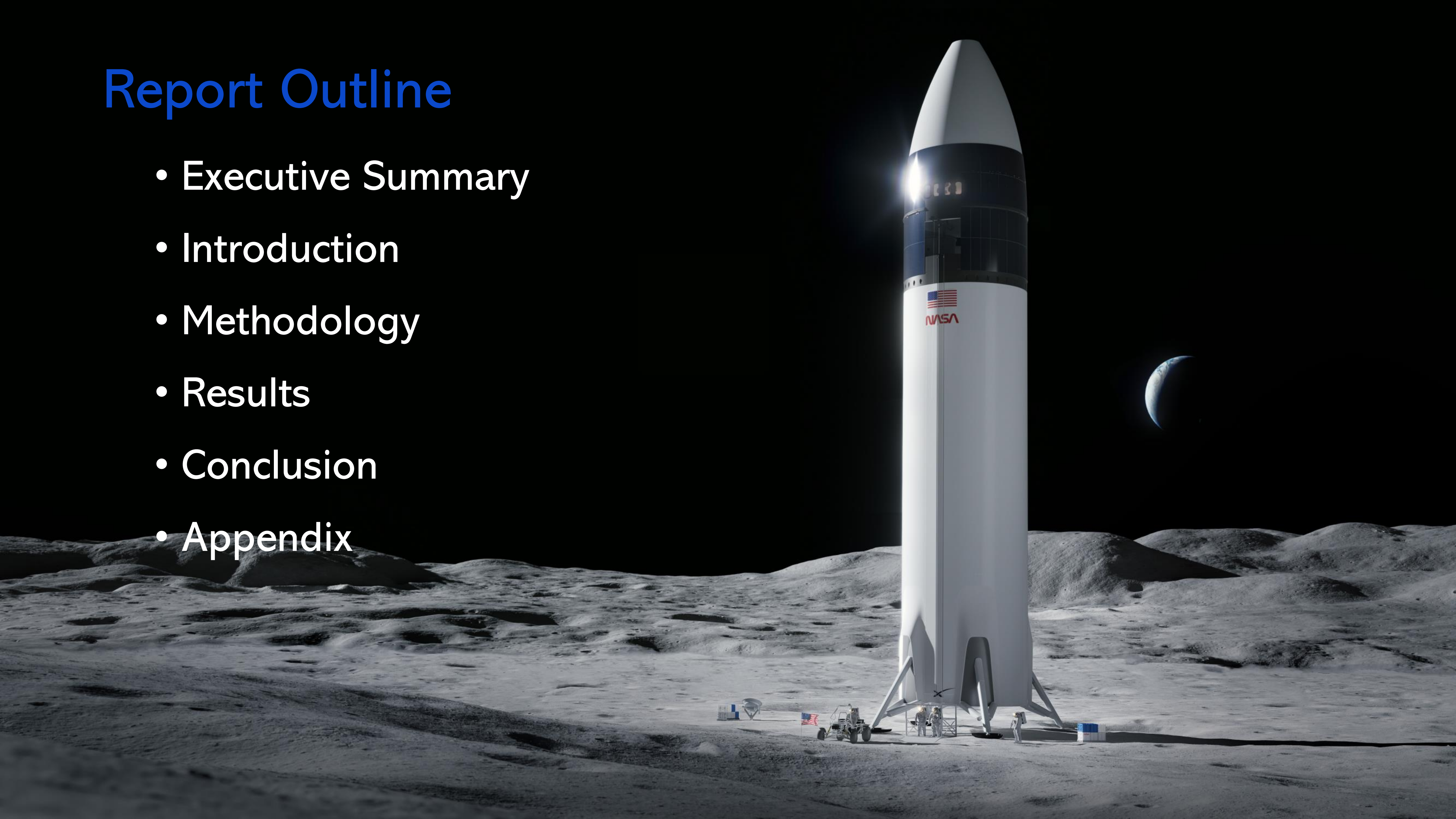
IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

THET ZIN
27/10/2021



Report Outline

Executive Summary

- Summary of 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)
- Summary of all results
  - Exploratory data analysis results
  - Interactive analytics demo in screenshots
  - Predictive analysis results

Introduction

SpaceX advertises Falcon 9 rocket launches cost much lower than other providers because SpaceX can reuse their first stage. In this project, we are predicting if the Falcon 9 first stage will land successfully. If we can determine if the first stage will land, the cost of a launch can be significantly lower.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - SpaceX API
  - Web Scraping Wikipedia
- Perform data wrangling
  - Drop irrelevant columns from the dataset and One-hot-encode categorical data
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Explore dataset using Scatter Plots, Bar Chats, Line Chats and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

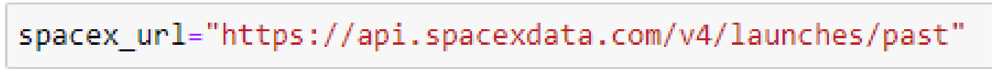
Data Collection Methodology

- SpaceX API
  - Data is collected from SpaceX API using the url "*https://api.spacexdata.com/v4/launches/past*"
  - API returns the JSON results which are turned into Pandas Dataframe
  - Requested data cleaned and exported into CSV file
- Web Scraping Wikipedia
  - Extract a Falcon 9 launch records HTML table from Wikipedia
  - Parse the table using BeautifulSoup and convert it into a Pandas Data Frame
  - Data Frame exported as CSV file

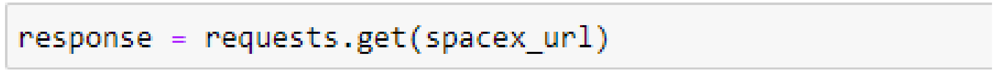
Data Collection – SpaceX API Process Flow

GitHub link to Notebook

1. Getting response from API

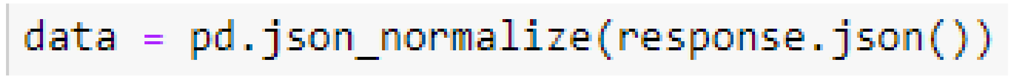


```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```



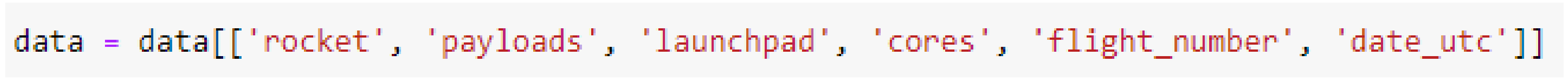
```
response = requests.get(spacex_url)
```

2. Response content is decoded as Json using .json() and turned into Pandas Dataframe using .json_normalize()



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

3. A subset of the dataframe is taken and only the features we want are kept



```
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]
```

Data Collection – SpaceX API Process Flow

4. Predefined helper functions will help to extract data from API and populate into lists. These lists are then turned into dictionary and then to Pandas Dataframe

`getBoosterVersion(data)`

Uses ‘rocket’ column and append to ‘BoosterVersion’

`getPayloadData(data)`

Uses ‘payloads’ column and append to ‘PayloadMass’ and ‘Orbit’

`getLaunchSite(data)`

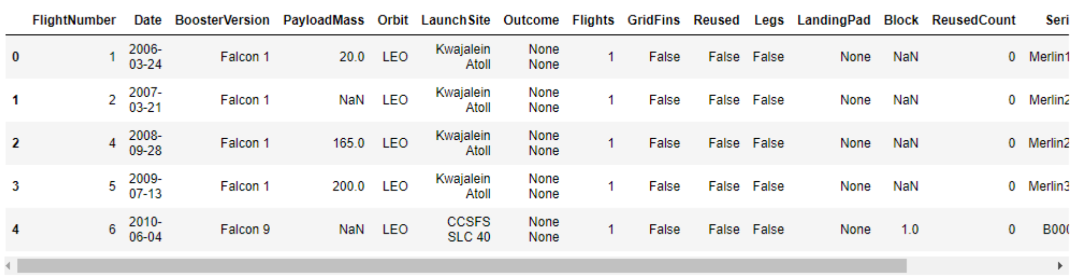
Uses ‘launchpad’ column and append to ‘Longitude’, ‘Latitude’ and ‘LaunchSite’

`getCoreData(data)`

Uses ‘core’ column and append to ‘Outcome’, ‘Flights’, ‘GridFins’, ‘Reused’, ‘Legs’ and ‘LandingPad’

```
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

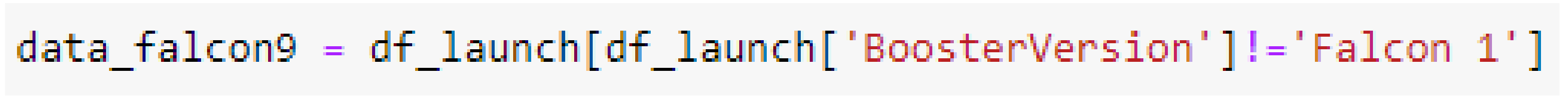
```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,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



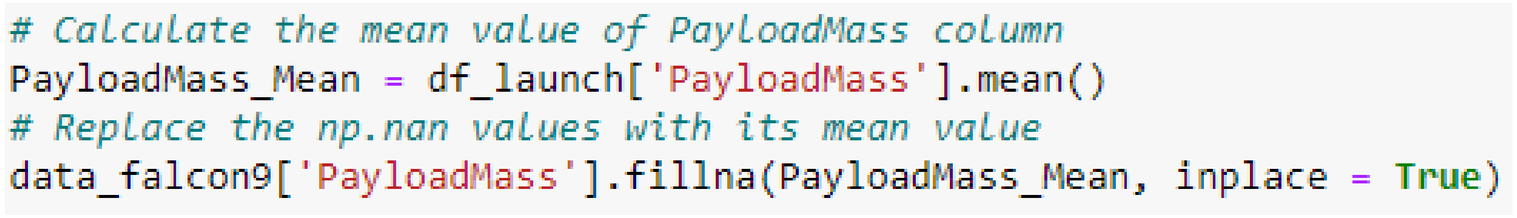
	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Seri
0	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin1
1	2	2007-03-21	Falcon 1	NaN	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2
2	4	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2
3	5	2009-07-13	Falcon 1	200.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin3
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B00

Data Collection – SpaceX API Process Flow

5. Filter the dataframe to only include Falcon 9 launches and missing values are handled. Missing values in ‘PayloadMass’ column are replaced with mean value

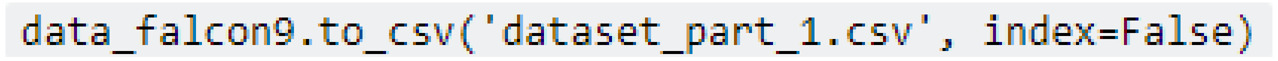


```
data_falcon9 = df_launch[df_launch['BoosterVersion']!='Falcon 1']
```



```
# Calculate the mean value of PayloadMass column
PayloadMass_Mean = df_launch['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].fillna(PayloadMass_Mean, inplace = True)
```

6. The dataframe is then exported as CSV file

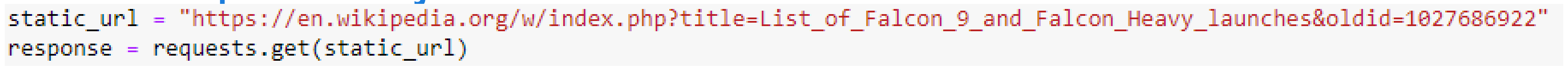


```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection – Scraping Process Flow

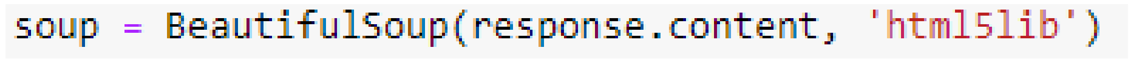
Github link to notebook

1. Get a response object from the static URL



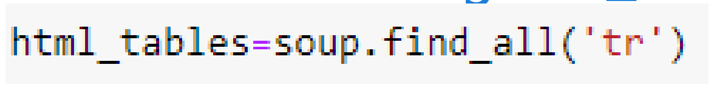
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
response = requests.get(static_url)
```

2. Create a BeautifulSoup object from the response



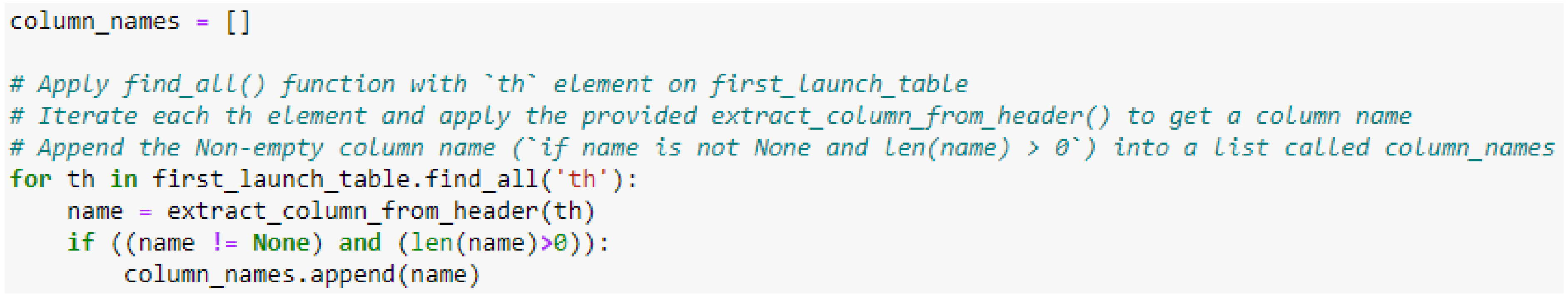
```
soup = BeautifulSoup(response.content, 'html5lib')
```

3. Extract tables using find_all



```
html_tables=soup.find_all('tr')
```

4. Column names are extracted using predefined function

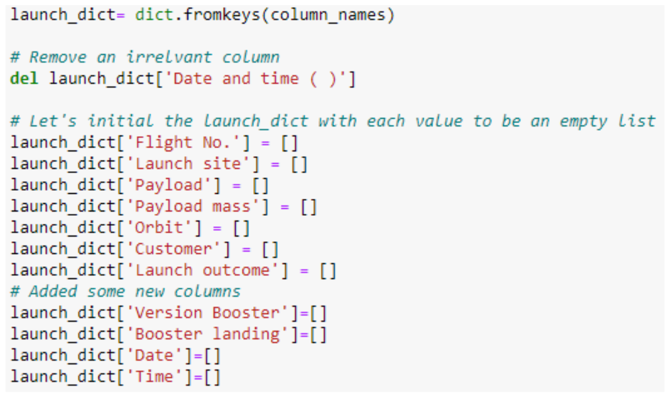


```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
for th in first_launch_table.find_all('th'):
    name = extract_column_from_header(th)
    if ((name != None) and (len(name)>0)):
        column_names.append(name)
```

Data Collection – Scraping Process Flow

5. Create empty dictionary with keys from the extracted column names

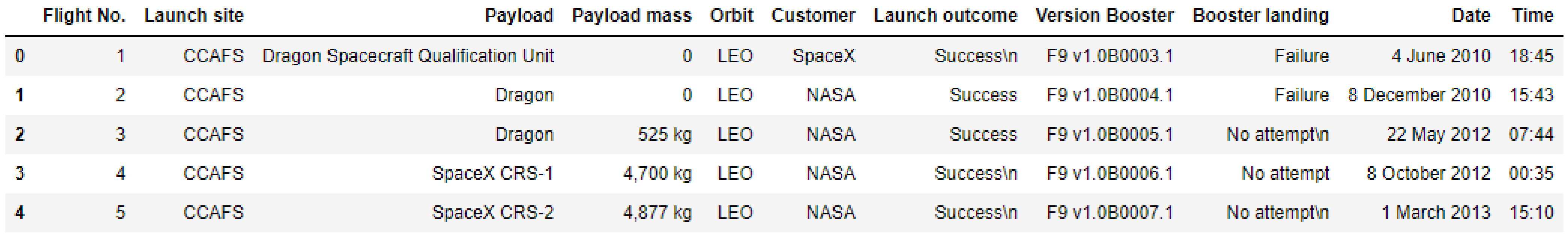


```
launch_dict= dict.fromkeys(column_names)

# 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. Fill up the dictionary with records extracted from table rows
7. Data Frame is created from the launch_dict and exported as CSV file

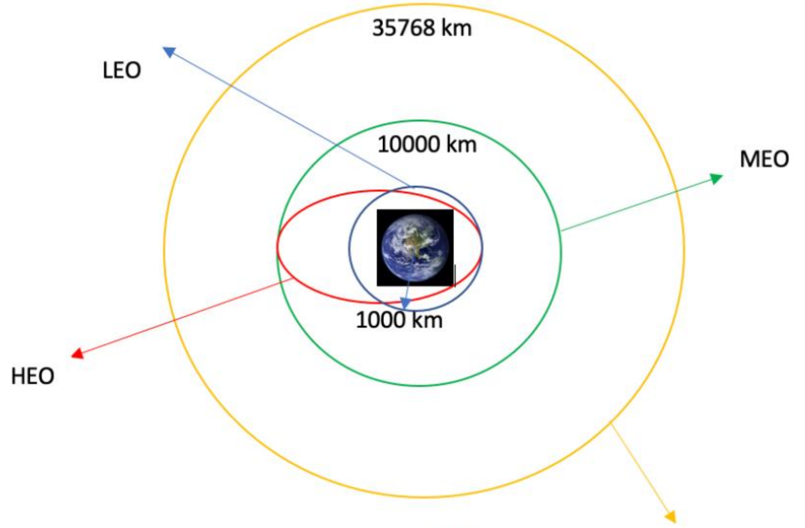


	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	F9 v1.0B0003.1	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	NASA	Success	F9 v1.0B0005.1	No attempt\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success\n	F9 v1.0B0006.1	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success\n	F9 v1.0B0007.1	No attempt\n	1 March 2013	15:10

Data Wrangling

GitHub link to Notebook

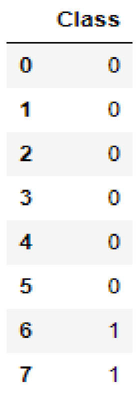
- In this section, we are converting landing outcomes into Training Labels with “1” means the booster successfully landed “0” means it was unsuccessful.
- In the dataset under “Outcome” column, True Ocean means the mission outcome was successfully landed to a specific region of the ocean while False Ocean means the mission outcome was unsuccessfully landed to a specific region of the ocean. True RTLS means the mission outcome was successfully landed to a ground pad False RTLS means the mission outcome was unsuccessfully landed to a ground pad. True ASDS means the mission outcome was successfully landed on a drone ship False ASDS means the mission outcome was unsuccessfully landed on a drone ship.



Common Orbit Types

Process Flow To Perform EDA On Dataset

Calculate the number of launches on each site → Calculate the number and occurrence of each orbit → Calculate the number and occurrence of mission outcome per orbit type → Create a landing outcome label from Outcome column →



	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

1 – successful
0 – not successful

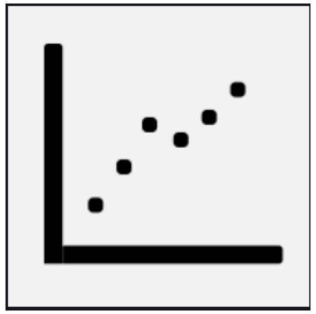
EDA with Data Visualization

[Github Link to Notebook]

Scatter Plots

- Flight Number vs Payload Mass
- Flight Number vs Launch Site
- Payload vs Launch Site
- Orbit vs Flight Number
- Payload vs Orbit Type
- Orbit vs Payload Mass

From scatter plots, we can observe relationships between two numeric variables. The dots in a scatter plot not only report the values of individual data points, but also patterns when the data are taken as a whole. Identification of correlational relationships are common with scatter plots.



Bar Chart

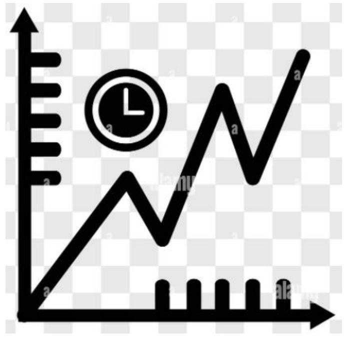
- Success Rate vs Orbit

Using bar charts, we can compare values between different groups. In our assignment, bar chart is used for visualizing success rate of each orbit.

Line Chart

- Year vs Average Success Rate

Line charts are useful in that they show data variables and trends very clearly and can help to make predictions about the results of data not yet recorded. In our case, it is used to visualize the trend of average success rate over years.



EDA with SQL

Github Link to Notebook

SQL queries are performed in the python notebook to understand more about the SpaceX data set. Following tasks are performed using SQL queries.

- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome in ground pad was achieved
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
- 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

Build an Interactive Map with Folium

GITHUB URL FOR NOTEBOOK

- In this Launch Site Locations Analysis, a folium map is initialized, and a circle is drawn using latitude and longitude coordinates of each launch site from spacex_launch_geo.csv. Then each launch site is labelled.

- The map is then enhanced by adding launch outcomes for each site. Using data from the class column of spacex_df markers for all launch records are created . If a launch was successful (class=1), a GREEN marker is added and if a launch was failed, a RED marker is added. MarkerCluster object is created to simplify a map containing many markers having the same coordinate.

- Haversine formula is used to calculate the distance between two coordinates. Lines are drawn from a selected launch site to various landmarks such as railways, highways, coastlines, cities etc., to measure the distance between them. Distance is then displayed.

Build a Dashboard with Plotly Dash

Github Link to Souce Code

- This dashboard application contains input components such as a dropdown list and a range slider to interact with a pie chart and a scatter point chart.

- From the dropdown list, success rate for all the launch sites together and for individual sites can be visualized using pie charts.

- The slider interacts with Scatter plot which shows the relationship with Outcome and Payload Mass (Kg) for the different Booster Versions

Predictive Analysis (Classification)

Predictive Analysis (Machine Learning) is done in 3 steps.

GITHUB URL FOR NOTEBOOK

1. Data Preprocessing
   - Importing necessary libraries
   - Load the data frame
   - Apply Feature Scaling on the data set (Standardization)
   - Splitting data set into training data and test data

2. Training Models
   - Create machine learning algorithm object
   - Find the best parameters using GridSearchCV

3. Evaluating Models
   - Calculate the algorithm's accuracy
   - Plot Confusion Matrix

Results

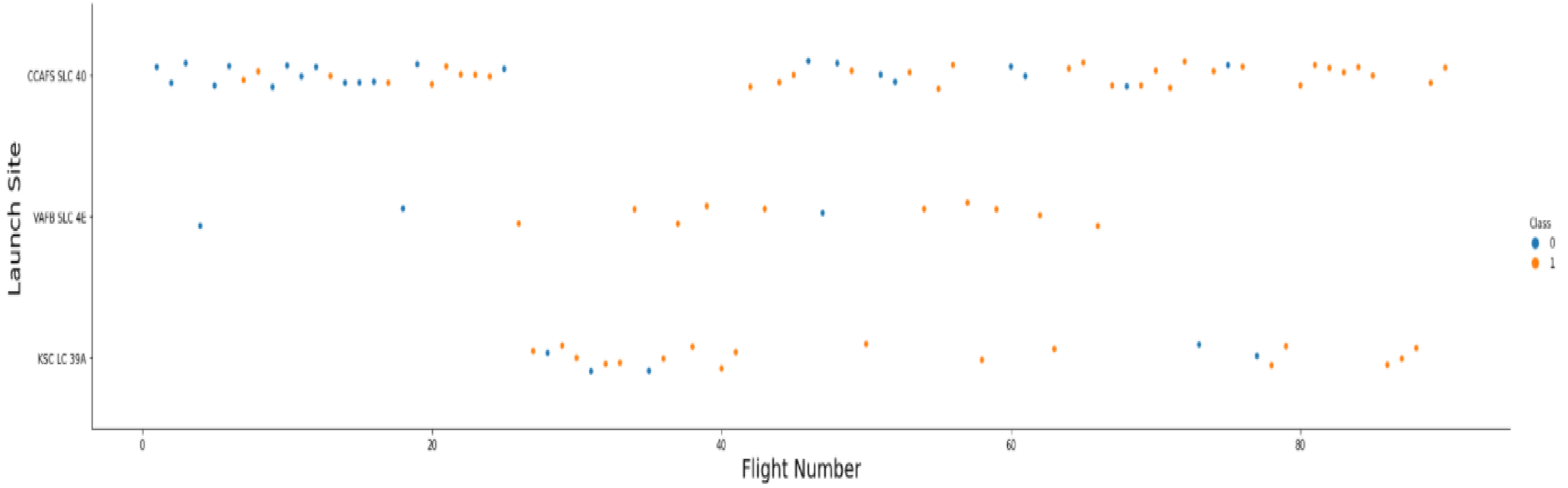
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

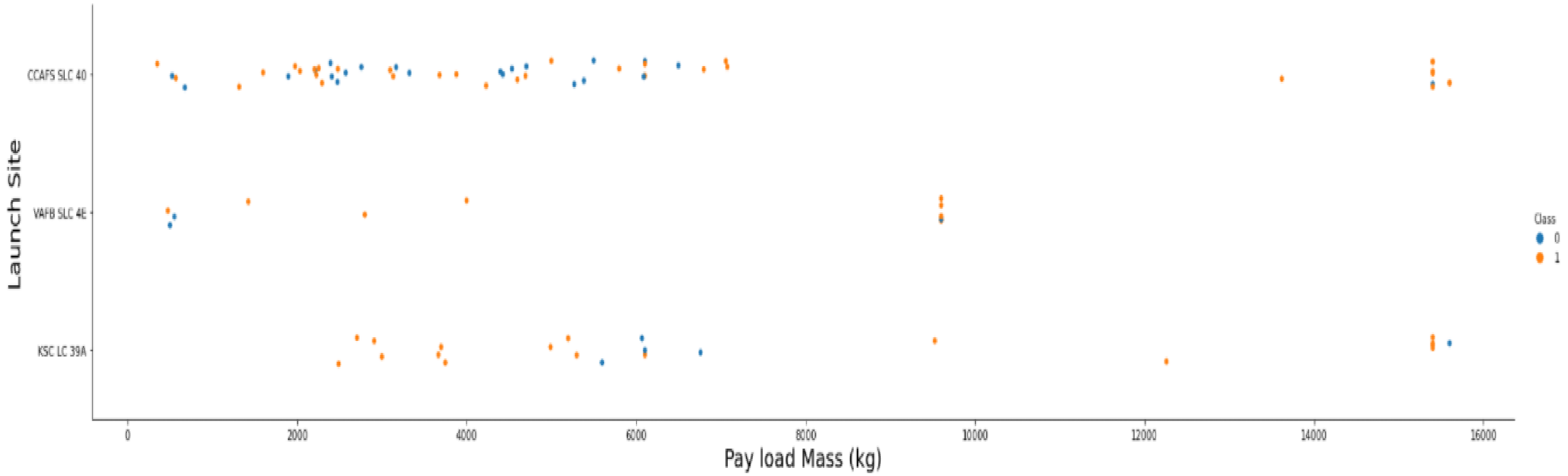
Flight Number vs. Launch Site

In the Flight Number vs Launch Site plot, success rate increases clearly with the increased number of flights for launch site CCAFS SLC 40. Similar trend can be seen for the launch site VAFB SLC 4E as well but for KSC LC 39A, the correlation between success rate and number of flight seems weaker than other launch sites.



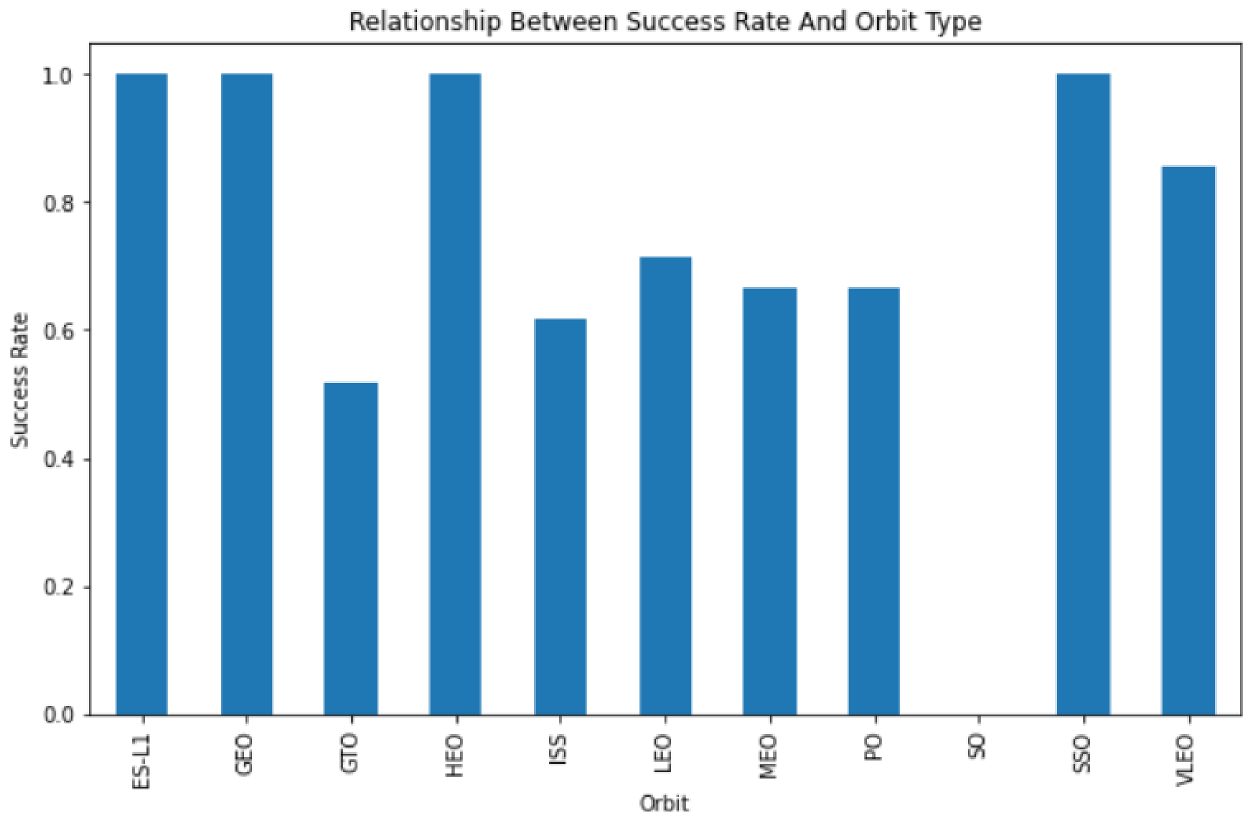
Payload vs. Launch Site

In the Pay load Mass vs Launch Site plot, success rate for larger payload above 12000kg for launch site CCAFS SLC 40. Similar pattern can be seen for the launch site VAFB SLC where payload is less than 10000kg for all the launches. For KSC LC 39A, the success rate of payload between 6000kg and 8000kg is very low.



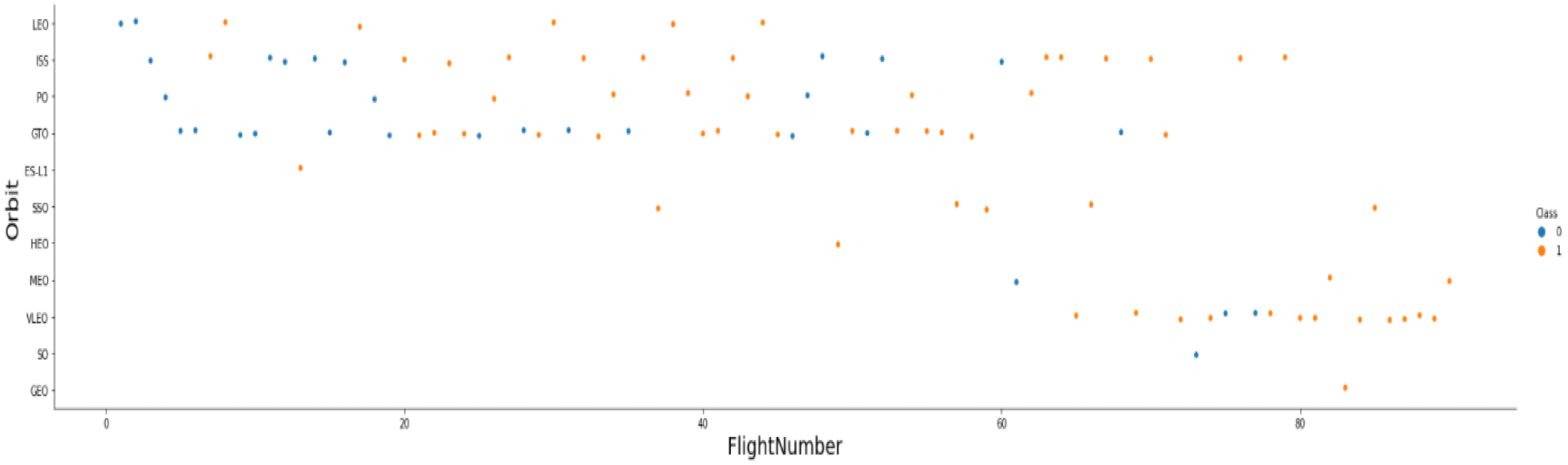
Success Rate vs. Orbit Type

Following bar chart Success Rate vs Orbit shows the relationship between success rate and orbit type. From the chart, we can observe launches to orbits such as ES-L1, GEO, HEO and SSO have the highest success rates.



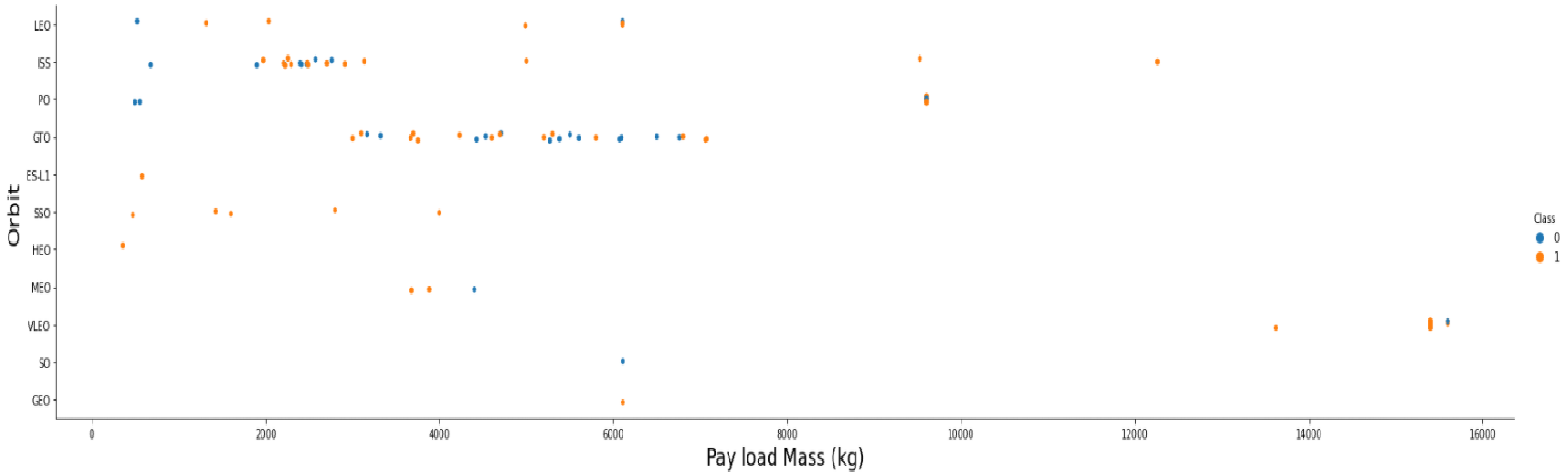
Flight Number vs. Orbit Type

We can see in the graph Flight Number vs Orbit Type that in the LEO orbit the Success appears related to the number of flights, however, there seems to be no relationship between flight number with other orbits.



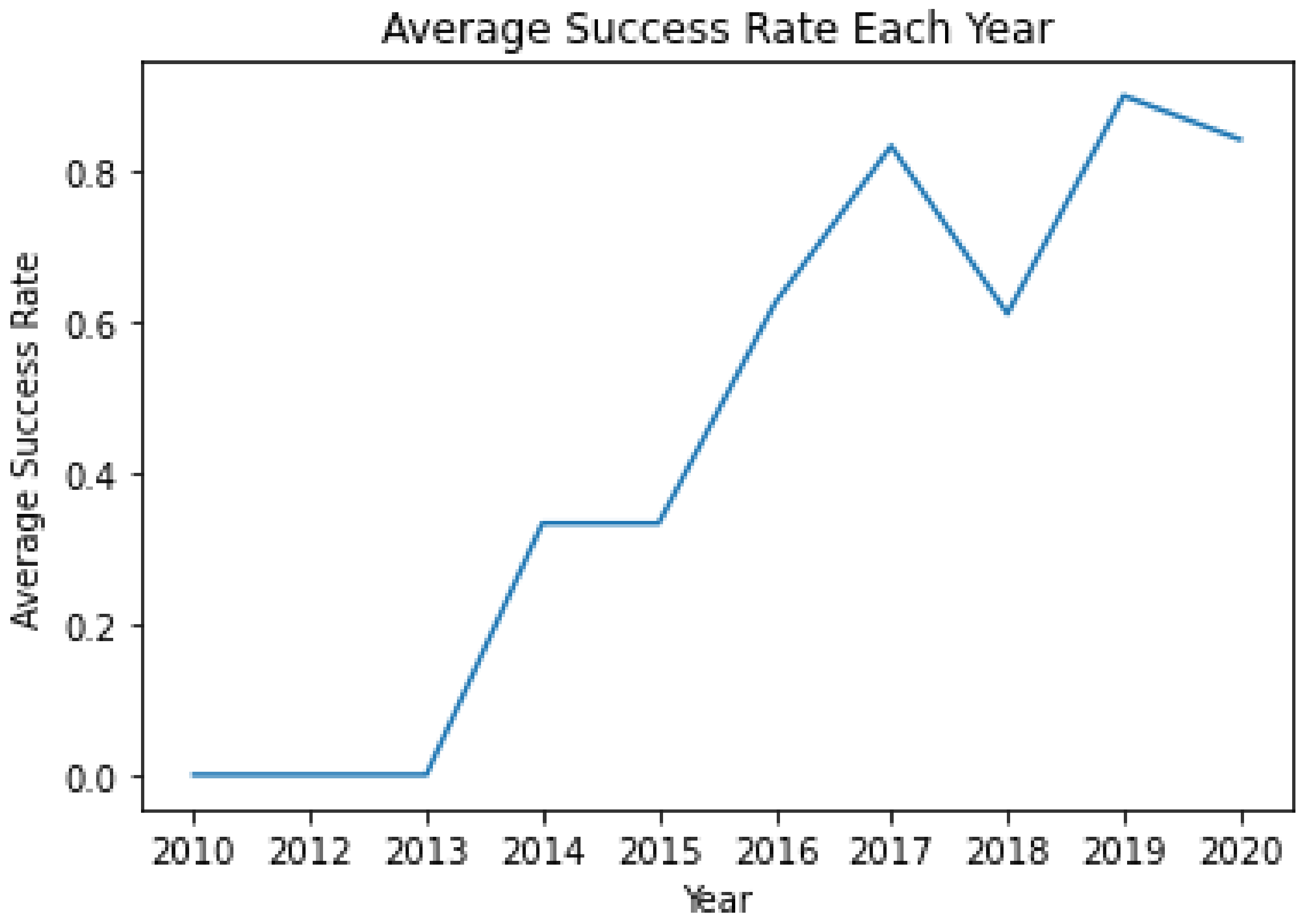
Payload vs. Orbit Type

In the Payload vs Orbit Type chart, we can see heavier payloads yield higher successful landing or positive landing rate 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.



Launch Success Yearly Trend

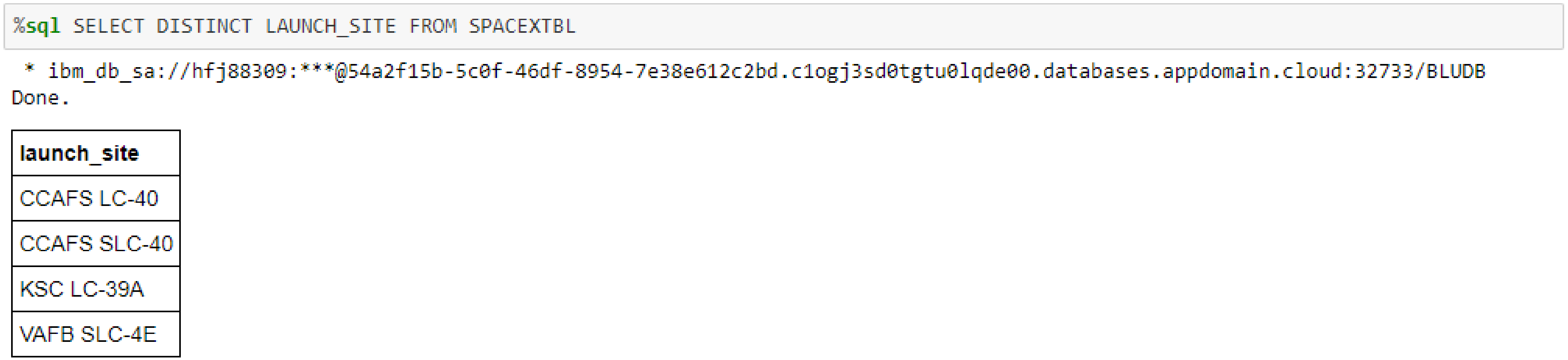
We can plot a line chart, Year vs Average Success Rate. We can observe the upward trend of the graph and the success rate since 2013 kept increasing till 2020



All Launch Site Names

Task 1 - Find the names of the unique launch sites.

- Using DISTINCT function, the result will have unique values of launch sites



```
%sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL
```

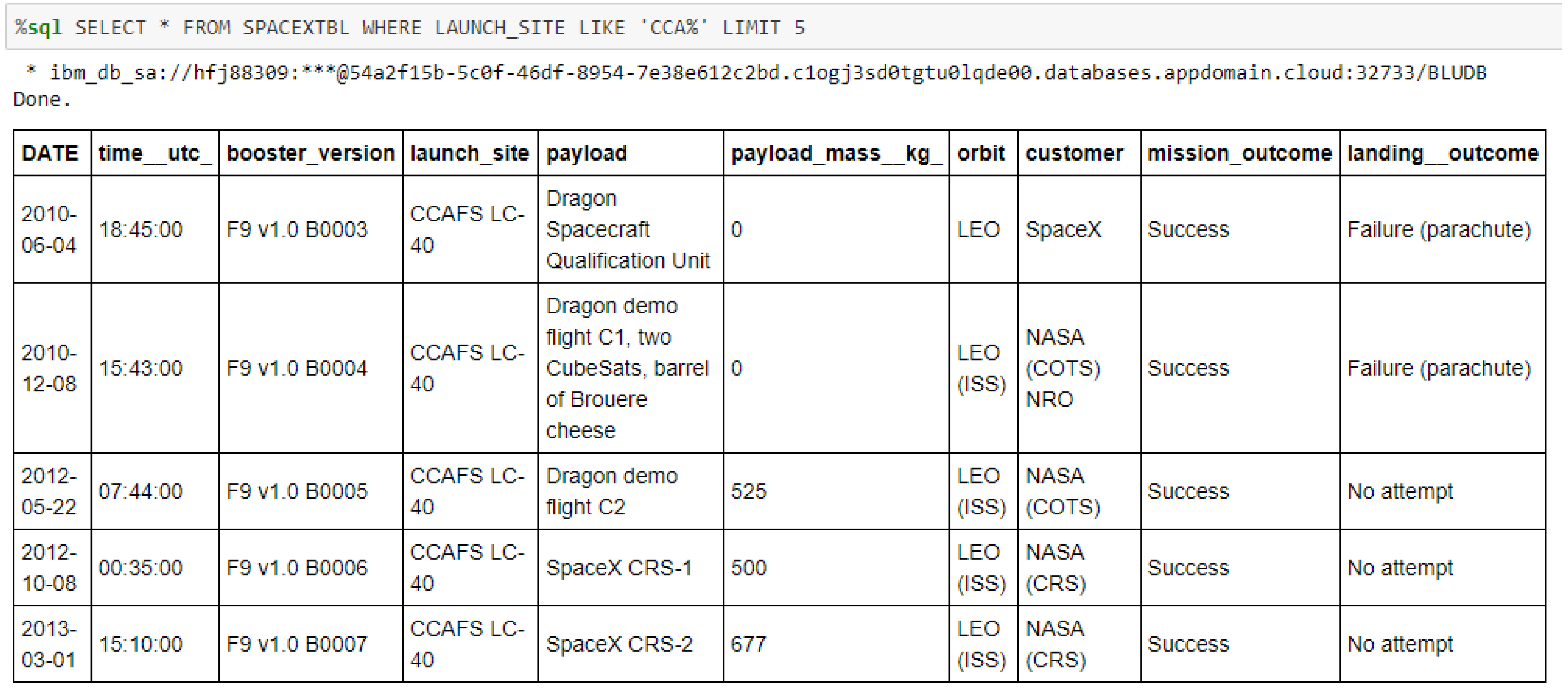
```
 * ibm_db_sa://hfj88309:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.
```

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

Launch Site Names Begin with 'CCA'

Task 2 - Find 5 records where launch sites begin with `CCA`

- LIKE operator determines whether a specific string matches a specific string pattern. ‘CCA%’ means the string pattern starts with ‘CCA’
- LIMIT 5 shows top 5 results of the query



```
%sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5
```

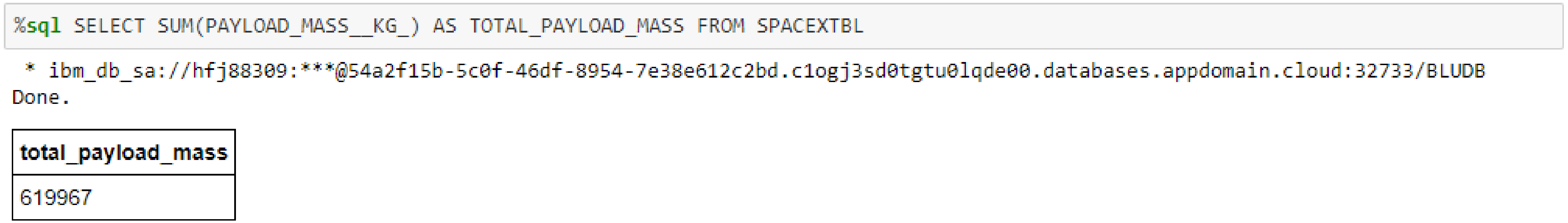
```
 * ibm_db_sa://hfj88309:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.
```

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	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	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

Task 3 - Calculate the total payload carried by boosters from NASA

- SUM function adds up the values in the ‘PAYLOAD_MASS_KG_’ column



```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS TOTAL_PAYLOAD_MASS FROM SPACEXTBL
```

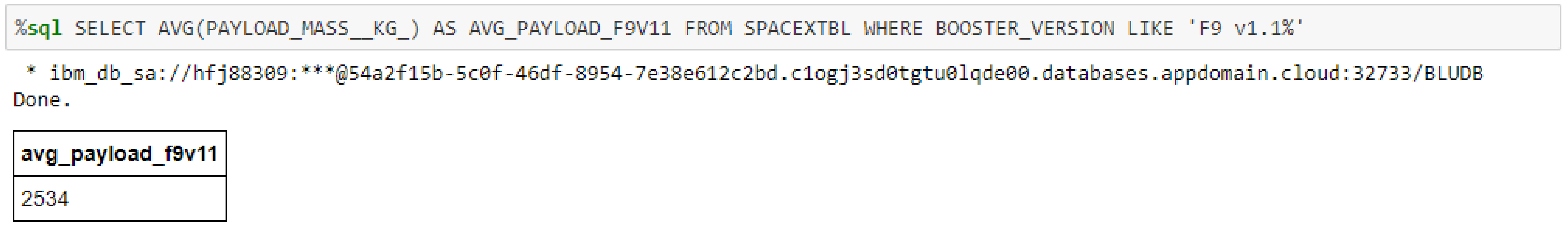
```
 * ibm_db_sa://hfj88309:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.
```

total_payload_mass
619967

Average Payload Mass by F9 v1.1

Task 4 - Calculate the average payload mass carried by booster version F9 v1.1

- AVG function calculates the average value of the column. LIKE operator selects entries with string pattern starts with ‘F9 v1.1’ in the ‘BOOSTER_VERSION’ column



```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS AVG_PAYLOAD_F9V11 FROM SPACEXTBL WHERE BOOSTER_VERSION LIKE 'F9 v1.1%'
```

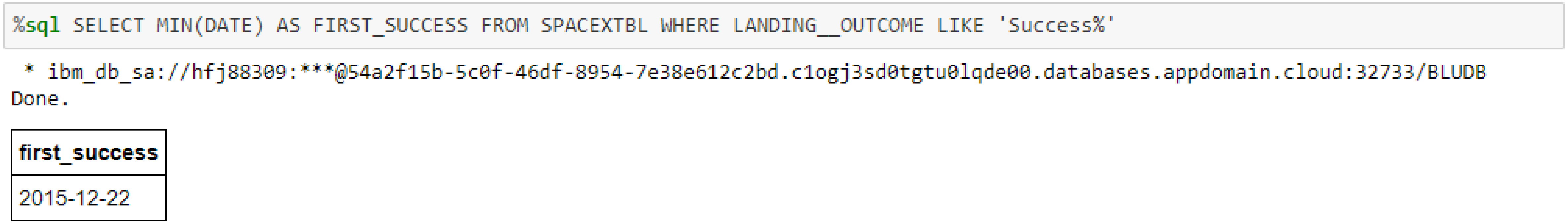
```
 * ibm_db_sa://hfj88309:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.
```

avg_payload_f9v11
2534

First Successful Ground Landing Date

Task 5 - Find the dates of the first successful landing outcome on ground pad

- MIN function on ‘DATE’ column shows the earliest entry
- LIKE operator selects only the landing outcome which has ‘Success’ in its value



```
%sql SELECT MIN(DATE) AS FIRST_SUCCESS FROM SPACEXTBL WHERE LANDING__OUTCOME LIKE 'Success%'
```

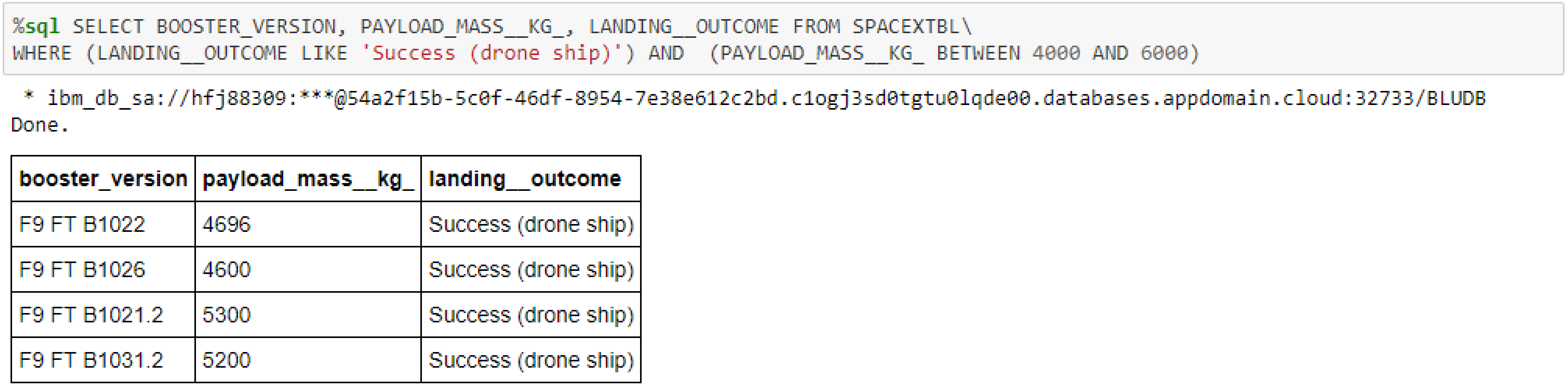
```
 * ibm_db_sa://hfj88309:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.
```

first_success
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

Task 6 - List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

- Two conditions are joined with AND operator after WHERE clause
- BETWEEN 4000 AND 6000 shows 4000kg < payload < 6000kg



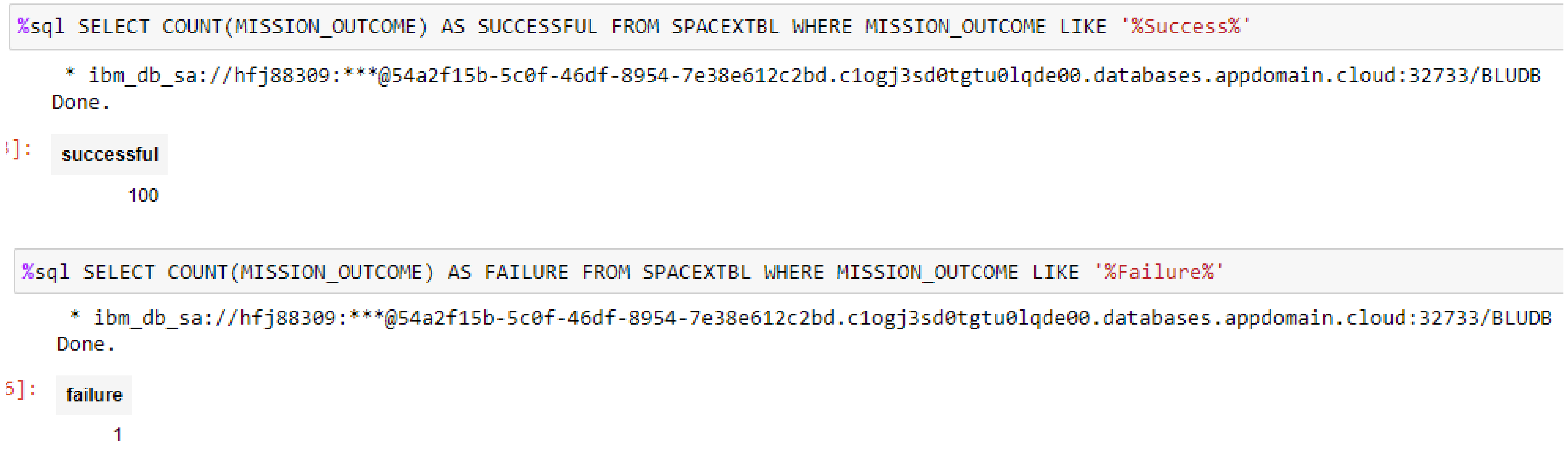
```
%sql SELECT BOOSTER_VERSION, PAYLOAD_MASS__KG_, LANDING__OUTCOME FROM SPACEXTBL\
WHERE (LANDING__OUTCOME LIKE 'Success (drone ship)') AND  (PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000)
```

```
 * ibm_db_sa://hfj88309:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.
```

booster_version	payload_mass__kg_	landing__outcome
F9 FT B1022	4696	Success (drone ship)
F9 FT B1026	4600	Success (drone ship)
F9 FT B1021.2	5300	Success (drone ship)
F9 FT B1031.2	5200	Success (drone ship)

Total Number of Successful and Failure Mission Outcomes

- Task 7 - Calculate the total number of successful and failure mission outcome
  - Using LIKE operator, total number of mission outcomes are displayed



```
%sql SELECT COUNT(MISSION_OUTCOME) AS SUCCESSFUL FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE '%Success%'
```

```
 * ibm_db_sa://hfj88309:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.
```

]:

successful
100

```
%sql SELECT COUNT(MISSION_OUTCOME) AS FAILURE FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE '%Failure%'
```

```
 * ibm_db_sa://hfj88309:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.
```

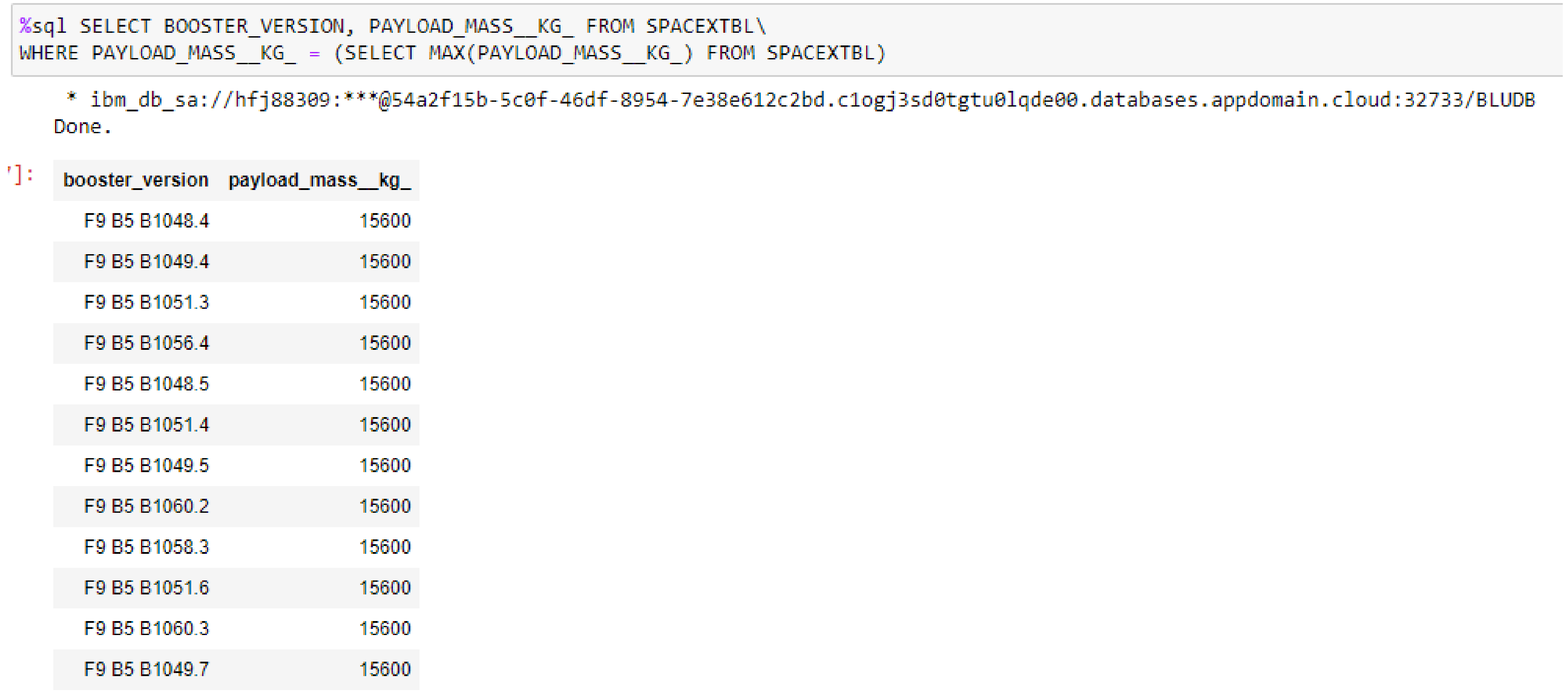
5]:

failure
1

Boosters Carried Maximum Payload

Task 8 - List the names of the booster which have carried the maximum payload mass

- Nested SELECT statement is used to perform this query



```
%sql SELECT BOOSTER_VERSION, PAYLOAD_MASS__KG_ FROM SPACEXTBL\
WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
```

```
 * ibm_db_sa://hfj88309:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.
```

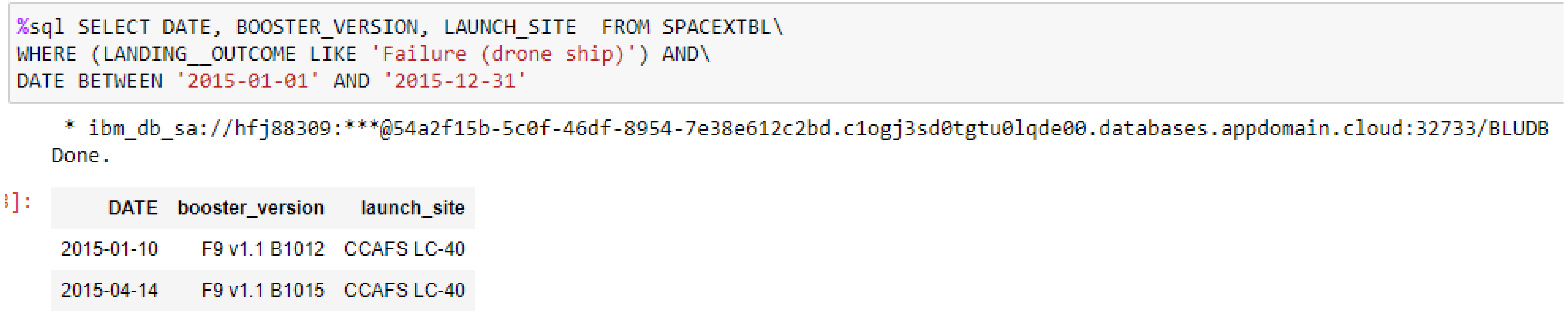
']:

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

2015 Launch Records

Task 9 - List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

- There are two conditions joined with AND in this case. First condition is a LIKE operator. Second is a date range selected using BETWEEN AND



```
%sql SELECT DATE, BOOSTER_VERSION, LAUNCH_SITE  FROM SPACEXTBL\
WHERE (LANDING__OUTCOME LIKE 'Failure (drone ship)') AND\
DATE BETWEEN '2015-01-01' AND '2015-12-31'
```

```
 * ibm_db_sa://hfj88309:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.
```

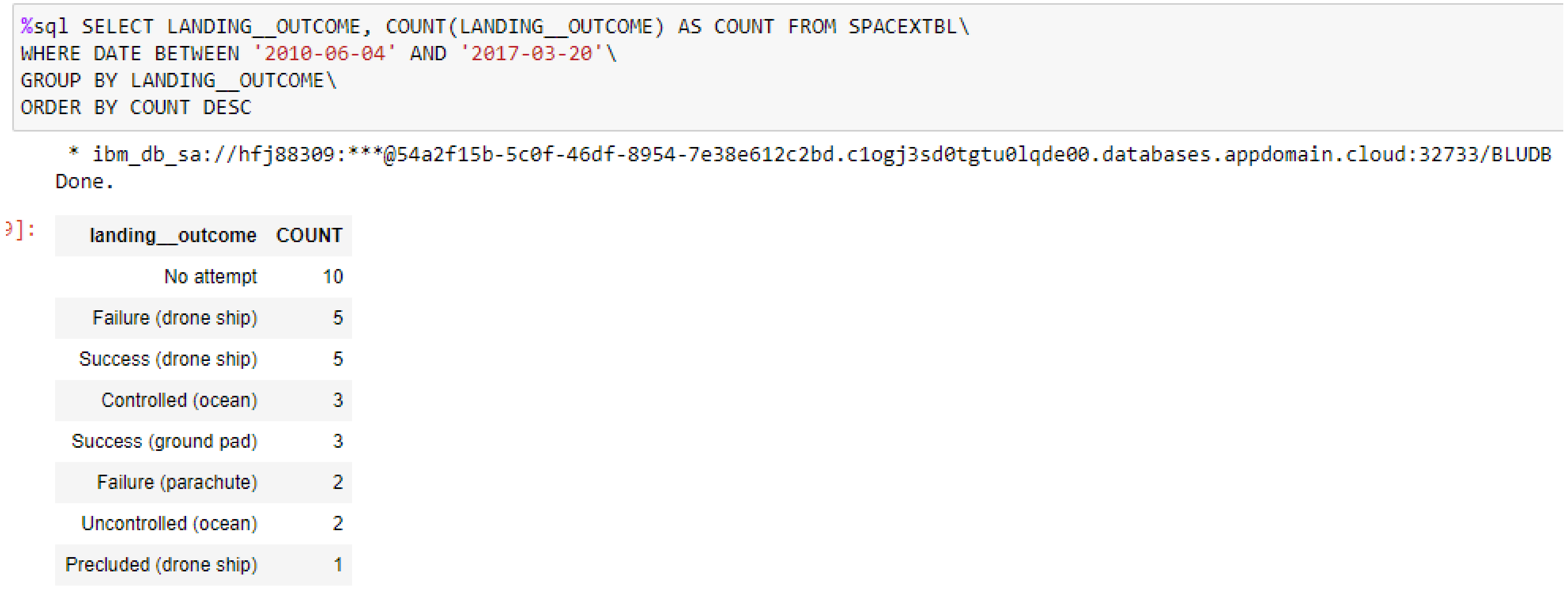
]:

DATE	booster_version	launch_site
2015-01-10	F9 v1.1 B1012	CCAFS LC-40
2015-04-14	F9 v1.1 B1015	CCAFS LC-40

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

Task 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

- Date range is set using BETWEEN AND. The GROUP BY statement groups rows that have the same values into summary rows and the count of landing outcome is shown in descending order using ORDER BY … DESC.



```
%sql SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) AS COUNT FROM SPACEXTBL\
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'\
GROUP BY LANDING__OUTCOME\
ORDER BY COUNT DESC
```

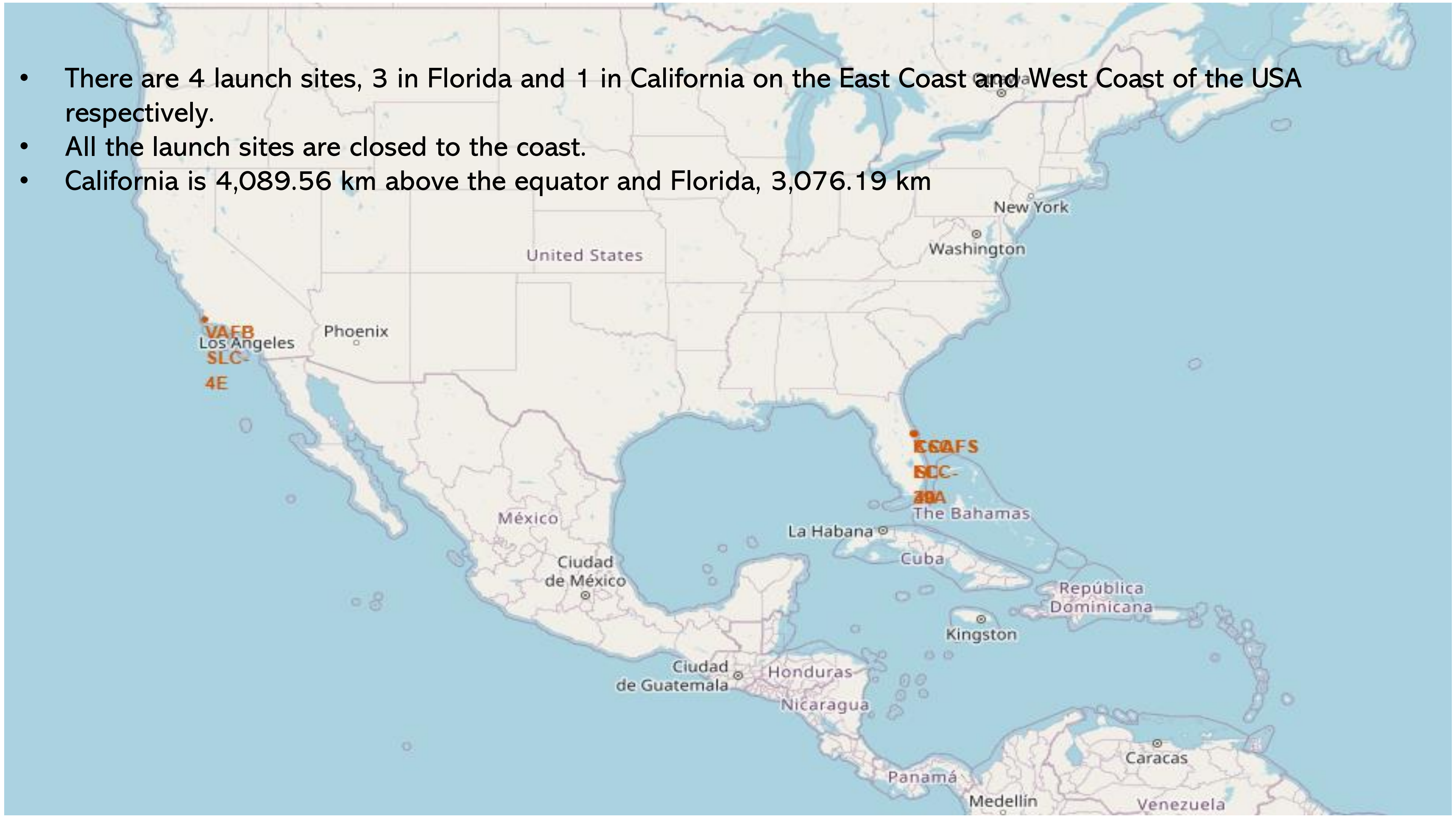
```
 * ibm_db_sa://hfj88309:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.
```

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

Section 4

Launch Sites Proximities Analysis

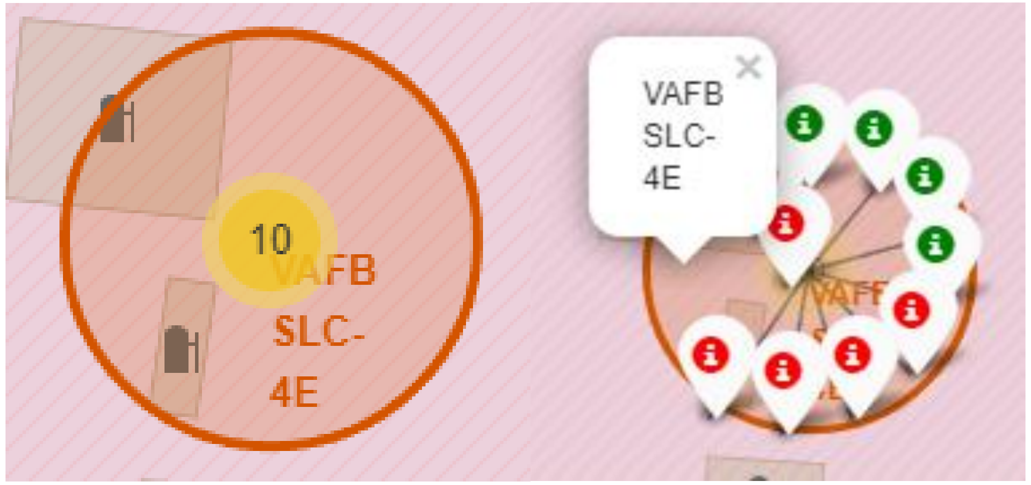
Circle Markers Of All Launch Sites



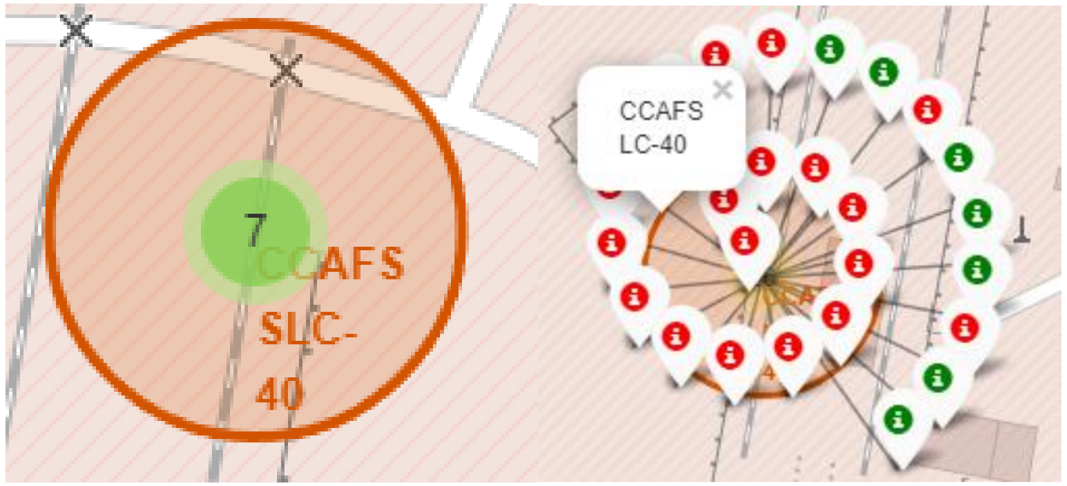
- There are 4 launch sites, 3 in Florida and 1 in California on the East Coast and West Coast of the USA respectively.
- All the launch sites are closed to the coast.
- California is 4,089.56 km above the equator and Florida, 3,076.19 km

Color-labelled Launch Records For Each Site

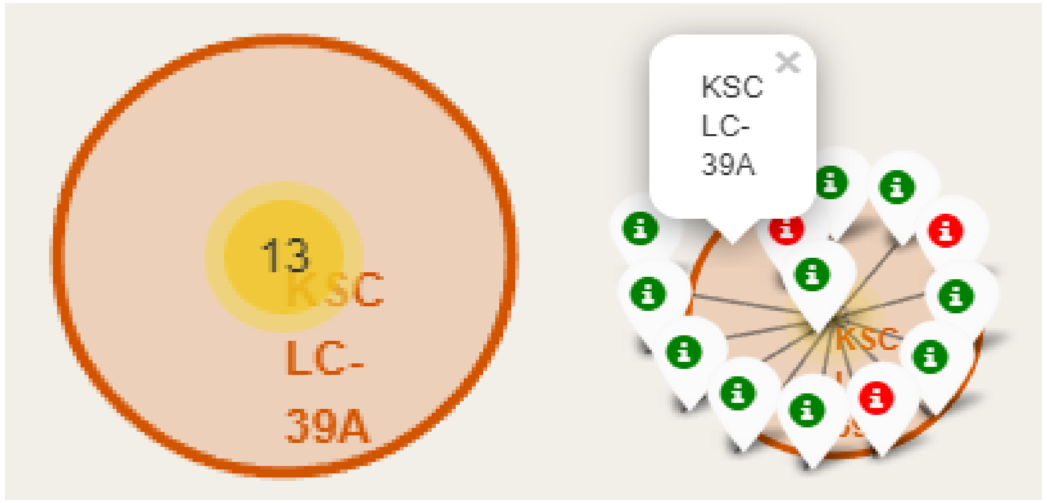
VAFB SLC-4E (California)



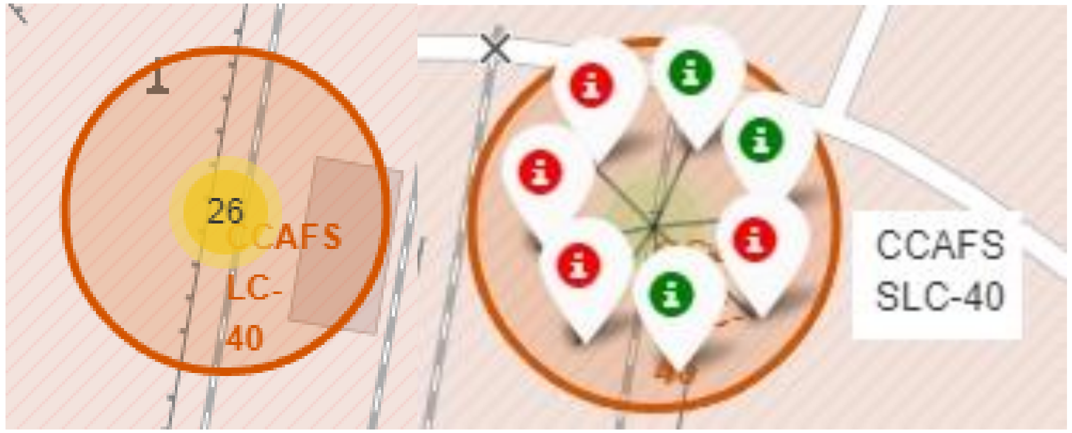
CCAF SLC-40 (Florida)



KSC LC-39A (Florida)



CCAFS SLC-40 (Florida)

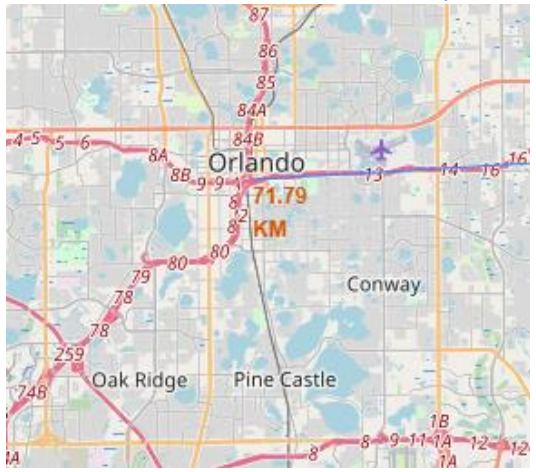


Green Marker – Successful
Red Marker – Failed

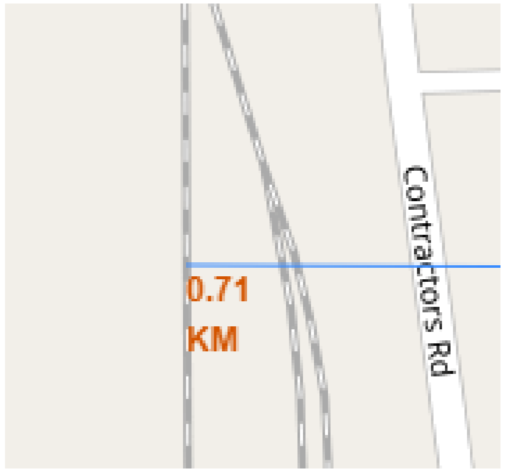
Distances Of Landmarks From KSC LC-39A

Distance to coast

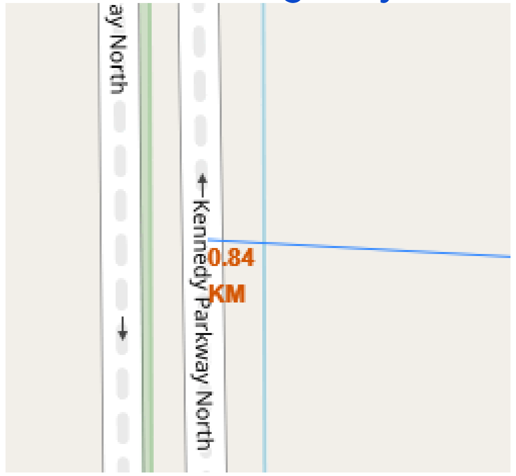
Distance to Orlando City



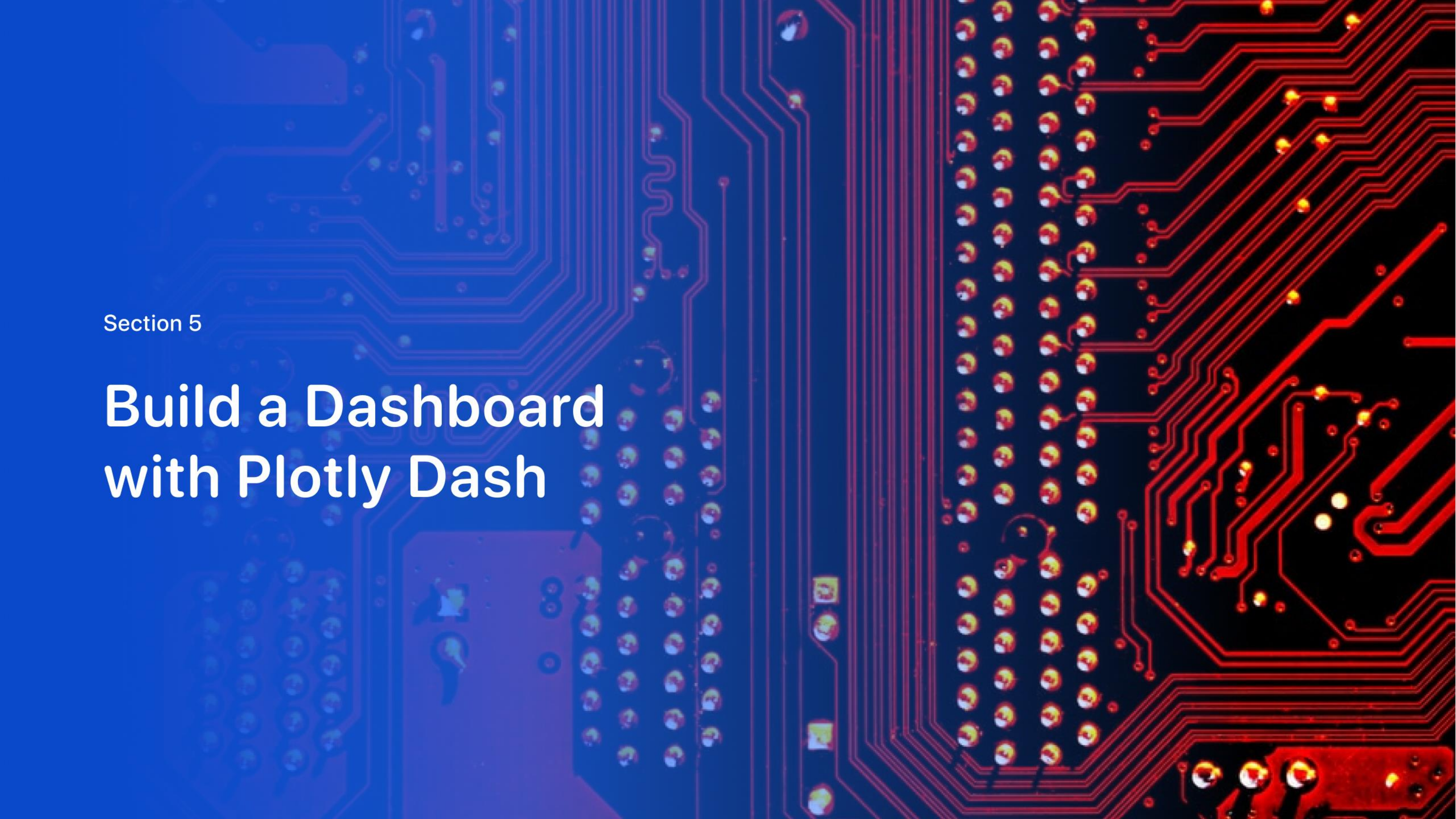
Distance to Railroad



Distance to Highway



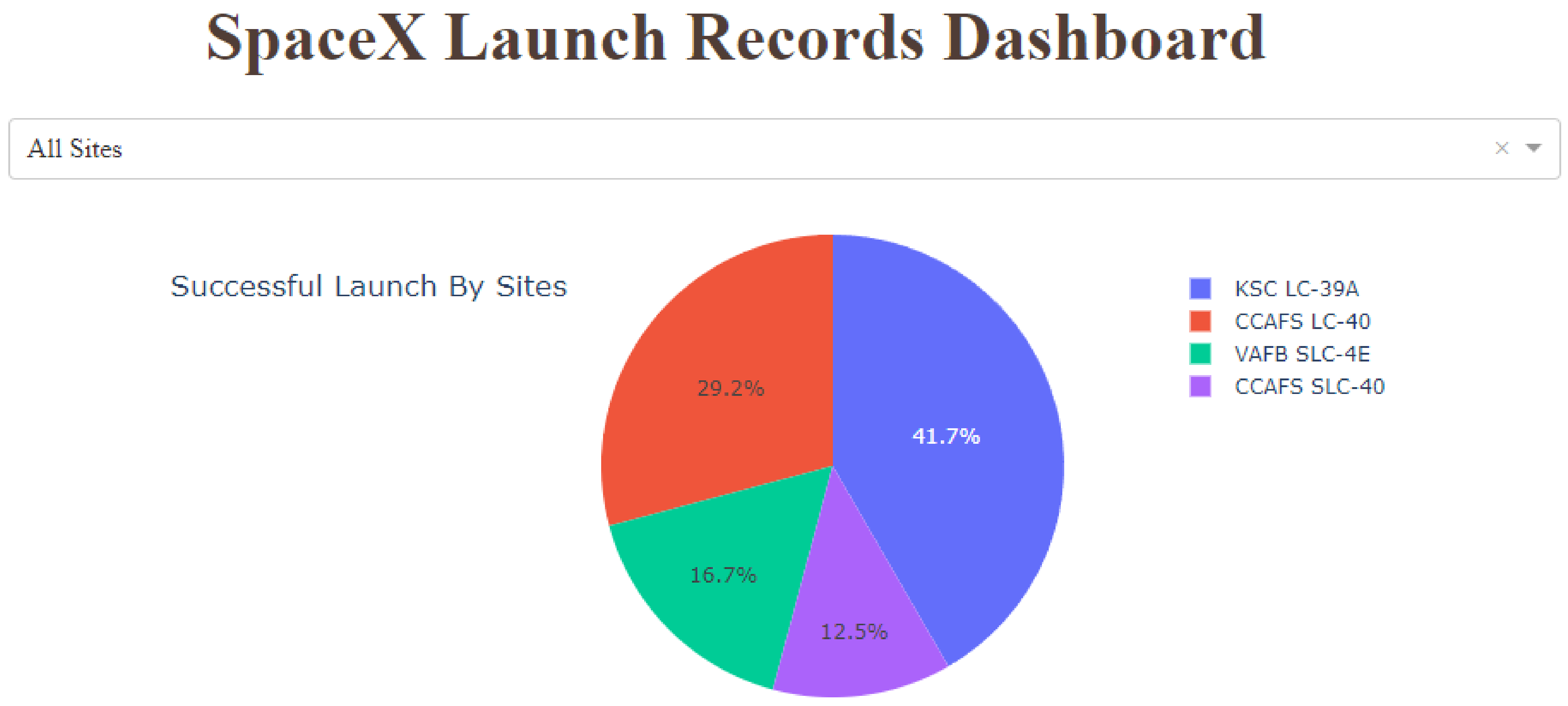
- Selected launch site, KSC LC-39A, is 7.44km from the coast.
- Orlando City is located 71.79km for the launch site.
- Nearest railroad is 0.71km.
- Nearest highway, Kennedy Parkway North, is 0.84km away.



Section 5

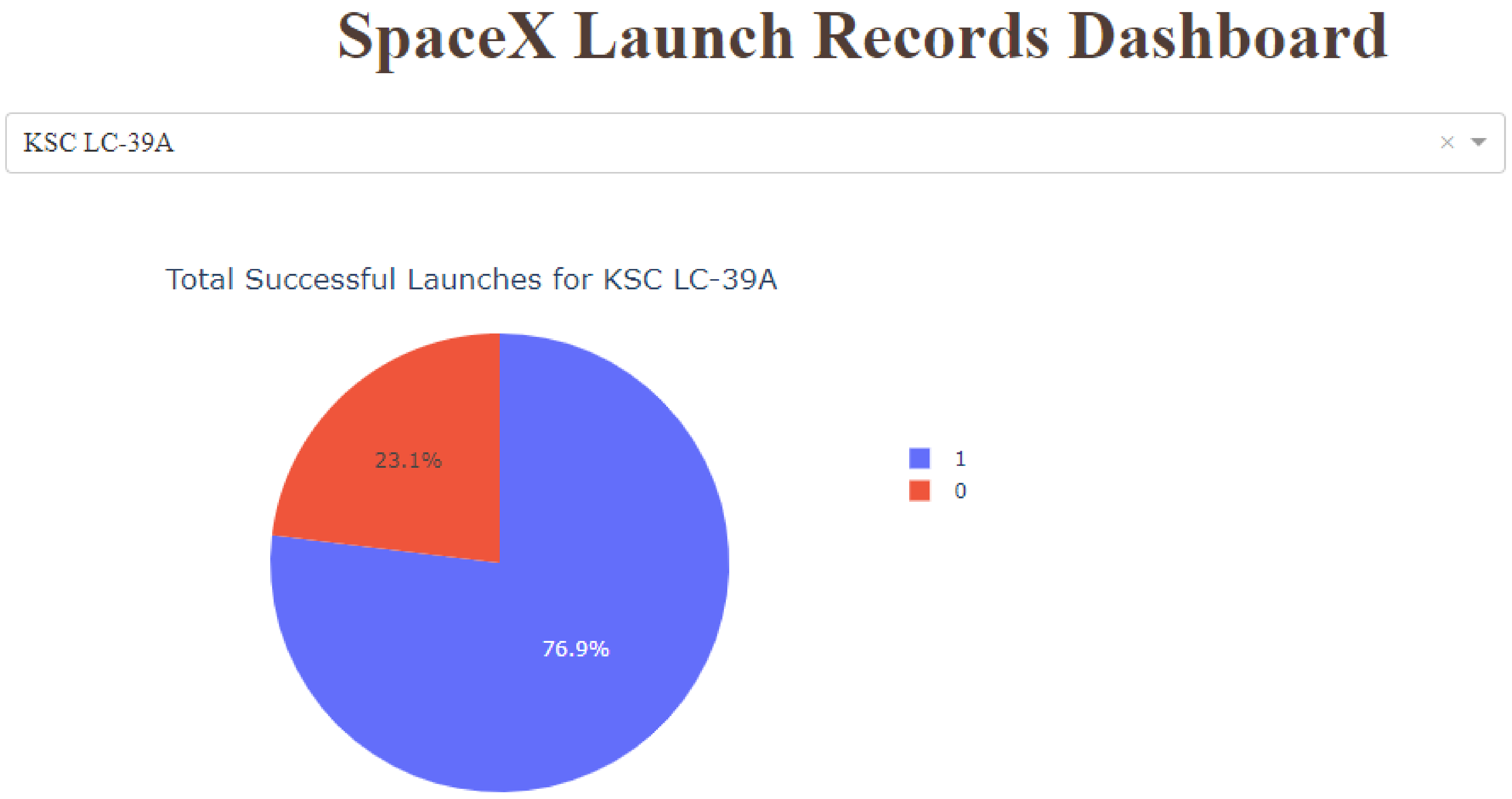
Build a Dashboard with Plotly Dash

Successful Launch By sites



This Pie Chart shows that launch site KSC LC-39A has the highest success ratio with 41.7% of all launches.

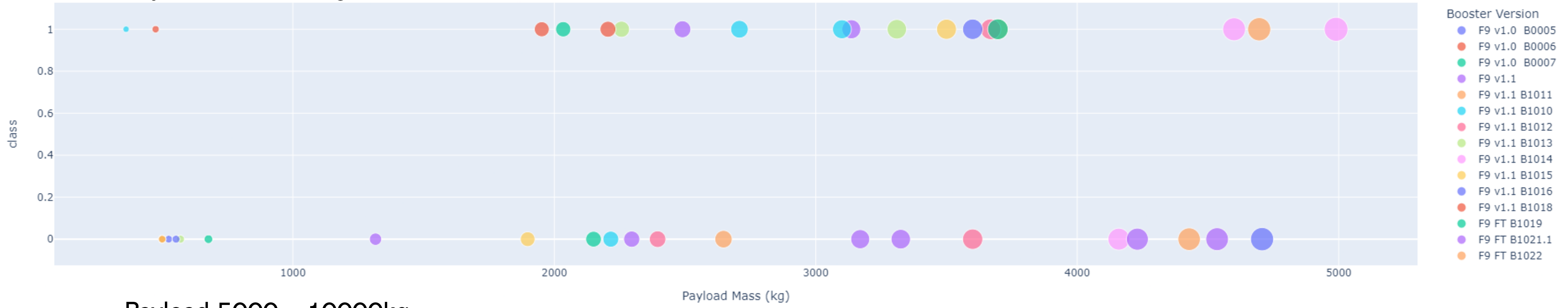
Launch Site With Highest Success Ration


SpaceX Launch Records Dashboard
KSC LC-39A
Total Successful Launches for KSC LC-39A
23.1%
76.9%
1
0

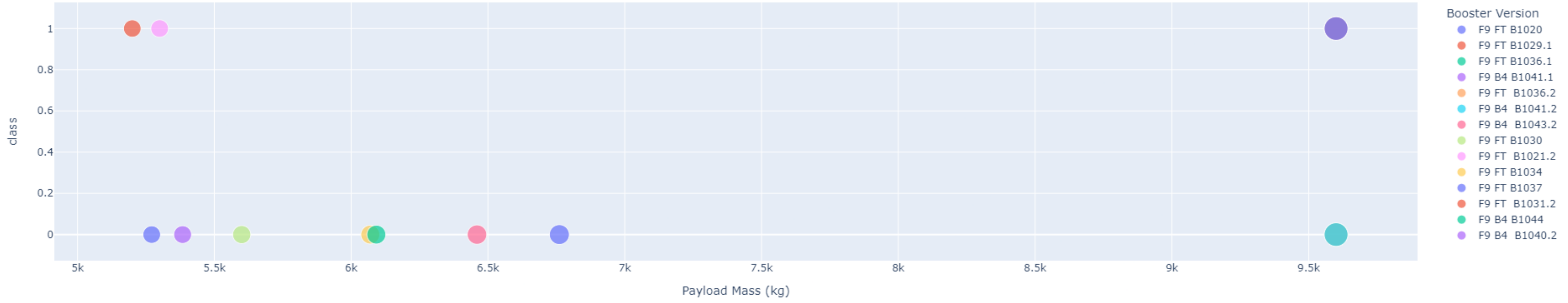
KSC LC-39 A has success rate of 76.9% and failure rate of 23.1%

Payload vs. Launch Outcome Scatter Plot For All Sites, With Different Payload Selected In The Range Slider

Payload 0 – 5000kg



Payload 5000 – 10000kg

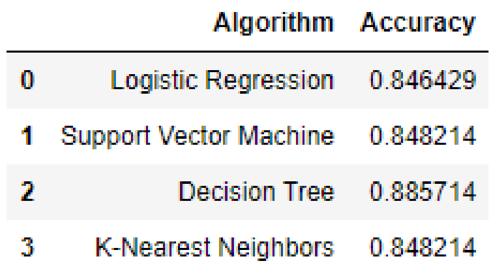


From the scatter plots, we can see that boosters of payload below 5000kg has higher success rate than boosters of payload between 5000kg and 10000kg.

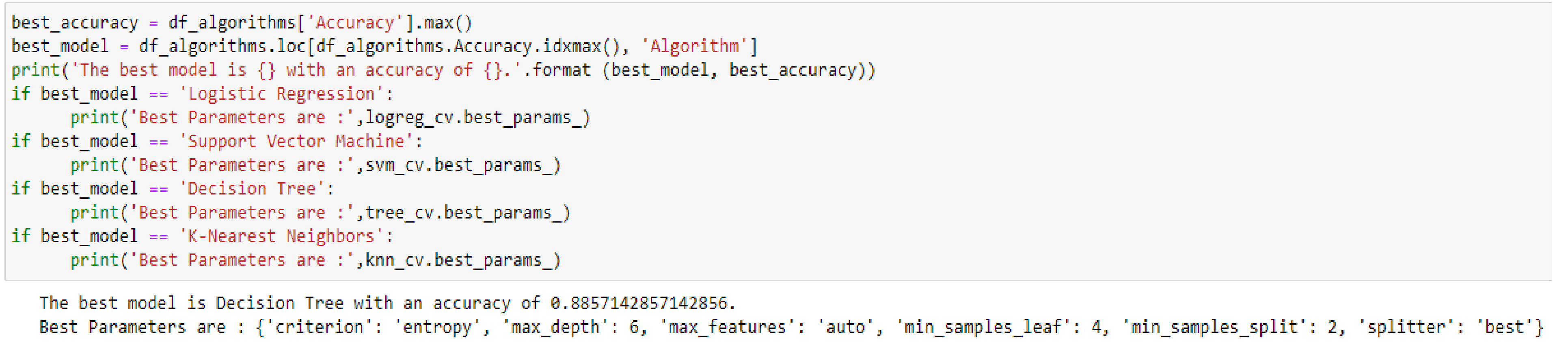
Section 6

Predictive Analysis (Classification)

Classification Accuracy

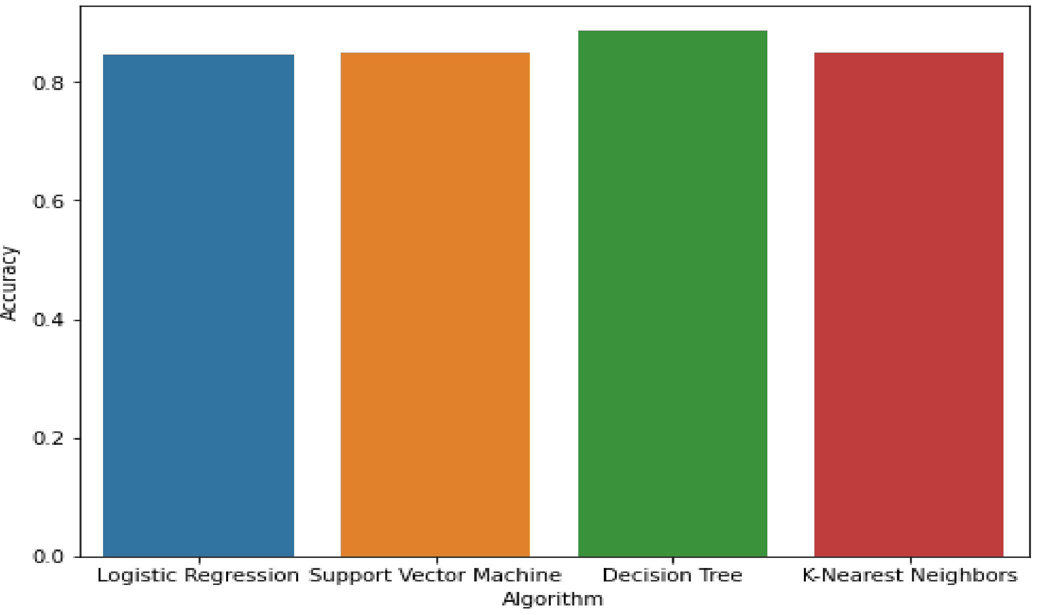


	Algorithm	Accuracy
0	Logistic Regression	0.846429
1	Support Vector Machine	0.848214
2	Decision Tree	0.885714
3	K-Nearest Neighbors	0.848214



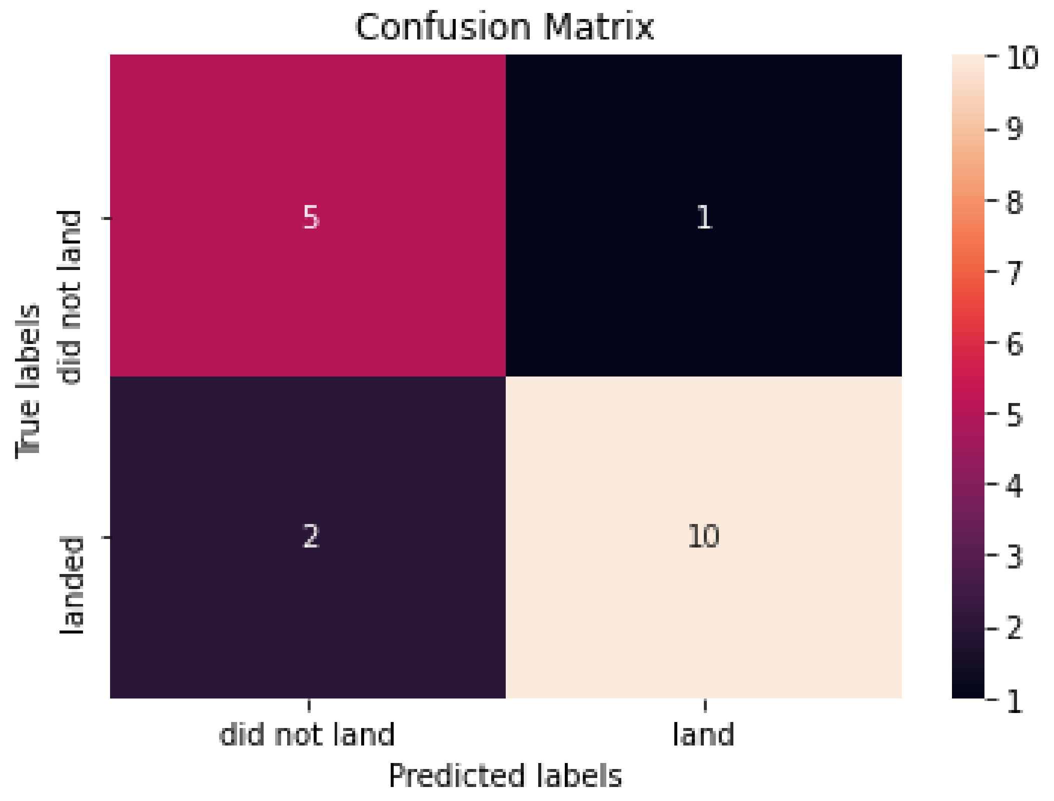
```
best_accuracy = df_algorithms['Accuracy'].max()
best_model = df_algorithms.loc[df_algorithms.Accuracy.idxmax(), 'Algorithm']
print('The best model is {} with an accuracy of {}.'.format (best_model, best_accuracy))
if best_model == 'Logistic Regression':
    print('Best Parameters are :',logreg_cv.best_params_)
if best_model == 'Support Vector Machine':
    print('Best Parameters are :',svm_cv.best_params_)
if best_model == 'Decision Tree':
    print('Best Parameters are :',tree_cv.best_params_)
if best_model == 'K-Nearest Neighbors':
    print('Best Parameters are :',knn_cv.best_params_)
```

```
The best model is Decision Tree with an accuracy of 0.8857142857142856.
Best Parameters are : {'criterion': 'entropy', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 2, 'splitter': 'best'}
```



As we can see from the bar chart and calculations, the accuracy of each model is not very far off from one another. Out of all the models, Decision Tree model has the highest accuracy.

Confusion Matrix of Decision Tree Model



The Tree model made 15 correct predictions out of 18. There are 3 incorrect predictions, 1 of them is False Positive (model predicted landed but the first stage did not land) and 2 of them is False Negative (model predicted did not land but the first stage did land).

Conclusions

- Success rate increases with the increased number of flights
- We can observe launches to orbits such as ES-L1, GEO, HEO and SSO have the highest success rates.
- Success rate since 2013 kept increasing till 2020
- KSC LC-39A has the highest success ratio of all launch sites and success rate of 76.9%
- Boosters of payload below 5000kg has higher success rate than boosters of payload between 5000kg and 10000kg.
- Decision Tree model is the best model to perform Predictive Analysis (Classification) with an accuracy of 88.57%

Appendix

Datasets used in Python Jupyter Notebooks

- *dataset_part_1.csv*
- *dataset_part_2.csv*
- *dataset_part_3.csv*
- *spacex_launch_dash.csv*
- *spacex_web_scraped.csv*

Thank you!