches in certain orbits. For example , heavier payloads improve the success rate for the LEO orbit. Another finding is that decreasing the payload weight for a GTO orbit improves the success of a launch.

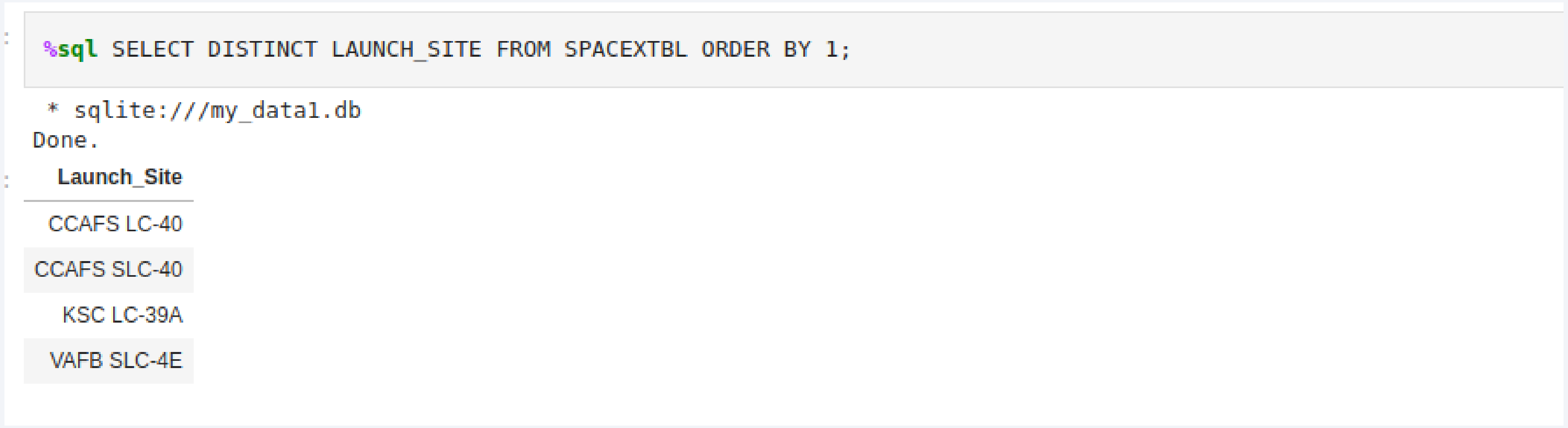
Launch Success Yearly Trend



Since 2013, we can see an increase in the SpaceX Rocket success rate.

All Launch Site Names

SQL Query and Result



```
%sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL ORDER BY 1;

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

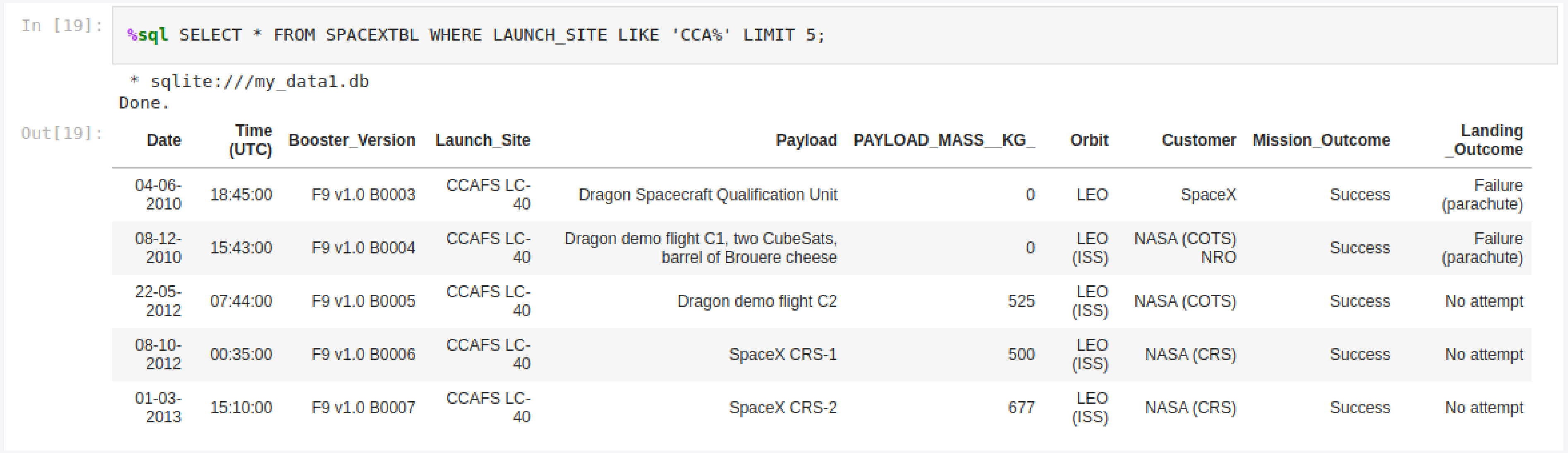
Launch_Site
CCAFS LC-40
CCAFS SLC-40
KSC LC-39A
VAFB SLC-4E

Explanation :

The use of DISTINCT in the query allows to remove duplicate LAUNCH_SITE.

Launch Site Names Begin with 'CCA'

SQL Query and Result



```
In [19]: %sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;

 * sqlite:///my_data1.db
Done.
Out[19]:
```

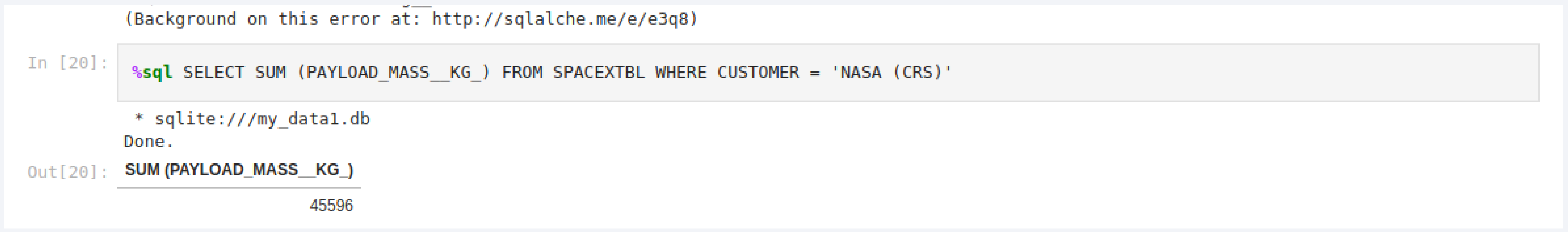
Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight 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

Explanation:

The WHERE clause followed by LIKE clause filters launch sites that contain the substring CCA . LIMIT 5 shows 5 records from filtering.

Total Payload Mass

SQL Query and Result



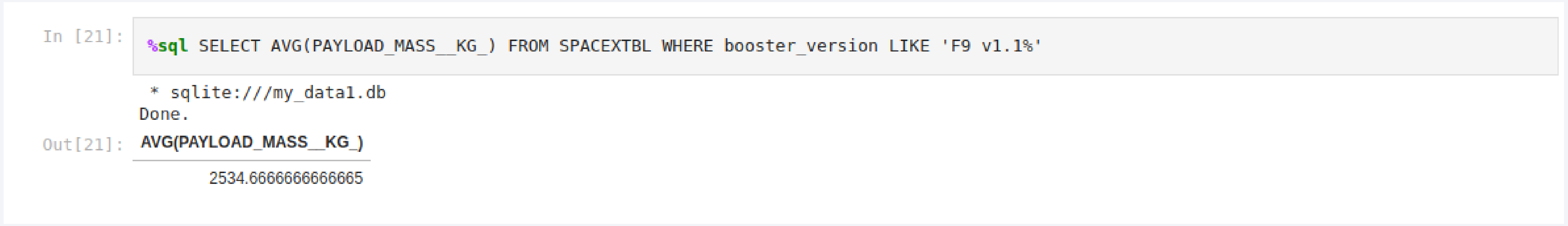
```
(Background on this error at: http://sqlalche.me/e/e3q8)

In [20]: %sql SELECT SUM (PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE CUSTOMER = 'NASA (CRS)'

 * sqlite:///my_data1.db
Done.
Out[20]: SUM (PAYLOAD_MASS__KG_)
                          45596
```

Explanation:

This query returns the sum of all payload masses where the customer is NASA (CRS).

Average Payload Mass by F9 v1.1

SQL Query and Result



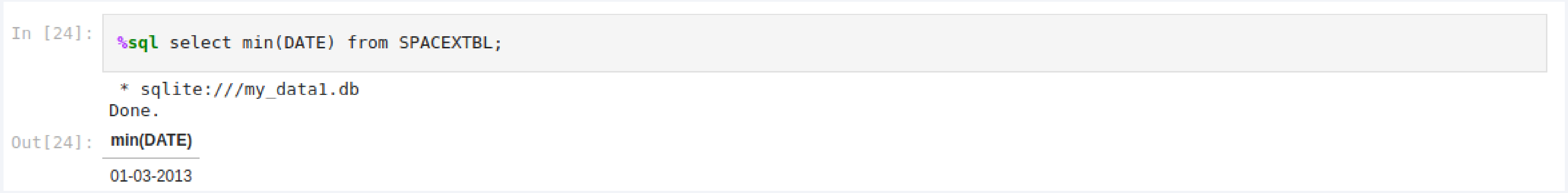
```
In [21]: %sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE booster_version LIKE 'F9 v1.1%'

 * sqlite:///my_data1.db
Done.

Out[21]: AVG(PAYLOAD_MASS__KG_)
                2534.6666666666665
```

Explanation:

This query returns the average of all payload masses where the booster version contains the substring F9 v1.1

First Successful Ground Landing Date

SQL Query and Result



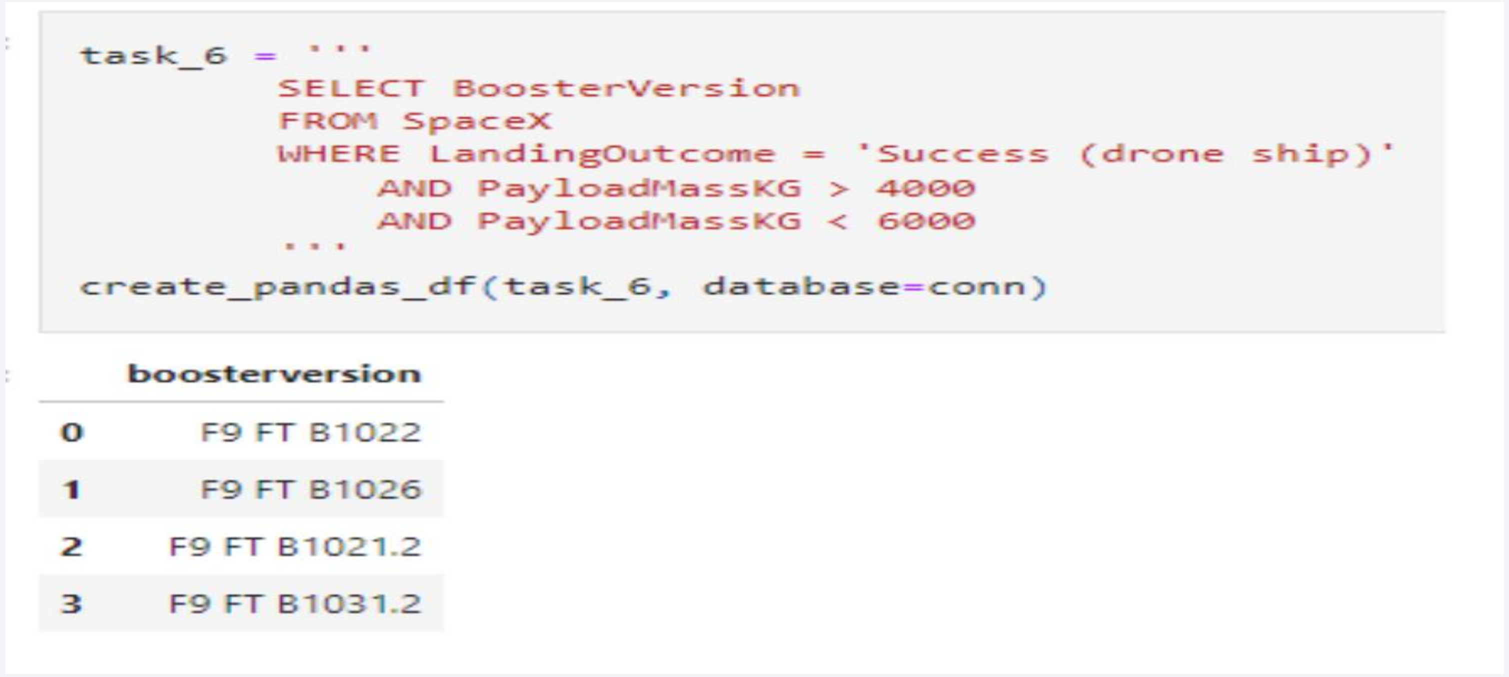
```
In [24]: %sql select min(DATE) from SPACEXTBL;

 * sqlite:///my_data1.db
Done.
Out[24]: min(DATE)
         01-03-2013
```

Explanation:

With this query we select the oldest date in the data

Successful Drone Ship Landing with Payload between 4000 and 6000

SQL Query and Result



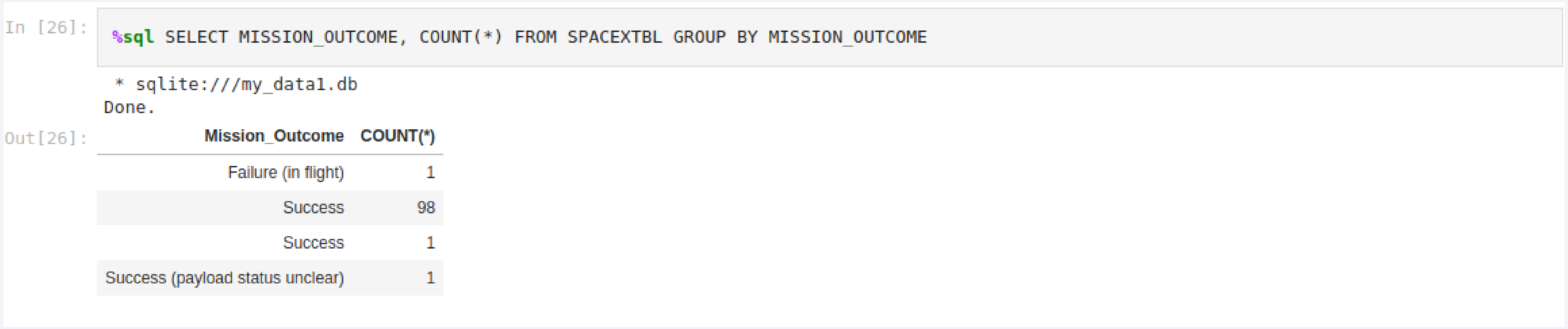
```
task_6 = '''
        SELECT BoosterVersion
        FROM SpaceX
        WHERE LandingOutcome = 'Success (drone ship)'
            AND PayloadMassKG > 4000
            AND PayloadMassKG < 6000
        '''
create_pandas_df(task_6, database=conn)
```

```
   boosterversion
0     F9 FT B1022
1     F9 FT B1026
2   F9 FT B1021.2
3   F9 FT B1031.2
```

Explanation:
This query returns the booster verson where landing was successful and payload mass in between 4000 and 6000 kg. The WHERE and AND clauses filter the dataset.

Total Number of Successful and Failure Mission Outcomes

SQL Query and Result



```
In [26]: %sql SELECT MISSION_OUTCOME, COUNT(*) FROM SPACEXTBL GROUP BY MISSION_OUTCOME

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

Out[26]:

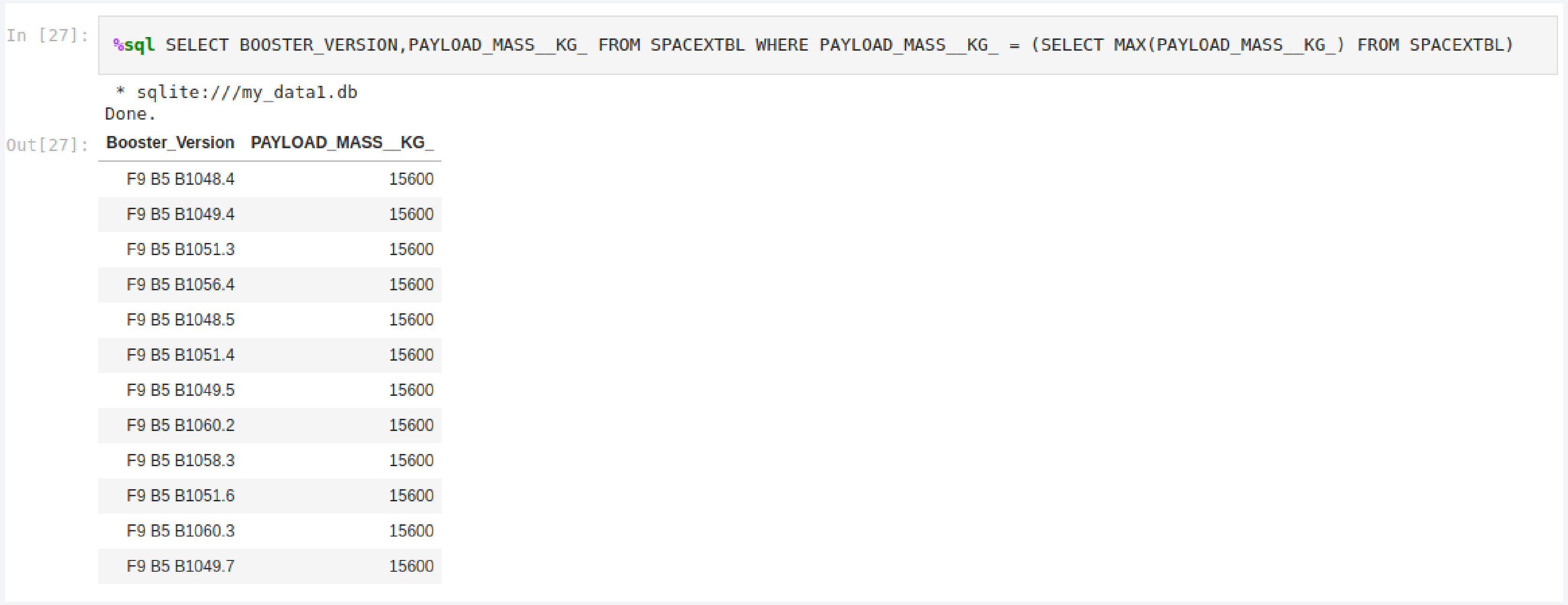
Mission_Outcome	COUNT(*)
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Explanation:

With this code we SELECT the mission outcome which shows the mission outcomes in a tables which lets us see the Failure and Success count

Boosters Carried Maximum Payload

SQL Query and Result



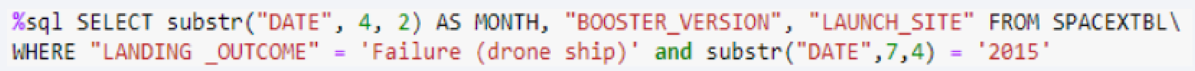
```
In [27]: %sql SELECT BOOSTER_VERSION,PAYLOAD_MASS__KG_ FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)

 * sqlite:///my_data1.db
Done.
Out[27]:
```

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

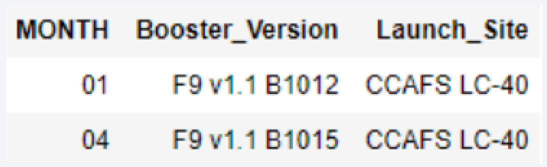
Explanation:
We used a subquery to filter data by returning only the heaviest payload mass with MAX function. The main query used sub query result and returns unique booster verson with the heaviest payload mass

2015 Launch Records

SQL Query and Result



```
%sql SELECT substr("DATE", 4, 2) AS MONTH, "BOOSTER_VERSION", "LAUNCH_SITE" FROM SPACEXTBL\
WHERE "LANDING _OUTCOME" = 'Failure (drone ship)' and substr("DATE",7,4) = '2015'
```



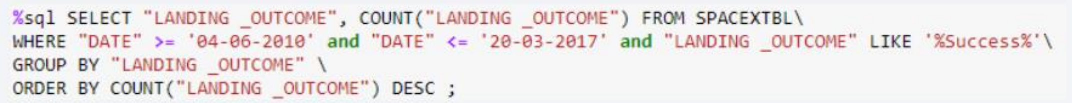
MONTH	Booster_Version	Launch_Site
01	F9 v1.1 B1012	CCAFS LC-40
04	F9 v1.1 B1015	CCAFS LC-40

Explanation:

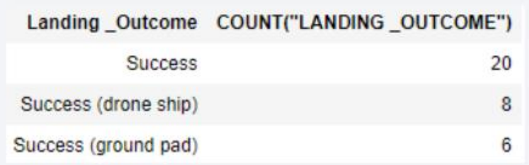
This query returns month, booster version, launch sites where landing where landing was unsuccessful and landing date took place in 2015. Substr function process date in order to take months or year . Substr(DATE, 4,2) shows months. Substr (DATE, 7,4) show year

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

SQL Query and Result



```
%sql SELECT "LANDING _OUTCOME", COUNT("LANDING _OUTCOME") FROM SPACEXTBL\
WHERE "DATE" >= '04-06-2010' and "DATE" <= '20-03-2017' and "LANDING _OUTCOME" LIKE '%Success%'\
GROUP BY "LANDING _OUTCOME" \
ORDER BY COUNT("LANDING _OUTCOME") DESC ;
```



Landing _Outcome	COUNT("LANDING _OUTCOME")
Success	20
Success (drone ship)	8
Success (ground pad)	6

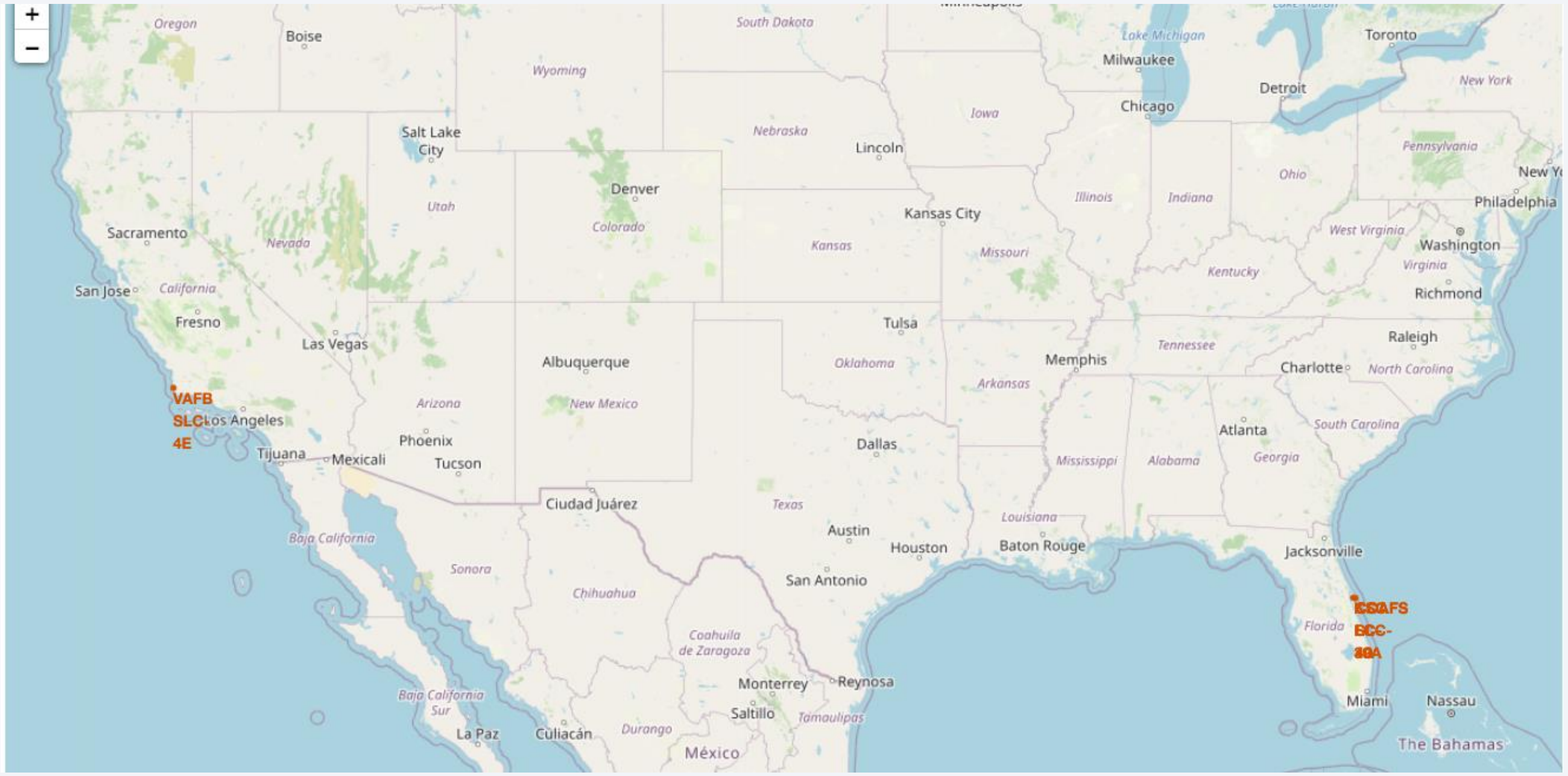
Explanation:

This query returns landing outcomes and their count where mission was successful and date is between 04/06/2010 and 20/03/2017. The GROUP BY clause groups results by landing outcome and ORDER BY COUNT DESC shows results in decreasing order.

Section 3

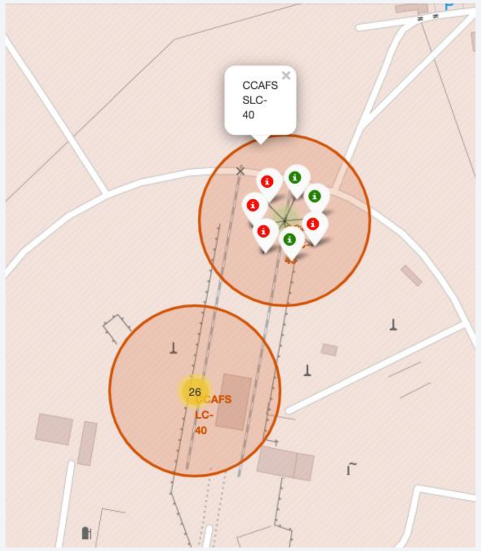
Launch Sites Proximities Analysis

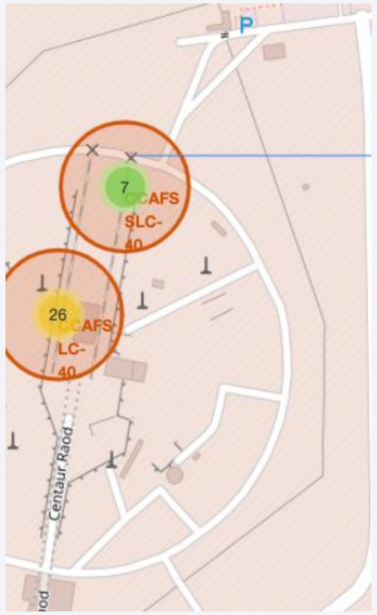
Folium map – Ground stations



We see that Space X launch sites are located on the coast of the United States

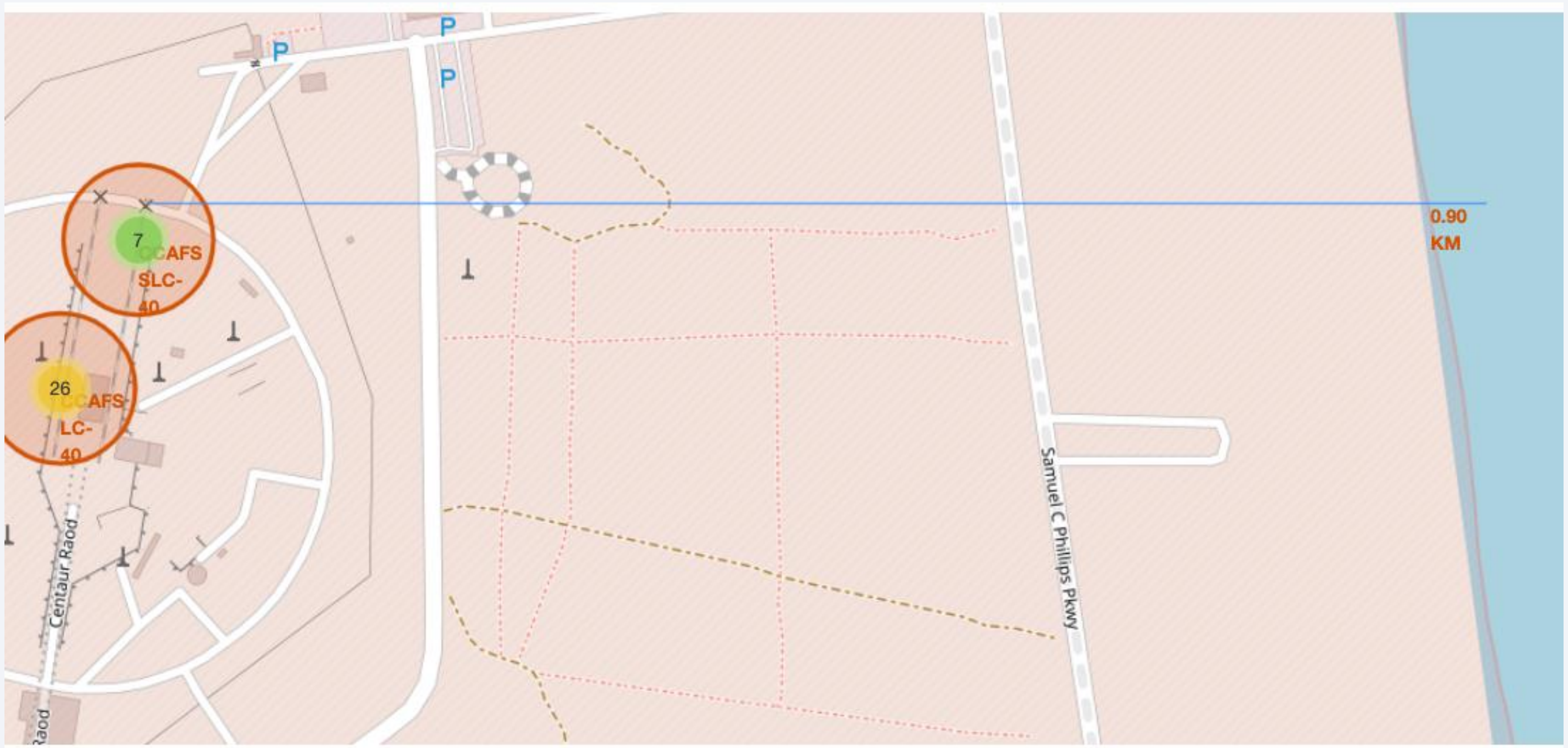
Markers showing launch sites with color labels





Green Marker shows successful Launches and Red Marker shows Failures

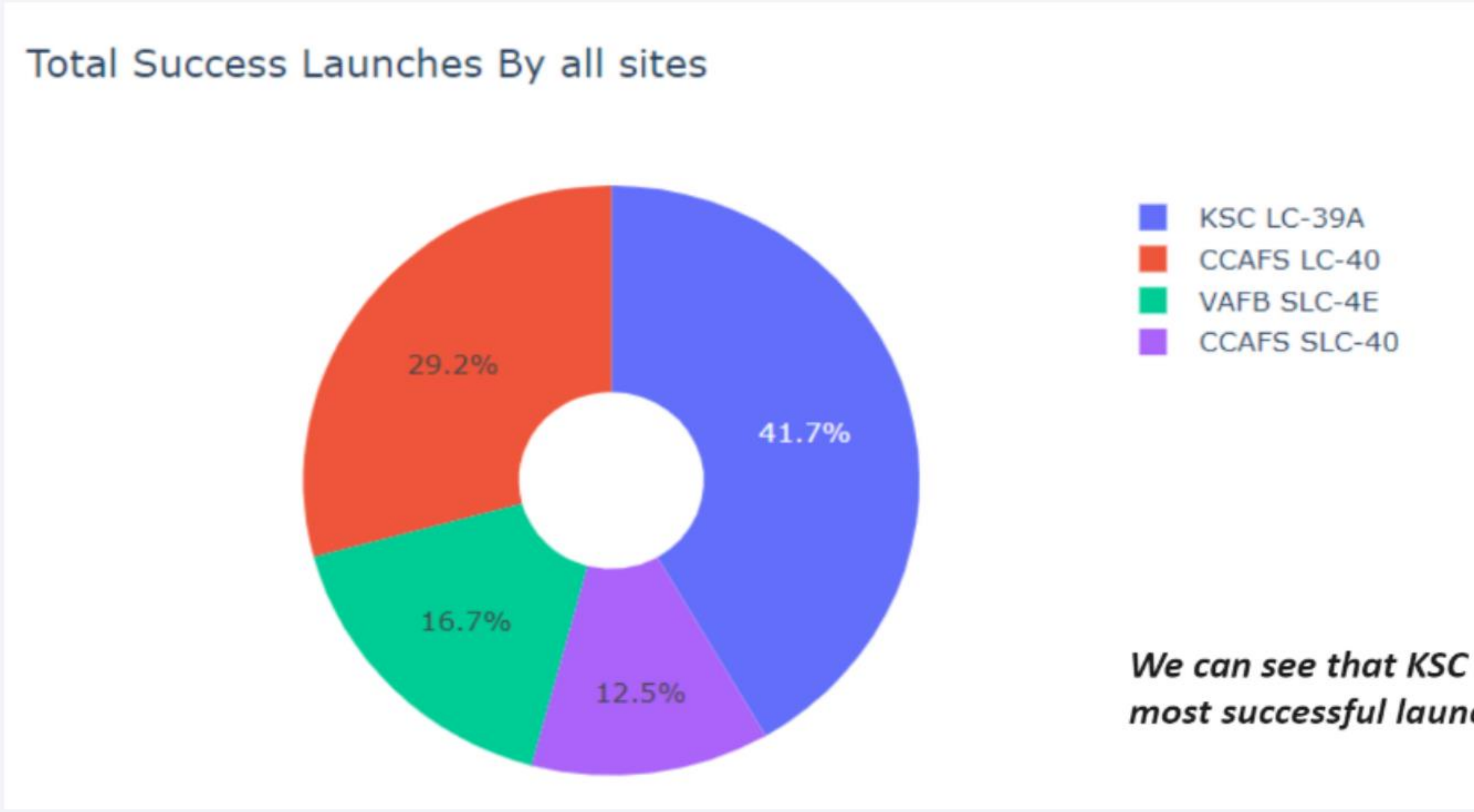
Launch Site distance to landmarks



Section 4

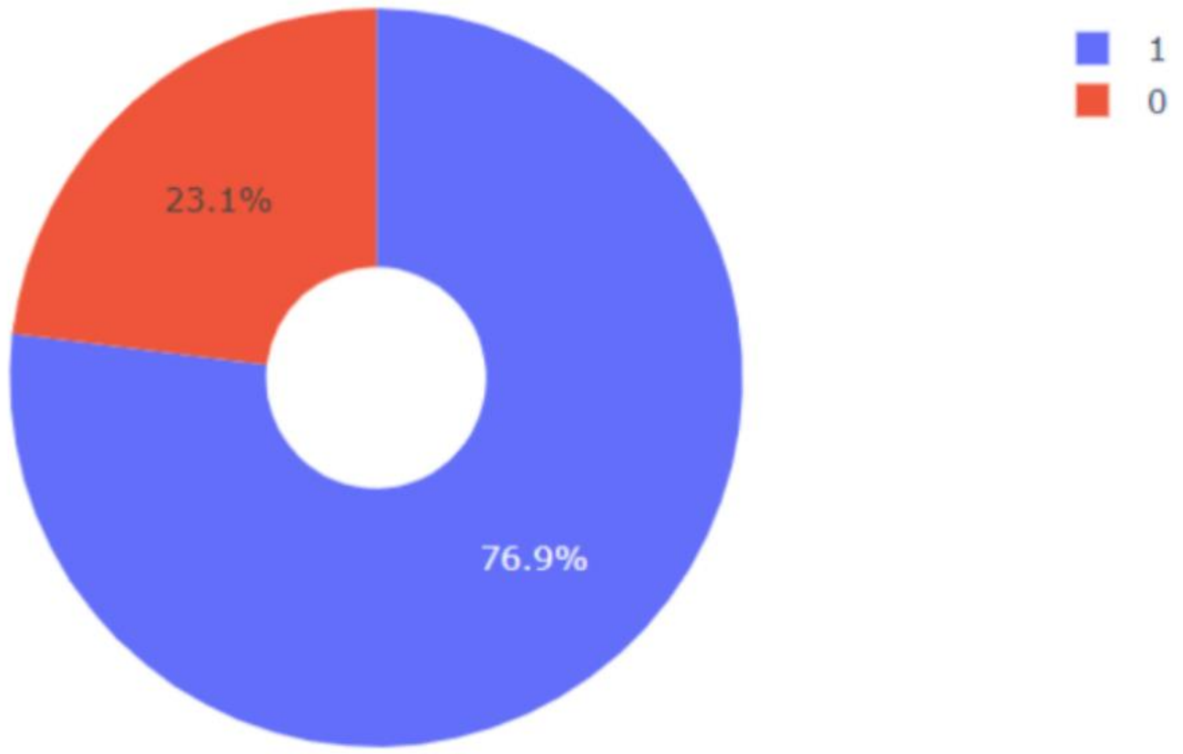
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



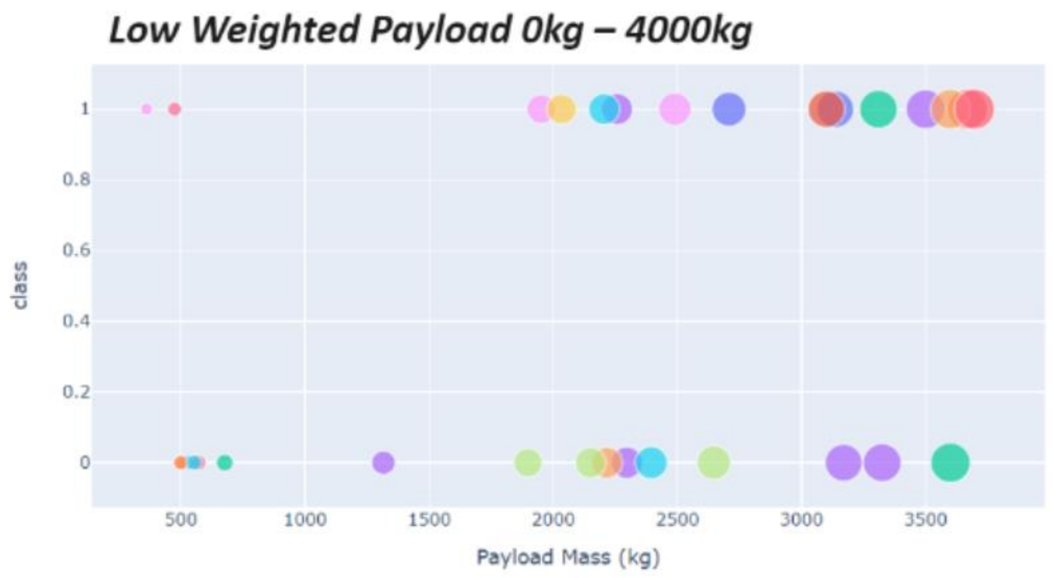
We can see that KSC LC-39A had the most successful launches from all the sites

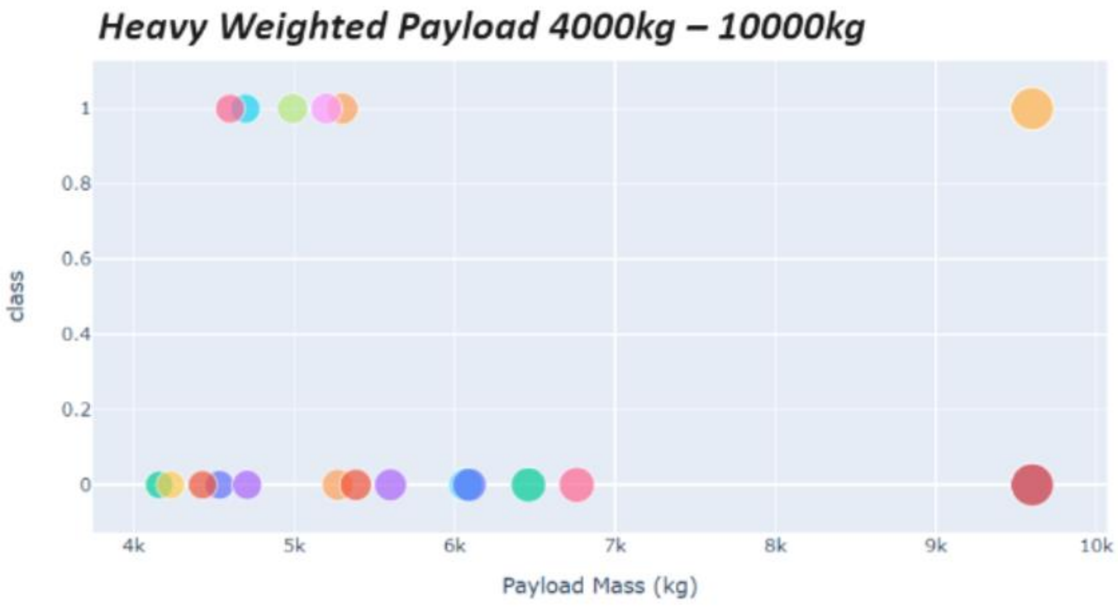
Pie chart showing the Launch site with the highest launch success ratio



KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider



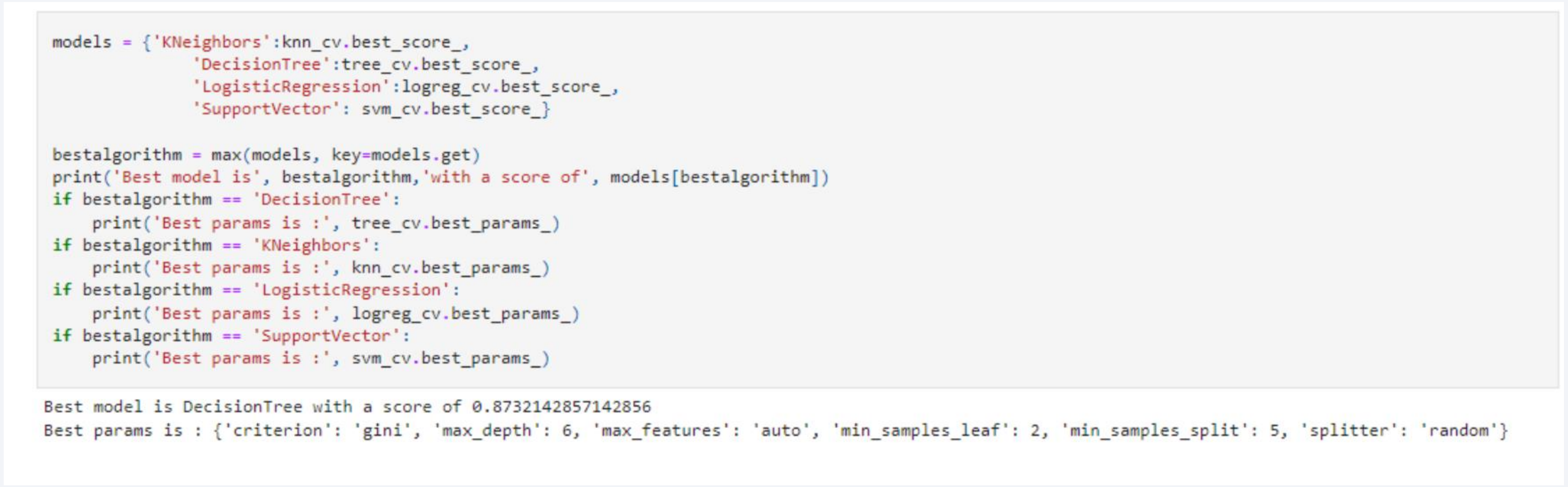


We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 5

Predictive Analysis (Classification)

Classification Accuracy

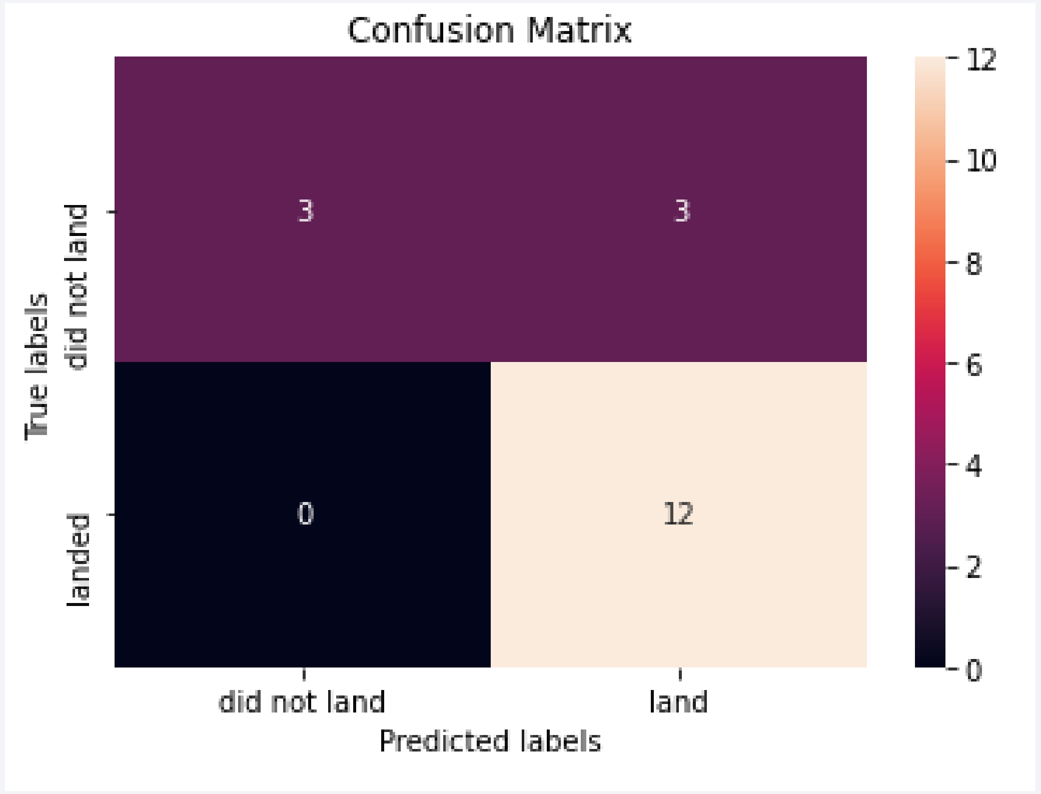


```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

The decision tree classifier is the model with the highest classification accuracy

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 ES-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!