TITLE: REPORT ON PREDICTING THE LANDING OF SPACE X FALCON 9 FIRST STAGE.

NAME: BOBBAI THOMAS

DATE: 30/06/2024

OUTLINE

- Executive summary
- Introduction
- Methodology
  - Data wrangling methodology
  - Interactive visual analytics
  - Predictive analysis
- Results
  - Visualization – charts
  - Sql
  - Map chart
  - Dashboard
  - Predictive
- Discussion
  - Findings
- Conclusion
  - Conclusion and Implication
- Appendix

EXECUTIVE SUMMARY

- This Report is focusing on the objective of predicting the landing of Space X Falcon 9 if it will be successful or it will crash while landing.
- I retrieve the data in two different methods, the first is by web scrapping , using the Request module of python to retrieve the data from Space X API site which is; https://api.spacexdata.com/v4/launches/past which both have a total of 94 rows and 17 columns of data, where columns comprises of; FlightNumber, Date, BoosterVersion, PayloadMass, Orbit, LaunchSite, Outcome, Flights, GridFins, Reused, Legs, LandingPad, Block, ReusedCount, Serial, Longitude and Latitude.

EXECUTIVE SUMMARY

- Data wrangling was carryout using both sql and pandas on python, after cleaning and organizing the data, we end up having a total of 90 rows and 80 columns due to the creation of dummy values for categorical columns such as Orbit, Launchsite, Landingpad and Serial column.
- Data visualization is being carry out with Dash library for interactive visualization and Folium for Geo spatial representation of data.
- Predictive analysis was carry out using classification model such as Logistic Regression, Support Vector Machine, Decision Tree and K-NN. Evaluation of each model was carryout to know the suitable model to predict the landing of Space X Falcon 9.

INTRODUCTION

SpaceX, known for its innovative approaches to space travel, offers Falcon 9 rocket launches at a significantly lower cost of $62 million compared to other providers whose costs can reach upwards of $165 million. A major factor contributing to these savings is SpaceX's ability to reuse the first stage of their rockets. Consequently, predicting the successful landing of the first stage can be crucial in estimating the cost efficiency of a launch. This information becomes particularly valuable for companies considering competing with SpaceX in the space launch market.

Objective

The objective of this report is to create a machine learning pipeline that predicts the successful landing of the Falcon 9's first stage based on available historical data. This prediction model can aid alternate companies in making informed bids against SpaceX by evaluating the potential cost savings and competitive pricing strategies.

METHODOLOGY

- Data Collection: Using Python to carry out the task, data were collected from Space X API site (https://api.spacexdata.com/v4/launches/past), the data consist of 94 rows and 17 columns, the columns are; FlightNumber, Date, BoosterVersion, PayloadMass, Orbit, LaunchSite, Outcome, Flights, GridFins, Reused, Legs, LandingPad, Block, ReusedCount, Serial, Longitude and Latitude.
- Data Wrangling: The data consist of both Falcon 9 and Falcon 1, therefore we remove the Falcon 1 to focus on only Falcon 9, after doing that, we are left with only 90 rows out of 94.
- Missing values: We have two columns with missing values, the columns are; Payload Mass and LandingPad, with each missing 5 and 26 values respectively. Payload Mass missing value is replace with the mean of all the values in the column and missing value on LandingPad column where left that way, representing that the LandingPad is not use.
- The number of time each outcome of the landing took place is verify, which is shown on the table. The successful one where assign a value 1 and the unsuccessful one is 0 and with that outcome, a new column is added making the dataset to have 18 column.

OUTCOME	COUNT	CLASS
True ASDS	41	1
None None	19	0
True RTLS	14	1
False ASDS	6	0
True Ocean	5	1
False Ocean	2	0
None ASDS	2	0
False RTLS	1	0

METHODOLOGY

BRIEF SUMMARY OF THE DATASET

```
df.head(5)
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857	0
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857	0
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857	0
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632093	0
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857	0

METHODOLOGY

EXPLOTARY DATA ANALYSIS: Using Visualization to shop relationship between some of the attributes, the visualization are carried out base on the following using Python Matplotlib and Seaborn:

- Pay Load Mass VS Flight Number: this visualization shows the Payload Mass base on the Flight Number focusing on the success rate using Scatter Chart.
- Launch Site VS Flight Number: The visualization shows the Launchsite base on Flight Number focusing on the success rate using Scatter Chart.
- Launch Site VS Pay Load Mass: This is to know if the load mass have a significant effect on Launch Site focusing on the success rate using Scatter Chart.
- Success Rate VS Orbit type: This is to know if there is any relationship between the two.
- Flight Number VS Orbit type: We want to see if there is any relationship between FlightNumber and Orbit type focusing on the success rate using Scatter Chart..
- Payload vs. Orbit: Scatter point charts to reveal the relationship between Payload and Orbit type focusing on the success rate.
- Year VS Success Rate: This will reveal the relationship between the Year and Success Rate, using line chart.

METHODOLOGY

PREDICTIVE ANALYSIS

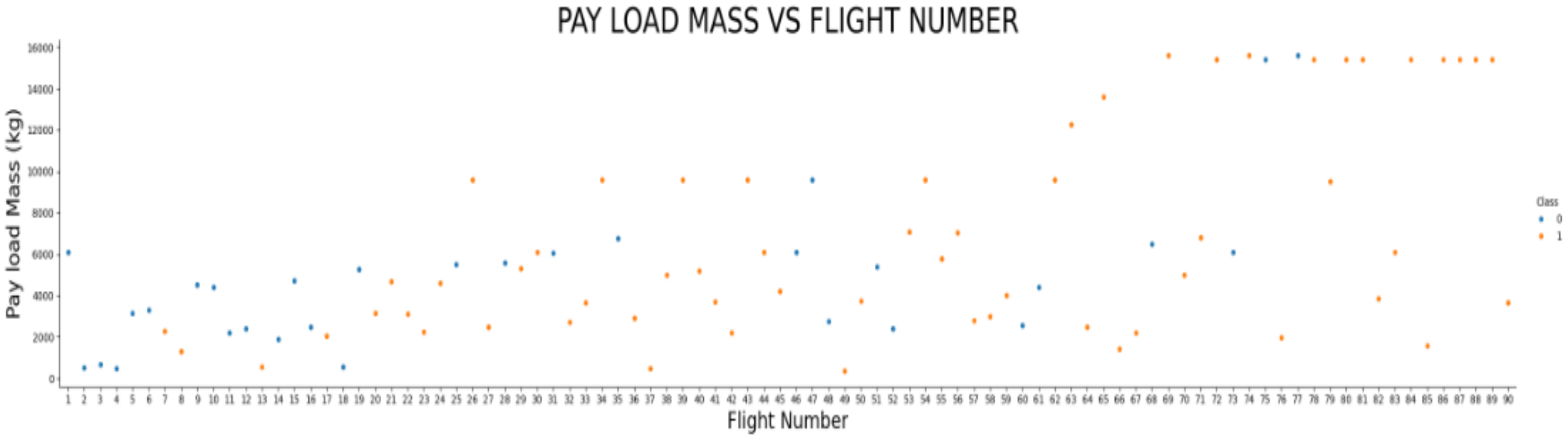
Several machine learning models were trained and tested to predict the successful landing of the first stage of the Falcon 9. The models included in the evaluation were:

1. K-Nearest Neighbors (KNN)
2. Decision Tree
3. Support Vector Machine (SVM)
4. Logistic Regression

- The dataset used for training these models consisted of historical launch data, including features such as launch site, payload mass, orbit, and other relevant parameters.

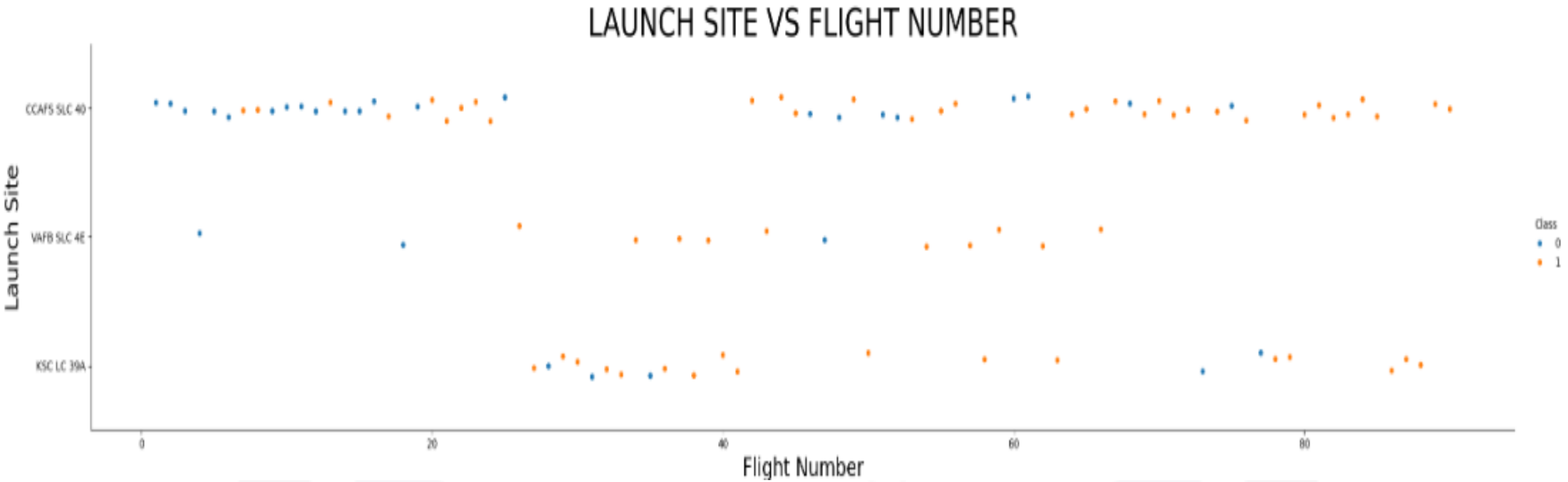
RESULTS

EDA VISUALIZATION



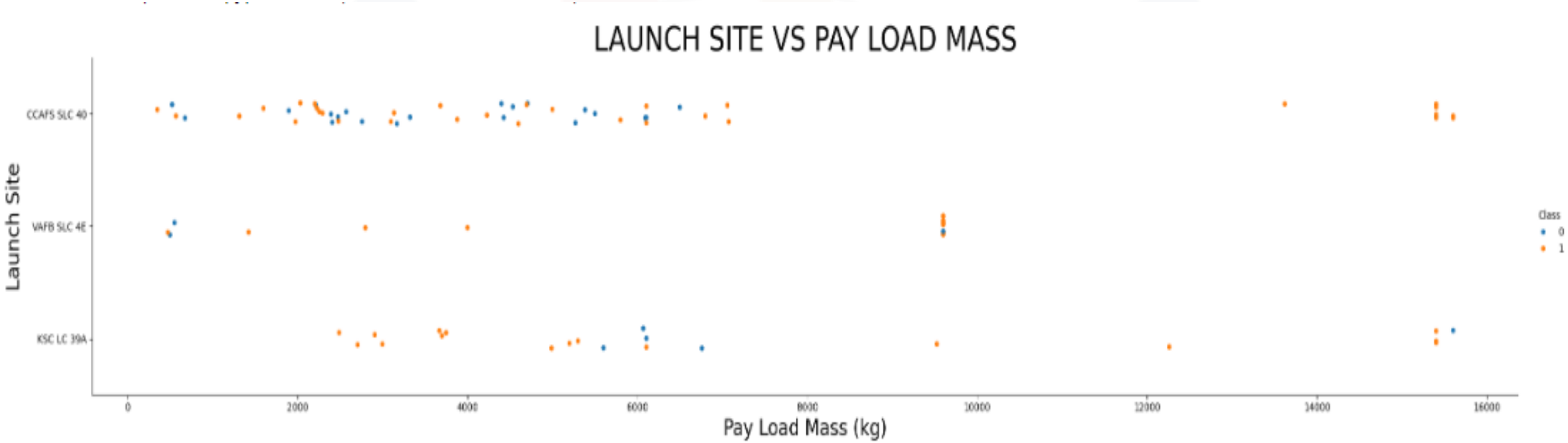
RESULTS

EDA VISUALIZATION



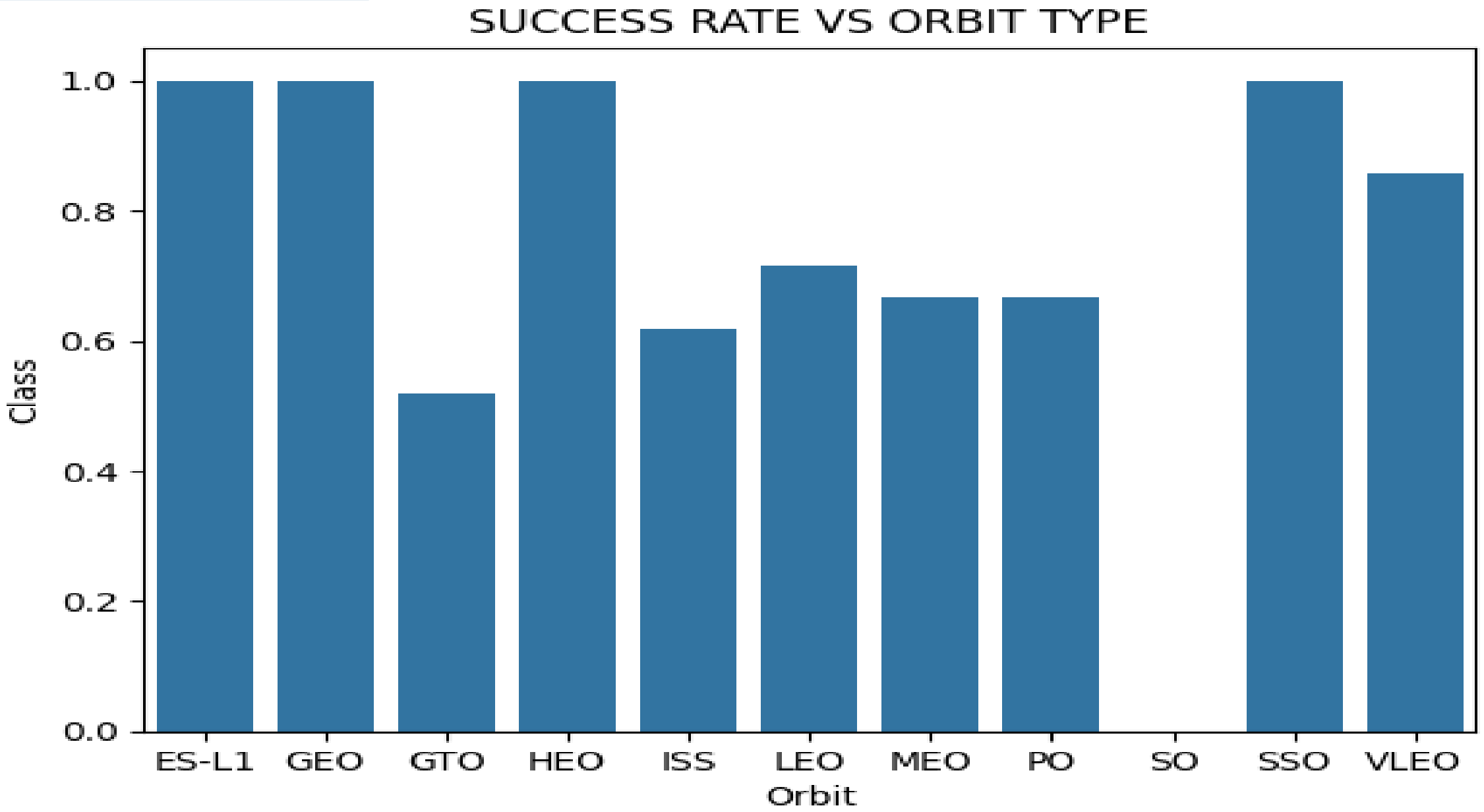
RESULTS

EDA VISUALIZATION



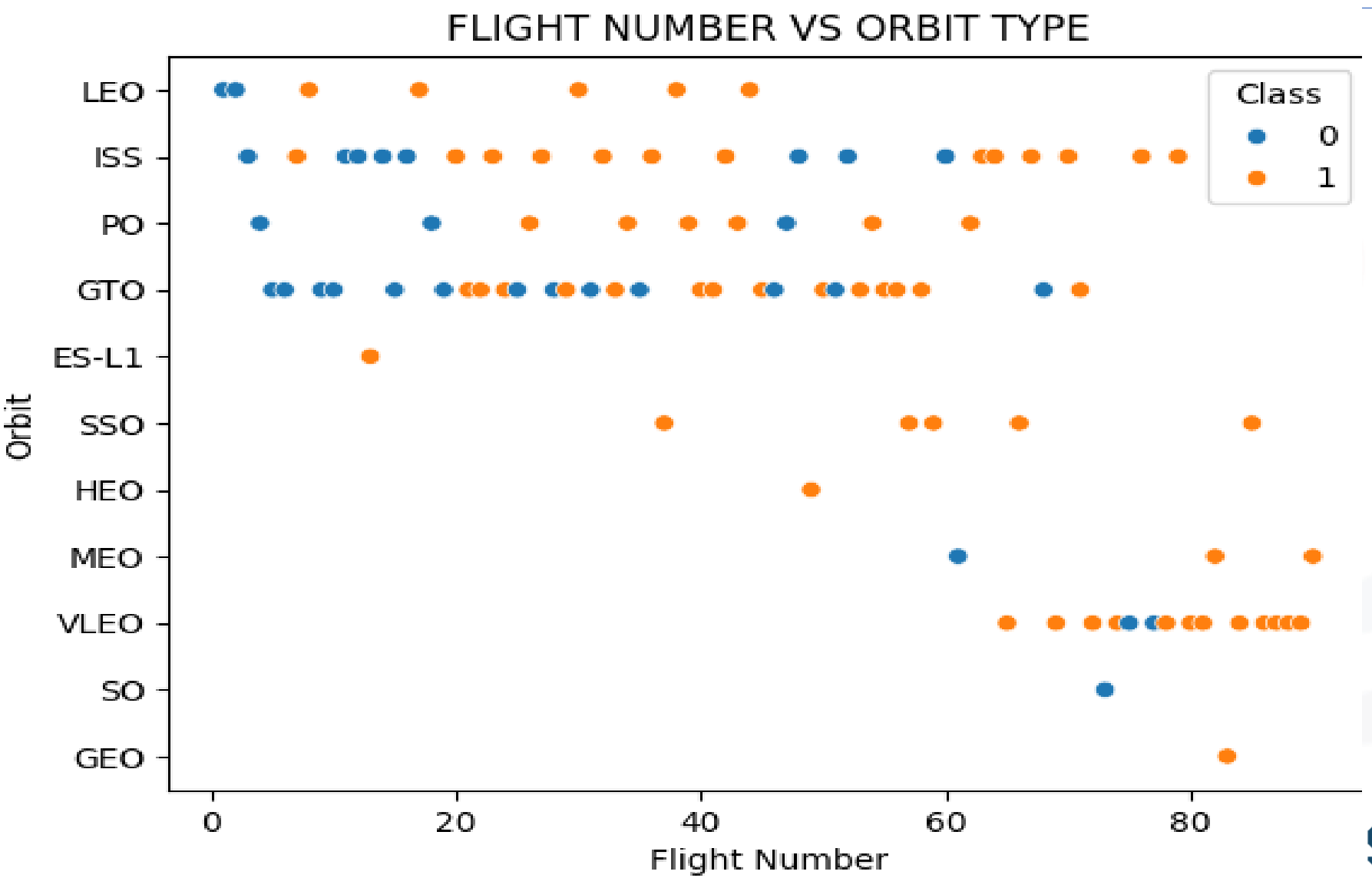
RESULTS

EDA VISUALIZATION



RESULTS

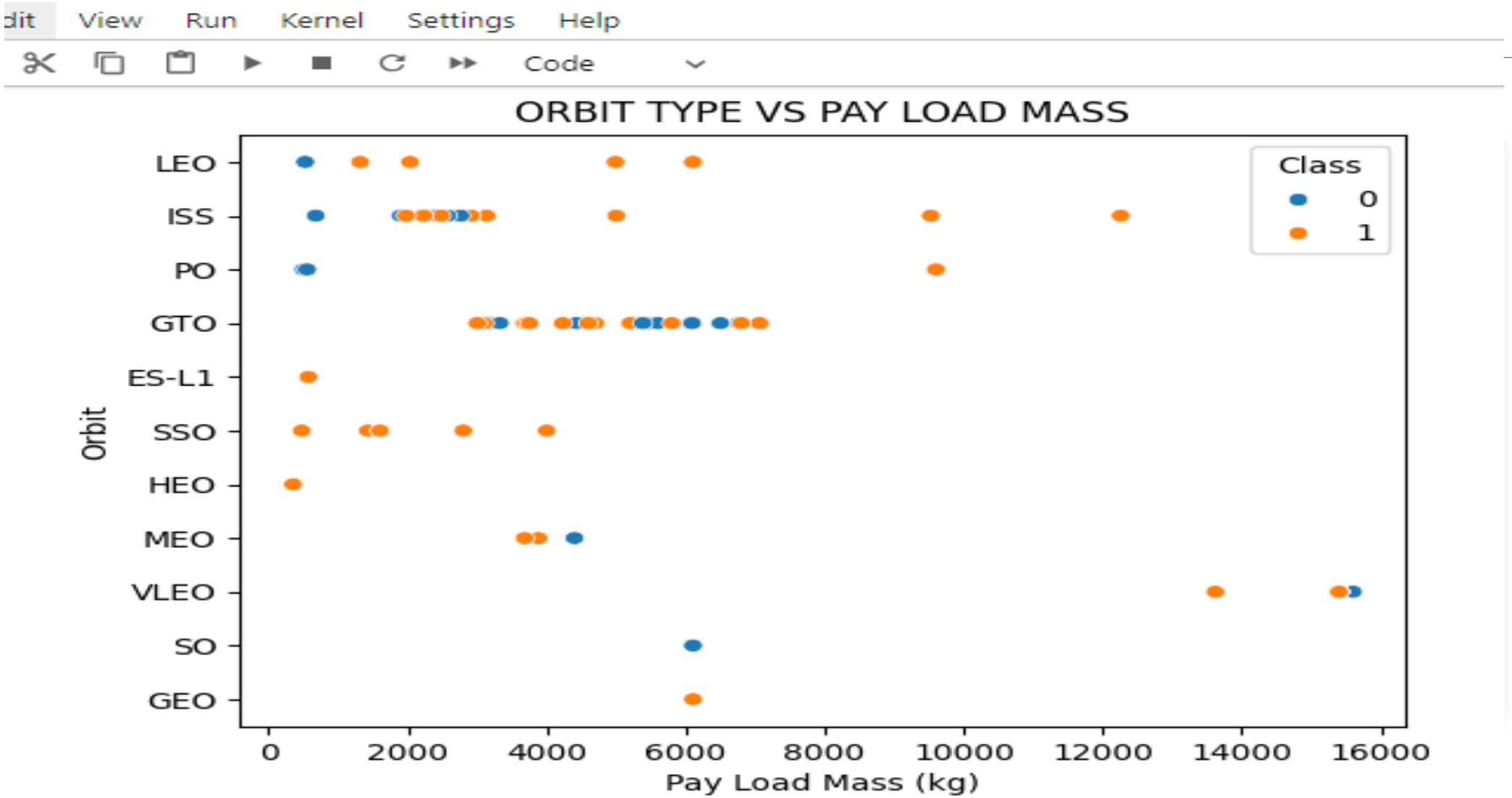
EDA VISUALIZATION



SKILLS NETWORK

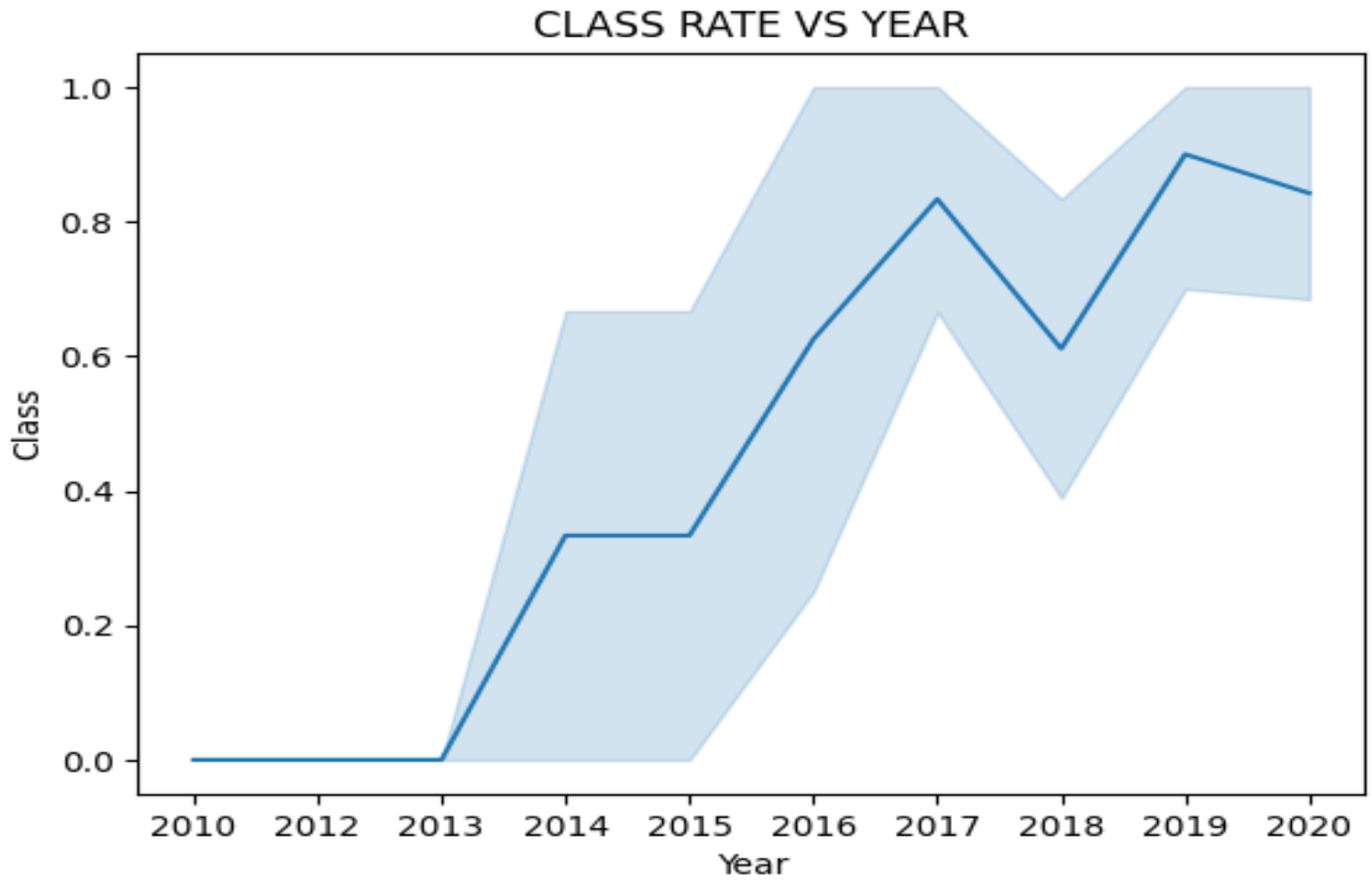
RESULTS

EDA VISUALIZATION



RESULTS

EDA VISUALIZATION



RESULTS

EDA SQL RESULTS

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

RESULTS

EDA SQL RESULTS

Task 2

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

[11]:
```
%%sql
SELECT * FROM SPACEXTABLE
WHERE Launch_Site LIKE 'CCA%'
limit 5
```

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

[11]:

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

RESULTS

EDA SQL RESULTS

Task 3

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

[12]:
```
%%sql
SELECT SUM(PAYLOAD_MASS__KG_) AS TOTAL_MASS_NASA
FROM SPACEXTABLE
WHERE Customer = 'NASA (CRS)'
```

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

[12]:

TOTAL_MASS_NASA
45596

RESULTS

EDA SQL RESULTS

Task 4

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

[13]:
```
%%sql
SELECT AVG(PAYLOAD_MASS__KG_) AS AVG_MASS_NASA
FROM SPACEXTABLE
WHERE Booster_Version = 'F9 v1.1'
```

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

[13]:

AVG_MASS_NASA
2928.4

RESULTS

EDA SQL RESULTS

Task 5

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

Hint:Use min function

[14]:
```
%%sql
SELECT Date
FROM SPACEXTABLE
WHERE Landing_Outcome = 'Success'
LIMIT 1
```

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

[14]:

Date
2018-07-22

RESULTS

EDA SQL RESULTS

Task 6

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 SPACEXTABLE
WHERE PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000 AND Mission_Outcome='Success'
```

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

[15]:

Booster_Version
F9 v1.1
F9 v1.1 B1011
F9 v1.1 B1014
F9 v1.1 B1016
F9 FT B1020
F9 FT B1022
F9 FT B1026
F9 FT B1030
F9 FT B1021.2
F9 FT B1032.1

RESULTS

EDA SQL RESULTS

Task 7

List the total number of successful and failure mission outcomes

[16]:
```
%%sql
SELECT COUNT(Mission_Outcome)
FROM SPACEXTABLE
```

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

[16]:

COUNT(Mission_Outcome)
101

RESULTS

EDA SQL RESULTS

Task 8

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

[17]:
```
%%sql
SELECT Booster_Version
FROM SPACEXTABLE
WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE)
```

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

[17]:

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

RESULTS

EDA SQL RESULTS

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.

[18]:
```
%%sql
SELECT substr(Date,6,2) AS MONTH, Landing_Outcome, Booster_Version, Launch_Site
FROM SPACEXTABLE
WHERE substr(Date,0,5) = '2015' AND Landing_Outcome LIKE 'Failure%'
```

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

[18]:

MONTH	Landing_Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

RESULTS

EDA SQL RESULTS

Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

Task 10

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

[19]:
```
%%sql
SELECT * FROM SPACEXTABLE
WHERE DATE>'2010-06-04' AND DATE < '2017-03-20'
ORDER BY Landing_Outcome DESC
```

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

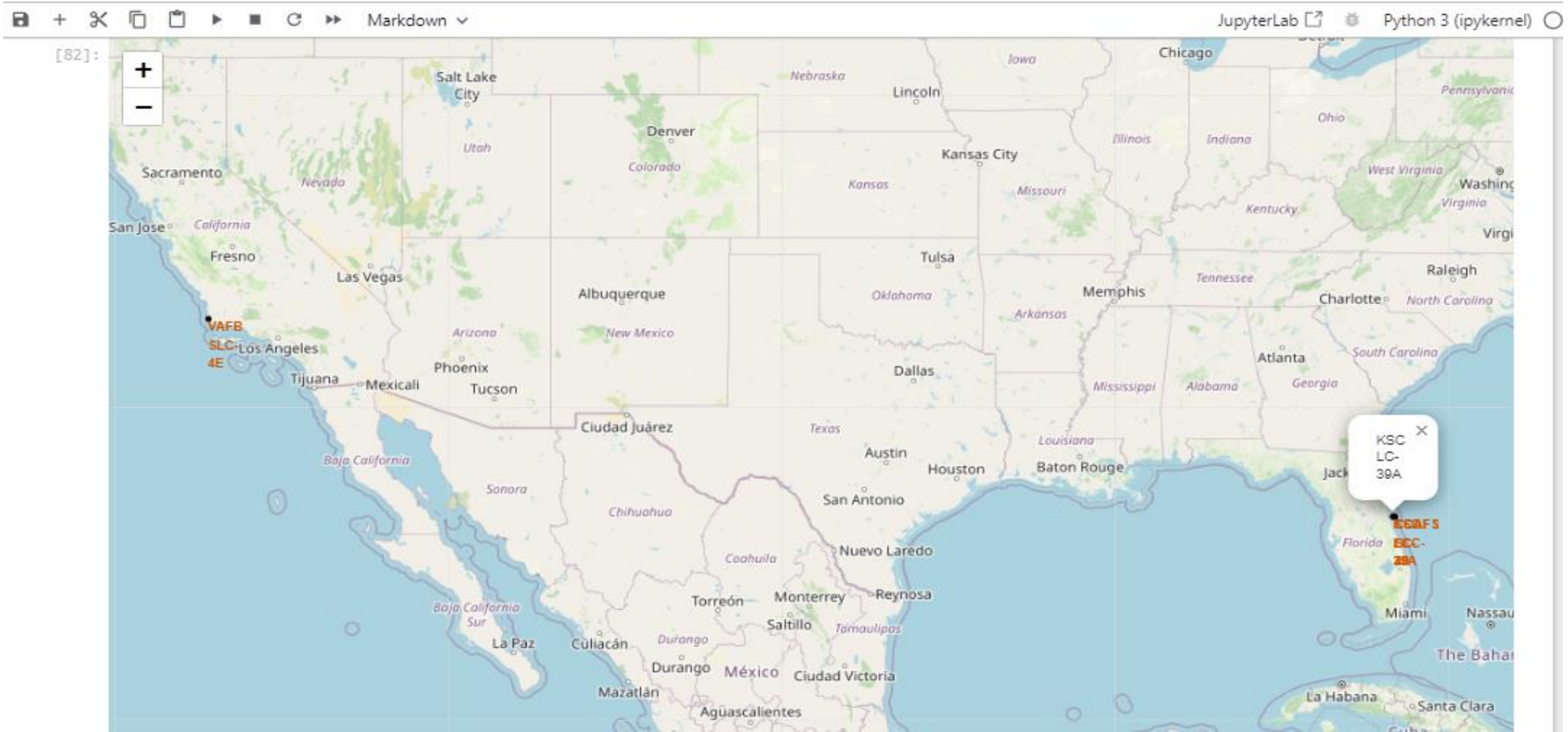
[19]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2013-09-29	16:00:00	F9 v1.1 B1003	VAFB SLC-4E	CASSIOPE	500	Polar LEO	MDA	Success	Uncontrolled (ocean)
2014-09-21	5:52:00	F9 v1.1 B1010	CCAFS LC-40	SpaceX CRS-4	2216	LEO (ISS)	NASA (CRS)	Success	Uncontrolled (ocean)
2015-12-22	1:29:00	F9 FT B1019	CCAFS LC-40	OG2 Mission 2 11 Orbcomm-OG2 satellites	2034	LEO	Orbcomm	Success	Success (ground pad)
2016-07-18	4:45:00	F9 FT B1025.1	CCAFS LC-40	SpaceX CRS-9	2257	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2016-04-08	20:43:00	F9 FT B1021.1	CCAFS LC-40	SpaceX CRS-8	3136	LEO (ISS)	NASA (CRS)	Success	Success (drone ship)
2016-			CCAFS LC-				SKY Perfect JSAT		Success (drone

RESULTS

MAP WITH FOLIUM RESULTS

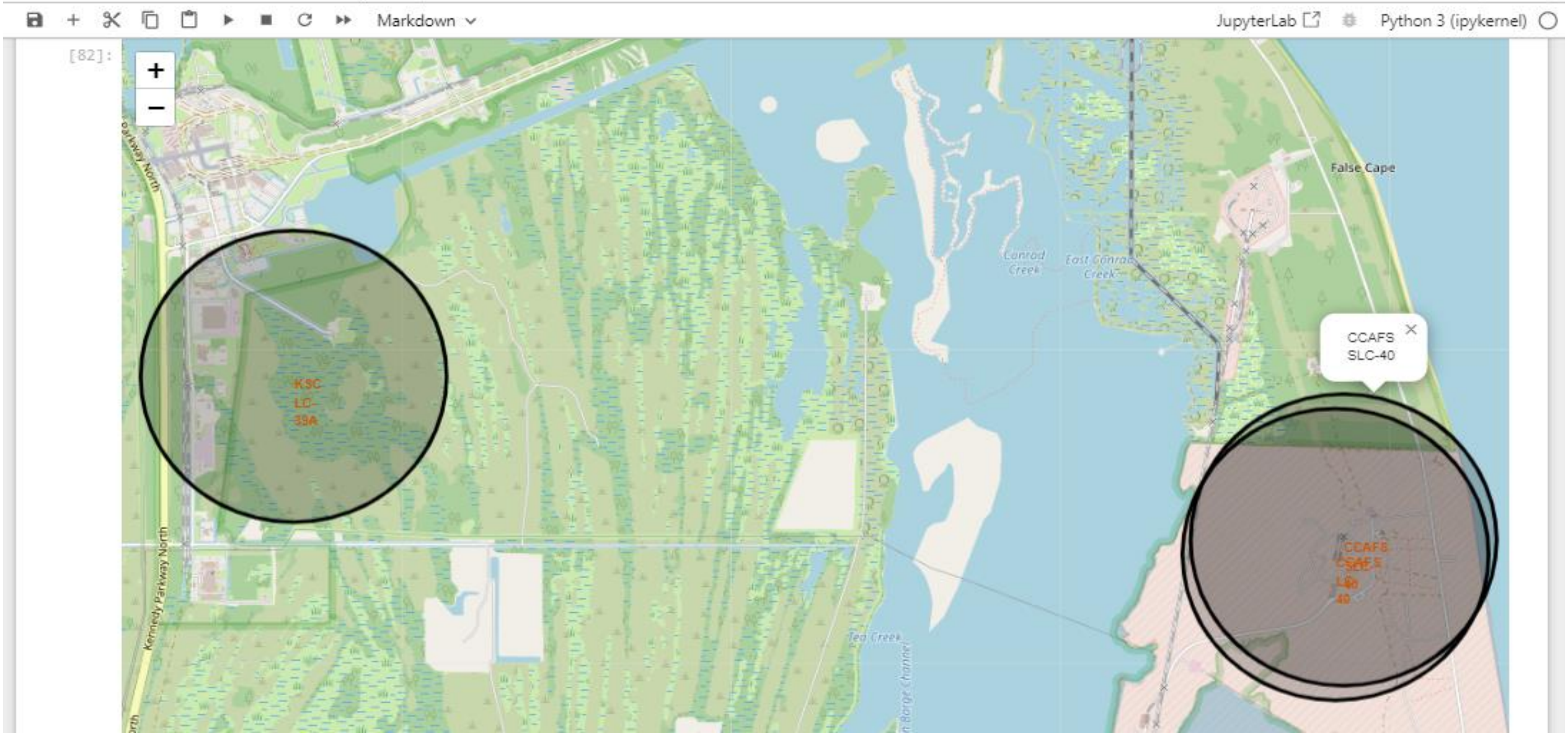
MAP WITH MARKED LAUNCH SITES



RESULTS

MAP WITH FOLIUM RESULTS

MARKING THE SUCCESS/FAIL LAUNCHSITE

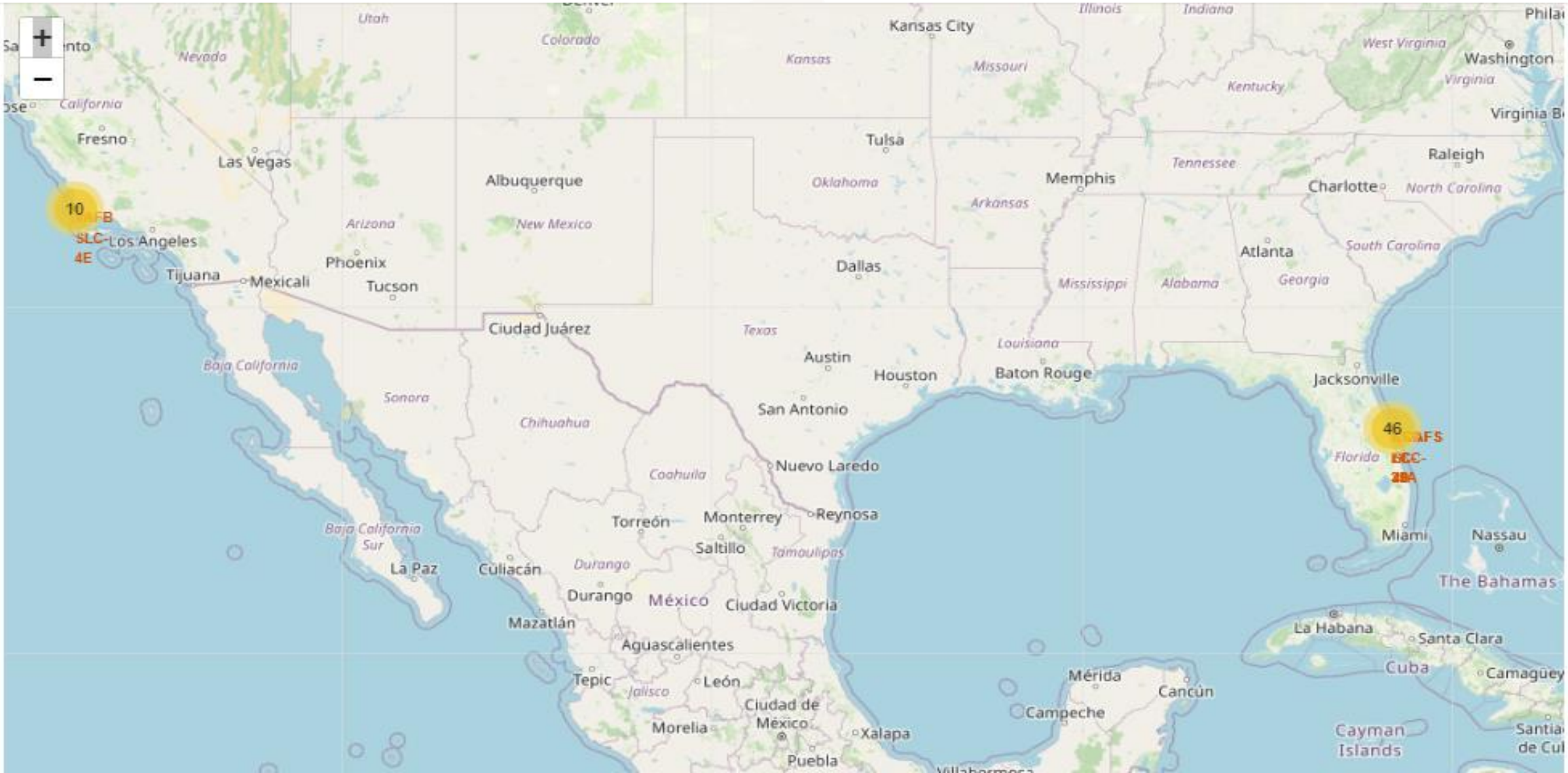


RESULTS

MAP WITH FOLIUM RESULTS

MARKING THE SUCCESS/FAIL LAUNCHSITE

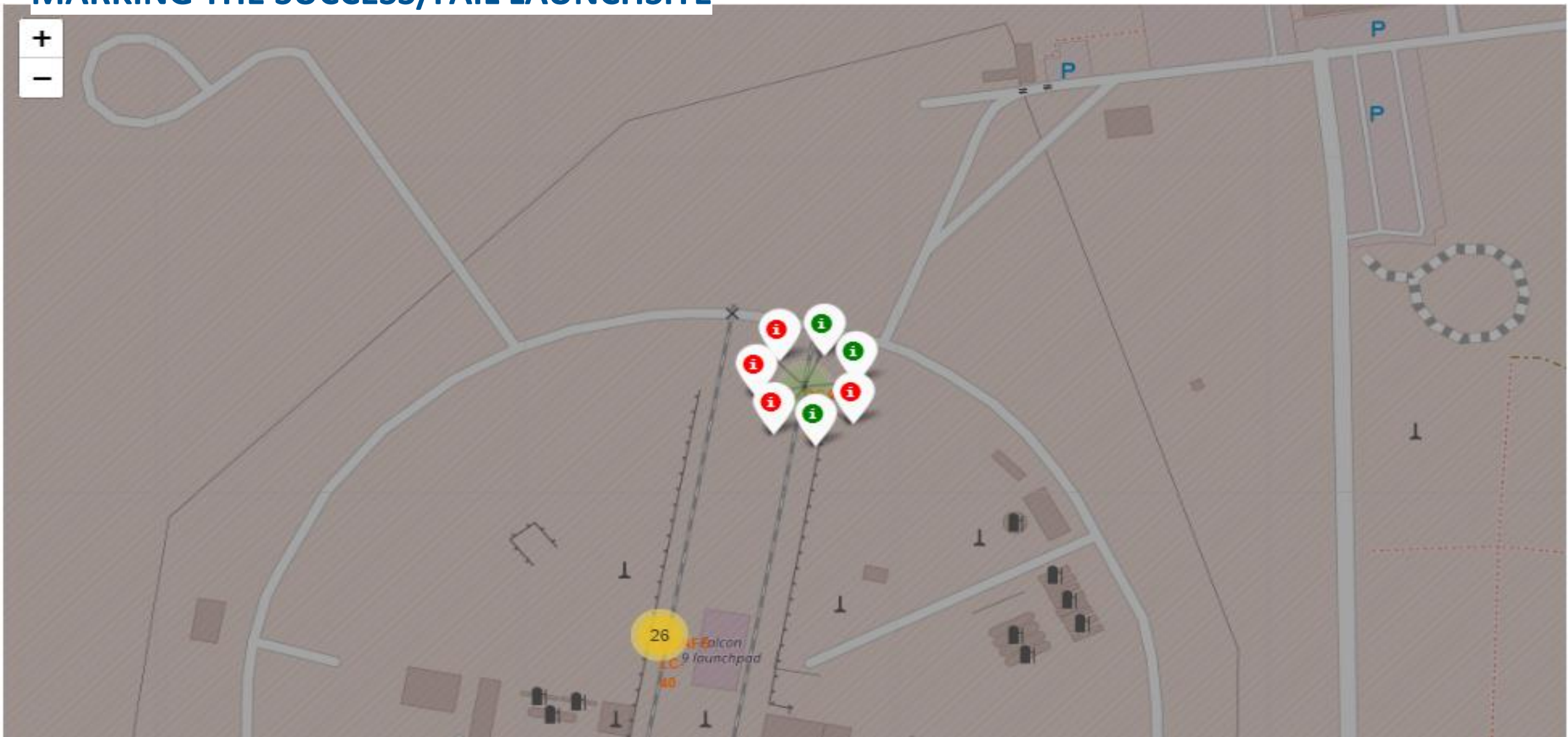
[86]:



RESULTS

MAP WITH FOLIUM RESULTS

MARKING THE SUCCESS/FAIL LAUNCHSITE



RESULTS

MAP WITH FOLIUM RESULTS

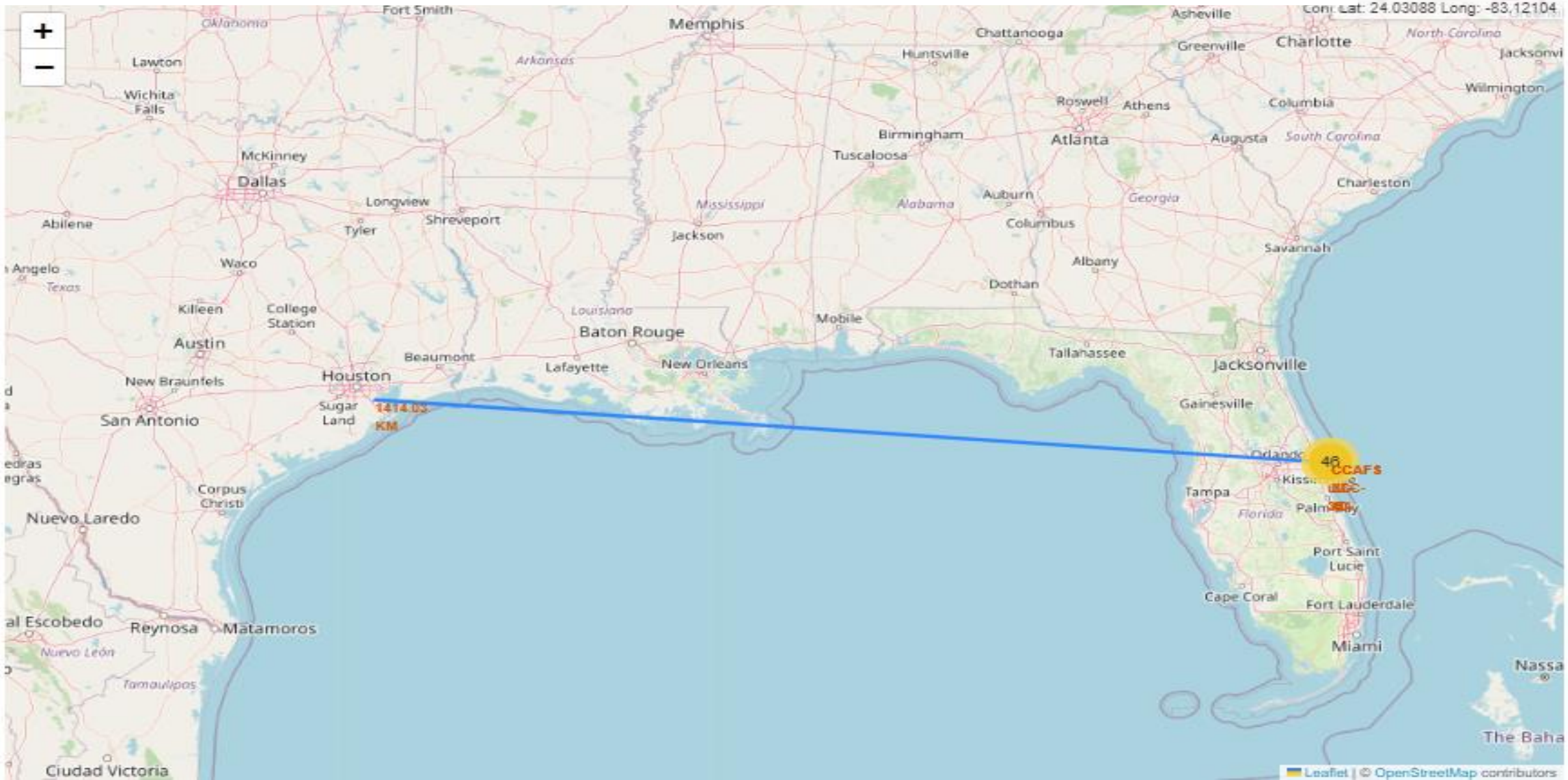
MOUSE POSITION ADDED

RESULTS

MAP WITH FOLIUM RESULTS

CREATING POLYLINE LAUNCH SITE TO COASTAL LINE WITH DISTANCE CALCULATED

[109]:

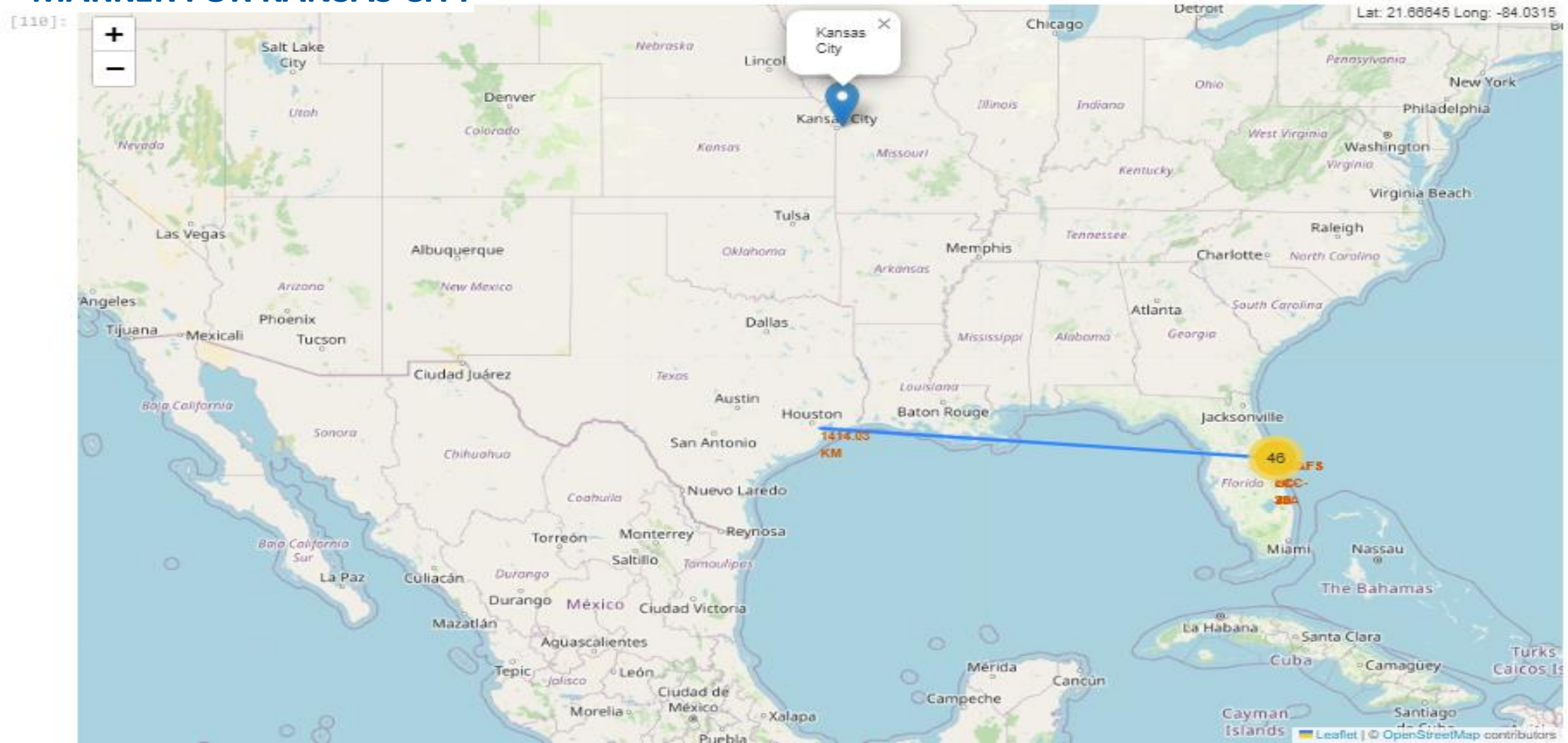


RESULTS

MAP WITH FOLIUM RESULTS

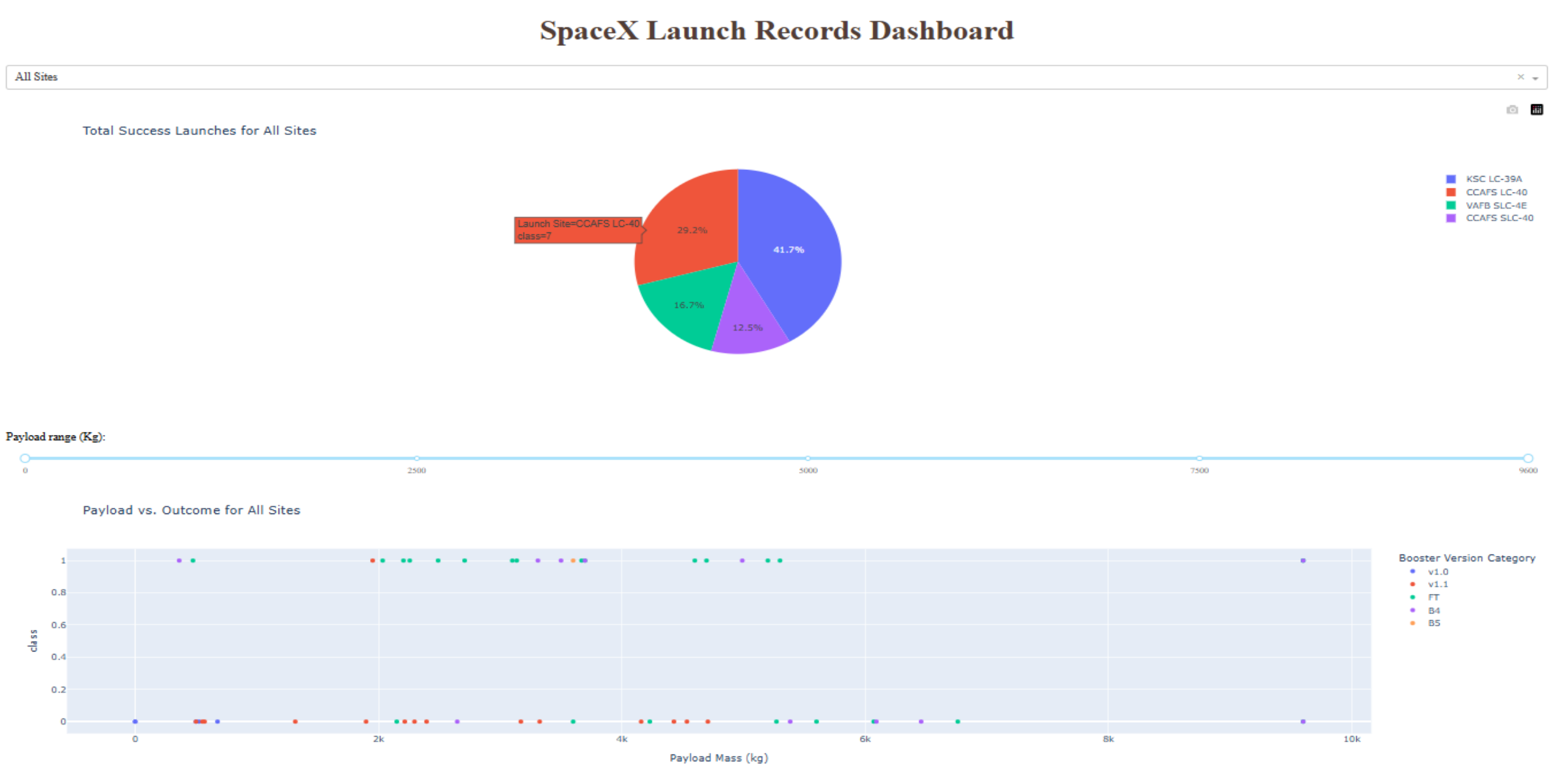
MARKER FOR KANSAS CITY

[110]:



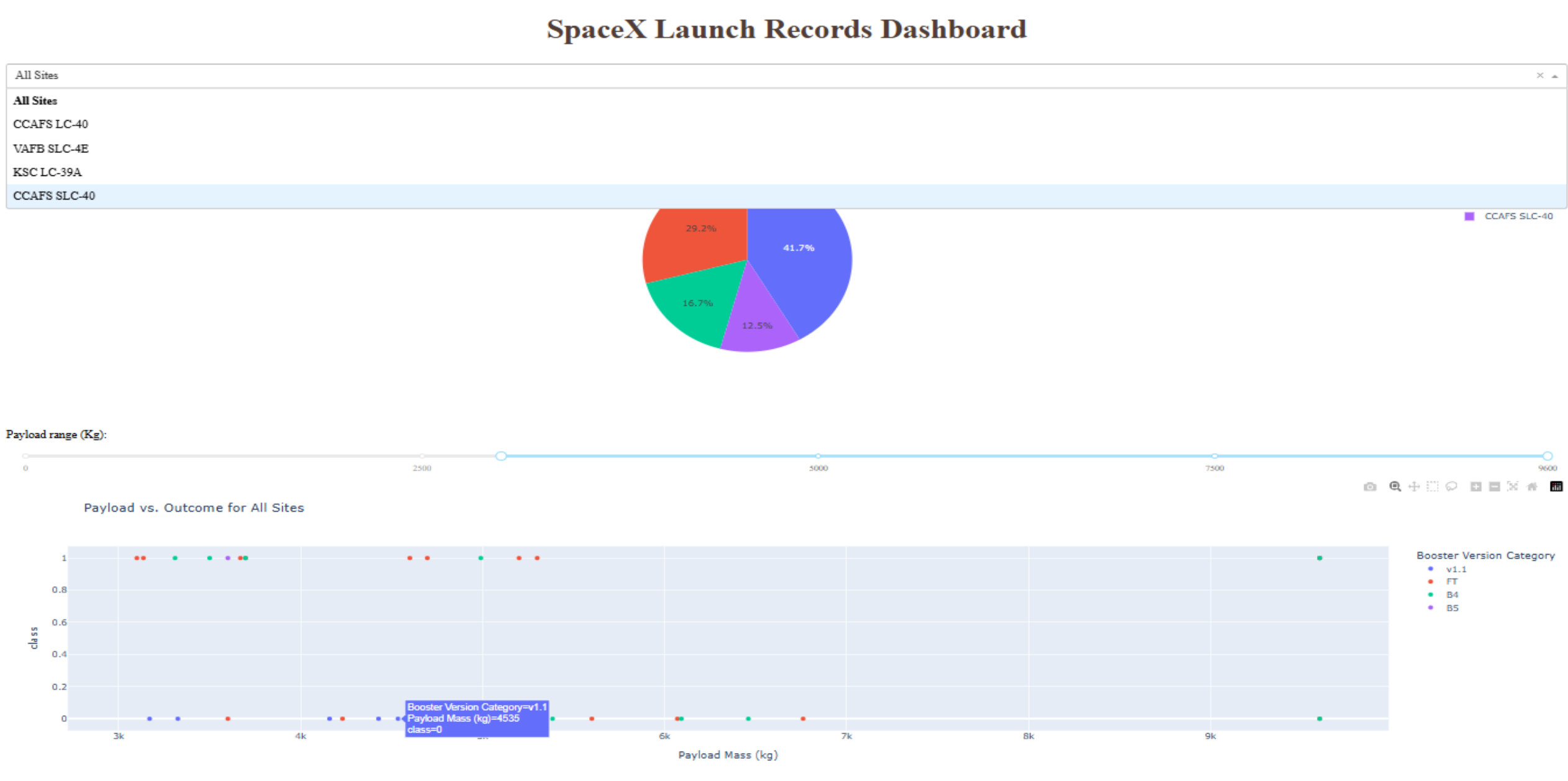
RESULT

DASH VISUALIZATION



RESULTS

DASH VISUALIZATION



RESULTS

PREDICTIVE ANALYSIS RESULTS

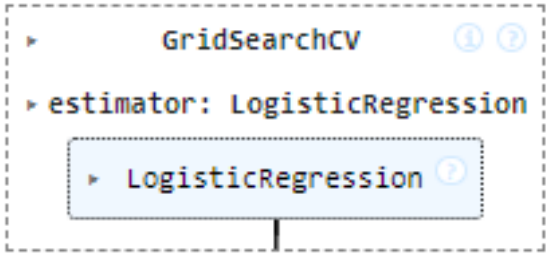
TASK 4

Create a logistic regression object then create a GridSearchCV object `logreg_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

[14]:
```
parameters ={'C':[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}
```

[15]:
```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()
logreg_cv=GridSearchCV(lr, parameters, cv=10)
logreg_cv.fit(X_train, Y_train)
```

[15]:



We output the `GridSearchCV` object for logistic regression. We display the best parameters using the data attribute `best_params_` and the accuracy on the validation data using the data attribute `best_score_`.

[17]:
```
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
```

RESULTS

PREDICTIVE ANALYSIS RESULTS

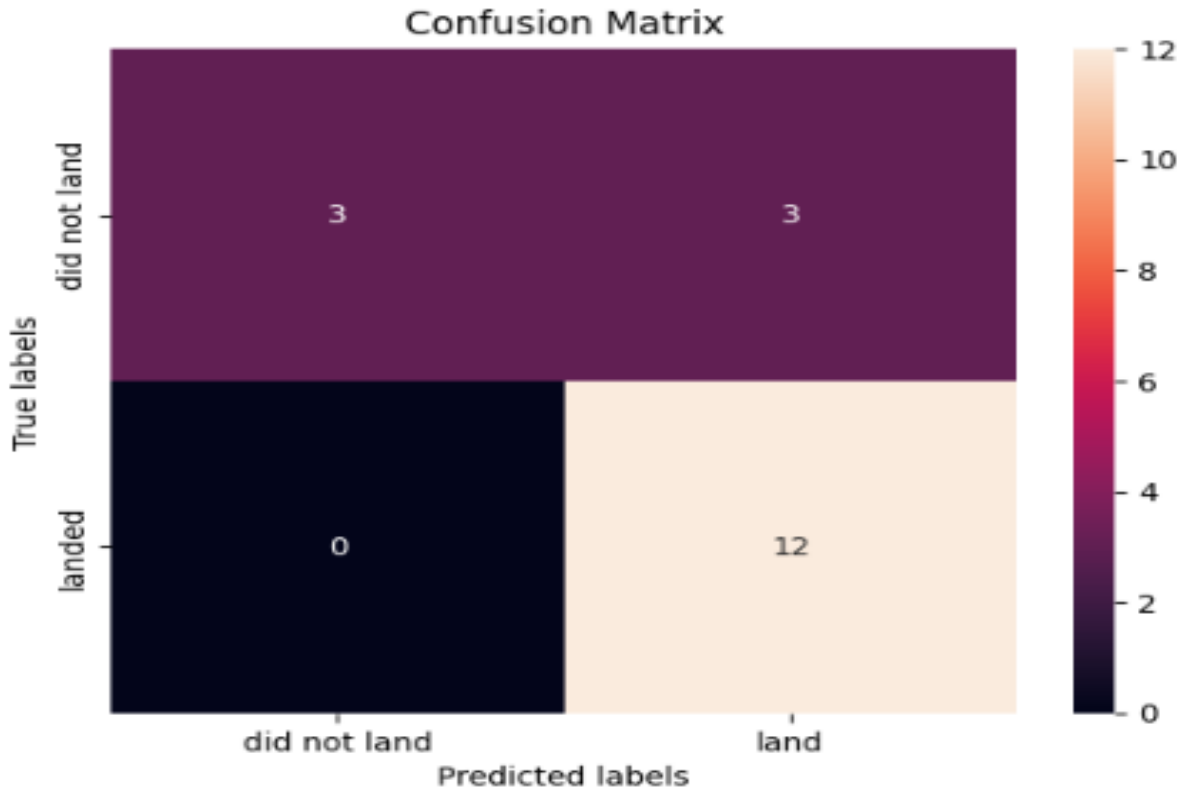
Code

[23]:
```
accuracy = logreg_cv.best_estimator_.score(X_test, Y_test)
accuracy
```

[23]:
```
0.8333333333333334
```

```
Lets look at the confusion matrix:
```

[24]:
```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



SKILLS NETWORK

RESULTS

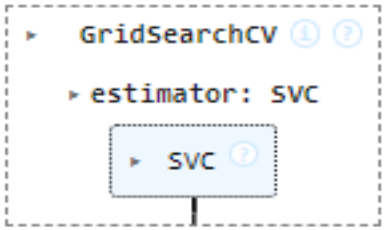
PREDICTIVE ANALYSIS RESULTS

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`.

[42]:
```
parameters1 = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
               'C': np.logspace(-3, 3, 5),
               'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
```

[43]:
```
svm_cv=GridSearchCV(svm, parameters1, cv=10)
svm_cv.fit(X_train, Y_train)
```

[43]:



[44]:
```
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
```

TASK 7

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

[45]:
```
accuracy1 = svm_cv.best_estimator_.score(X_test, Y_test)
accuracy1
```

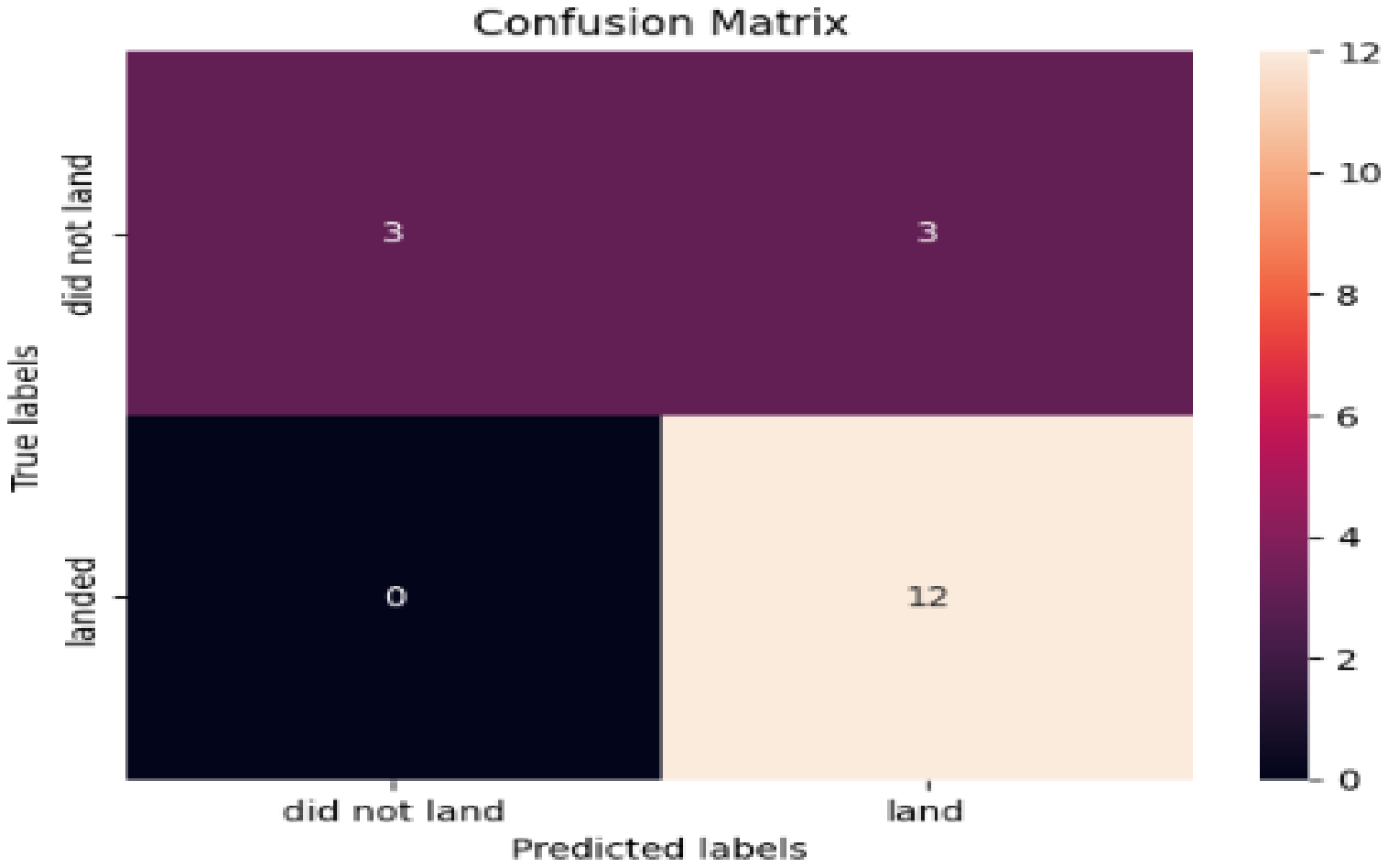
[45]:
```
0.8333333333333334
```

RESULTS

PREDICTIVE ANALYSIS RESULTS

We can plot the confusion matrix

[46]:
```
yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



RESULTS

PREDICTIVE ANALYSIS RESULTS

[21]:
```
parameters2 = {'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()
```

[27]:
```
tree_cv=GridSearchCV(tree, parameters2, cv=10)
tree_cv.fit(X_train, Y_train)
```

```
C:\Users\THOMAS ABUI\AppData\Roaming\Python\Python311\site-packages\sklearn\model_selection\_validation.py:547: FitFailedWarning:
3240 fits failed out of a total of 6480.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
3240 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\THOMAS ABUI\AppData\Roaming\Python\Python311\site-packages\sklearn\model_selection\_validation.py", line 895, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\THOMAS ABUI\AppData\Roaming\Python\Python311\site-packages\sklearn\base.py", line 1467, in wrapper
    estimator._validate_params()
  File "C:\Users\THOMAS ABUI\AppData\Roaming\Python\Python311\site-packages\sklearn\base.py", line 666, in _validate_params
    validate_parameter_constraints(
  File "C:\Users\THOMAS ABUI\AppData\Roaming\Python\Python311\site-packages\sklearn\utils\_param_validation.py", line 95, in validate_parameter_const
raints
    raise InvalidParameterError(
```

[28]:
```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split':
2, 'splitter': 'random'}
accuracy : 0.875
```

RESULTS

PREDICTIVE ANALYSIS RESULTS

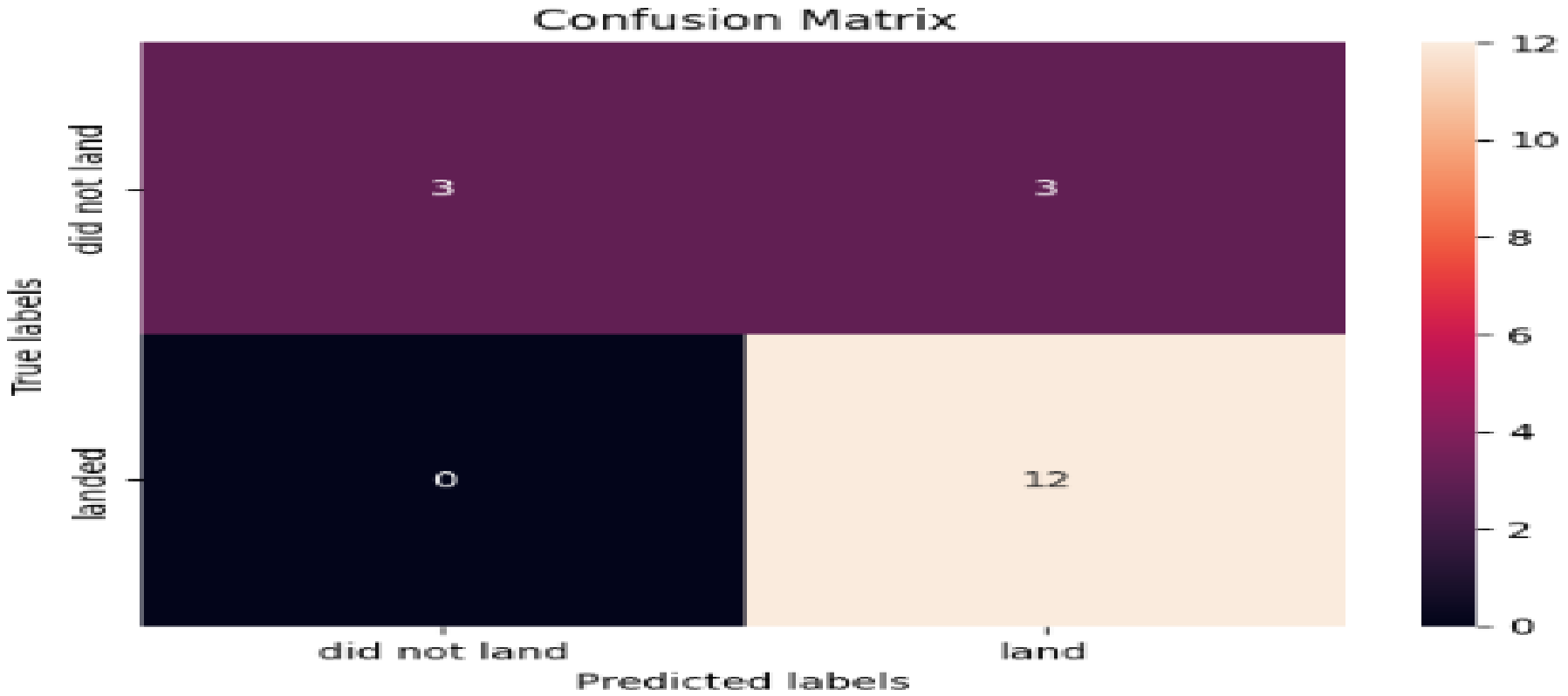
Markdown

[29]:
```
accuracy2 = tree_cv.score(X_test, Y_test)
accuracy2
```

[29]:
```
0.8333333333333334
```

We can plot the confusion matrix

[30]:
```
yhat_tr = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat_tr)
```



RESULTS

PREDICTIVE ANALYSIS RESULTS

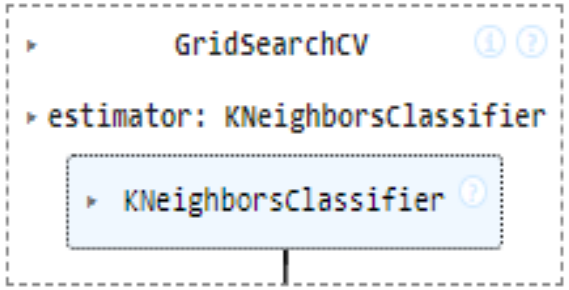
Create a k nearest neighbors object then create a `GridSearchCV` object `knn_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

[31]:
```
parameters3 = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
               'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
               'p': [1,2]}

KNN = KNeighborsClassifier()
```

[32]:
```
knn_cv=GridSearchCV(KNN, parameters3, cv=10)
knn_cv.fit(X_train, Y_train)
```

[32]:



[33]:
```
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
```

RESULTS

PREDICTIVE ANALYSIS RESULTS

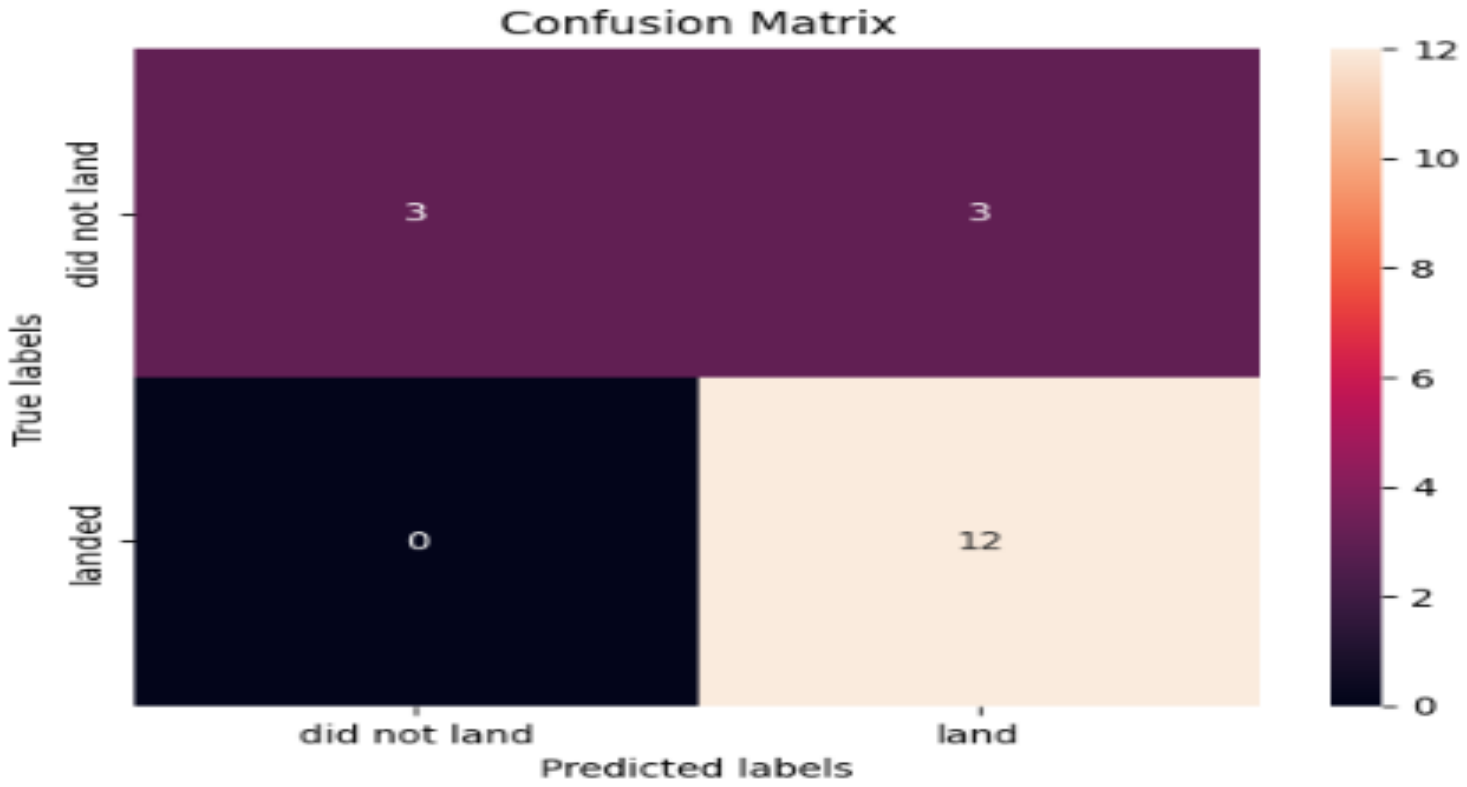
Markdown

[34]:
```
accuracy3 = knn_cv.best_estimator_.score(X_test, Y_test)
accuracy3
```

[34]:
```
0.8333333333333334
```

We can plot the confusion matrix

[35]:
```
yhat_kn = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat_kn)
```



RESULTS

PREDICTIVE ANALYSIS RESULTS

TASK 12

Find the method performs best:

[36]:
```
Report = {'Models':['KNN','Decison_Tree','SVM','Logistic_regression'],
          'Accuracy_Score':[accuracy3,accuracy2,accuracy1,accuracy]

           }
Report=pd.DataFrame(Report)
Report
```

[36]:

	Models	Accuracy_Score
0	KNN	0.833333
1	Decison_Tree	0.833333
2	SVM	0.833333
3	Logistic_regression	0.833333

DISCUSSION

All four models achieved an accuracy score of approximately 83.33%. This indicates that each model has a similar capability in predicting the landing success of the Falcon 9's first stage.

1. **K-Nearest Neighbors (KNN)**: This model relies on the similarity between data points. Its high accuracy suggests that historical launch data have distinct patterns that can be effectively used for prediction.
2. **Decision Tree**: This model works by making decisions based on the values of input features. Its performance indicates that the criteria used to make these decisions are robust and relevant to the outcome.
3. **Support Vector Machine (SVM)**: SVM tries to find the best boundary that separates different classes. Its comparable accuracy shows that the decision boundary between successful and unsuccessful landings is well-defined in the feature space.
4. **Logistic Regression**: This model predicts probabilities of binary outcomes. Its effectiveness suggests that the probability distribution of features between successful and unsuccessful landings is distinguishable.

CONCLUSION

- The models developed provide a reliable prediction (83.33% accuracy) of the Falcon 9 first stage landing success. This predictive capability is crucial for companies looking to compete with SpaceX by estimating the potential cost savings and setting competitive prices for rocket launches.
- **Implications**
- By using these models, alternate companies can:
- Assess the risk and feasibility of achieving cost-effective launches by predicting the likelihood of the first stage landing success.
- Make data-driven decisions to optimize their launch strategies and cost structures.
- Develop competitive bids against SpaceX by accurately estimating potential savings from reusable first stages.

APPENDIX

- https://api.spacexdata.com/v4/launches/past