IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Vishal H. Suryawanshi

09-AUG-2022

Outline

Executive Summary

- Summary of methodologies
    - Data collection using SpaceX API
    - Data collection from Wikipedia using Web Scrapping
    - Data Wrangling
    - Exploratory Data Analysis using SQL
    - Exploratory Data Analysis using Data Visualization
    - Interactive Dashboard using Folium and Dash
    - Classification using Machine
- Summary of all results
    - Exploratory Data Analysis result
    - Interactive Dashboard Screenshot
    - Classification using machine learning result

Introduction

- Project background and context

  SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upwards 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.
  In this project we will determine the price of each launch. We will do this by gathering information about Space X and creating dashboards for the team. We will also determine if SpaceX will reuse the first stage. Instead of using rocket science to determine if the first stage will land successfully, we will train a machine learning model and use public information to predict if SpaceX will reuse the first stage.

- Problems you want to find answers

  1. Which factors determines if the rocket will land successfully or not?
  2. What is relation between different features which determines the success rate?
  3. Which are the best operating conditions to ensure the successful landing of the rocket?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data is collected using SpaceX API and Launch data on Wikipedia is collected using Web Scrapping.
- Performed data wrangling
  - In data wrangling step handled missing values, data types and categorical values.
- Performed exploratory data analysis (EDA) using visualization and SQL
- Performed interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

SpaceX launch data is collected using below two methods:

1. SpaceX API:

   Using SpaceX API, we got the data in JASON format then we converted that JSON data into pandas DataFrame and performed data wrangling.

   SpaceX URL : https://api.spacexdata.com/v4/launches/past

   Payload Data: https://api.spacexdata.com/v4/payloads/

   Rocket Data: https://api.spacexdata.com/v4/rockets/

   Launch pad data: https://api.spacexdata.com/v4/launchpads/

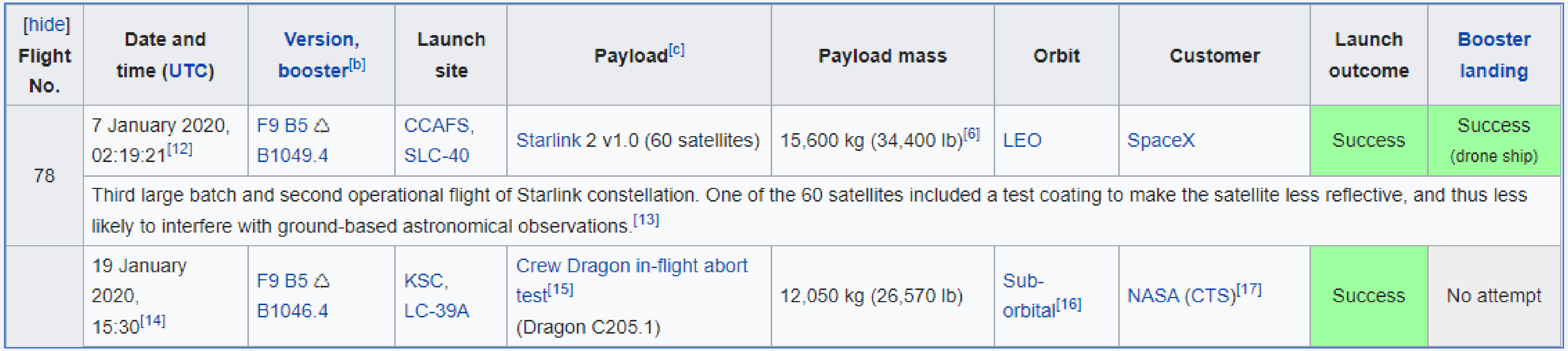
   After collecting all data, we performed data wrangling like handling missing values.

Data Collection

2. Web Scrapping:

Using Web scrapping we collected data of Falcon 9 historical launch records from a Wikipedia page titled `List of Falcon 9 and Falcon Heavy launches`.

URL : https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches

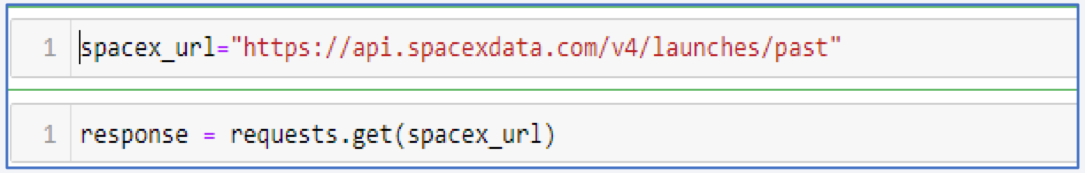


[hide] Flight No.	Date and time (UTC)	Version, booster[b]	Launch site	Payload[c]	Payload mass	Orbit	Customer	Launch outcome	Booster landing
78	7 January 2020, 02:19:21[12]	F9 B5 △ B1049.4	CCAFS, SLC-40	Starlink 2 v1.0 (60 satellites)	15,600 kg (34,400 lb)[6]	LEO	SpaceX	Success	Success (drone ship)
	Third large batch and second operational flight of Starlink constellation. One of the 60 satellites included a test coating to make the satellite less reflective, and thus less likely to interfere with ground-based astronomical observations.[13]								
	19 January 2020, 15:30[14]	F9 B5 △ B1046.4	KSC, LC-39A	Crew Dragon in-flight abort test[15] (Dragon C205.1)	12,050 kg (26,570 lb)	Sub-orbital[16]	NASA (CTS)[17]	Success	No attempt

Data Collection – SpaceX API

- We used get request to the SpaceX API and got the data in json format. Converted that json data into pandas DataFrame using pandas library.
- After storing data in the dataframe we performed basic data wrangling steps like handling missing values.
- GitHub URL: https://github.com/vishal-suryawanshi/DS101-Capstone-Project/blob/69af60c3829fe75645b52de5d57bf1b3bcd157ee/SpaceX%20Data%20Collection%20API.ipynb

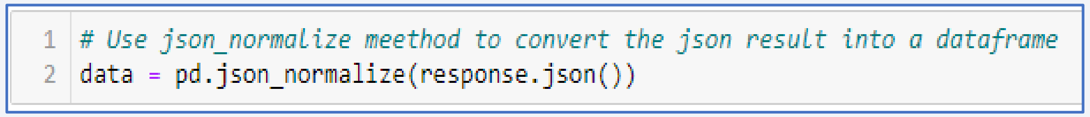
1. Get request to SpaceX API



```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

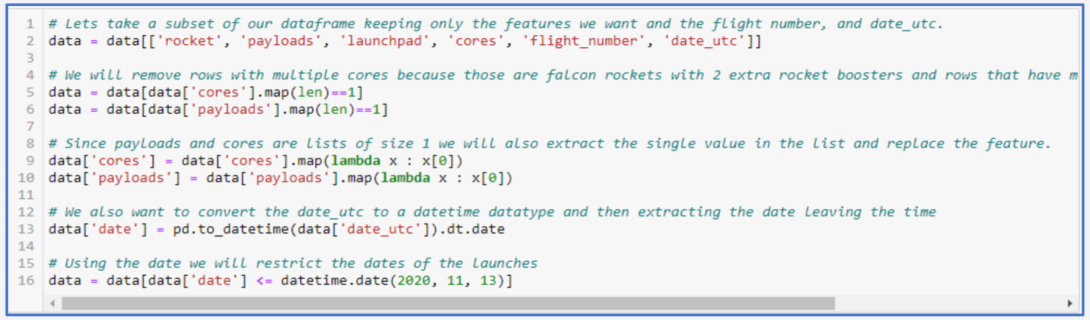
```
response = requests.get(spacex_url)
```

2. Json data to pandas dataframe



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

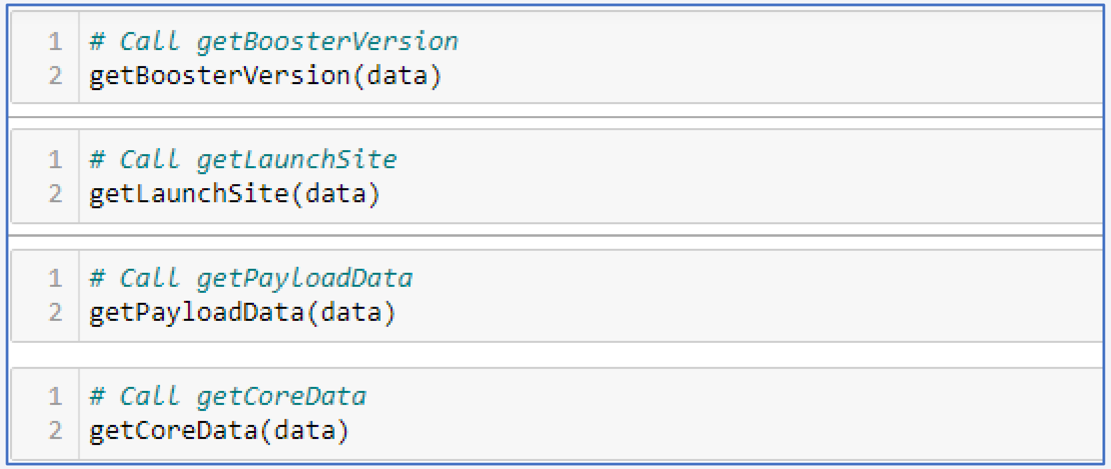
3. use the API again to get information about the launches using the IDs



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

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

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

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

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

Data Collection – SpaceX API

4. API calls to get Booster Version, Launch Site Payload and Core Data.



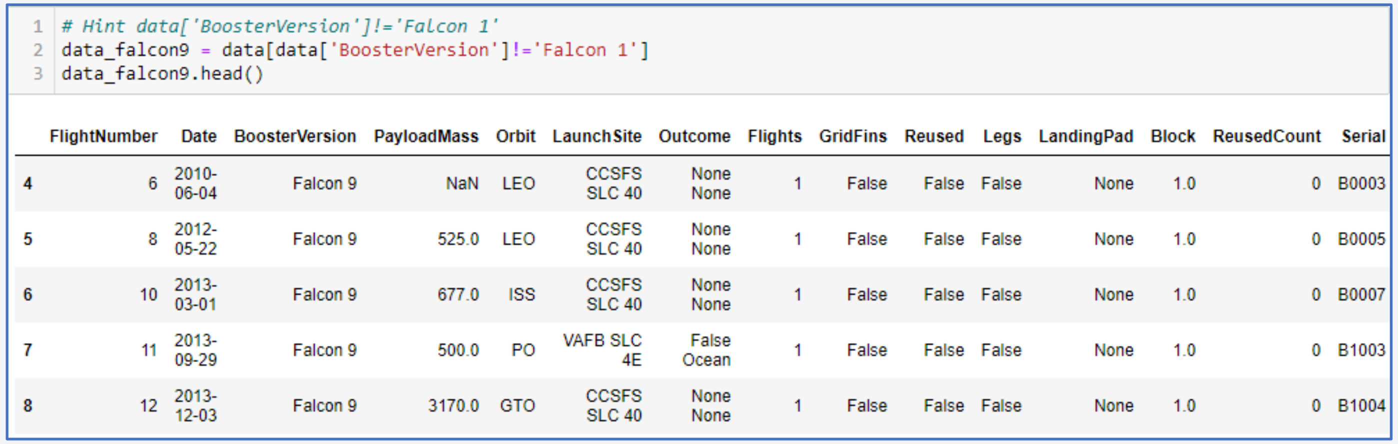
```
# Call getBoosterVersion
getBoosterVersion(data)
```

```
# Call getLaunchSite
getLaunchSite(data)
```

```
# Call getPayloadData
getPayloadData(data)
```

```
# Call getCoreData
getCoreData(data)
```

5. Filter out Falcone 9 data:



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

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003
5	8	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0005
6	10	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0007
7	11	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None	1.0	0	B1003
8	12	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B1004

Data Collection - Scraping

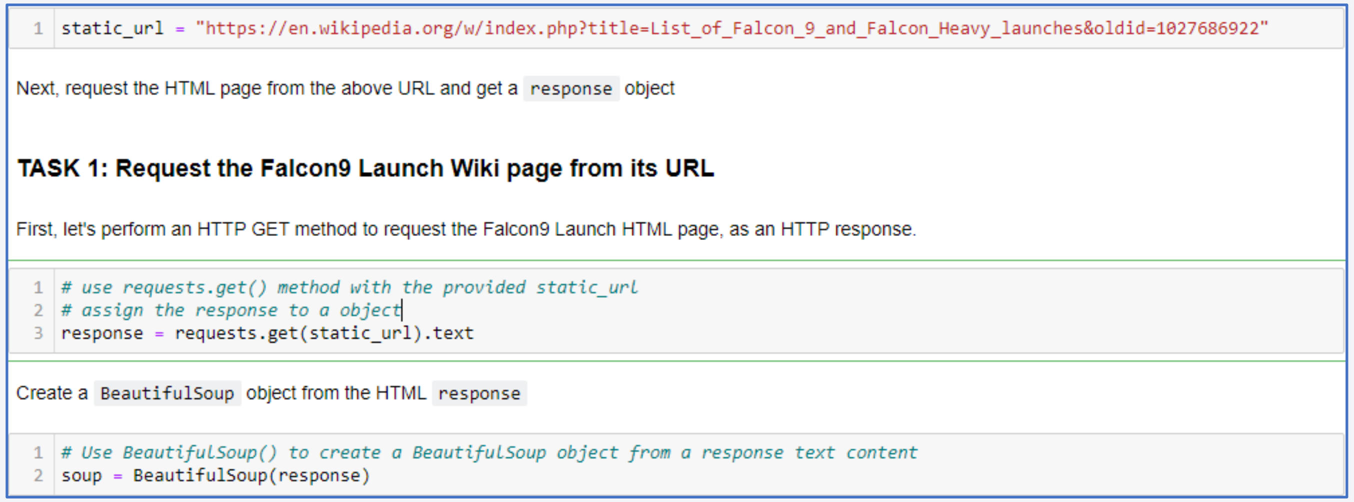
We performed web scraping to collect Falcon 9 historical launch records from a Wikipedia page titled `List of Falcon 9 and Falcon Heavy launches`

https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches

GitHub URL:

https://github.com/vishal-suryawanshi/DS101-Capstone-Project/blob/80271795ef0633663fd37a7d6aff3846620cdbcd/Webscraping.ipynb

1. Get data using http get request and using that response create BeautifulSoup object



```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

Next, request the HTML page from the above URL and get a `response` object

TASK 1: Request the Falcon9 Launch Wiki page from its URL

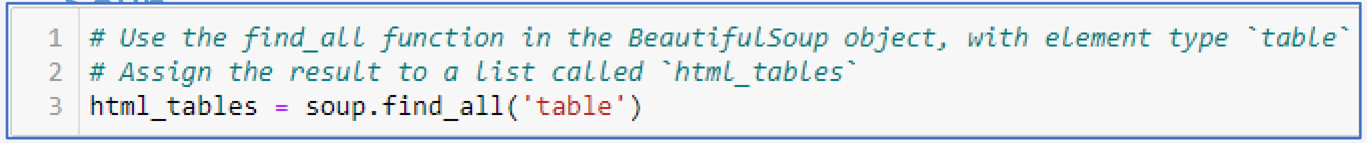
First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

```
# 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`

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response)
```

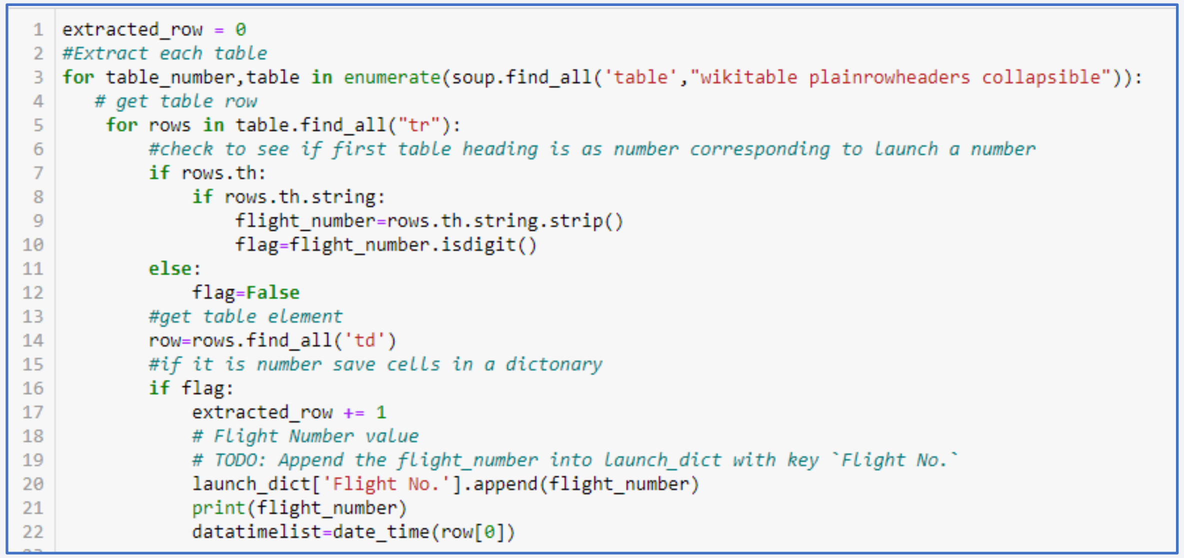
2. Extract Column and headers of table using beautiful Soup



```
# 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')
```

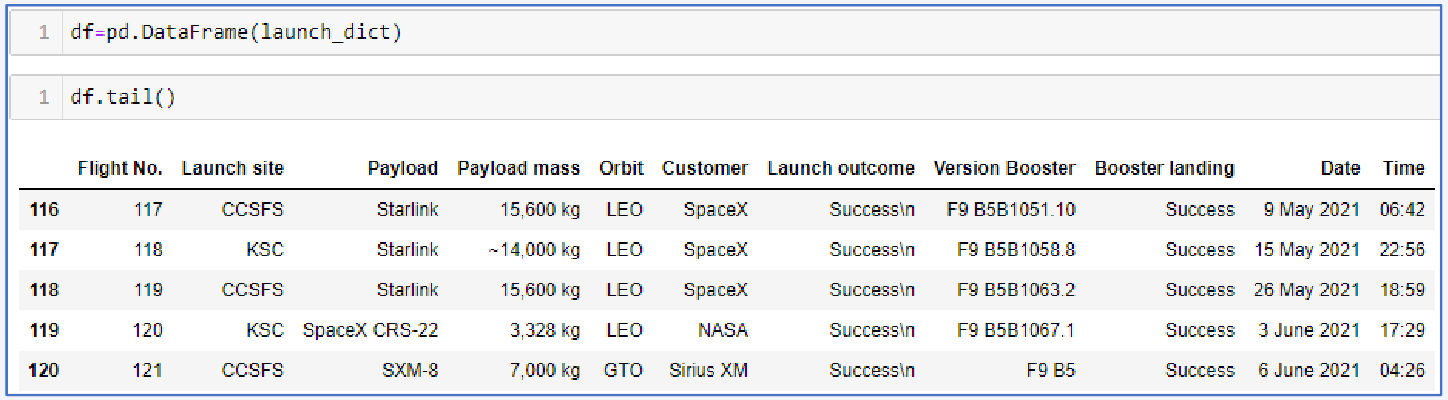
Data Collection - Scraping

3. Extract all row from response:



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

4. Store data into pandas dataframe:



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

```python
df.tail()
```

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
116	117	CCSFS	Starlink	15,600 kg	LEO	SpaceX	Success\n	F9 B5B1051.10	Success	9 May 2021	06:42
117	118	KSC	Starlink	~14,000 kg	LEO	SpaceX	Success\n	F9 B5B1058.8	Success	15 May 2021	22:56
118	119	CCSFS	Starlink	15,600 kg	LEO	SpaceX	Success\n	F9 B5B1063.2	Success	26 May 2021	18:59
119	120	KSC	SpaceX CRS-22	3,328 kg	LEO	NASA	Success\n	F9 B5B1067.1	Success	3 June 2021	17:29
120	121	CCSFS	SXM-8	7,000 kg	GTO	Sirius XM	Success\n	F9 B5	Success	6 June 2021	04:26

Data Wrangling

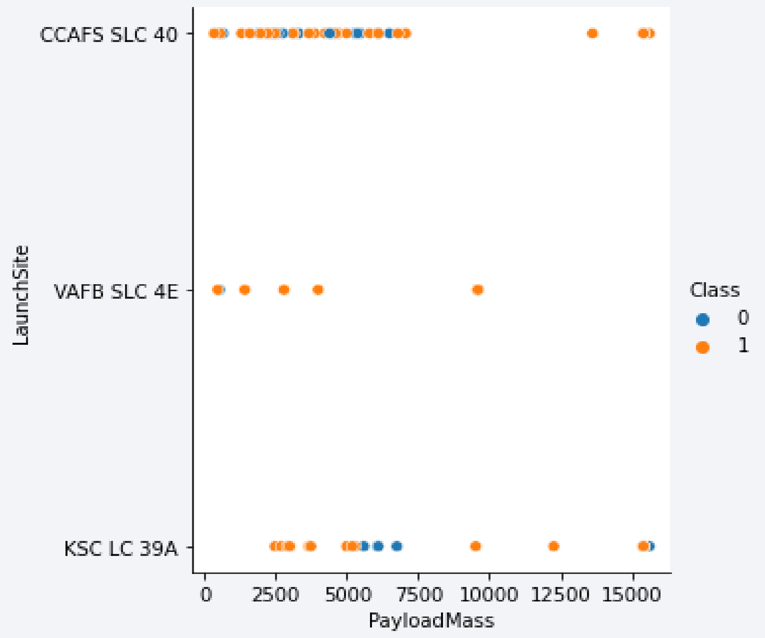
- In data wrangling step we performed Exploratory Analysis and determined training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.
- We calculated number of occurrences in each orbit.
- We calculated the number and occurrences of mission outcome per orbit type

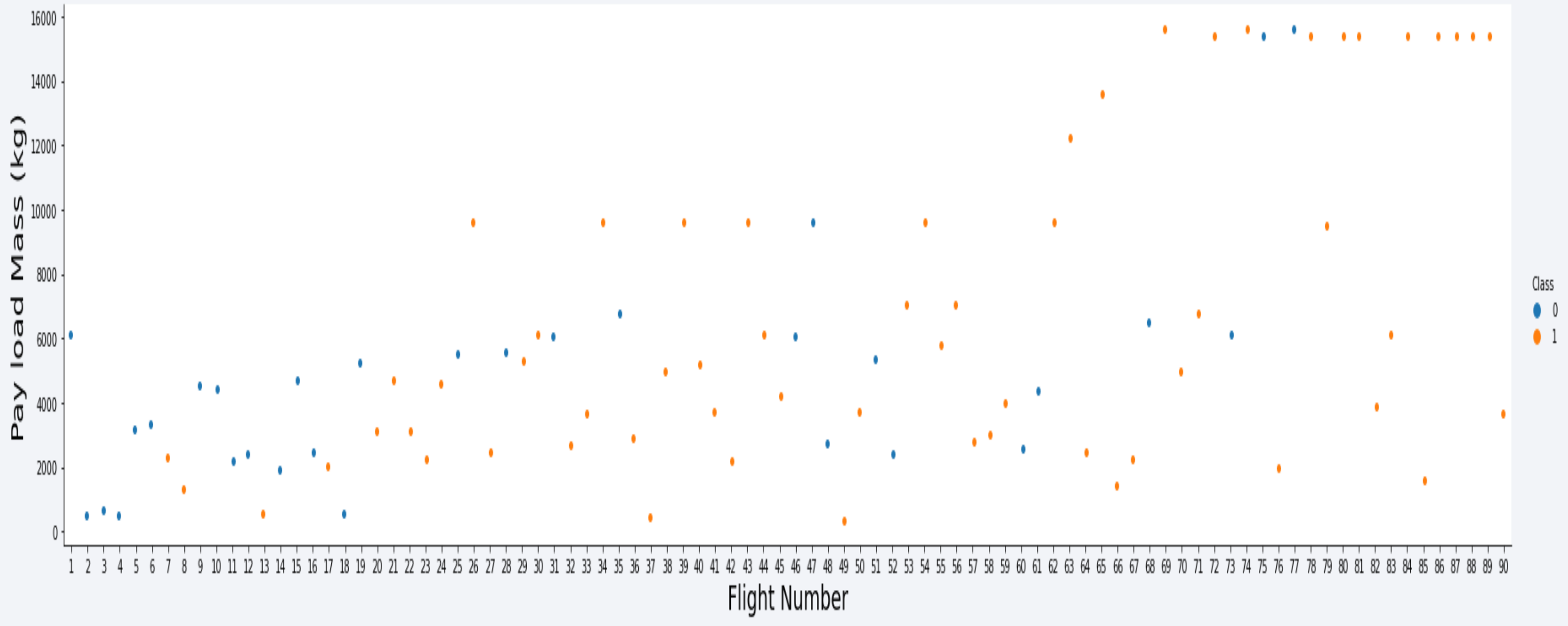
GitHub URL:

https://github.com/vishal-suryawanshi/DS101-Capstone-Project/blob/80271795ef0633663fd37a7d6aff3846620cdbcd/Spacex%20Data%20wrangling.ipynb

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, flight number and orbit type, the launch success yearly trend.





We observed Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launch site there are no rockets launched for heavy payload mass(greater than 10000).

We see that different launch sites have different success rates. CCAFS LC-40, has a success rate of 60 %, while KSC LC-39A and VAFB SLC 4E has a success rate of 77%.

GitHub URL: https://github.com/vishal-suryawanshi/DS101-Capstone-Project/blob/a8436b682bb1aeaa29fabf196a9aae5c7790fbbc/EDA%20Data%20Visualization%20.ipynb

EDA with SQL

- We loaded SpaceX data using sqlalchemy and performed Exploratory data analysis using sql extension in 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 space 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.
- GitHub URL: https://github.com/vishal-suryawanshi/DS101-Capstone-Project/blob/0c6d52f6236f6c566e3a25c58988d12fda3262f7/jupyter-labs-eda-sql-coursera_sqllite.ipynb

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.
- GitHub URL: https://github.com/vishal-suryawanshi/DS101-Capstone-Project/blob/4d1b92a0416da8e701942014d342092807b4d250/Interactive%20map%20using%20Folium.ipynb
- Folium Images: https://github.com/vishal-suryawanshi/DS101-Capstone-Project/tree/main/Folium%20Images

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, After selecting any launch site from dropdown it will show the percentage of successful and Failed launch for that launch site.
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version. Also provided Payload Mass slider from range 0 Kg to 1000 Kg.
- GitHub URL:
  - Python File: https://github.com/vishal-suryawanshi/DS101-Capstone-Project/blob/80271795ef0633663fd37a7d6aff3846620cdbcd/Interactive%20Dashboard%20with%20Ploty%20Dash.py
  - Images: https://github.com/vishal-suryawanshi/DS101-Capstone-Project/tree/main/Images

Predictive Analysis (Classification)

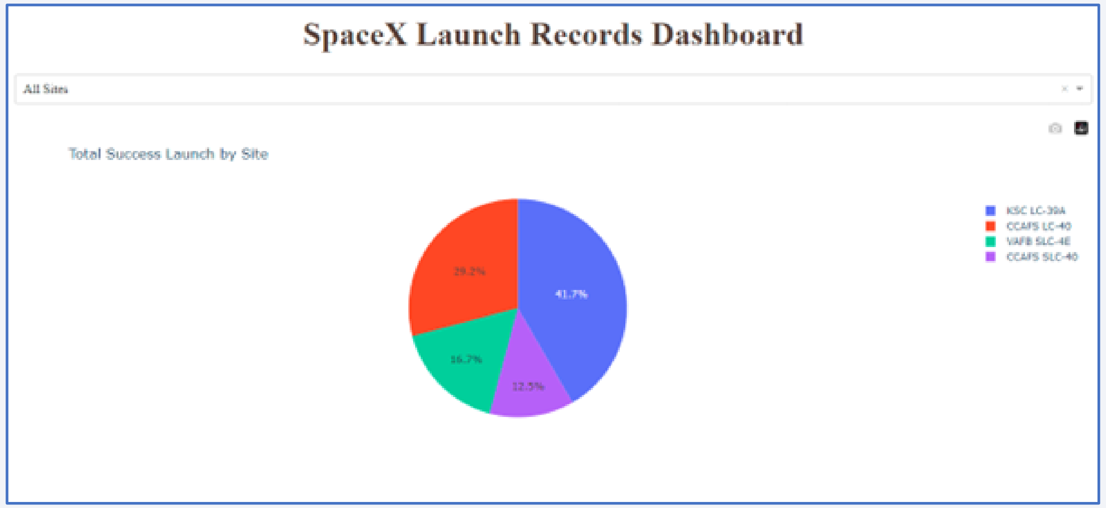
- We loaded the data using into DataFrame using pandas, transformed the data, split our data into training and testing using train_test_split.
- We built Logistic Regression, KNN, SVM and Decision tree machine learning classification models and tuned with different hyperparameters using GridSearchCV.
- 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. In our case it is decision tree is a best model with accuracy of 87%.
- GitHub URL: https://github.com/vishal-suryawanshi/DS101-Capstone-Project/blob/5712c6733aa974b6db25d59ce0f926f687abc094/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

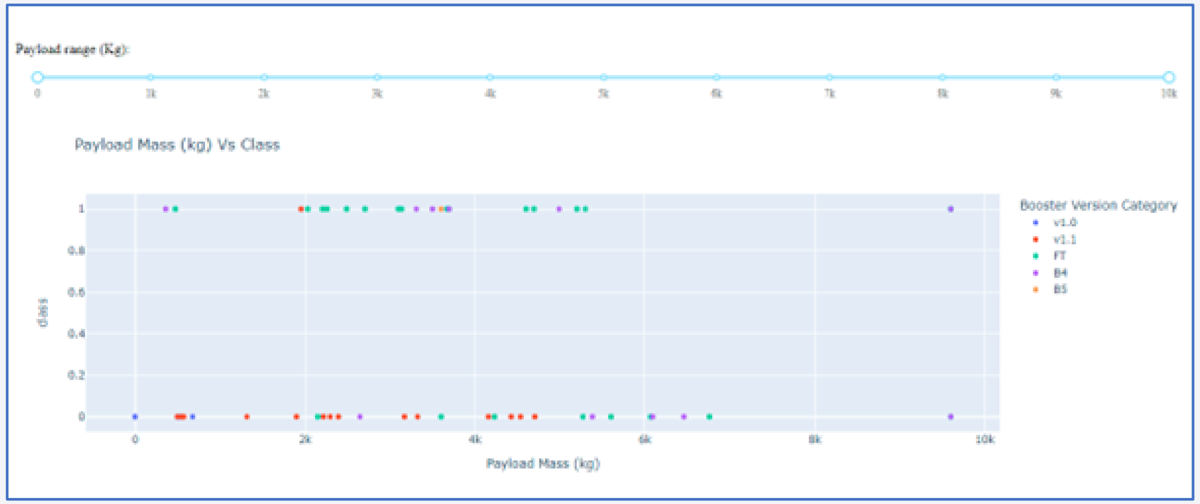
Results

- Exploratory data analysis results
  - different launch sites have different success rates. CCAFS LC-40, has a success rate of 60 %, while KSC LC-39A and VAFB SLC 4E has a success rate of 77%.
  - For the VAFB-SLC launch site there are no rockets launched for heavy payload mass(greater than 10000).
  - In the 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.
  - 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.
  - Success rate since 2013 kept increasing till 2020.

Results

- Interactive analytics demo in screenshots





Results

- Predictive analysis results
  - Logistic Regression accuracy: 0.8464285714285713
  - SVM accuracy: 0.8482142857142856
  - Decision Tree accuracy: 0.8625
  - KNN accuracy: 0.8482142857142858

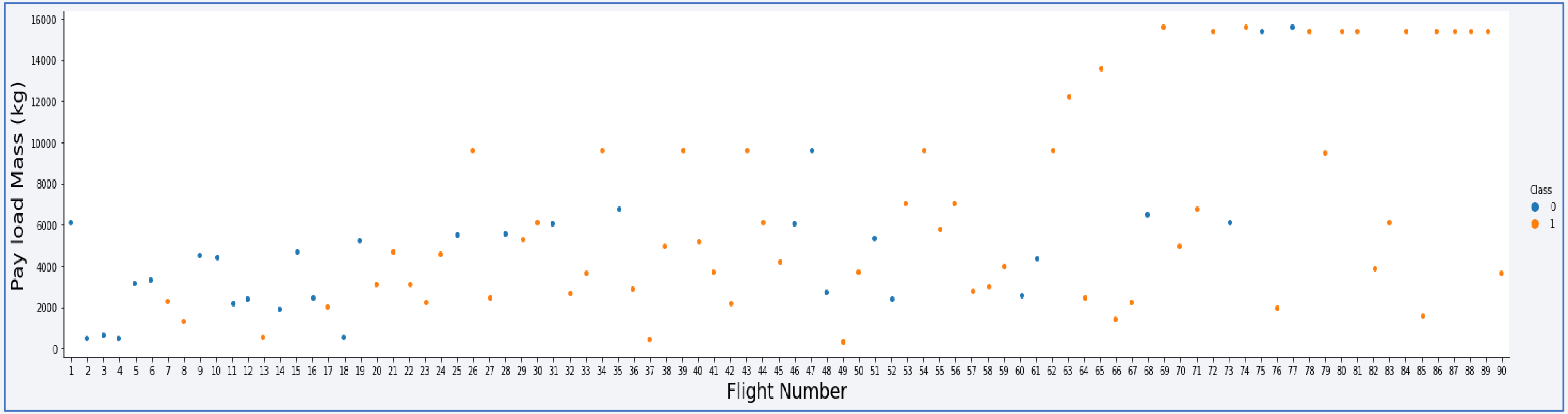
Decision Tree performs best with accuracy of 86.25%

Best Parameters of tree :
{'criterion': 'gini', 'max_depth': 2, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'best'}

Section 2

Insights drawn from EDA

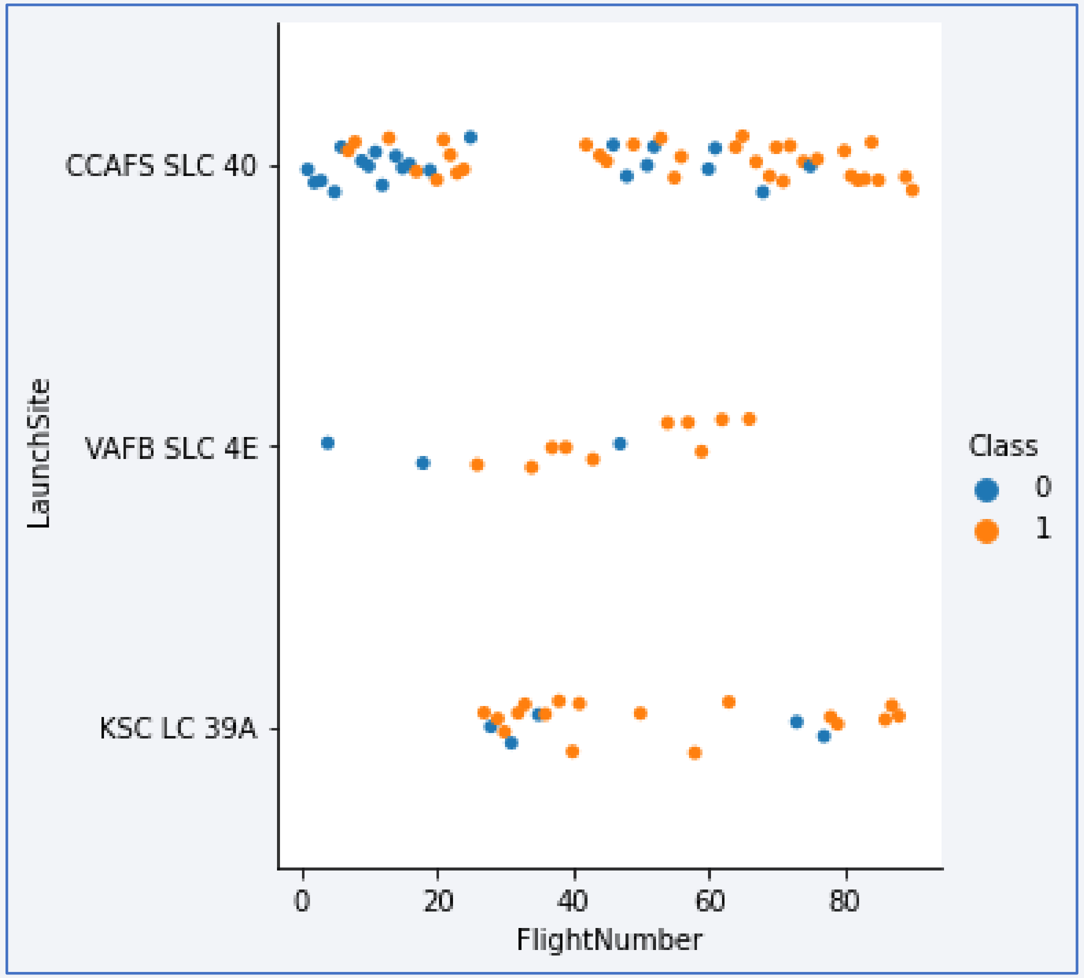
Flight Number vs. Launch Site



- We see that as the flight number increases, the first stage is more likely to land successfully.
- The payload mass is also important; it seems the more massive the payload, the less likely the first stage will return.

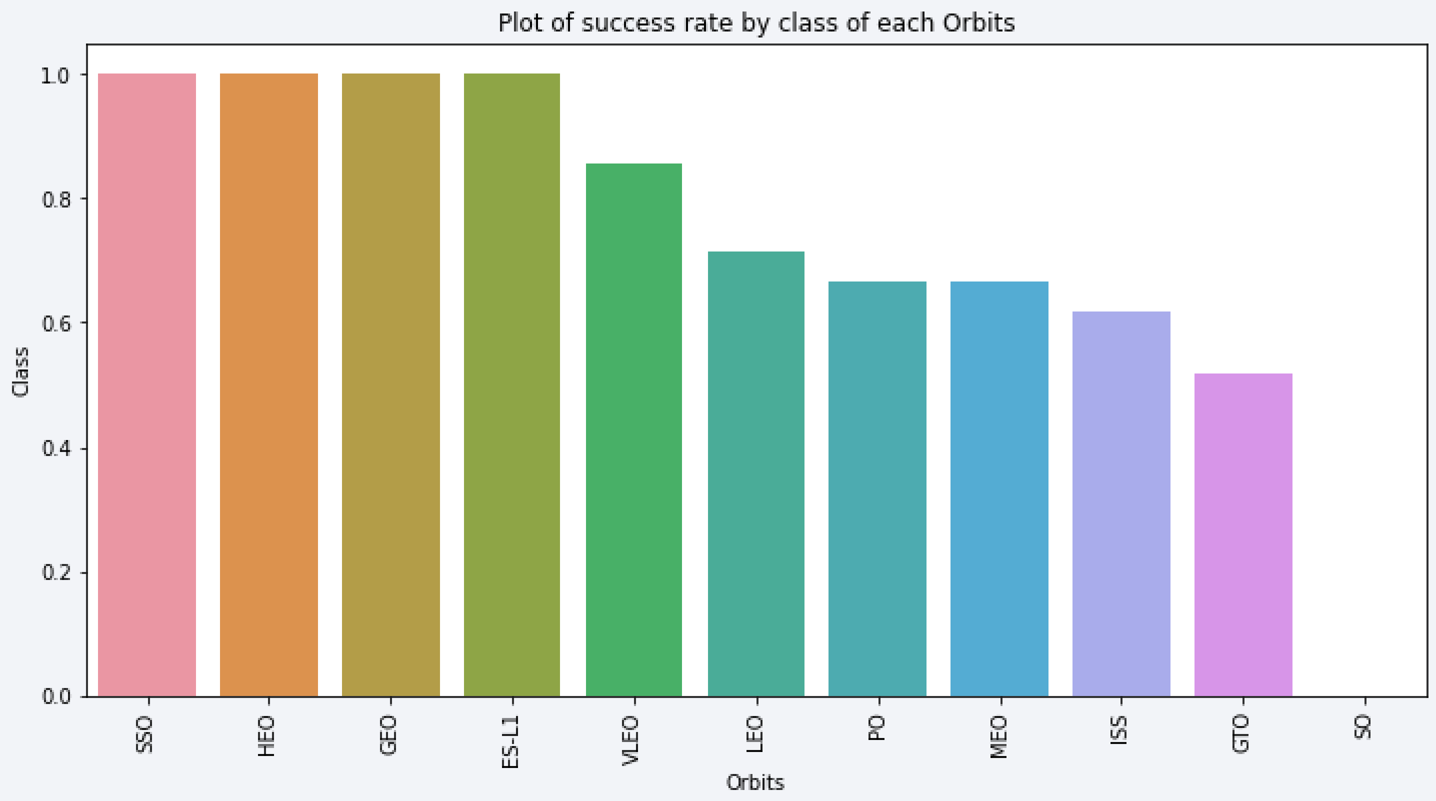
Payload vs. Launch Site

- We see that different launch sites have different success rates.
- CCAFS LC-40, has a success rate of 60 %, while KSC LC-39A and VAFB SLC 4E has a success rate of 77%.



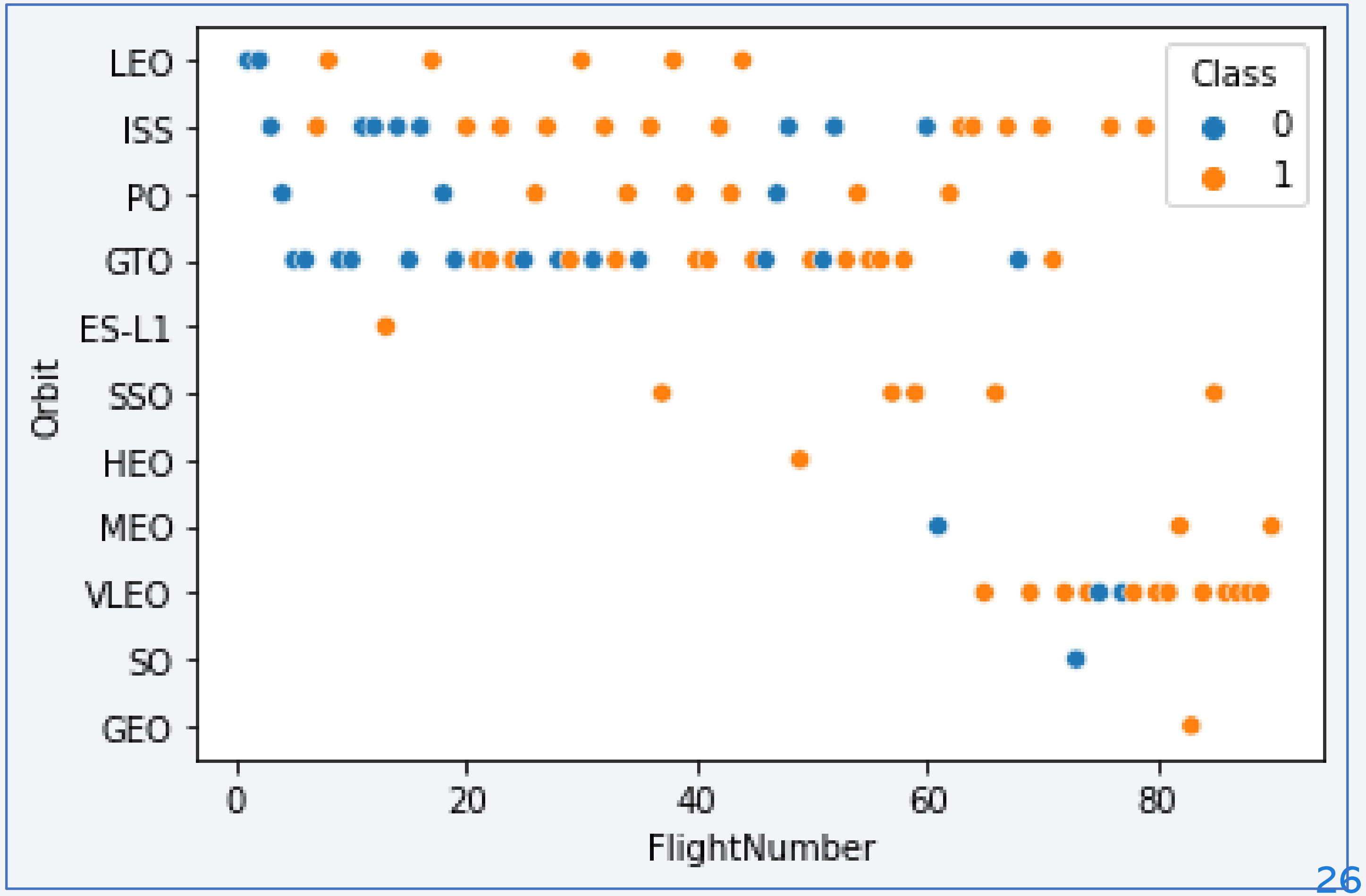
Success Rate vs. Orbit Type

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



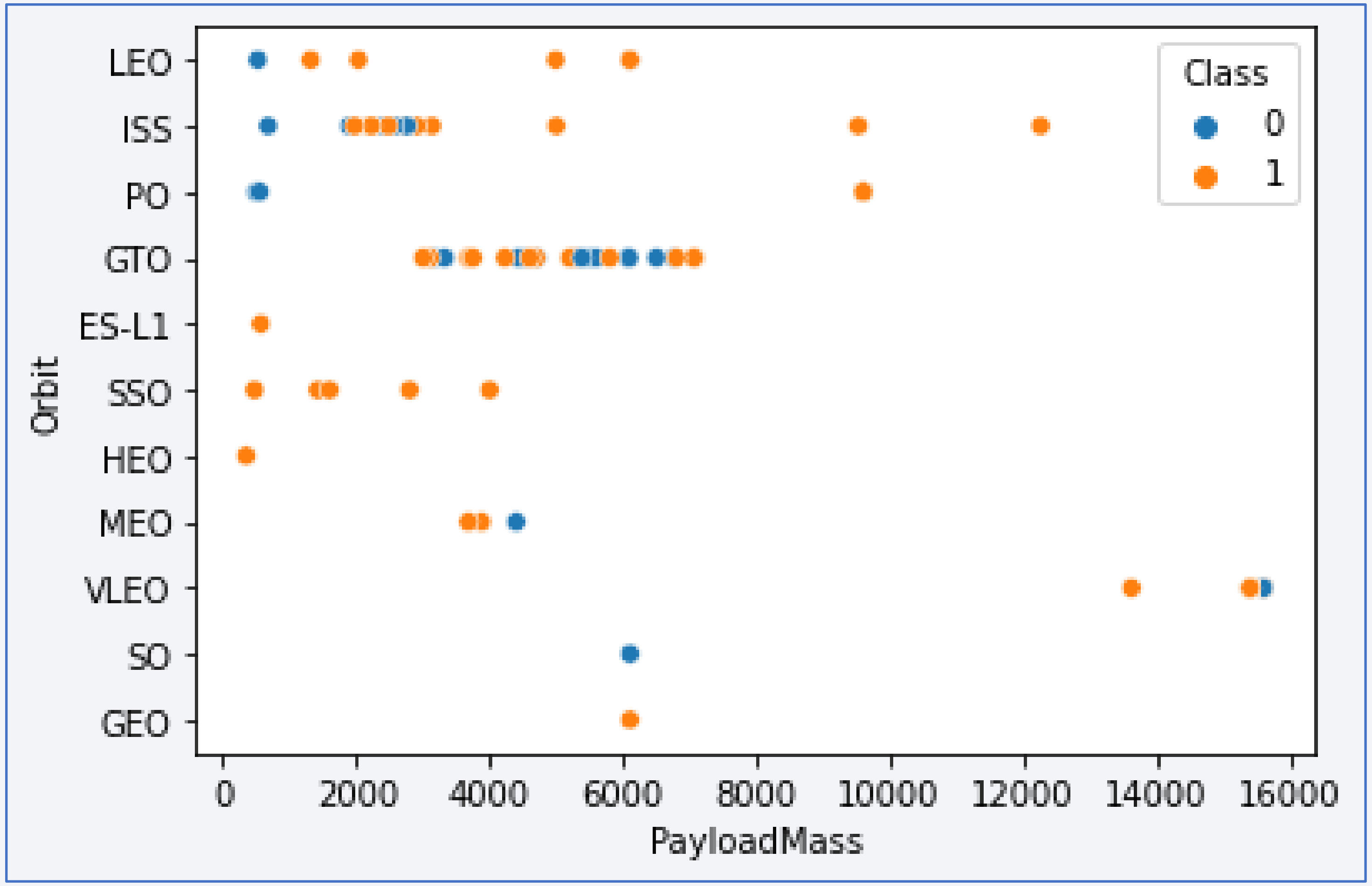
Flight Number vs. Orbit Type

- 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.
- VLEO and SSO has more success rate

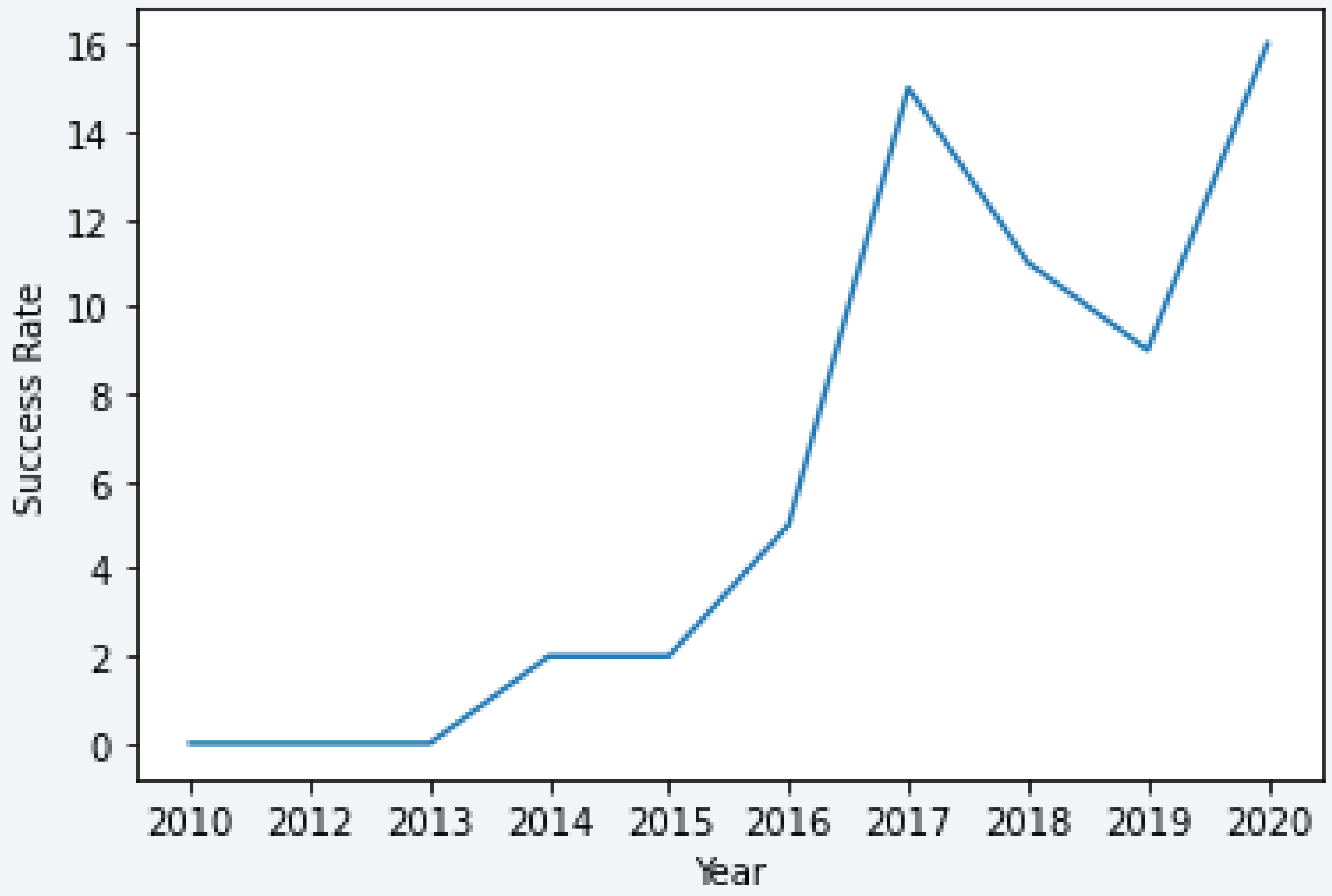


Payload vs. Orbit Type

- 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.

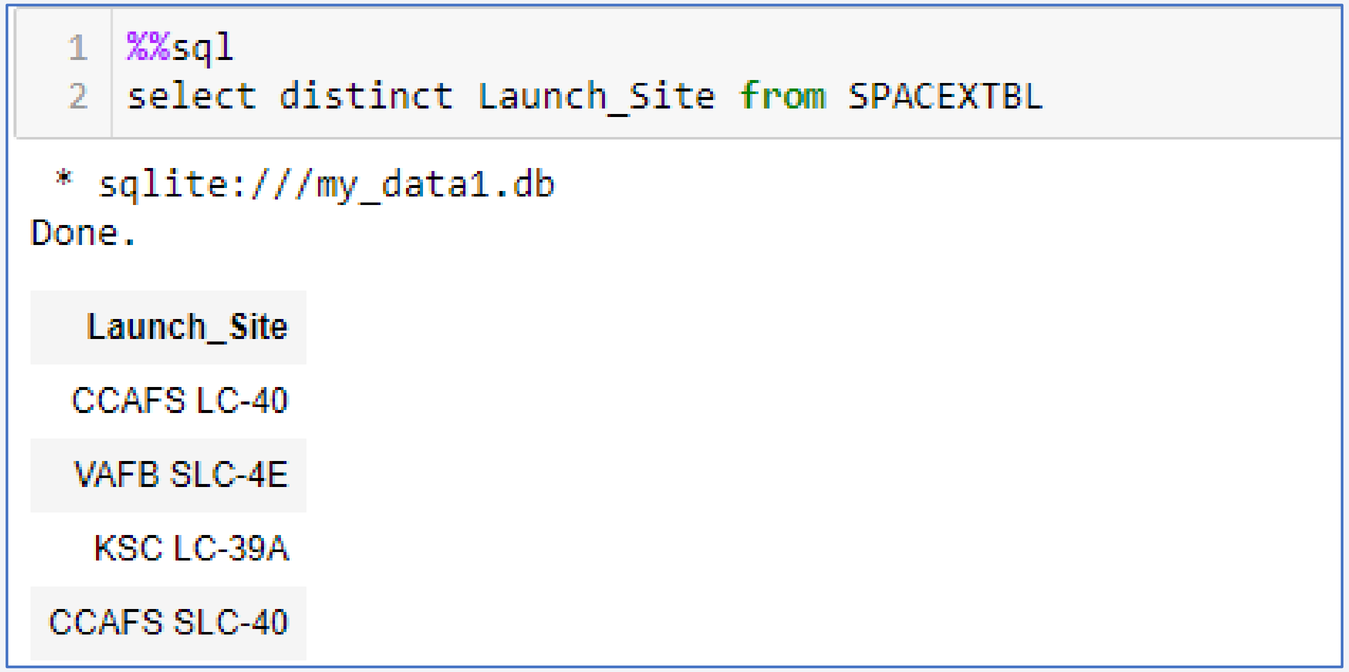


Launch Success Yearly Trend



Success rate since 2013 kept increasing till 2020

All Launch Site Names



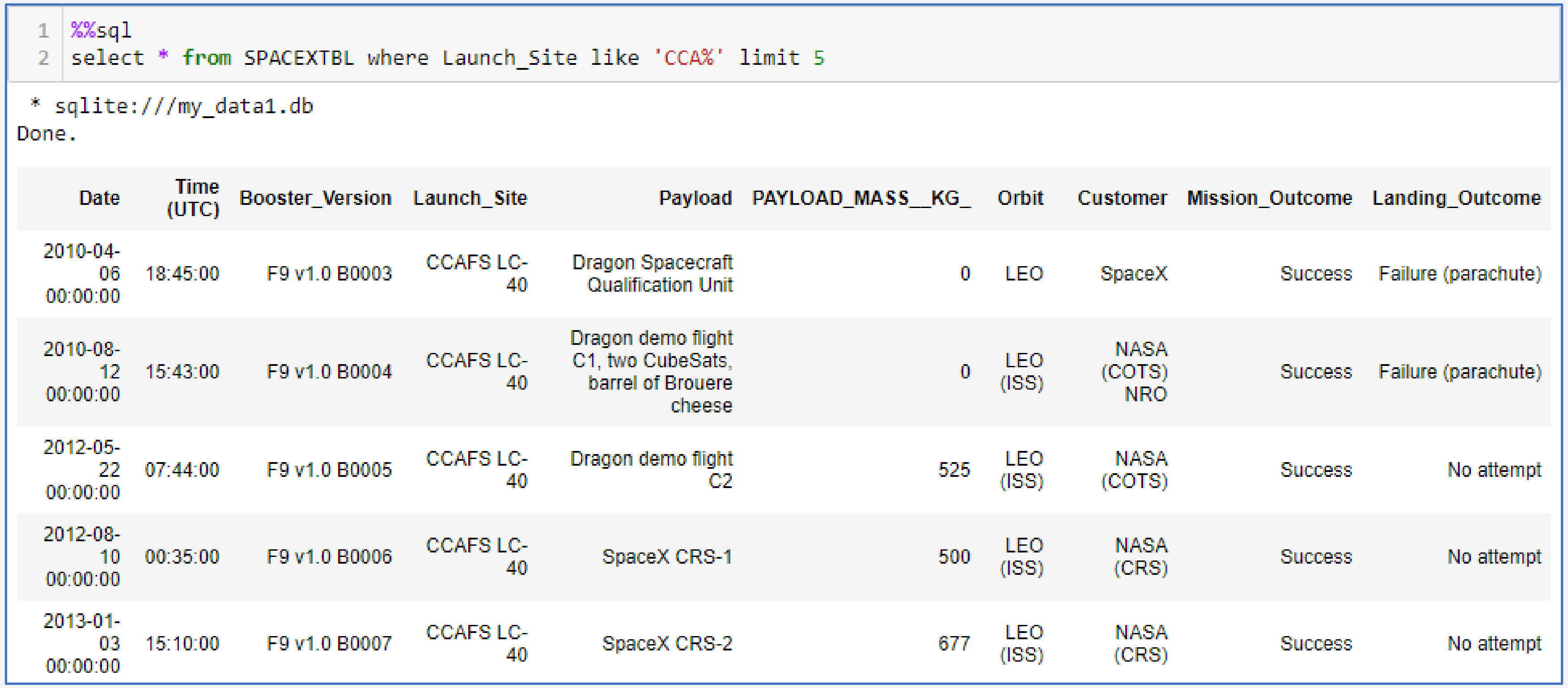
```
%%sql
select distinct Launch_Site from SPACEXTBL
```

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

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

- We used distinct Key word to get unique Launch site names.
- There are total 4 launch sites.

Launch Site Names Begin with 'CCA'



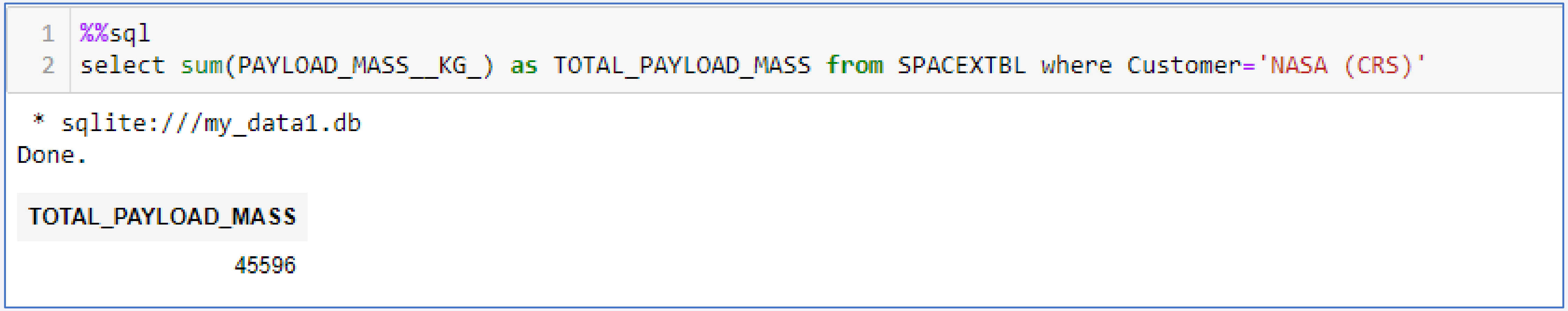
```
%%sql
select * from SPACEXTBL where Launch_Site like 'CCA%' limit 5
```

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

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

- We use like key word to match pattern and limit key word to limit the number of records

Total Payload Mass

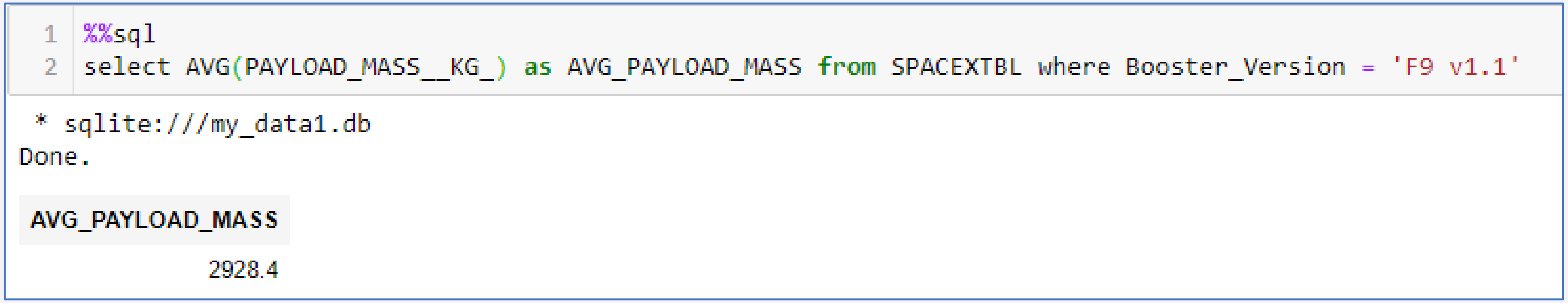


```
%%sql
select sum(PAYLOAD_MASS__KG_) as TOTAL_PAYLOAD_MASS from SPACEXTBL where Customer='NASA (CRS)'
```

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

TOTAL_PAYLOAD_MASS
             45596
```

- Calculate the total payload carried by boosters from NASA we used SUM function of sql with Customer filter.

Average Payload Mass by F9 v1.1



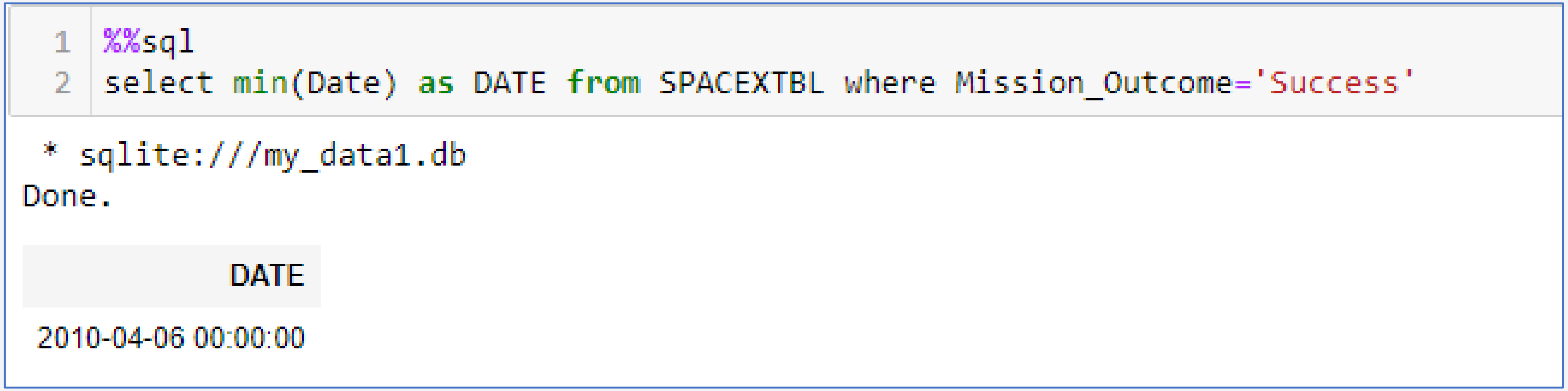
```
%%sql
select AVG(PAYLOAD_MASS__KG_) as AVG_PAYLOAD_MASS from SPACEXTBL where Booster_Version = 'F9 v1.1'
```

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

AVG_PAYLOAD_MASS
2928.4

- To calculate the average payload mass carried by booster version F9 v1.1 we used AVG function of SQL with filter on Booster version

First Successful Ground Landing Date



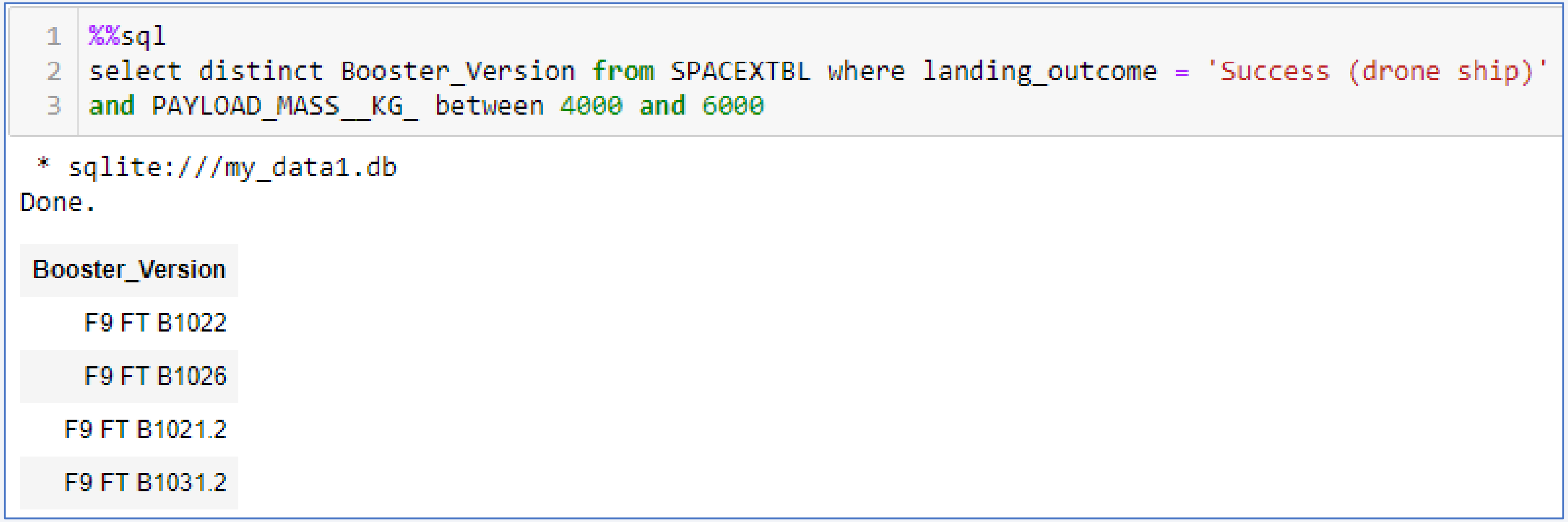
```
%%sql
select min(Date) as DATE from SPACEXTBL where Mission_Outcome='Success'
```

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

DATE
2010-04-06 00:00:00

- To find the dates of the first successful landing outcome on ground pad we used MIN function of sql on date column with filter on Mission Outcome column

Successful Drone Ship Landing with Payload between 4000 and 6000



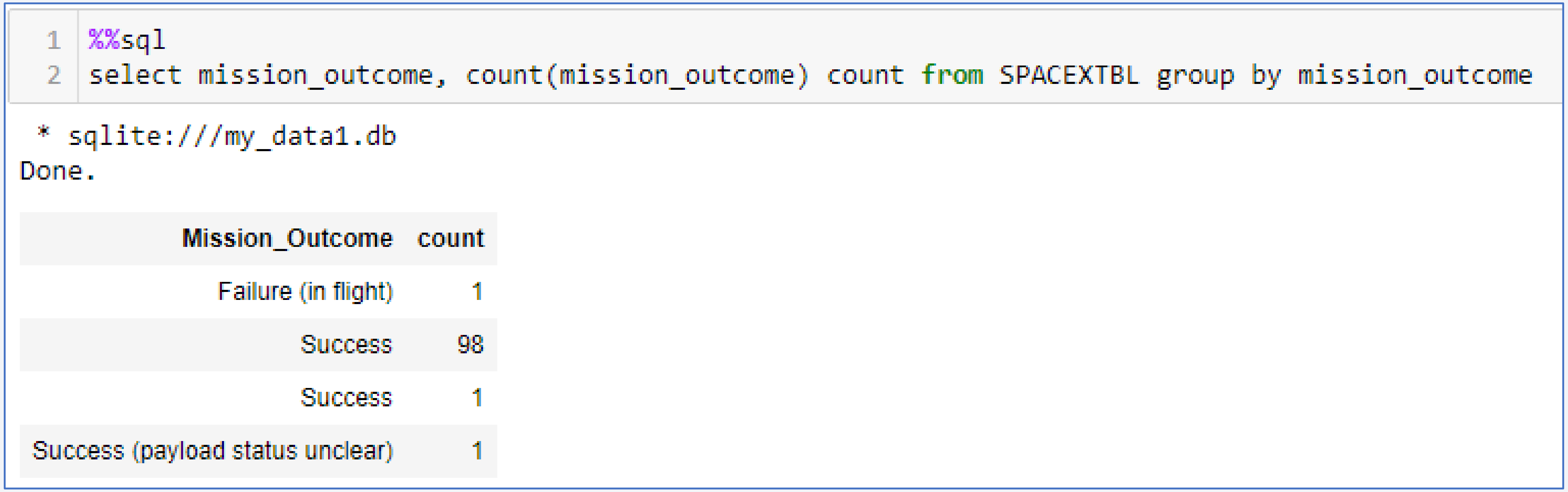
```
%%sql
select distinct Booster_Version from SPACEXTBL where landing_outcome = 'Success (drone ship)'
and PAYLOAD_MASS__KG_ between 4000 and 6000
```

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

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

- We used where clause and between key word to find the successful lading with Payload mass between 4000 and 6000.

Total Number of Successful and Failure Mission Outcomes



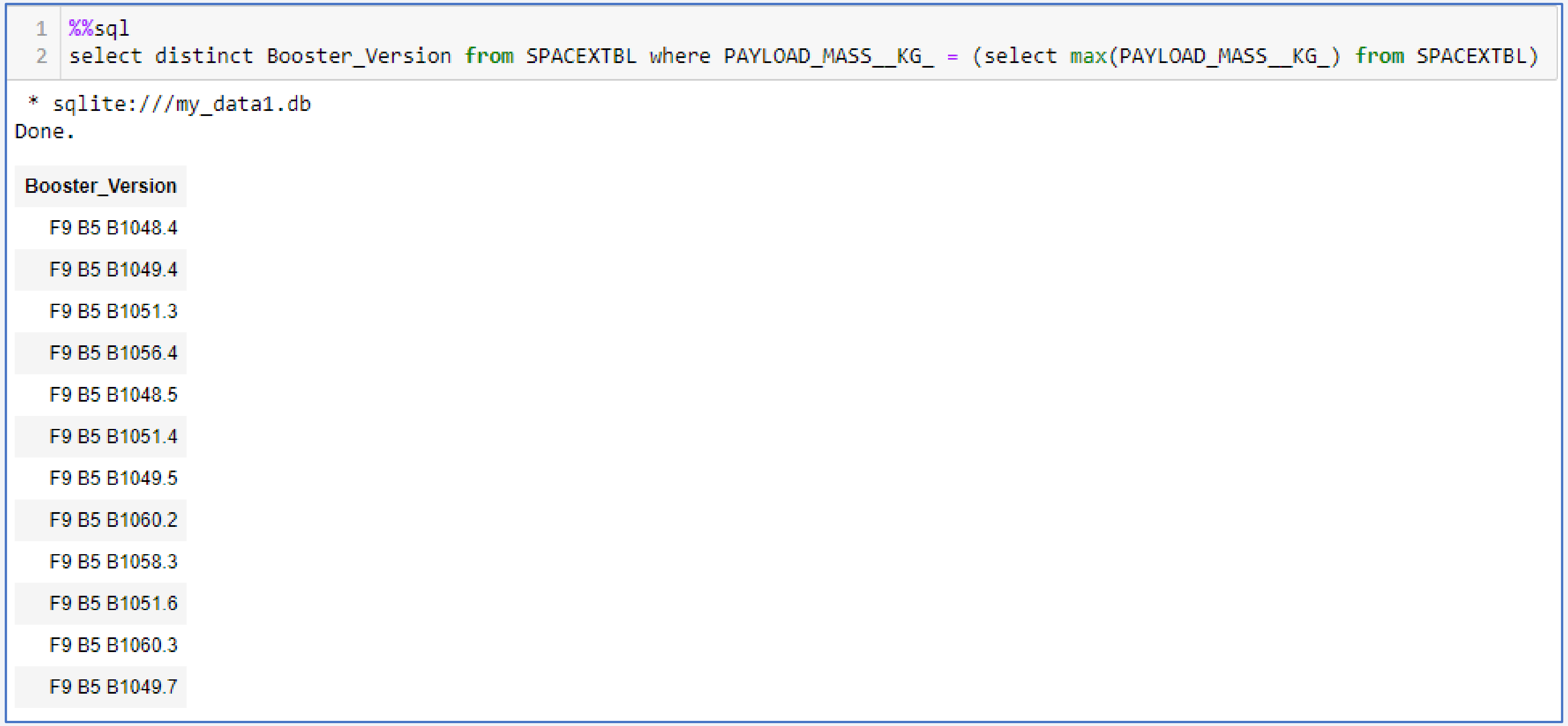
```
%%sql
select mission_outcome, count(mission_outcome) count from SPACEXTBL group by mission_outcome
```

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

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

- To calculate the total number of successful and failure mission outcomes we used count aggregate function of SQL and group by clause on mission outcome.

Boosters Carried Maximum Payload



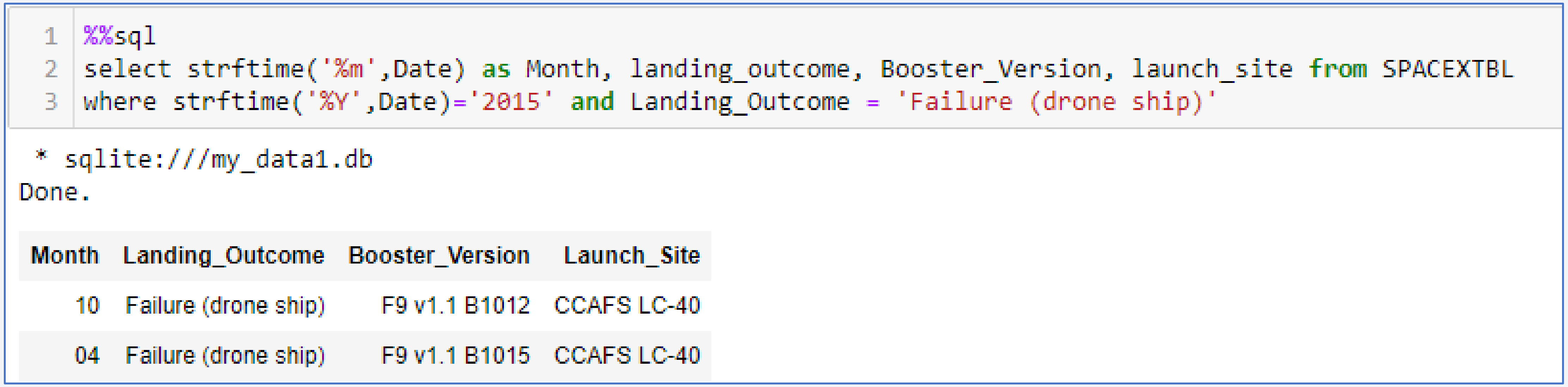
```
%%sql
select distinct Booster_Version from SPACEXTBL where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTBL)
```

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

Booster_Version
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

- There are total 12 boosters which carried maximum payload mass.
  To find out this we used sub-query to find out maximum payload mass. And the filtered all booters with maximum payload mass. Also used distinct key word to avoid duplicate result.

2015 Launch Records



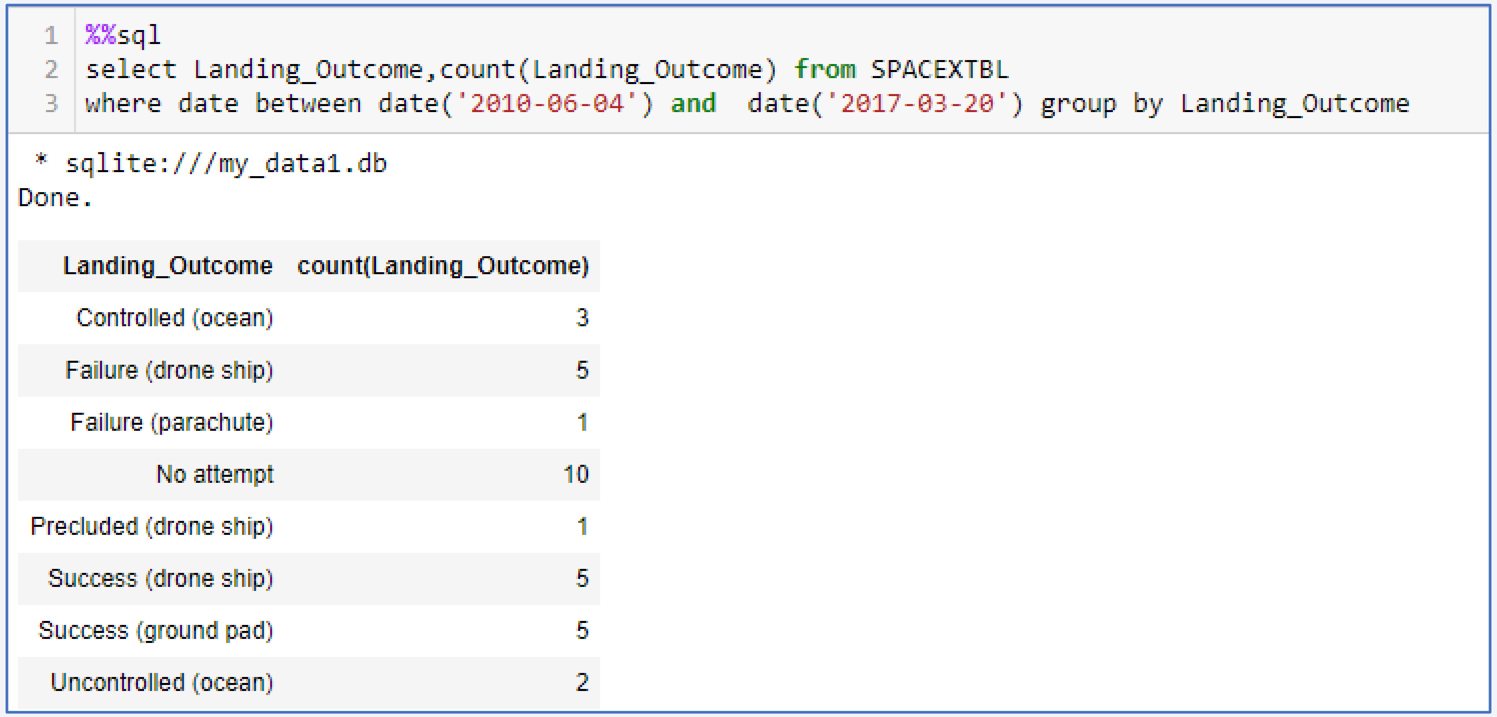
```
%%sql
select strftime('%m',Date) as Month, landing_outcome, Booster_Version, launch_site from SPACEXTBL
where strftime('%Y',Date)='2015' and Landing_Outcome = 'Failure (drone ship)'
```

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

Month	Landing_Outcome	Booster_Version	Launch_Site
10	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

- We used strftime() function of sql to extract month and year from date and compared that year with 2015 also used filter on landing outcome table.

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



```
%%sql
select Landing_Outcome,count(Landing_Outcome) from SPACEXTBL
where date between date('2010-06-04') and  date('2017-03-20') group by Landing_Outcome
```

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

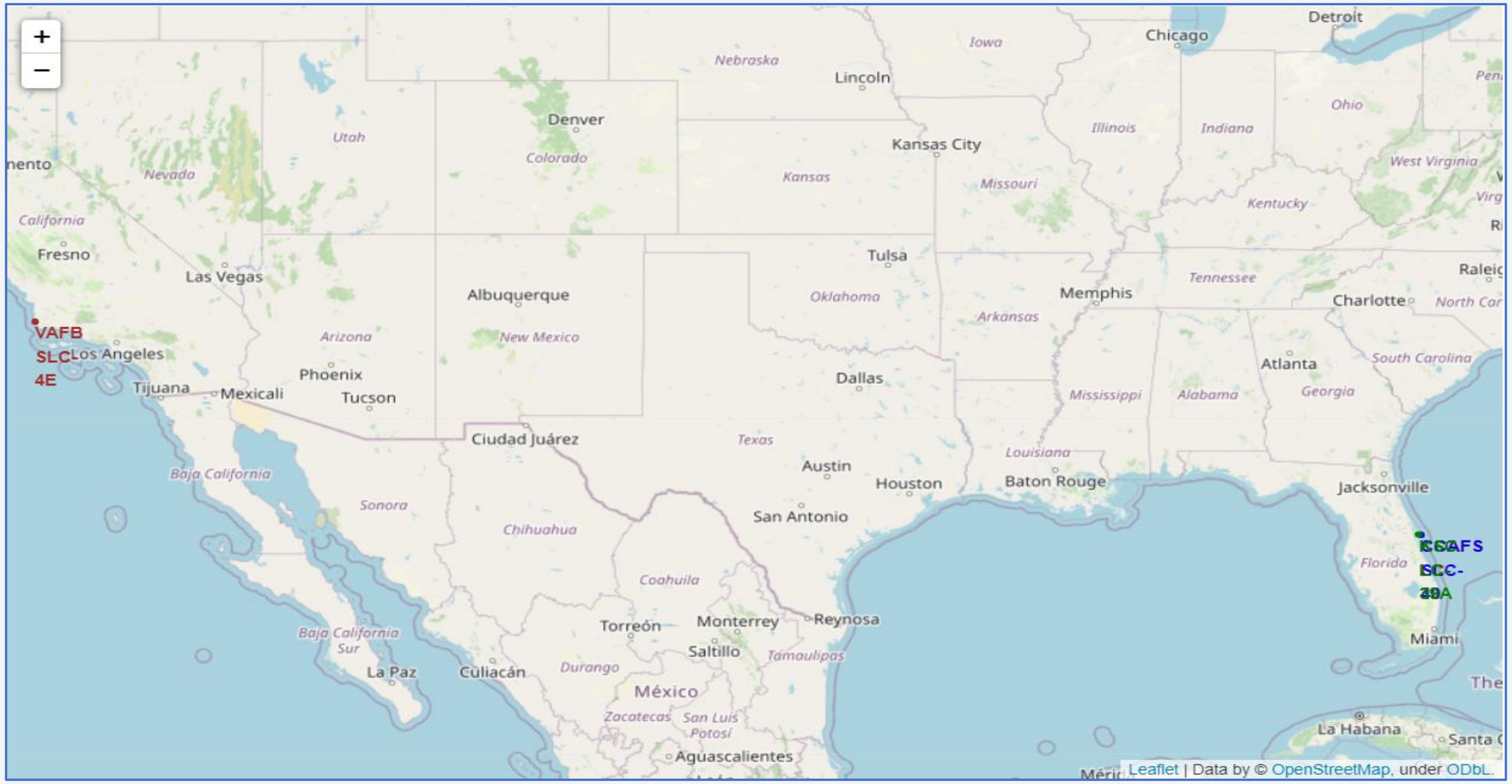
Landing_Outcome	count(Landing_Outcome)
Controlled (ocean)	3
Failure (drone ship)	5
Failure (parachute)	1
No attempt	10
Precluded (drone ship)	1
Success (drone ship)	5
Success (ground pad)	5
Uncontrolled (ocean)	2

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

Section 3

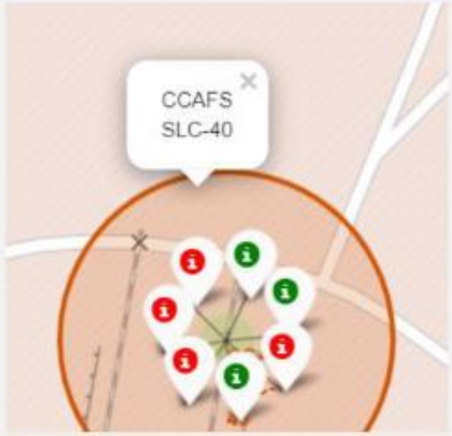
Launch Sites Proximities Analysis

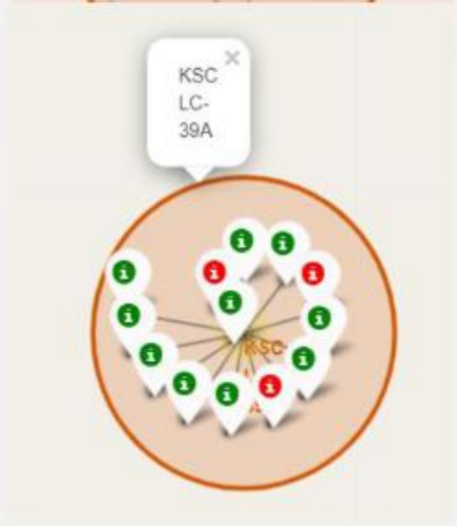
Folium Map: All launch Sites

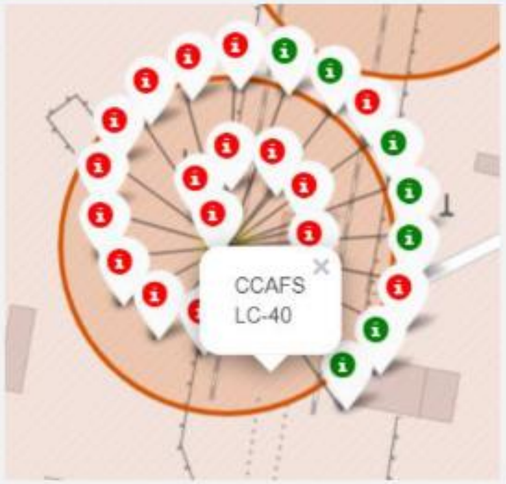


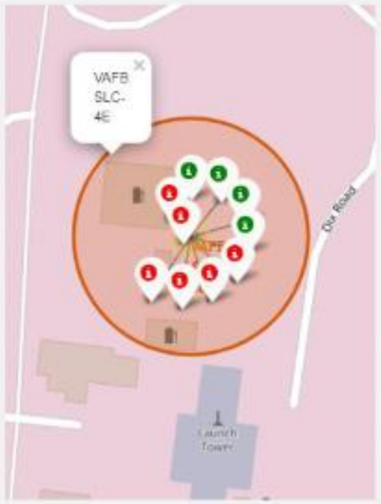
There are total four launch sites of SpaceX one present west coastline of America and other 3 present at east coastline

Folium Map: Launch Outcome









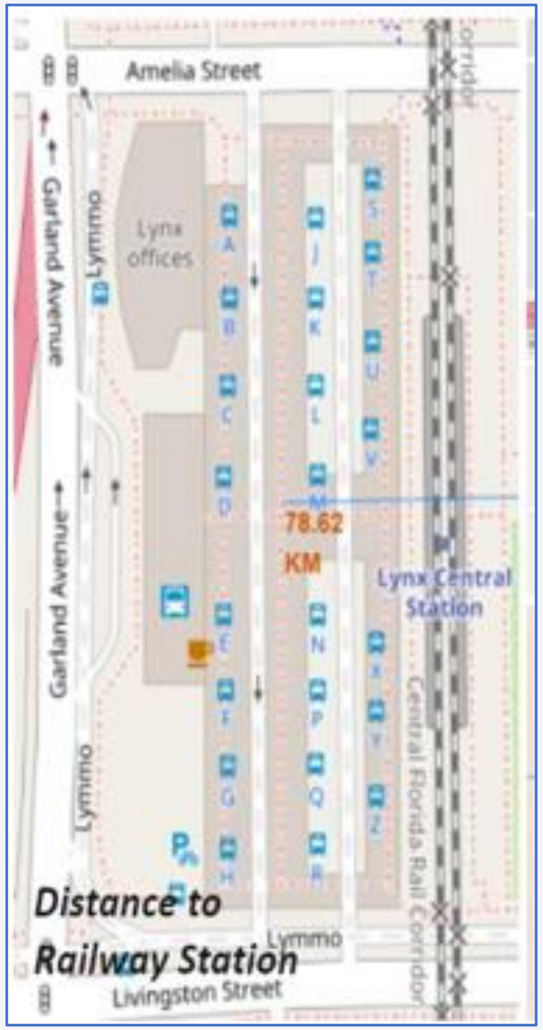
California Launch Site

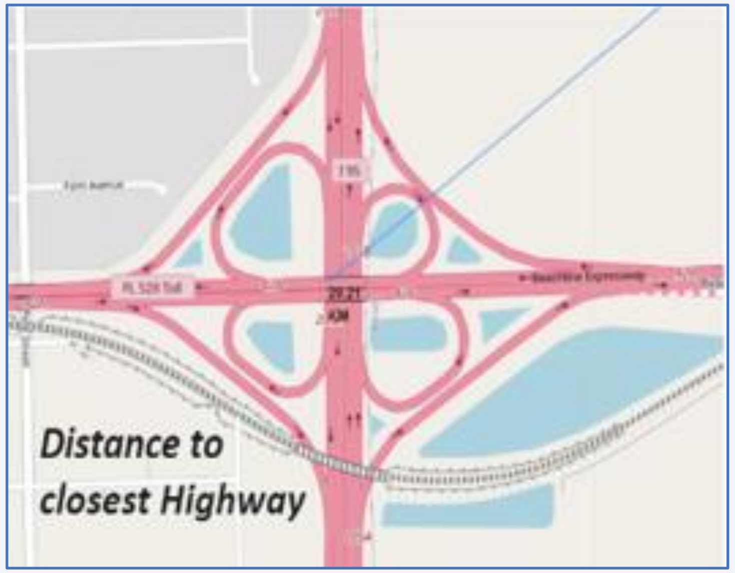
Florida Launch Sites

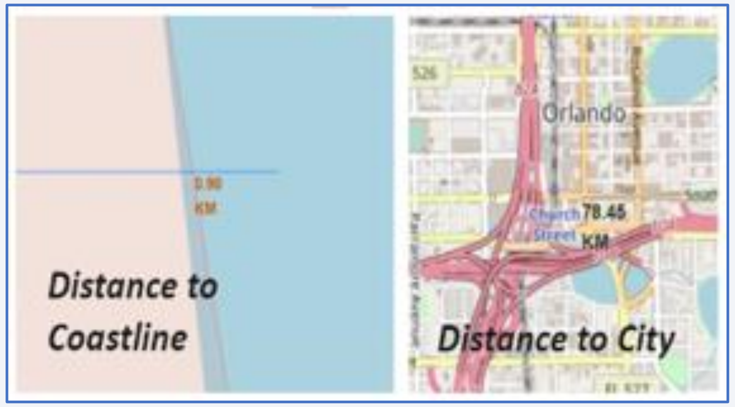
Green Marker shows successful Launches and Red Marker shows Failures

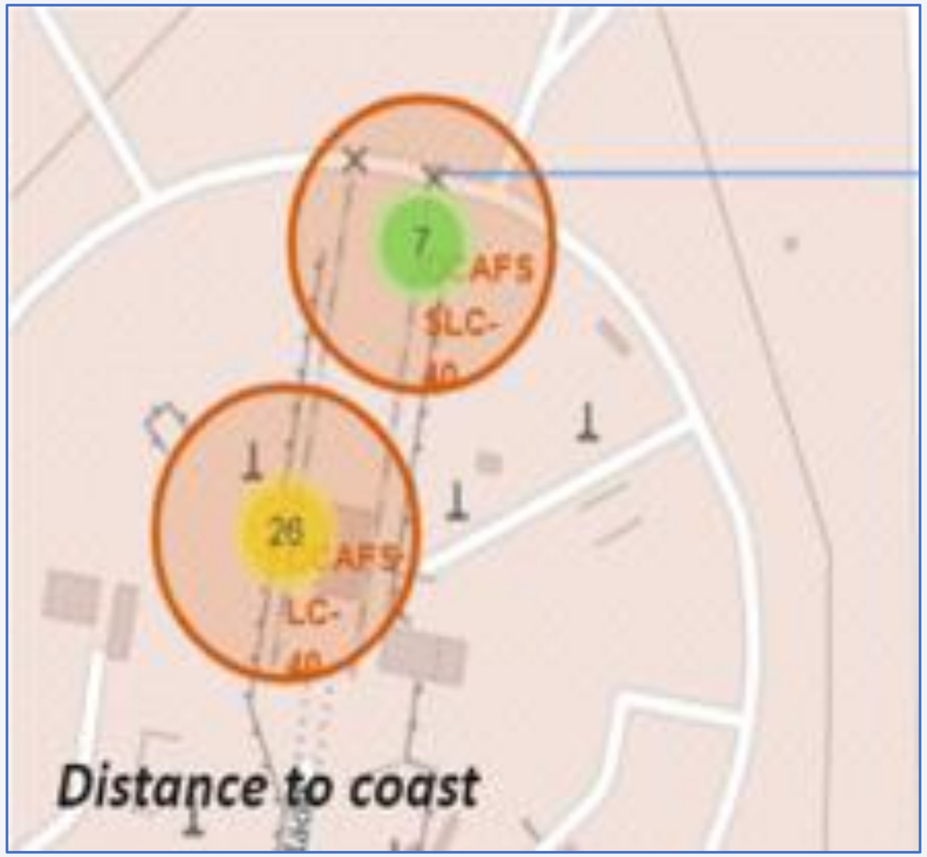
37

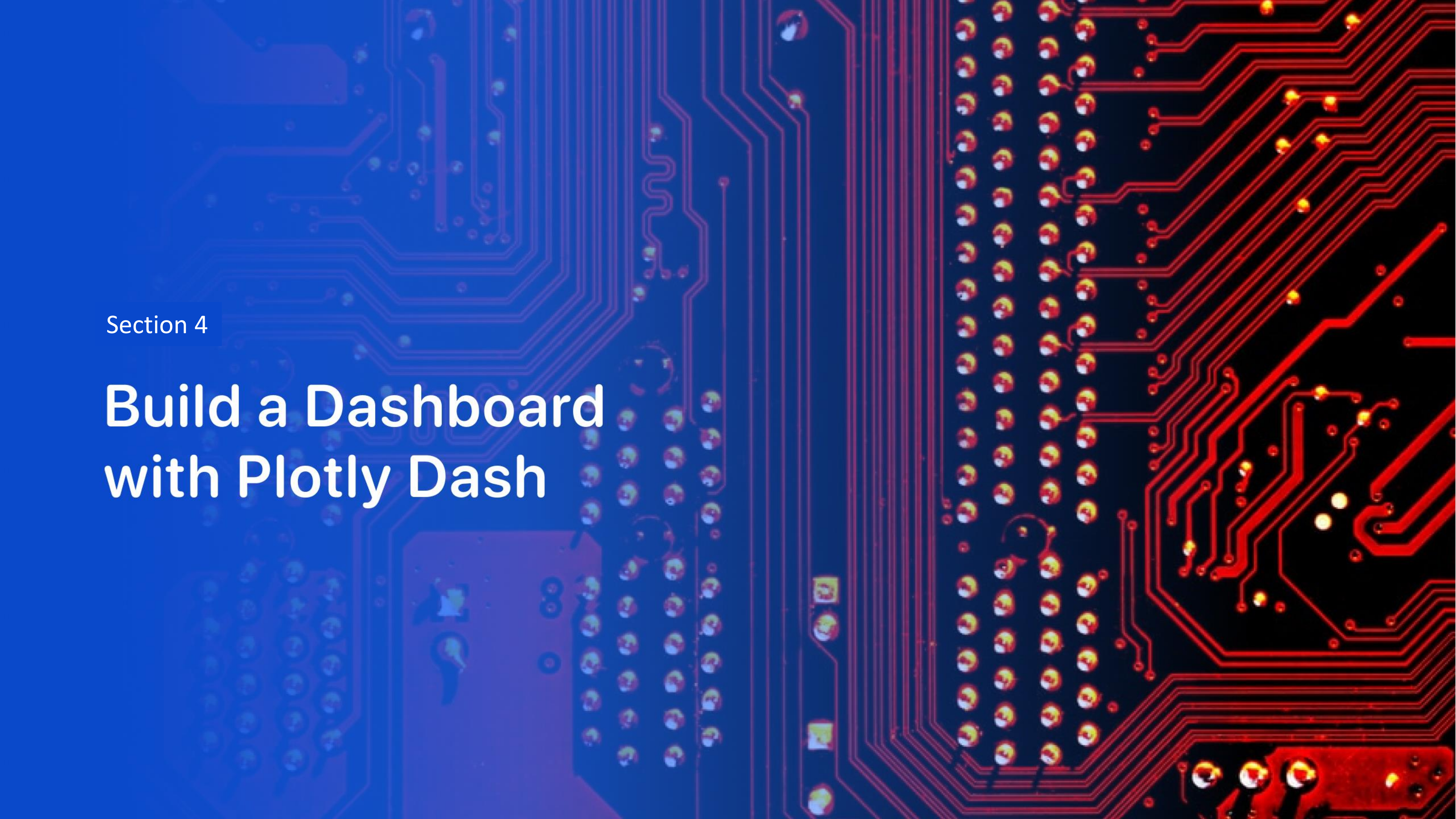
Folium Map: Launch site to its proximities







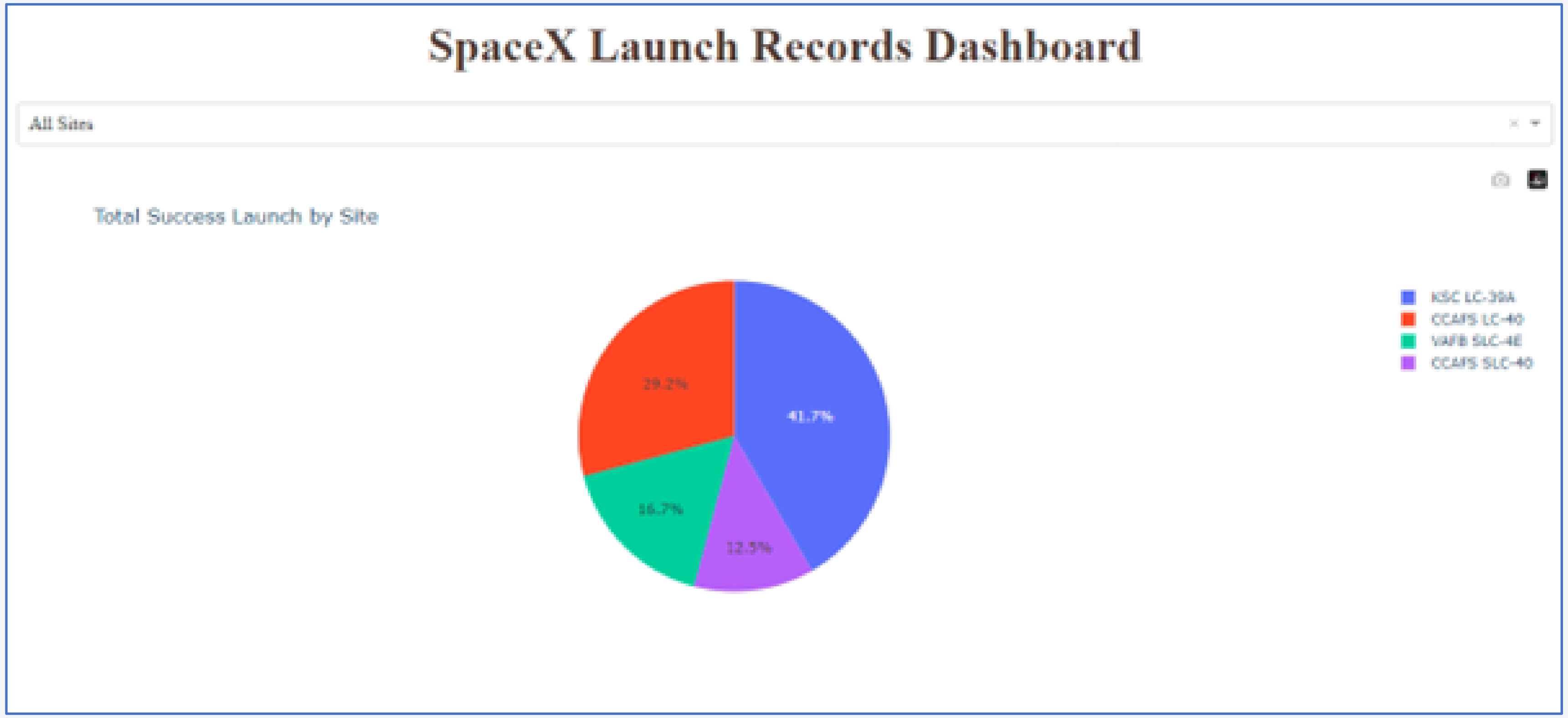




Section 4

Build a Dashboard with Plotly Dash

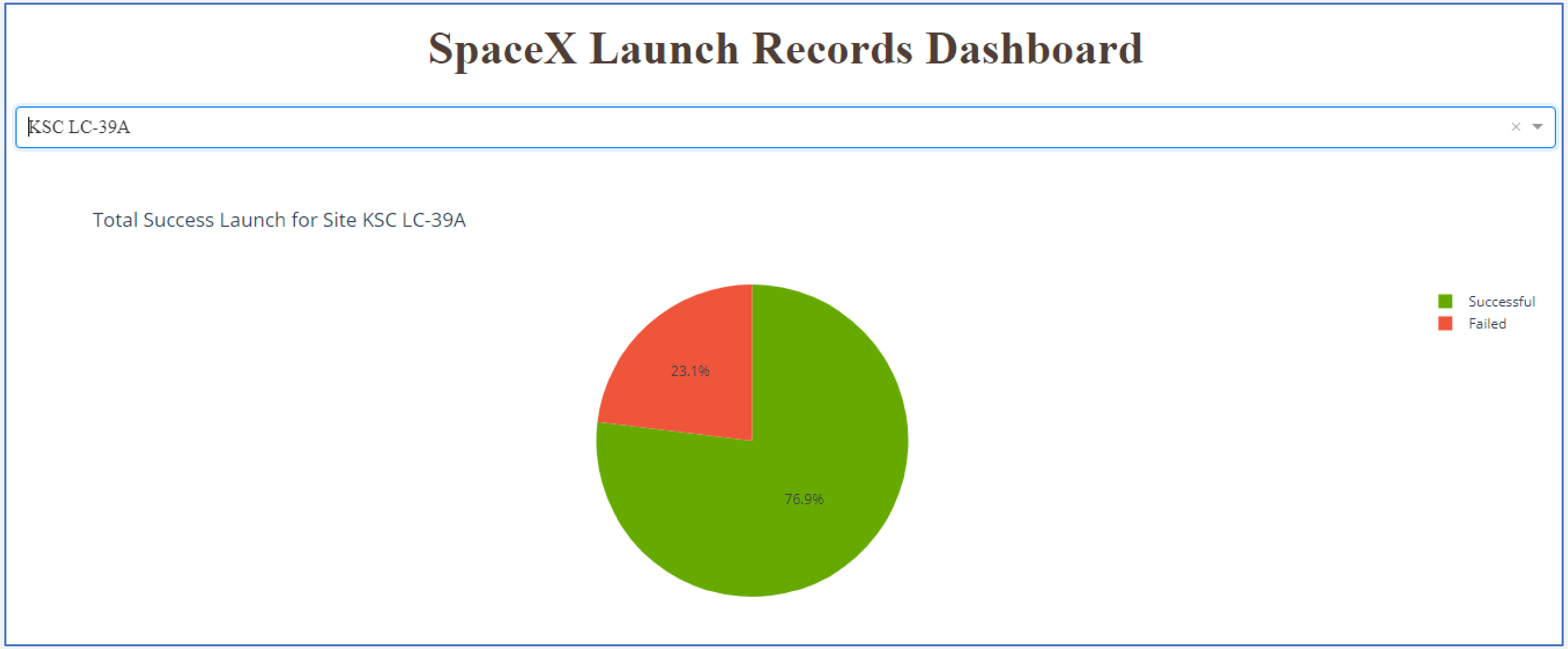
Pie chart: success percentage achieved by each launch site


SpaceX Launch Records Dashboard

All Sites

- KSC LC-39A has most successful launches from among all sites

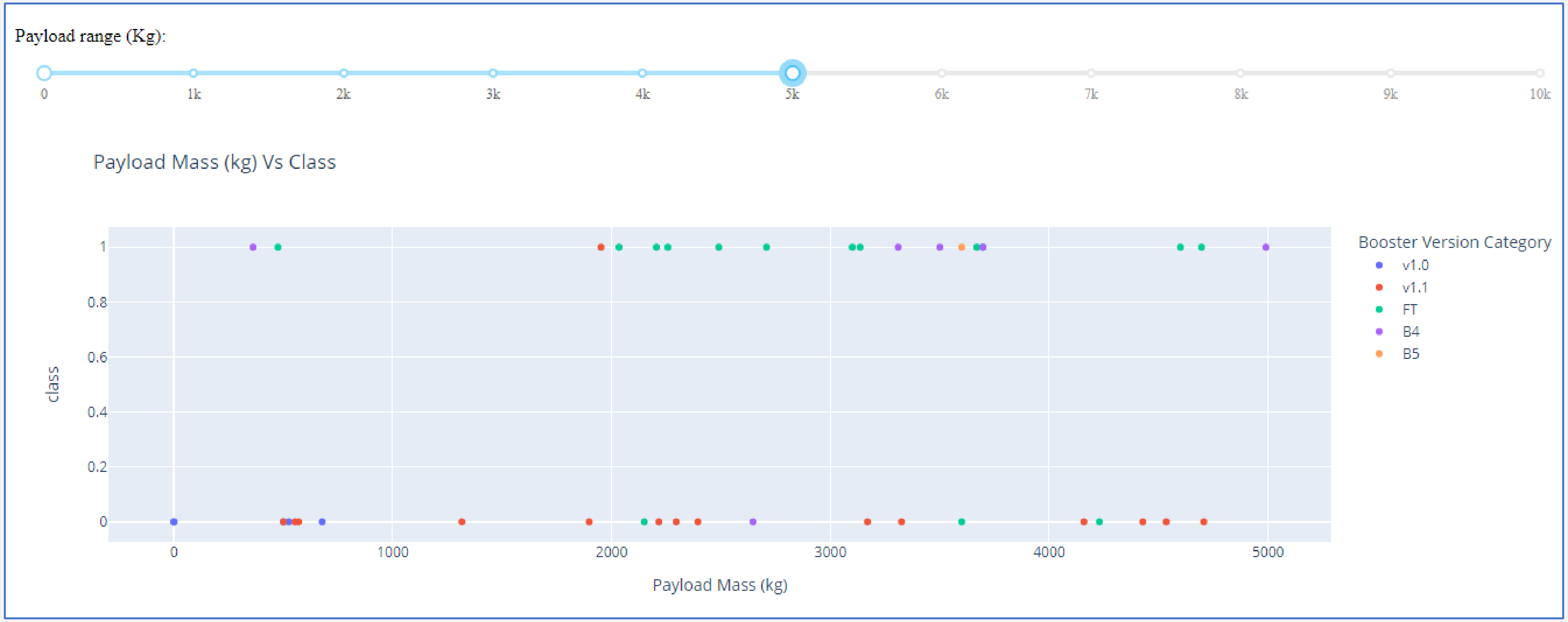
Pie chart: Launch site with the highest launch success ratio


SpaceX Launch Records Dashboard

KSC LC-39A

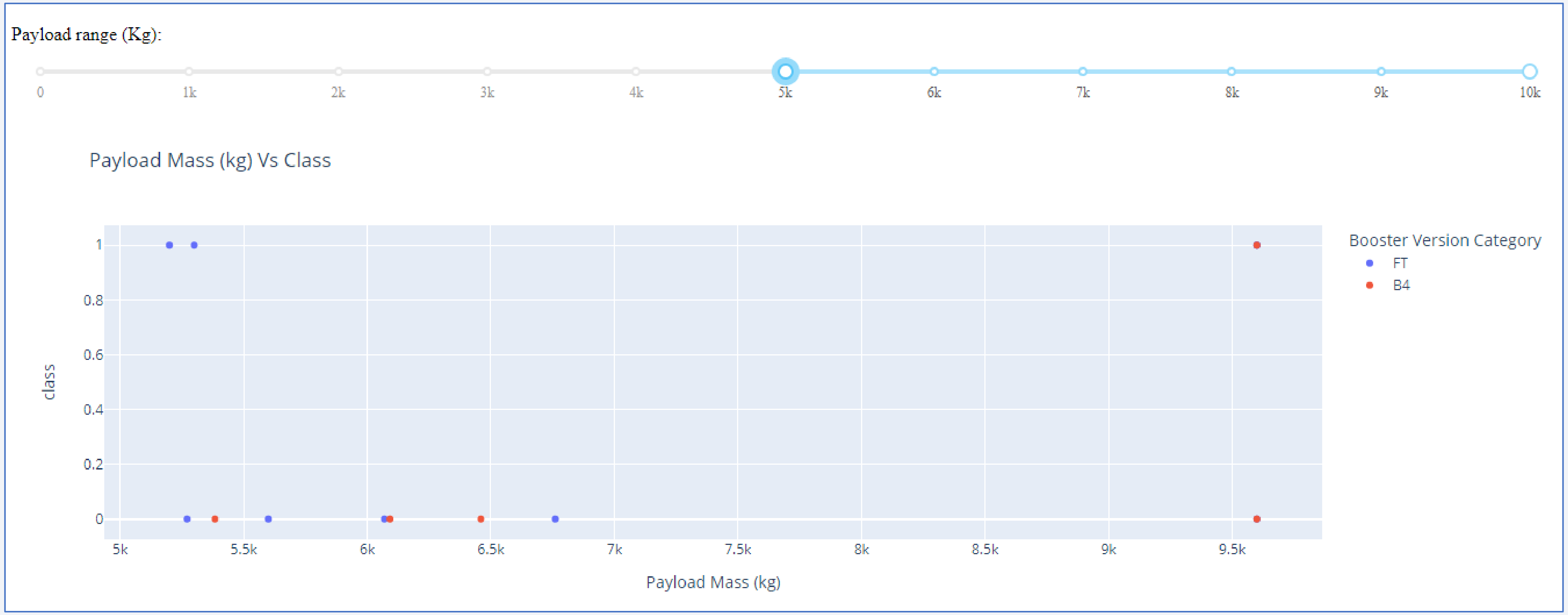
- KSC LC-39A achieved highest success rate of 76.9% and 23.1% failure rate

Scatter plot of Payload vs Launch Outcome



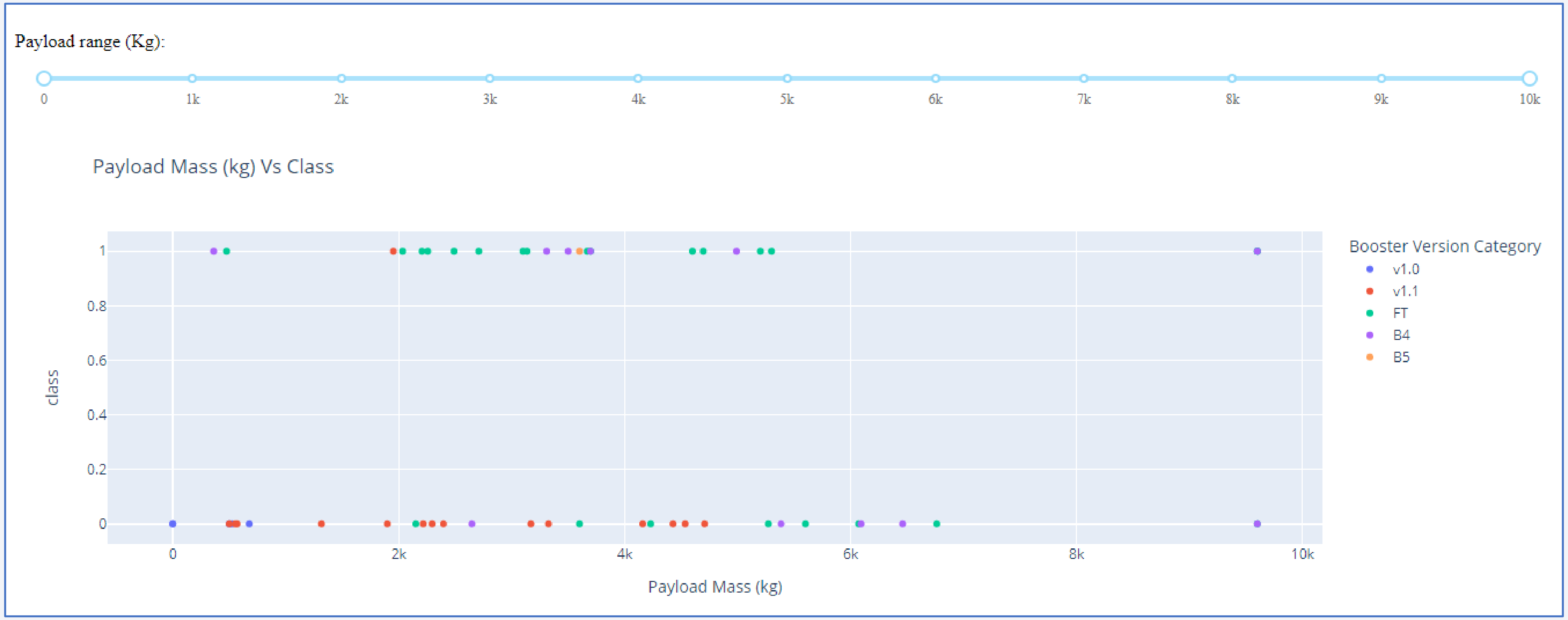
Payload range between 0 Kg to 5000 Kg

Scatter plot of Payload vs Launch Outcome



Payload range between 5000 Kg to 10000 Kg

Scatter plot of Payload vs Launch Outcome



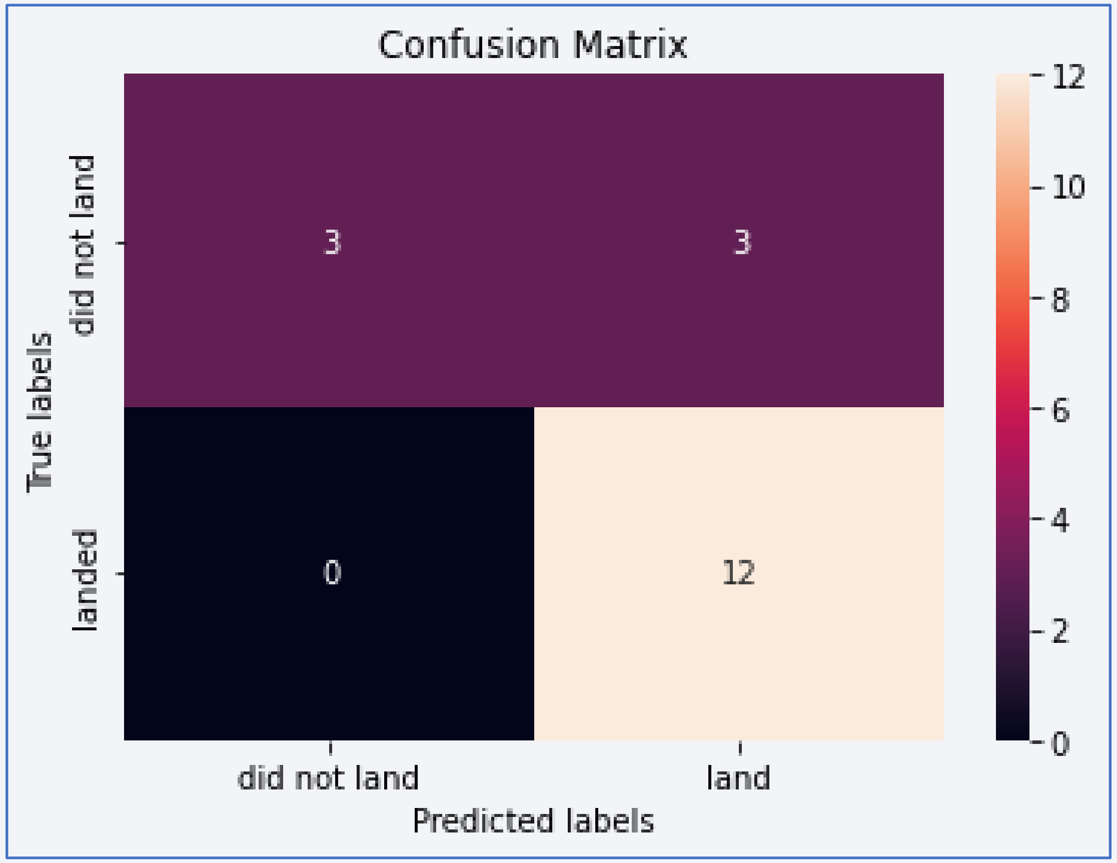
Payload range between 0 Kg to 10000 Kg

Section 5

Predictive Analysis (Classification)

Confusion Matrix

- Best performing model is Decision Tree with accuracy of 87%
- 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.

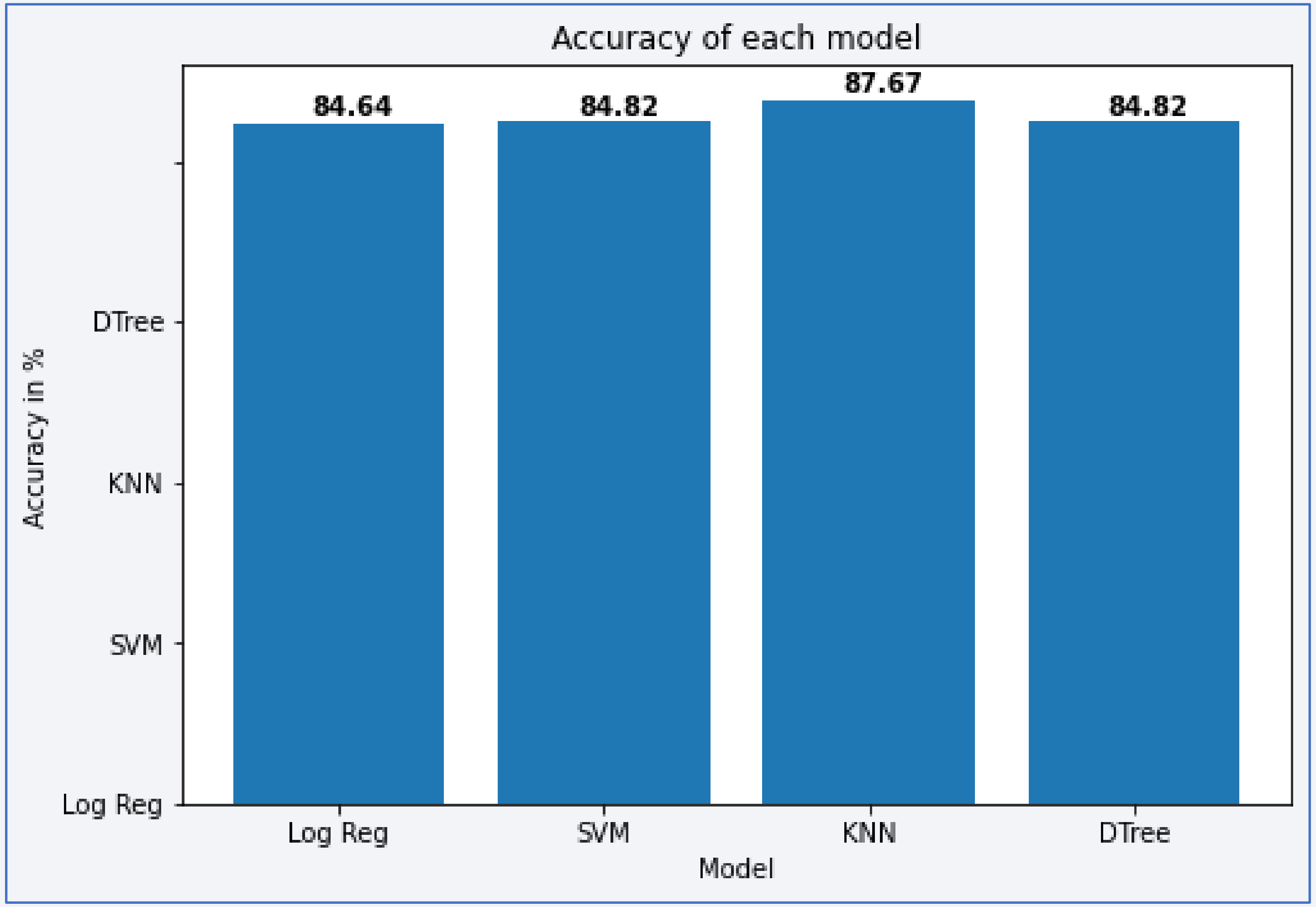


Conclusions

We can conclude that:

- As the flight number increases, the first stage is more likely to land successfully.
- It seems the more massive the payload, the less likely the first stage will return.
- With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.
- Success rate since 2013 kept increasing till 2020
- Orbits ES-L1, GEO, HEO, SSO, VLEO has the most success rate.
- KSC LC-39A has the most successful launches of any sites.
- The Decision tree classifier is the best machine learning algorithm for this task.

Classification Accuracy



Decision Tree model has highest accuracy of 87.67%

Thank you!