IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

I. Daniela Beinarovica
11.05.2023

Outline

Executive Summary

- Summary of methodologies
  - Data collection using APIs & Web scraping
  - Data wrangling to remove unneeded data and missing values
  - Exploratory Data analysis using SQL and Data visualizations
  - Interactive Visual analytics with Folium
  - Machine learning with Logistic regression, Support Vector Machine, Decision tree and K closes neighbor methods
- Summary of all results
  - The most used launch site is also the one which has the highest success rate: KSC LC-39A
  - Orbits ES-L1, GEO, HEO, SSO and VLEO have the highest average success rate.
  - Most launches (per all sites) carries a payload mass below 9000 kg and have had a success rate of over 80%.

Introduction

SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars which is relatively affordable when compared to the 165+ million dollars price tag set by other providers. These savings are due to the SpaceX being able to re-use the first stage of the launch-vehicle.

Since SpaceX data is publicly available, other providers can use it to improve their technologies and lower the cost thus becoming more competitive by using their resources more rationally.

This leads to the main question:

what factors are likely lead to the first stage launch-vehicle landing successfully?

Section 1

Methodology

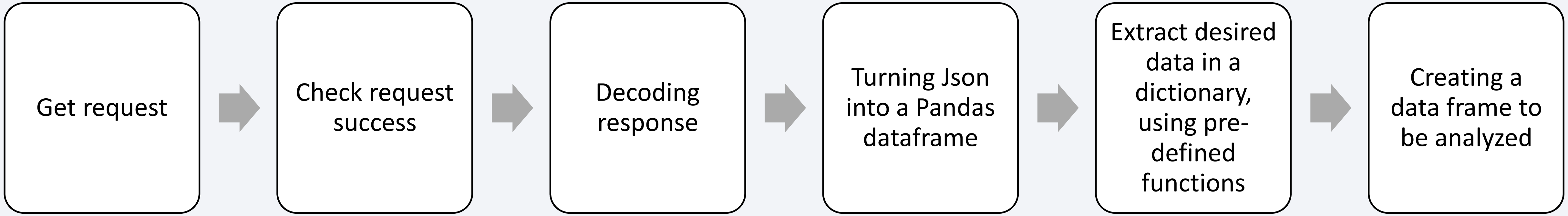
Methodology

Executive Summary

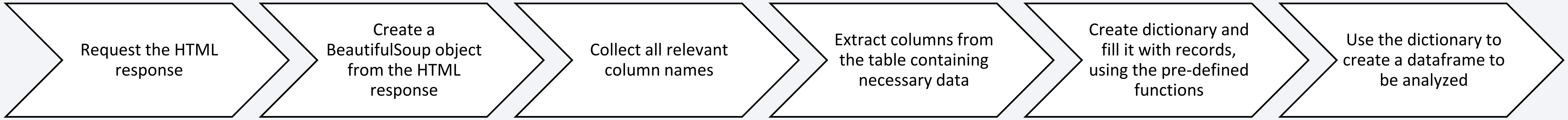
- Data collection methodology:
  - SpaceX API and webscraping the SpaceX Wikipedia page
- Perform data wrangling
  - Data was normalized & cleaned by getting rid of unnecessary data and empty values
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Exploratory Data Analysis, finding the best Hyperparameters and evaluating the methods to determine which performs best

Data Collection

- SpaceX API was used for requesting and parsing the data :

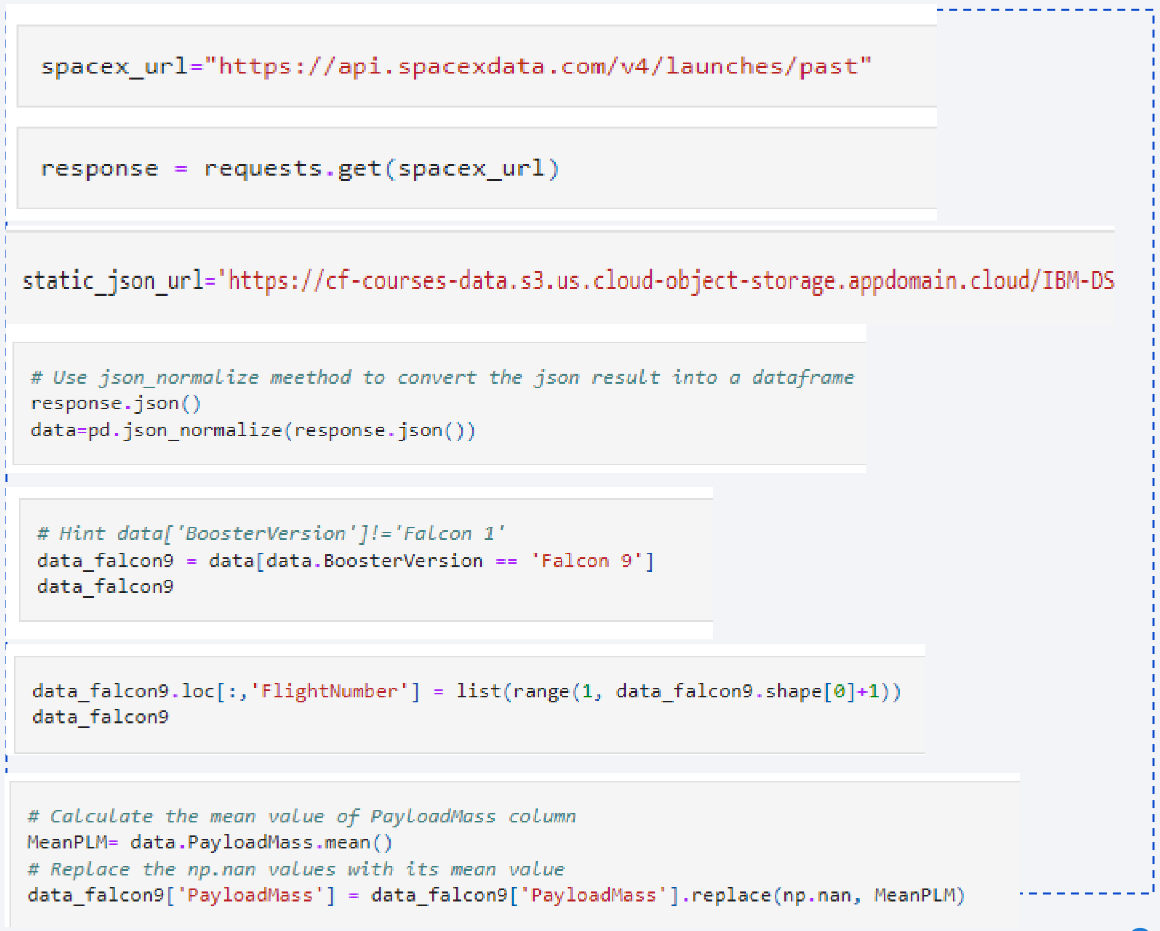

Get request
Check request success
Decoding response
Turning Json into a Pandas dataframe
Extract desired data in a dictionary, using pre-defined functions
Creating a data frame to be analyzed

- as well as by webscraping the SpaceX Wikipedia page


Request the HTML response
Create a BeautifulSoup object from the HTML response
Collect all relevant column names
Extract columns from the table containing necessary data
Create dictionary and fill it with records, using the pre-defined functions
Use the dictionary to create a dataframe to be analyzed

Data Collection – SpaceX API

- The data was collected using the SpaceX API and further extracting the specific data needed for this analysis.
- The extracted data was wrangled and formatted, e.g., by filtering out unnecessary data and deleting entries with missing values.
- The notebook can be accessed here: [Lab1](Lab1).



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

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

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS
```

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

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = data[data.BoosterVersion == 'Falcon 9']
data_falcon9
```

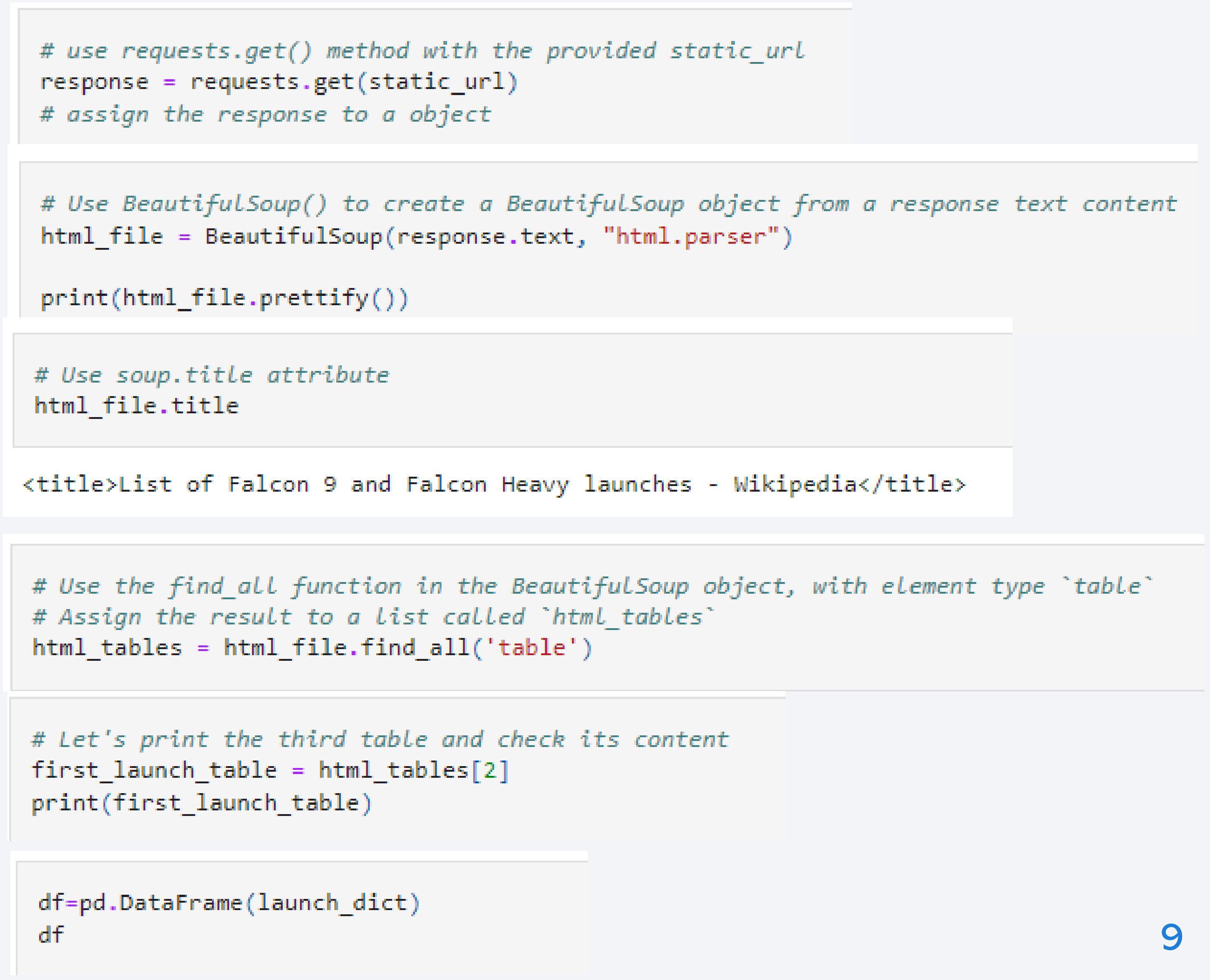
```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

```
# Calculate the mean value of PayloadMass column
MeanPLM= data.PayloadMass.mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, MeanPLM)
```

Data Collection - Scraping

- Data about launches was acquired by performing webscraping on the SpaceX Wikipedia page, locating the table that contains the necessary data and extracting column names and records required for our analysis, which were then stored in a dataframe.

- The notebook is available here: [Lab3]()



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

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
html_file = BeautifulSoup(response.text, "html.parser")

print(html_file.prettify())
```

```
# Use soup.title attribute
html_file.title
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

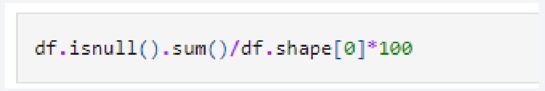
```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = html_file.find_all('table')
```

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

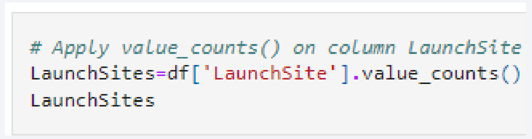
```
df=pd.DataFrame(launch_dict)
df
```

Data Wrangling

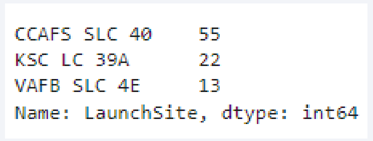
- Data was wrangled by performing Exploratory data analysis
  - Determining the % of missing values for each attribute;
  - Calculating the number of launches for each launch site;
  - Calculating the number and occurrence of each orbit;
  - Calculating the number and occurrence of mission outcome per orbit type;
- and Determining Training Labels
  - And calculating the success rate.
- The notebook can be accessed here: [Lab2](Lab2).



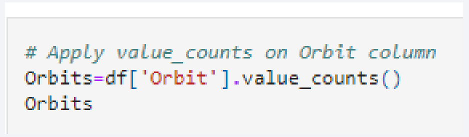
```
df.isnull().sum()/df.shape[0]*100
```



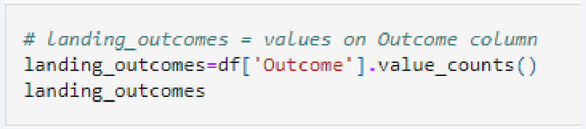
```
# Apply value_counts() on column LaunchSite
LaunchSites=df['LaunchSite'].value_counts()
LaunchSites
```



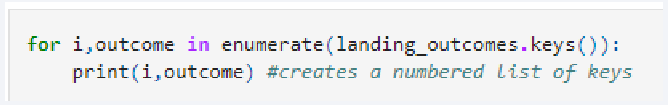
```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```



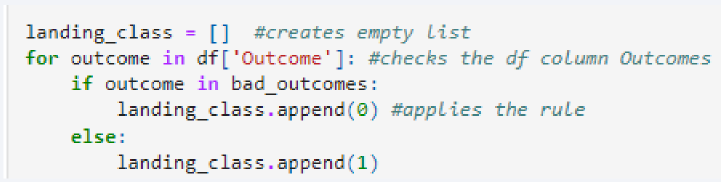
```
# Apply value_counts on Orbit column
Orbits=df['Orbit'].value_counts()
Orbits
```



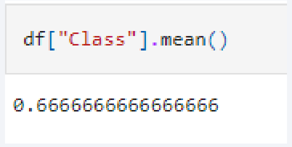
```
# landing_outcomes = values on Outcome column
landing_outcomes=df['Outcome'].value_counts()
landing_outcomes
```



```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome) #creates a numbered list of keys
```



```
landing_class = []  #creates empty list
for outcome in df['Outcome']: #checks the df column Outcomes
    if outcome in bad_outcomes:
        landing_class.append(0) #applies the rule
    else:
        landing_class.append(1)
```

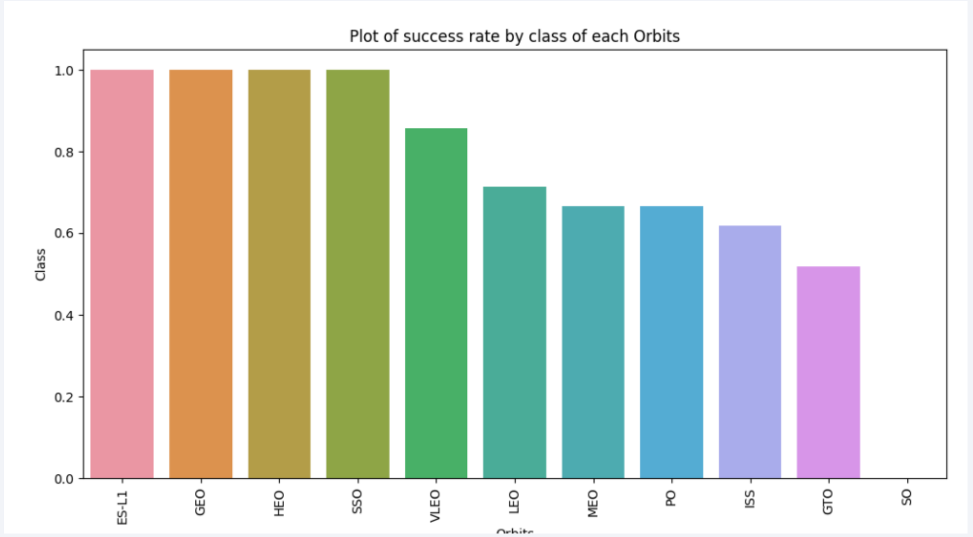


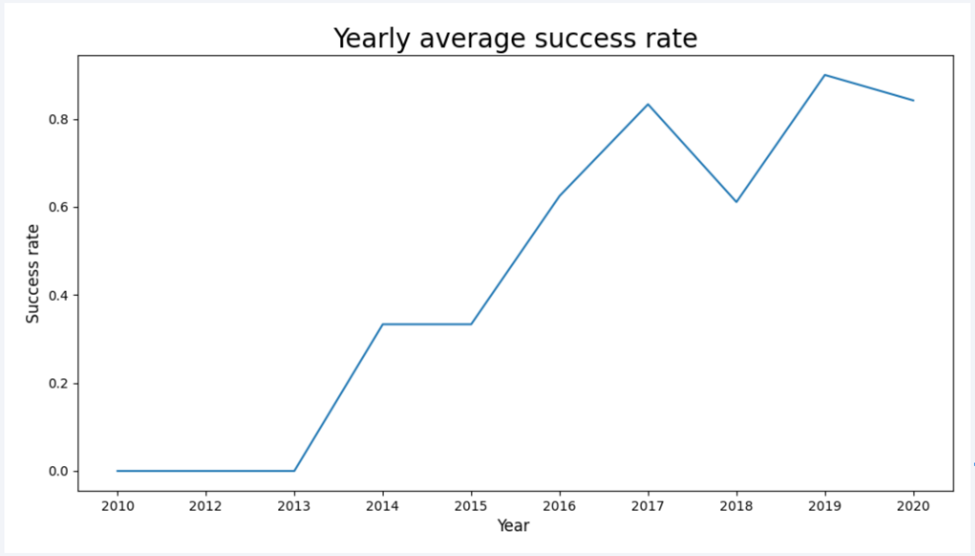
```
df["Class"].mean()
```

```
0.6666666666666666
```

EDA with Data Visualization

- In order to gain first insights, the data was visualized with scatterplots, bar charts and line plots, as these allow to quickly and easily identify patterns.
- In the notebook various relationships between FlightNumber, PayloadMass, LaunchSite and Orbit were visualized.
- Success rate per Orbit, as well as the Yearly success rate were visualized.
- The notebook can be accessed here: Lab4.





EDA with SQL

- The following SQL queries were performed:
  - Identification of unique launch sites
  - Selection of records that contain launch sites beginning with “ CCA” ;
  - Displaying the total payload mass carried by boosters launched by NASA (CRS);
  - Displaying the average payload mass carried by booster version F9 v1.1;
  - Finding the date of the first successful landing outcome;
  - Listing the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000;
  - Listing the total number of successful and failure mission outcomes;
  - Listing the names of the booster_versions which have carried the maximum payload mass. Use a subquery;
  - Listing the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.
  - Ranking the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.
- The notebook is accessible here: Lab5

Build an Interactive Map with Folium

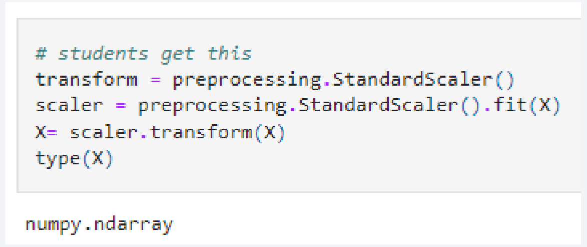
- Using Follium the maps were marked with all the launch sites as well as the successful and failed launches for each site by creating marker clusters.
- Additionally, the proximity to following objects from the CAFS LC-40 were calculated:
  - The coastline – should a critical error occur, putting the launch vehicle over water is a safe option in terms of not endangering infrastructure;
  - Railroad – this mode of transportation is convenient for hauling heavy cargo, in our case, parts required for the spacecraft.
  - Cape Canaveral Space Force Station – proximity to a military installation carries its own benefits and risks, e.g., should any test fights endanger military infrastructure, DoD could potentially shut everything down.
- The notebook is available here: Lab6

Build a Dashboard with Plotly Dash

- An interactive dashboard containing a pie chart, a range slider and scatter plot was created to allow for easy and quick data filtering:
  - Using the drop-down list one can see total launches for all sites (default) or for a specific site;
  - The selection of a particular launch site displays a pie chart of successful and failed launches for the site.
  - Range slider allows one to filter the payload, which results in the scatterplot displaying the success rate (class) per the selected payload mass range and the corresponding booster version used to carry said payload.
- Python file containing the dashboard code can be accessed here: Dash_App

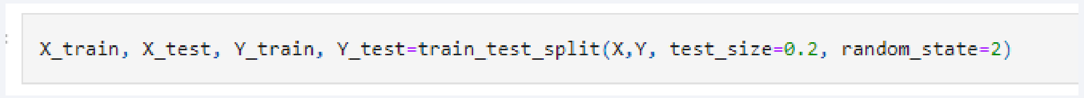
Predictive Analysis (Classification)

- Data was loaded using NumPy and Pandas.
- Data was transformed using the preprocessing function of StandardScaler.
- Then the data set was split into training and testing sets using the train_test_split() function;
- Logistic Regression, Support Vector Machine, Decision Tree and K Closest neighbor models were used (with tuned hyperparameters) and evaluated using the .score() method.
- All models yielded similar accuracy scores (however the Decision Tree accuracy score tends to change if the code is run repeatedly).
- The notebook is available here: Lab7

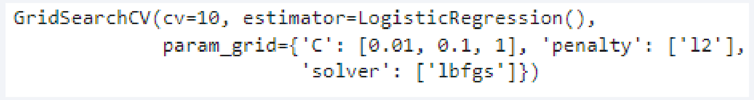


```
# students get this
transform = preprocessing.StandardScaler()
scaler = preprocessing.StandardScaler().fit(X)
X= scaler.transform(X)
type(X)
```

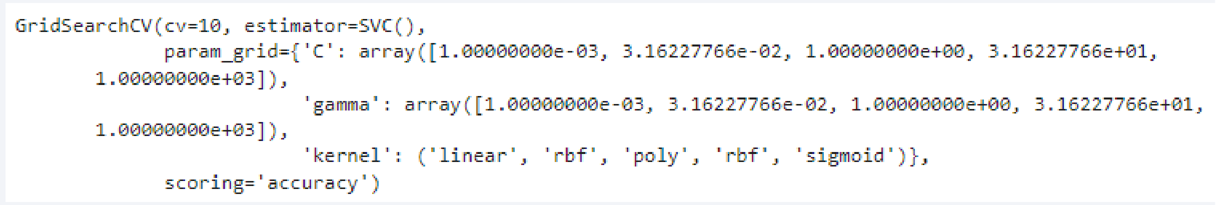
```
numpy.ndarray
```



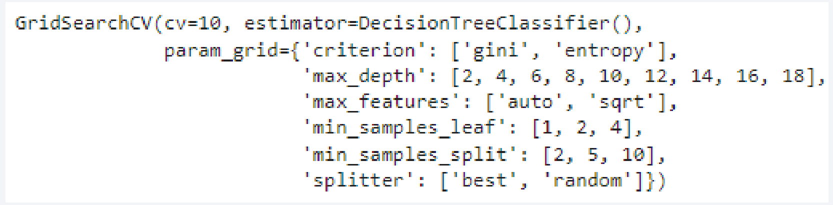
```
X_train, X_test, Y_train, Y_test=train_test_split(X,Y, test_size=0.2, random_state=2)
```



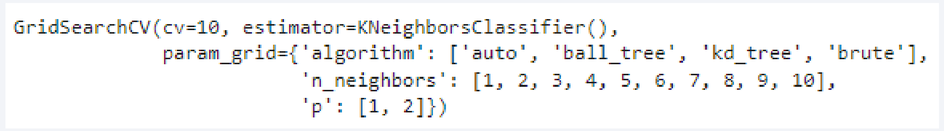
```
GridSearchCV(cv=10, estimator=LogisticRegression(),
             param_grid={'C': [0.01, 0.1, 1], 'penalty': ['l2'],
                         'solver': ['lbfgs']})
```



```
GridSearchCV(cv=10, estimator=SVC(),
             param_grid={'C': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.16227766e+01,
       1.00000000e+03]),
                         'gamma': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.16227766e+01,
       1.00000000e+03]),
                         'kernel': ('linear', 'rbf', 'poly', 'rbf', 'sigmoid')},
             scoring='accuracy')
```



```
GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18],
                         'max_features': ['auto', 'sqrt'],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10],
                         'splitter': ['best', 'random']})
```



```
GridSearchCV(cv=10, estimator=KNeighborsClassifier(),
             param_grid={'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
                         'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                         'p': [1, 2]})
```

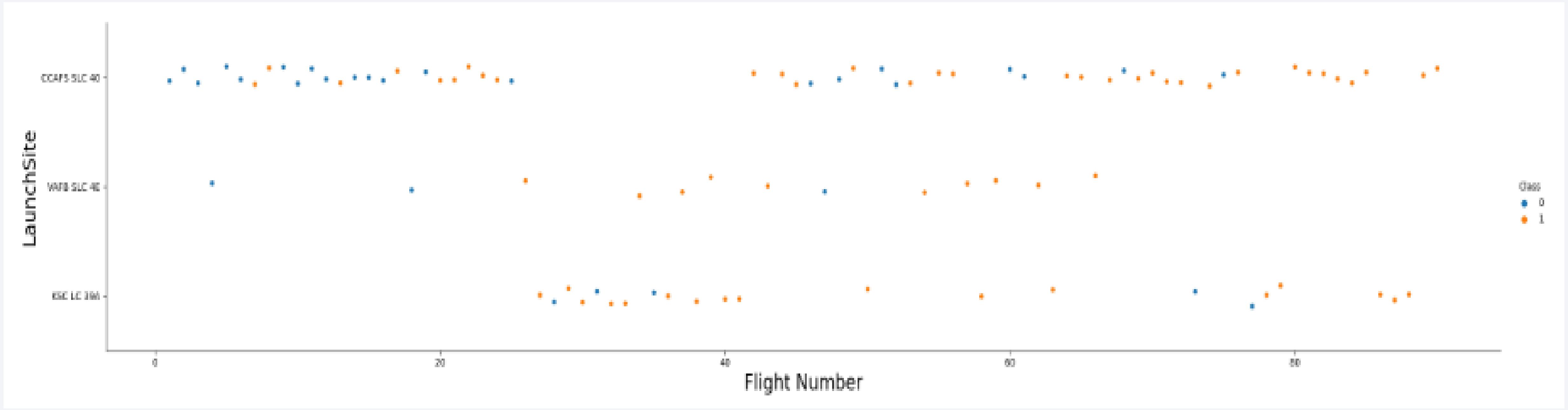
Results

- Exploratory data analysis results:
    - The success rate climbed steadily starting 2013, but experienced slight setbacks in 2018.
- Interactive analytics demo in screenshots (see below)
- Predictive analysis results:
    - The decision tree model yielded various accuracy scores, thus making it an ureliable one in this context. The same code had to be run several times to get the result matching one of the Exam answer option.

Section 2

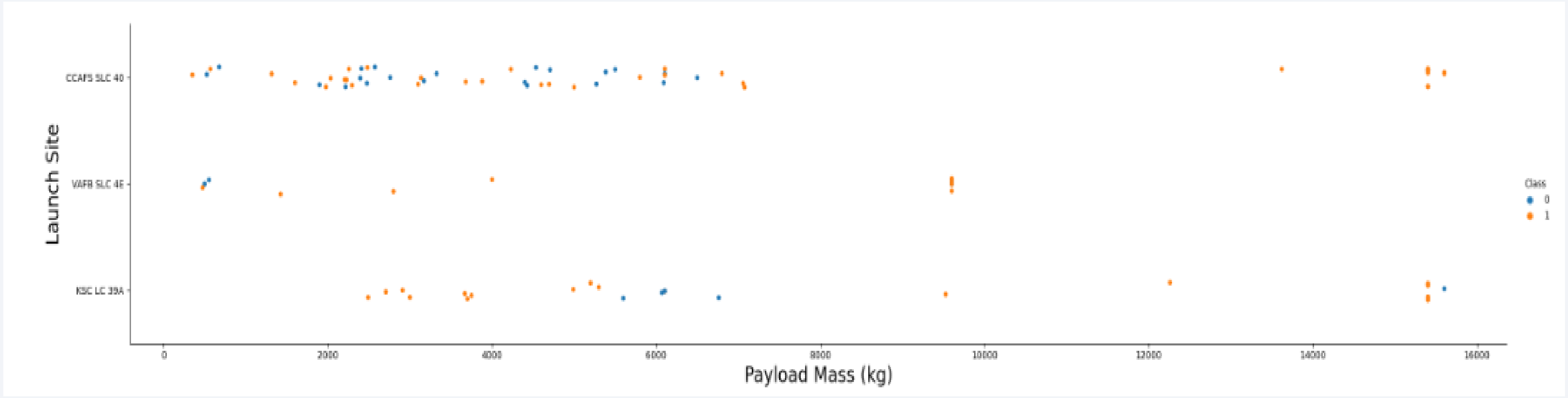
Insights drawn from EDA

Flight Number vs. Launch Site



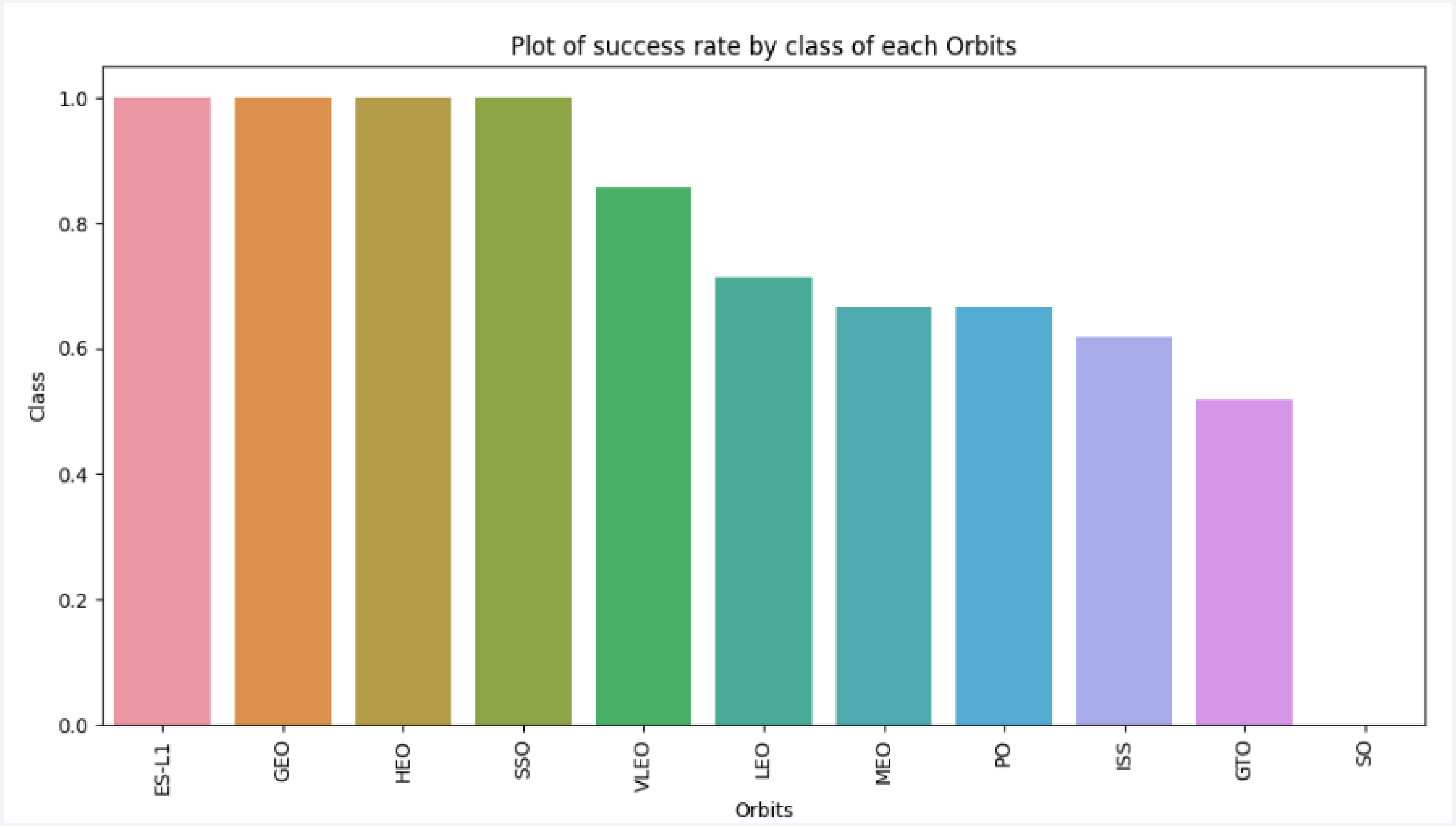
As demonstrated in the graph above, as the flight number increases, so does the success rate, with launch site CCAFS LC-40 being the most utilized and yielding the most success.

Payload vs. Launch Site



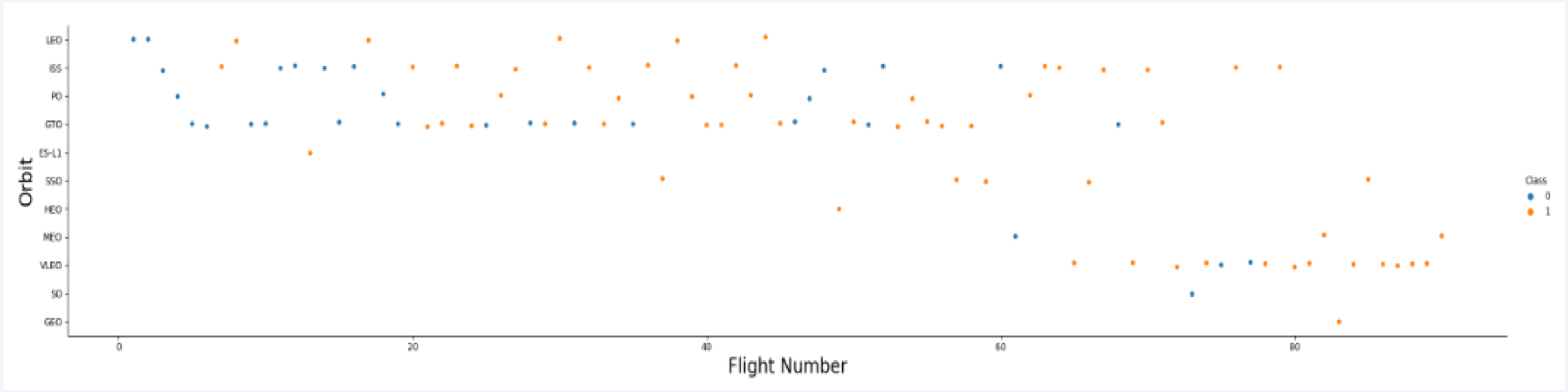
As demonstrated above, the launch site CAFS SLC 40 is very successful with carrying payloads of 10 000 kg and above

Success Rate vs. Orbit Type



- As illustrated by the bar graph, the orbits ES-L1, GEO, HEO, SSO and VLEO have the highest average success rate.

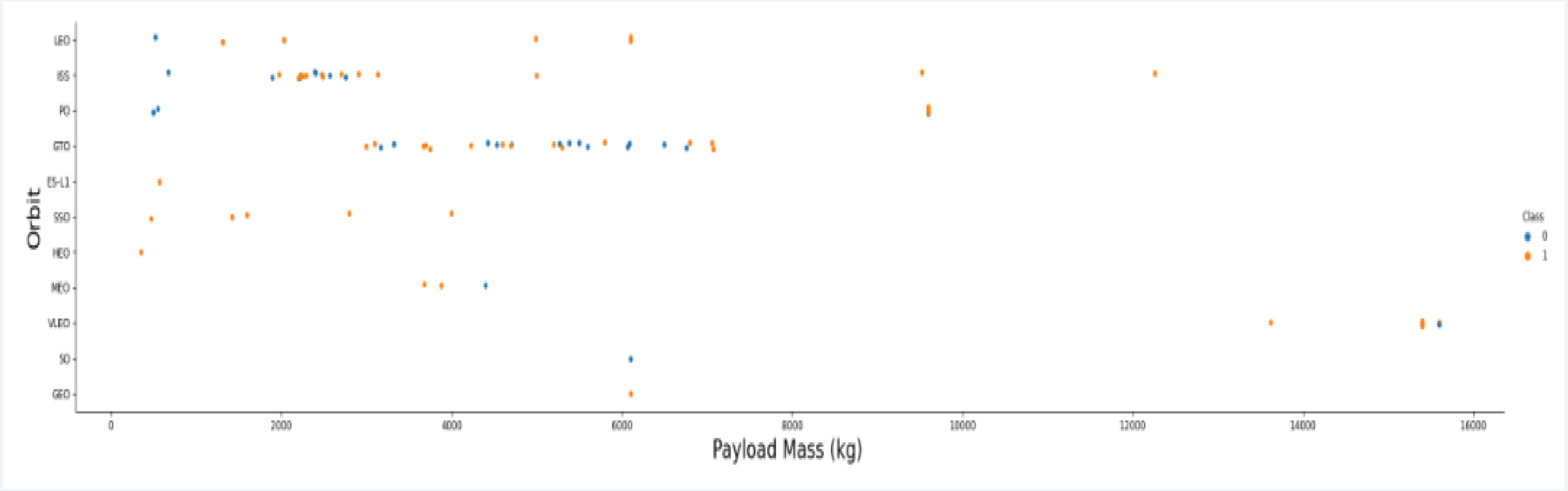
Flight Number vs. Orbit Type



As illustrated by the scatterplot, the Low Earth Orbit (LEO) which is between 160 and 2000 km above Earth, has the highest and most consistent success rate.

The Very Low Earth Orbit (VLEO) which is an altitude of below 400 km also has a high success rate.

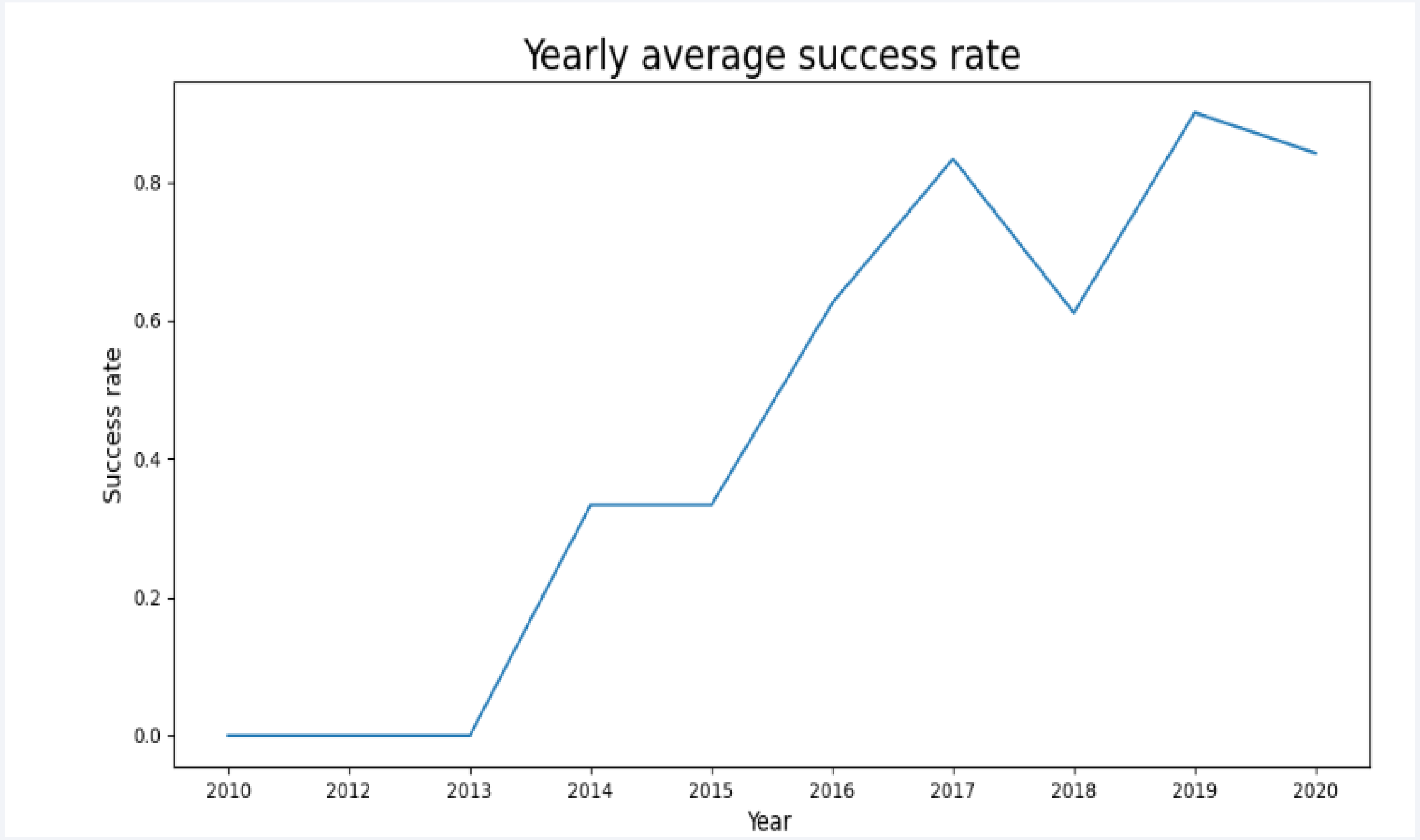
Payload vs. Orbit Type



- For flights carrying heavy payloads, positive landing rate are more often for Polar, LEO and ISS orbits.

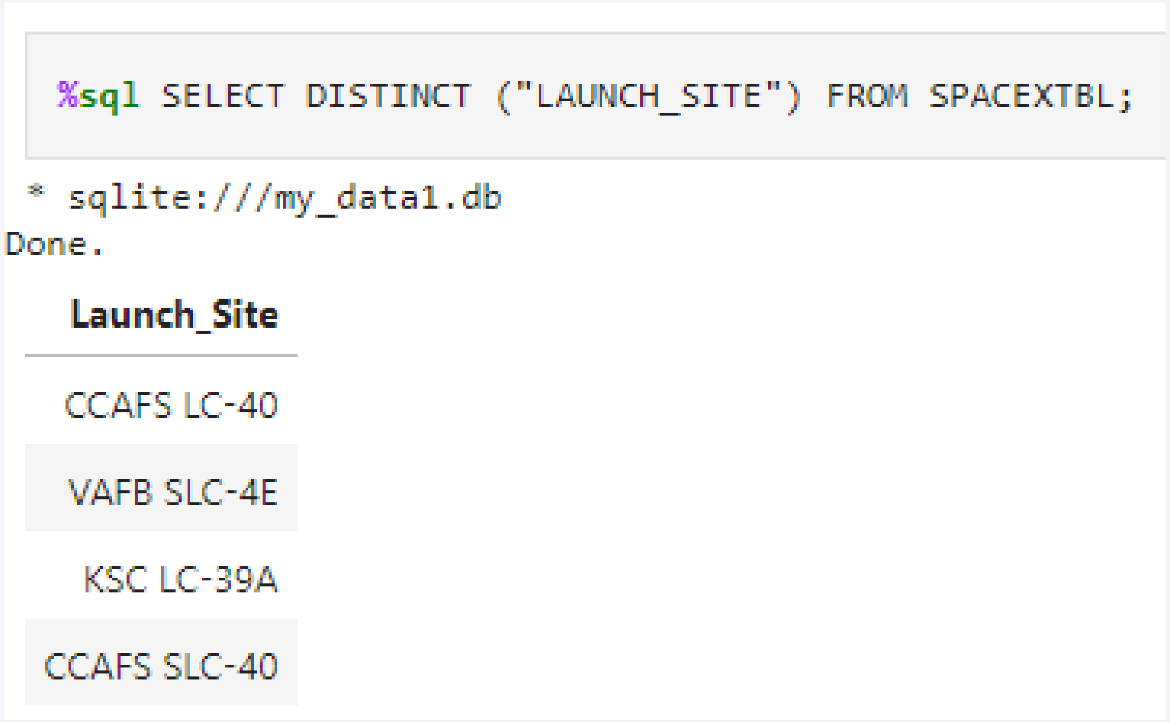
Launch Success Yearly Trend

- As illustrated by the graph on the right, SpaceX Yearly average success rate climbed steadily until 2018, and then again experienced an increase until 2020.



All Launch Site Names

- To find all unique launch sites, an SQL statement using the DISTICT clause was applied to the data set yielding a list of 4 unique launch sites.



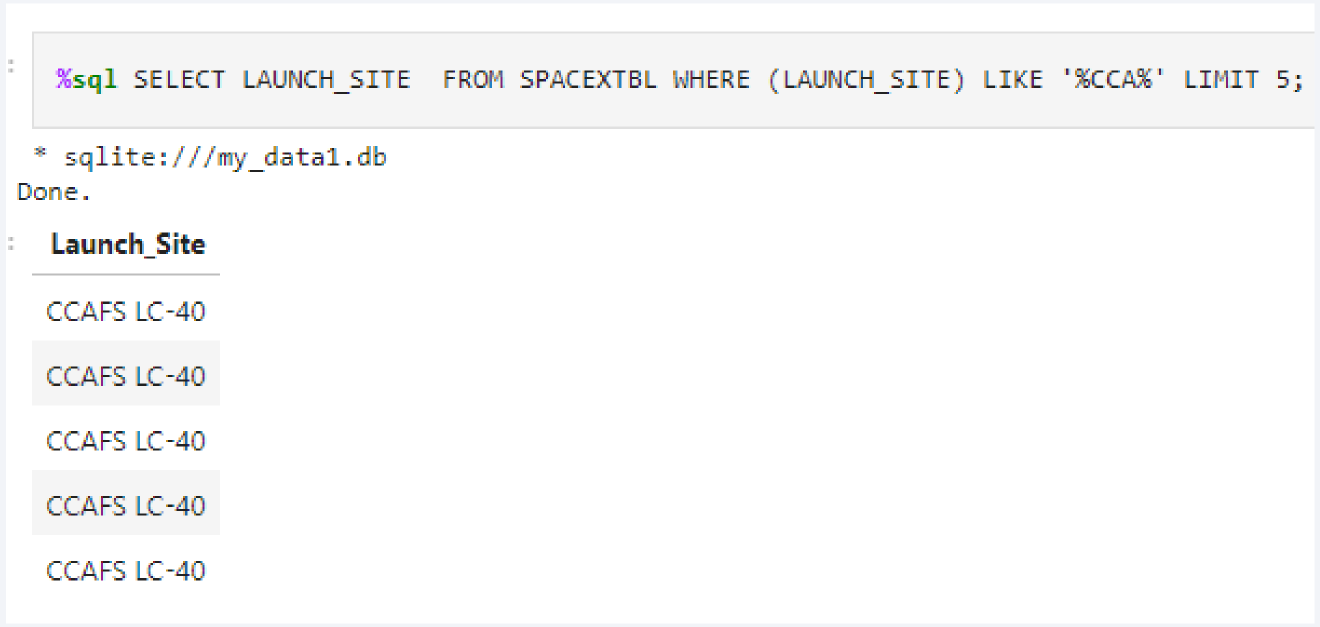
```
%sql SELECT DISTINCT ("LAUNCH_SITE") FROM SPACEXTBL;
```

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

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

Launch Site Names Begin with 'CCA'

- Finding 5 records where launch sites begin with `CCA` was achieved by utilizing the LIKE clause in combination with the wildcard symbol % and setting the LIMIT to 5 records.



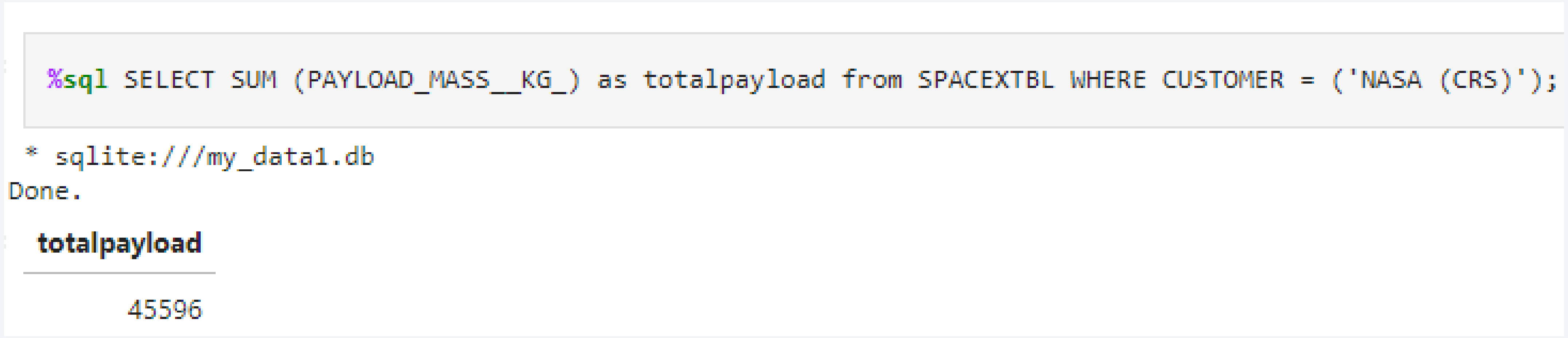
```
%sql SELECT LAUNCH_SITE  FROM SPACEXTBL WHERE (LAUNCH_SITE) LIKE '%CCA%' LIMIT 5;
```

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

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Total Payload Mass

- Calculating the total payload carried by boosters from NASA was achieved by utilizing the SUM function and filtering records WHERE the customer was NASA (CRS).

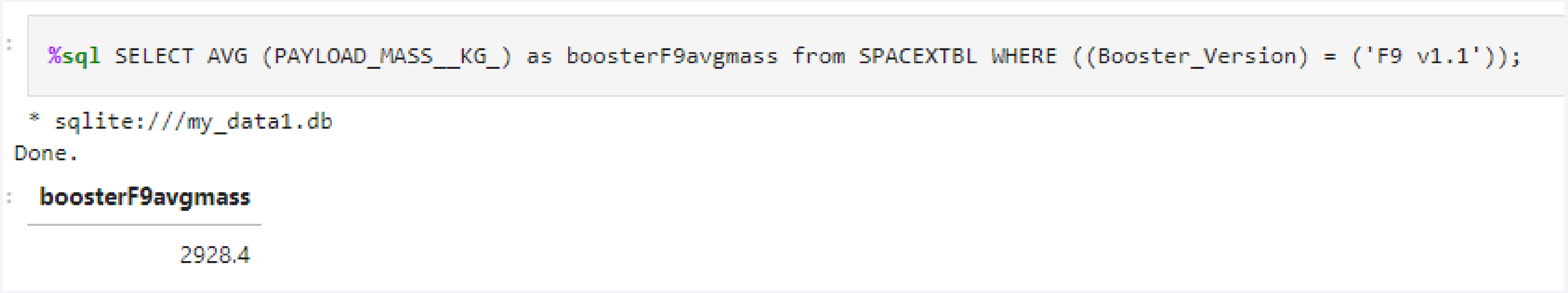


```
%sql SELECT SUM (PAYLOAD_MASS__KG_) as totalpayload from SPACEXTBL WHERE CUSTOMER = ('NASA (CRS)');
```

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

totalpayload
45596

Average Payload Mass by F9 v1.1



```
%sql SELECT AVG (PAYLOAD_MASS__KG_) as boosterF9avgmass from SPACEXTBL WHERE ((Booster_Version) = ('F9 v1.1'));
```

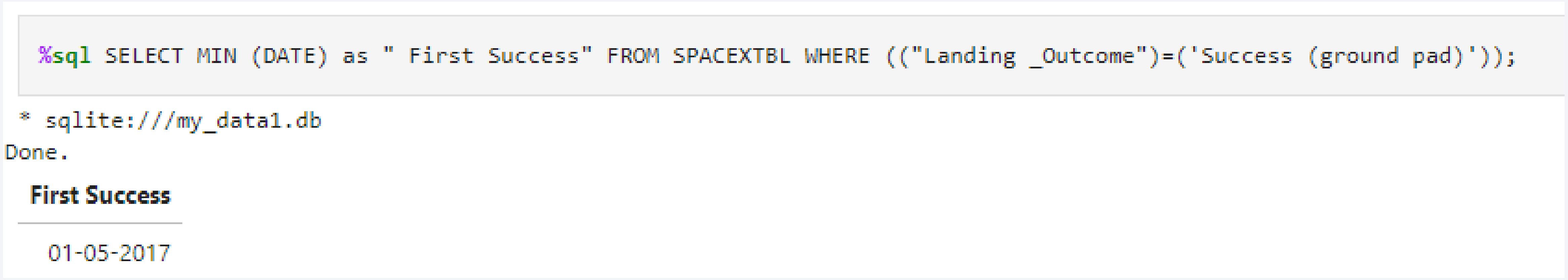
```
 * sqlite:///my_data1.db
Done.
```

boosterF9avgmass
2928.4

- As per the query results, the average mass carried by the boosters F9 v1.1 was 2 928.4 kg or ~ 6 456.01689 pounds.

First Successful Ground Landing Date

- The first successful landing on ground pad was achieved on May 1st, 2017.



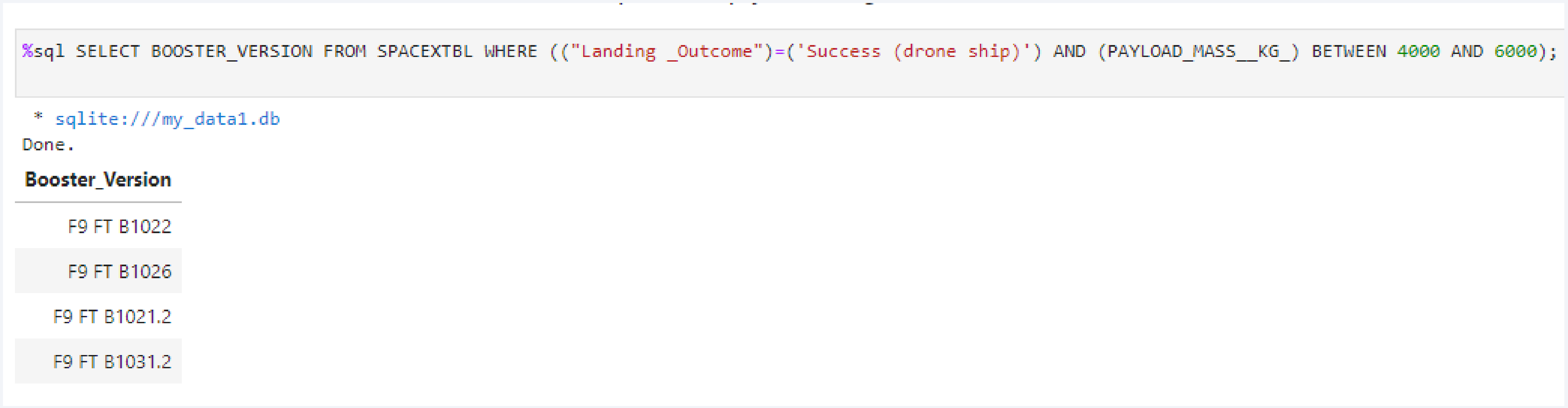
```
%sql SELECT MIN (DATE) as " First Success" FROM SPACEXTBL WHERE (("Landing _Outcome")=('Success (ground pad)'));
```

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

First Success
01-05-2017

Successful Drone Ship Landing with Payload between 4000 and 6000

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



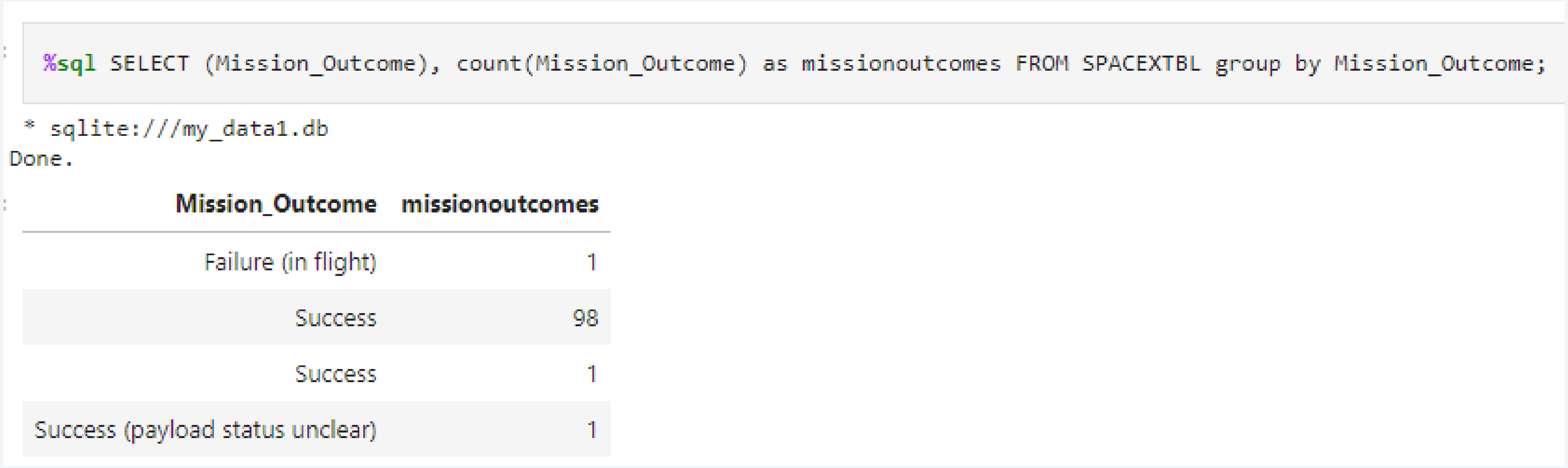
```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE (("Landing _Outcome")=('Success (drone ship)') AND (PAYLOAD_MASS__KG_) BETWEEN 4000 AND 6000);
```

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



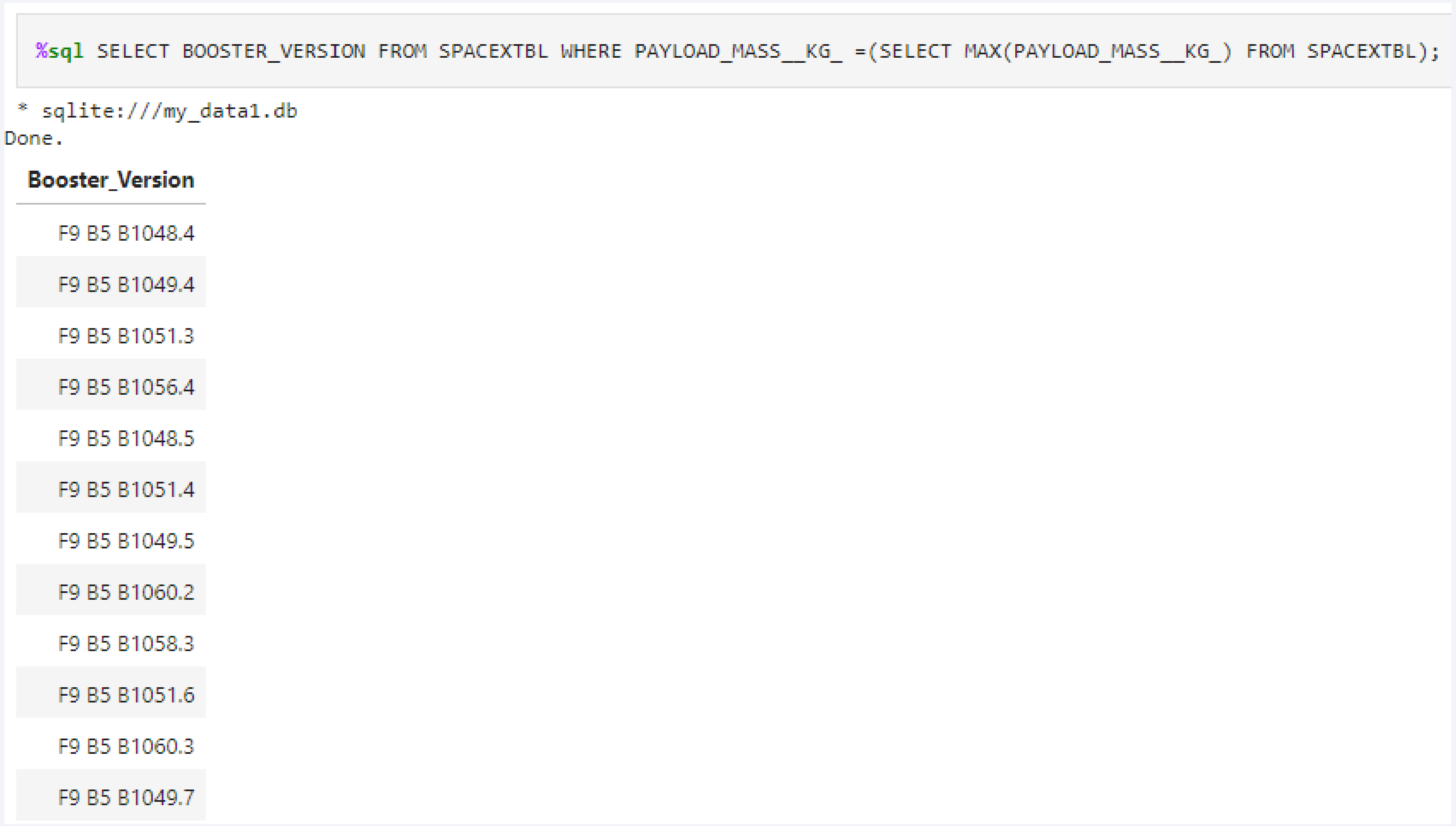
```
%sql SELECT (Mission_Outcome), count(Mission_Outcome) as missionoutcomes FROM SPACEXTBL group by Mission_Outcome;
```

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

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

Boosters Carried Maximum Payload

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



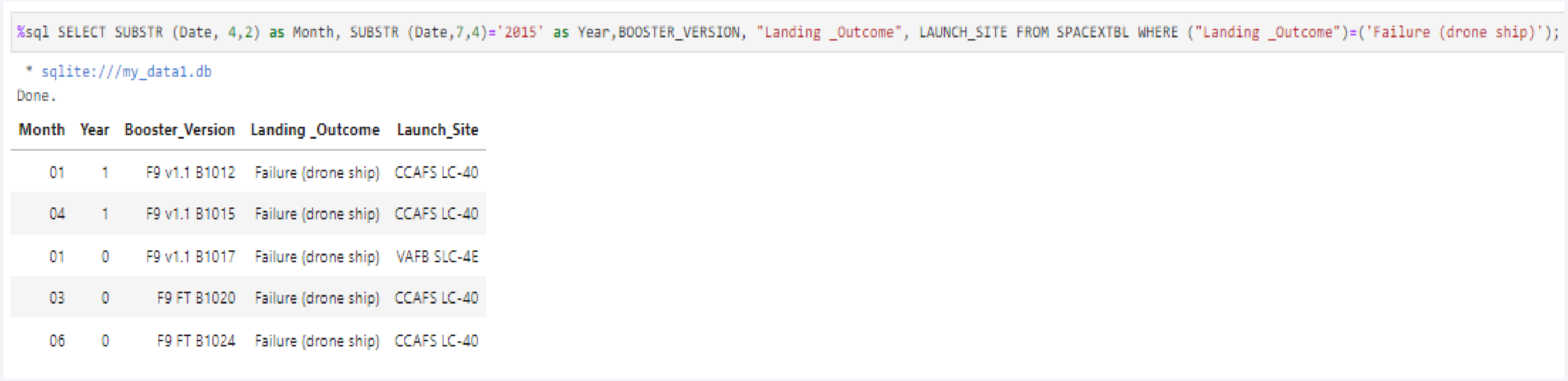
```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ =(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

```
 * 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

2015 Launch Records

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



```
%sql SELECT SUBSTR (Date, 4,2) as Month, SUBSTR (Date,7,4)='2015' as Year,BOOSTER_VERSION, "Landing _Outcome", LAUNCH_SITE FROM SPACEXTBL WHERE ("Landing _Outcome")=('Failure (drone ship)');
```

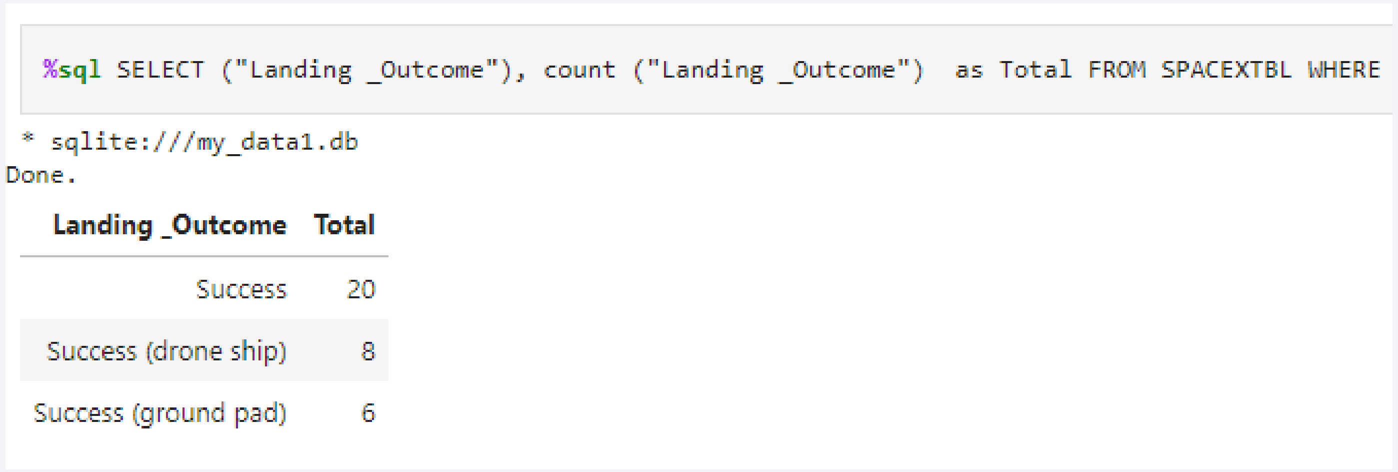
```
 * sqlite:///my_data1.db
Done.
```

Month	Year	Booster_Version	Landing _Outcome	Launch_Site
01	1	F9 v1.1 B1012	Failure (drone ship)	CCAFS LC-40
04	1	F9 v1.1 B1015	Failure (drone ship)	CCAFS LC-40
01	0	F9 v1.1 B1017	Failure (drone ship)	VAFB SLC-4E
03	0	F9 FT B1020	Failure (drone ship)	CCAFS LC-40
06	0	F9 FT B1024	Failure (drone ship)	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.
- Full query:

%sql SELECT ("Landing _Outcome"), count ("Landing _Outcome") as Total FROM SPACEXTBL WHERE (("Landing _Outcome") LIKE ('Success%')) AND DATE BETWEEN ('04-06-2010') AND ('20-03-2017') group by ("Landing _Outcome") order by (“Total") DESC;



```
%sql SELECT ("Landing _Outcome"), count ("Landing _Outcome")  as Total FROM SPACEXTBL WHERE
```

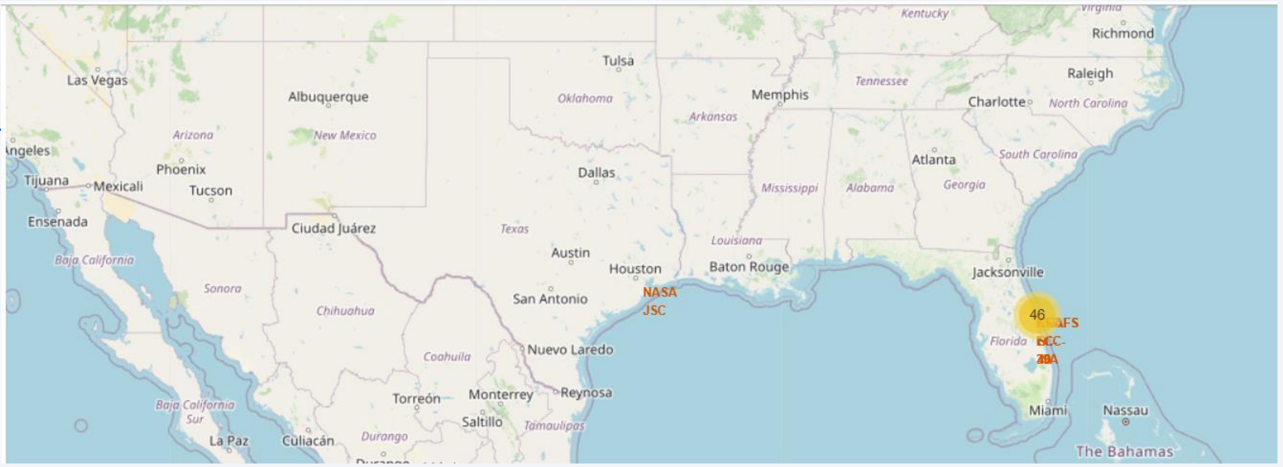
```
 * sqlite:///my_data1.db
Done.
```

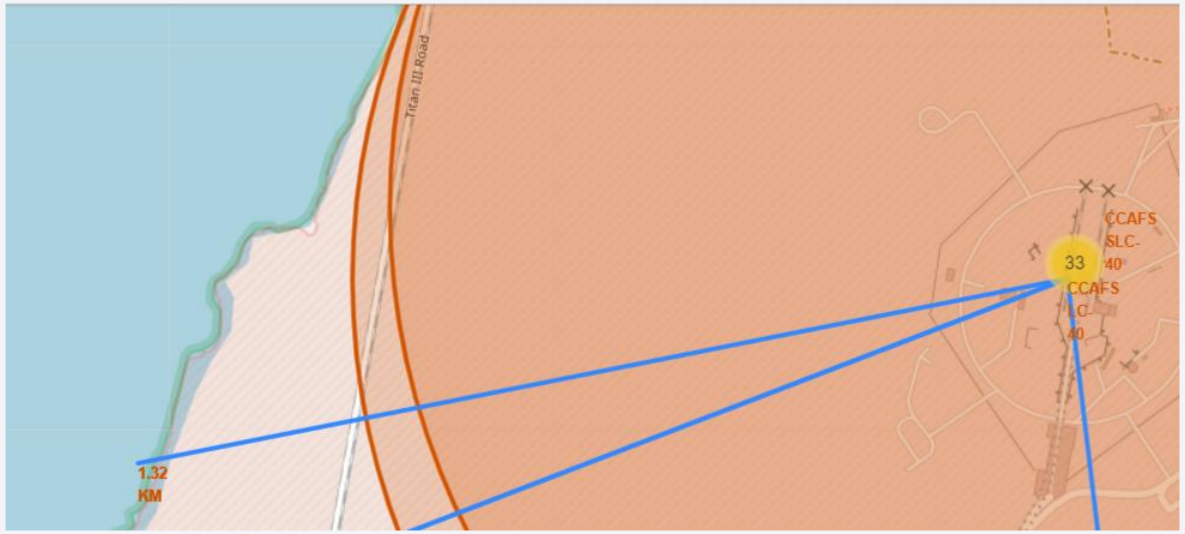
Landing _Outcome	Total
Success	20
Success (drone ship)	8
Success (ground pad)	6

Section 3

Launch Sites Proximities Analysis

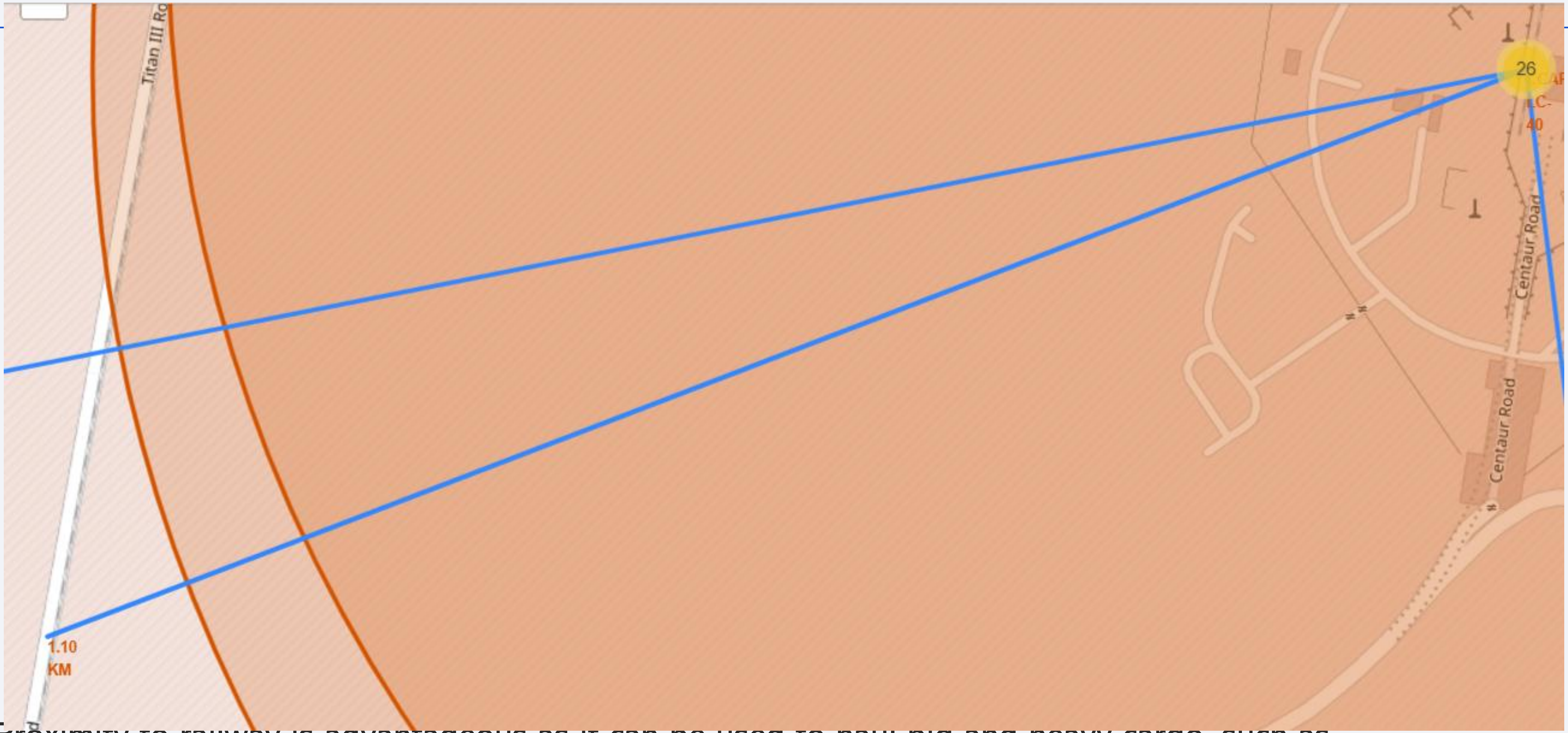
Proximity to the coastline





- Proximity to water is an important factor as it can be used as a crash site should a catastrophic error occur.

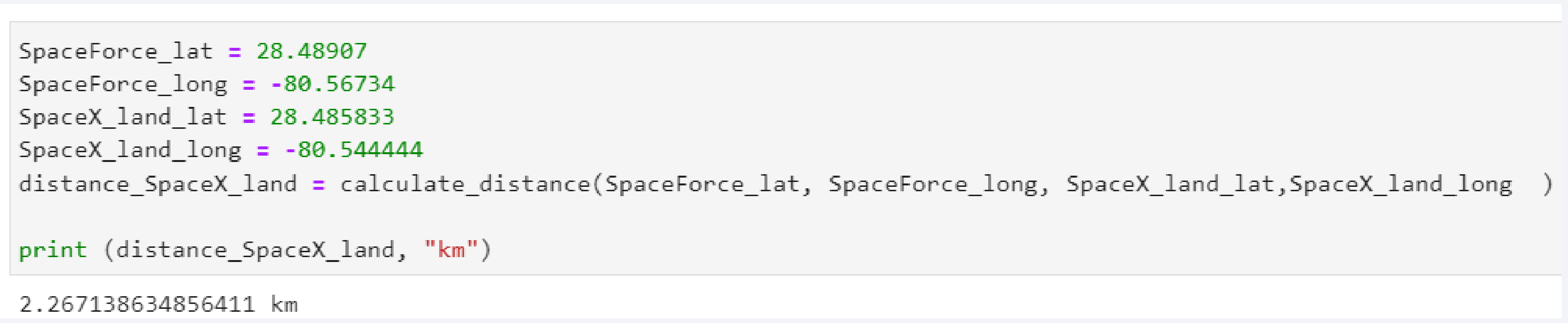
Proximity to the railway



- Proximity to railway is advantageous as it can be used to haul big and heavy cargo, such as equipment.

Proximity to military installations

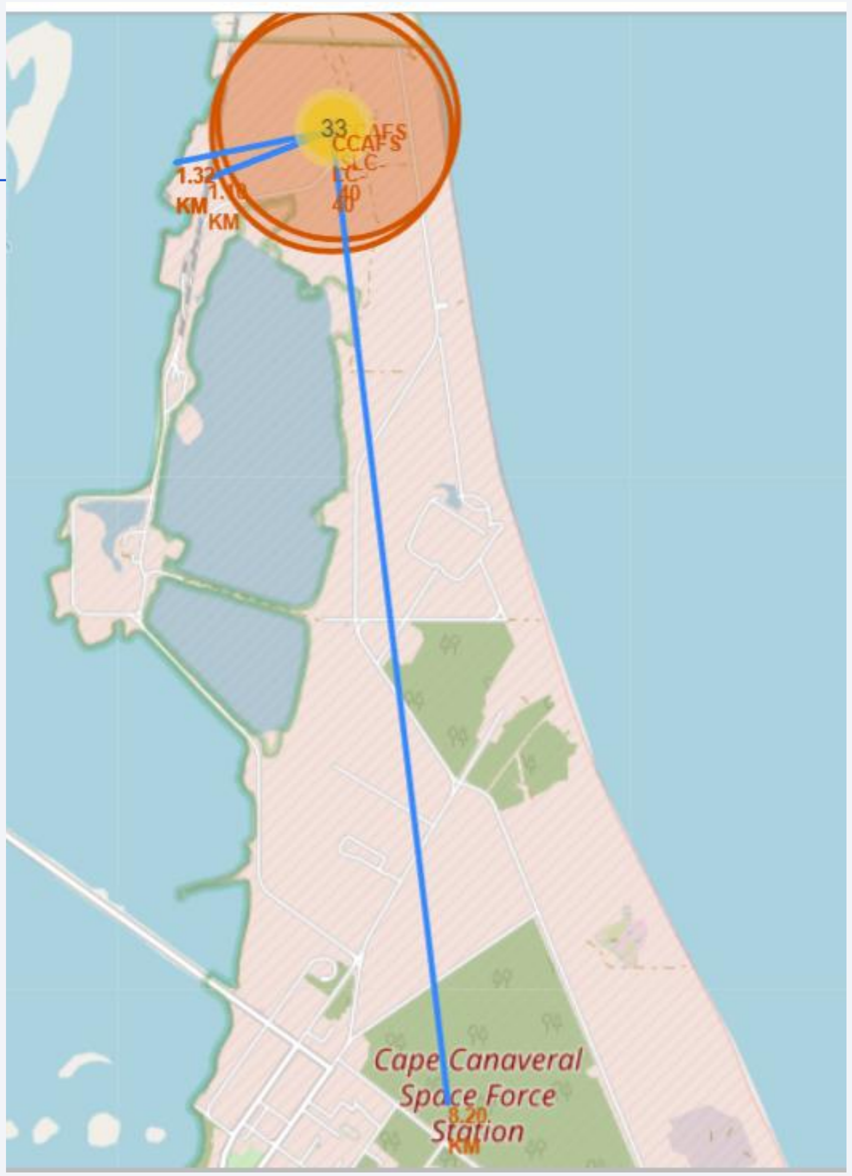
- Proximity to military infrastructure: Cape Canaveral Space Force station houses the launch sites CCAFS LC-40 and CCAFS SLC-40.
- It is also near SpaceX landing sites 1 & 2.

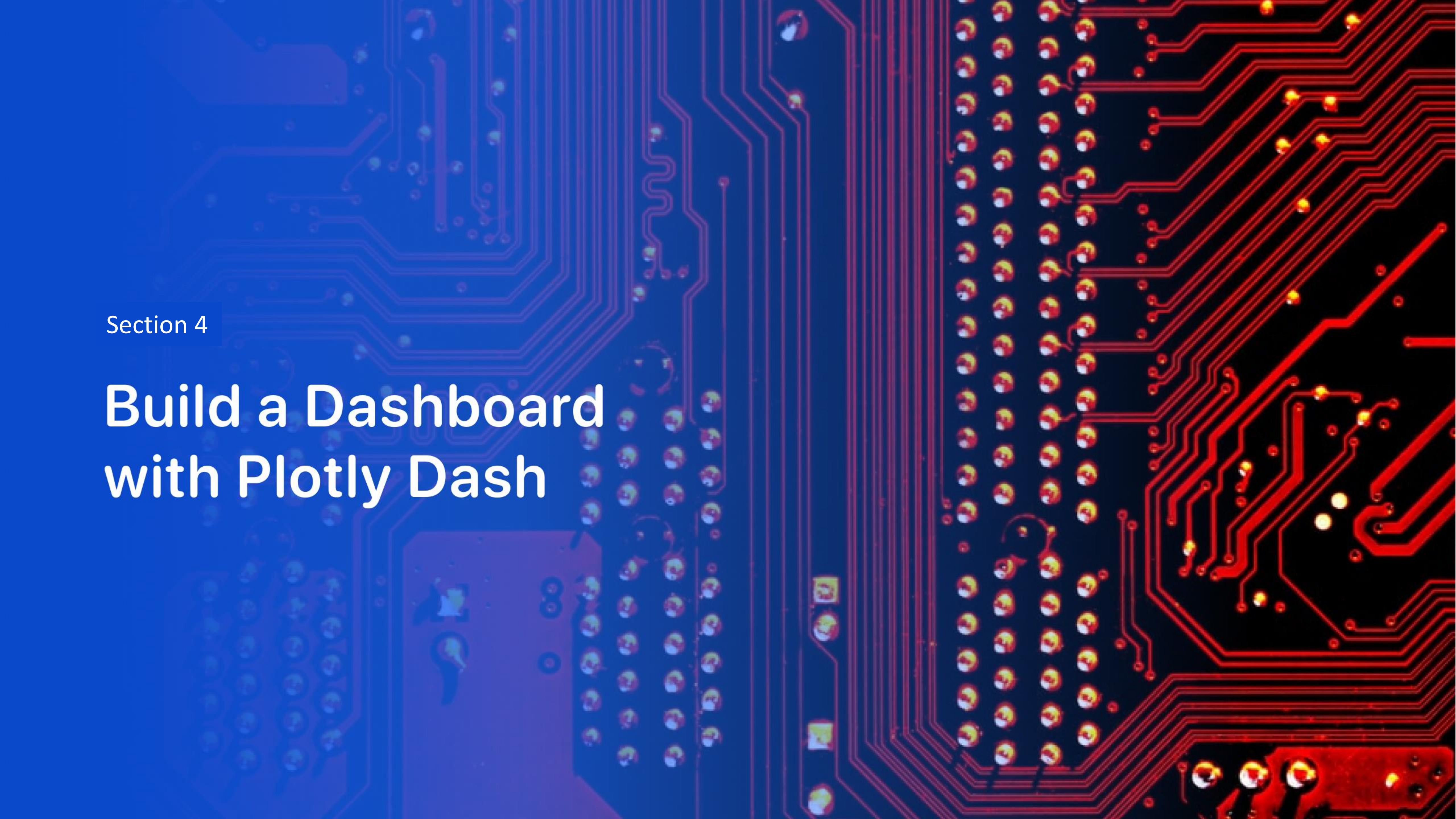


```
SpaceForce_lat = 28.48907
SpaceForce_long = -80.56734
SpaceX_land_lat = 28.485833
SpaceX_land_long = -80.544444
distance_SpaceX_land = calculate_distance(SpaceForce_lat, SpaceForce_long, SpaceX_land_lat,SpaceX_land_long  )

print (distance_SpaceX_land, "km")
```

```
2.267138634856411 km
```

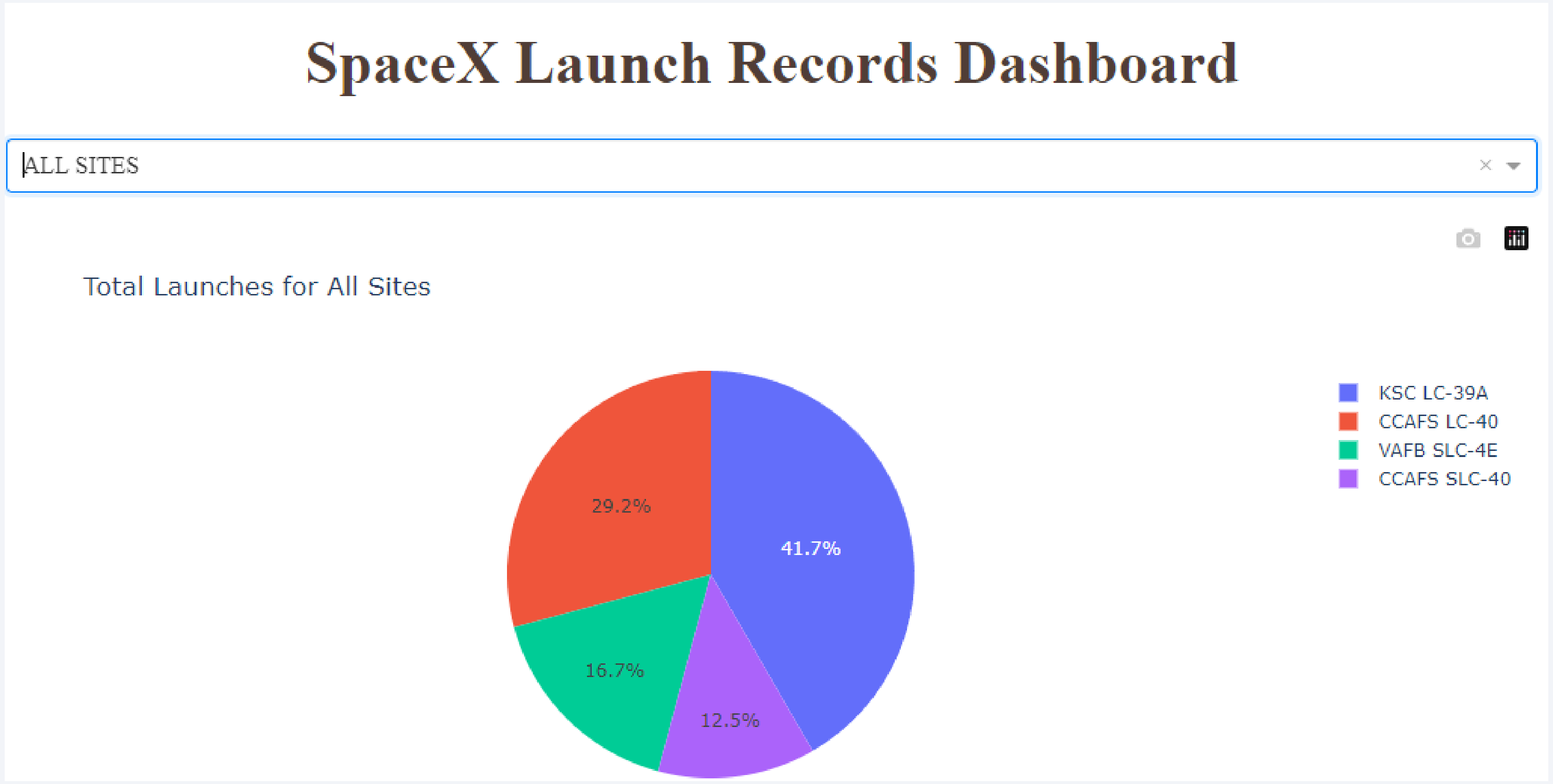




Section 4

Build a Dashboard with Plotly Dash

Launch data : All sites

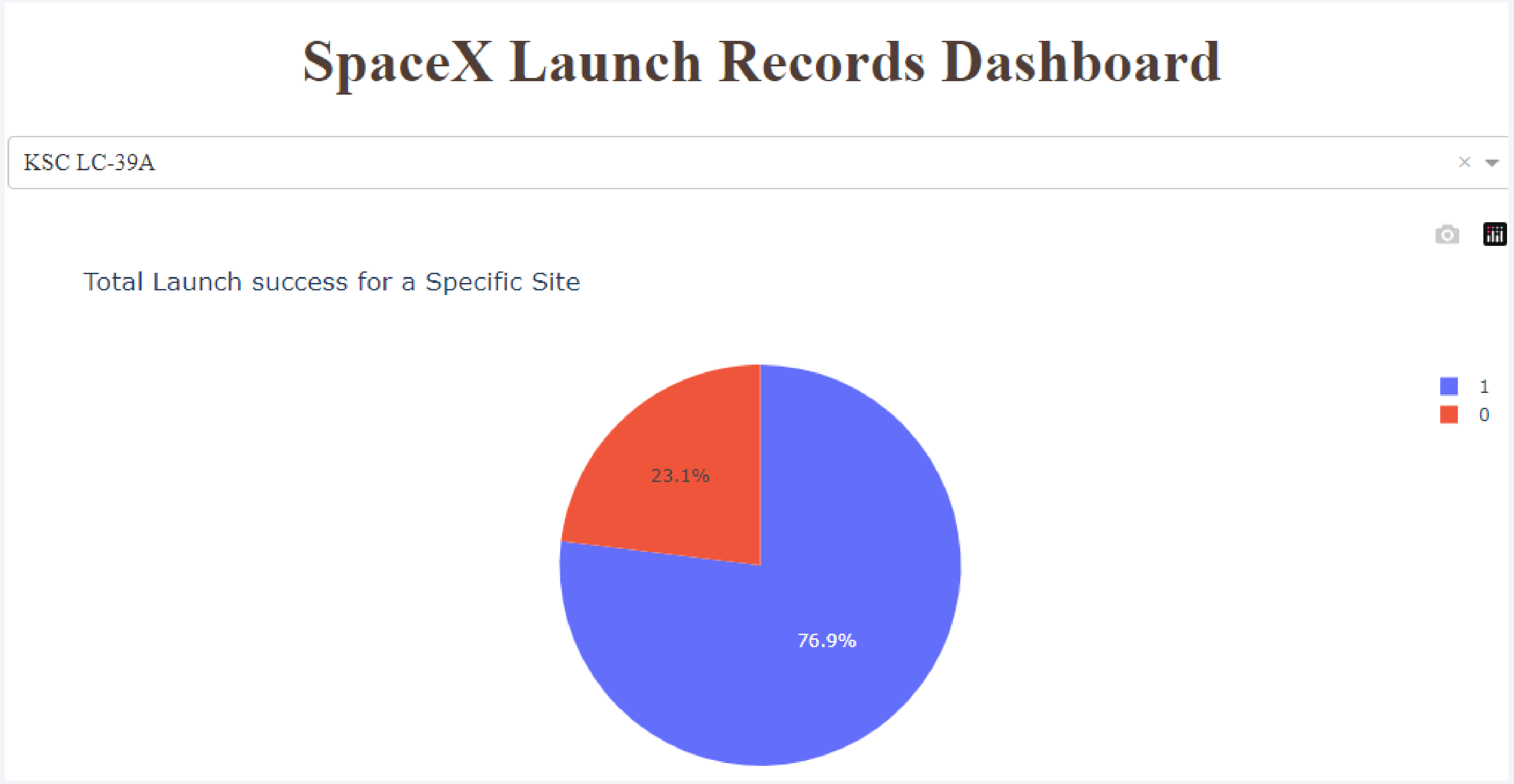


SpaceX Launch Records Dashboard

ALL SITES

- As illustrated above, the launch site KSC LC-39A has the highest number of launches.

Highest launch success ratio

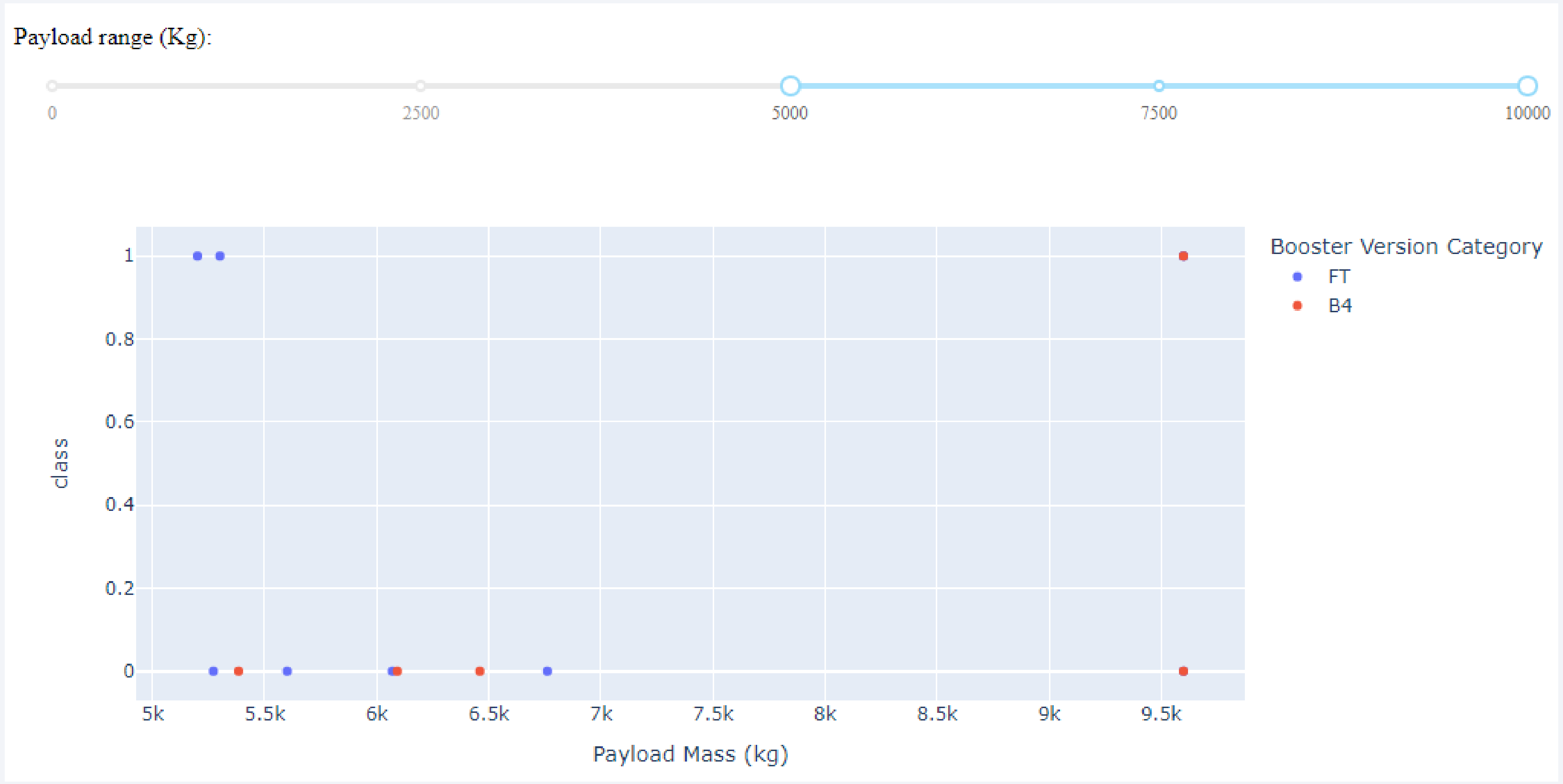

SpaceX Launch Records Dashboard

KSC LC-39A

Total Launch success for a Specific Site

As illustrated above, launch site KSC LC-39A has the highest success rate.

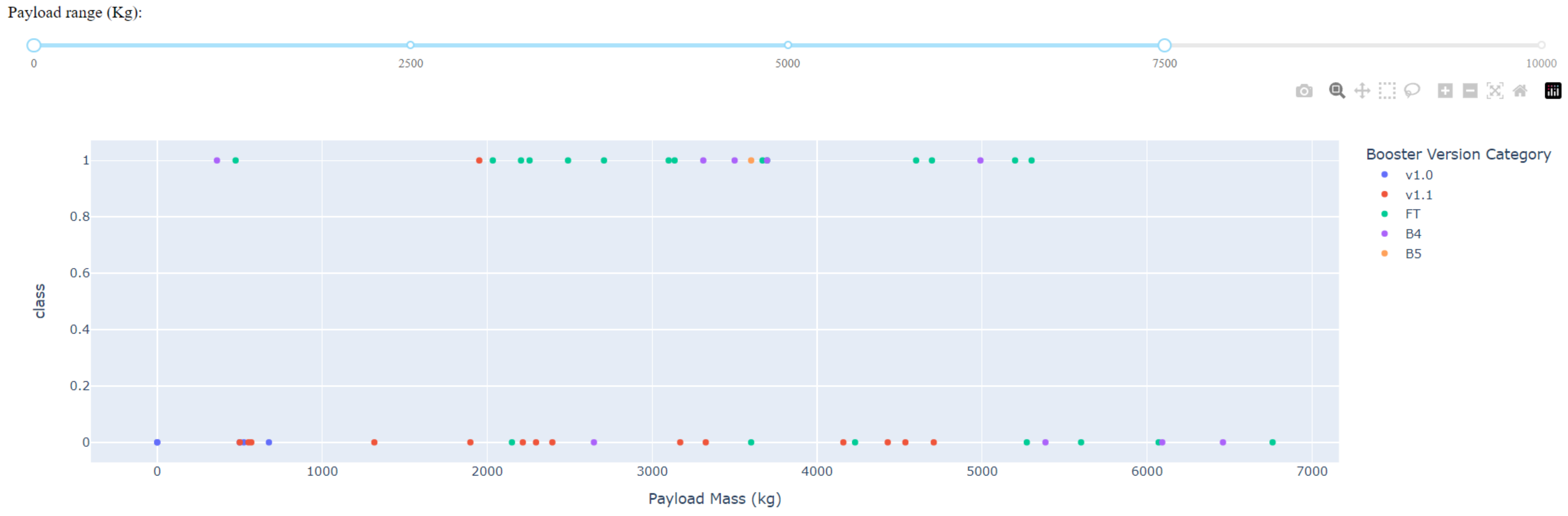
Payload vs. Launch Outcome



As per our data, the two booster models used to carry payloads between 5000 and 10 000 are FT and B4, however the success rate for both boosters is low:

20% for B4 and ~33% for FT.

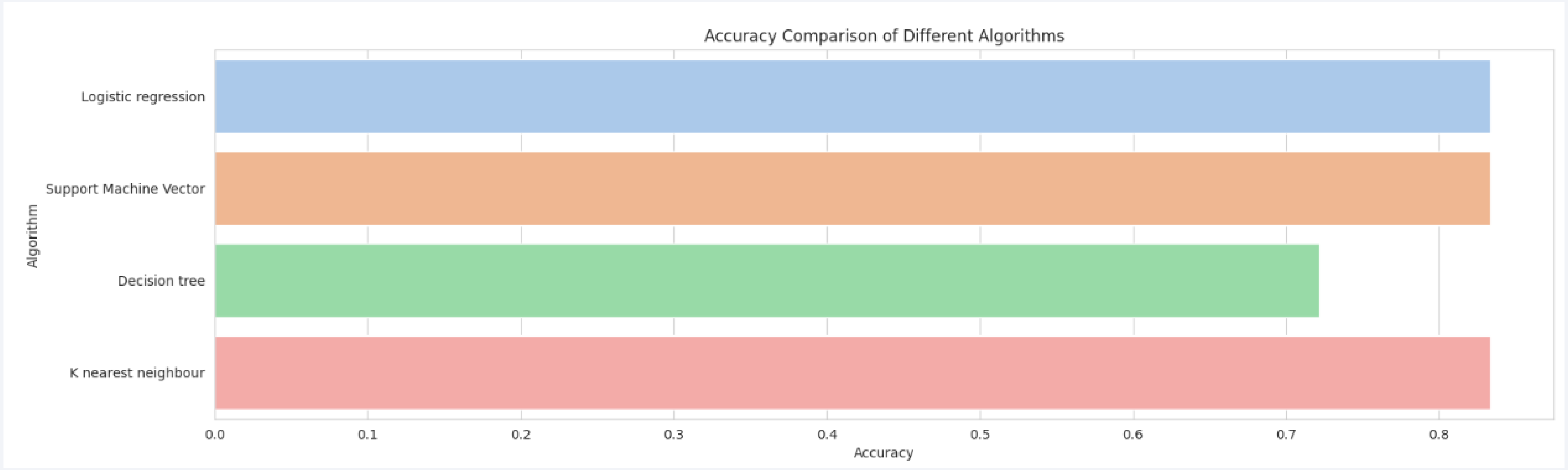
The data also suggests that there is less success with payloads weighing up to 5000 kg.
The most successful Booster in this weight range is the FT.



Section 5

Predictive Analysis (Classification)

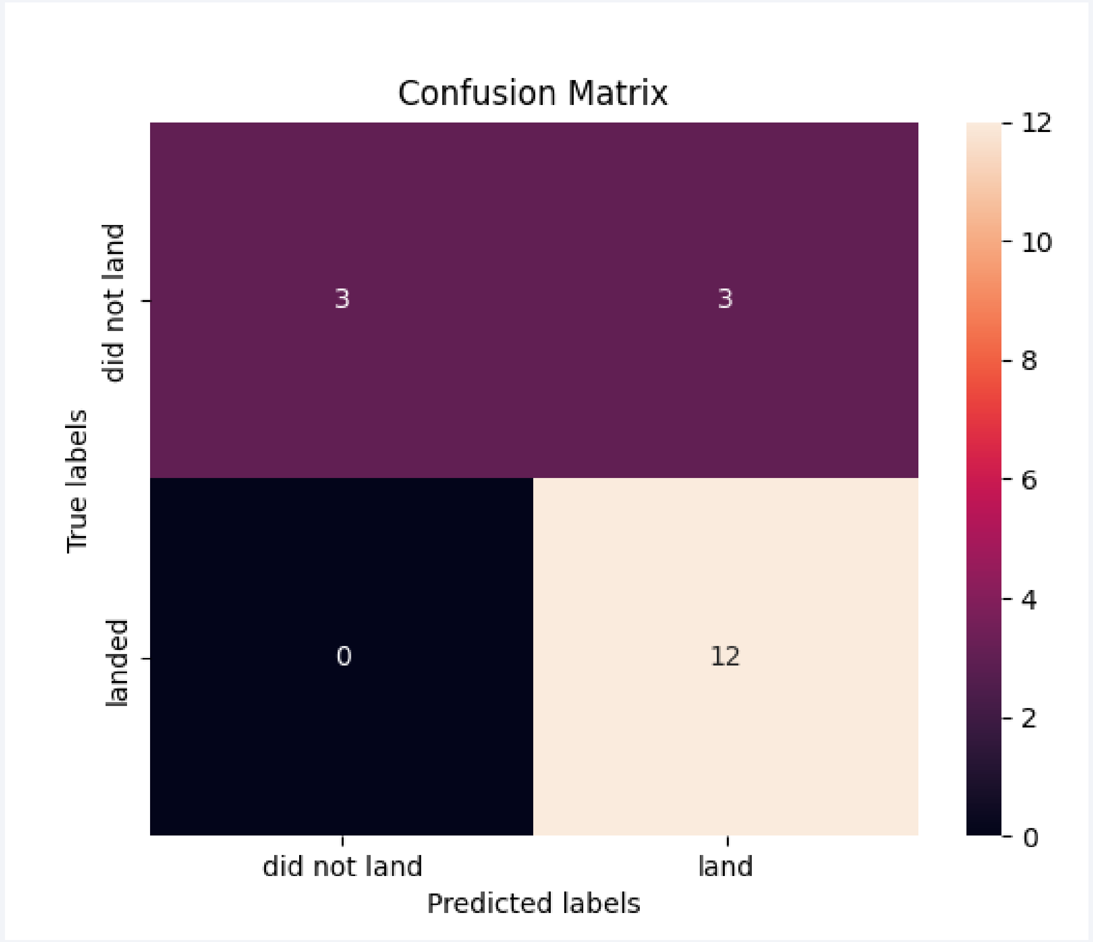
Classification Accuracy



- As illustrated above, there is not much difference between accuracy scores.

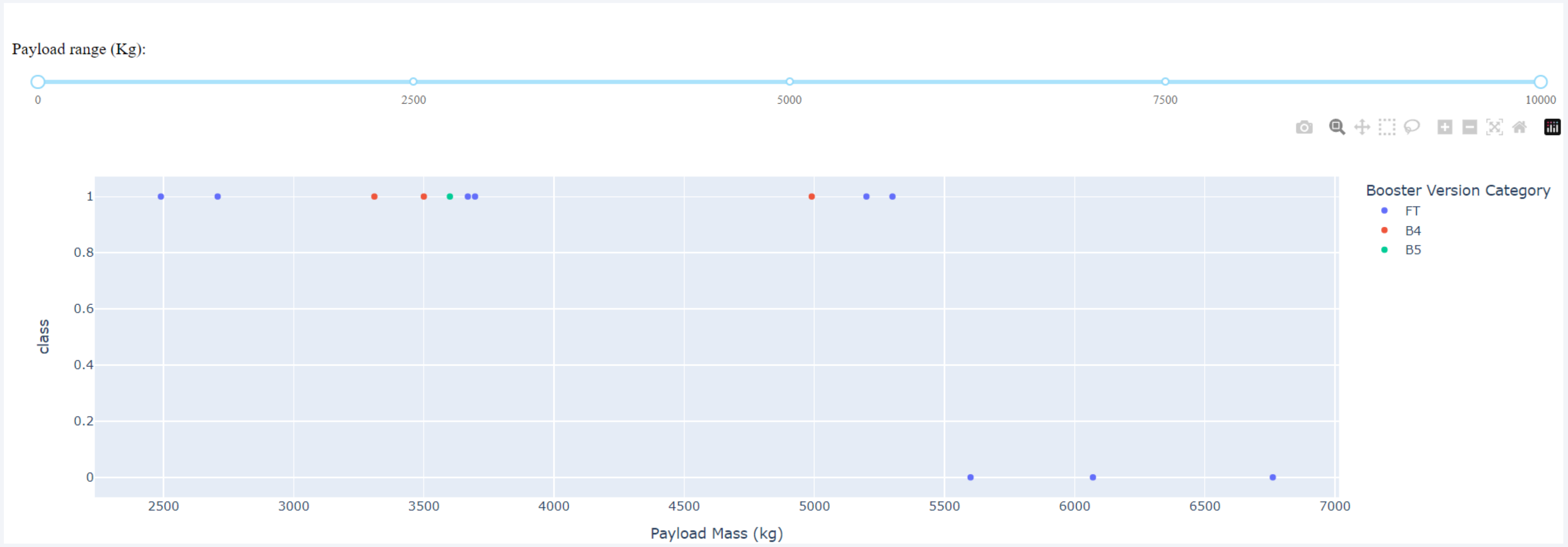
Confusion Matrix

- As the accuracy scores were nearly identical, so are the confusion matrices for Logistic regression, SVM and K Nearest neighbor – all having this confusion matrix:



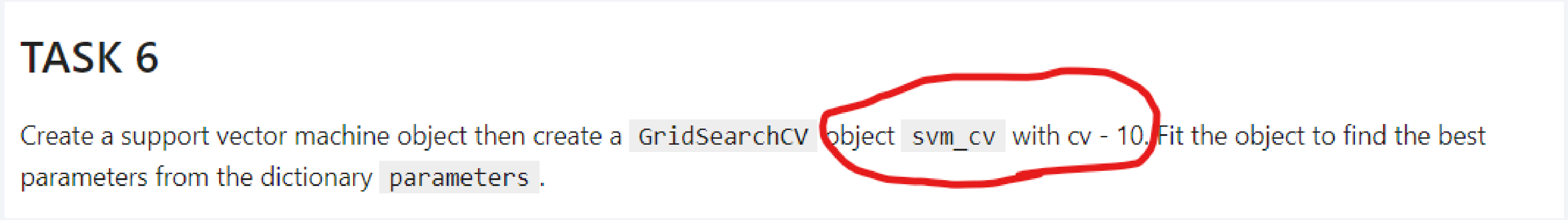
Conclusions

- The most used launch site is also the one which has the highest success rate: **KSC LC-39A**
- Orbits ES-L1, GEO, HEO, SSO and VLEO have the highest average success rate.
- Most launches (per all sites) carries a payload mass below 6000 kg and have had varying success rates, KSC LC-39A having ~73% success rate with payload up to 5500kg:



Appendix

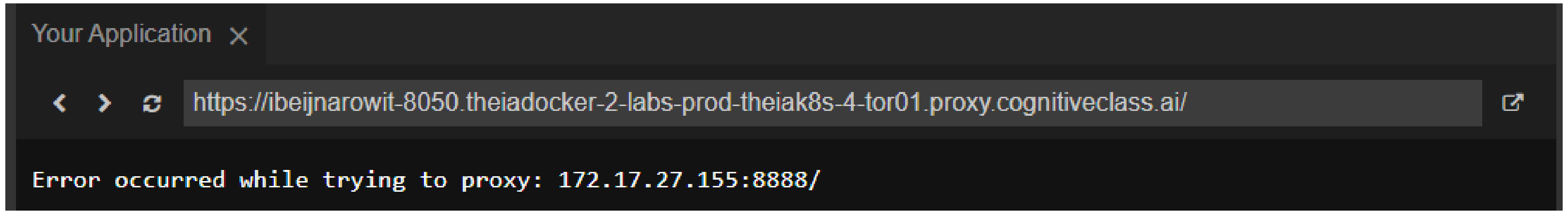
- The decision tree model yielded various accuracy scores, thus making it an unreliable one in this context. The same code had to be run several times to get the result matching one of the Exam answer options.
- Final – Prediction – lab gets stuck by Task 6, yielding no output when run on Chrome. I managed to make it work by running it **via Edge**.
- Despite having acknowledged the issue, the staff have not corrected the Prediction lab notebook and Task 6 still displays a cv value of -10 – this should be cv=10.



TASK 6

Create a support vector machine object then create a `GridSearchCV` object `svm_cv` with cv - 10. Fit the object to find the best parameters from the dictionary `parameters`.

- Plotly Dash often runs into errors:

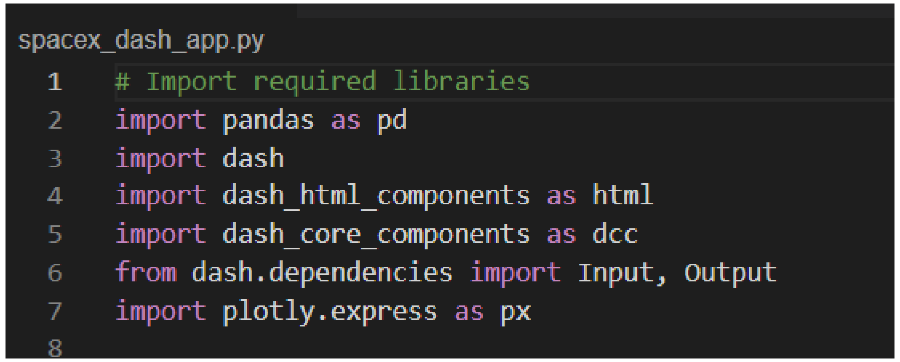


Your Application

https://ibeijnarowit-8050.theiadocker-2-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/

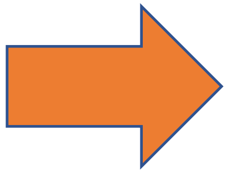
```
Error occurred while trying to proxy: 172.17.27.155:8888/
```

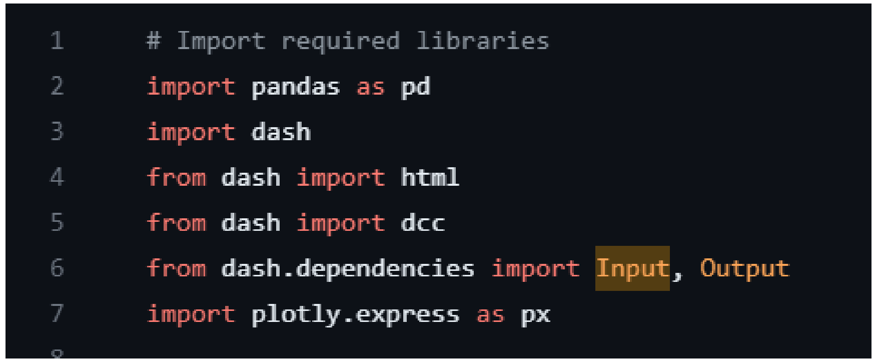
- Plotly Dash template needs to be updated as the libraries in the default template have expired:



spacex_dash_app.py

```python
# Import required libraries
import pandas as pd
import dash
import dash_html_components as html
import dash_core_components as dcc
from dash.dependencies import Input, Output
import plotly.express as px
```





```python
# Import required libraries
import pandas as pd
import dash
from dash import html
from dash import dcc
from dash.dependencies import Input, Output
import plotly.express as px
```

- By not correcting these errors the Course creators and staff are only creating themselves extra work.
- This also leads to frustration in students which is not conducive to learning and motivation.

Thank you!