IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Mir Ahmed Ali Shah
29th February 2024

Outline

Executive Summary

- Summary of methodologies
  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result

Introduction

- Project background and context

  SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 62 million dollars; other providers cost 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. We will try to predict if the Falcon 9 first stage will land successfully which can help to decide if an alternate company wants to bid against SpaceX for a rocket launch.

- Problems you want to find answers
  - What factors determine if the rocket will land successfully?
  - The interaction amongst various features that determine the success rate of a successful landing.
  - What operating conditions needs to be in place to ensure a successful landing program?

Section 1

Methodology

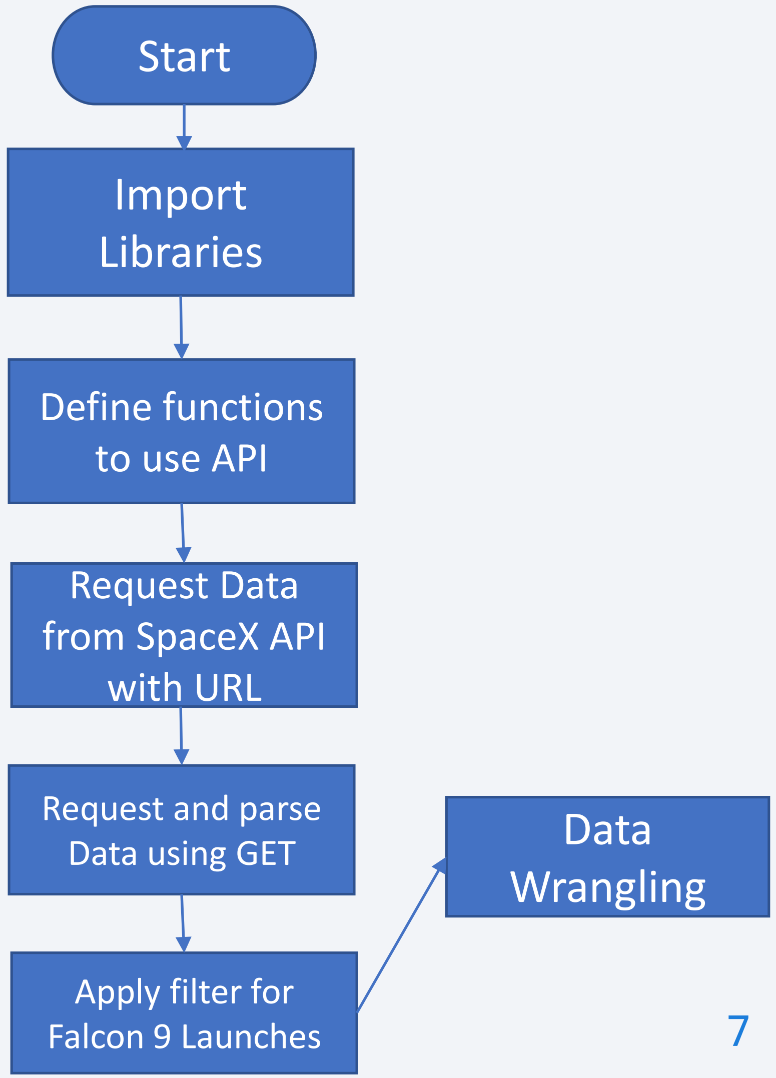
Methodology

Executive Summary

- Data collection methodology:
  - Request to the SpaceX API and clean the requested data
- Perform data wrangling
  - Request and parse the SpaceX launch data using the GET request then filter the dataframe to only include Falcon 9 launches
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

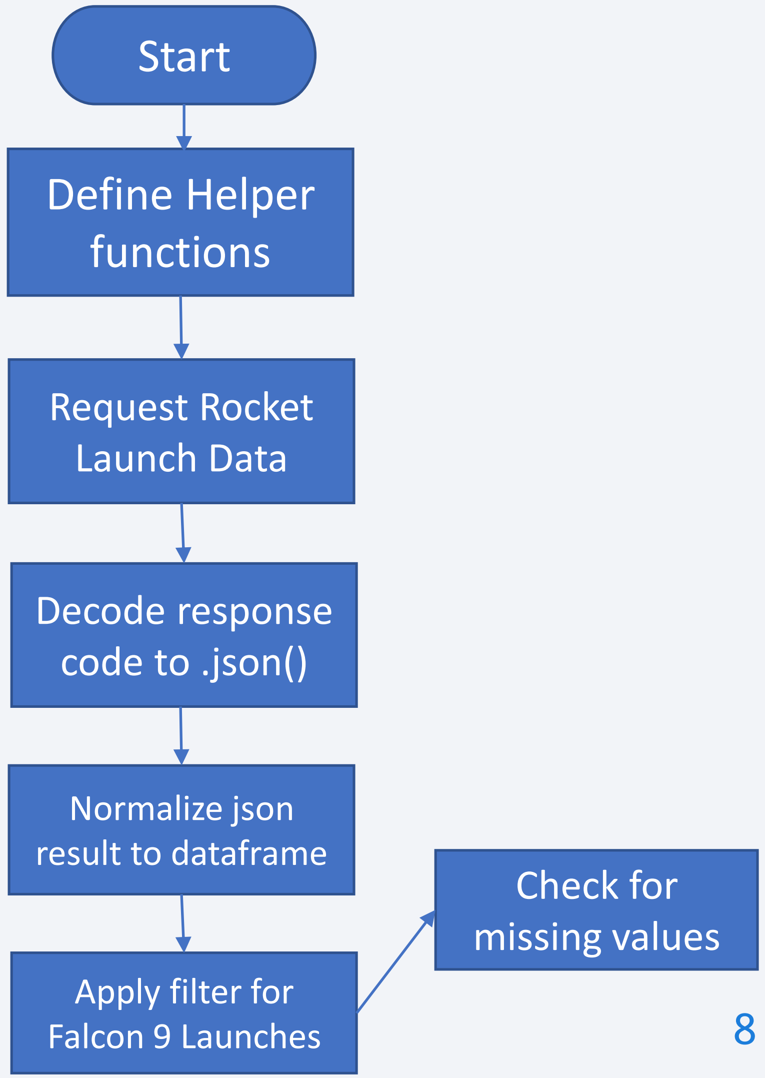
Data Collection

- Import Libraries and Define Auxiliary Functions
- define a series of helper functions that will help us use the API to extract information using identification numbers in the launch data
- Requested rocket launch data from SpaceX API with the following URL: https://api.spacexdata.com/v4/launches/past
- Request and parse the SpaceX launch data using the GET request
- Filter the dataframe to only include Falcon 9 launches


Start
Import Libraries
Define functions to use API
Request Data from SpaceX API with URL
Request and parse Data using GET
Apply filter for Falcon 9 Launches
Data Wrangling

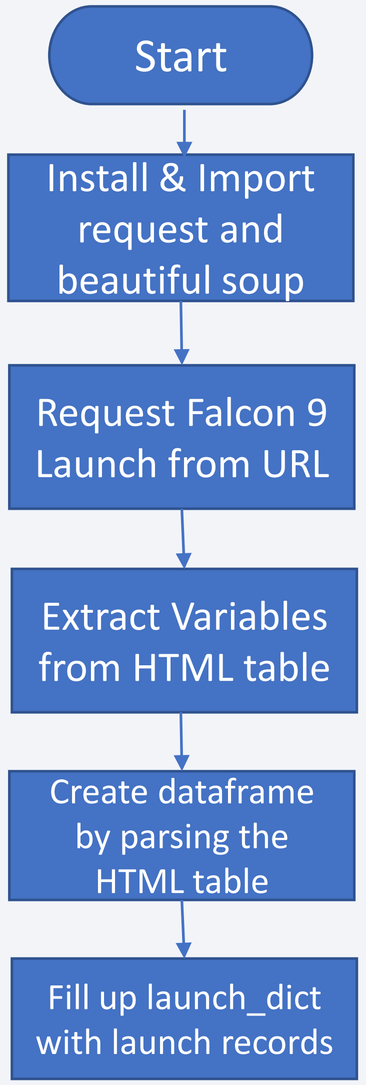
Data Collection - SpaceX API

- Define a series of helper functions that will help use the API to extract information using Identification numbers in launch data.
- Request rocket launch data from Spacex API
- Check content of reponse and make sure status code displays 200
- Decode response code to .json() and turn it into pandas dataframe
- Use json_normalize method to convert the json result into a dataframe
- Call getBoosterVersion
- Call getLaunchSite
- Call getPayloadData
- Call getCoreData
- Filter the dataframe to only include Falcon 9 launches
- https://github.com/ahmed1481/testrepo123/blob/main/jupyter-labs-spacex-data-collection-api.ipynb


Start
Define Helper functions
Request Rocket Launch Data
Decode response code to .json()
Normalize json result to dataframe
Apply filter for Falcon 9 Launches
Check for missing values

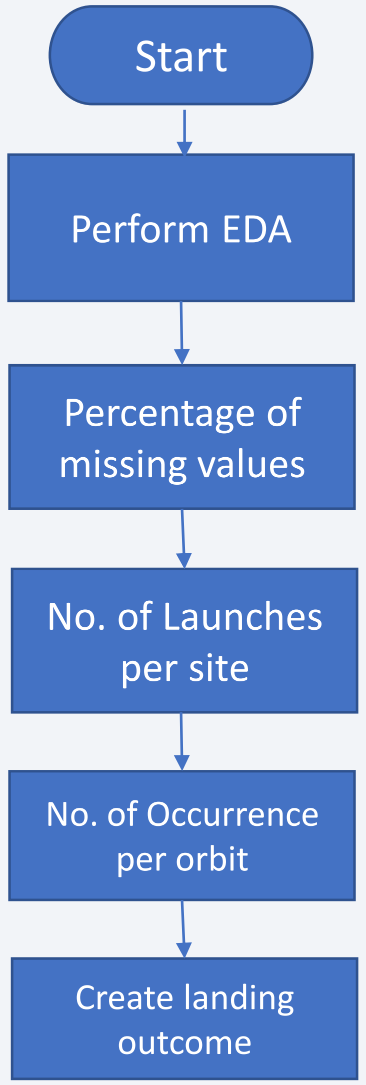
Data Collection - Scraping

- Install and import requests and beautiful soup. Provide some helper functions to process web scraped HTML table
  - def date_time(table_cells):
  - def booster_version(table_cells):
  - def landing_status(table_cells):
- Request the Falcon9 Launch Wiki page from its URL
- Extract all column/variable names from the HTML table header
- Create a dataframe by parsing the launch HTML tables
- https://github.com/ahmed1481/testrepo123/blob/main/jupyter-labs-webscraping.ipynb


Start
Install & Import request and beautiful soup
Request Falcon 9 Launch from URL
Extract Variables from HTML table
Create dataframe by parsing the HTML table
Fill up launch_dict with launch records

Data Wrangling

- Perform EDA and determine Training Labels
- Load SpaceX Dataset
- Identify and calculate the percentage of missing values in each attribute
- Identify which columns are numerical and categorical
- Calculate number of launches on each site
- Calculate the number and occurrence of each orbit
- Calculate number and occurrence of mission outcome of the orbits
- Create a landing outcome label from Outcome column
- https://github.com/ahmed1481/testrepo123/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb



EDA with Data Visualization

- Visualization done to find how the Flight Number and Payload variables would affect the launch outcome. We see that as the flight number increases, the first stage is more likely to land successfully. We see that different launch sites have different success rates
- Visualized the relationship between Flight Number and Launch Site to explain the patterns you found in the scatter point plots
- Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launch site there are no rockets launched for heavy payload mass (greater than 10000)
- Analyzed the plotted bar chart try to find which orbits have high success rate
- Visualize the relationship between Flight Number and Orbit type
- Visualize the relationship between Payload and Orbit type
- Plotted the Payload vs. Orbit scatter point charts to reveal the relationship between Payload and Orbit type
- Plotted a line chart with x axis to be Year and y axis to be average success rate, to get the average launch success trend
- https://github.com/ahmed1481/testrepo123/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL(1)

- SQL Query “%sql select distinct Launch_Site from SPACEXTBL” to display the names of the unique launch sites in the space mission
- SQL Query “%sql select * from SPACEXTBL where Launch_Site like 'CCA%' limit 5” to display 5 records where launch sites begin with the string 'CCA'
- SQL Query “%sql select sum(payload_mass__kg_) from SPACEXTBL WHERE customer = 'NASA (CRS)'” to display the total payload mass carried by boosters launched by NASA (CRS)
- SQL Query “%sql select avg(payload_mass__kg_) from SPACEXTBL WHERE booster_version = 'F9 v1.1'” to display average payload mass carried by booster version F9 v1.1
- SQL Query “%sql select min(DATE) from SPACEXTBL WHERE landing_outcome = 'Success (ground pad)'” to list the date when the first successful landing outcome in ground pad was achieved

EDA with SQL(2)

- SQL Query “%sql select booster_version from SPACEXTBL where landing_outcome = 'Success (drone ship)’ and payload_mass__kg_ between 4000 and 6000” to list the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- SQL Query “%sql select mission_outcome, count(mission_outcome) from SPACEXTBL GROUP BY mission_outcome” to list the total number of successful and failure mission outcomes
- SQL Query “%sql select booster_version, payload_mass__kg_ from SPACEXTBL where payload_mass__kg_ = (select max(payload_mass__kg_) from SPACEXTBL)” to list the names of the booster_versions which have carried the maximum payload mass
- SQL Query “%sql SELECT substr(Date,6,2) as month, DATE,BOOSTER_VERSION, LAUNCH_SITE, [Landing_Outcome] FROM SPACEXTBL where [Landing_Outcome] = 'Failure (drone ship)' and substr(Date,0,5)='2015';” to 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
- SQL Query “%sql select count(landing_outcome), landing_outcome from SPACEXTBL where DATE between '2010-06-04' and '2017-03-20' group by landing_outcome order by count(landing_outcome) desc;” to rank the count of landing outcomes or Success between the date 2010-06-04 and 2017-03-20, in descending order
- https://github.com/ahmed1481/testrepo123/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

To discover many interesting insights related to the launch sites location using folium, in a very interactive way:

- We first created a folium Map object
- Used folium.Circle to add a highlighted circle area with a text label on a specific coordinate
- Created and added folium.Circle and folium.Marker for each launch site on the site map
- Enhanced the map by adding the launch outcomes for each site, and see which sites have high success rates
- Recall that data frame spacex_df has detailed launch records, and the class column indicates if this launch was successful or not
- Created a new column in launch_sites dataframe called marker_color to store the marker colors based on the class value
- For each launch result in spacex_df data frame, add a folium.Marker to marker_cluster
- Mark down a point on the closest coastline using MousePosition and calculate the distance between the coastline point and the launch site. After obtained its coordinate, create a folium.Marker to show the distance
- Drawed a PolyLine between a launch site to its closest city, railway, highway, etc. You need to use MousePosition to find their coordinates on the map first

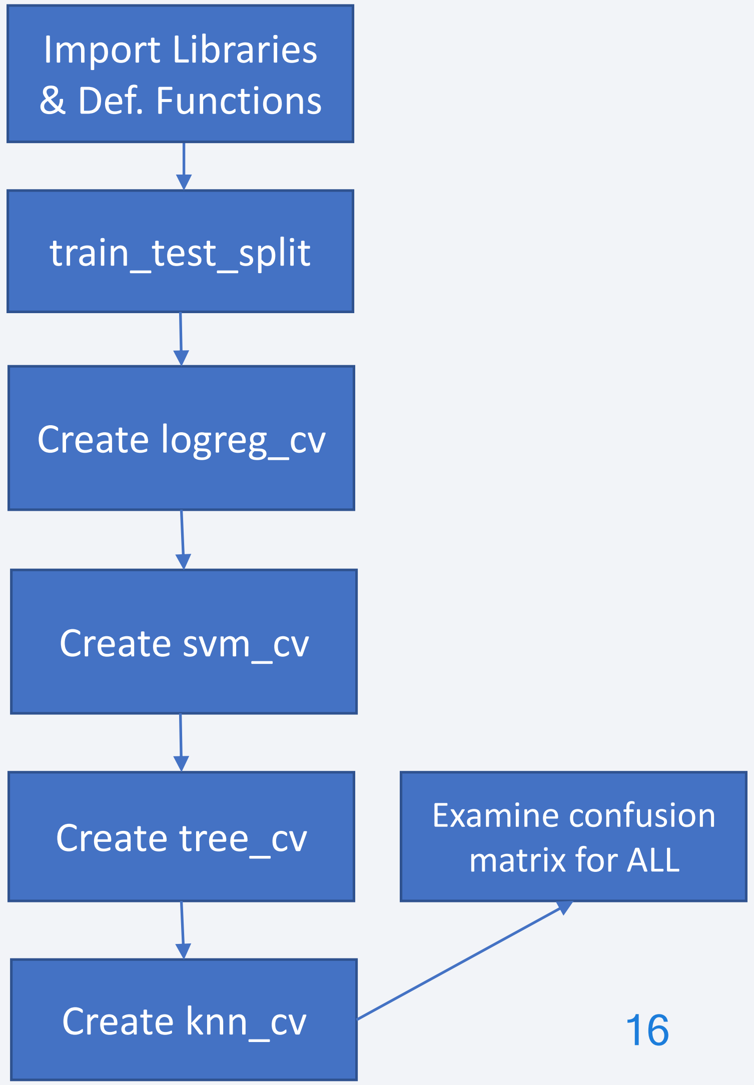
https://github.com/ahmed1481/testrepo123/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- 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.
- Added a Launch Site Drop-down Input Component
- Added a callback function to render success-pie-chart based on selected site dropdown
- Added a Range Slider to Select Payload
- Added a callback function to render the success-payload-scatter-chart scatter plot
- https://github.com/ahmed1481/testrepo123/blob/main/spacex_dash_app.py
- https://ahmedmir-8050.theiadockernext-1-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/

Predictive Analysis (Classification)

- Import Libraries and Define Auxiliary Functions.
- We split the data into training and testing data using the function train_test_split. The training data is divided into validation data, a second set used for training data; then the models are trained and hyperparameters are selected using the function GridSearchCV.
- Fit the object to find the best parameters from the dictionary parameters for all models.
- Create a logistic regression object then create a GridSearchCV object logreg_cv with cv = 10. Calculate the accuracy on the test data using the method score:
- Create a support vector machine object then create a GridSearchCV object svm_cv with cv - 10. Calculate the accuracy on the test data using the method score:
- Create a decision tree classifier object then create a GridSearchCV object tree_cv with cv = 10. Calculate the accuracy of tree_cv on the test data using the method score:
- Create a k nearest neighbors object then create a GridSearchCV object knn_cv with cv = 10. Calculate the accuracy of knn_cv on the test data using the method score:
- Examining the confusion matrix, we Find the method performs best.
- Add the GitHub URL of your completed predictive analysis lab, as an external reference and peer-review purpose

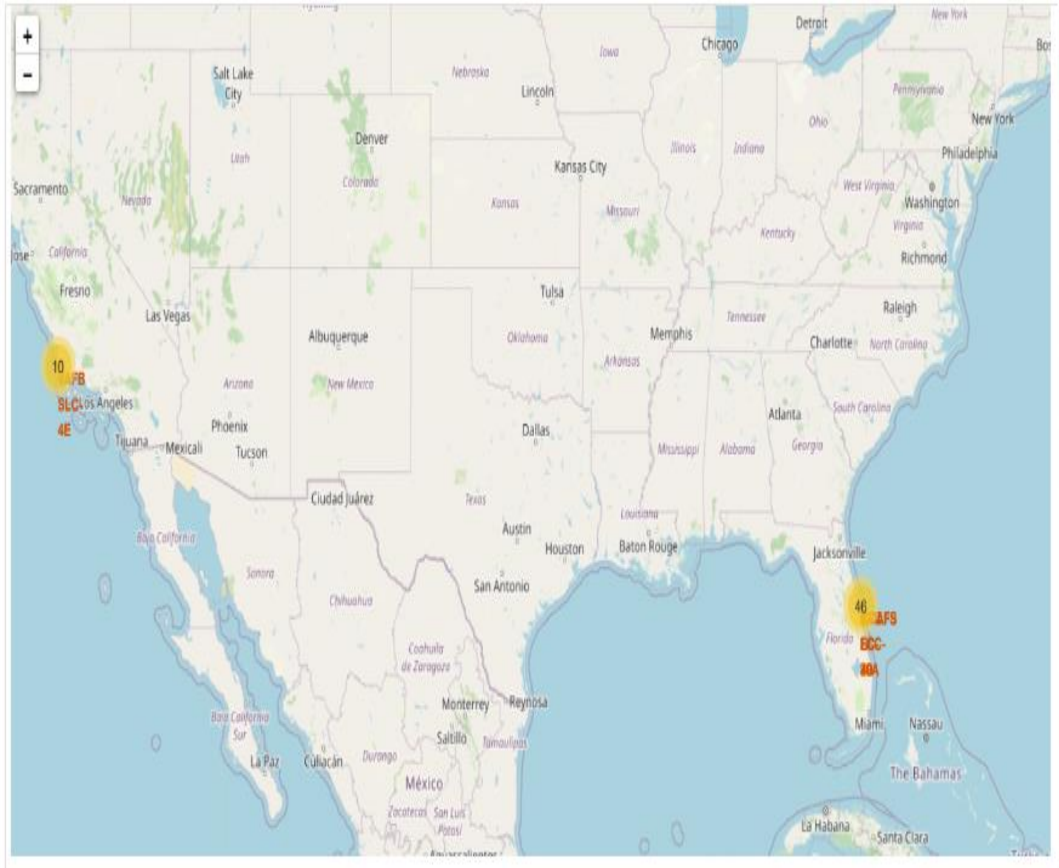


Results

- Exploratory data analysis results
  - We see that as the flight number increases, the first stage is more likely to land successfully
  - The payload mass is also important; it seems the more massive the payload, the less likely the first stage will return
  - We see that different launch sites have different success rates. CCAFS LC-40, has a success rate of 60 %, while KSC LC-39A and VAFB SLC 4E has a success rate of 77%
  - If you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000)
  - We see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit
  - With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS
  - We can observe that the success rate since 2013 kept increasing till 2020

Results

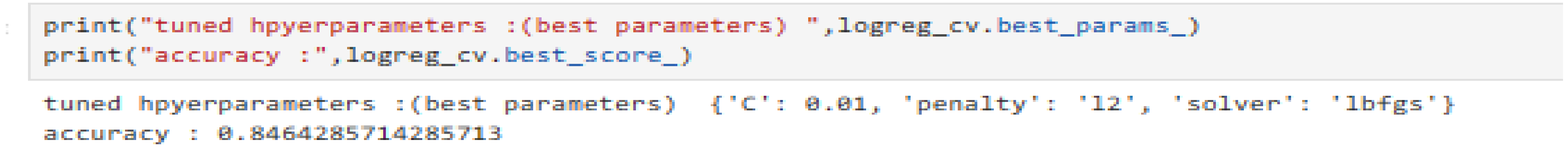
- Interactive analytics demo in screenshots



CCAFS SLC-40
26
CCAFS LC-40

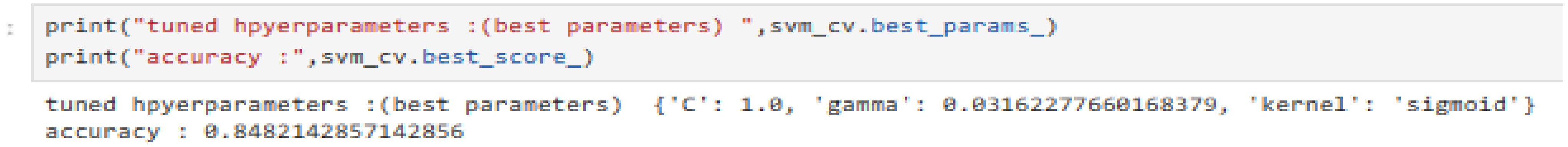
Results

- Predictive analysis results



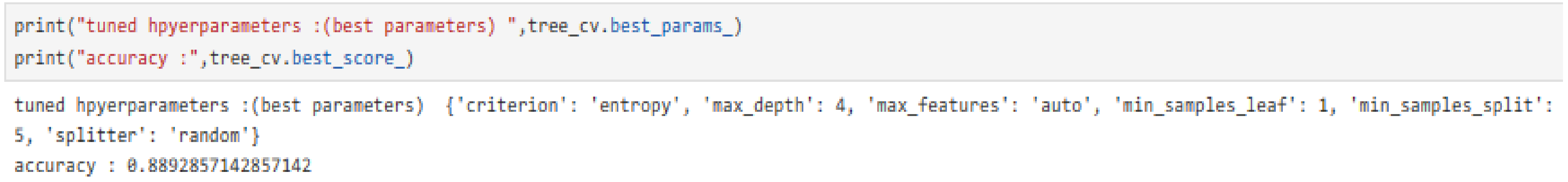
```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("accuracy :",logreg_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8464285714285713
```



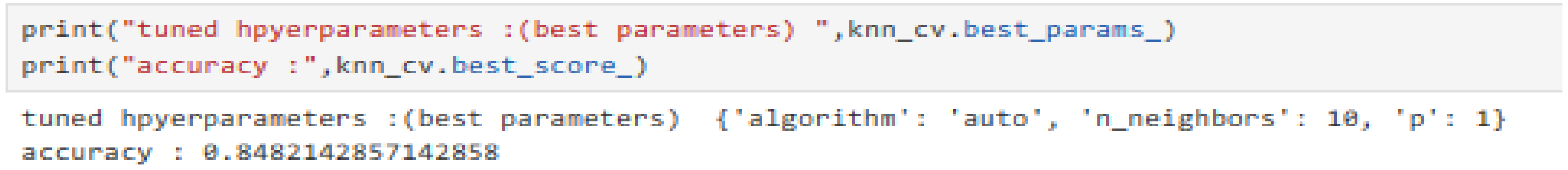
```
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8482142857142856
```



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

```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 4, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 
5, 'splitter': 'random'}
accuracy : 0.8892857142857142
```



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

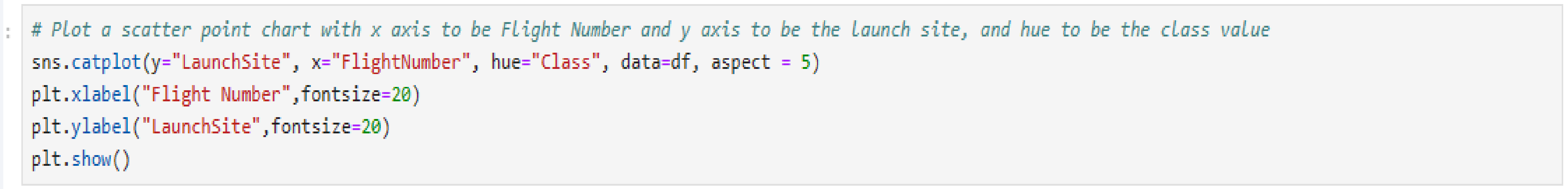
```
tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
accuracy : 0.8482142857142858
```

Section 2

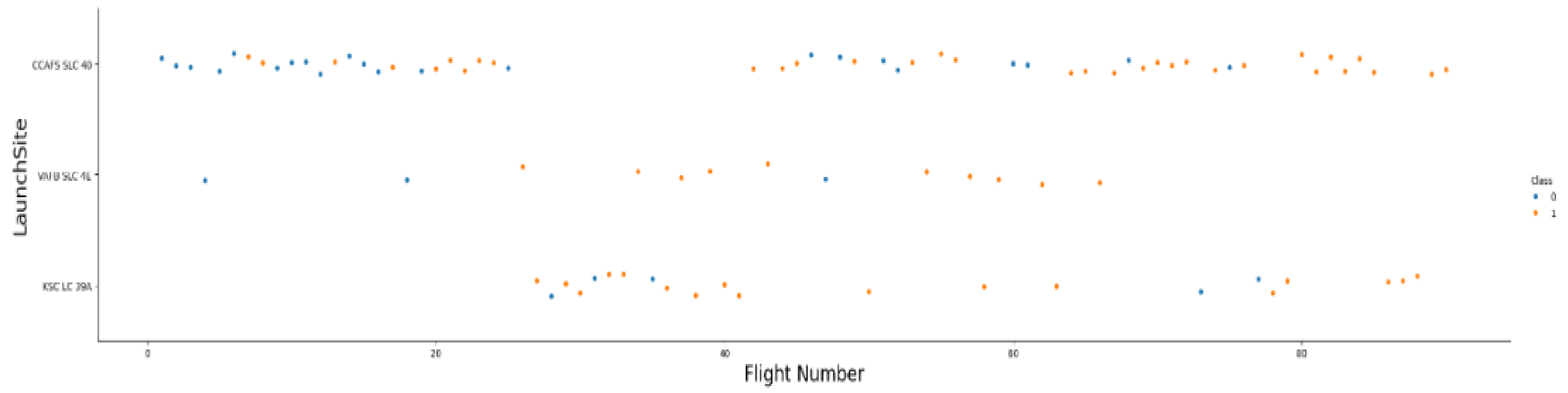
Insights drawn from EDA

Flight Number vs. Launch Site

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

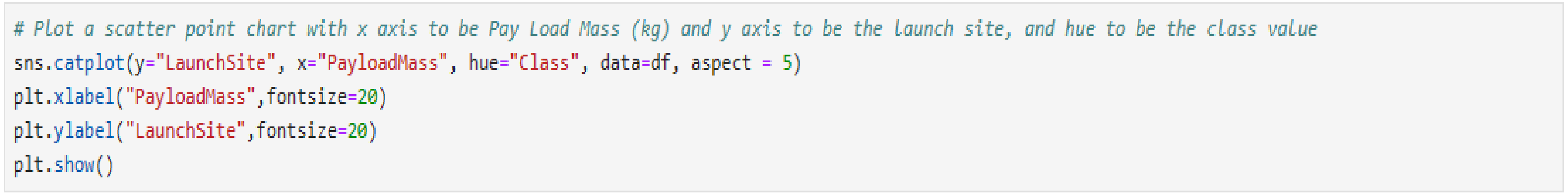


```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.show()
```

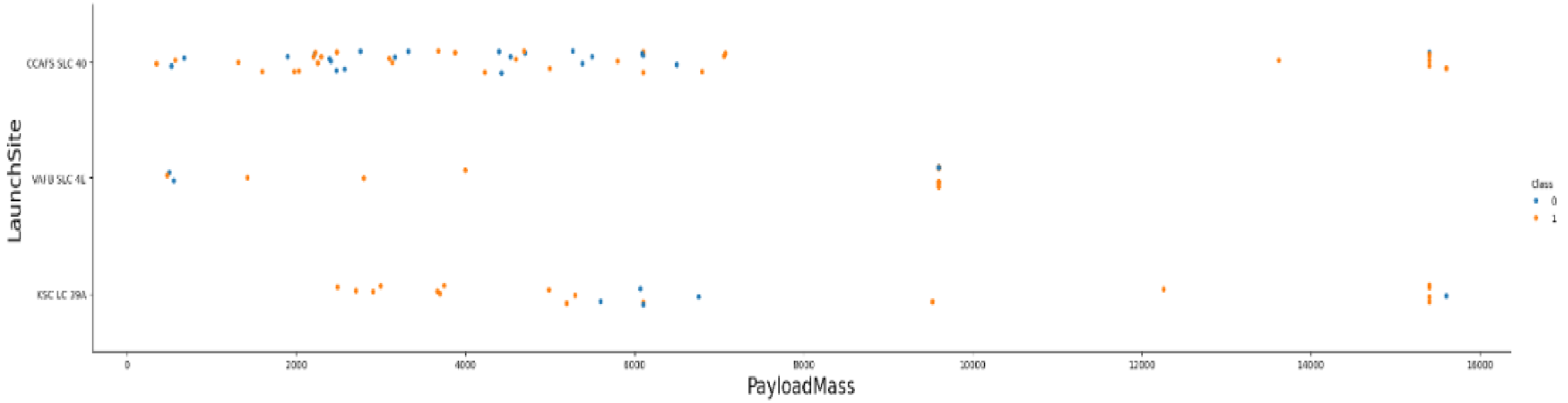


Payload vs. Launch Site

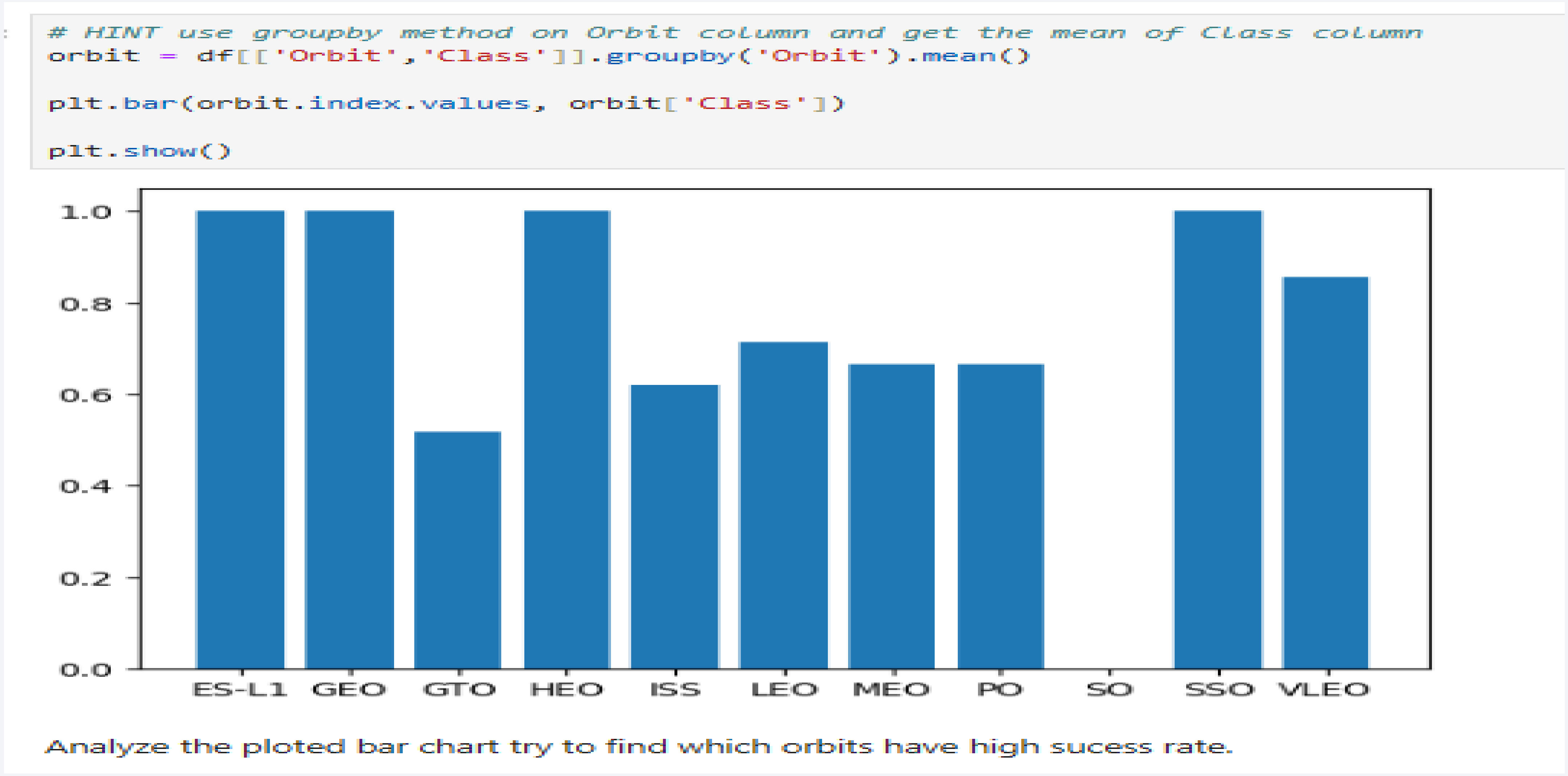
- Now if you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).



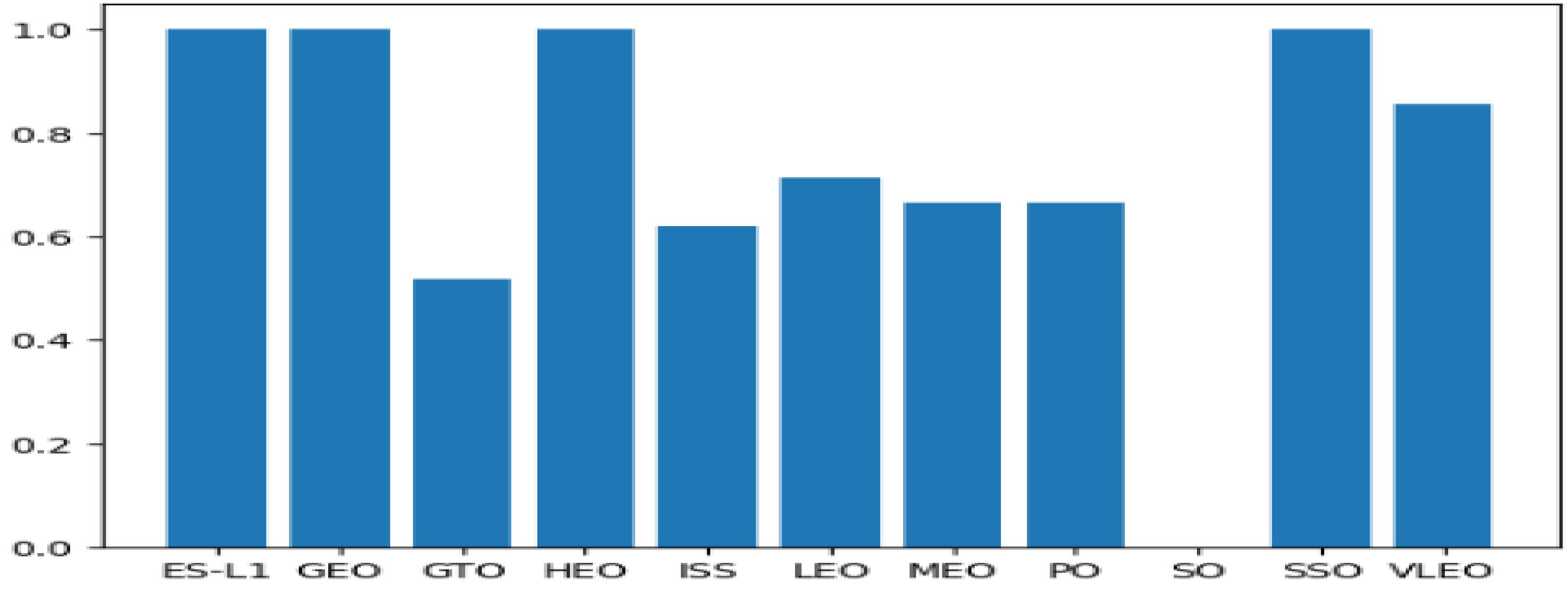
```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("PayloadMass",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.show()
```



Success Rate vs. Orbit Type



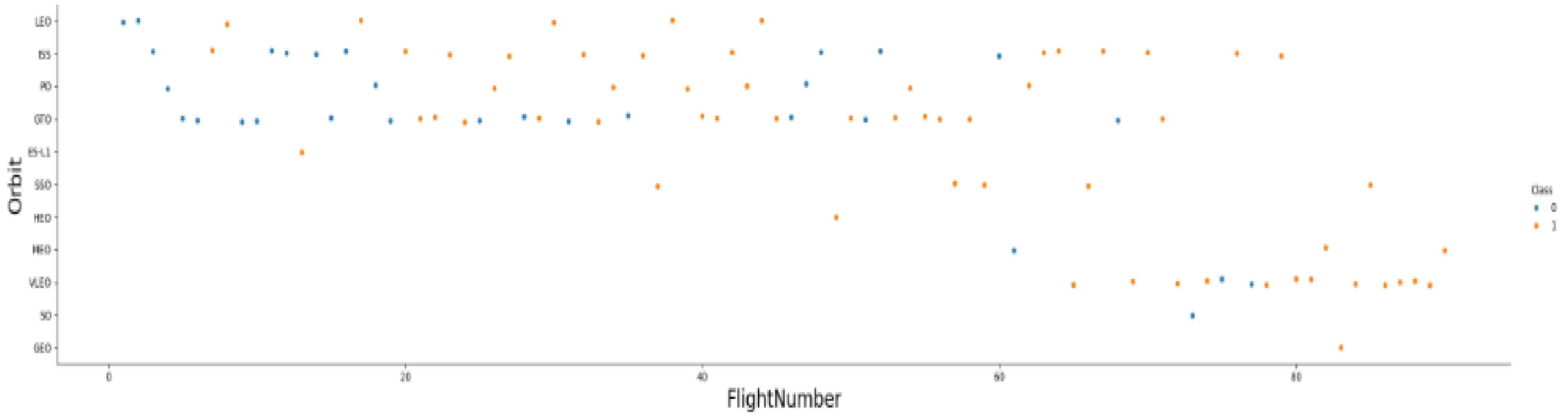
```
# HINT use groupby method on Orbit column and get the mean of Class column
orbit = df[['Orbit','Class']].groupby('Orbit').mean()

plt.bar(orbit.index.values, orbit['Class'])

plt.show()
```



Analyze the ploted bar chart try to find which orbits have high sucess rate.

Flight Number vs. Orbit Type

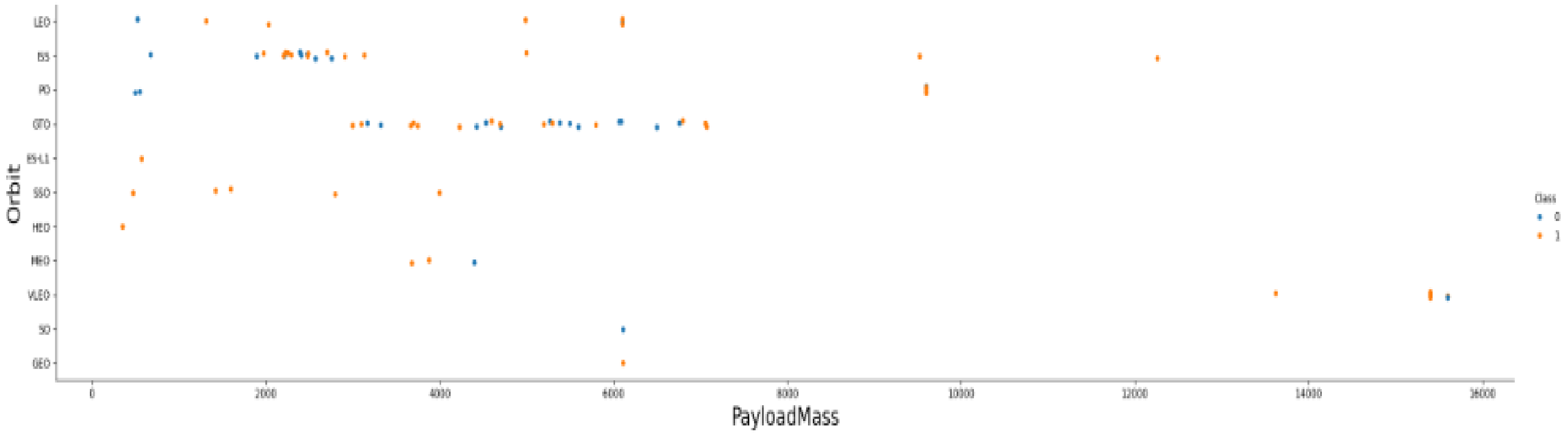
```python
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("FlightNumber",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

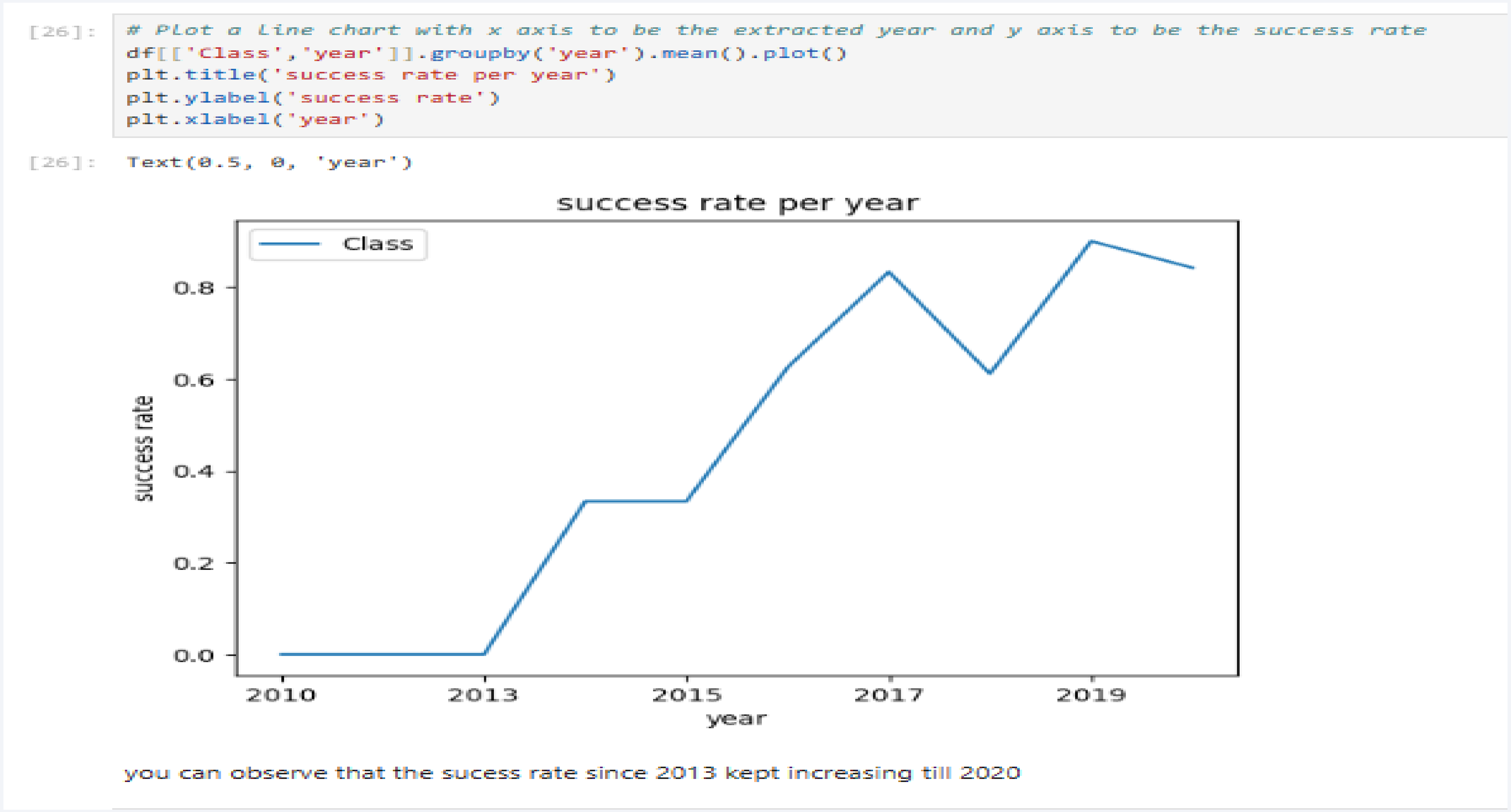
Payload vs. Orbit Type

```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("PayloadMass",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

Launch Success Yearly Trend

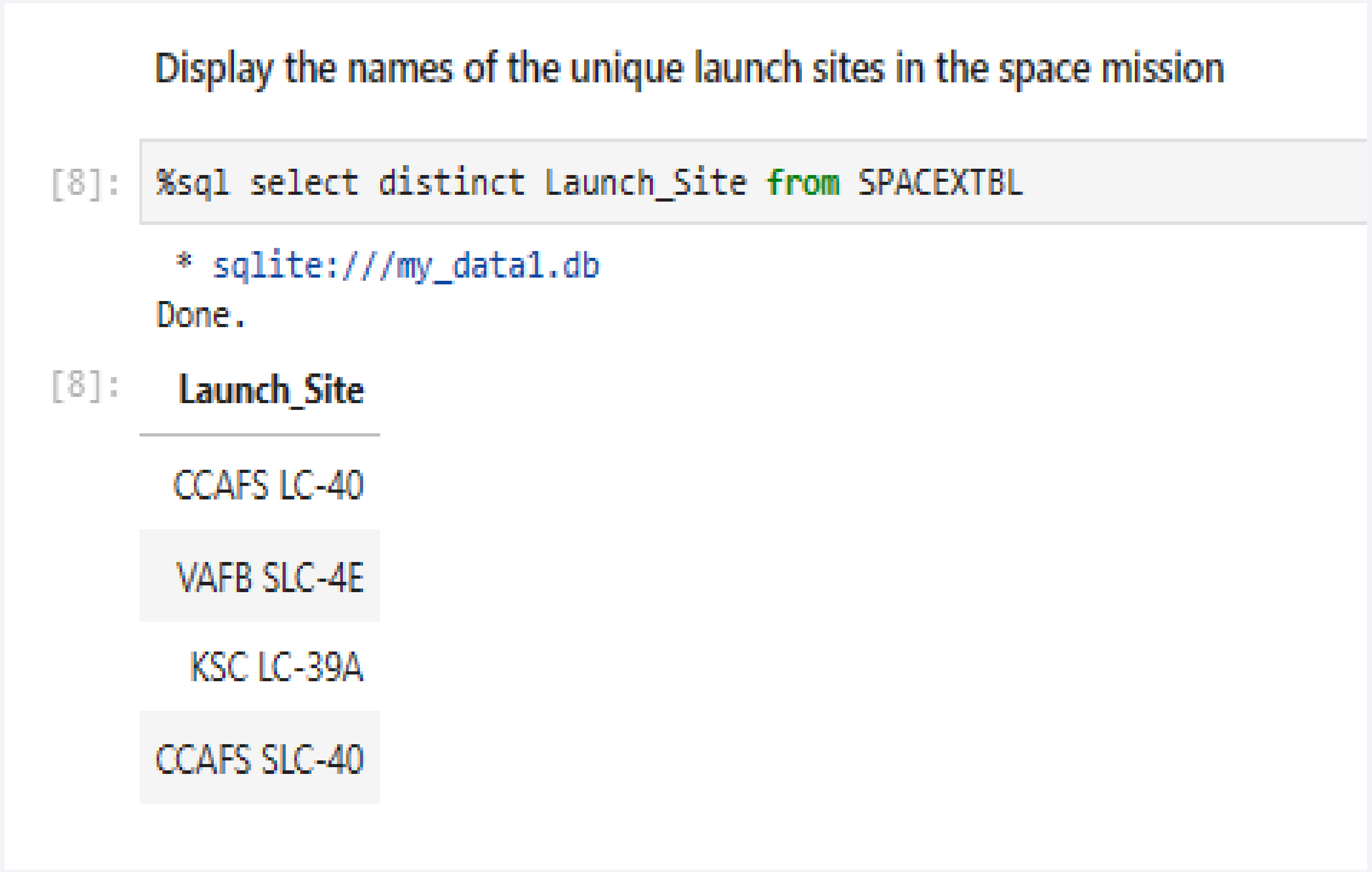


```
[26]:  # Plot a line chart with x axis to be the extracted year and y axis to be the success rate
       df[['Class','year']].groupby('year').mean().plot()
       plt.title('success rate per year')
       plt.ylabel('success rate')
       plt.xlabel('year')
```

```
[26]:  Text(0.5, 0, 'year')
```

you can observe that the sucess rate since 2013 kept increasing till 2020

All Launch Site Names



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

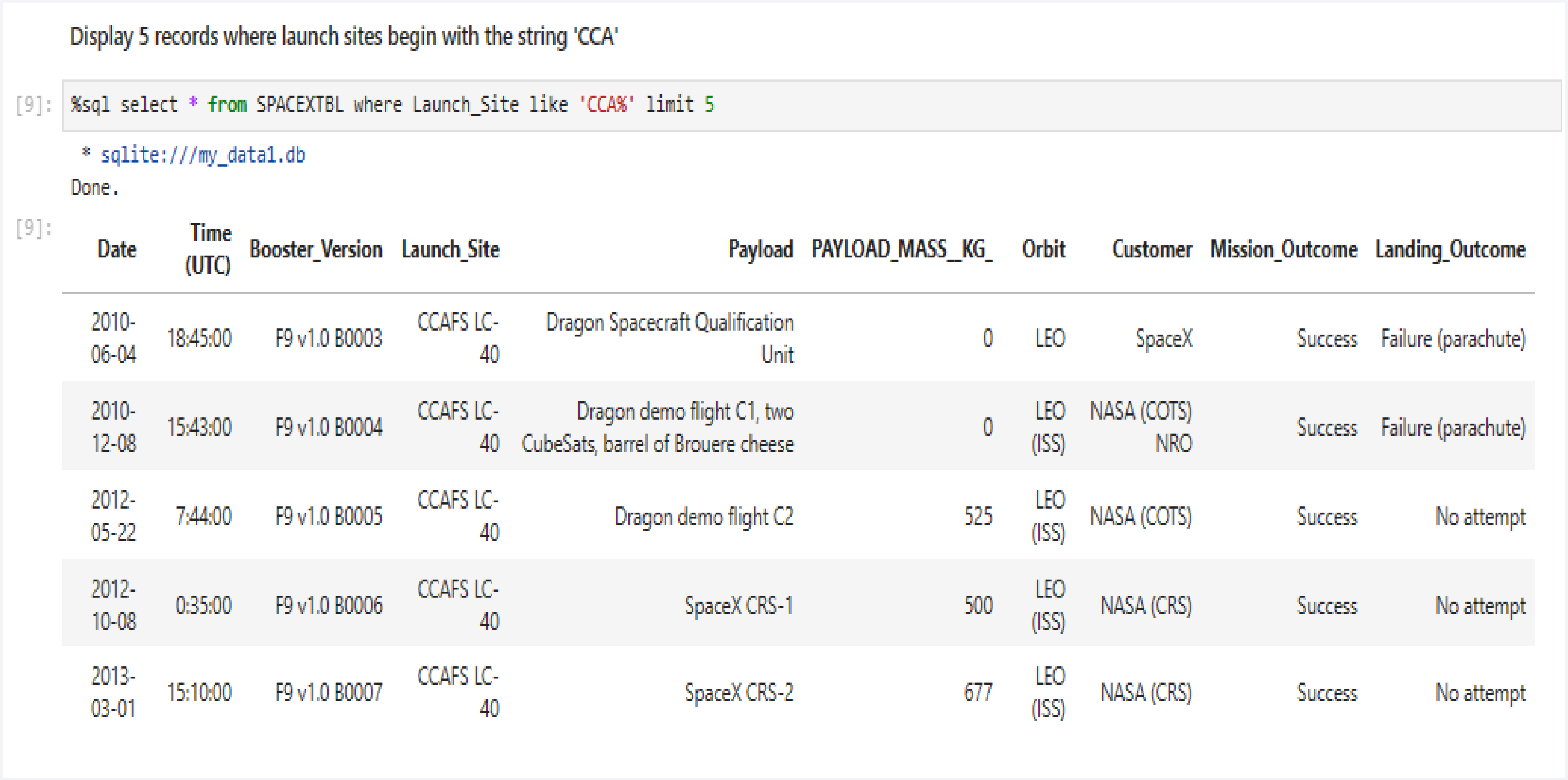
```
[8]: %sql select distinct Launch_Site from SPACEXTBL
```

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

[8]:

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

Launch Site Names Begin with 'CCA'



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

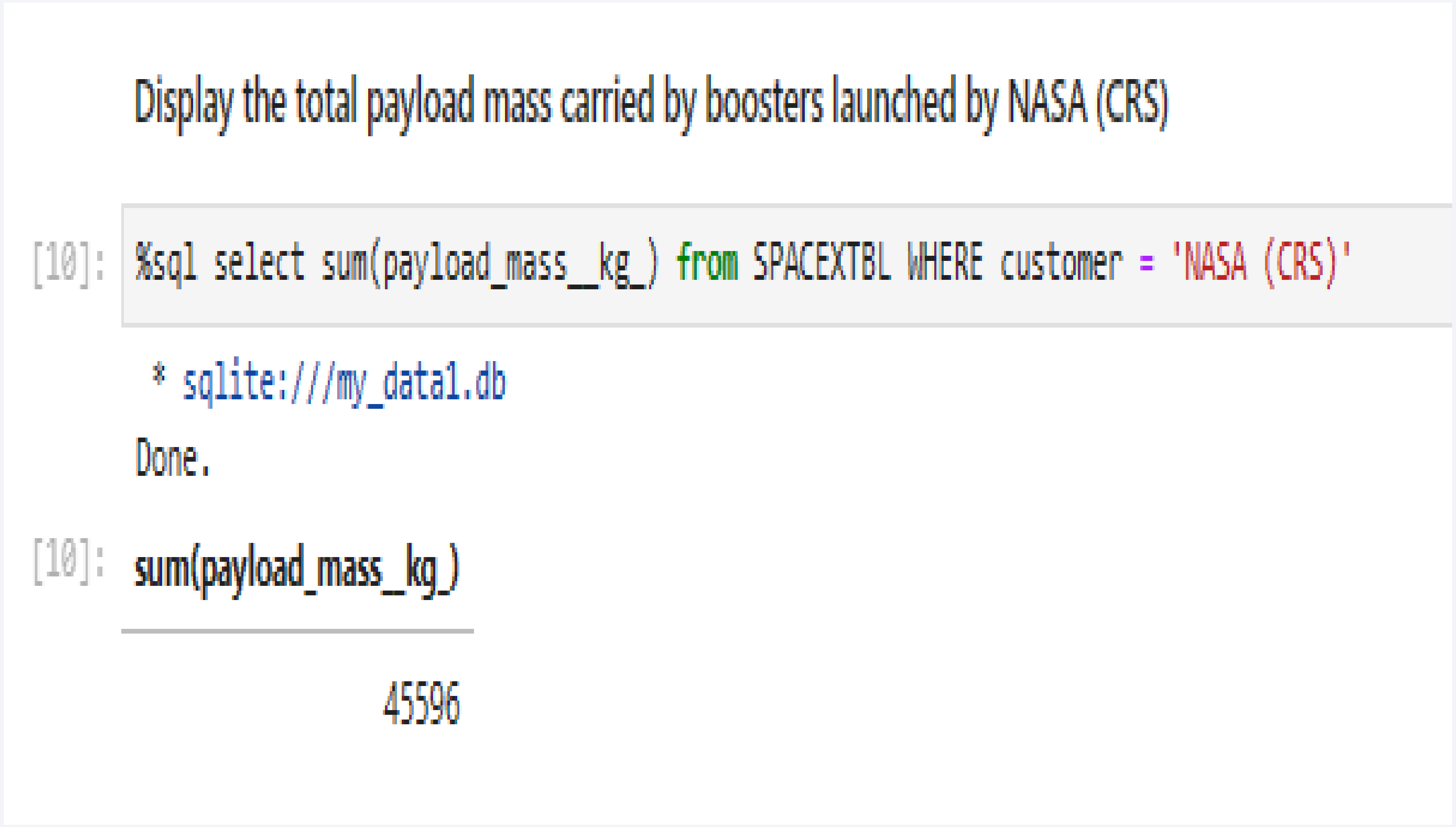
```
[9]: %sql select * from SPACEXTBL where Launch_Site like 'CCA%' limit 5
```

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

[9]:

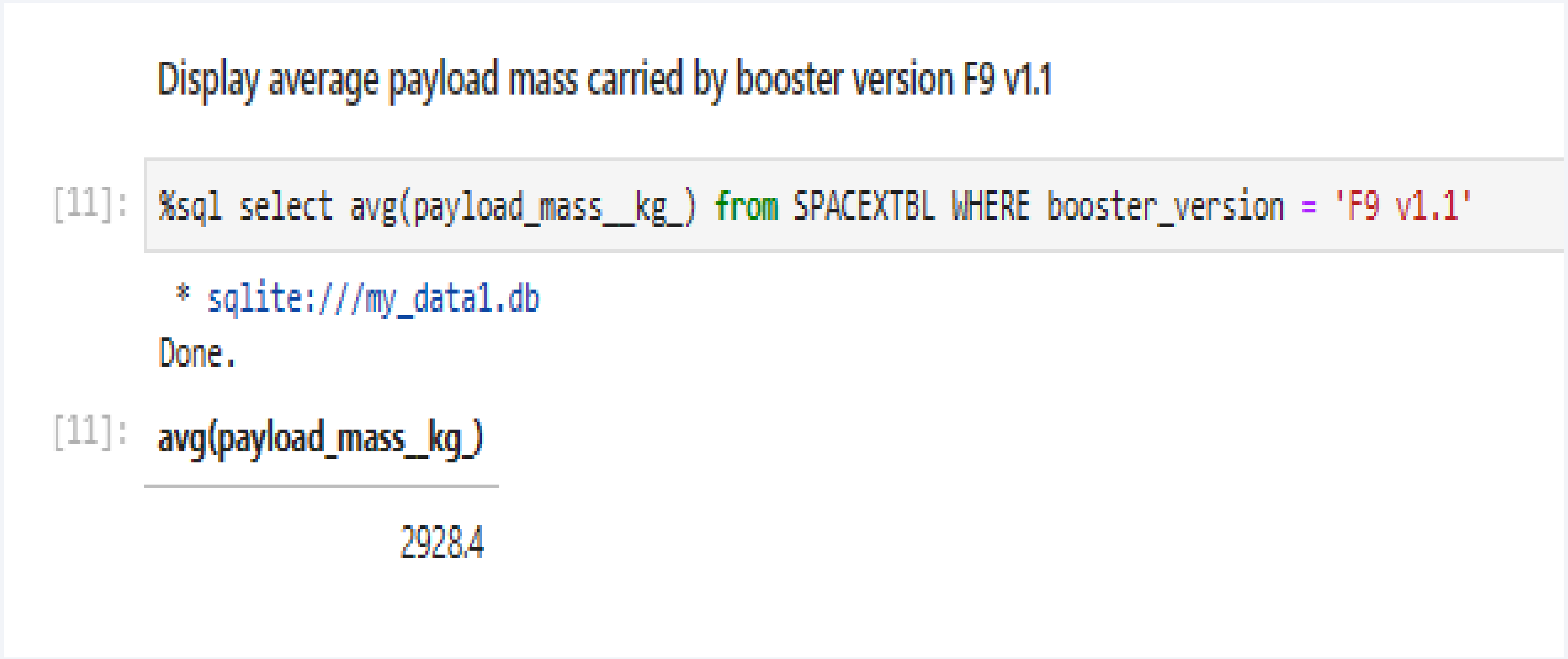
Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight 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


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

```
[10]: %sql select sum(payload_mass__kg_) from SPACEXTBL WHERE customer = 'NASA (CRS)'

 * sqlite:///my_data1.db
Done.

[10]: sum(payload_mass__kg_)
                      45596
```

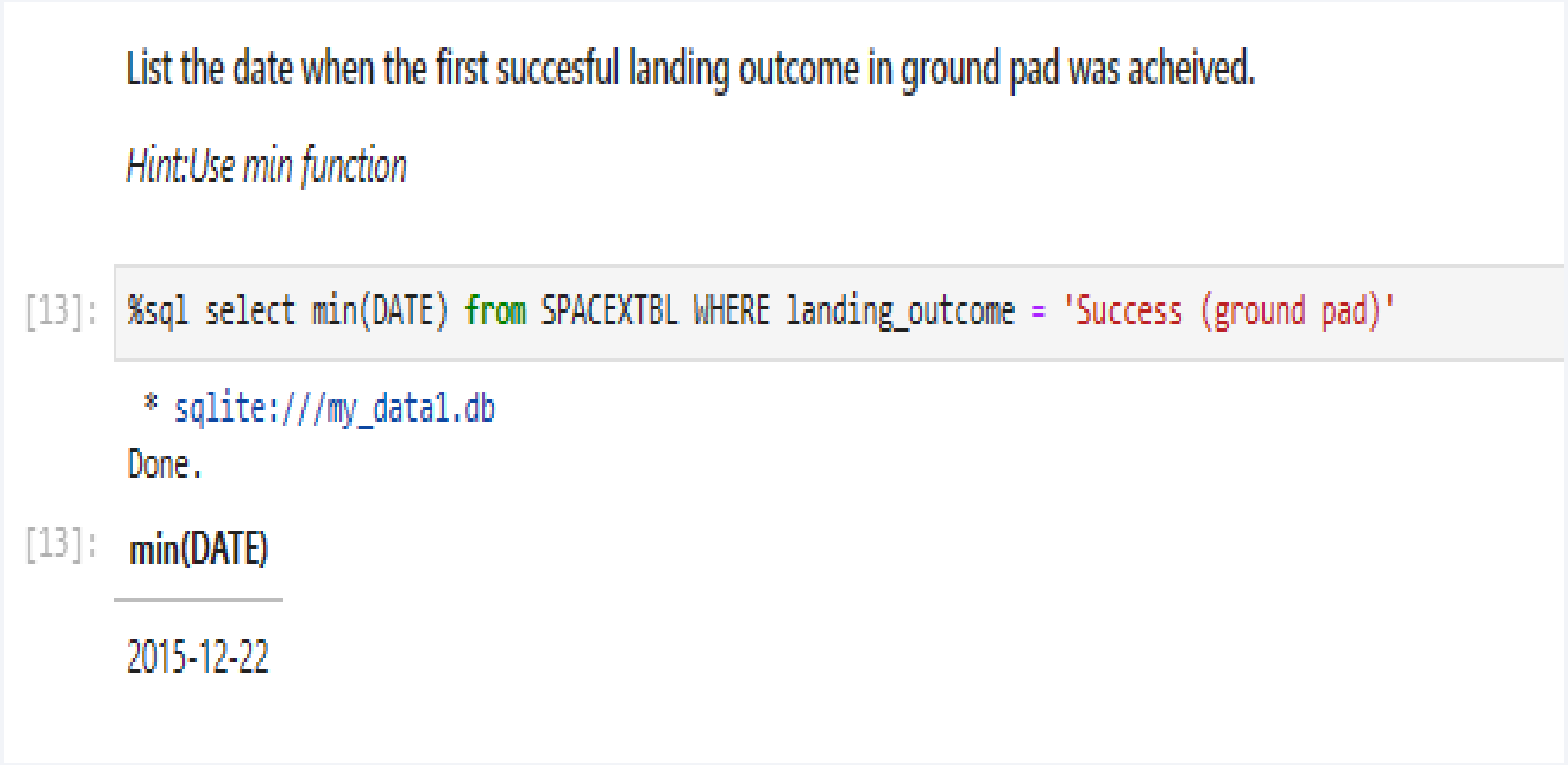
Average Payload Mass by F9 v1.1



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

```
[11]: %sql select avg(payload_mass__kg_) from SPACEXTBL WHERE booster_version = 'F9 v1.1'

 * sqlite:///my_data1.db
Done.

[11]: avg(payload_mass__kg_)
                    2928.4
```

First Successful Ground Landing Date

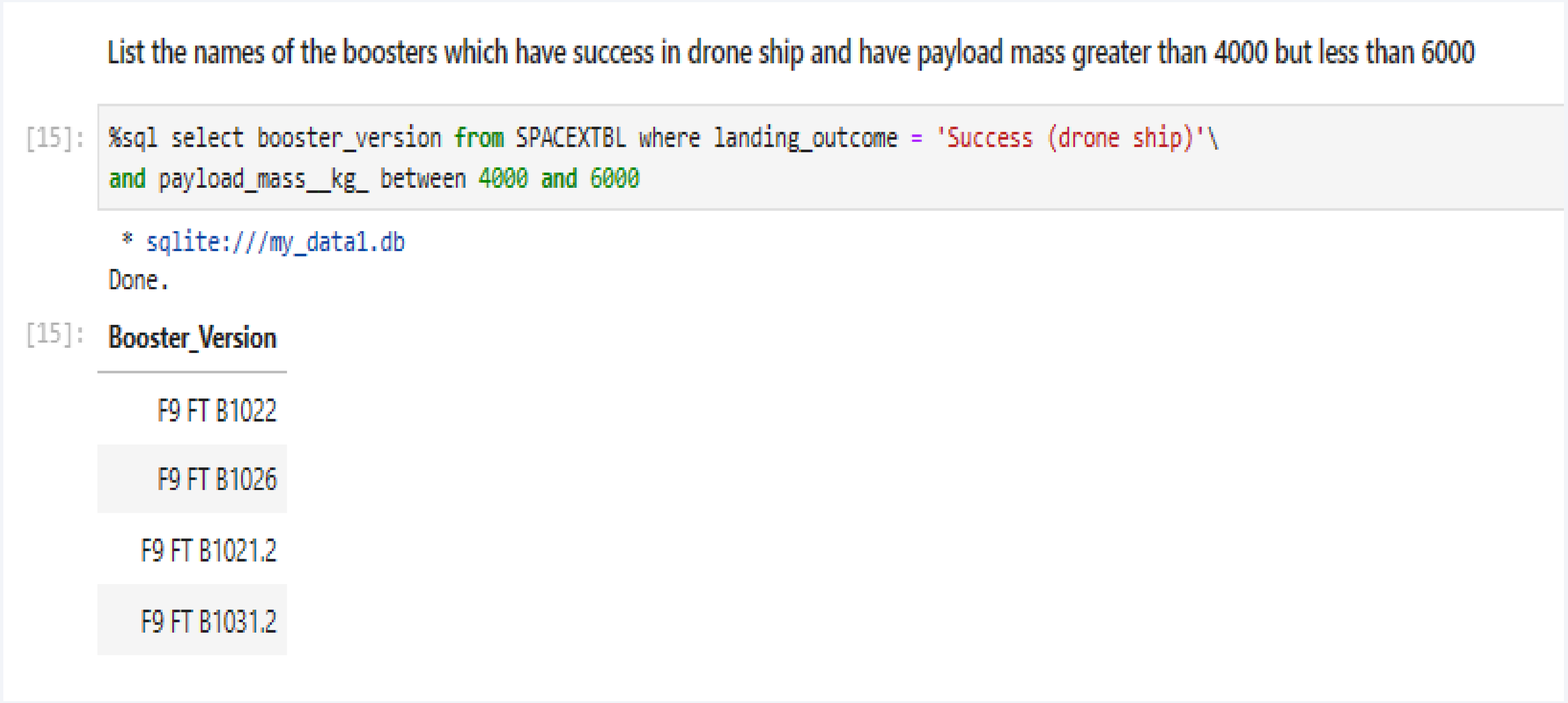


List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

```
[13]: %sql select min(DATE) from SPACEXTBL WHERE landing_outcome = 'Success (ground pad)'

 * sqlite:///my_data1.db
Done.

[13]: min(DATE)
      2015-12-22
```

Successful Drone Ship Landing with Payload between 4000 and 6000



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

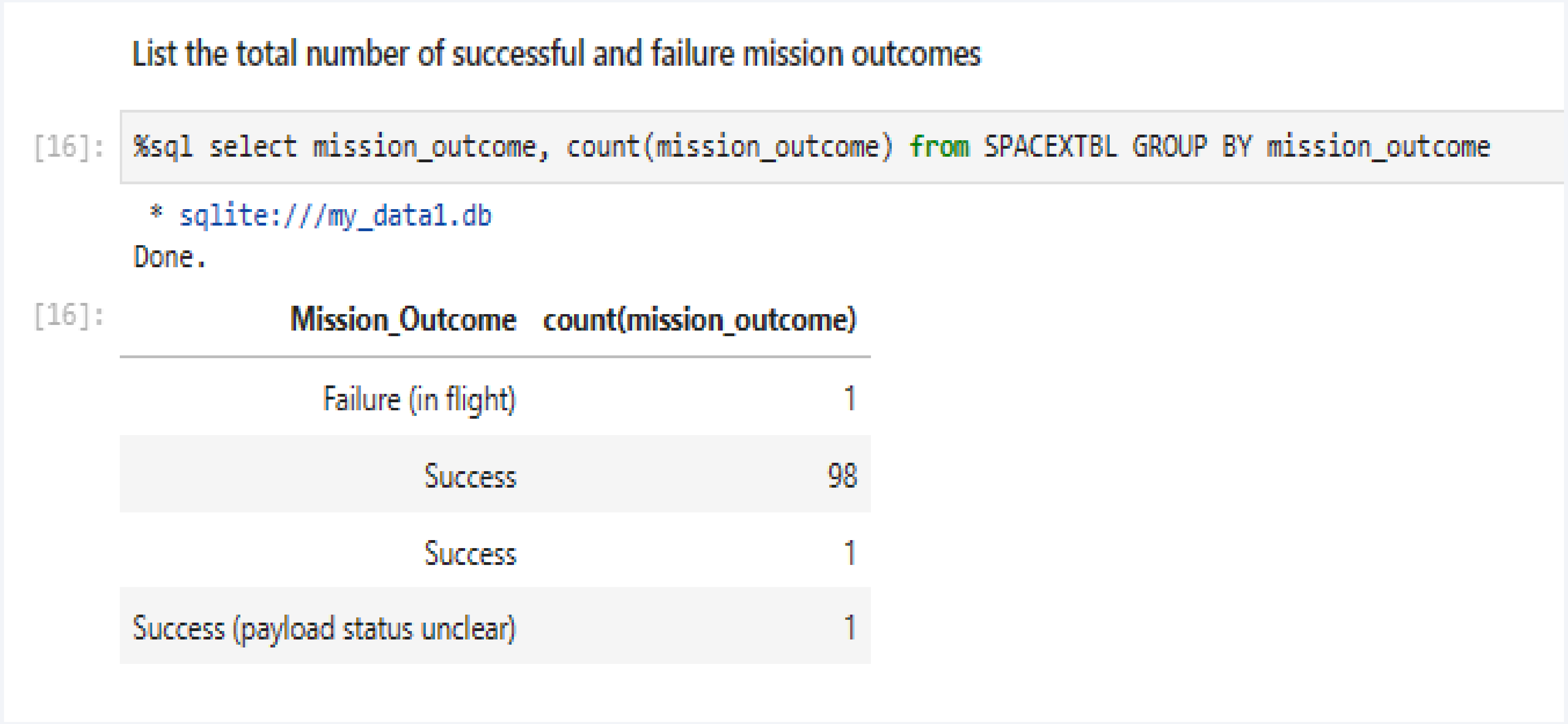
```
[15]: %sql select booster_version from SPACEXTBL where landing_outcome = 'Success (drone ship)'\
      and payload_mass__kg_ between 4000 and 6000
```

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

[15]:

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

Total Number of Successful and Failure Mission Outcomes



List the total number of successful and failure mission outcomes

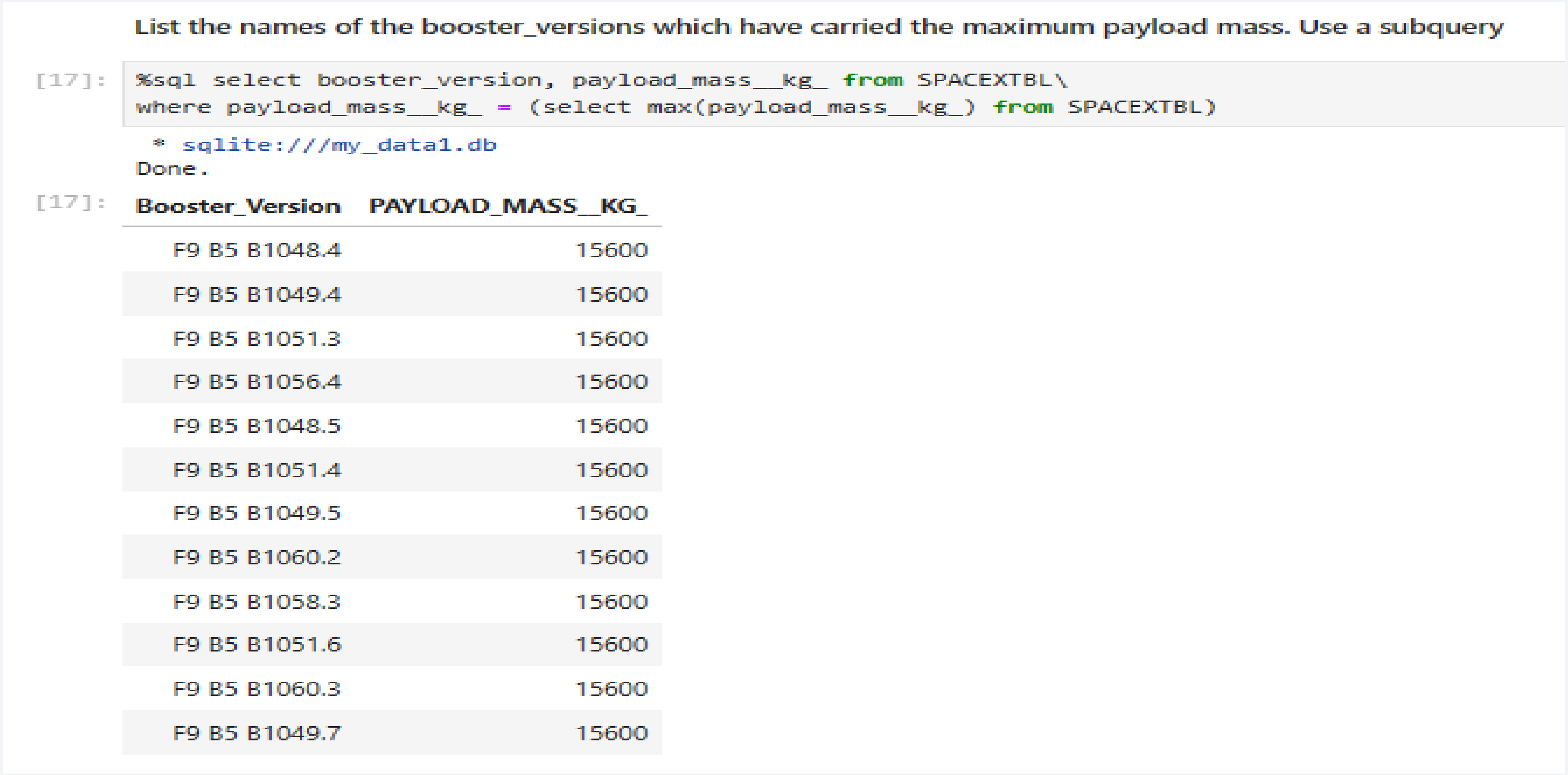
```
[16]: %sql select mission_outcome, count(mission_outcome) from SPACEXTBL GROUP BY mission_outcome
```

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

[16]:

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

Boosters Carried Maximum Payload



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

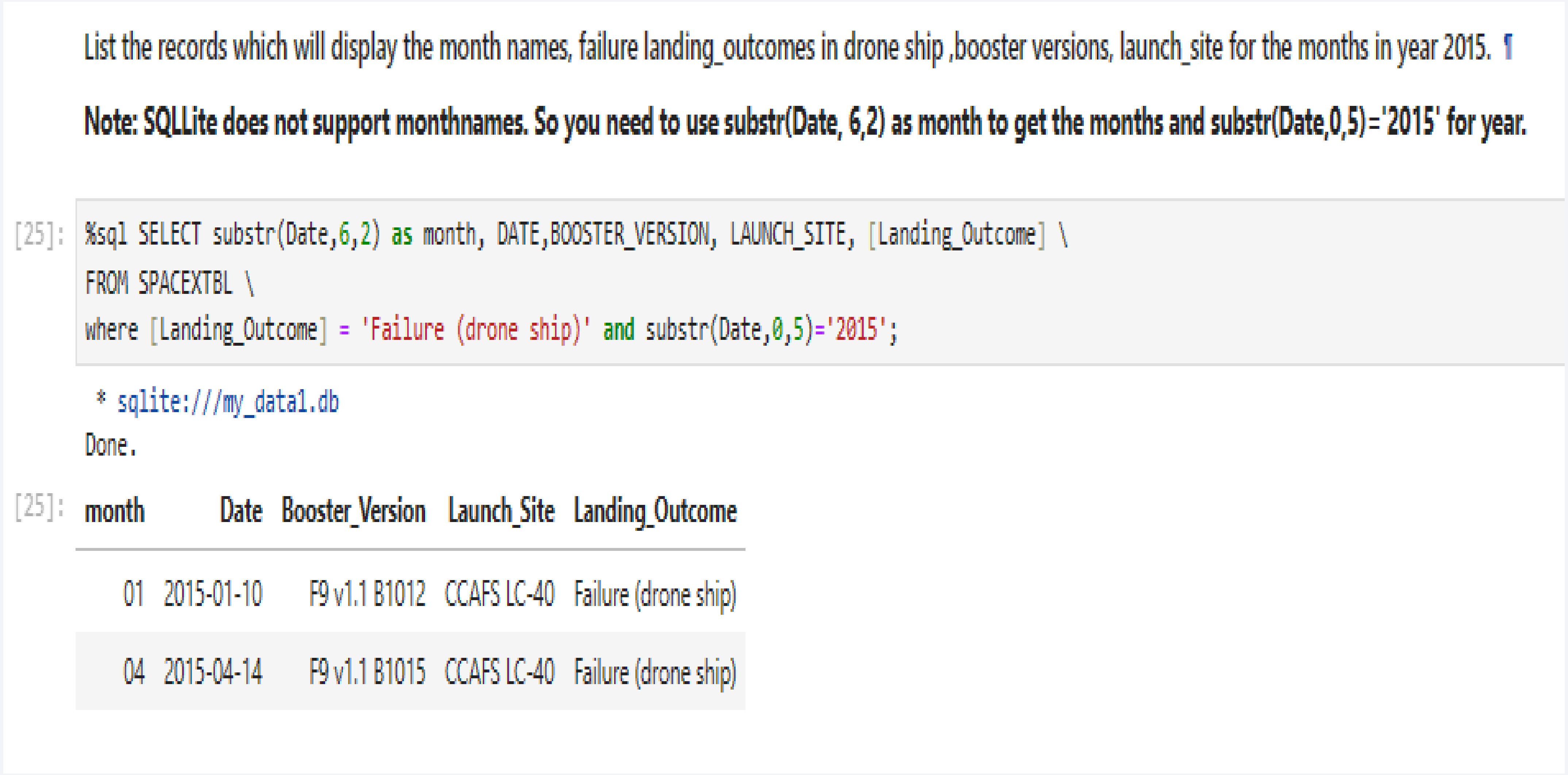
```
[17]: %sql select booster_version, payload_mass__kg_ from SPACEXTBL\
      where payload_mass__kg_ = (select max(payload_mass__kg_) from SPACEXTBL)
```

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

[17]:

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



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.

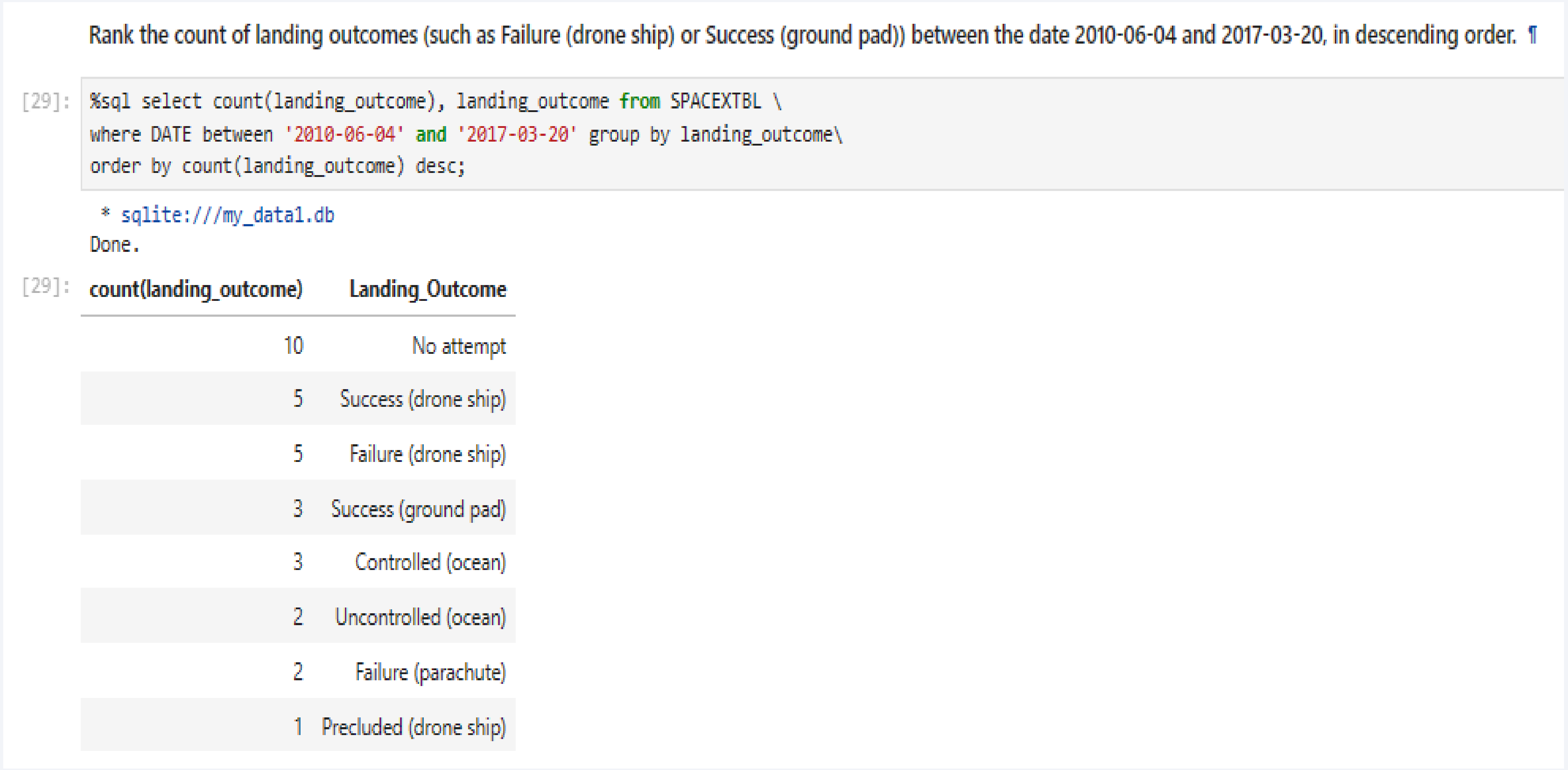
```
[25]: %sql SELECT substr(Date,6,2) as month, DATE,BOOSTER_VERSION, LAUNCH_SITE, [Landing_Outcome] \
FROM SPACEXTBL \
where [Landing_Outcome] = 'Failure (drone ship)' and substr(Date,0,5)='2015';
```

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

[25]:

month	Date	Booster_Version	Launch_Site	Landing_Outcome
01	2015-01-10	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
04	2015-04-14	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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. ¶

```
[29]: %sql select count(landing_outcome), landing_outcome from SPACEXTBL \
      where DATE between '2010-06-04' and '2017-03-20' group by landing_outcome\
      order by count(landing_outcome) desc;
```

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

[29]:

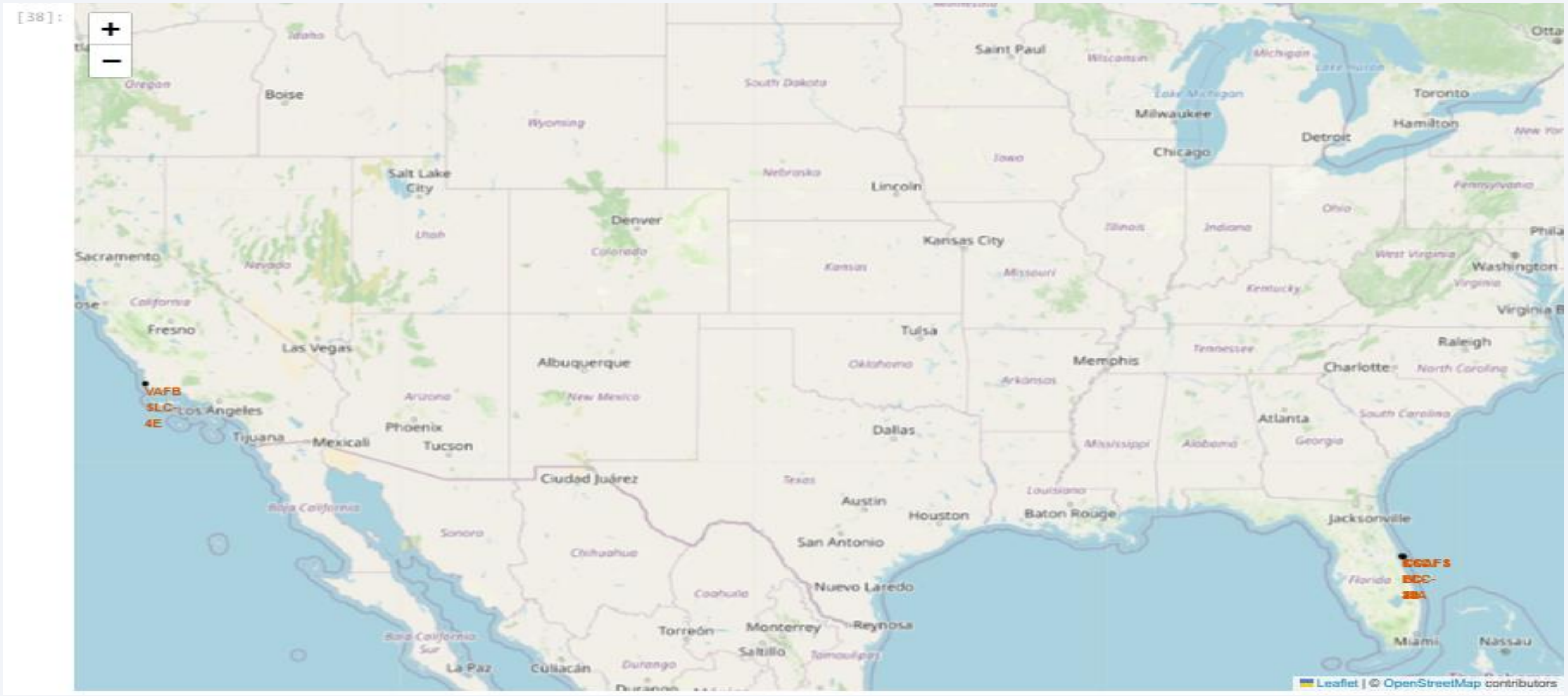
count(landing_outcome)	Landing_Outcome
10	No attempt
5	Success (drone ship)
5	Failure (drone ship)
3	Success (ground pad)
3	Controlled (ocean)
2	Uncontrolled (ocean)
2	Failure (parachute)
1	Precluded (drone ship)

Section 3

Launch Sites Proximities Analysis

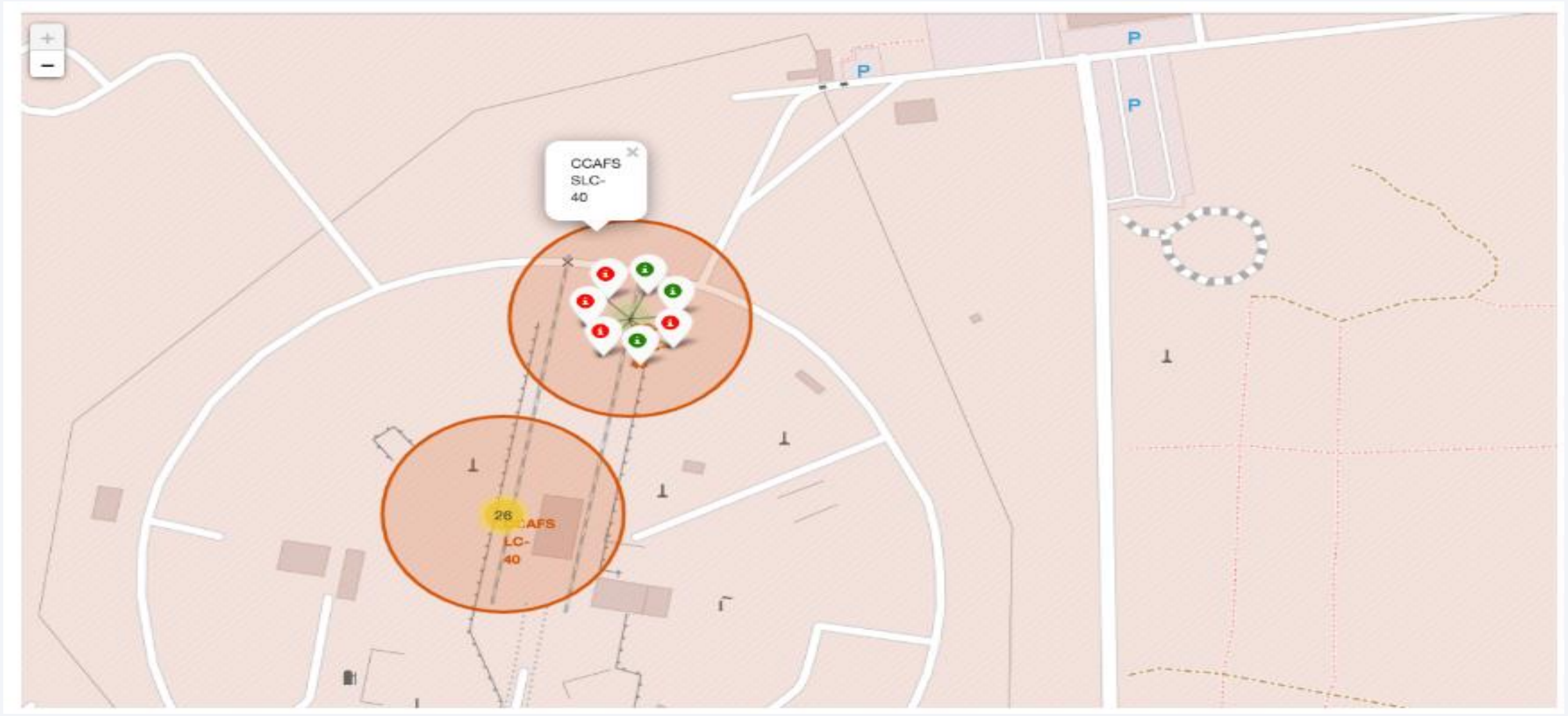
Folium Map Showing ALL Launch Sites

For each launch site, added a Circle object based on its coordinate (Lat, Long) values. In addition, added Launch site name as a popup label



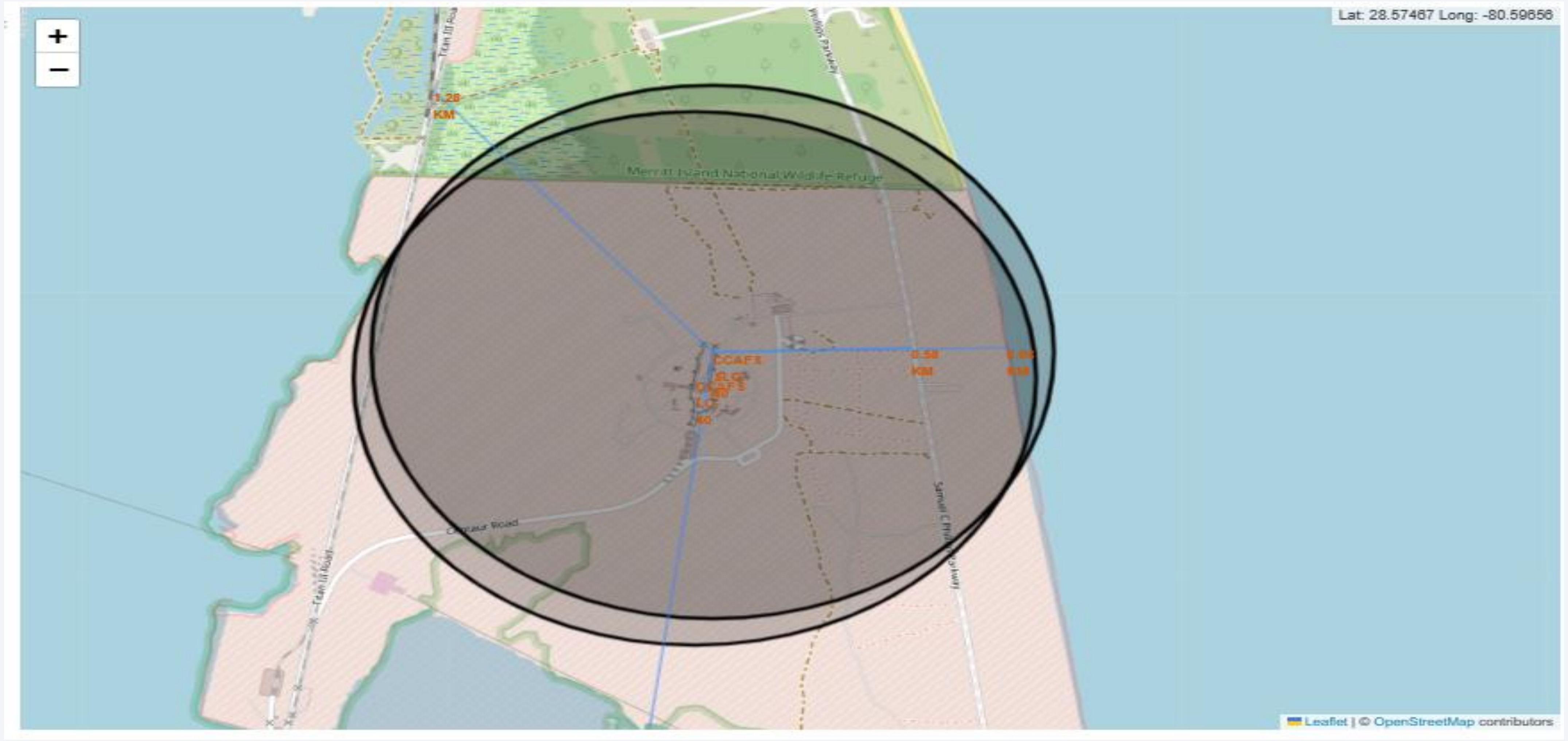
Folium Map Color-labeled Markers

From the color-labeled markers in marker clusters, you should be able to easily identify which launch sites have relatively high success rates.



Folium Map Launch Site Proximities

Create a marker with distance to a closest city, railway, highway, etc.

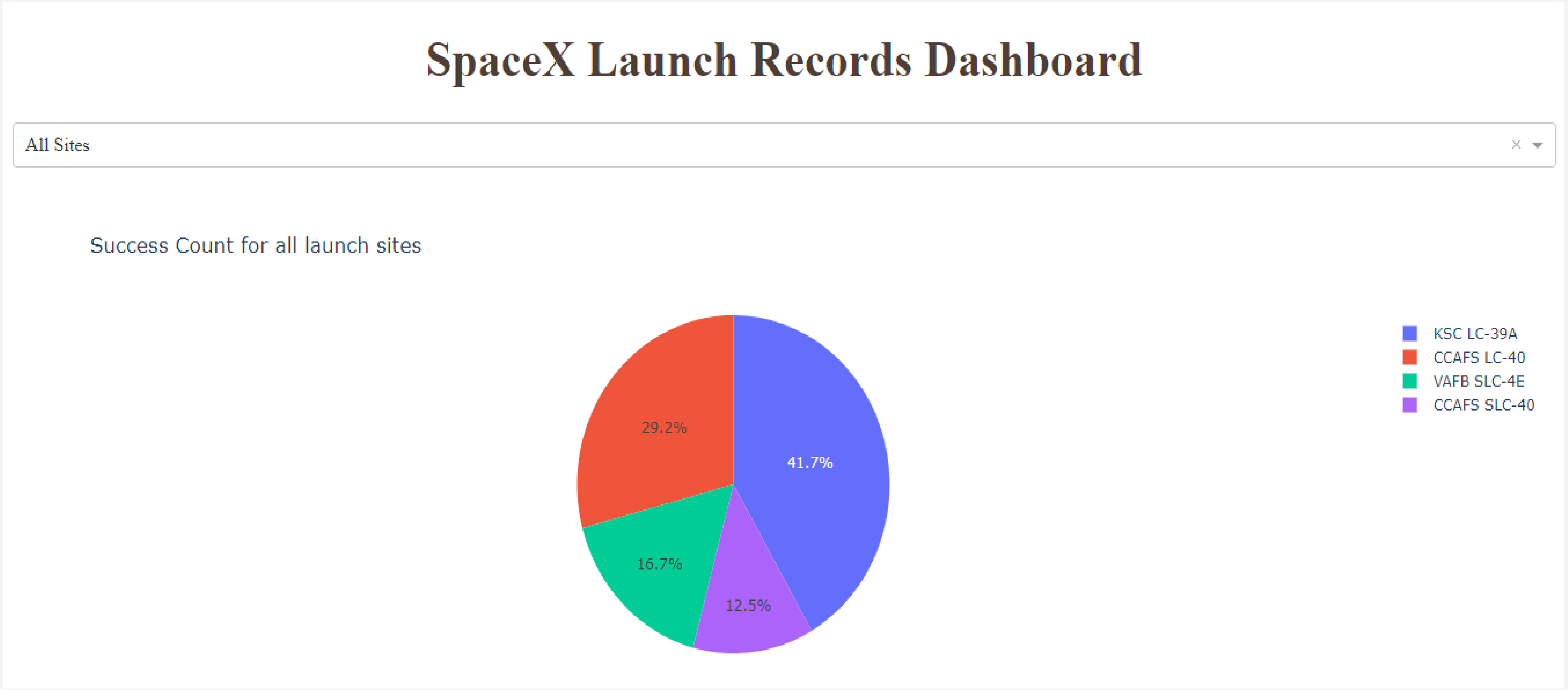

Lat: 28.57467 Long: -80.59656
+
−
1.28 KM
Merritt Island National Wildlife Refuge
CCAFS
0.58 KM
Centaur Road
Samuel C Phillips Parkway
Leaflet | © OpenStreetMap contributors

Section 4

Build a Dashboard with Plotly Dash

Dashboard Success Count for All Launch Sites

We see that different launch sites have different success rates.



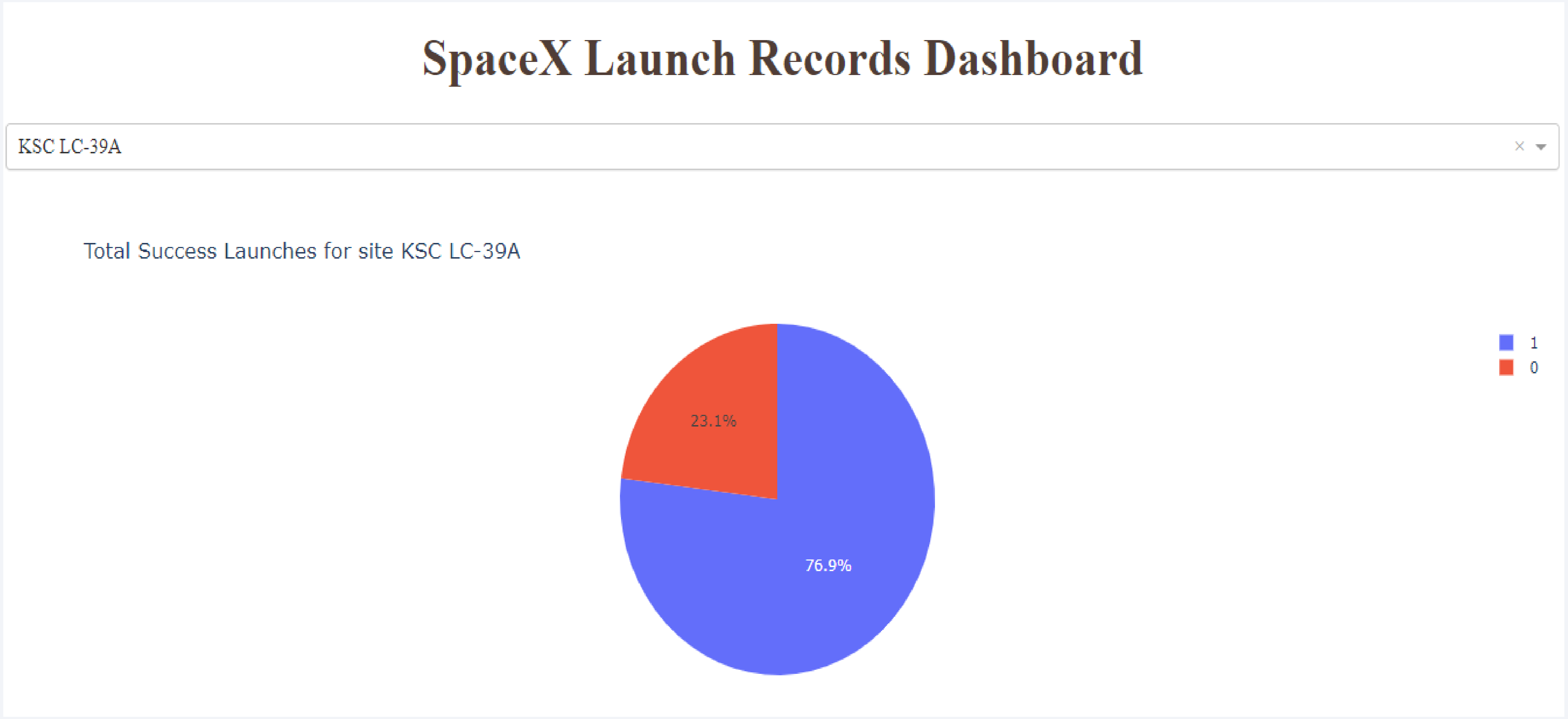
SpaceX Launch Records Dashboard

All Sites

Success Count for all launch sites

Dashboard Highest Launch Success Rate

This site KSC LC-39A has a highest launch success rate of 77%.

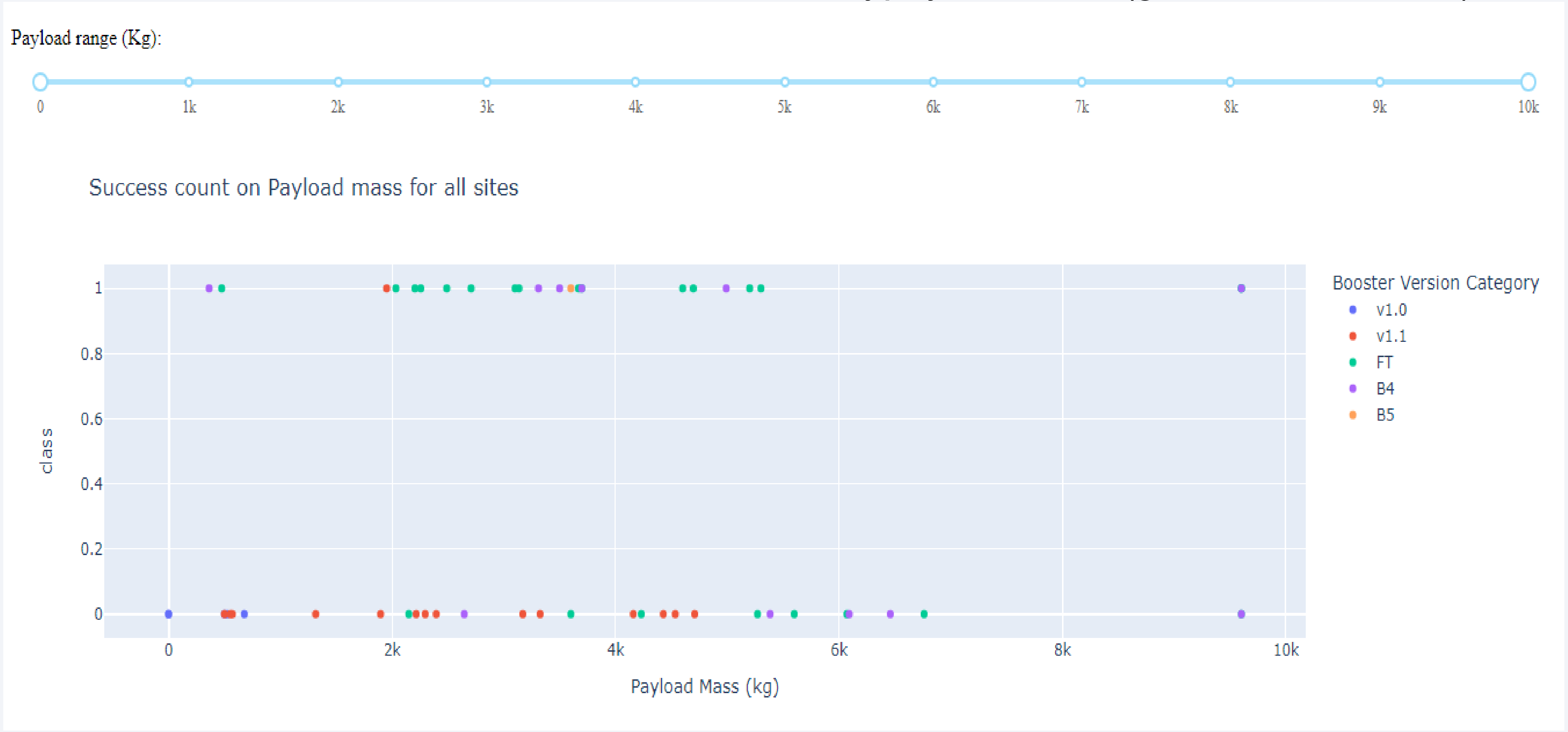

SpaceX Launch Records Dashboard

KSC LC-39A

Total Success Launches for site KSC LC-39A

Dashboard Payload Range Effect on Success Rate

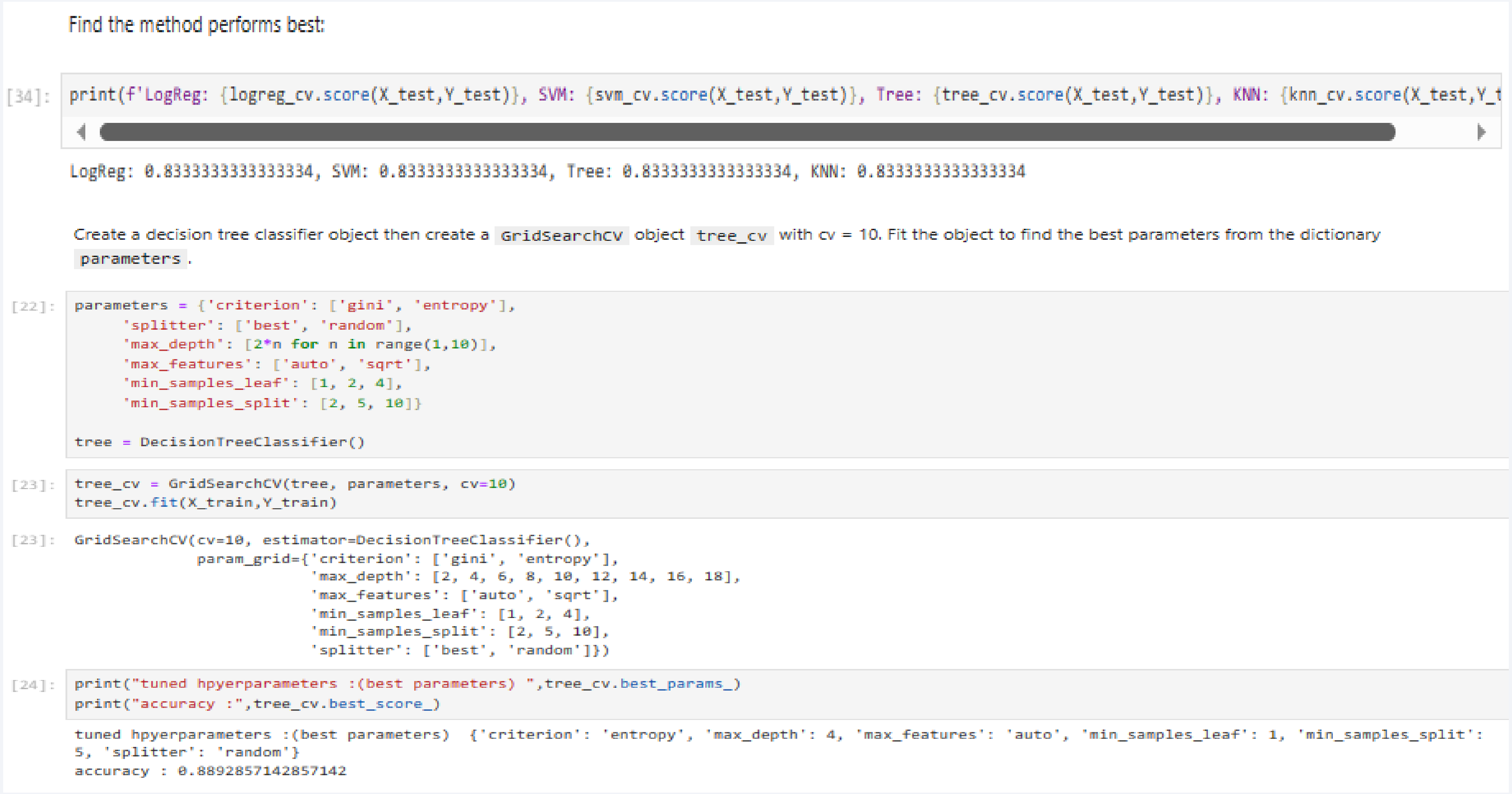
We observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).



Section 5

Predictive Analysis (Classification)

Classification Accuracy



Find the method performs best:

```
[34]: print(f'LogReg: {logreg_cv.score(X_test,Y_test)}, SVM: {svm_cv.score(X_test,Y_test)}, Tree: {tree_cv.score(X_test,Y_test)}, KNN: {knn_cv.score(X_test,Y_t
```

```
LogReg: 0.8333333333333334, SVM: 0.8333333333333334, Tree: 0.8333333333333334, KNN: 0.8333333333333334
```

Create a decision tree classifier object then create a `GridSearchCV` object `tree_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

```
[22]: parameters = {'criterion': ['gini', 'entropy'],
           'splitter': ['best', 'random'],
           'max_depth': [2*n for n in range(1,10)],
           'max_features': ['auto', 'sqrt'],
           'min_samples_leaf': [1, 2, 4],
           'min_samples_split': [2, 5, 10]}

      tree = DecisionTreeClassifier()
```

```
[23]: tree_cv = GridSearchCV(tree, parameters, cv=10)
      tree_cv.fit(X_train,Y_train)
```

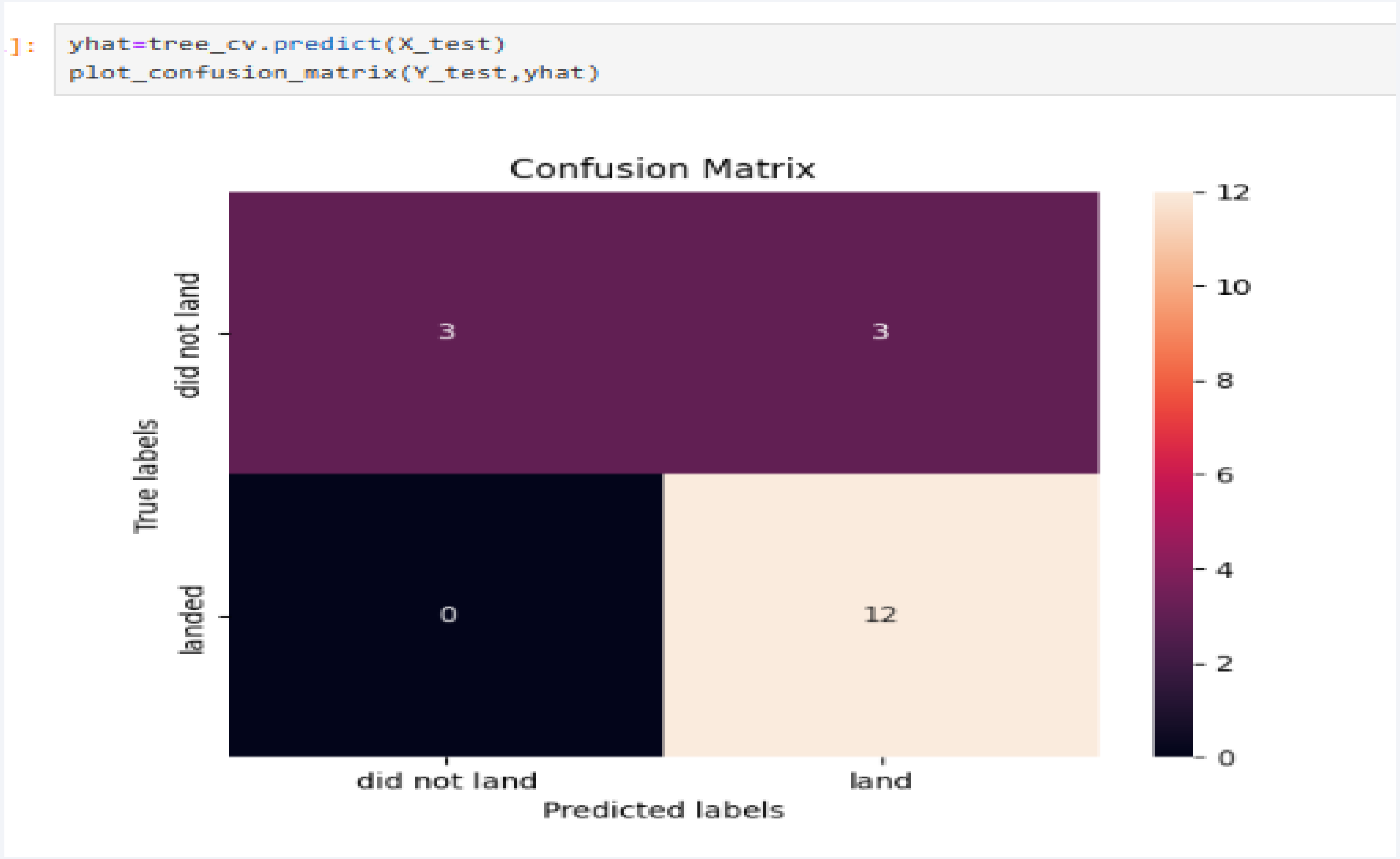
```
[23]: GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
                   param_grid={'criterion': ['gini', 'entropy'],
                               'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18],
                               'max_features': ['auto', 'sqrt'],
                          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

```
[24]: print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
      print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 4, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split':
5, 'splitter': 'random'}
accuracy : 0.8892857142857142
```

Confusion Matrix

All classification models had the same confusion matrices. The main problem is false positives for all the models. Decision Tree classification proved to be the best among all.

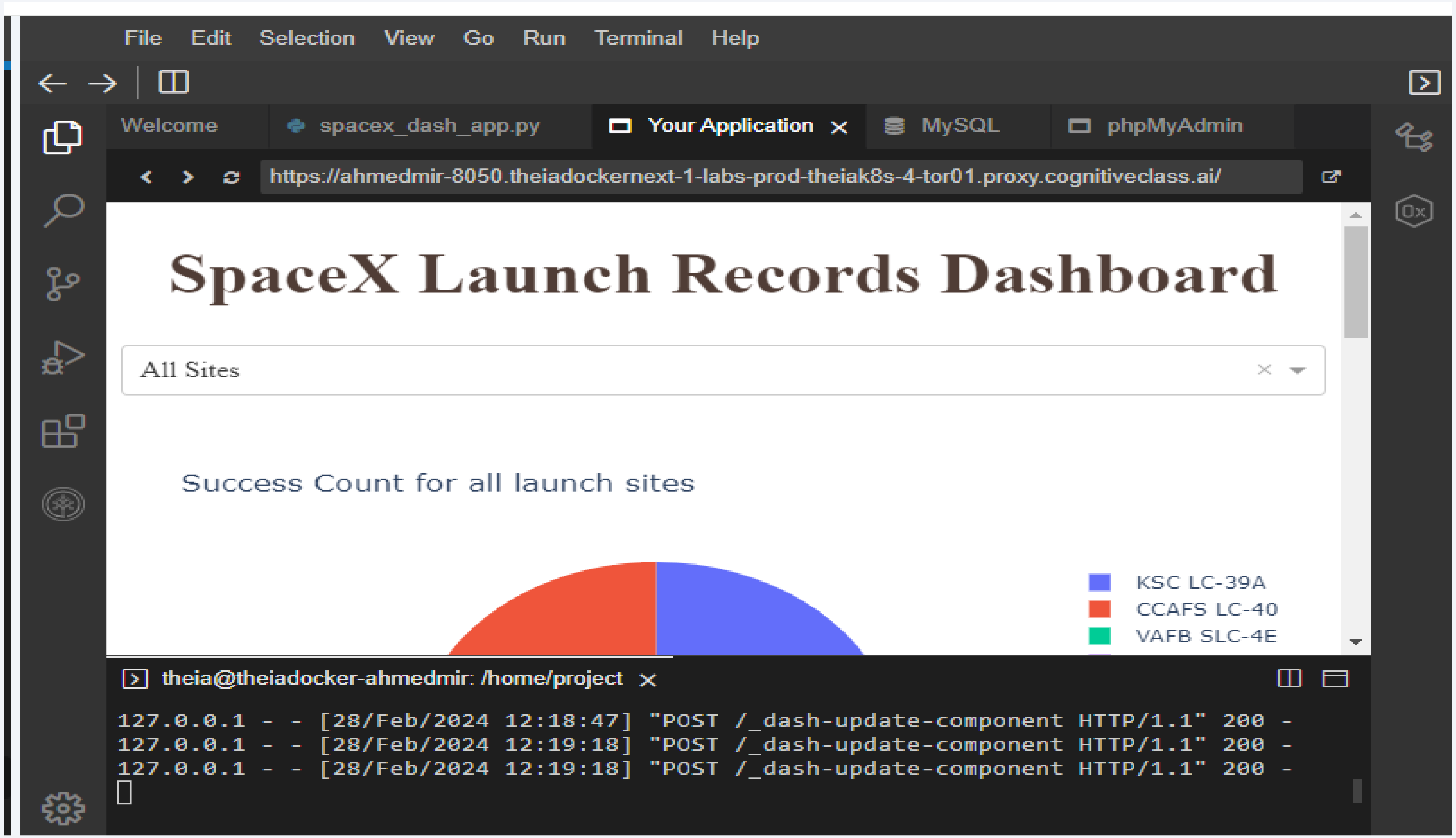


```
]: yhat=tree_cv.predict(X_test)
   plot_confusion_matrix(Y_test,yhat)
```

Conclusions

- Enhanced the Folium map by adding the launch outcomes for each site to see which sites have high success rates
- If you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).
- With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS
- We observe that the success rate since 2013 kept increasing till 2020
- We see that different launch sites have different success rates. CCAFS LC-40, has a success rate of 60 %, while KSC LC-39A and VAFB SLC 4E has a success rate of 77%
- Examining the confusion matrix, we found Decision Tree Classification model performs best

Appendix


File
Edit
Selection
View
Go
Run
Terminal
Help
Welcome
spacex_dash_app.py
Your Application
MySQL
phpMyAdmin
https://ahmedmir-8050.theiadockernext-1-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/
SpaceX Launch Records Dashboard
All Sites
Success Count for all launch sites
KSC LC-39A
CCAFS LC-40
VAFB SLC-4E
theia@theiadocker-ahmedmir: /home/project

```
127.0.0.1 - - [28/Feb/2024 12:18:47] "POST /_dash-update-component HTTP/1.1" 200 -
127.0.0.1 - - [28/Feb/2024 12:19:18] "POST /_dash-update-component HTTP/1.1" 200 -
127.0.0.1 - - [28/Feb/2024 12:19:18] "POST /_dash-update-component HTTP/1.1" 200 -
```

Thank you!