IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

John Ekwere
December, 2023

Outline

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

Executive Summary

- This project is focused on analyzing spaceX Falcon 9 dataset and building a predictive model to determine the success or failure of the first stage landing. This will help other companies decide if they will place a bid against spaceX.

- Summary of methodologies

  The method used in this project involves:

  - Gathering Data through web scrapping and use of API
  - Data transformation and wrangling
  - Exploring the dataset using SQL and visuals
  - Build an interactive dashboard using folium and plotly dash so that users can interact with the data
  - Build a predictive model to determine the success or failure of the first stage landing.

- Summary of all results

  - The best model that was trained based on result is decision tree model with an accuracy score of over 80%

Introduction

- Project background and context
  - With increasing interest in space exploration, a lot more people are interested in exploring business opportunities in the sector despite its high cost. The high cost of space ventures has deterred some investors from venturing into the business. SpaceX has a competitive advantages which reduces their total cost by over 60% if they are able to reuse the first part of the space craft.
- Problems you want to find answers
  - The main problem is to determine if the first stage will land successfully.
  - Does the launch site impact on the success rate?
  - What parameters affect the landing success

Section 1

Methodology

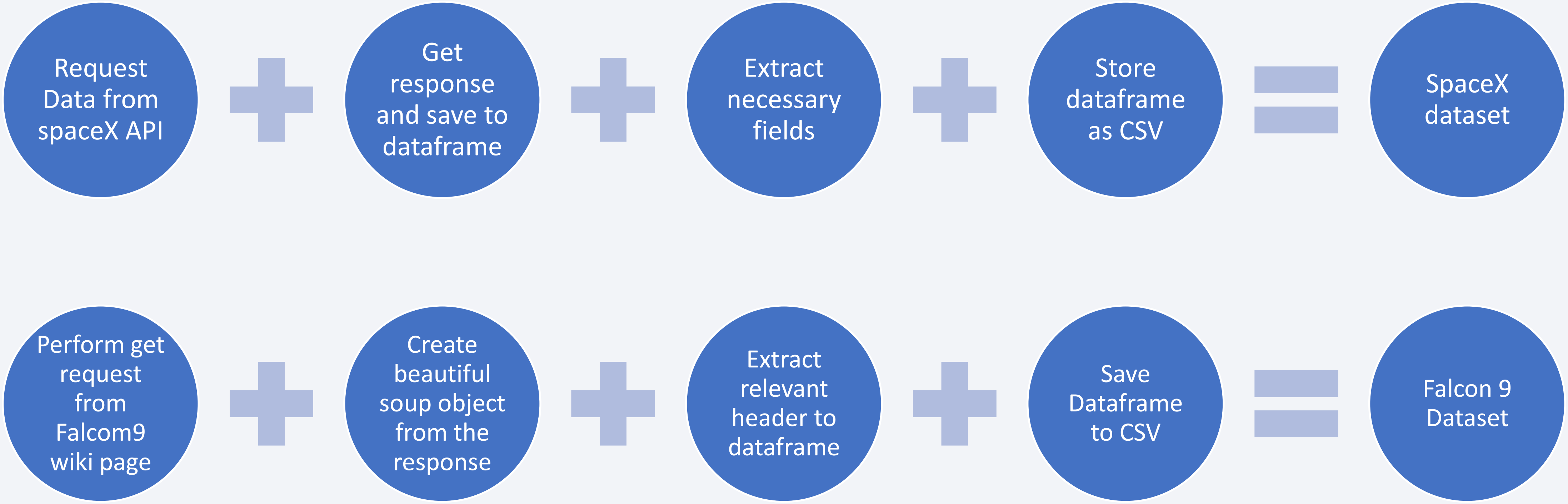
Methodology

Executive Summary

- Data collection methodology:
  - The data was collected via scrapping Falcon 9 and Falcon Heavy launch records on Wikipedia and through SpaceX API
- Perform data wrangling
  - The data was cleaned and prepared for analysis and modelling. E.g missing values were replaced with the mean of the column and creating a class column.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - The data was preprocessed and split into train and test data which were used to train and test different models before select the best model based on accuracy score.

Data Collection

- The data for this project were gotten from structured tables using REST API and beautiful soup (web scrapping)



Request Data from spaceX API + Get response and save to dataframe + Extract necessary fields + Store dataframe as CSV = SpaceX dataset

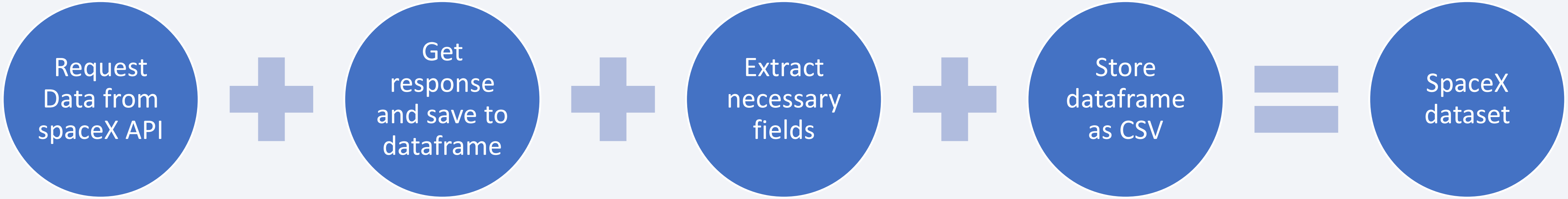
Perform get request from Falcom9 wiki page + Create beautiful soup object from the response + Extract relevant header to dataframe + Save Dataframe to CSV = Falcon 9 Dataset

Data Collection – SpaceX API

API request made to SpaceX API

Helper functions were used to extract relevant data

Dataframe was finaaly stored as a CSV file



Request Data from spaceX API

Get response and save to dataframe

Extract necessary fields

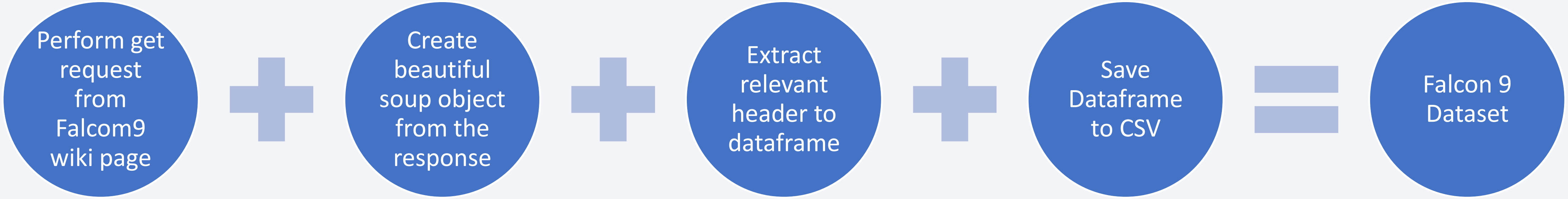
Store dataframe as CSV

SpaceX dataset

GitHub Link

Data Collection - Scraping

- Perform get request from Falcon9 wiki page
- Create beautiful soup object using the response from request
- Extract relevant headers to dataframe using helper functions
- Convert and store dataframe in CSV


Perform get request from Falcom9 wiki page
Create beautiful soup object from the response
Extract relevant header to dataframe
Save Dataframe to CSV
Falcon 9 Dataset

GitHub Link

Data Wrangling

The data wrangling step involved the following:

- Checked for percentage of missing values and it was observed that only Landing Pad had missing values
- Check for the datatypes of the different variables to determine which will be encoded
- A new column ‘class’ was created using the outcome column to serve as the predictive class. In the class column, 1 signifies successful landing while 0 signifies failure to land

GitHub Link

EDA with Data Visualization

During exploratory data analysis, three major charts were used to explore relationships between variables.

- Scatter plot shows patterns and correlation between two or more variables. It was used to explores relationships between the following variables:
  - Flight Number and Launch Site
  - Payload and Launch Site
  - Flight number and Orbit type
  - Payload and Orbit type
- Line plot was used to show the success rate of lauches over years
- Bar plot was used to visualize the success rate of different Orbits

GitHub Link

EDA with SQL

The data was further explored using SQL to know the following:

- The unique launch sites in space mission were displayed
- 5 records where launch site begins with ‘CCA’
- Total payload mass carried by boosters launched by NASA (CRS)
- Average payload mass carried by booster version F9 v1.1
- Date when the first successful landing outcome in ground pad was achieved
- Names of boosters with success in droneship and have payload mass between 400 and 6000
- Total number of successful and failure mission outcomes
- Name the booster versions that have carried the maximum payload using sub query
- List failed landing outcome in droneship
- Rank the count of landing outcome between 2010-06-04 and 2017-03-20 in descending order

GitHub Link

Build an Interactive Map with Folium

The following map objects were created with the following reasons:

- Circle and marker were used to highlight launch site on the map by placing a circle mark on the launch site coordinate
- MousePosition was used to get coordinates for a particular location on the map
- Polyline was used to draw a line between two points on the map.

The map can be used to answer several questions regarding proximity of launch site and amenities within the area.

GitHub Link

Build a Dashboard with Plotly Dash

There were two charts on the dashboard; a pie chart and a line chart.

- Pie chart was used to visualize the success percentage for all the sites, showing which site had the highest success rate at a glance. It can also show the success and failure rate for the different launch sites.
- Line chart was used to visualize how payload affected mission outcome.

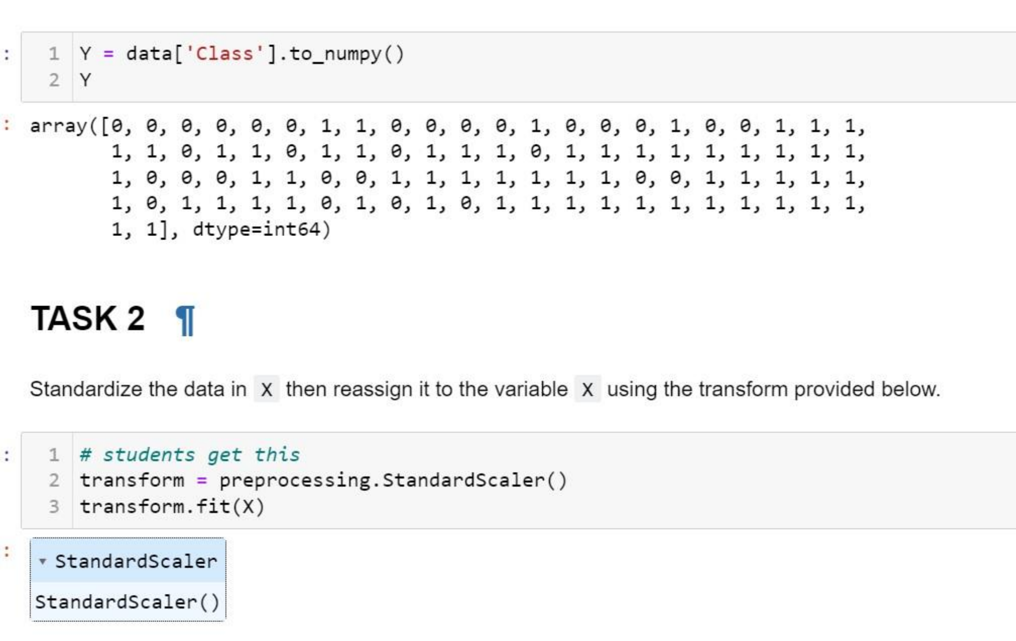
A dropdown menu was also added to aid interacting with the different sites

A slider was also added to aid interacting with the payload.

GitHub Link

Predictive Analysis (Classification)

GitHub Link



```
Y = data['Class'].to_numpy()
Y
```

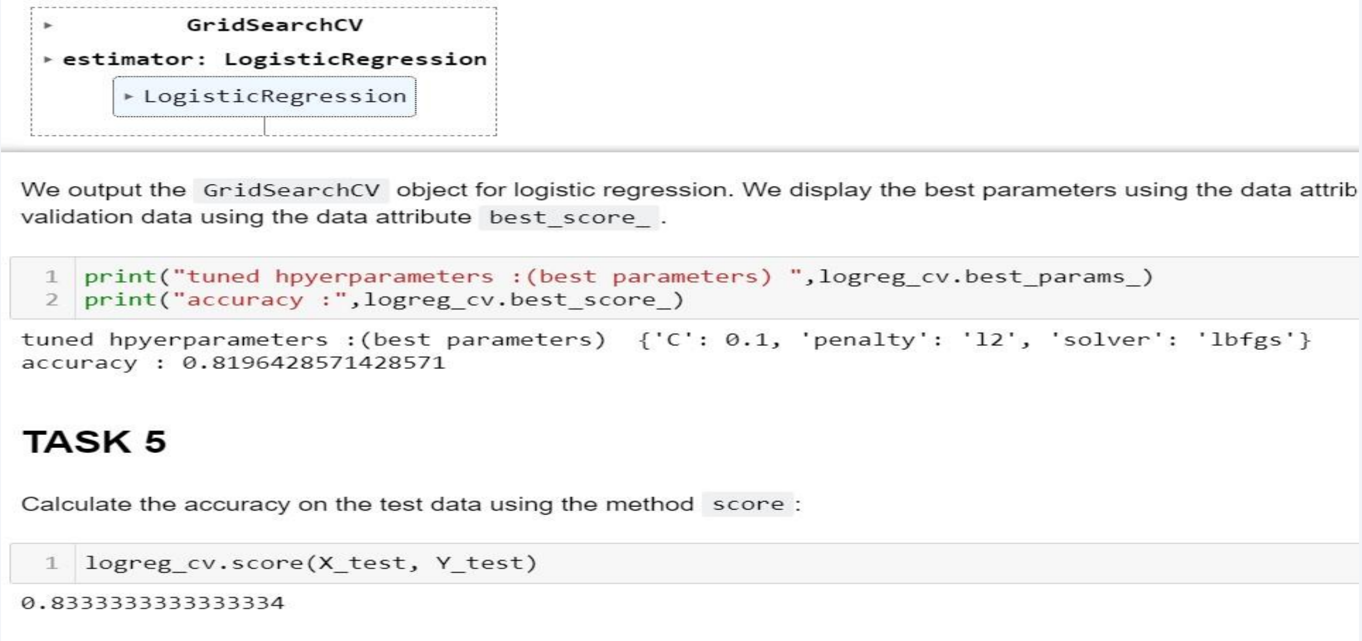
```
array([0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
       1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1], dtype=int64)
```

TASK 2 ¶

Standardize the data in `X` then reassign it to the variable `X` using the transform provided below.

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

```
▾ StandardScaler
StandardScaler()
```



```
▸ GridSearchCV
▸ estimator: LogisticRegression
  ▸ LogisticRegression
```

We output the `GridSearchCV` object for logistic regression. We display the best parameters using the data attrib validation data using the data attribute `best_score_`.

```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("accuracy :",logreg_cv.best_score_)
```

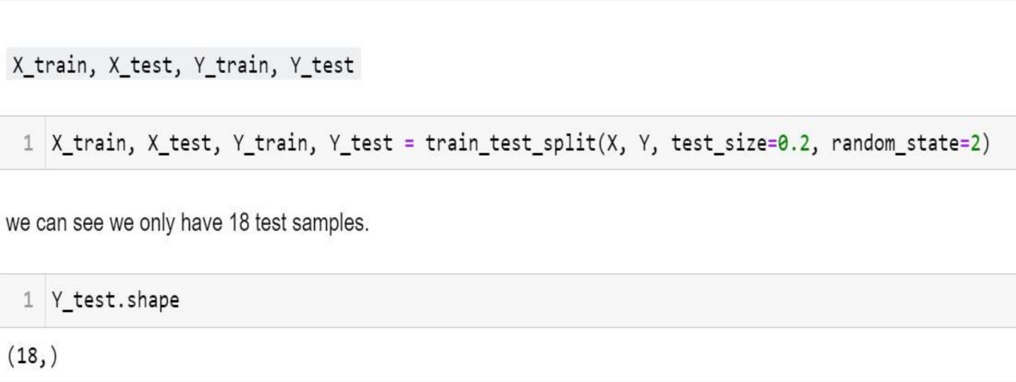
```
tuned hpyerparameters :(best parameters)  {'C': 0.1, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8196428571428571
```

TASK 5

Calculate the accuracy on the test data using the method `score`:

```
logreg_cv.score(X_test, Y_test)
```

```
0.8333333333333334
```



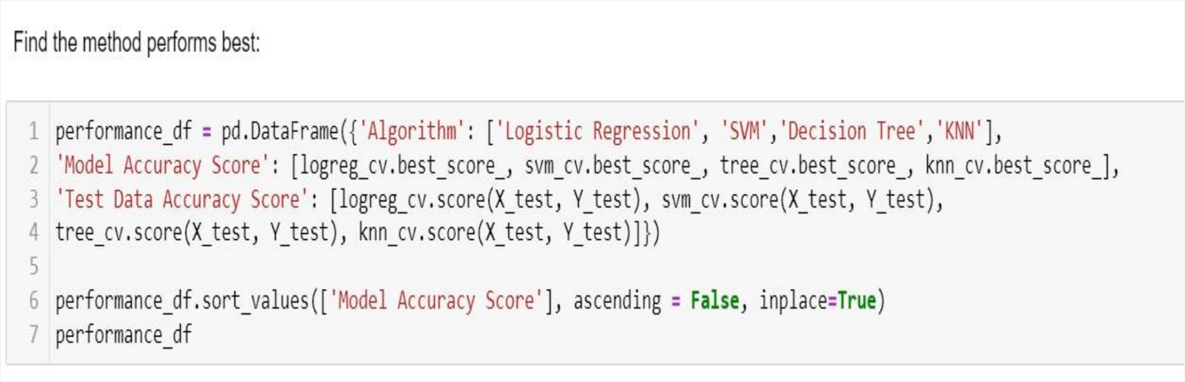
`X_train, X_test, Y_train, Y_test`

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

we can see we only have 18 test samples.

```
Y_test.shape
```

```
(18,)
```



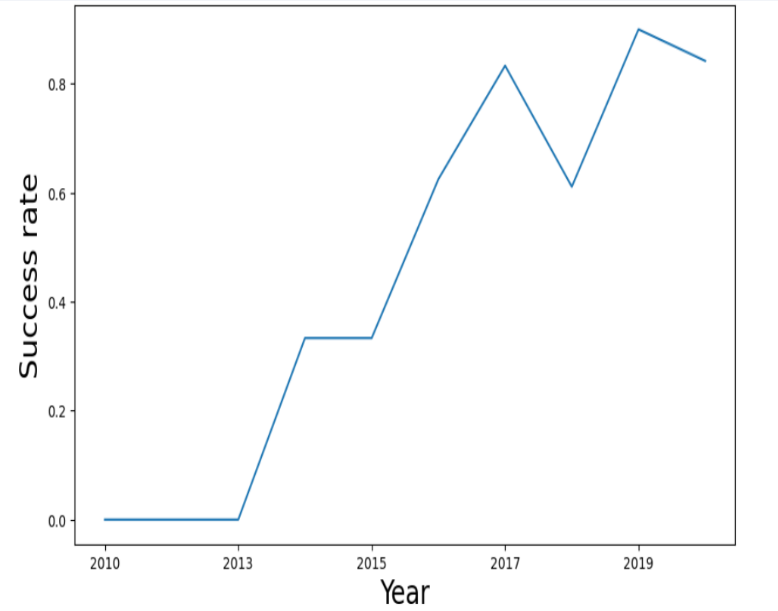
Find the method performs best:

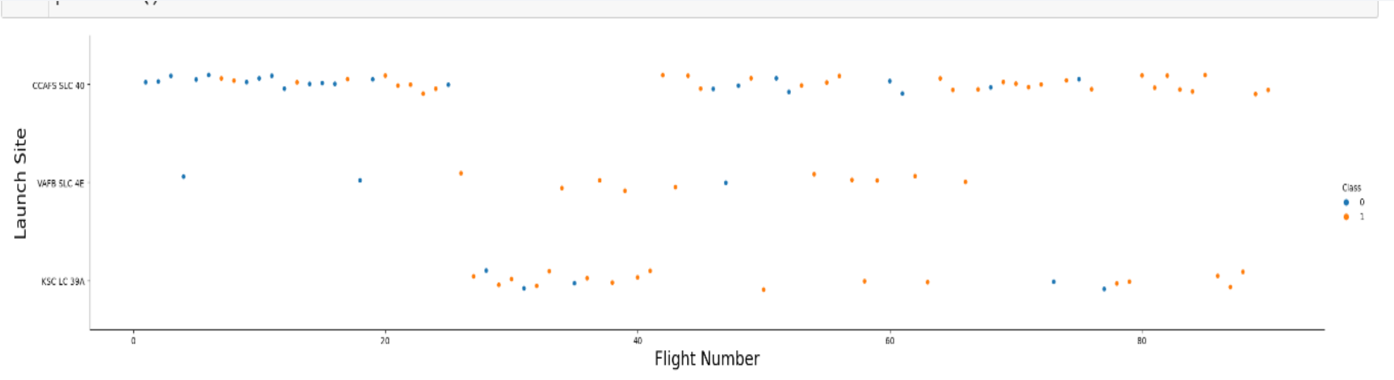
```
performance_df = pd.DataFrame({'Algorithm': ['Logistic Regression', 'SVM','Decision Tree','KNN'],
'Model Accuracy Score': [logreg_cv.best_score_, svm_cv.best_score_, tree_cv.best_score_, knn_cv.best_score_],
'Test Data Accuracy Score': [logreg_cv.score(X_test, Y_test), svm_cv.score(X_test, Y_test),
tree_cv.score(X_test, Y_test), knn_cv.score(X_test, Y_test)]})

performance_df.sort_values(['Model Accuracy Score'], ascending = False, inplace=True)
performance_df
```

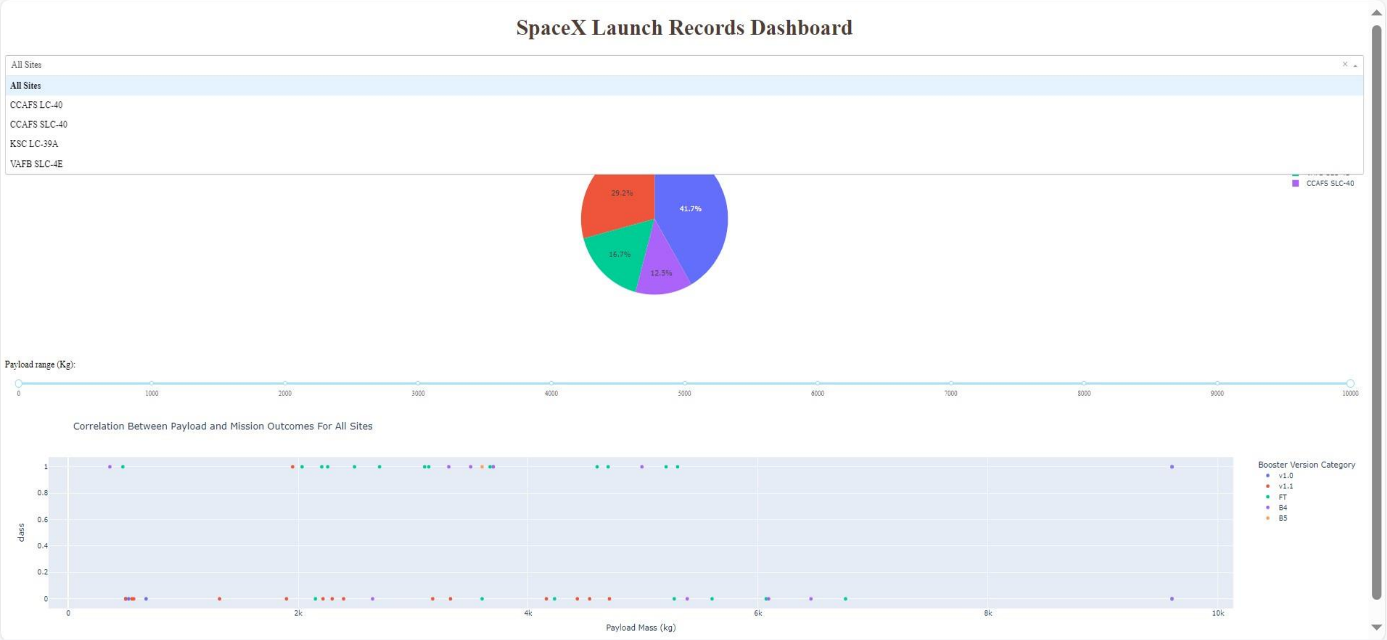
Results

Exploratory data analysis results

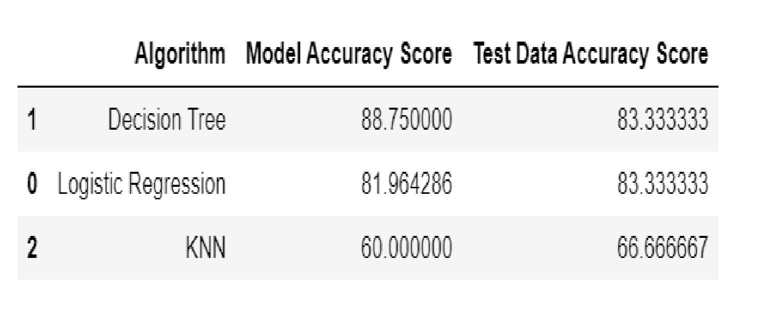




Interactive analytics



Predictive analysis results

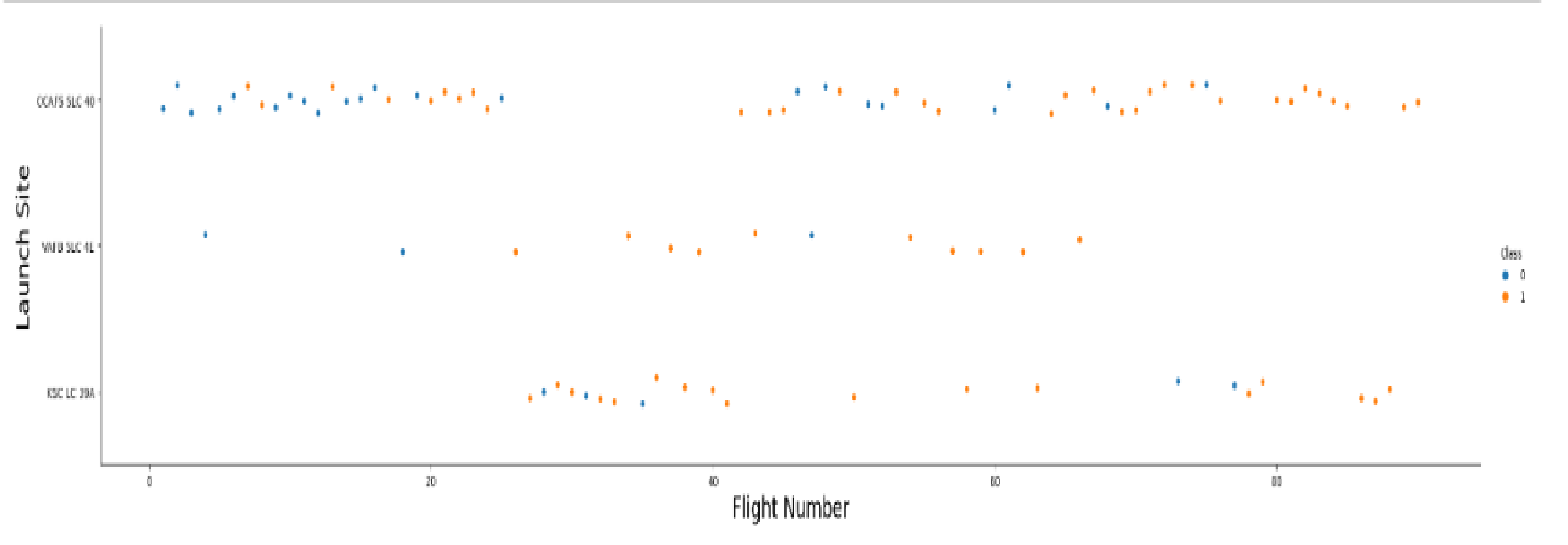


	Algorithm	Model Accuracy Score	Test Data Accuracy Score
1	Decision Tree	88.750000	83.333333
0	Logistic Regression	81.964286	83.333333
2	KNN	60.000000	66.666667

Section 2

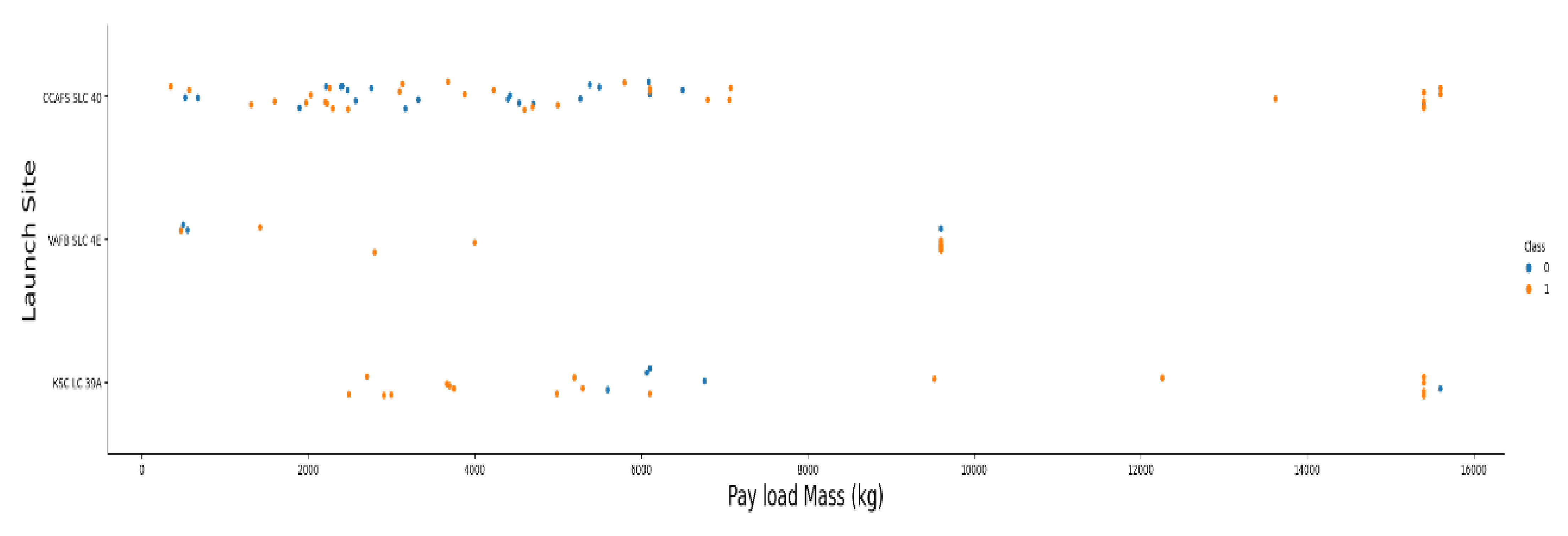
Insights drawn from EDA

Flight Number vs. Launch Site



- CCAPS SLC 40 was the first lauch site and experienced lots of failures and with time got better and started experience more success. KSC LC 39A was the last site to commence launch and has recorded very few failures.
- Note: 1 (orange) indicates success while 0 (blue) indicates failure

Payload vs. Launch Site

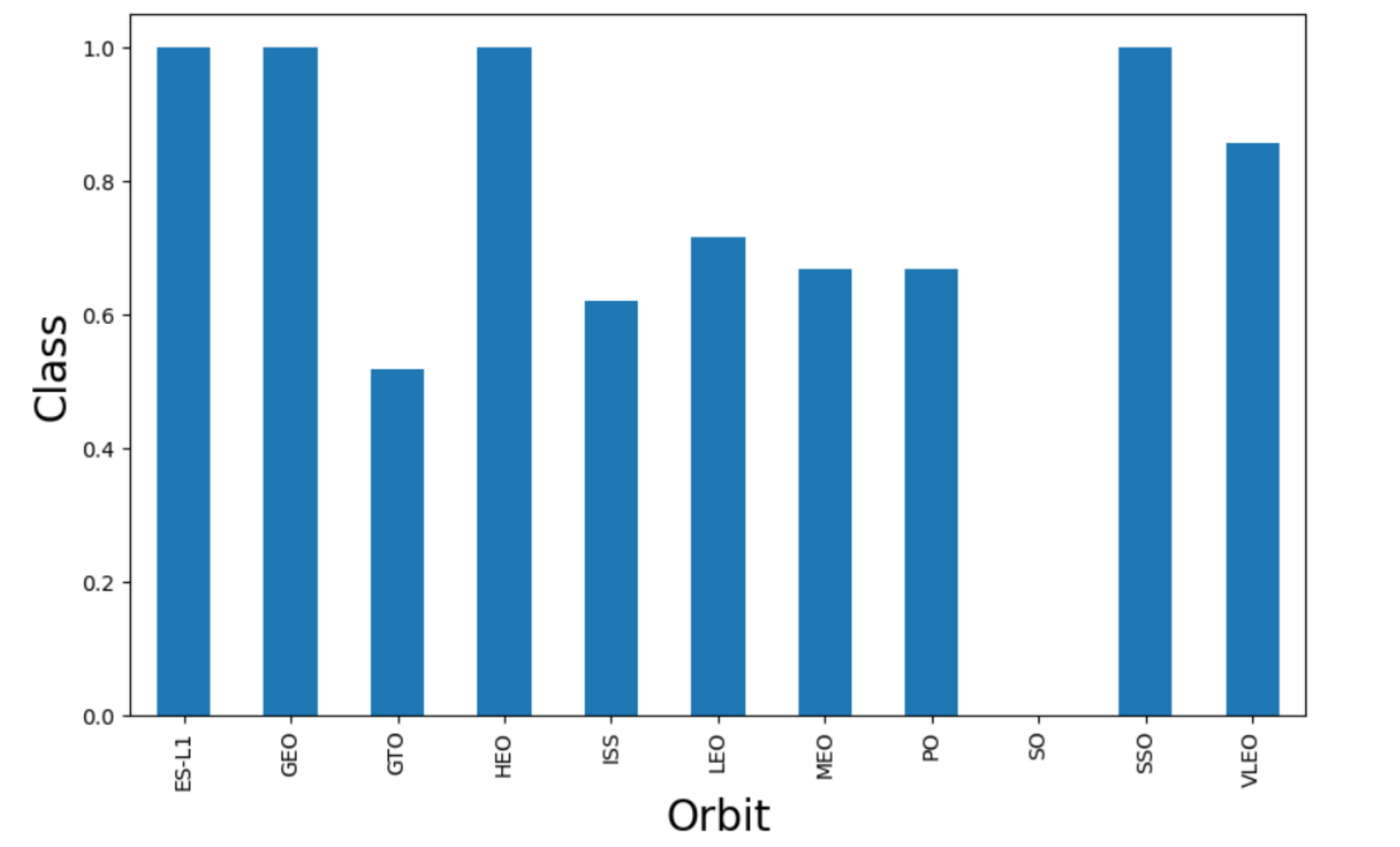


- For CCAPS SLC 40, very high payload above 7000 guarantees success as there are no failures in that region.
- There is also likely to be failure when the payload is between 5000 and 7000 for KSC LC 39A

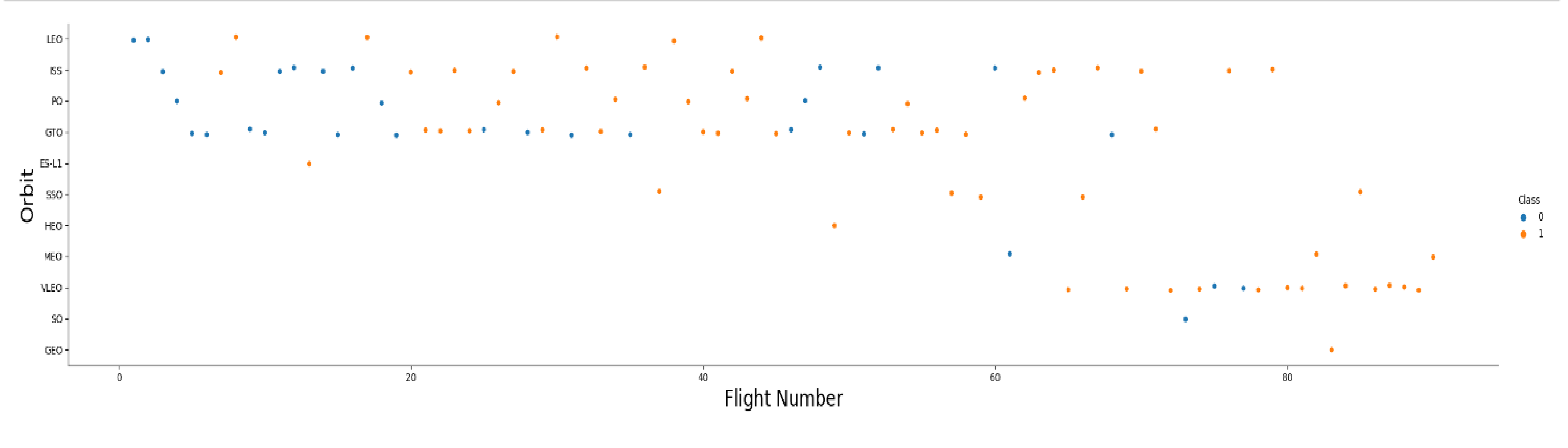
Success Rate vs. Orbit Type

- ESL-1, GEO, HEO AND SSO Orbit all had very high success rate of about 100%

- SO had no success rate and would require further investigation.

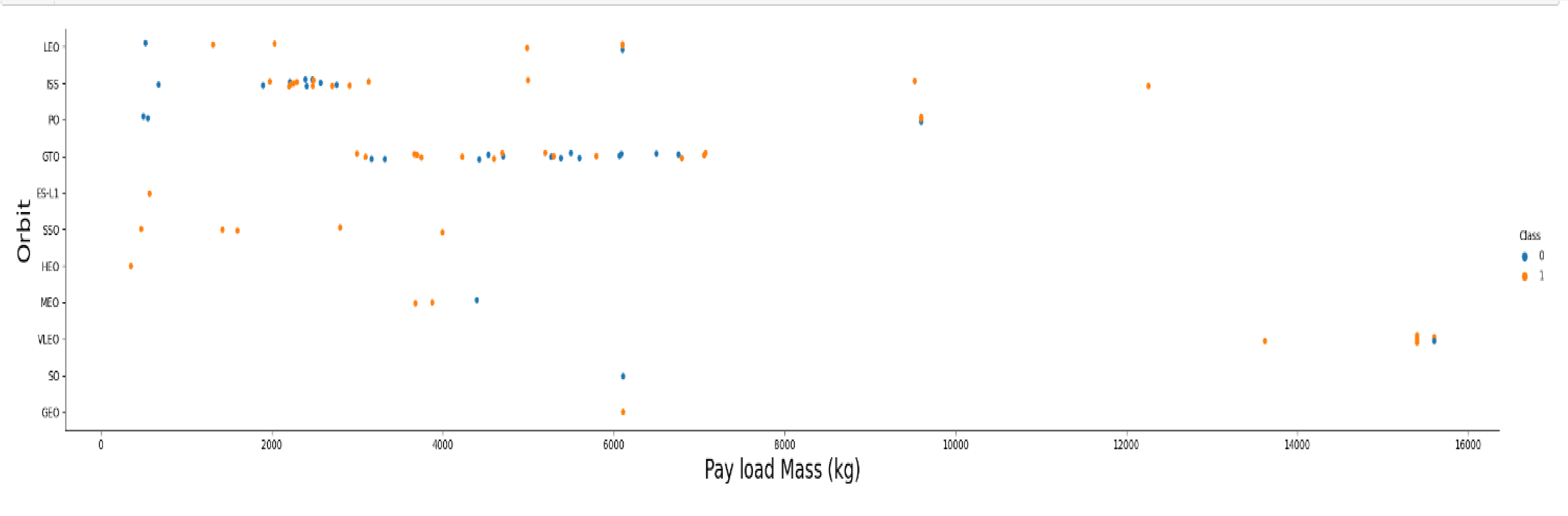


Flight Number vs. Orbit Type



- Most of the flight went off to LEO, ISS, PO and GTO.
- The first few flights did not record any success. Success rate eventually increased with flight number. As flight number increases, success rate increased

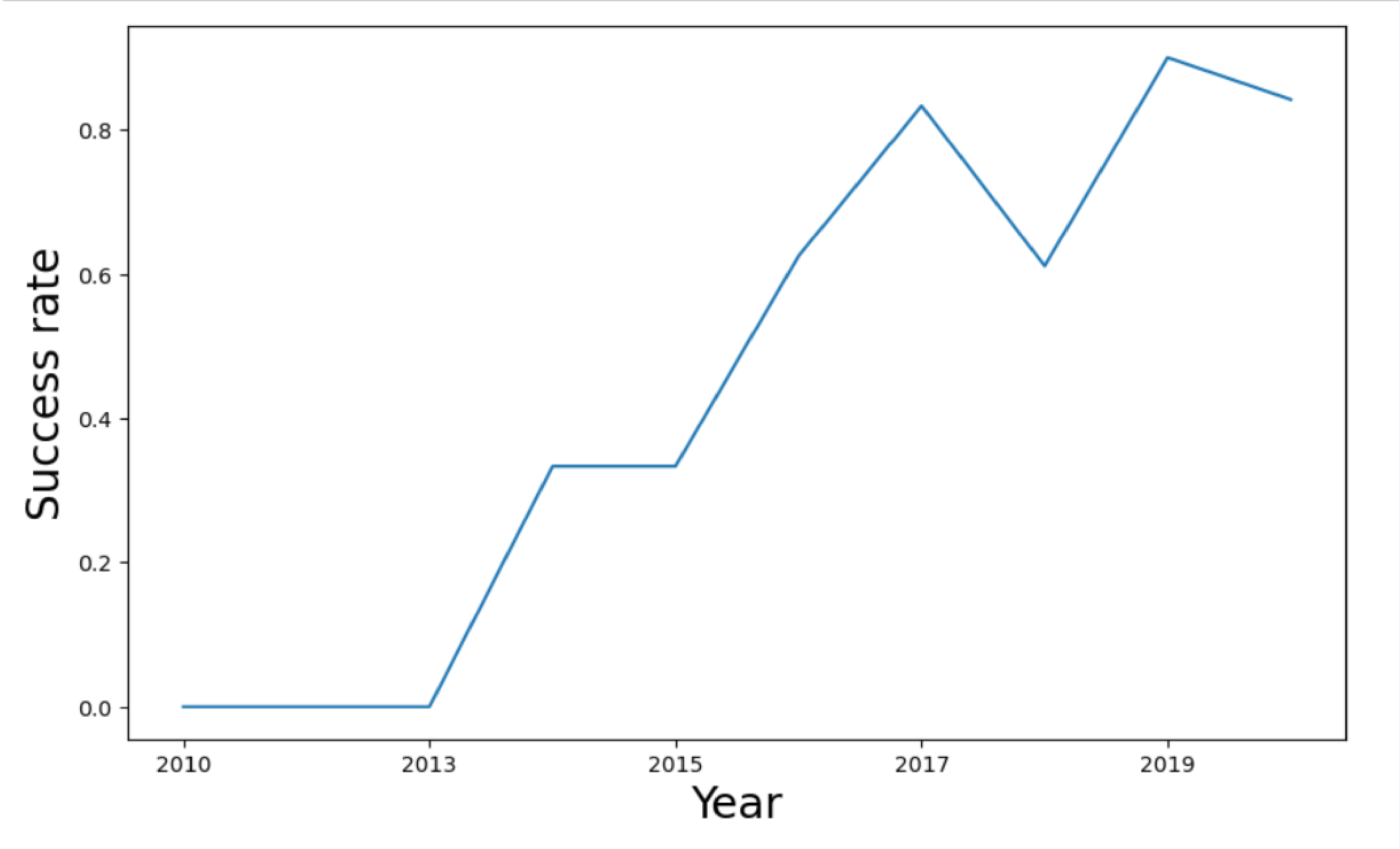
Payload vs. Orbit Type



- ES L1, SSO and HEO recorded 100% success with payload mass less than 4000kg
- GTO payload mass ranged from 3000kg to 7500kg but does not guarantee success at any point

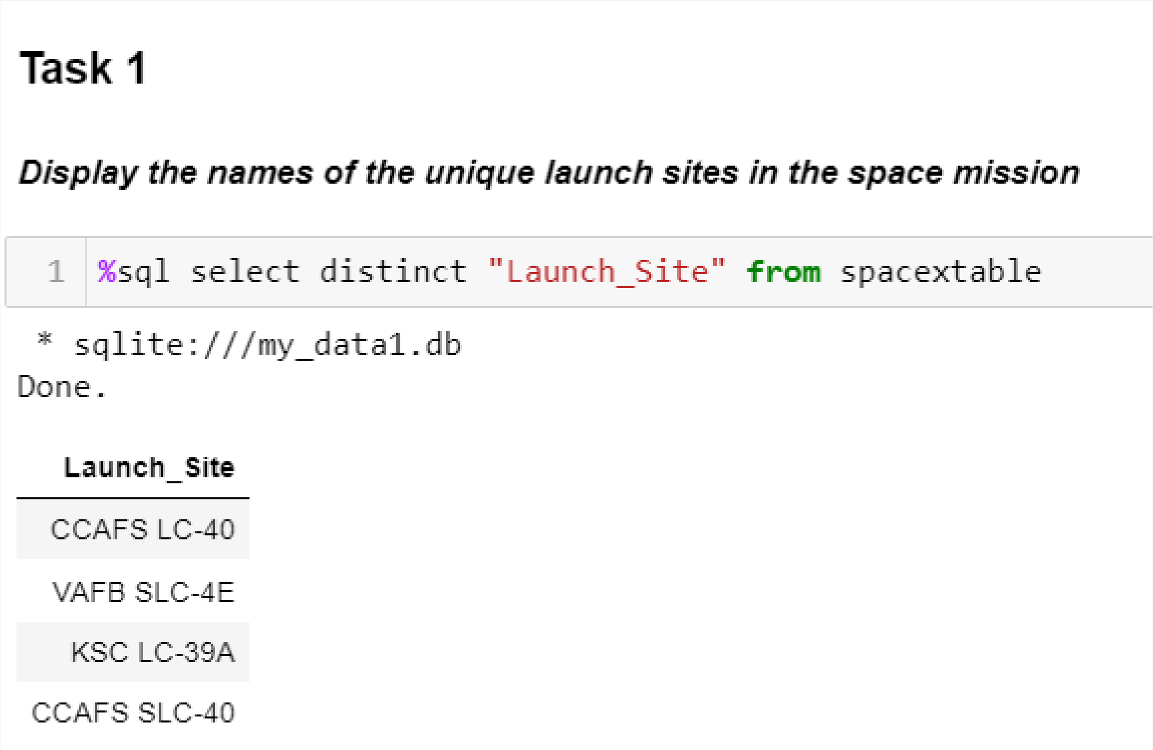
Launch Success Yearly Trend

- Success rate remained same from 2010 to 2013.
- It increased in 2014 and stayed that way in 2015.
- Between 2015 and 2017 there was a steady increase in success.
- Success rate dropped in in 2018 and 2020.



All Launch Site Names

- The query retrieves 4 unique Launch Sites location in the space mission
- This was achieved with the use of distinct keyword



Task 1

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

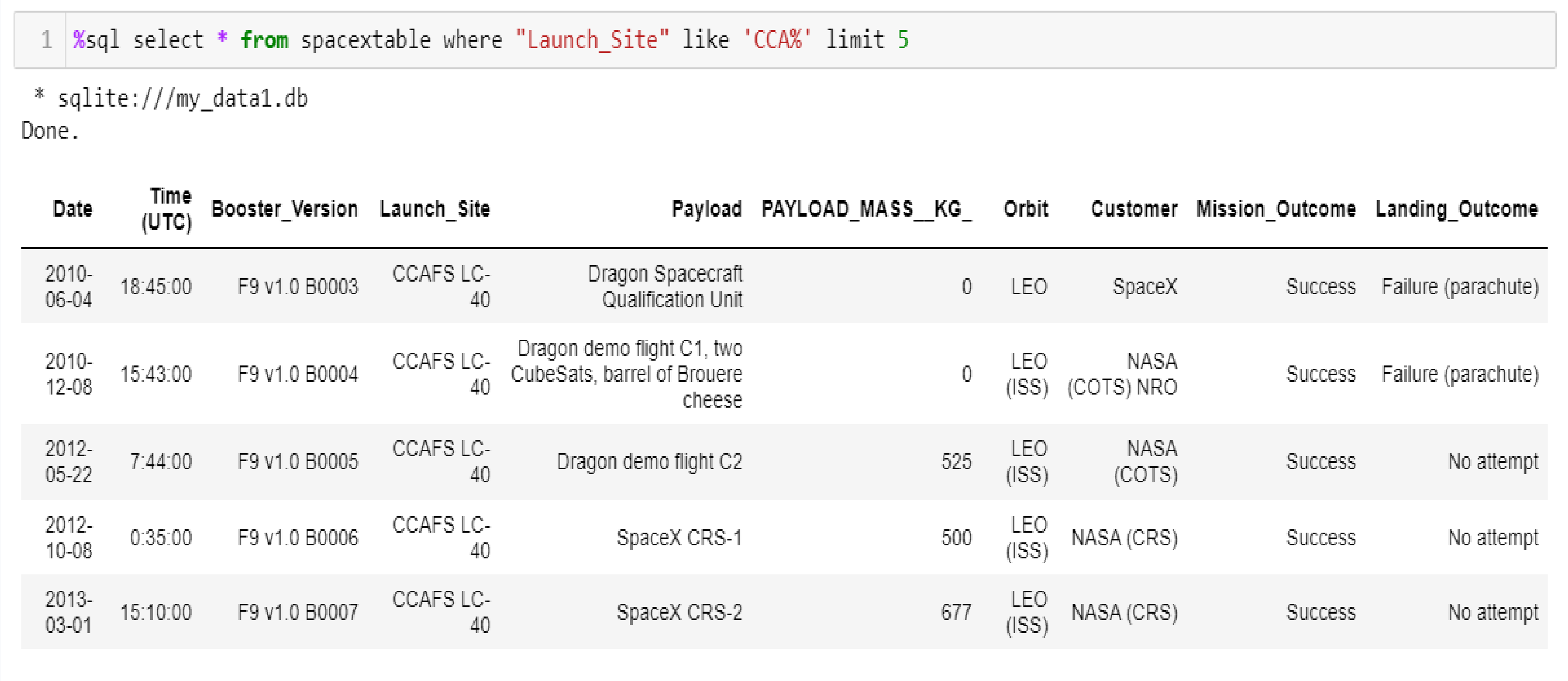
```
%sql select distinct "Launch_Site" from spacextable
```

```
 * 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'

- The like keyword and wildcard symbol were used to search for records that launch sites begin with CCA.
- Limit was used to return only 5 records.



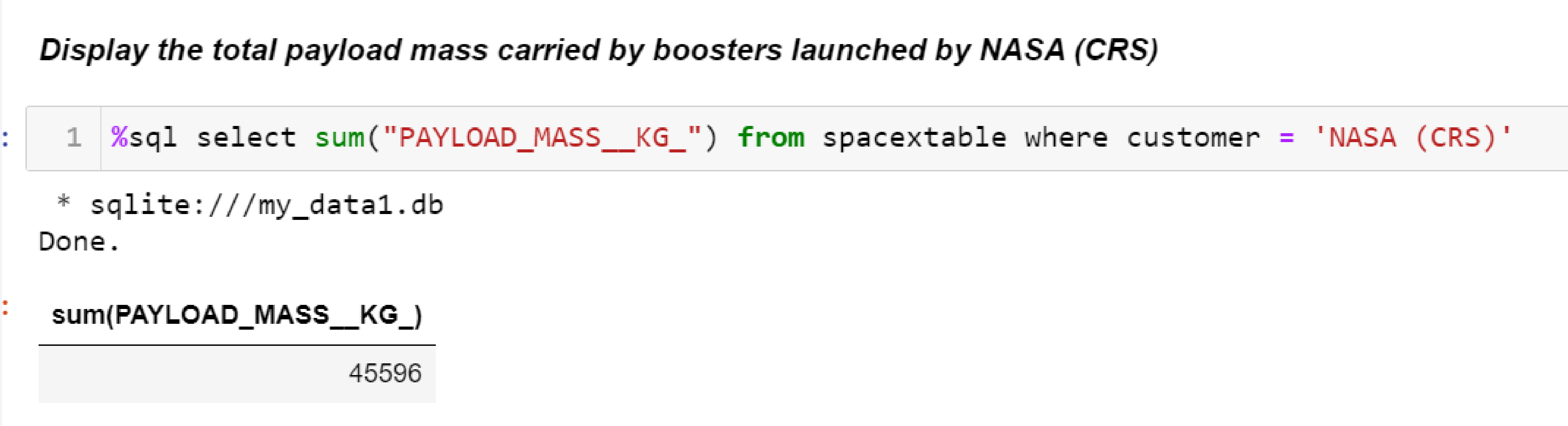
```
%sql select * from spacextable where "Launch_Site" like 'CCA%' limit 5
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	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	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	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

- The total payload mass carried by boosters and launched by NASA is 45,596kg and was achieved using sum aggregate and specifying the customer using a where clause



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

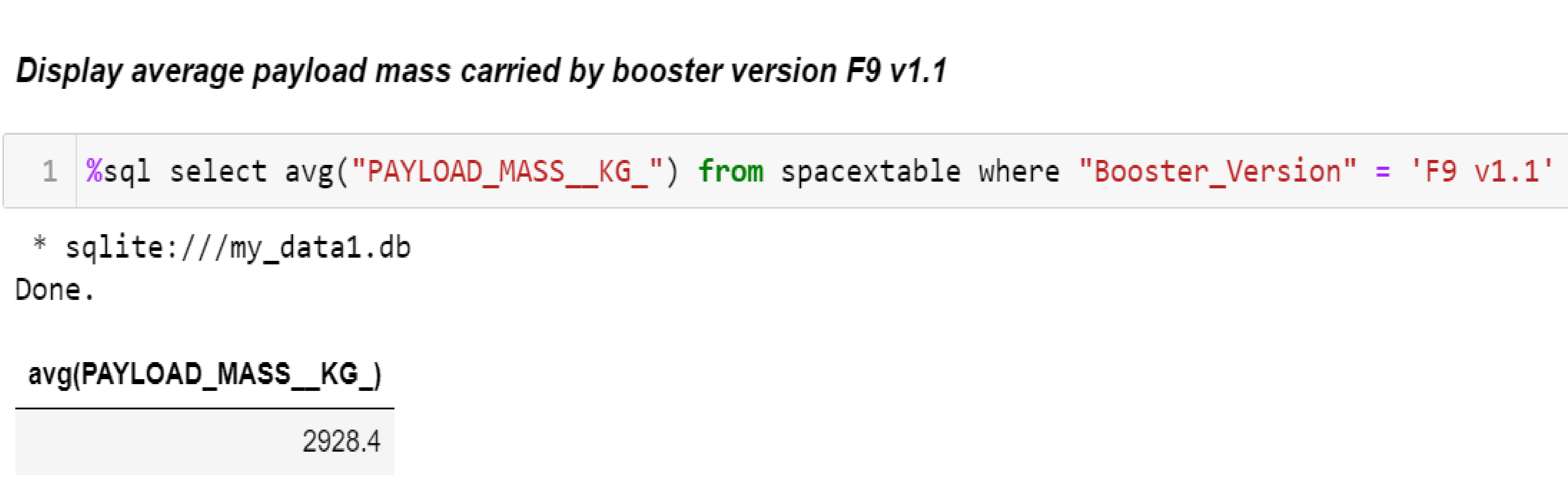
```
%sql select sum("PAYLOAD_MASS__KG_") from spacextable where customer = 'NASA (CRS)'
```

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

sum(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

- This was achieved using average aggregate function and specifying a where clause



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

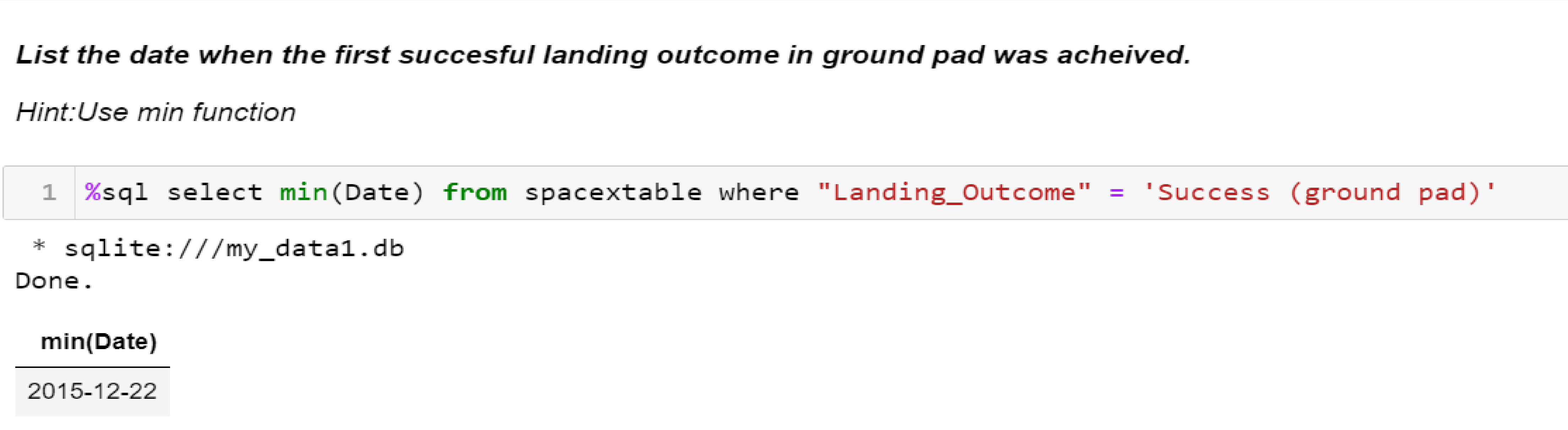
```
%sql select avg("PAYLOAD_MASS__KG_") from spacextable where "Booster_Version" = 'F9 v1.1'
```

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

avg(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

- This was achieved using min function on the date column and specifying the type of landing outcome using a where clause.



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

Hint:Use min function

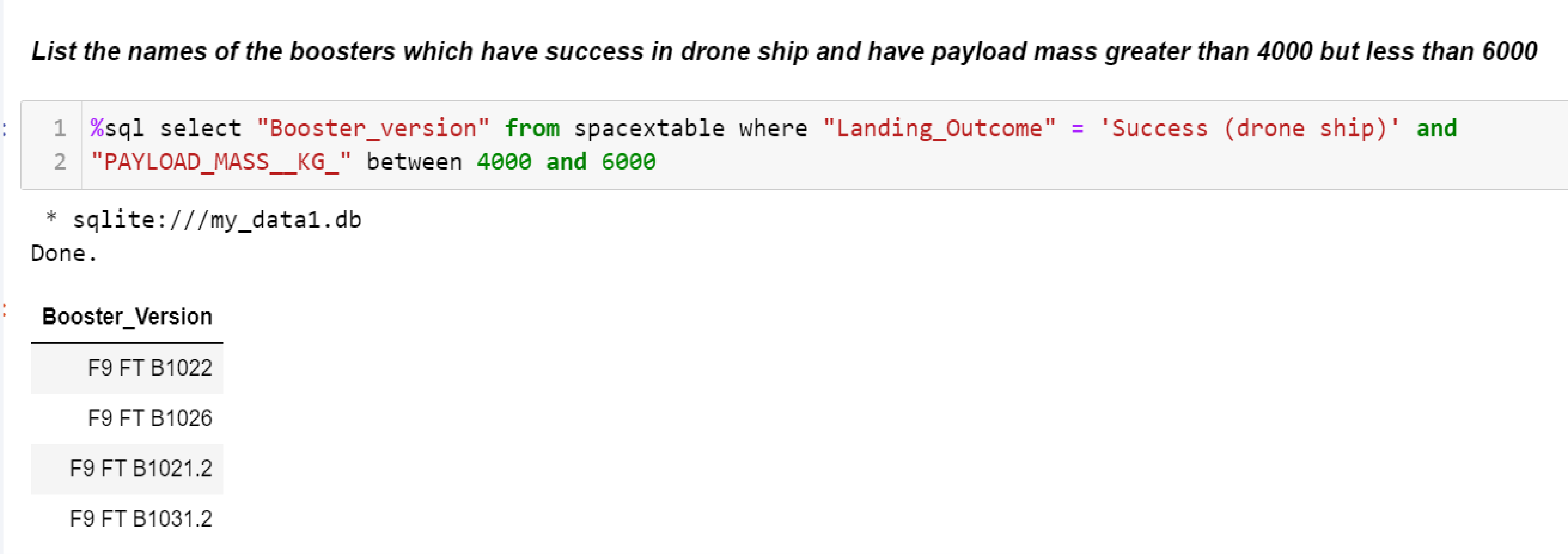
```
%sql select min(Date) from spacextable where "Landing_Outcome" = 'Success (ground pad)'
```

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

min(Date)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- This was achieved using conditional statements



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

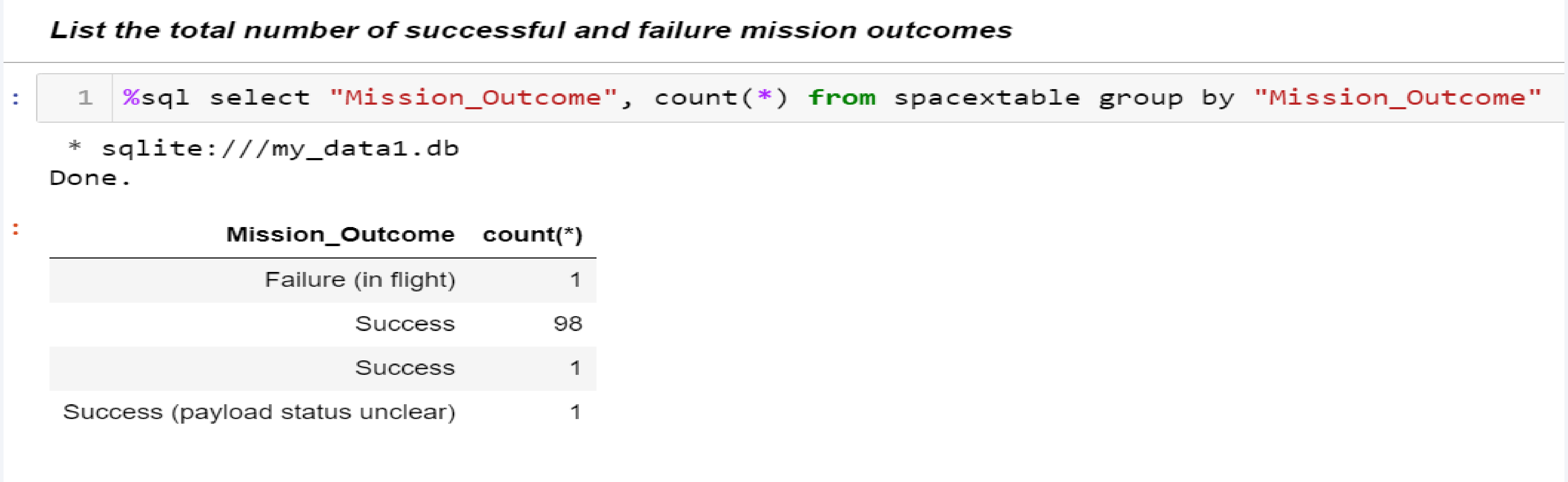
```
%sql select "Booster_version" from spacextable where "Landing_Outcome" = 'Success (drone ship)' and
"PAYLOAD_MASS__KG_" between 4000 and 6000
```

```
 * 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

- This was achieved using count and group by to specify the type of outcome



List the total number of successful and failure mission outcomes

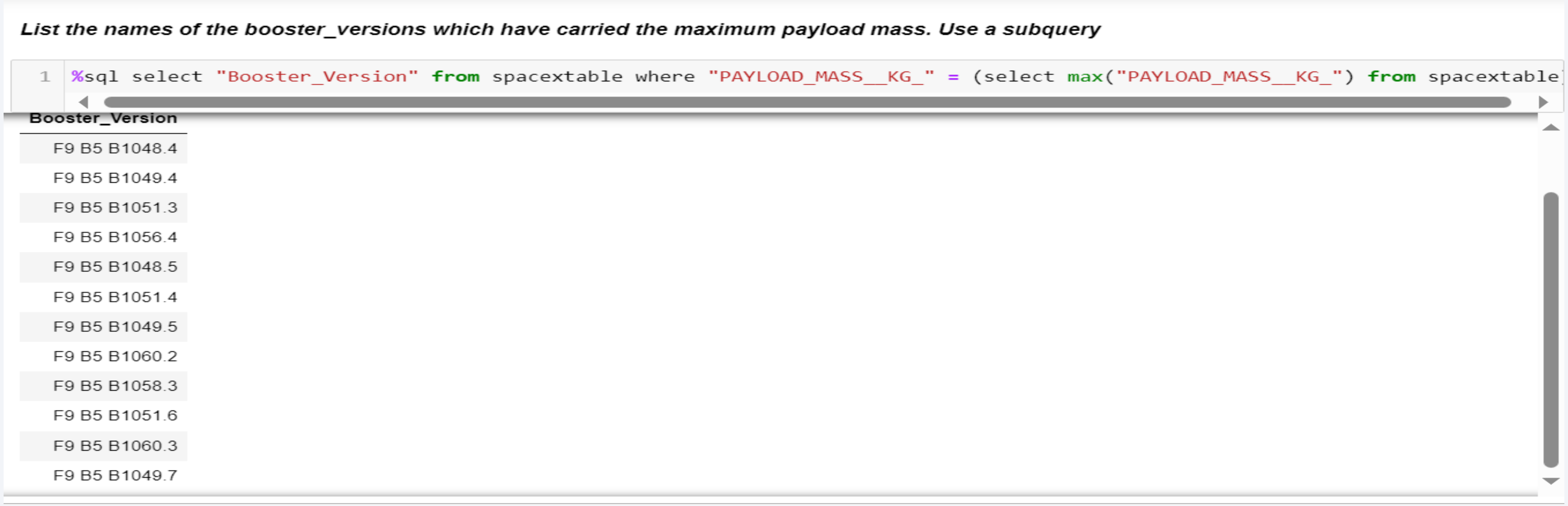
```
%sql select "Mission_Outcome", count(*) from spacextable group by "Mission_Outcome"
```

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

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

Boosters Carried Maximum Payload

- This was achieved using a sub query and specifying a where clause



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

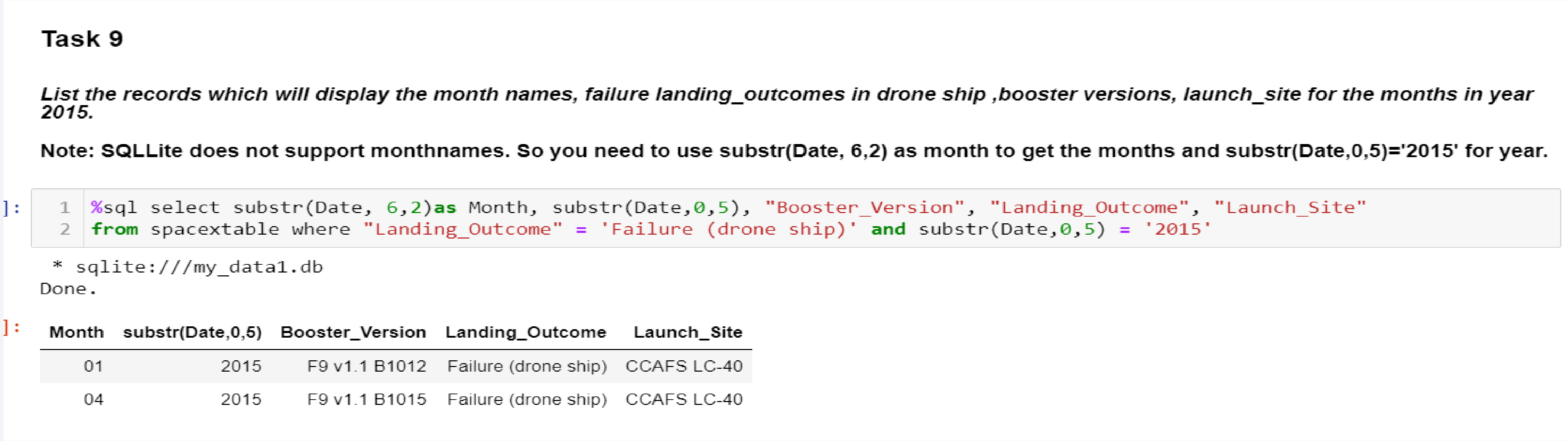
```
%sql select "Booster_Version" from spacextable where "PAYLOAD_MASS__KG_" = (select max("PAYLOAD_MASS__KG_") from spacextable
```

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

There are just 2 failed landing outcomes in drone ship in 2015.

- This was achieved using substr on the date column



Task 9

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

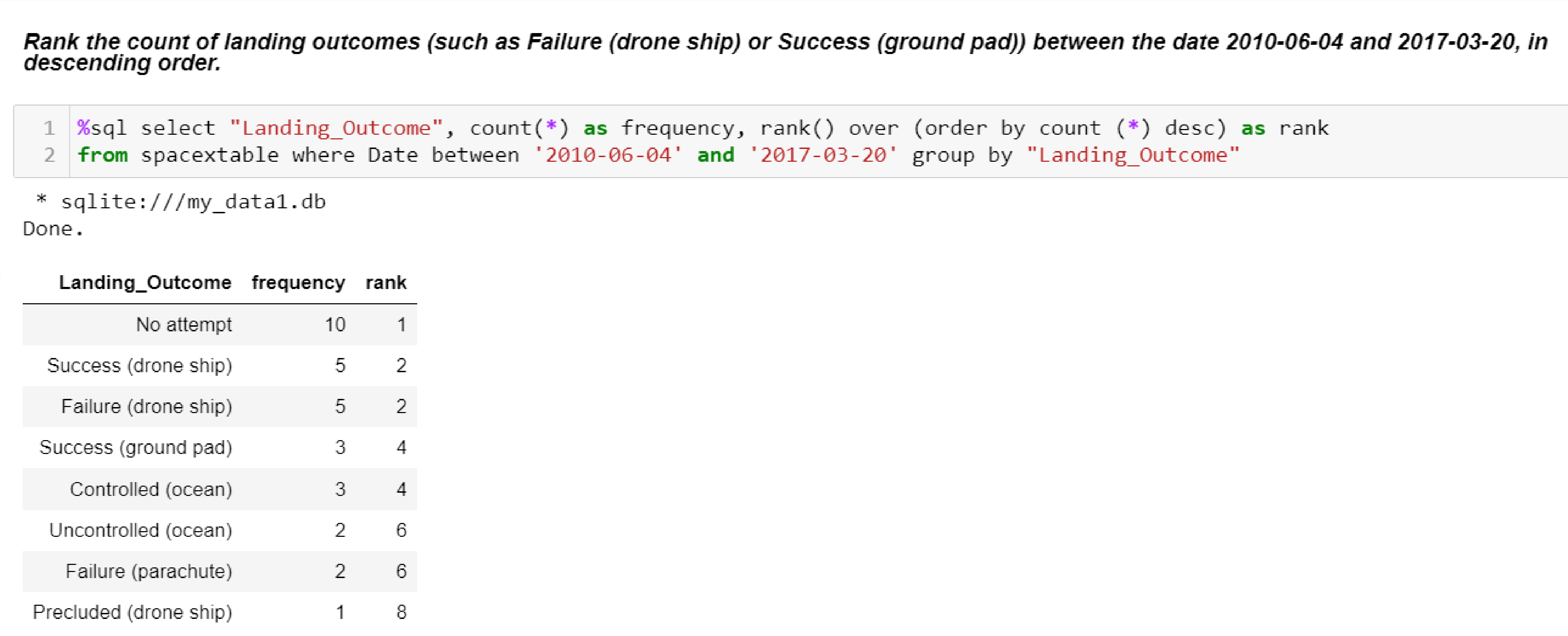
```
%sql select substr(Date, 6,2)as Month, substr(Date,0,5), "Booster_Version", "Landing_Outcome", "Launch_Site"
from spacextable where "Landing_Outcome" = 'Failure (drone ship)' and substr(Date,0,5) = '2015'
```

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

Month	substr(Date,0,5)	Booster_Version	Landing_Outcome	Launch_Site
01	2015	F9 v1.1 B1012	Failure (drone ship)	CCAFS LC-40
04	2015	F9 v1.1 B1015	Failure (drone ship)	CCAFS LC-40

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

- This was achieved using Rank() and specifying an 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", count(*) as frequency, rank() over (order by count (*) desc) as rank
from spacextable where Date between '2010-06-04' and '2017-03-20' group by "Landing_Outcome"
```

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

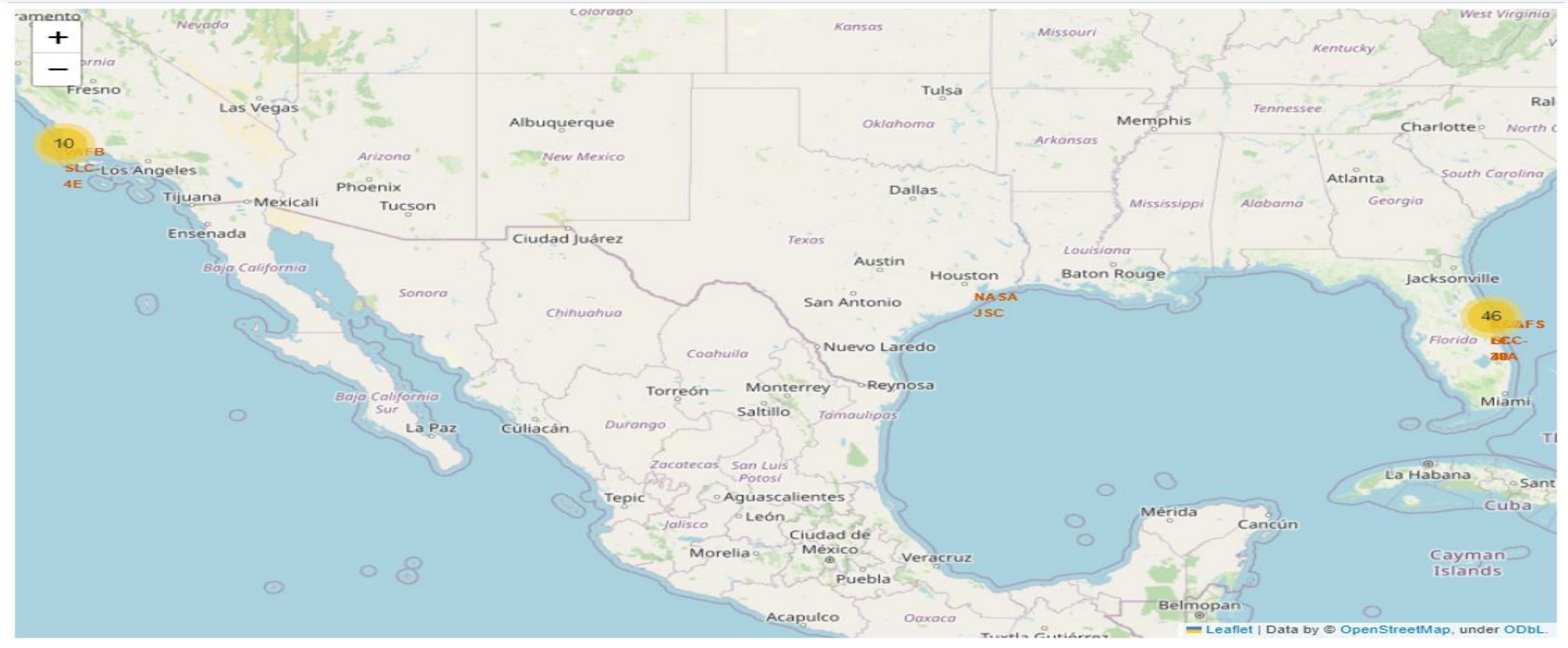
Landing_Outcome	frequency	rank
No attempt	10	1
Success (drone ship)	5	2
Failure (drone ship)	5	2
Success (ground pad)	3	4
Controlled (ocean)	3	4
Uncontrolled (ocean)	2	6
Failure (parachute)	2	6
Precluded (drone ship)	1	8

Section 3

Launch Sites Proximities Analysis

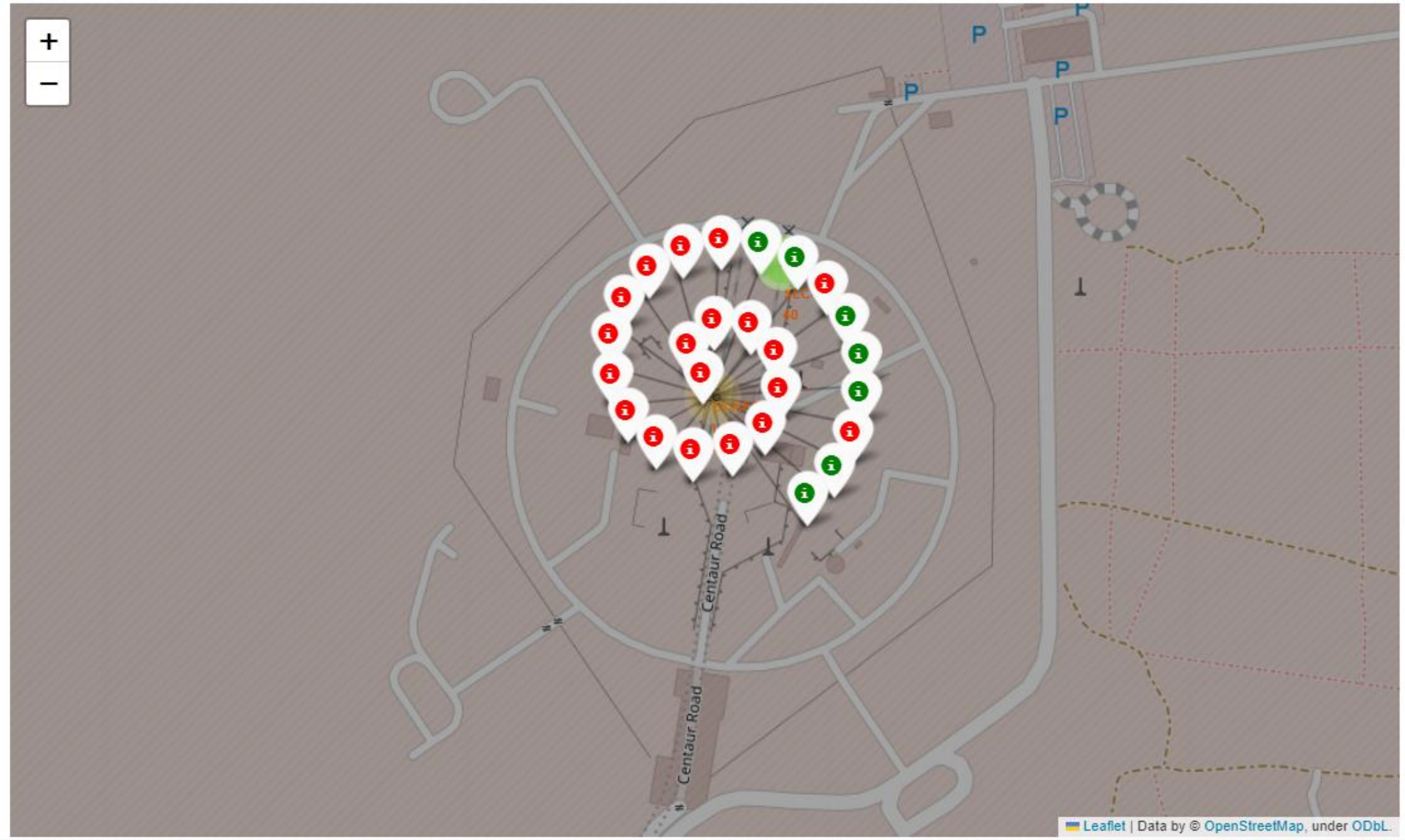
Folium Map showing All Launch Sites

- 10 Launch sites are close to Los Angeles while 46 are close to Jacksonville.
- All Launch sites are close to the sea.

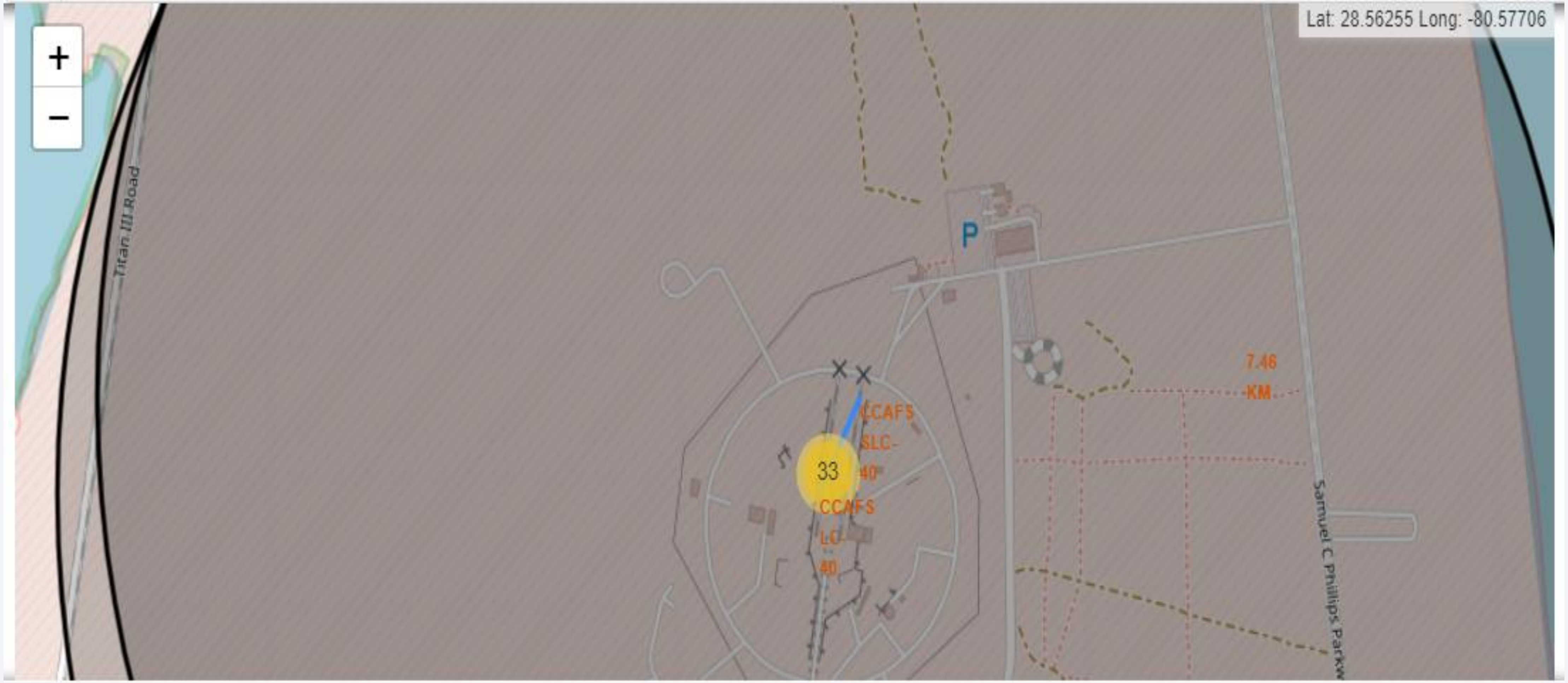


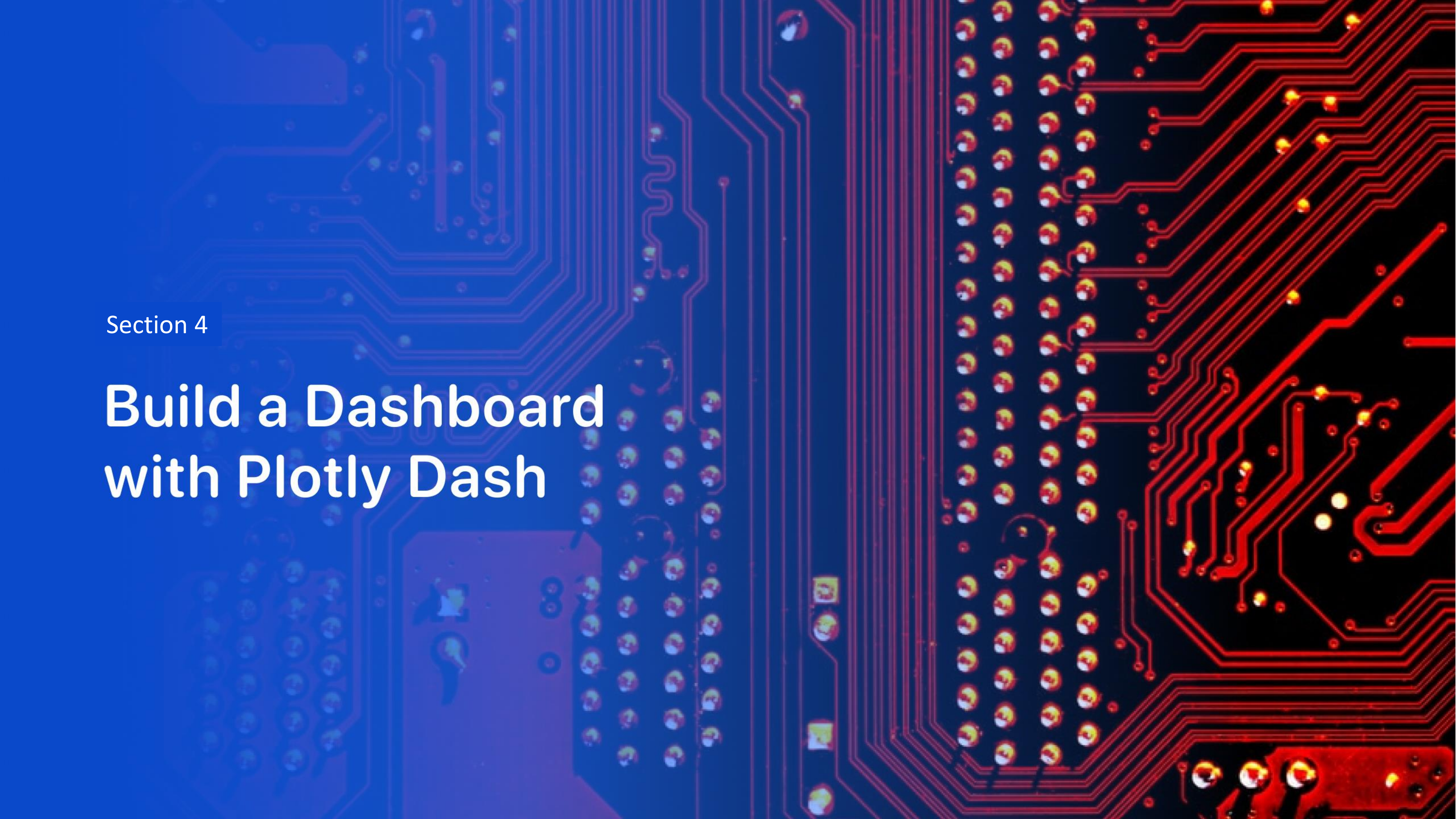
<Folium Map Showing labeled outcome

- In the below launch site, there were more failed launches than successful launches.
- Note: green indicates successful while red indicates failure



Folium Map Showing proximity of Launch Site to coastline

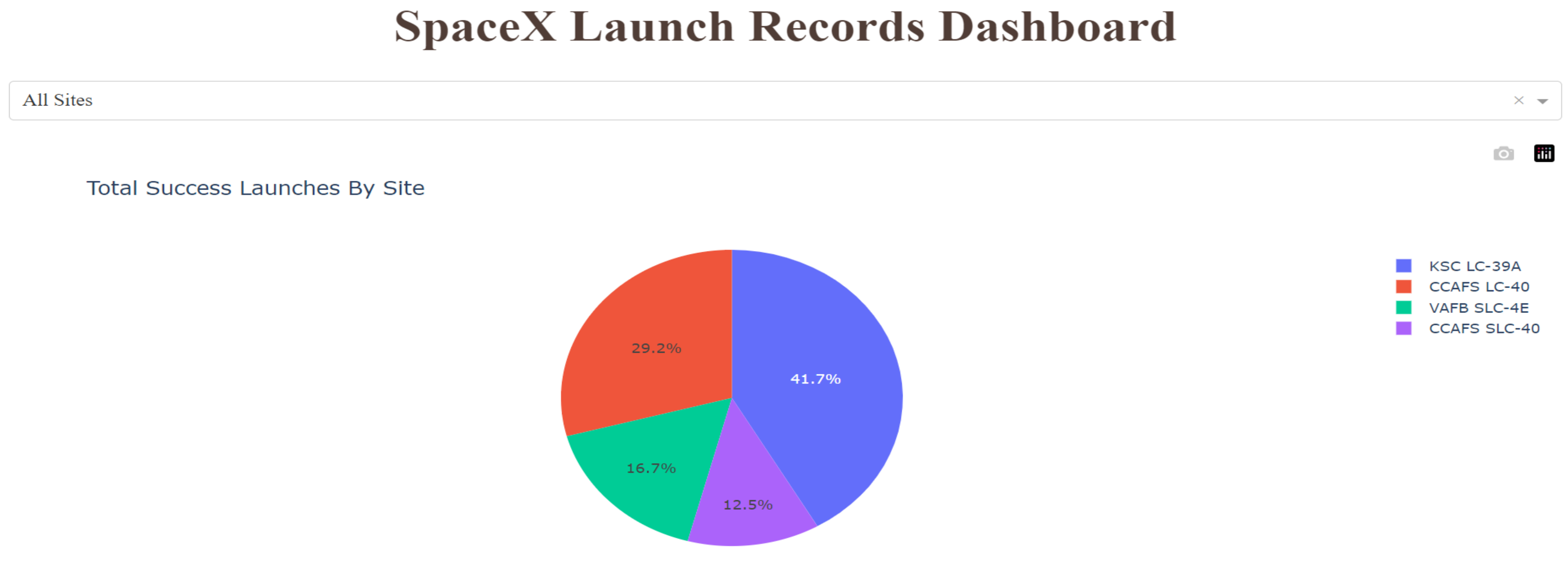




Section 4

Build a Dashboard with Plotly Dash

Launch success count for all sites

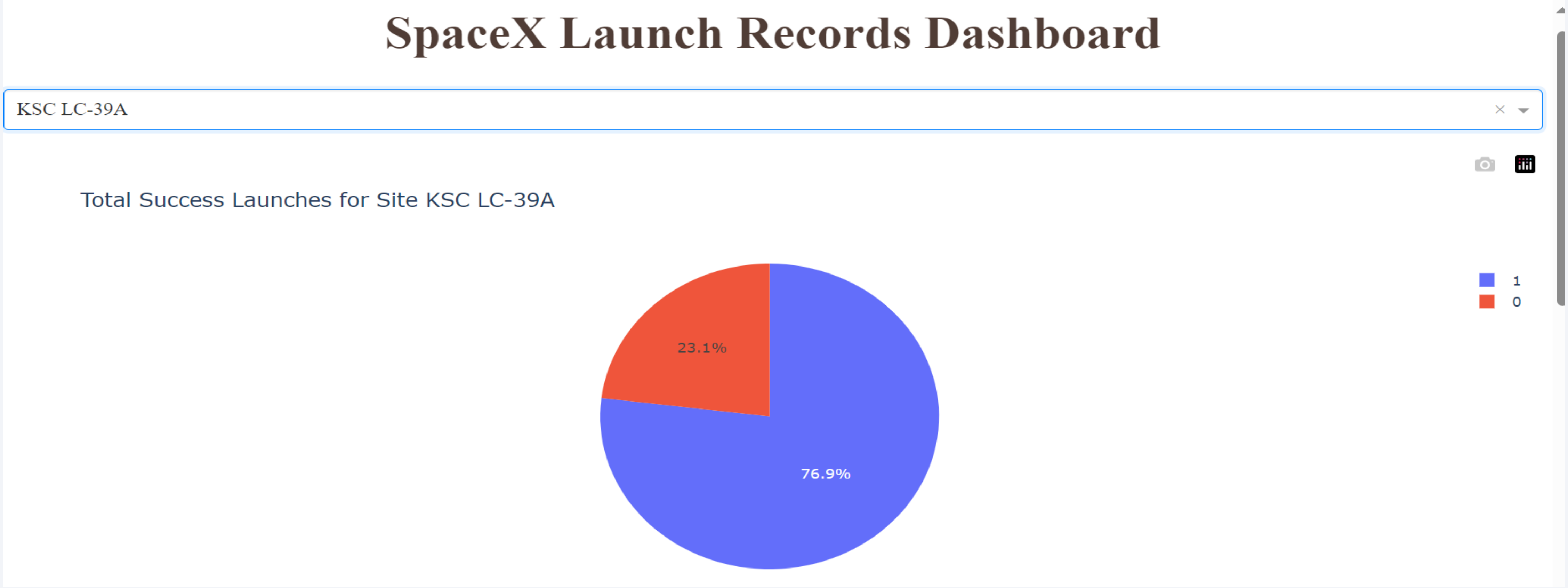


SpaceX Launch Records Dashboard

All Sites

- Over 40% of the total successes was achieved by KSC LC-39A which is the highest followed by CCAFS LC-40 while the least success was recorded by CCAFS SLC-40.

Launch Site with Highest Success Rate



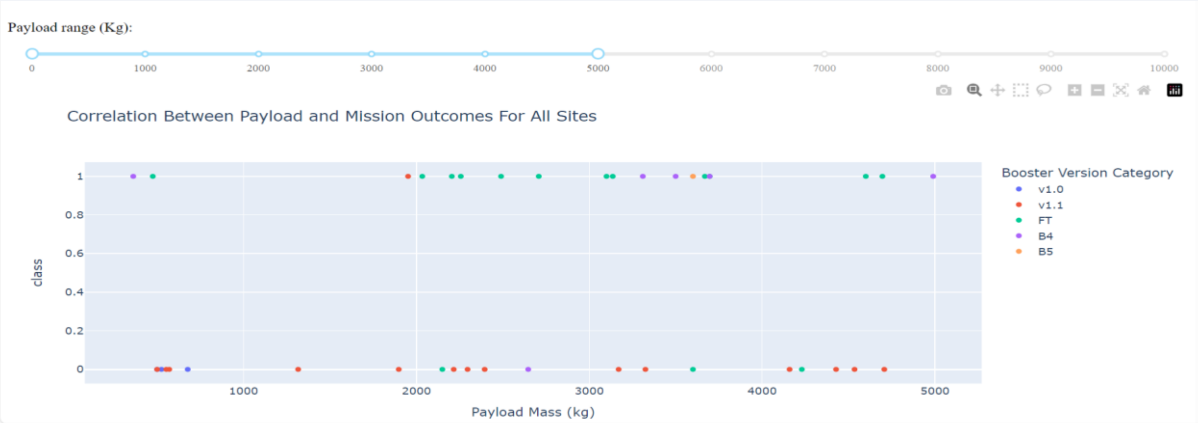
SpaceX Launch Records Dashboard

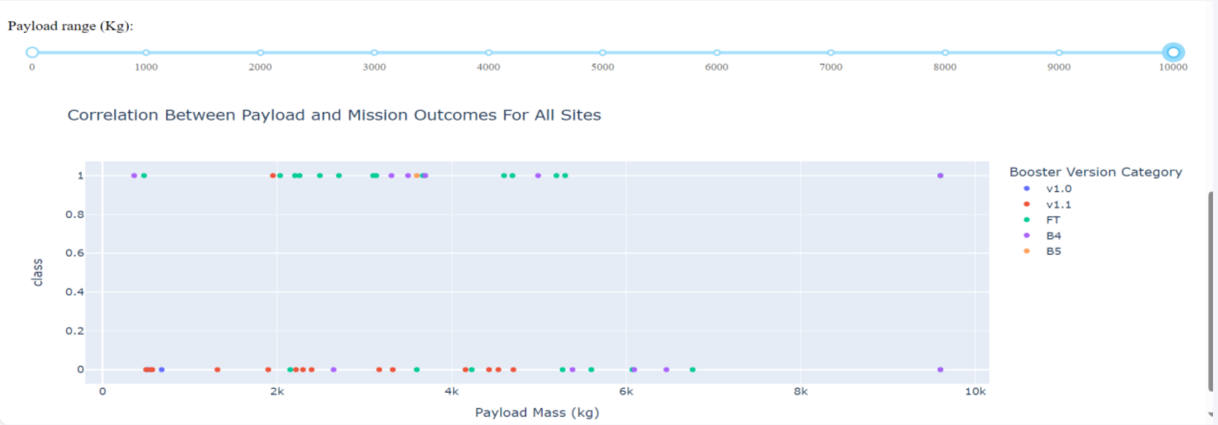
KSC LC-39A

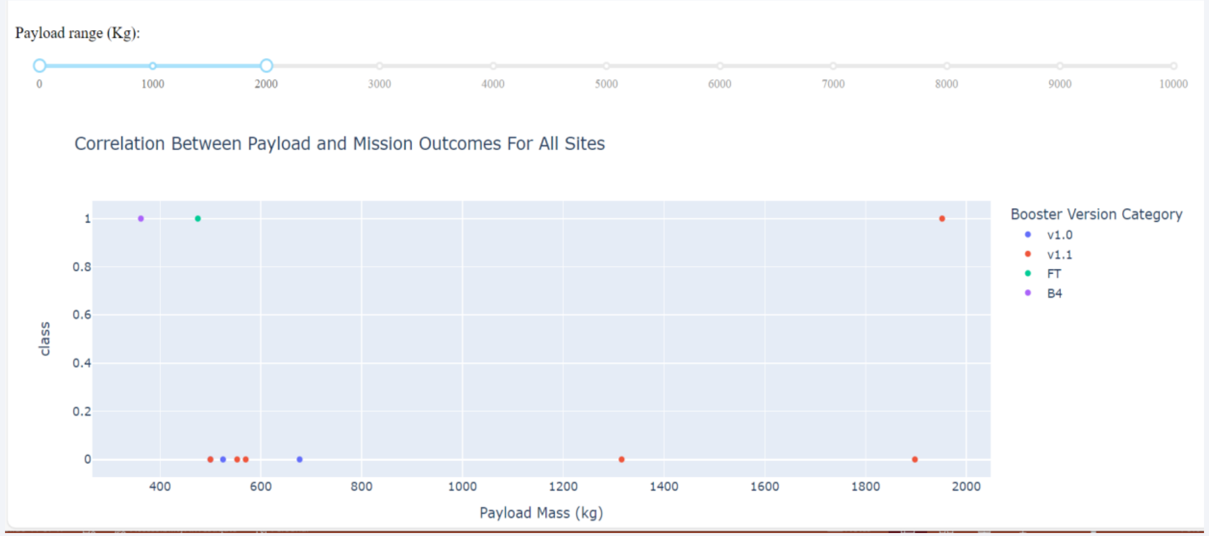
Total Success Launches for Site KSC LC-39A

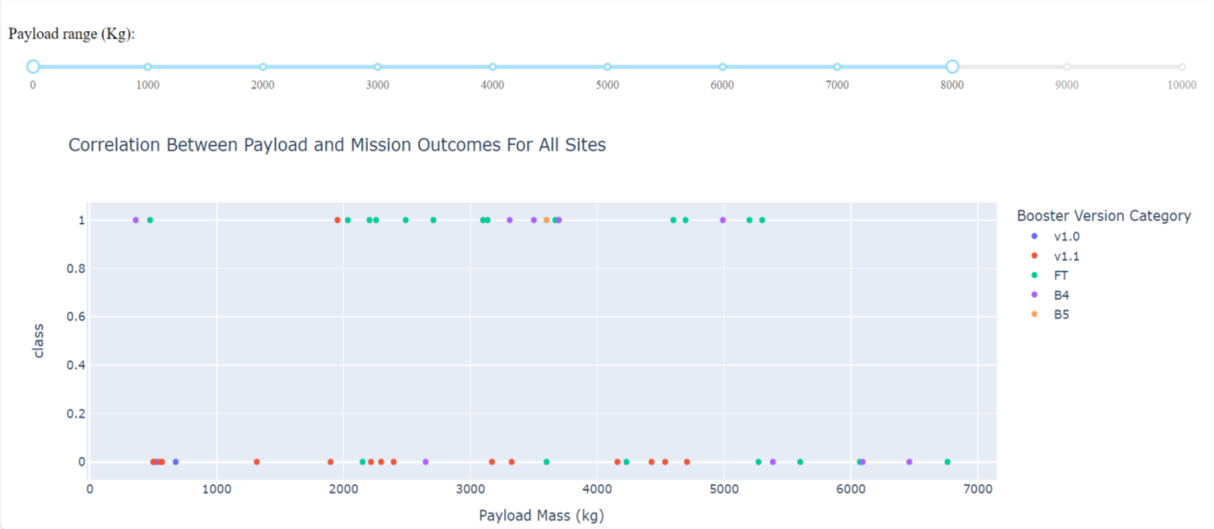
- KSC LC-39A recorded over 75% success

Payload vs. Launch Outcome







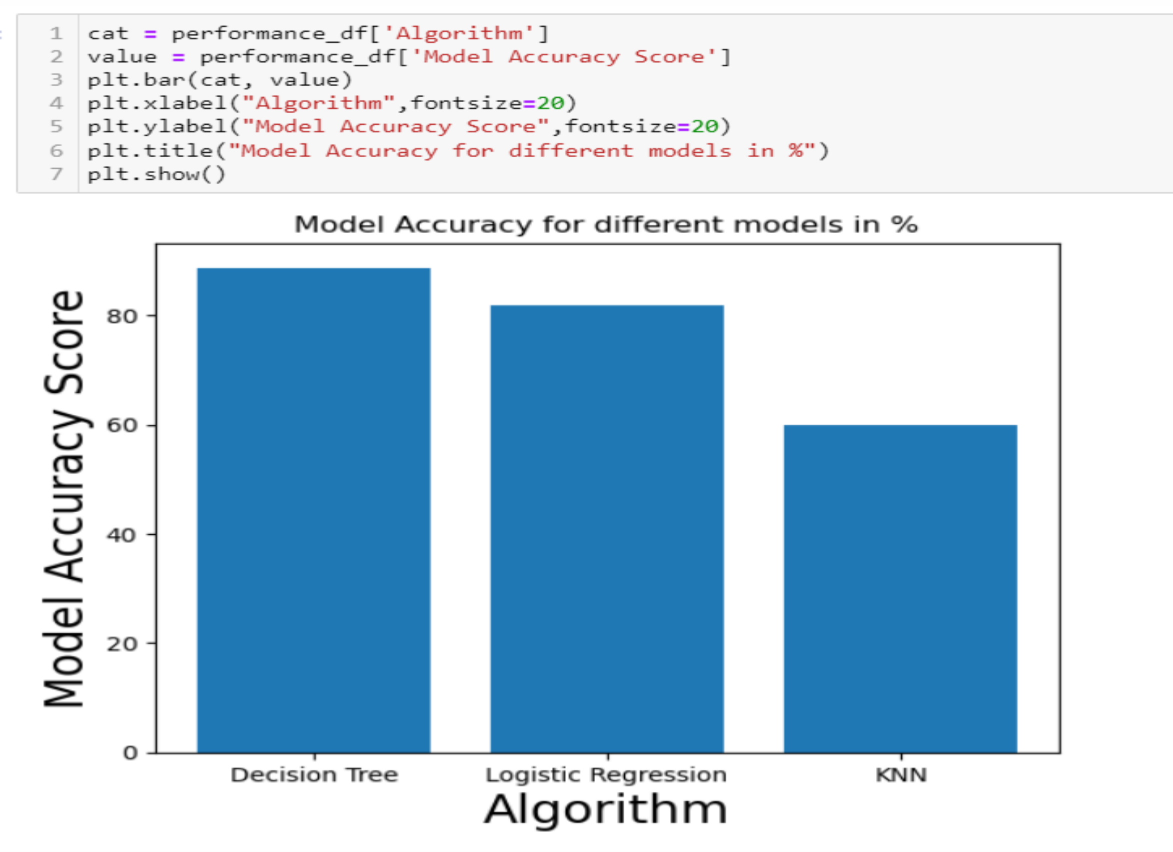


Section 5

Predictive Analysis (Classification)

Classification Accuracy

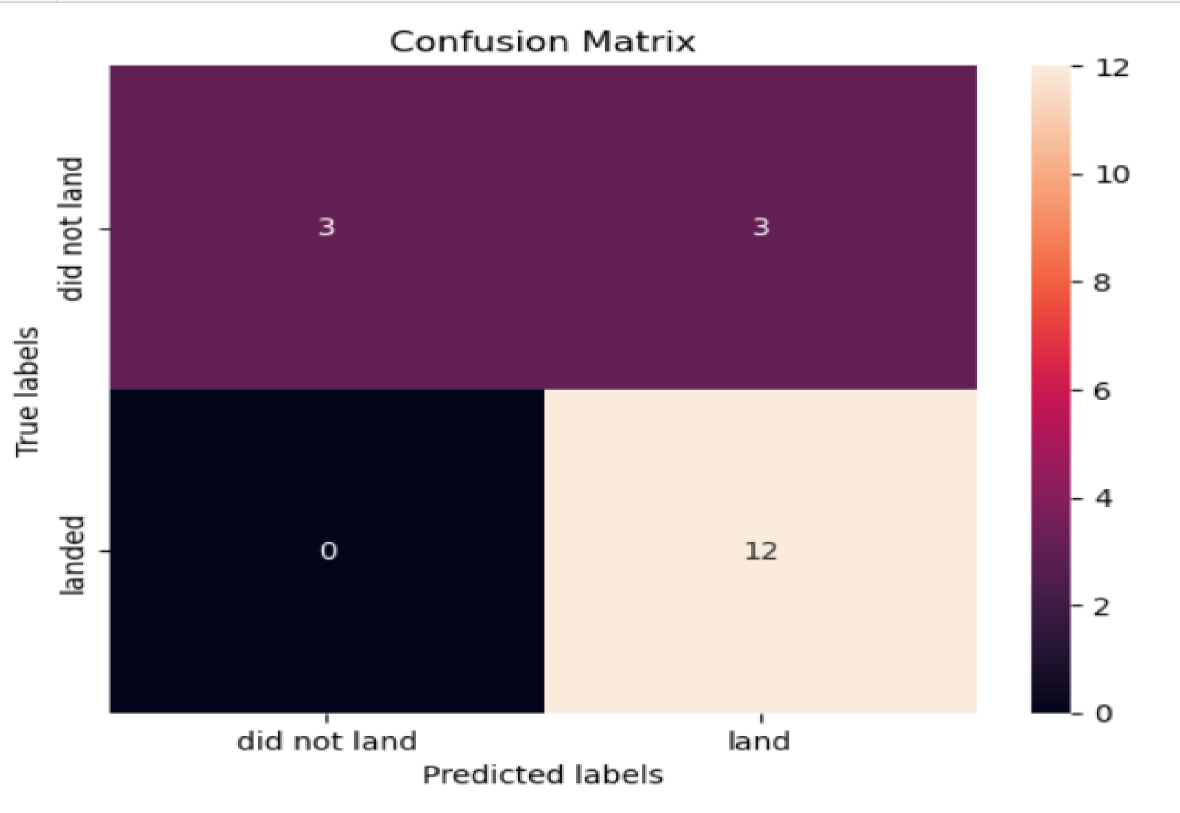
- Decision Tree model has the highest accuracy.



```
cat = performance_df['Algorithm']
value = performance_df['Model Accuracy Score']
plt.bar(cat, value)
plt.xlabel("Algorithm",fontsize=20)
plt.ylabel("Model Accuracy Score",fontsize=20)
plt.title("Model Accuracy for different models in %")
plt.show()
```

Confusion Matrix

- Decision tree model is the best and was able to predict all the land class correct and was only able to predict half of the did not land class correctly. This creates a false negative.

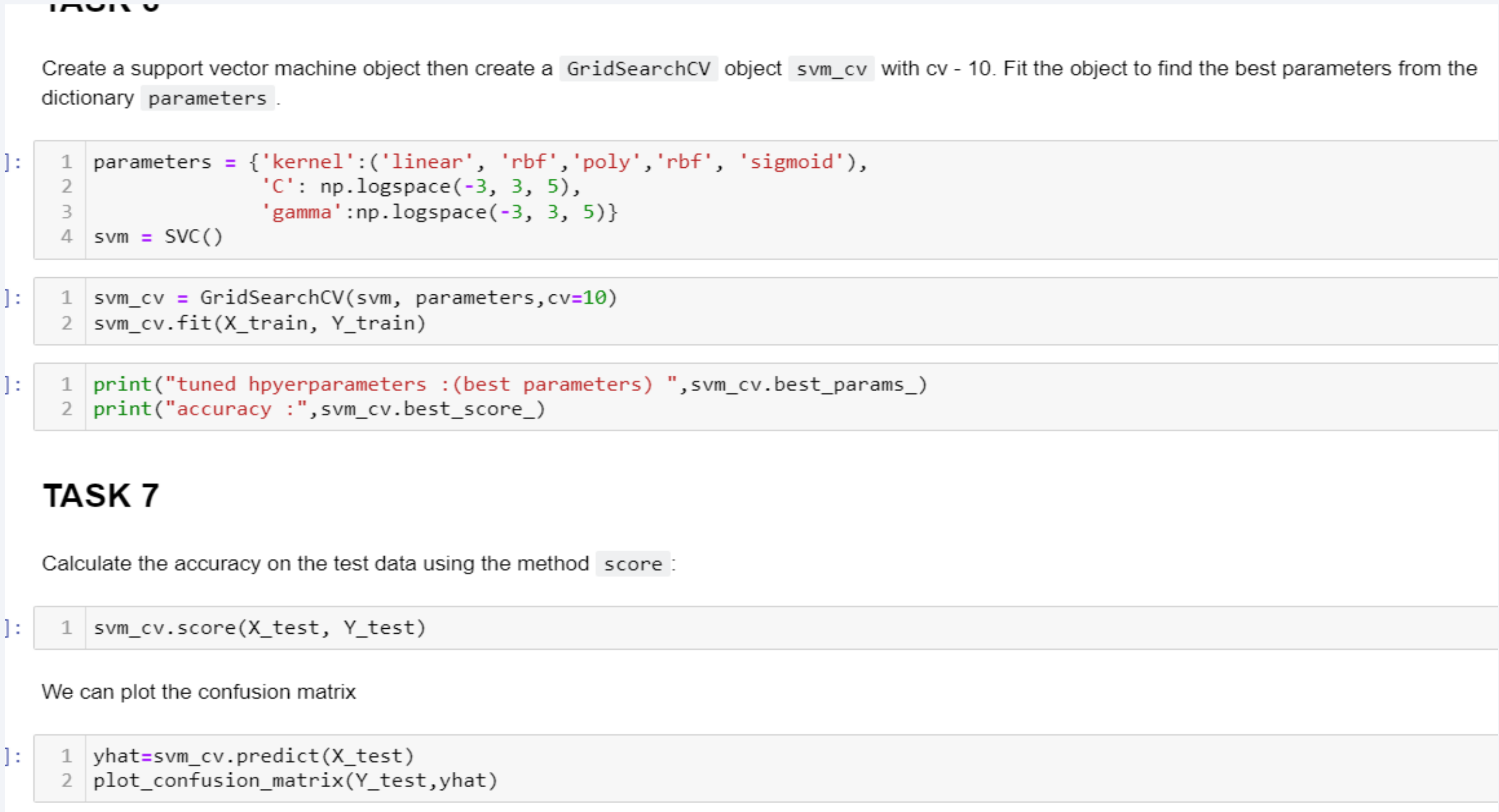


Conclusions

- There are four landing sites in the project and KSC LC 39A has the highest success rate with over 75% success rate
- ESL-1, GEO, HEO AND SSO Orbit all had very high success rate of about 100%
- First successful ground landing date was 22nd of December 2015
- The average payload for F9 v1.1 is 2928.4kg
- The best performing model is decision tree with an accuracy score of over 80%. With 100% correct prediction of successful landing, the model can be deployed. This will help the company in bid decisions.

Appendix

- Code snippet of correct SVM that kept running and produced no output.



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` .

```
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
```

```
svm_cv = GridSearchCV(svm, parameters,cv=10)
svm_cv.fit(X_train, Y_train)
```

```
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)
```

TASK 7

Calculate the accuracy on the test data using the method `score` :

```
svm_cv.score(X_test, Y_test)
```

We can plot the confusion matrix

```
yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

Thank you!