IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Sheldon Munns
11 Sept 2022

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix

Executive Summary

- Summary of methodologies
  - Data Collection with API
  - Data Collection with Webscraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with data visualization
  - Visual Analytics with Folium
  - Dashboard with Plotly Dash
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis
  - Analysis with screenshots
  - Predictive Analysis using Machine Learning

Introduction

- The commercial space age is here, companies are making space travel affordable for everyone. Virgin Galactic is providing suborbital spaceflights. Rocket Lab is a small satellite provider. Blue Origin manufactures sub-orbital and orbital reusable rockets. Perhaps the most successful is SpaceX. 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 to find answers
  - Obtain SpaceX data concerning launches
  - Manipulate and visualize the data to make meaningful insights
  - Predictions using machine learning for decision making

Section 1

Methodology

Methodology

Executive Summary

Data collection methodology:

- Data was collected using SpaceX API and web scraping from Wikipedia

Perform data wrangling

- Data wrangling uses Exploratory Data Analysis to find patterns and outcomes and one hot encoding to turn the outcomes into training labels

Perform exploratory data analysis (EDA) using visualization and SQL

Perform interactive visual analytics using Folium and Plotly Dash

Perform predictive analysis using classification models

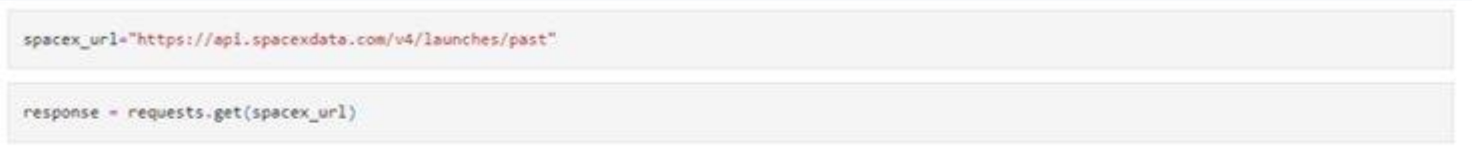
- To build models, tune the parameters, and then evaluate the various classification models for the best outcome

Data Collection

- The SpaceX launch data needed is gathered from an API, specifically the SpaceX REST API. This API provides data about launches, including information about the rocket used, payload delivered, launch specifications, landing specifications, and finally landing outcomes. Next step is to perform a get request to obtain the launch data, viewing the response it is in the form of a JSON. To convert this JSON to a dataframe, we can use the json_normalize function. This function will allow us to "normalize" the structured json data into a flat table.
- Data is also is also obtained using the Python BeautifulSoup package to web scrape some HTML tables that contain valuable Falcon 9 launch records. We then parse the data from those tables and convert them into a Pandas data frame for further visualization and analysis. We transform this raw data into a clean dataset which provides meaningful data.

Data Collection – SpaceX API

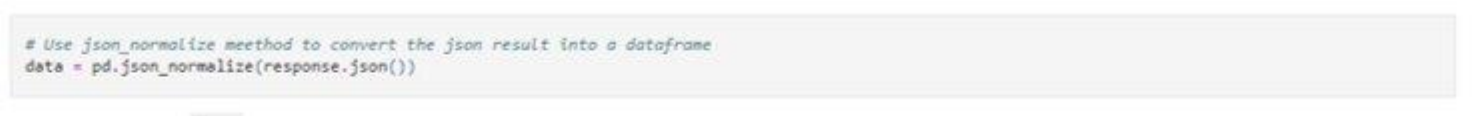
- Use API and get request to obtain the data from SpaceX



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

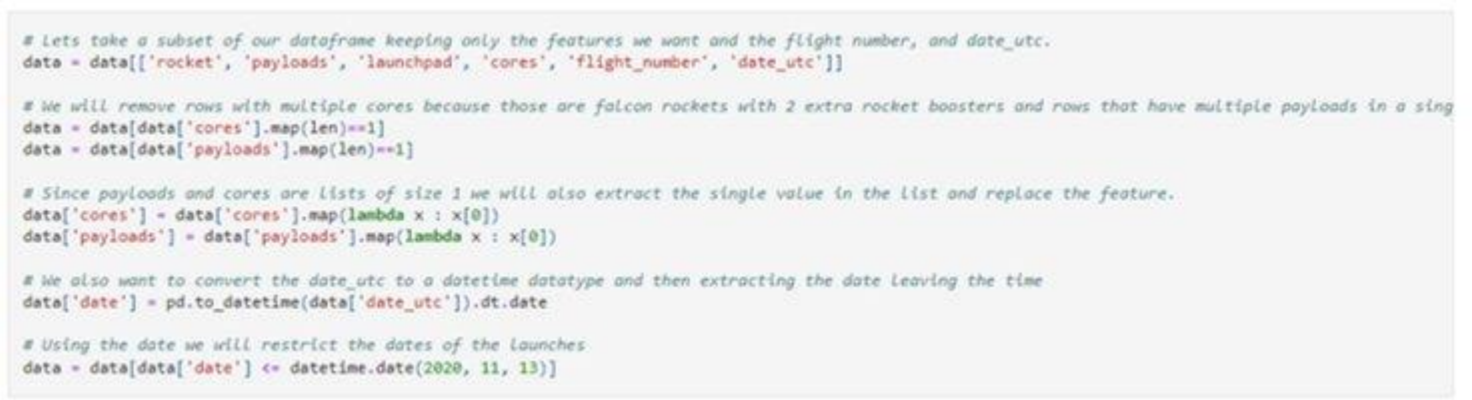
```
response = requests.get(spacex_url)
```

- Use json_normalize to convert the json results into a dataframe



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

- Data Cleaning and filling the missing values



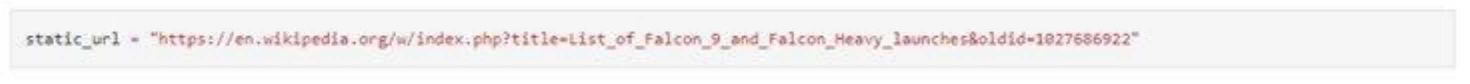
```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that have multiple payloads in a sing
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)]
```

- GitHub URL:

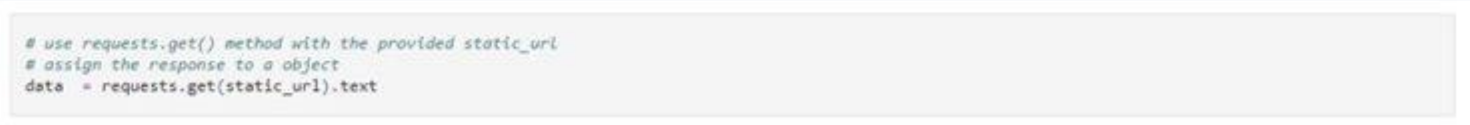
  https://github.com/samunns/datasciencecap/blob/main/Lab1_Spacex_Data_Collection-api.ipynb

Data Collection - Scraping

- HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response

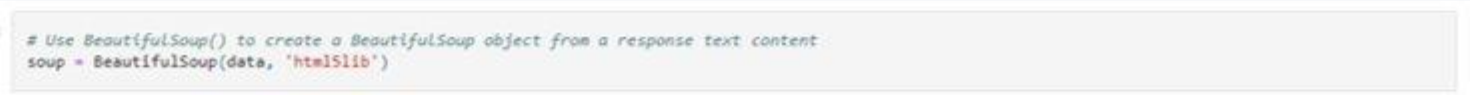


```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```



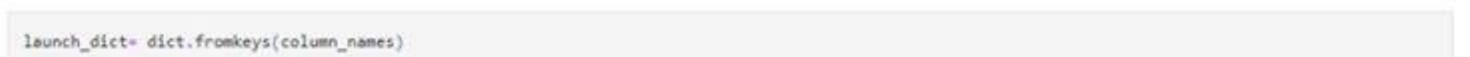
```
# use requests.get() method with the provided static_url
# assign the response to a object
data  = requests.get(static_url).text
```

- Create a BeautifulSoup object from the HTML response

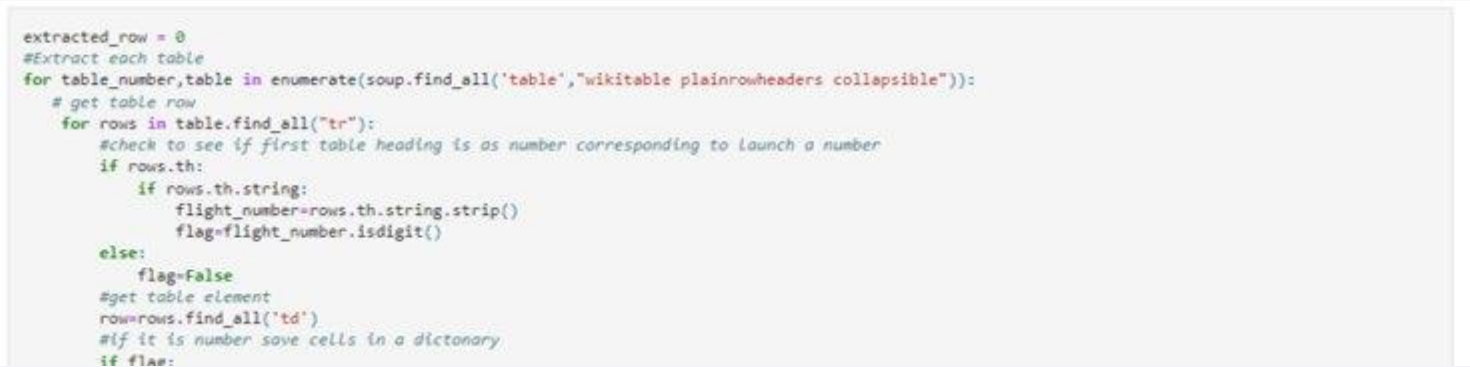


```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(data, 'html5lib')
```

- Create a data frame by parsing the launch HTML tables



```
launch_dict= dict.fromkeys(column_names)
```



```
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:
```

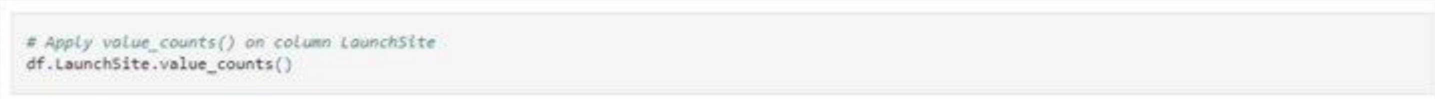
- GitHub URL:

  https://github.com/samunns/datasciencecap/blob/main/Webscraping.ipynb

Data Wrangling

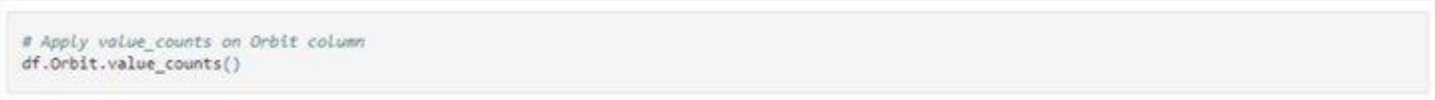
Perform Exploratory Data Analysis (EDA) to find patterns in the data and determine the label for training supervised models

- Calculate the number of launches on each site



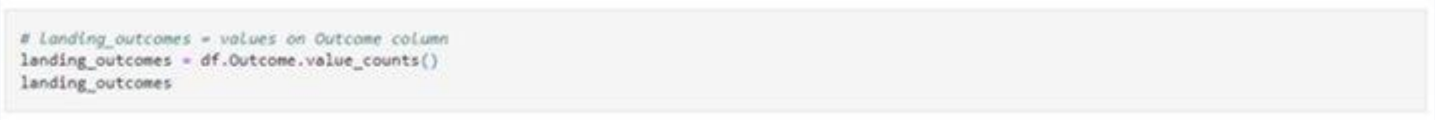
```
# Apply value_counts() on column LaunchSite
df.LaunchSite.value_counts()
```

- Calculate the number and occurrences of each orbit



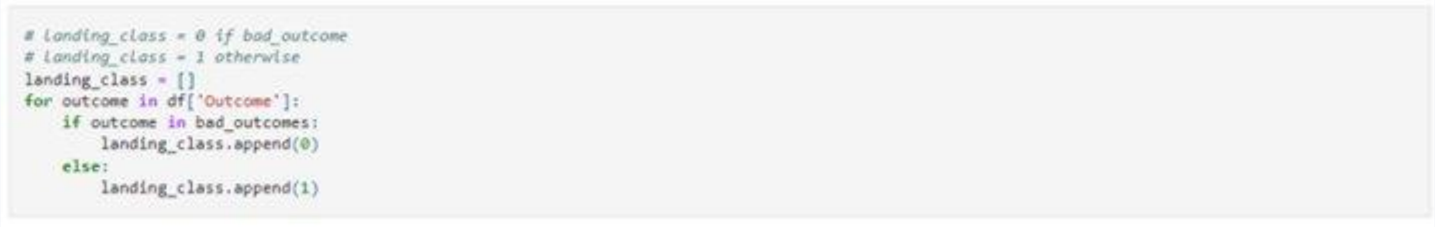
```
# Apply value_counts on Orbit column
df.Orbit.value_counts()
```

- Calculate the number and occurrence of mission outcome per orbit type



```
# landing_outcomes = values on Outcome column
landing_outcomes = df.Outcome.value_counts()
landing_outcomes
```

- Create a landing outcome label



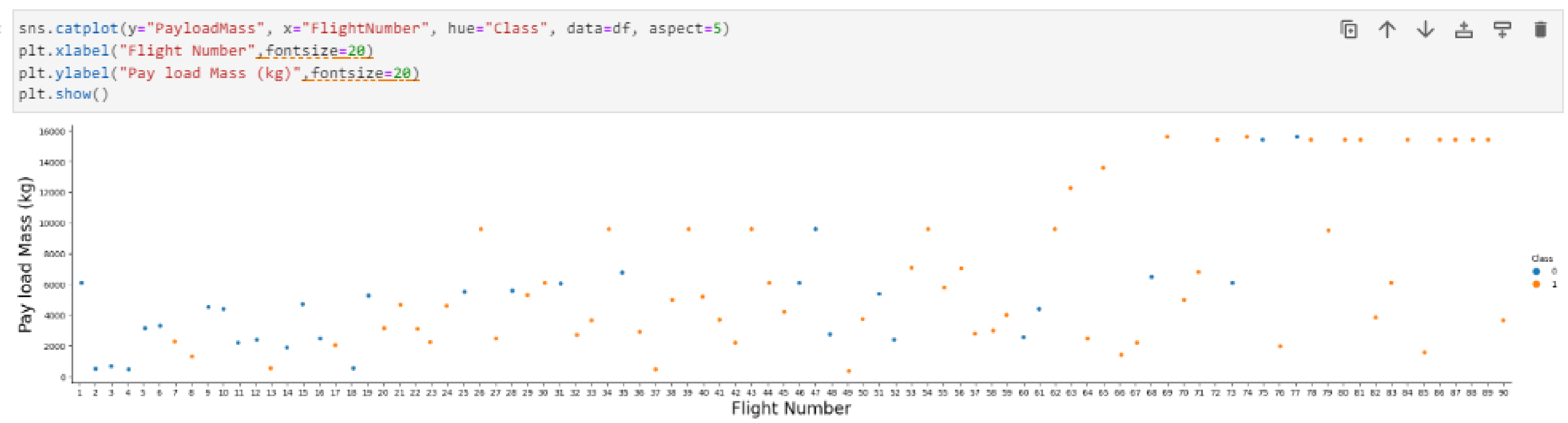
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for outcome in df['Outcome']:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

- GitHub URL:

  https://github.com/samunns/datasciencecap/blob/main/Lab2_Spacex_Data_Wrangling.ipynb

EDA with Data Visualization

- Use EDA and Feature engineering to predict landing success using scatter plots for visualization
- Relationships visualized: Flight Number and Launch Site, Payload and Launch Site, Flight Number and Orbit type, Payload and Orbit type



```
sns.catplot(y="PayloadMass", x="FlightNumber", hue="Class", data=df, aspect=5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```

- GitHub URL:
https://github.com/samunns/datasciencecap/blob/main/EDA-data%20visualization.ipynb

EDA with SQL

SQL queries performed to understand the SpaceX dataset:

- Names of the unique launch sites
- Launch sites beginning with the string “CCA”
- Total payload mass carried by boosters launched by NASA (CRS)
- Average payload mass carried by booster version F9 v1.1
- List dates when the first successful landing outcome in ground pad was achieved
- Names of boosters which have success in drone ship with payload mass greater than 4000 but less than 6000
- List total number of successful and failure mission outcomes
- List names of booster_versions carrying the maximum payload
- List failed landing_outcomes in drone ships, their booster versions, and launch site names for 2005
- Rank the count of landing outcomes between 2010 and 2017

- GitHub URL:

  https://github.com/samunns/datasciencecap/blob/main/EDA-SQL.ipynb

Build an Interactive Map with Folium

- After loading the appropriate libraries, we added each site's location on a map using site's latitude and longitude coordinates
- We used folium.circle to add a highlighted circle with a text label at each location.
- We then added launch outcomes at each site with 1’s and 0’s and added a green and red marker to mark success or failure
- We then analyzed the launch site proximities by calculating the distance and then using folium.marker to show the distance to landmarks

- GitHub URL:

  https://github.com/samunns/datasciencecap/blob/main/lab_jupyter_launch_site_location_Folium.ipynb

Build a Dashboard with Plotly Dash

- During the lab we built a dashboard with “Plotly dash” having a dropdown bar to display all launch sites or select an individual launch site
- Below the dropdown window a pie chart was built to visualize success rate for all launch sites; if an individual launch site was selected the pie chart would display total successful launches vs unsuccessful as a percentage
- Finally, to help identify the relationship of payload on success a scatter plot displayed the success count on payload mass for all or individual launch sites

- GitHub URL:
  - https://github.com/samunns/datasciencecap/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- Build the Models
  - Load the dataset into “numpy” and “pandas”
  - Transform the data
  - Split the data into training and test datasets
  - Choose the type of machine learning
  - Set the parameters and algorithms to “gridsearchcv” and fit it to the dataset

- Evaluate/Improve the Model
  - Calculate the accuracy of each model
  - Tuned hyperparameters for each model
  - Plot the confusion matrix

- Evaluate and Select the Best Model
  - Select the model with the highest accuracy

- GitHub URL:
  - https://github.com/samunns/datasciencecap/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

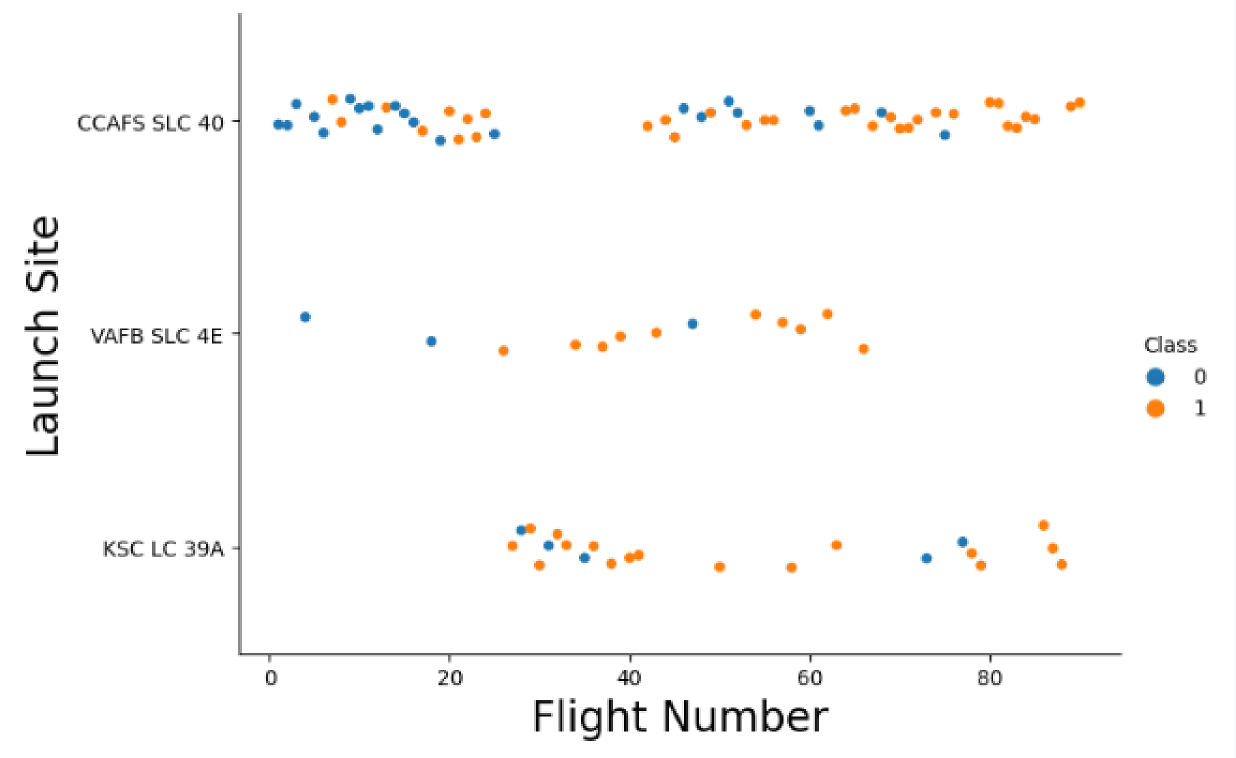
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

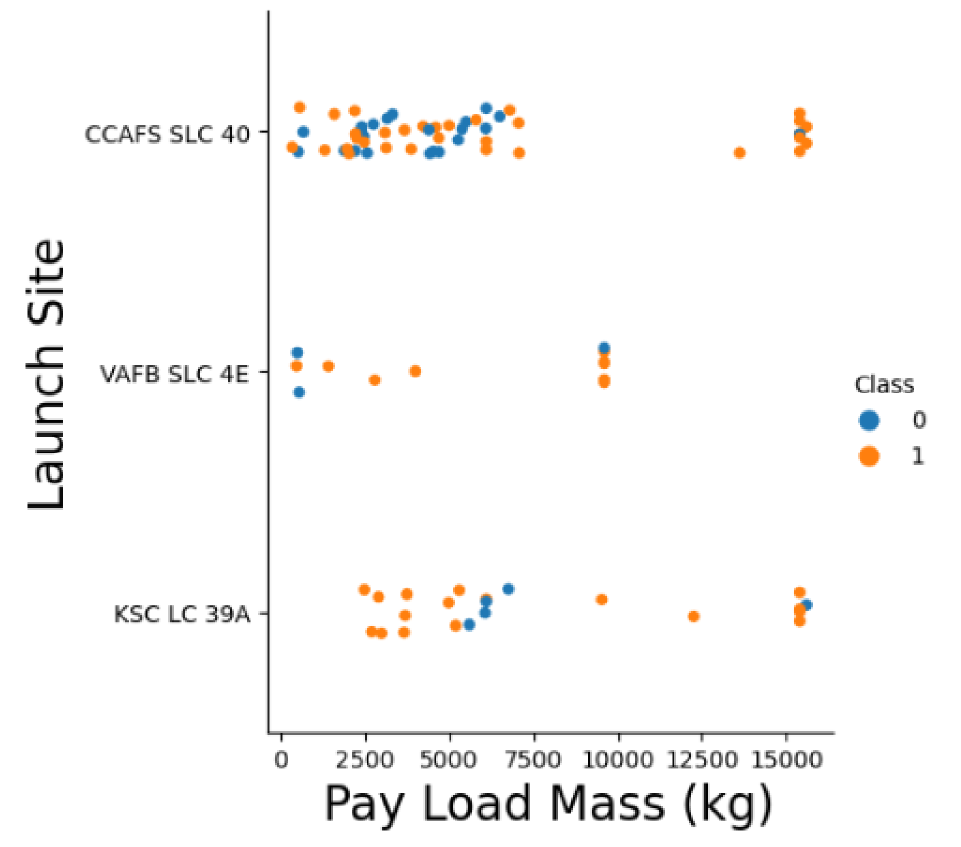
Flight Number vs. Launch Site

- Scatter plot shows that the larger the flight the higher the success rate
- However, CCAFS SLC 40 shows the least pattern of success rate from flight number



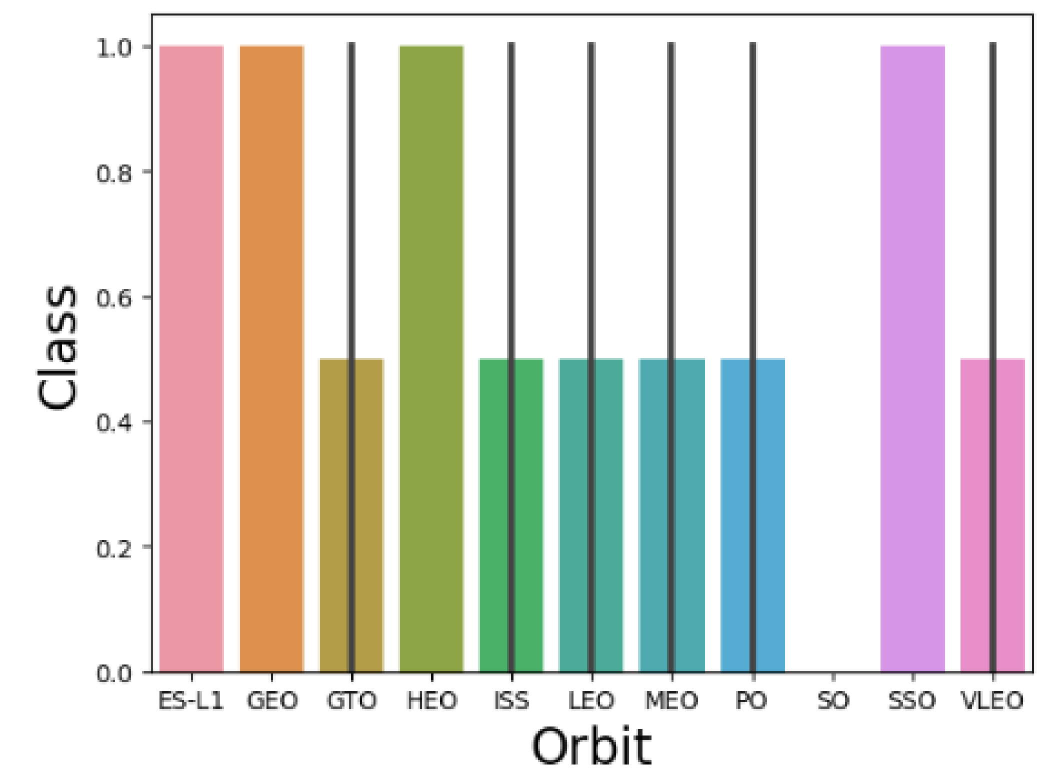
Payload vs. Launch Site

- Scatter plot shows that once the payload is above 7000 kg, the probability of success rate increases
- For the VAFB-SLC launchsite there are no rockets launched for heavy payload mass(greater than 10000)



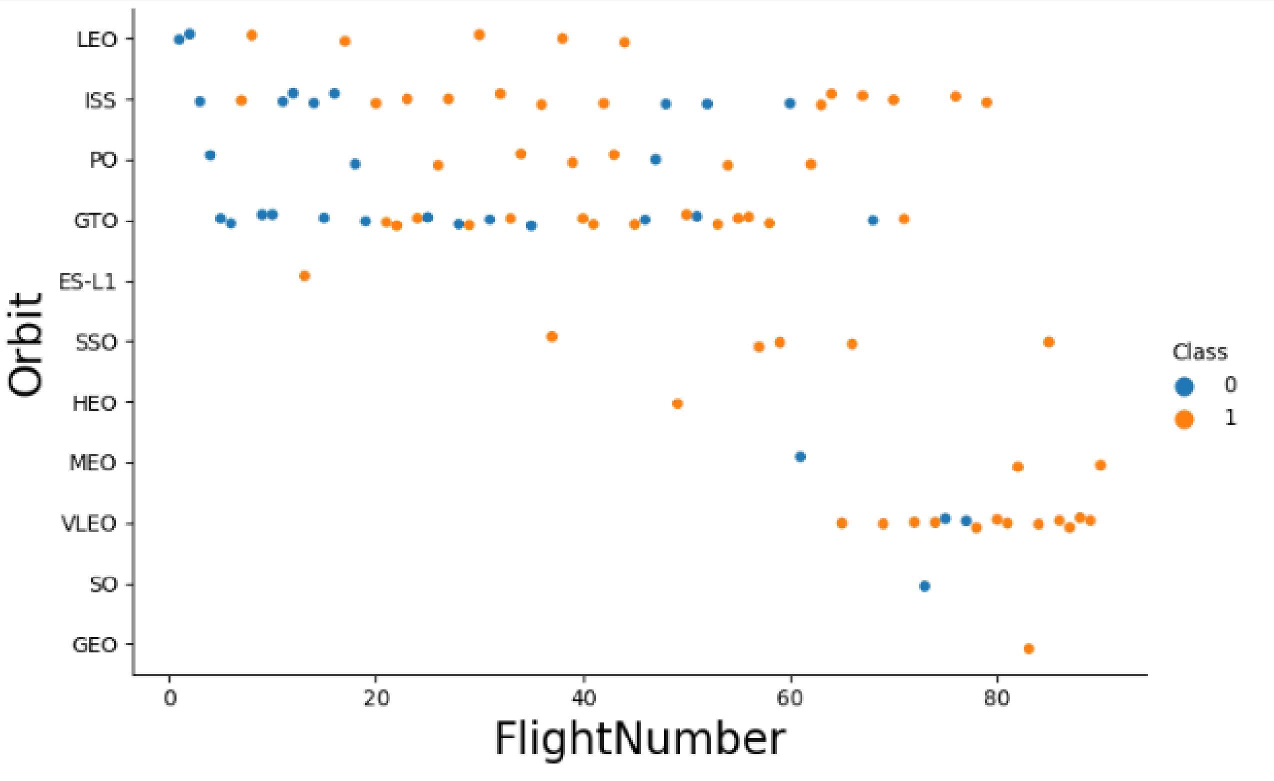
Success Rate vs. Orbit Type

- This bar chart depicts that orbits such as ES-L1, GEO, HEO, and SSO have landing outcomes of 100%
- Further investigation shows that some of those orbits only have one occurrence so more data would be needed to accurately depict 100%
- The remaining orbits (GTO, ISS, LEO, MEO, PO, VLEO) have landing outcomes of 50%
- SO is the only orbit with 0%



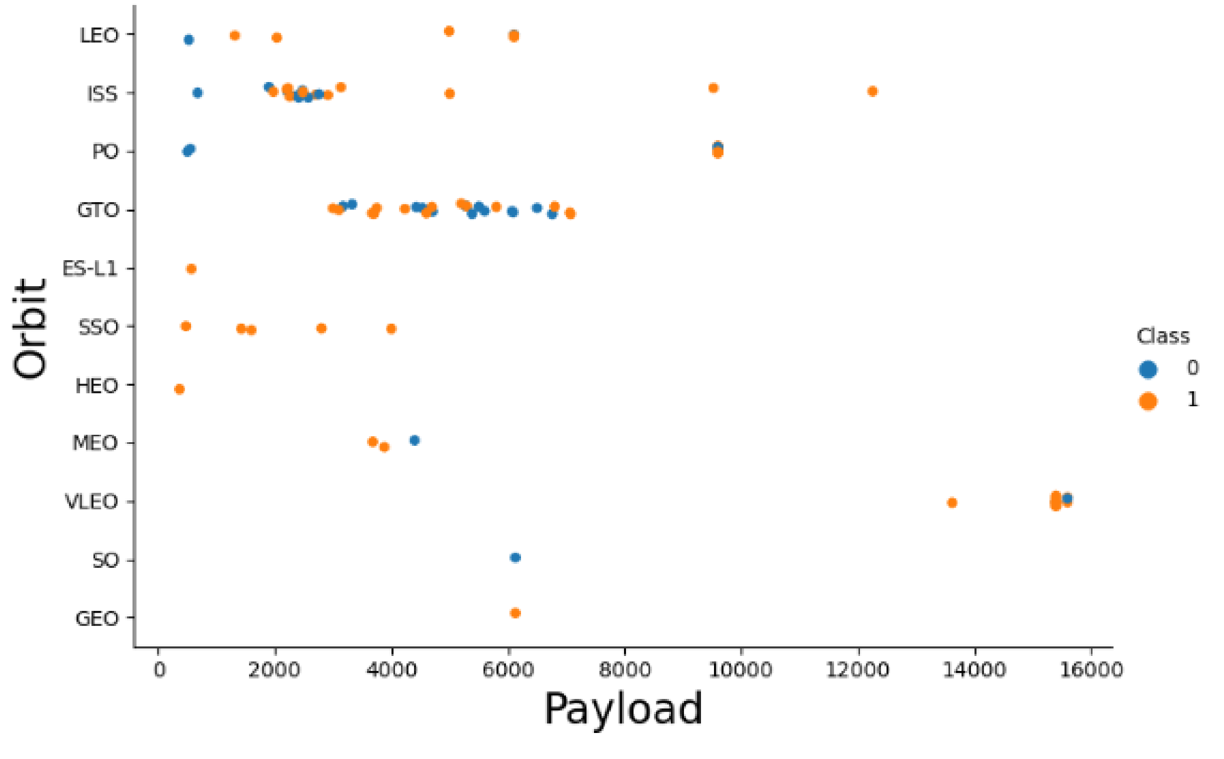
Flight Number vs. Orbit Type

- Scatter plot shows in the LEO orbit the success appears related to the flight number; on the other hand, there seems to be no relationship between flight number when in GTO orbit.



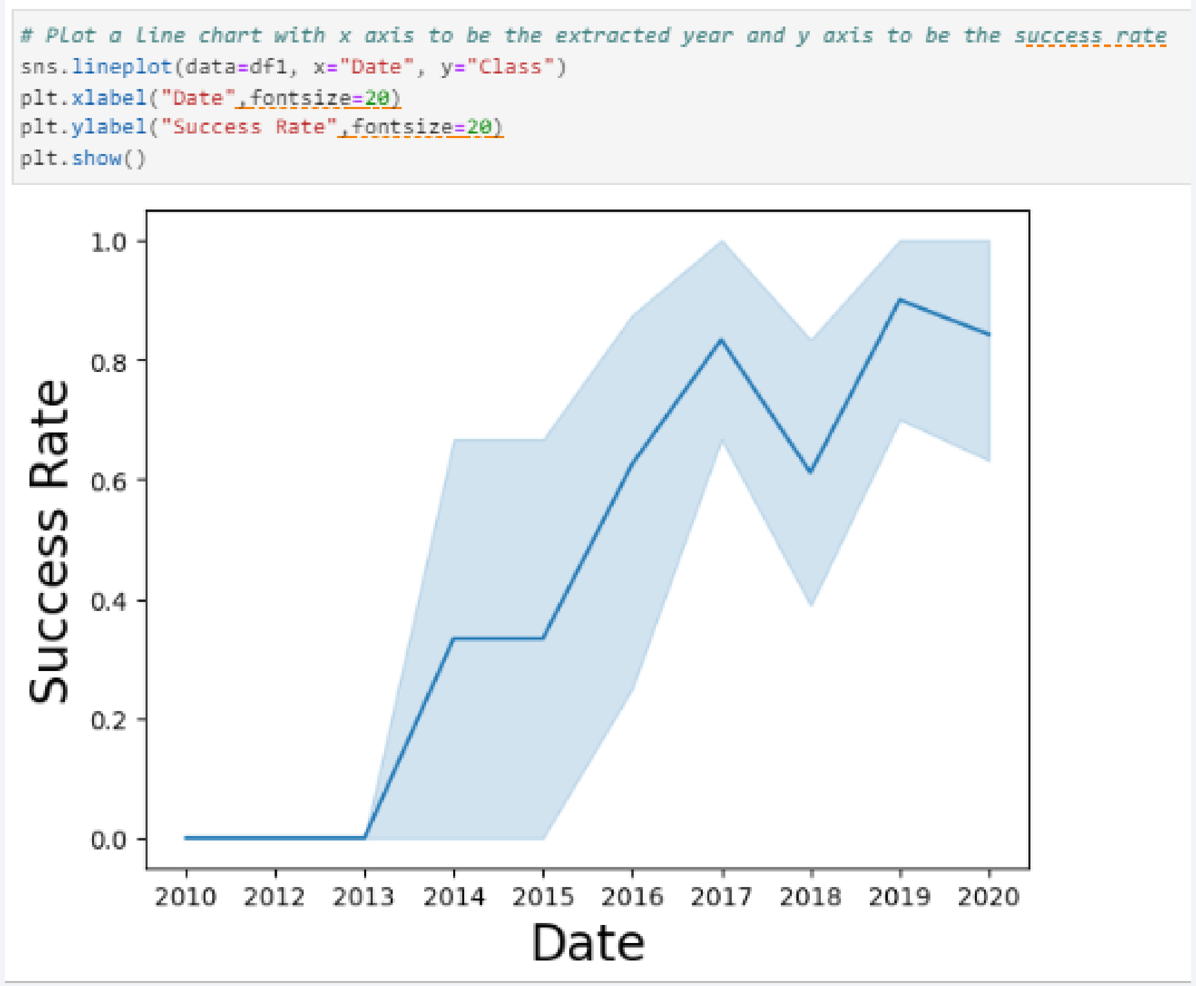
Payload vs. Orbit Type

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



Launch Success Yearly Trend

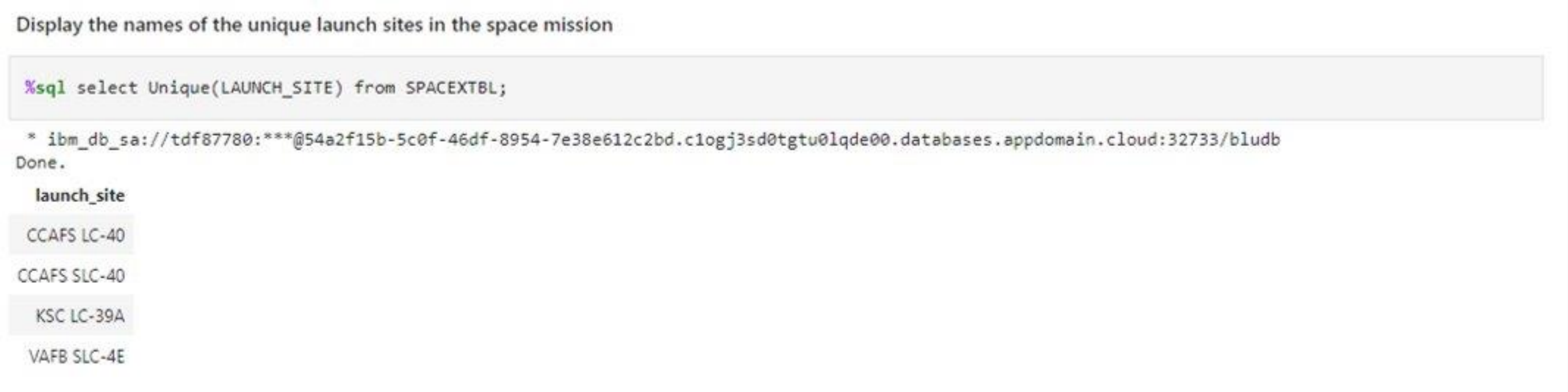
- Line chart of yearly average success rate from 2010 to 2020 with a steady increase starting in 2013



```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
sns.lineplot(data=df1, x="Date", y="Class")
plt.xlabel("Date",fontsize=20)
plt.ylabel("Success Rate",fontsize=20)
plt.show()
```

All Launch Site Names

- Use the “Unique” command to display launch sites



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

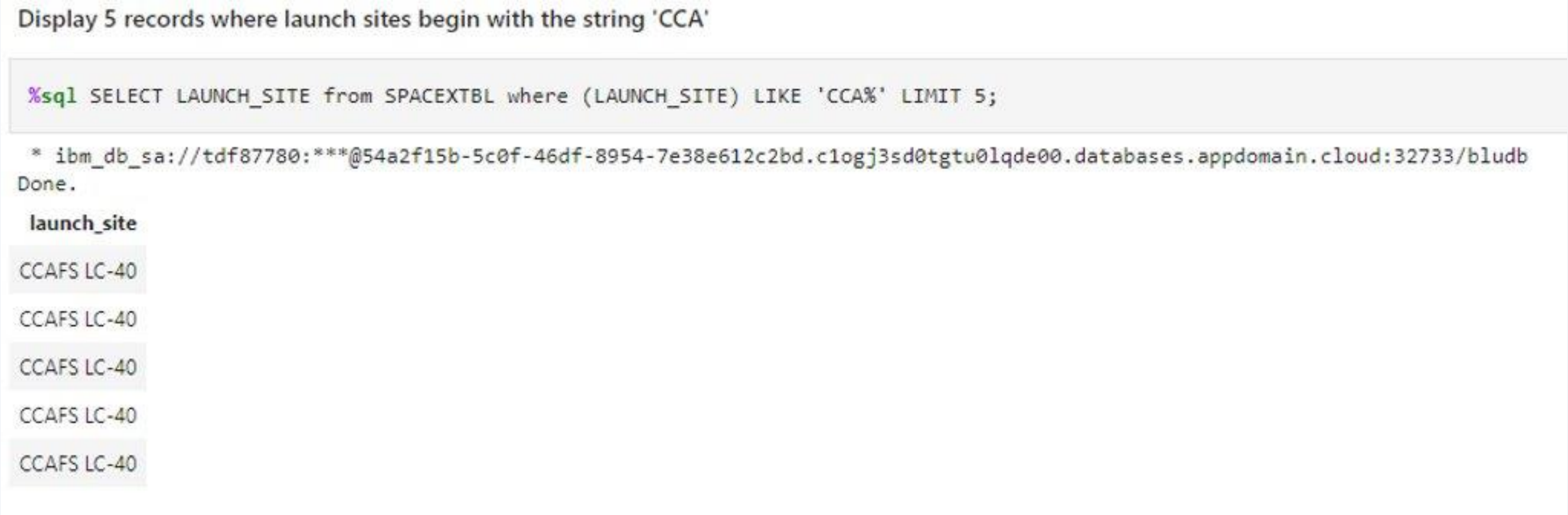
```
%sql select Unique(LAUNCH_SITE) from SPACEXTBL;
```

```
 * ibm_db_sa://tdf87780:***@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'

- Use the “like” command to find the launch sites with CCA in the name



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

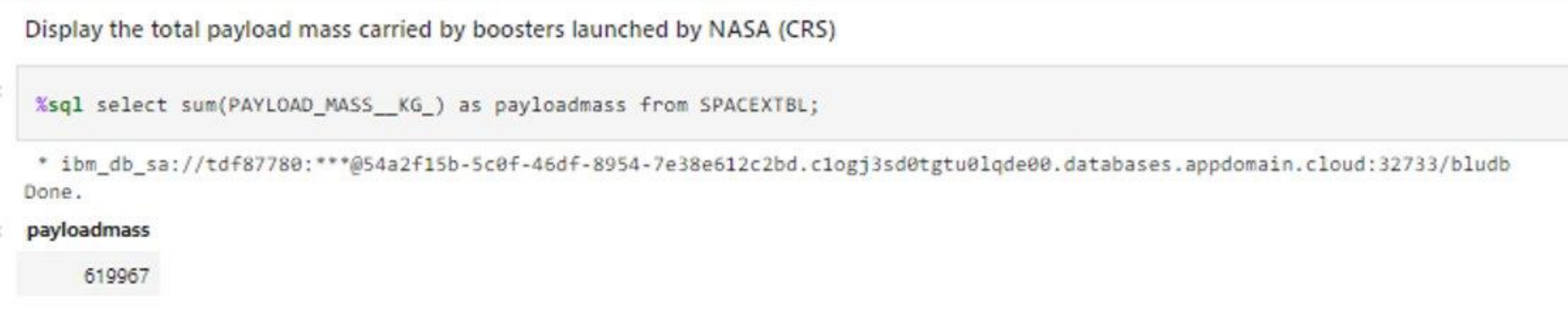
```
%sql SELECT LAUNCH_SITE from SPACEXTBL where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5;
```

```
 * ibm_db_sa://tdf87780:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

launch_site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Total Payload Mass

- Calculate the total payload carried by boosters from NASA



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

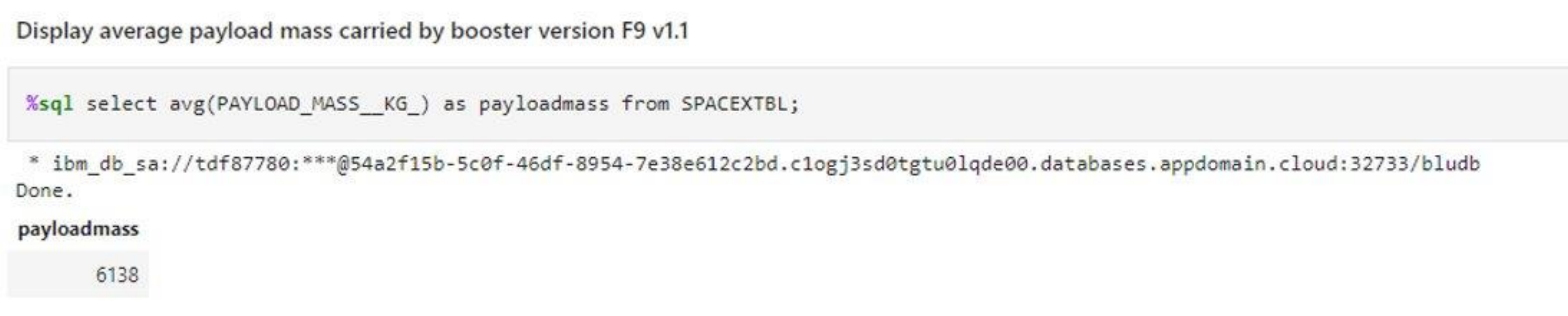
```
%sql select sum(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL;
```

```
 * ibm_db_sa://tdf87780:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

payloadmass
619967

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1 using “avg”



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

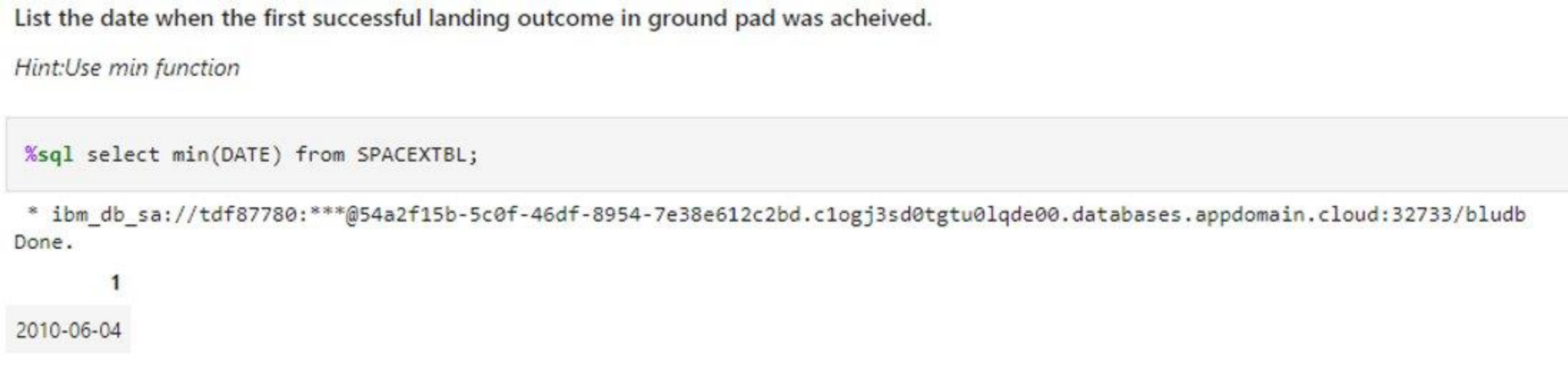
```
%sql select avg(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL;
```

```
 * ibm_db_sa://tdf87780:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

payloadmass
6138

First Successful Ground Landing Date

- List the date of the first successful landing outcome on ground pad using “min(date)”



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

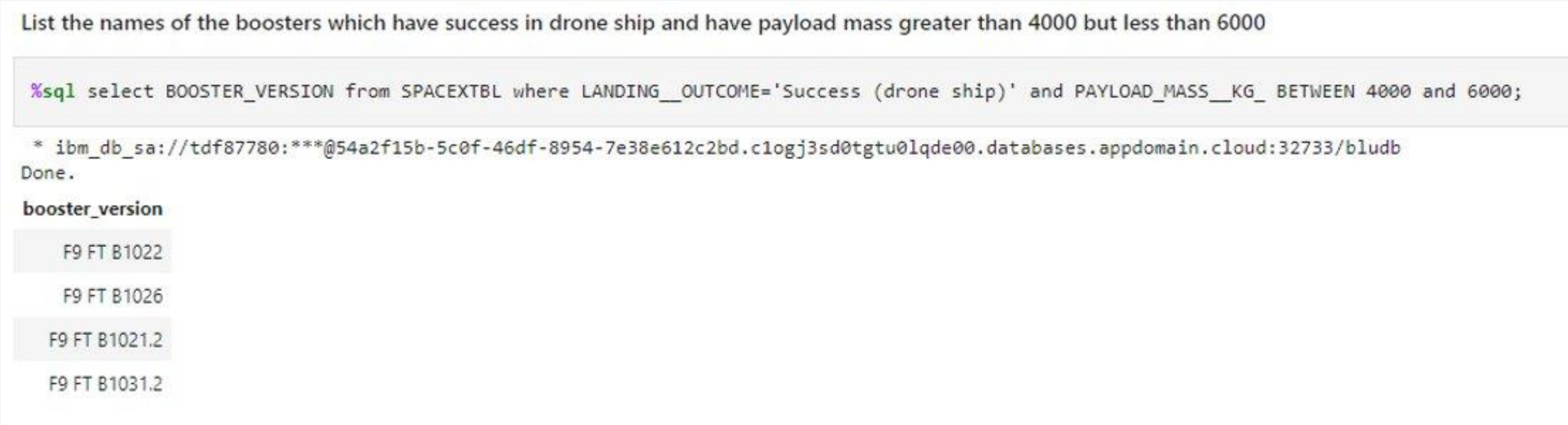
Hint:Use min function

```
%sql select min(DATE) from SPACEXTBL;
```

```
 * ibm_db_sa://tdf87780:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
         1
2010-06-04
```

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



List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

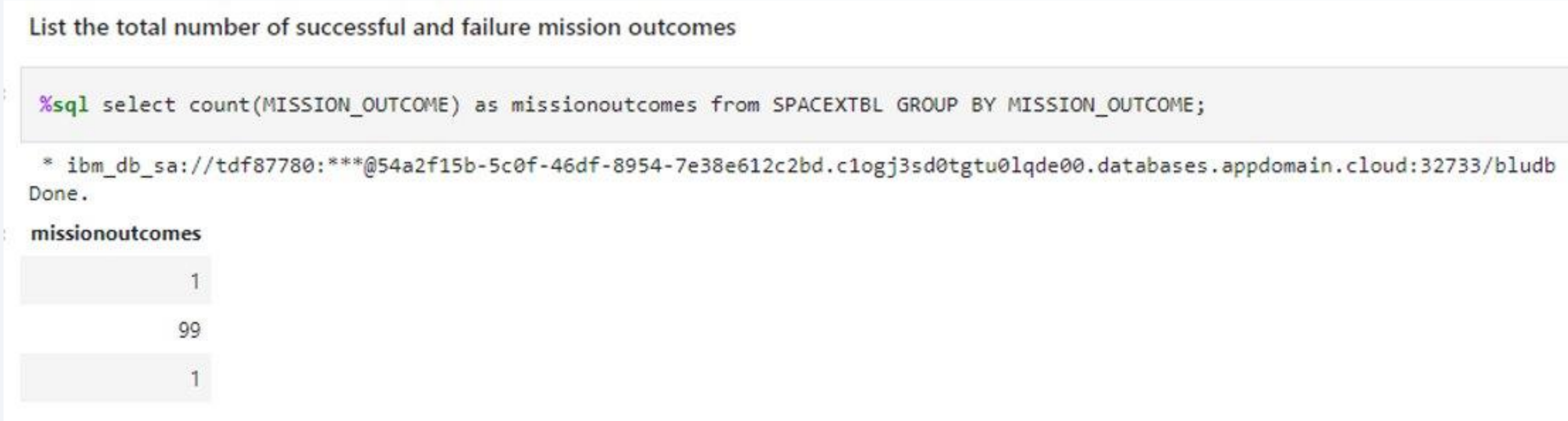
```
%sql select BOOSTER_VERSION from SPACEXTBL where LANDING__OUTCOME='Success (drone ship)' and PAYLOAD_MASS__KG_ BETWEEN 4000 and 6000;
```

```
 * ibm_db_sa://tdf87780:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

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

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes



List the total number of successful and failure mission outcomes

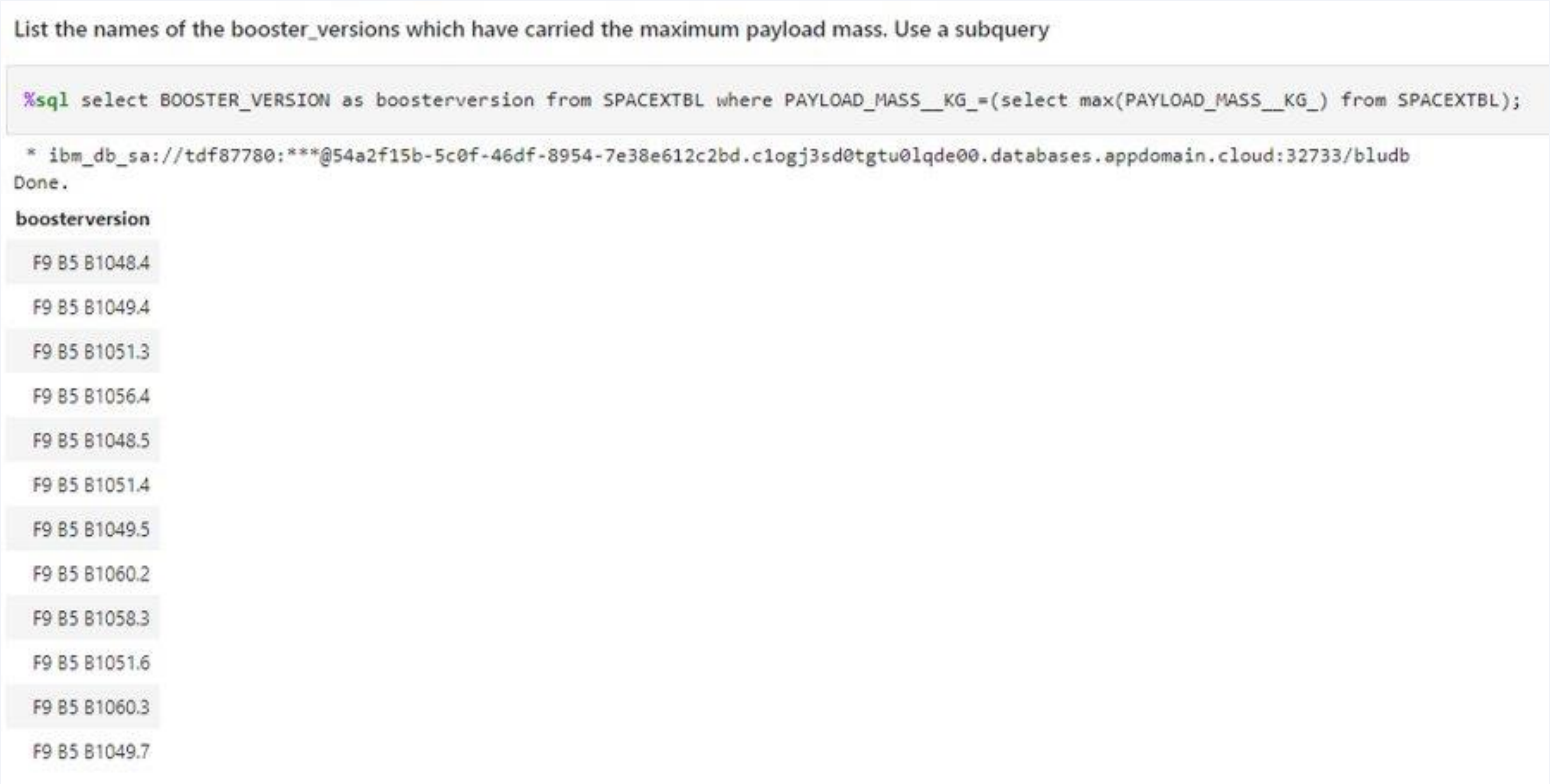
```
%sql select count(MISSION_OUTCOME) as missionoutcomes from SPACEXTBL GROUP BY MISSION_OUTCOME;
```

```
 * ibm_db_sa://tdf87780:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

missionoutcomes
1
99
1

Boosters Carried Maximum Payload

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



List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

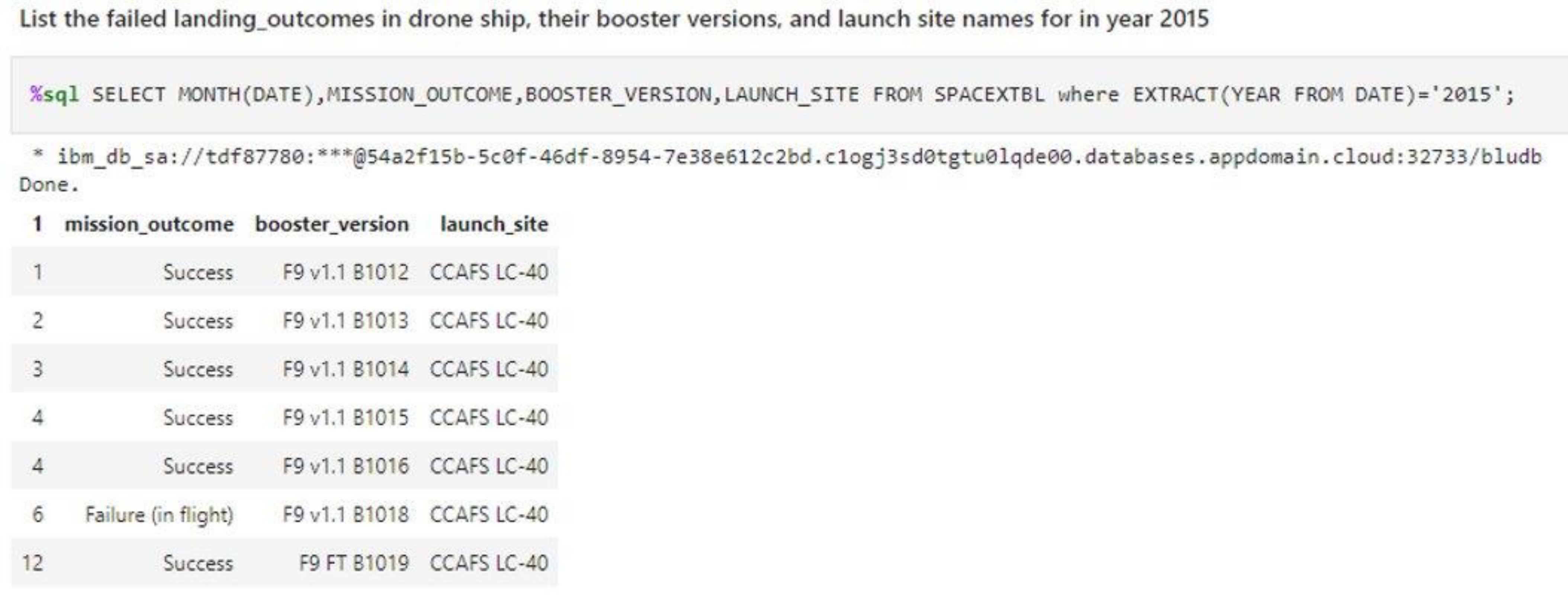
```
%sql select BOOSTER_VERSION as boosterversion from SPACEXTBL where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACEXTBL);
```

```
 * ibm_db_sa://tdf87780:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

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

2015 Launch Records

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



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

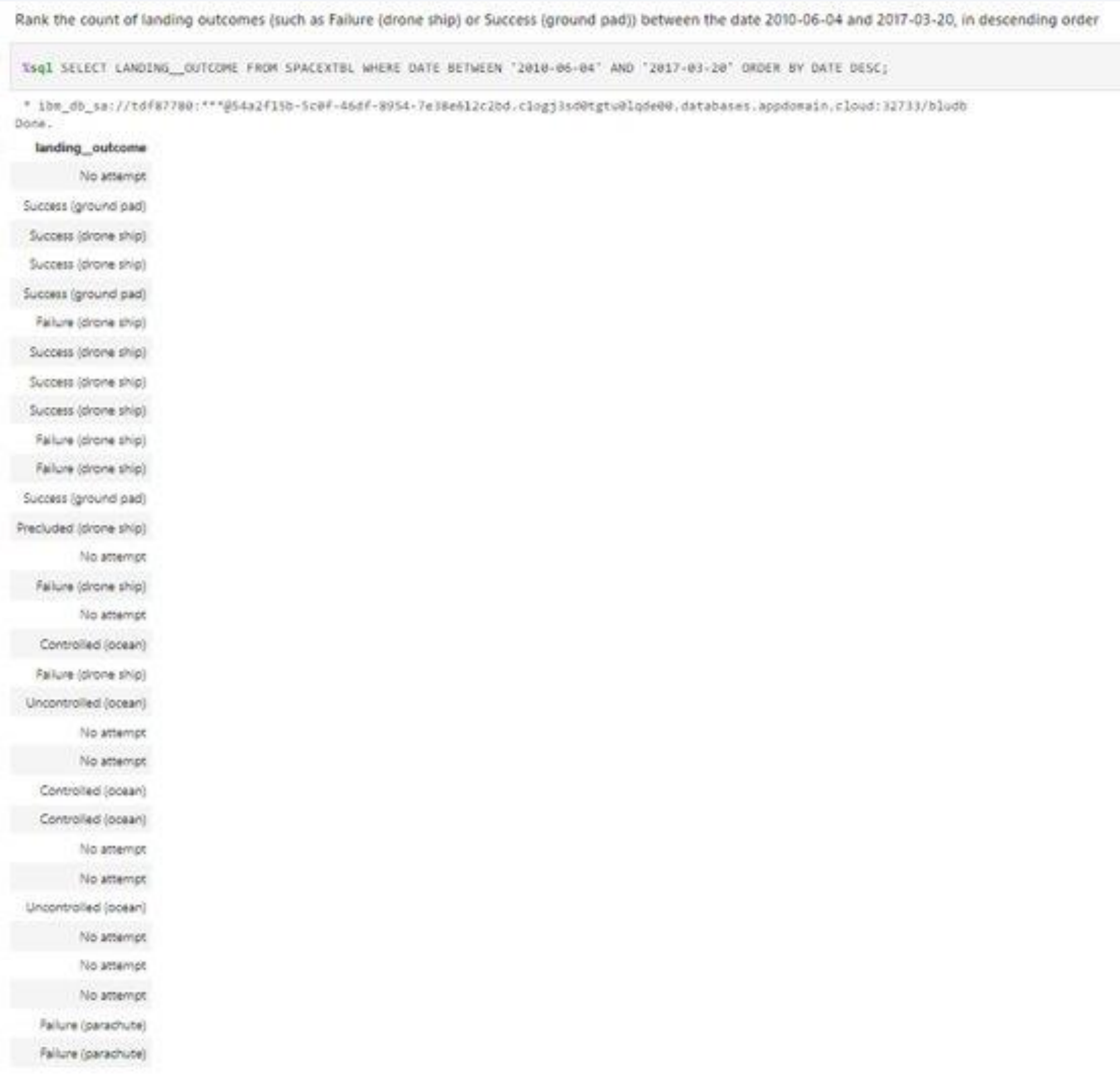
```
%sql SELECT MONTH(DATE),MISSION_OUTCOME,BOOSTER_VERSION,LAUNCH_SITE FROM SPACEXTBL where EXTRACT(YEAR FROM DATE)='2015';
```

```
 * ibm_db_sa://tdf87780:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

1	mission_outcome	booster_version	launch_site
1	Success	F9 v1.1 B1012	CCAFS LC-40
2	Success	F9 v1.1 B1013	CCAFS LC-40
3	Success	F9 v1.1 B1014	CCAFS LC-40
4	Success	F9 v1.1 B1015	CCAFS LC-40
4	Success	F9 v1.1 B1016	CCAFS LC-40
6	Failure (in flight)	F9 v1.1 B1018	CCAFS LC-40
12	Success	F9 FT B1019	CCAFS LC-40

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



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

```
%sql SELECT LANDING__OUTCOME FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' ORDER BY DATE DESC;
```

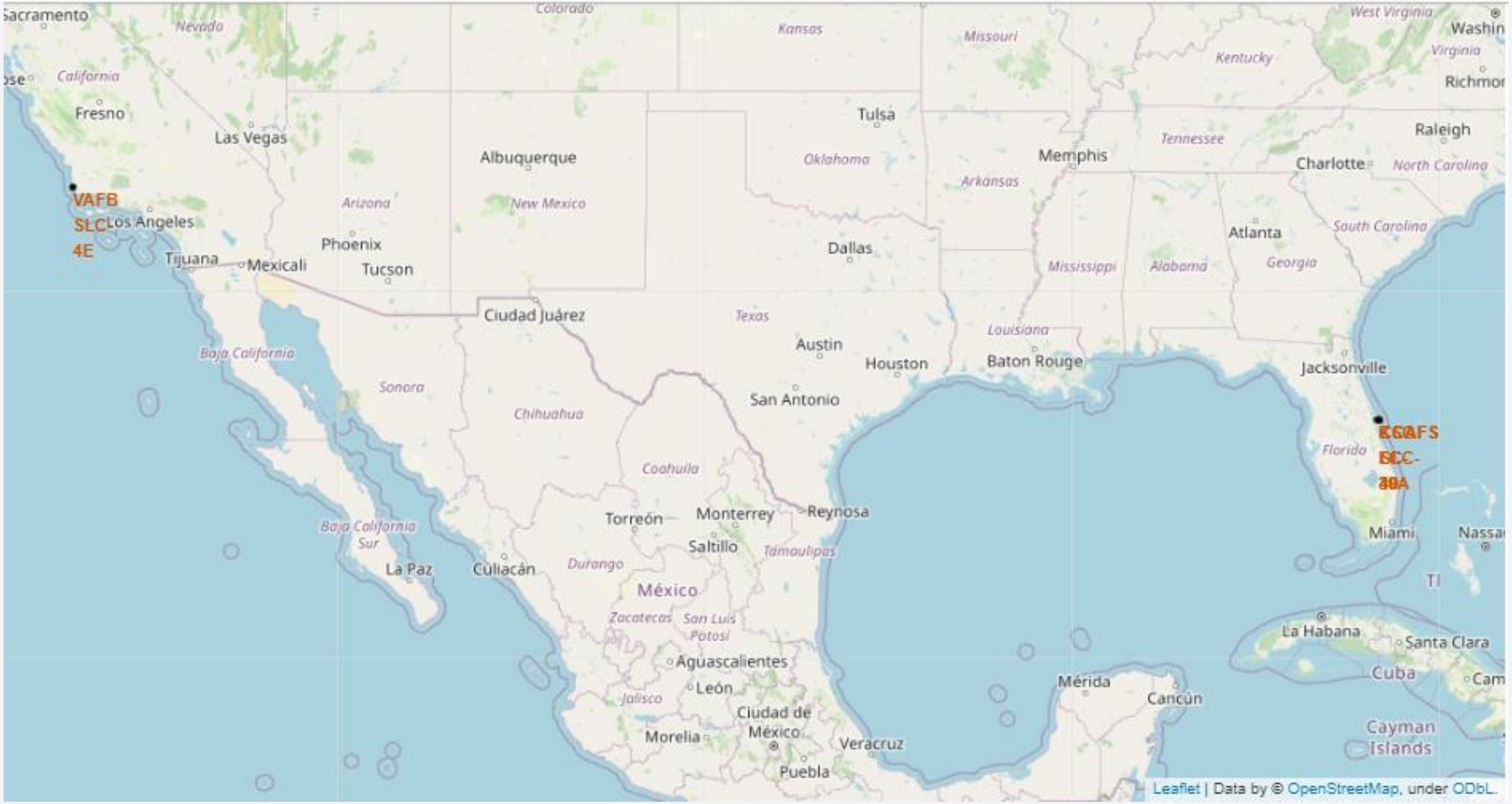
```
 * ibm_db_sa://tdf87780:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

landing__outcome
No attempt
Success (ground pad)
Success (drone ship)
Success (drone ship)
Success (ground pad)
Failure (drone ship)
Success (drone ship)
Success (drone ship)
Success (drone ship)
Failure (drone ship)
Failure (drone ship)
Success (ground pad)
Precluded (drone ship)
No attempt
Failure (drone ship)
No attempt
Controlled (ocean)
Failure (drone ship)
Uncontrolled (ocean)
No attempt
No attempt
Controlled (ocean)
Controlled (ocean)
No attempt
No attempt
Uncontrolled (ocean)
No attempt
No attempt
No attempt
Failure (parachute)
Failure (parachute)

Section 3

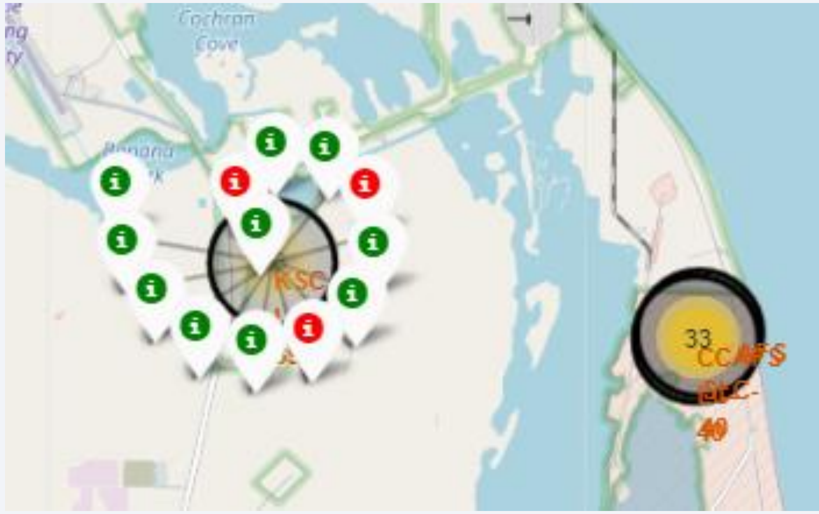
Launch Sites Proximities Analysis

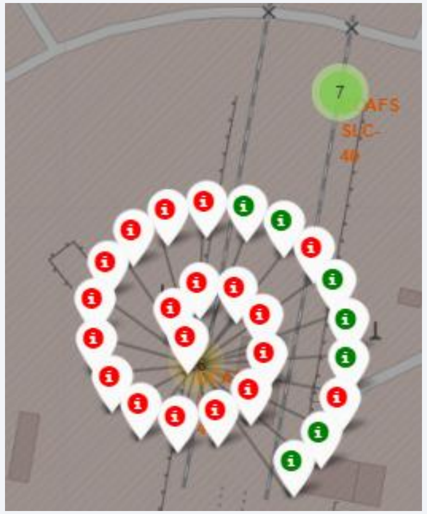
Location of Launch Sites

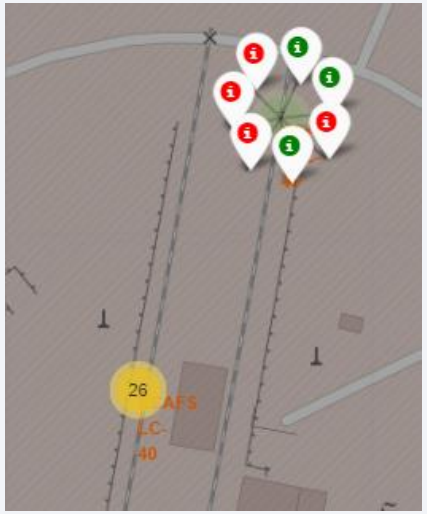


Launch Sites with Markers Showing Success/Unsuccess

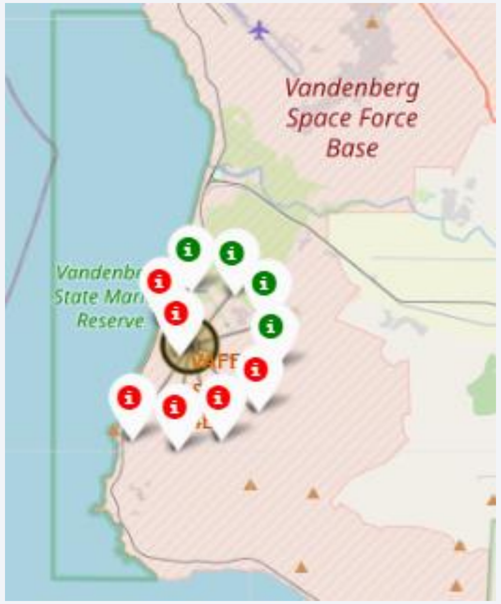
- Florida Site Launches:
  - Green – successful
  - Red - unsuccessful



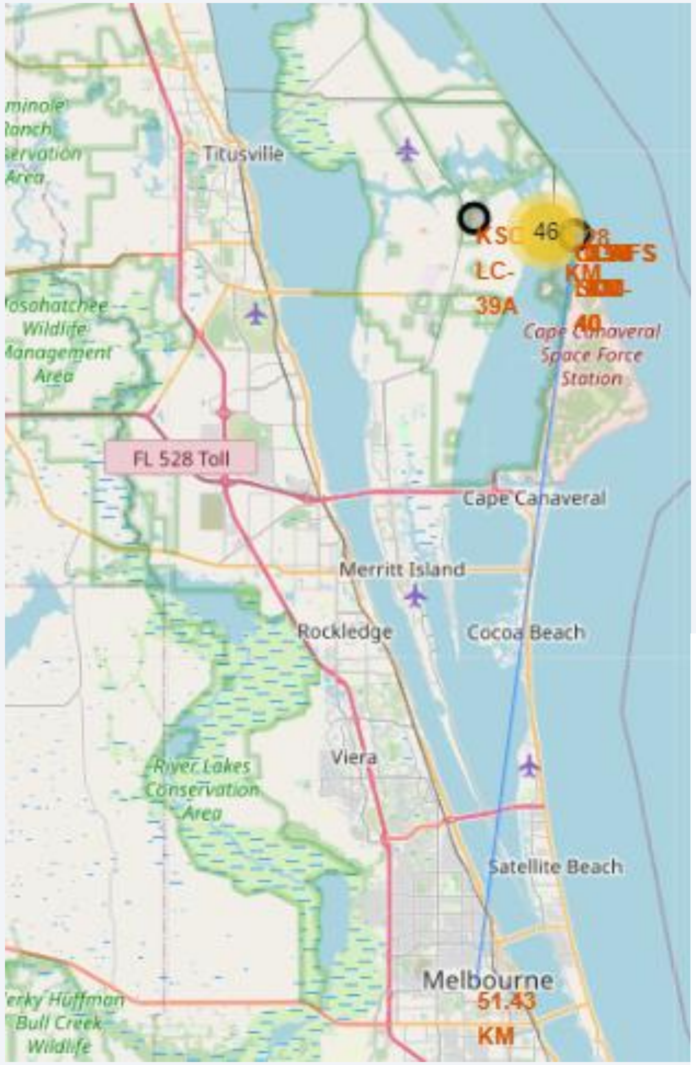


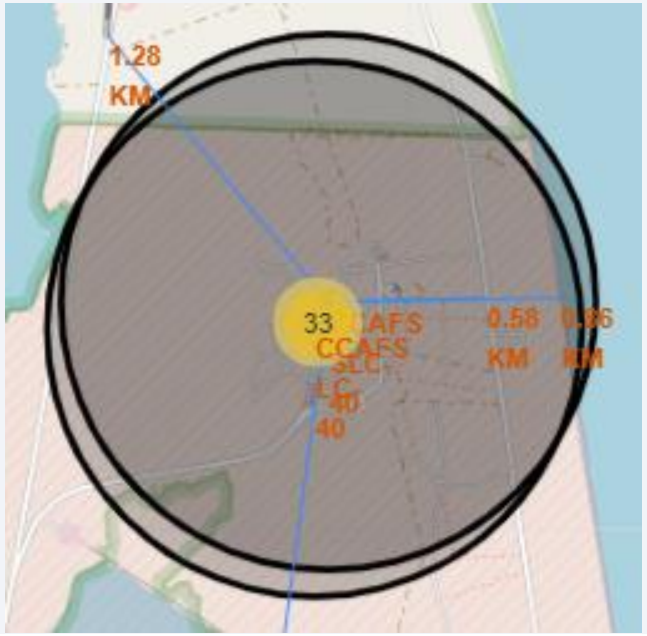


- California Site Launches:
  - Green – successful
  - Red - unsuccessful



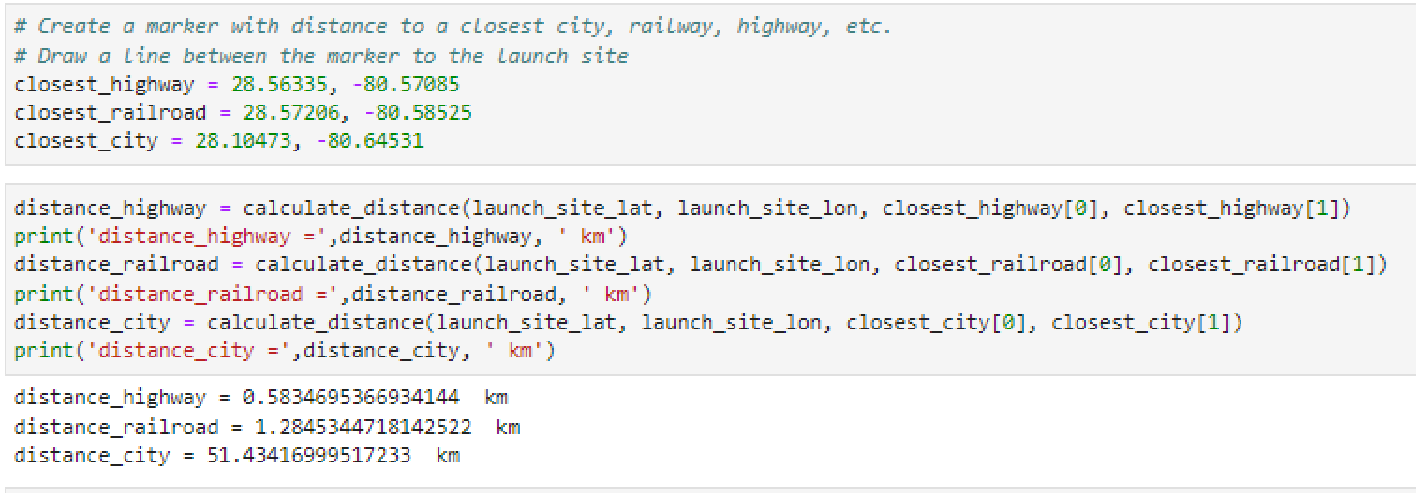
Launch Site Proximity to Landmarks





Screenshots of distance marks, with code below:

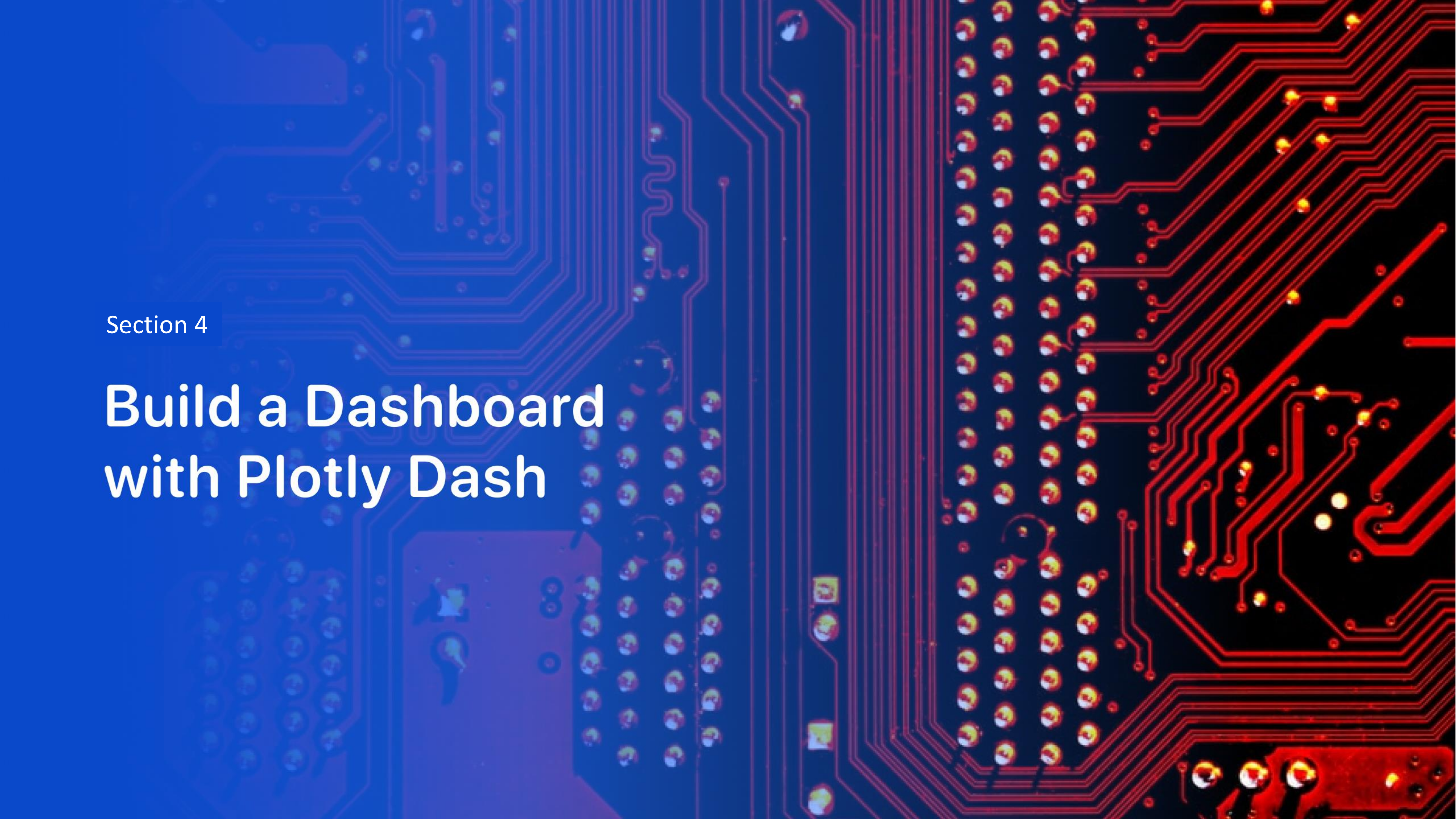
- Proximity to railways? **1.28km**
- Proximity to highways? **0.58 km**
- Proximity to coastline? **0.86 km**
- Do launch sites keep certain distance away from cities? **51.43 km**



```
# Create a marker with distance to a closest city, railway, highway, etc.
# Draw a line between the marker to the launch site
closest_highway = 28.56335, -80.57085
closest_railroad = 28.57206, -80.58525
closest_city = 28.10473, -80.64531
```

```
distance_highway = calculate_distance(launch_site_lat, launch_site_lon, closest_highway[0], closest_highway[1])
print('distance_highway =',distance_highway, ' km')
distance_railroad = calculate_distance(launch_site_lat, launch_site_lon, closest_railroad[0], closest_railroad[1])
print('distance_railroad =',distance_railroad, ' km')
distance_city = calculate_distance(launch_site_lat, launch_site_lon, closest_city[0], closest_city[1])
print('distance_city =',distance_city, ' km')
```

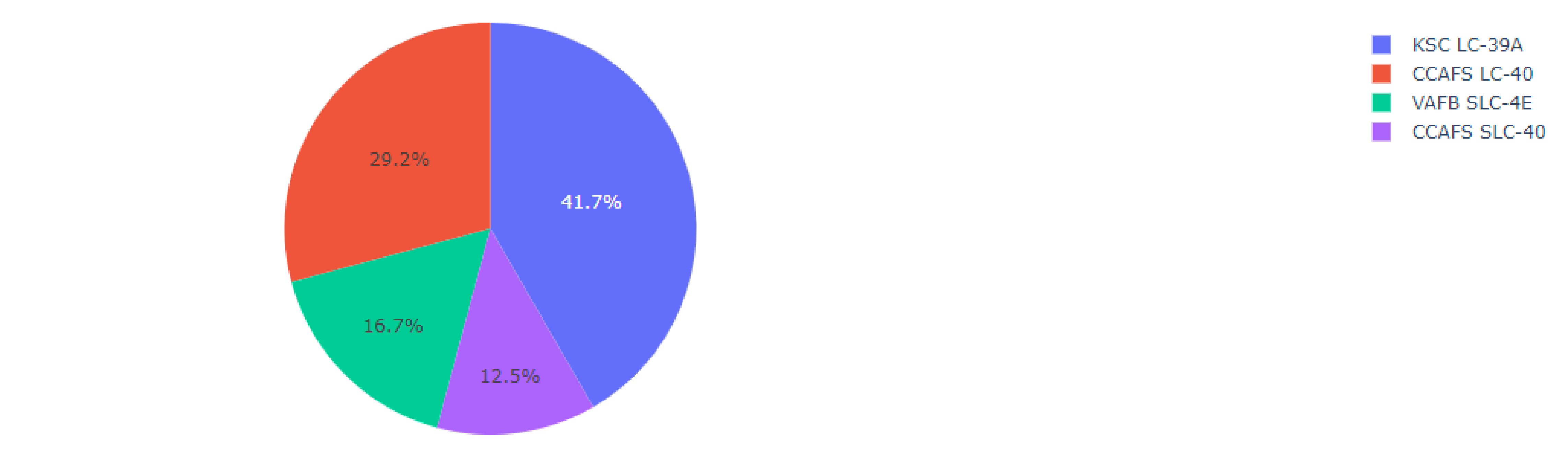
```
distance_highway = 0.5834695366934144  km
distance_railroad = 1.2845344718142522  km
distance_city = 51.43416999517233  km
```



Section 4

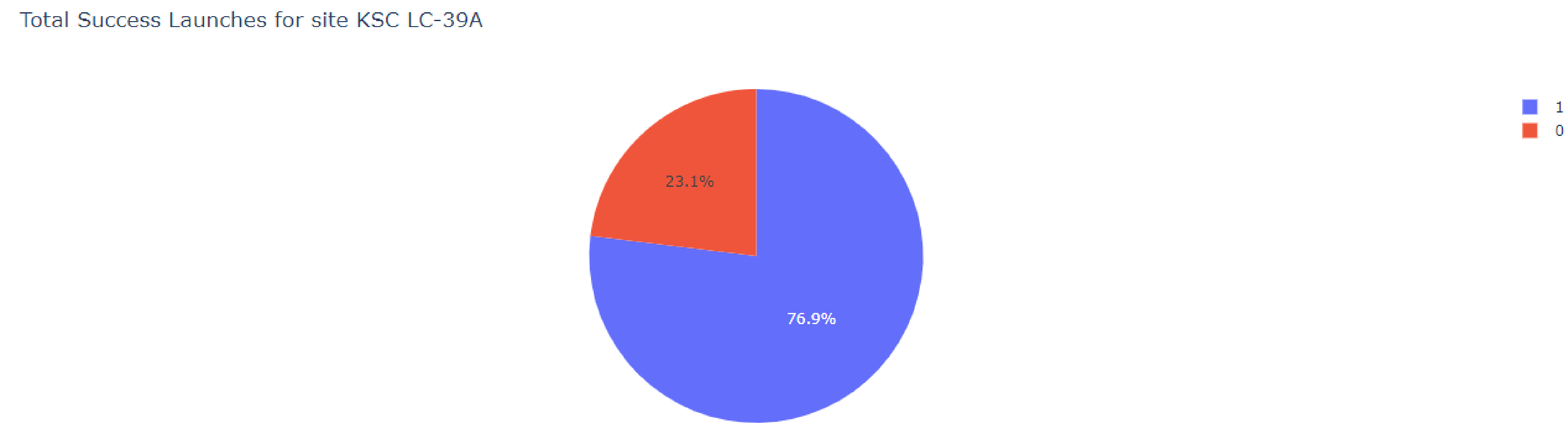
Build a Dashboard with Plotly Dash

Launch Site Success Percentage



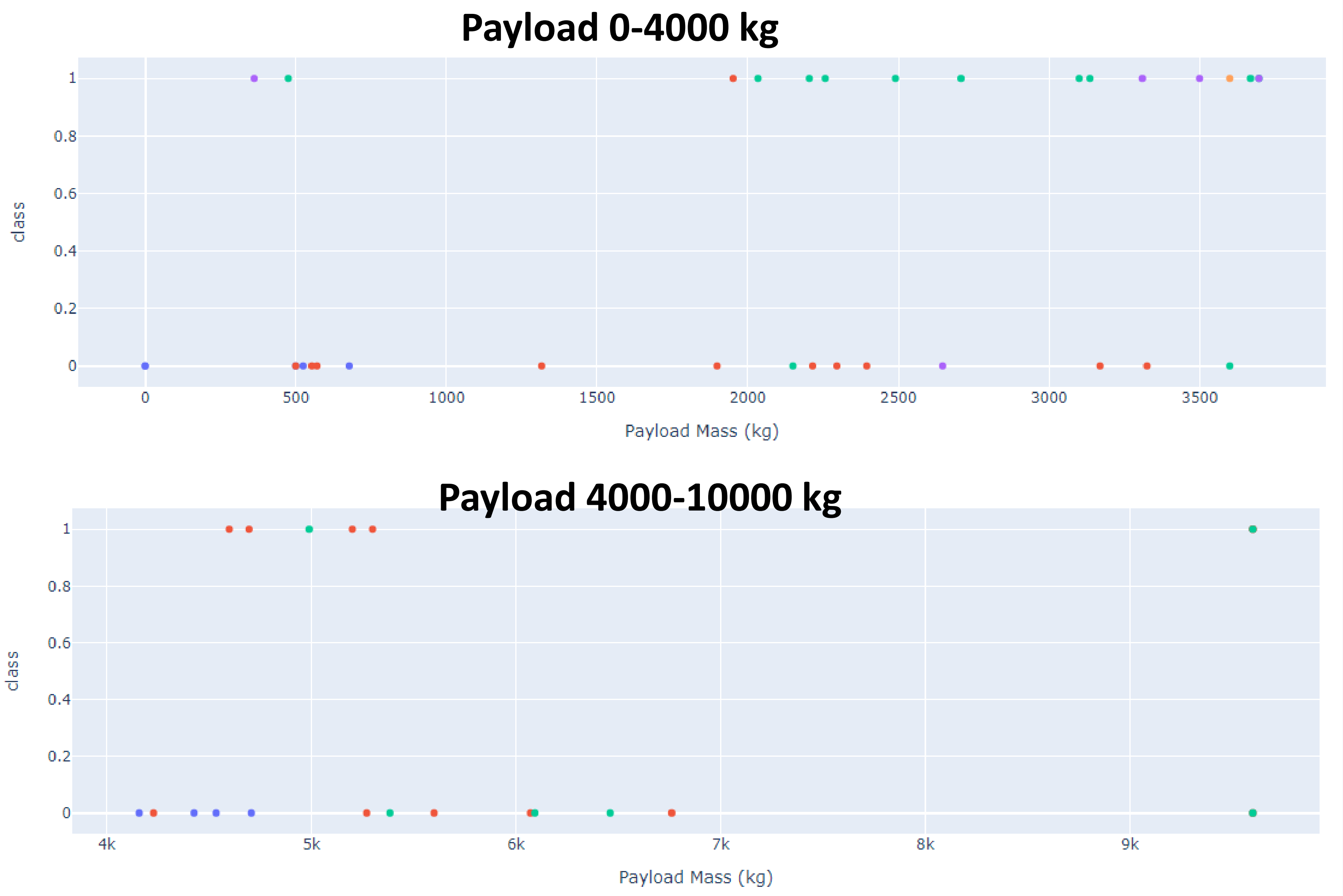
- KSC LC-39A had the largest success percentage with 41.7%

Highest Launch Success Ratio



- KSC LC-39A Success and Failure Rate Percentages

Payload vs Launch Outcome Visualization

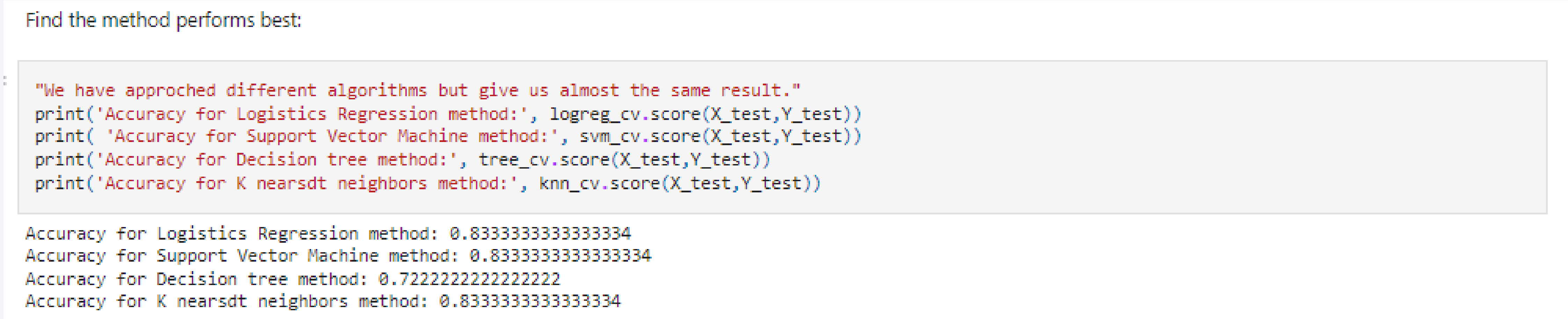


- Success rate for lower payloads is higher than the heavier payloads

Section 5

Predictive Analysis (Classification)

Classification Accuracy



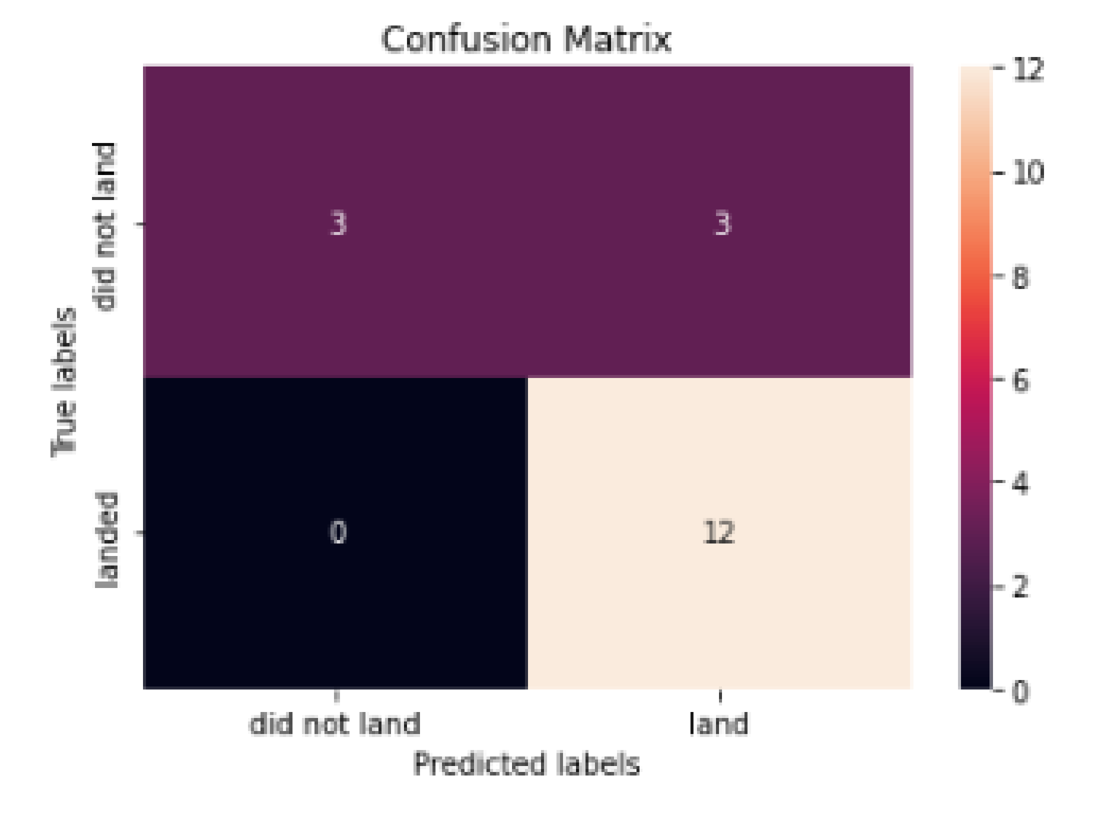
Find the method performs best:

```
"We have approched different algorithms but give us almost the same result."
print('Accuracy for Logistics Regression method:', logreg_cv.score(X_test,Y_test))
print( 'Accuracy for Support Vector Machine method:', svm_cv.score(X_test,Y_test))
print('Accuracy for Decision tree method:', tree_cv.score(X_test,Y_test))
print('Accuracy for K nearsdt neighbors method:', knn_cv.score(X_test,Y_test))
```

```
Accuracy for Logistics Regression method: 0.8333333333333334
Accuracy for Support Vector Machine method: 0.8333333333333334
Accuracy for Decision tree method: 0.7222222222222222
Accuracy for K nearsdt neighbors method: 0.8333333333333334
```

- As can be seen from the code above, all but the decision tree have almost identical classification accuracy scores

Confusion Matrix



- Examining the confusion matrix, we see that it can distinguish between the different classes; however, we see that the major problem is false positives (or unsuccessful landing identified as successful)

Conclusions

- Starting from the year 2013 there was a steady increase in successful SpaceX launches
- SSO has the highest success rate of 100% with more than 1 outcome
- KSC LC-39A had the highest success rate of 76.9%
- Lower weighted payloads (0 -4000 kg) had a higher success rate than the heavier payloads
- The classification algorithms performed roughly the same, with the tree having the lowest accuracy at 0.72

Appendix

- GitHub URL for all code:
    - https://github.com/samunns/datasciencecap

Thank you!