IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Ashley Yong
October 14, 2021

Outline

Executive Summary

- **Methodologies: Data Collection**
    - Data wrangling
    - EDA with Data Visualization
    - EDA with SQL
    - Building an interactive map with Folium
    - Building a Dashboard with Plotly Dash
    - Predictive analysis (Classification and Machine Learning)
- **Summary of all results**
    - Findings from data analysis
    - Interactive analysis
    - Results from machine learning models and classification

Introduction

- **Project background and context**
    - In this capstone, we will predict if the Falcon 9 first stage will land successfully. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX 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 SpaceX for a rocket launch.
- **Problems you want to find answers**
    - Which variables affect success rate the most?
    - How should we label our supervised models?

Section 1

Methodology

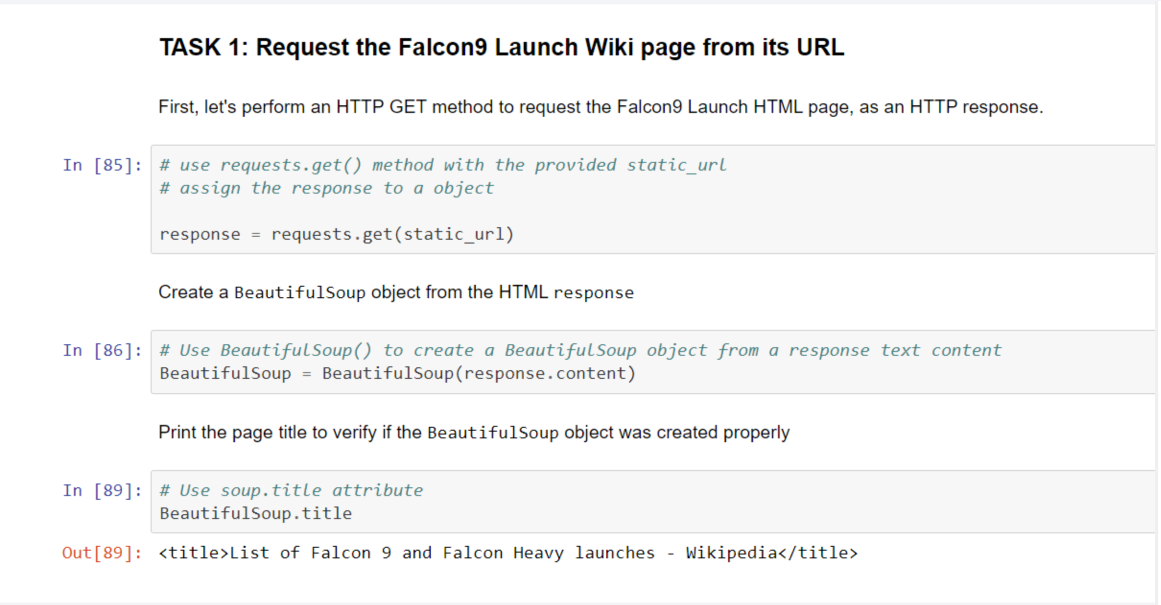
Methodology

Executive Summary

- Data collection methodology:
  - Collected from Wikipedia using web scraping methods in Python
  - REST API
- Perform data wrangling
  - Data cleaned to eliminate duplicates and separate failures from successes
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Used logistic regression, decision trees, KNN classification, and support vector machine methods to predict outcomes

Data Collection

- Data sets collected from Wikipedia using webscraping methods



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.

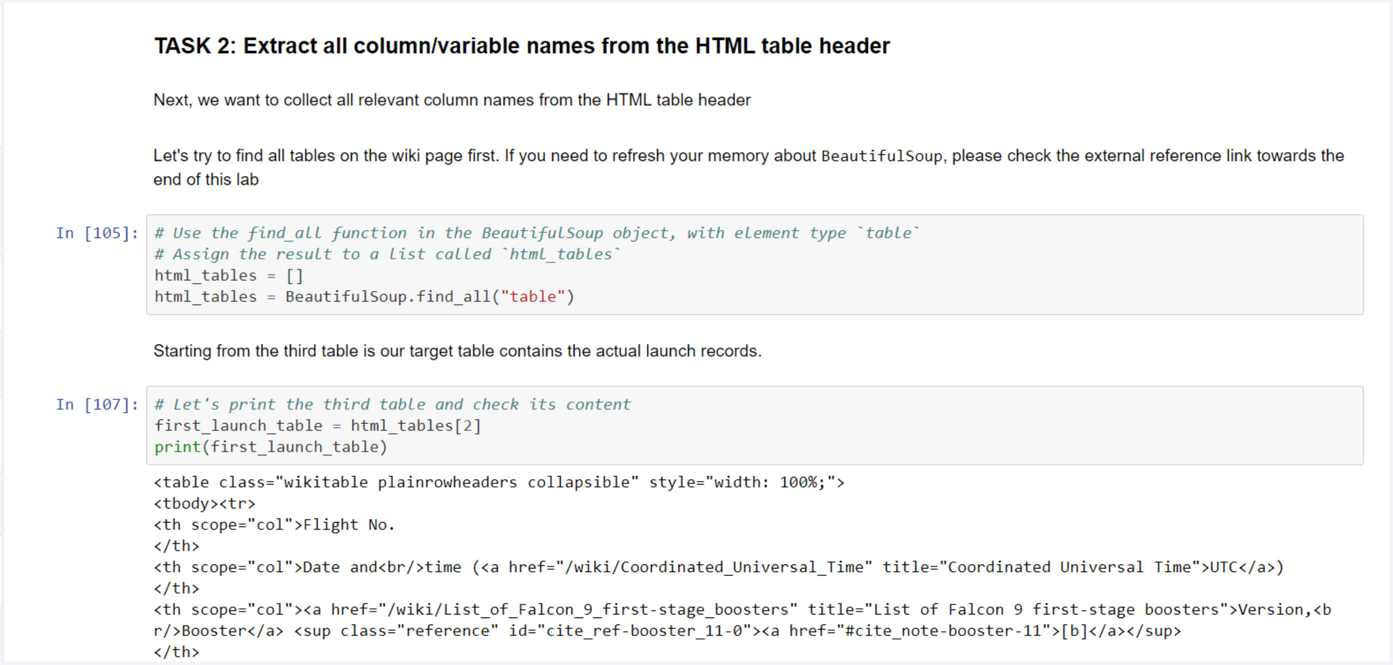
```
In [85]: # use requests.get() method with the provided static_url
         # assign the response to a object

         response = requests.get(static_url)
```

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

```
In [86]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
         BeautifulSoup = BeautifulSoup(response.content)
```

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

```
In [89]: # Use soup.title attribute
         BeautifulSoup.title

Out[89]: <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 [105]: # Use the find_all function in the BeautifulSoup object, with element type `table`
          # Assign the result to a list called `html_tables`
          html_tables = []
          html_tables = BeautifulSoup.find_all("table")
```

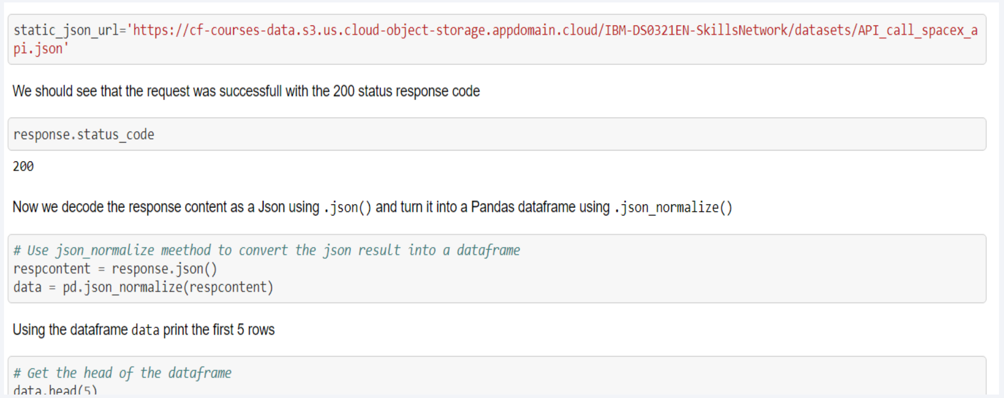
Starting from the third table is our target table contains the actual launch records.

```
In [107]: # Let's print the third table and check its content
          first_launch_table = html_tables[2]
          print(first_launch_table)

<table class="wikitable plainrowheaders collapsible" style="width: 100%;">
<tbody><tr>
<th scope="col">Flight No.
</th>
<th scope="col">Date and<br/>time (<a href="/wiki/Coordinated_Universal_Time" title="Coordinated Universal Time">UTC</a>)
</th>
<th scope="col"><a href="/wiki/List_of_Falcon_9_first-stage_boosters" title="List of Falcon 9 first-stage boosters">Version,<b
r/>Booster</a> <sup class="reference" id="cite_ref-booster_11-0"><a href="#cite_note-booster-11">[b]</a></sup>
</th>
```

Data Collection – SpaceX API

Step 1:

Request and parse the SpaceX launch data using the GET request



```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_a
pi.json'
```

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

```
response.status_code
```

```
200
```

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

```
# Use json_normalize meethod to convert the json result into a dataframe
respcontent = response.json()
data = pd.json_normalize(respcontent)
```

Using the dataframe `data` print the first 5 rows

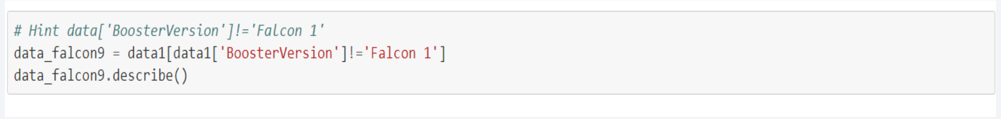
```
# Get the head of the dataframe
data.head(5)
```

Construct chosen variables into a dictionary and convert to subset of a dataframe

https://github.com/yamley/ibm-capstone/blob/master/Data%20Collection%20API%20Lab.ipynb

Step 2:

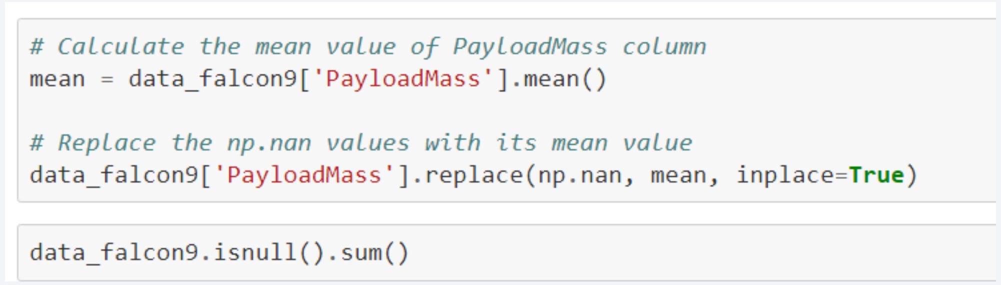
Filter dataframe to include only Falcon 9 launches



```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = data1[data1['BoosterVersion']!='Falcon 1']
data_falcon9.describe()
```

Step 3:

Eliminate or replace missing values



```
# Calculate the mean value of PayloadMass column
mean = data_falcon9['PayloadMass'].mean()

# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].replace(np.nan, mean, inplace=True)
```

```
data_falcon9.isnull().sum()
```

Data Collection - Scraping

Step 1:

Request the Falcon9 Launch Wiki page from its URL

Step 2:

Extract all column/variable names from the HTML table header

Step 3:

Create a dataframe by parsing the launch HTML tables

Use the BeautifulSoup plugin to extract data from web url, then extract the column names by filtering header tags within the html. Convert the parsed html tables into a dictionary, and then use pandas to transform into a dataframe.

https://github.com/yamley/ibm-capstone/blob/master/Data%20Collection%20with%20Web%20Scraping.ipynb

Data Wrangling

Step 1:

Calculate the number of launches on each site

Step 2:

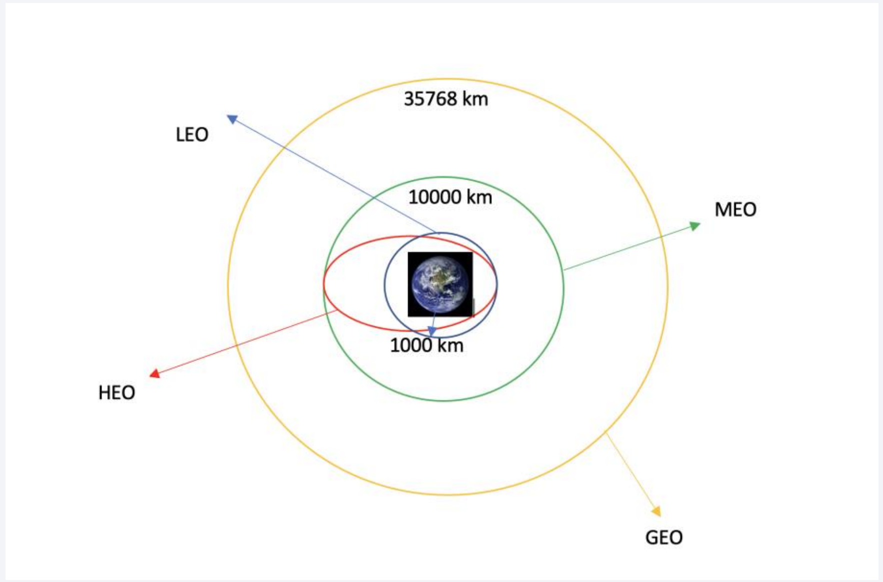
Calculate the number and occurrence of each orbit

Step 3:

Calculate the number and occurence of mission outcome per orbit type

Step 4:

Create a landing outcome label from Outcome column



Used value_counts() function to determined number of launches on each site, then used the same method to determine number and occurrences of each orbit. I then applied the same function to determined the amount of landing outcomes and made outcomes either 1 or 0 to more easily predict success or failures.

https://github.com/yamley/ibm-capstone/blob/master/Data%20Wrangling.ipynb

EDA with Data Visualization

Charts plotted: scatter plots, bar charts, and line chart to help visualize data

- Payload vs. Orbit Type
- Flight Number vs. Launch Site
- Payload vs. Launch Site
- Launch Success Yearly Trend
- Success Rate vs. Orbit Type

https://github.com/yamley/ibm-capstone/blob/master/EDA%20with%20Visualization.ipynb

EDA with SQL

SQL Queries

1. Unique launch sites in space mission
2. 5 records where launch site begins with the string 'CCA'
3. Total payload mass carried by boosters launched by NASA (CRS)
4. Avg payload mass carried by booster version F9 v1.1
5. Date when first successful landing outcome in ground pad was achieved
6. Names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
7. Total number of successful and failure mission outcomes
8. Names of the booster_versions which have carried the maximum payload mass. Use a subquery
9. Failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
10. 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

https://github.com/yamley/ibm-capstone/blob/master/EDA%20with%20SQL.ipynb

Predictive Analysis (Classification)

Step 1:

Create a NumPy array from column 'Class' in dataframe as variable Y

Step 2:

Standardize data in X and reassign to variable X

Step 3:

Split data into training and test sets

Step 4:

Use decision tree, logistic regression, and SVM methods with GridSearchCV objects to fit model

Step 5:

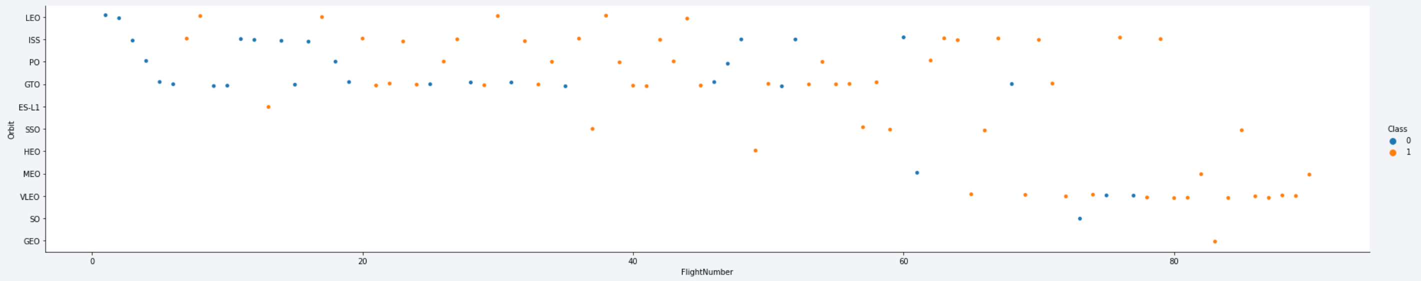
Find best parameters of models and evaluate success of models using score method

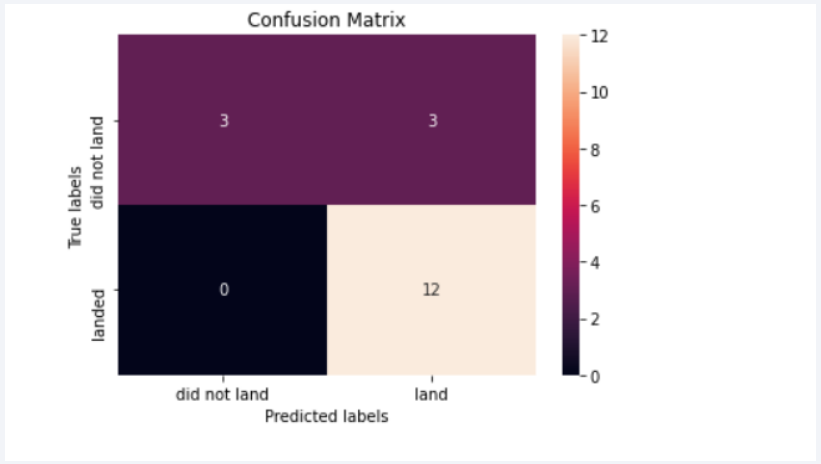
Performed exploratory Data Analysis and determined Training Labels in order to predict success of first stage. Used SVM, Classification Trees, and Logistic Regression when determining how to best fit data

https://github.com/yamley/ibm-capstone/blob/master/Machine%20Learning.ipynb

Results

- **Exploratory data analysis results:** Success rate since 2013 kept increasing till 2020
- Within the LEO orbit Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.





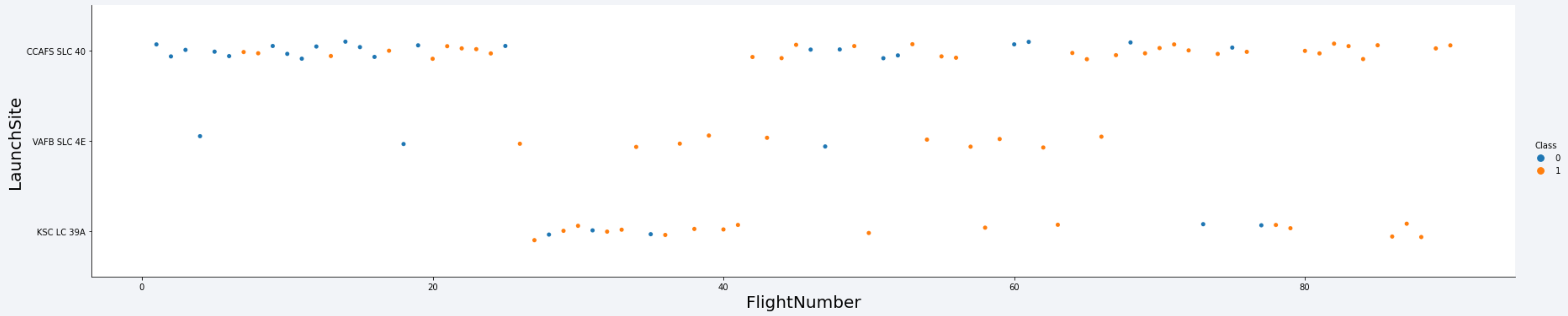
- **Predictive analysis:** decision tree classifier had the best parameter accuracy results, indicating this is the most accurate model

Section 2

Insights drawn from EDA

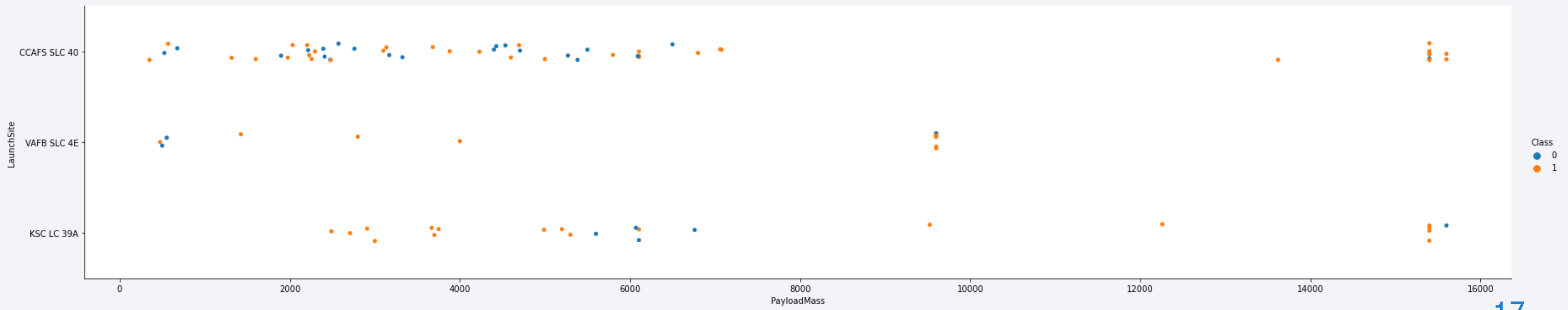
Flight Number vs. Launch Site

- The higher the flight number, the more success; VAFB SLC 4E had the least amount of failures

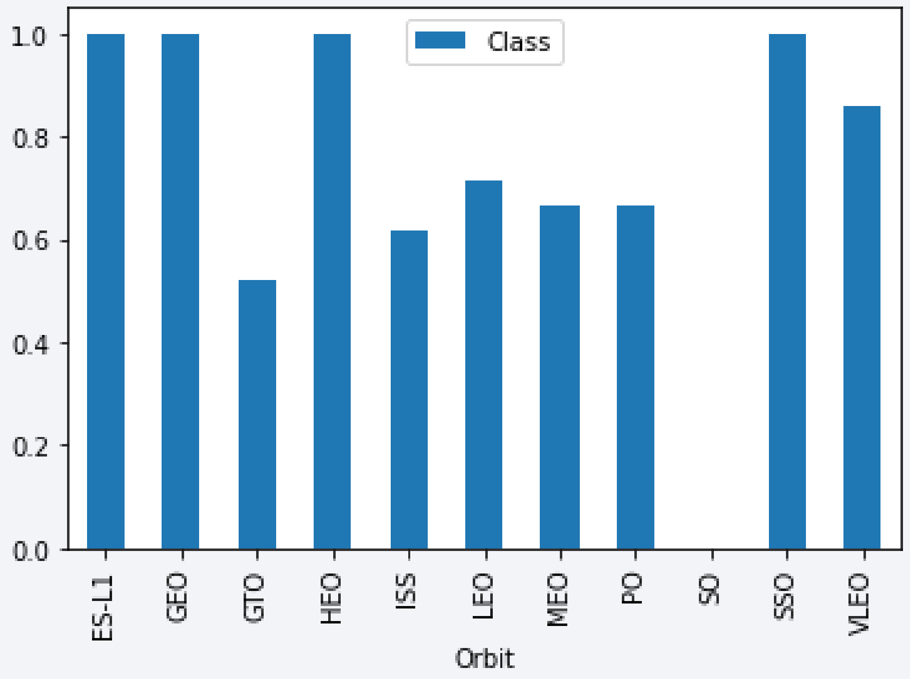


Payload vs. Launch Site

- For the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

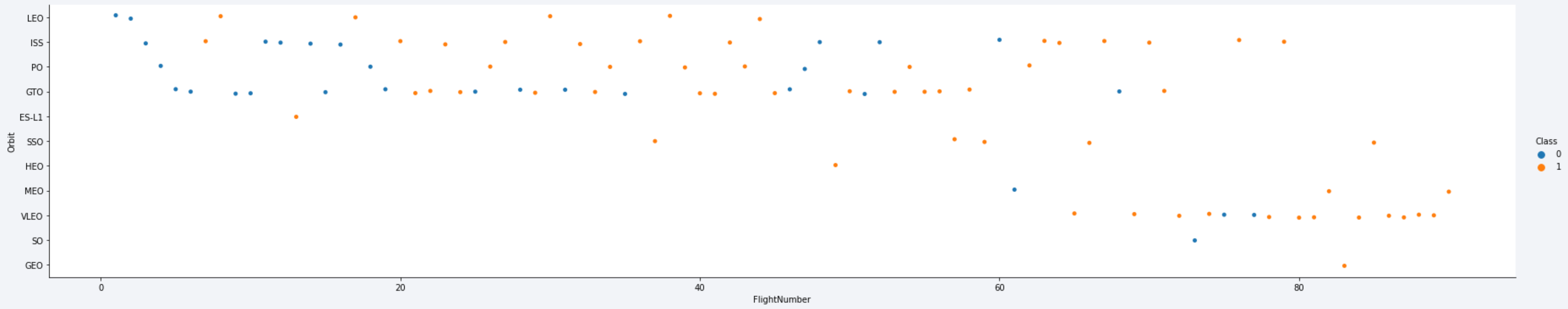


Success Rate vs. Orbit Type



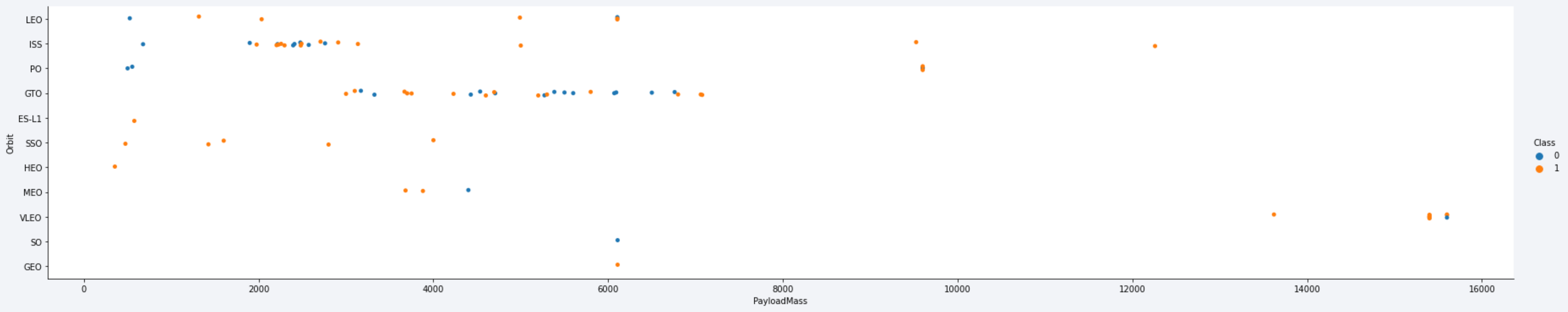
- ES-L1, GEO, HEO, and SSO have the highest success rates

Flight Number vs. Orbit Type



- For LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

Payload vs. Orbit Type



- With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.
- However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.

Launch Success Yearly Trend

- Show a line chart of yearly average success rate
- Show the screenshot of the scatter plot with explanations

All Launch Site Names

- 4 distinct launch sites identified in the data set that were involved in the space mission

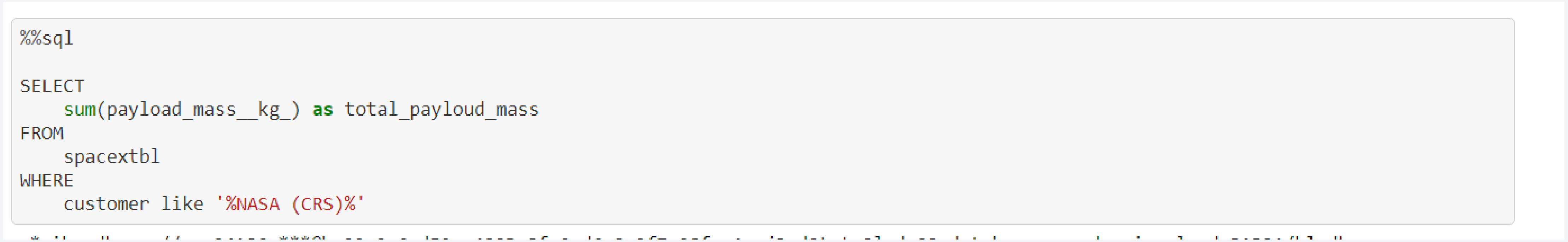
launch_site
CCAFS LC-40
CCAFS SLC-40
KSC LC-39A
VAFB SLC-4E

Launch Site Names Begin with 'CCA'

- Launch site name is CCAFS LC-40

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	None	0	LEO	SpaceX	Success	Failure (parachute)
2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	None	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	None	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	None	677	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-12	22:41:00	F9 v1.1	CCAFS LC-40	None	3170	GTO	SES	Success	No attempt

Total Payload Mass

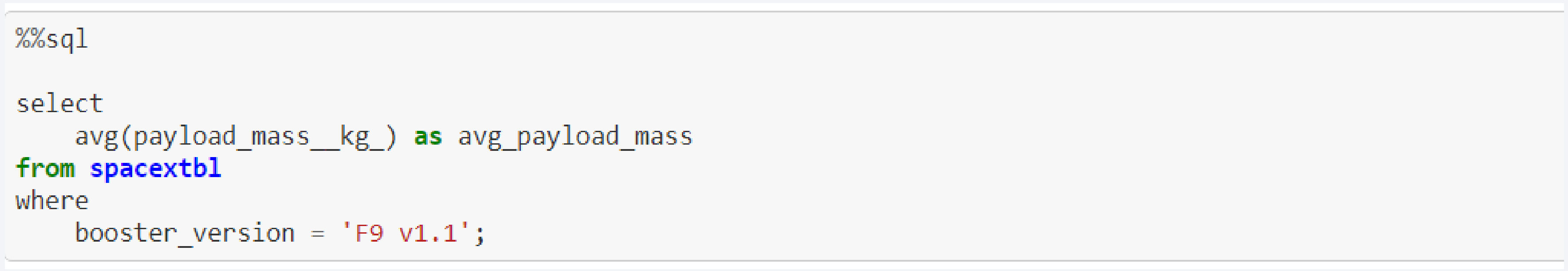


```
%%sql

SELECT
    sum(payload_mass__kg_) as total_payloud_mass
FROM
    spacextbl
WHERE
    customer like '%NASA (CRS)%'
```

total_payloud_mass
24624

- Total payload mass where customer name contains NASA (CRS)

Average Payload Mass by F9 v1.1

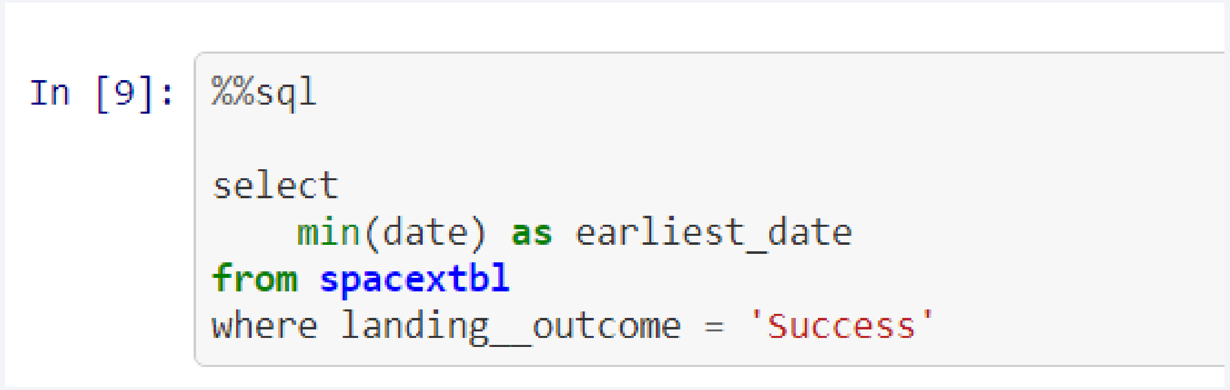


```
%%sql

select
    avg(payload_mass__kg_) as avg_payload_mass
from spacextbl
where
    booster_version = 'F9 v1.1';
```

avg_payload_mass
3676

- Total payload mass where booster_version is F9 v1.1

First Successful Ground Landing Date



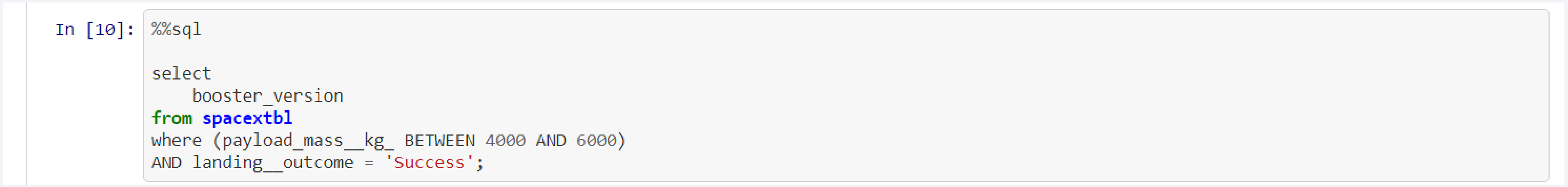
```
In [9]: %%sql

        select
            min(date) as earliest_date
        from spacextbl
        where landing__outcome = 'Success'
```

earliest_date
2018-03-12

Successful Drone Ship Landing with Payload between 4000 and 6000

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

booster_version
F9 B5 B1046.2
F9 B5 B1046.3
F9 B5 B1051.2
F9 B5B1062.1

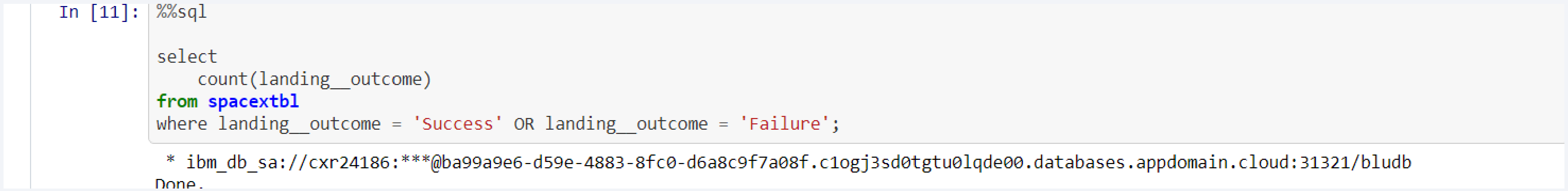


```
In [10]: %%sql

         select
             booster_version
         from spacextbl
         where (payload_mass__kg_ BETWEEN 4000 AND 6000)
         AND landing__outcome = 'Success';
```

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

1
19

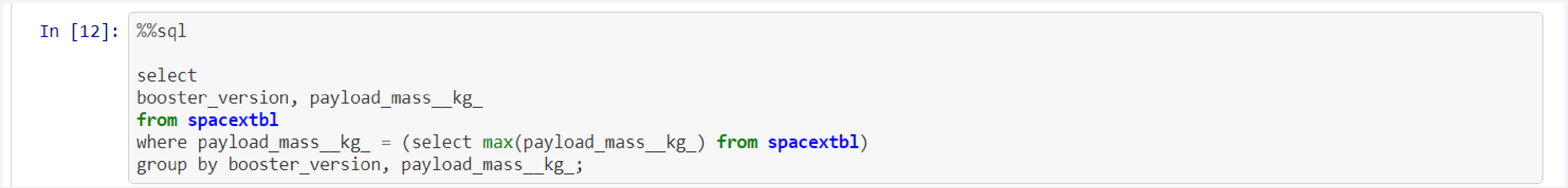


```
In [11]: %%sql

         select
             count(landing__outcome)
         from spacextbl
         where landing__outcome = 'Success' OR landing__outcome = 'Failure';

          * ibm_db_sa://cxr24186:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
         Done.
```

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass

booster_version	**payload_mass__kg_**
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1049.5	15600
F9 B5 B1058.3	15600
F9 B5 B1060.2	15600

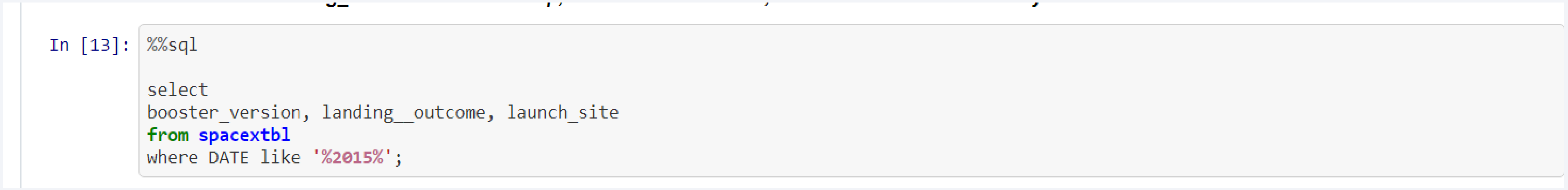


```
In [12]: %%sql

         select
         booster_version, payload_mass__kg_
         from spacextbl
         where payload_mass__kg_ = (select max(payload_mass__kg_) from spacextbl)
         group by booster_version, payload_mass__kg_;
```

2015 Launch Records

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

booster_version	landing__outcome	launch_site
F9 v1.1 B1012	Failure (drone ship)	CCAFS LC-40
F9 v1.1 B1013	Controlled (ocean)	CCAFS LC-40
F9 v1.1 B1014	No attempt	CCAFS LC-40



```
In [13]: %%sql

         select
         booster_version, landing__outcome, launch_site
         from spacextbl
         where DATE like '%2015%';
```

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

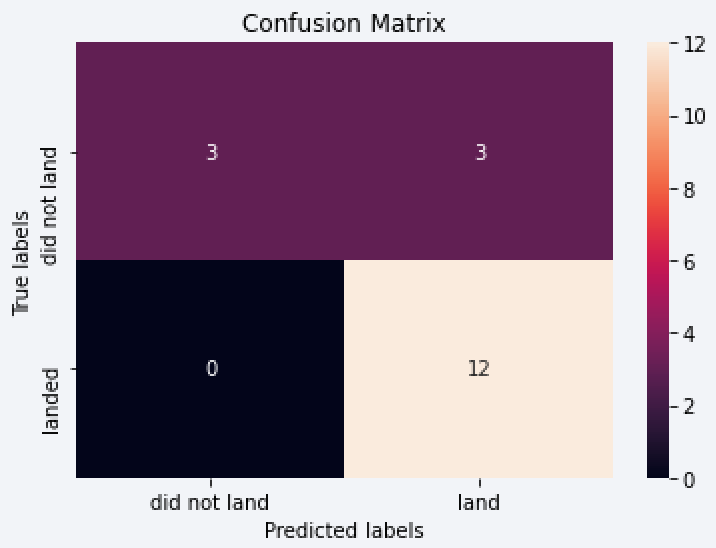
- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

count_landing_outcomes	**landing__outcome**
7	No attempt
2	Failure (drone ship)
2	Success (drone ship)
2	Success (ground pad)
1	Controlled (ocean)
1	Failure (parachute)

Section 6

Predictive Analysis (Classification)

Confusion Matrix



- Best performing model is the decision tree model with a score of 83.333%
- Highest accuracy parameters at 87.679%

Thank you!