Success factors - SpaceX

Soonyoul Kwon

Xyz@gmail.com

January 18, 2024

Table of Contents

- Executive Summary
- Introduction
- Methodology
- Results
  - Visualization – Charts
  - Dashboard
- Discussion
  - Findings & Implications
- Conclusion
- Appendix

EXECUTIVE SUMMARY

- Reusing the rockets is key success factor (KSF) of SpaceX
- To increase the cost saving effect, the business should focus its resource more on launching the rockets on the area with higher landing success rate.
- Not all features has had meaningful impact to the landing success.
- There are some factors that had significant impact on the success rate
  - CCAFS SLC 40 when the payload mass over 10,000.
  - The heavier payload mass, LEO, ISS, and PO.
  - KSC LC-39A has the highest success rate.

INTRODUCTION

- SpaceX has become an industry leader based on the comparative cost effectiveness.
- This is due to the reusability of the rockets
- Thus, the successful landing of them is critical
- I would like to discuss some success factors that affected the landing performance.

METHODOLOGY

- Data collection and wrangling
- Exploratory data analysis with SQL & Interactive map with Folium / Plotly Dash dashboard
- Predictive analysis

Data collection & wrangling methodology

- Data collected from SpaceX API
- Data normalization using .jason_normalize()
- Data filtering in pandas DataFrame
- Imputing missing data (i.e. mean value for null PayloadMass)

- Data scrapping using BeautifulSoup
- And Parsing data to dictionary to DataFrame
- Exploring datasets using df.methods

EDA and interactive visual analytics methodology

SQL

- Loading data from csv
- Converting to DataFrame
- Execute sql queries using %sql

Exploratory visualization (plots)

- matplotlib
- seaborn

Folium & Dash

- Locate coordinates in map using folium
- Calculate the distances between coordinates
- Draw lines between coordinates

- Creating a layout including dropdown implementation
- Adding callback functions using inputs and outputs

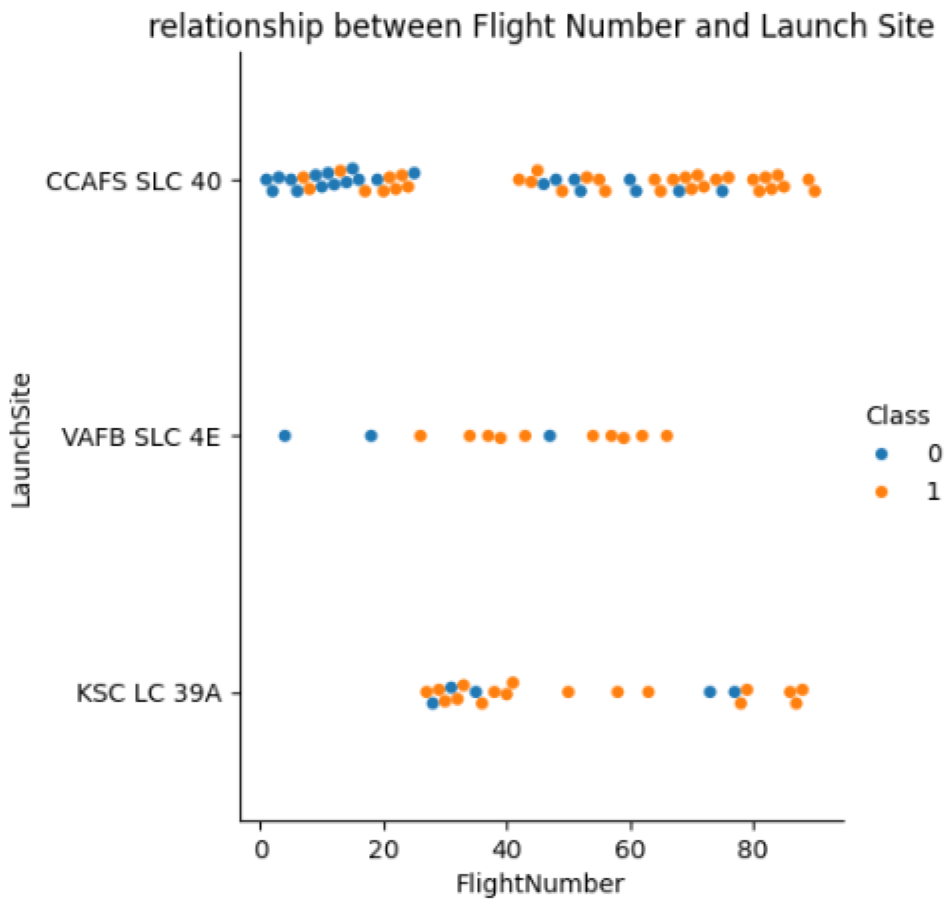
Predictive analysis methodology

- Split train and test data
- For each method
  (logistic regression, support vector machine,
  decision tree classifier, and k nearest neighbors)
  - Create object and fit the object to the train data to get the best parameters and accuracy score
  - Calculate the accuracy on the test data
  - Visualize false-positive/negative in confusion matrix

Results

- EDA with visualization
- EDA with SQL
- Interactive map with Folium
- plotly Dash dashboard
- Predictive analysis (classification)

EDA with visualization results 1

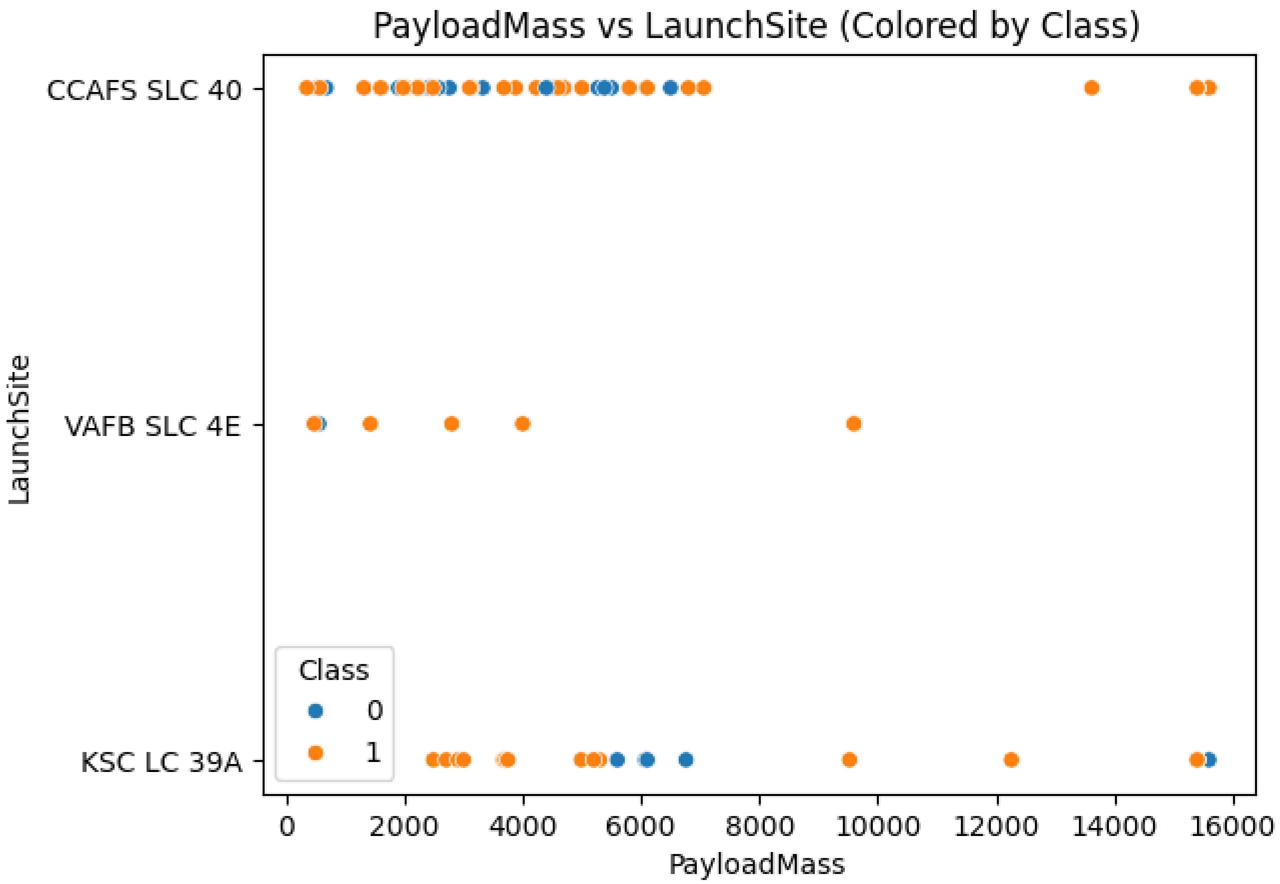


Flight Number vs. Launch site scatter chart

Overall, the higher the flight number, the higher chance of successful landing.

This pattern was more obvious for two launch sites: CCAFS SLC 40 and VAFB SLC 4E.

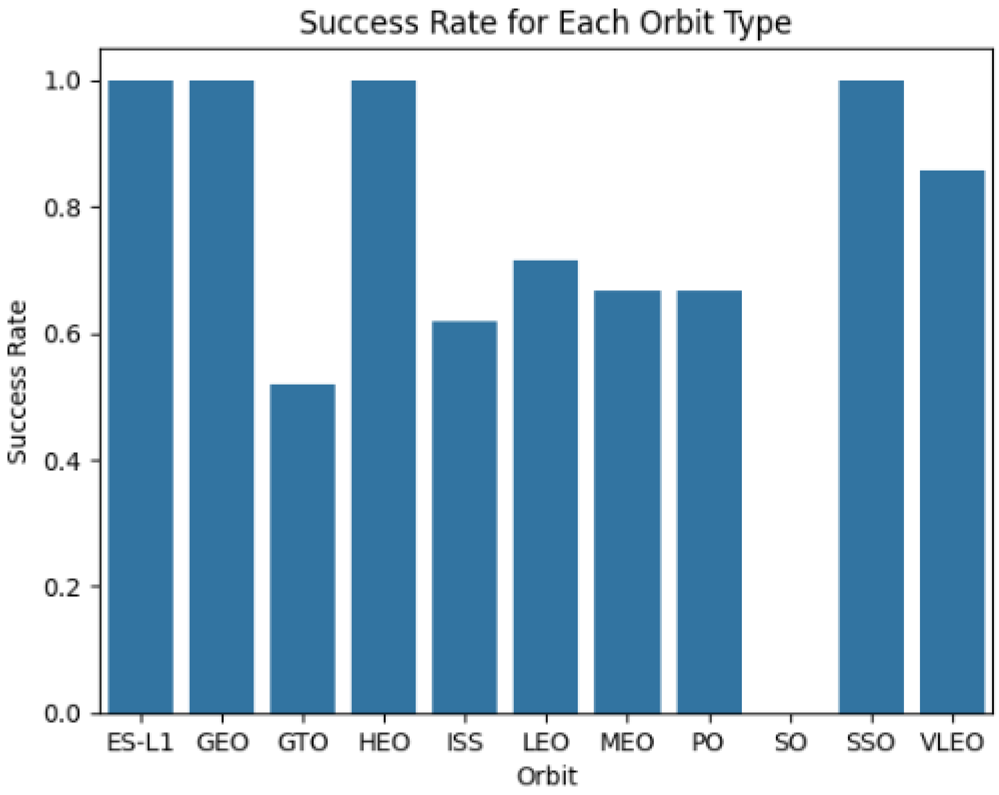
EDA with visualization results 2



Payload vs. Launch site scatter chart

It is noticeable that it was 100% success for the heavy load over 10,000 on the launch site CCAFS SLC 40.

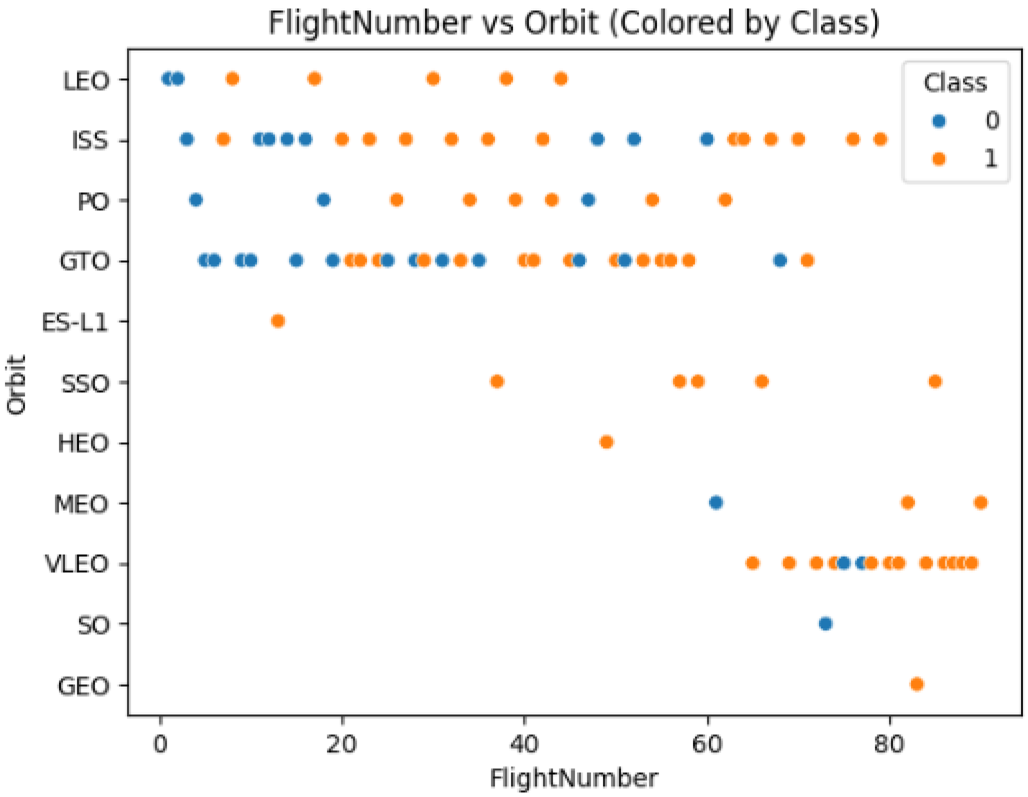
EDA with visualization results 3



Success rate vs. Orbit type bar chart

ES-L1, GEO, HEO and SSO had the highest success rates among all orbits.

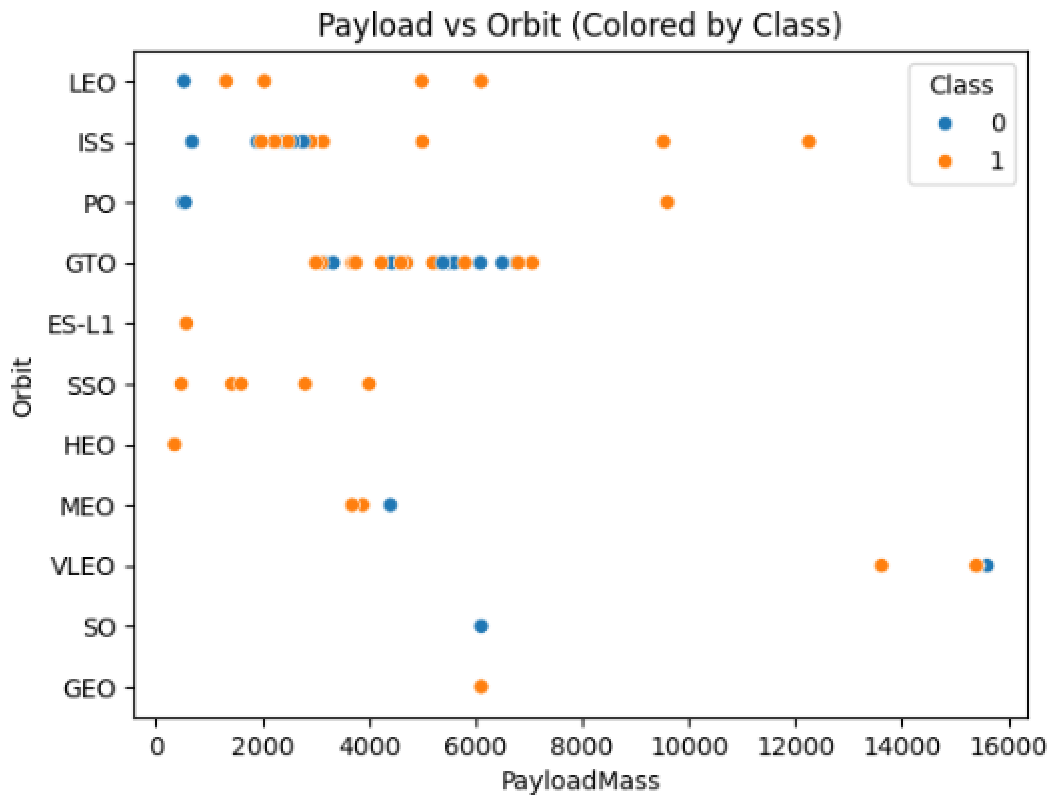
EDA with visualization results 4



Flight Number vs. Orbit type scatter chart

LEO orbit the Success appears related to the number of flights

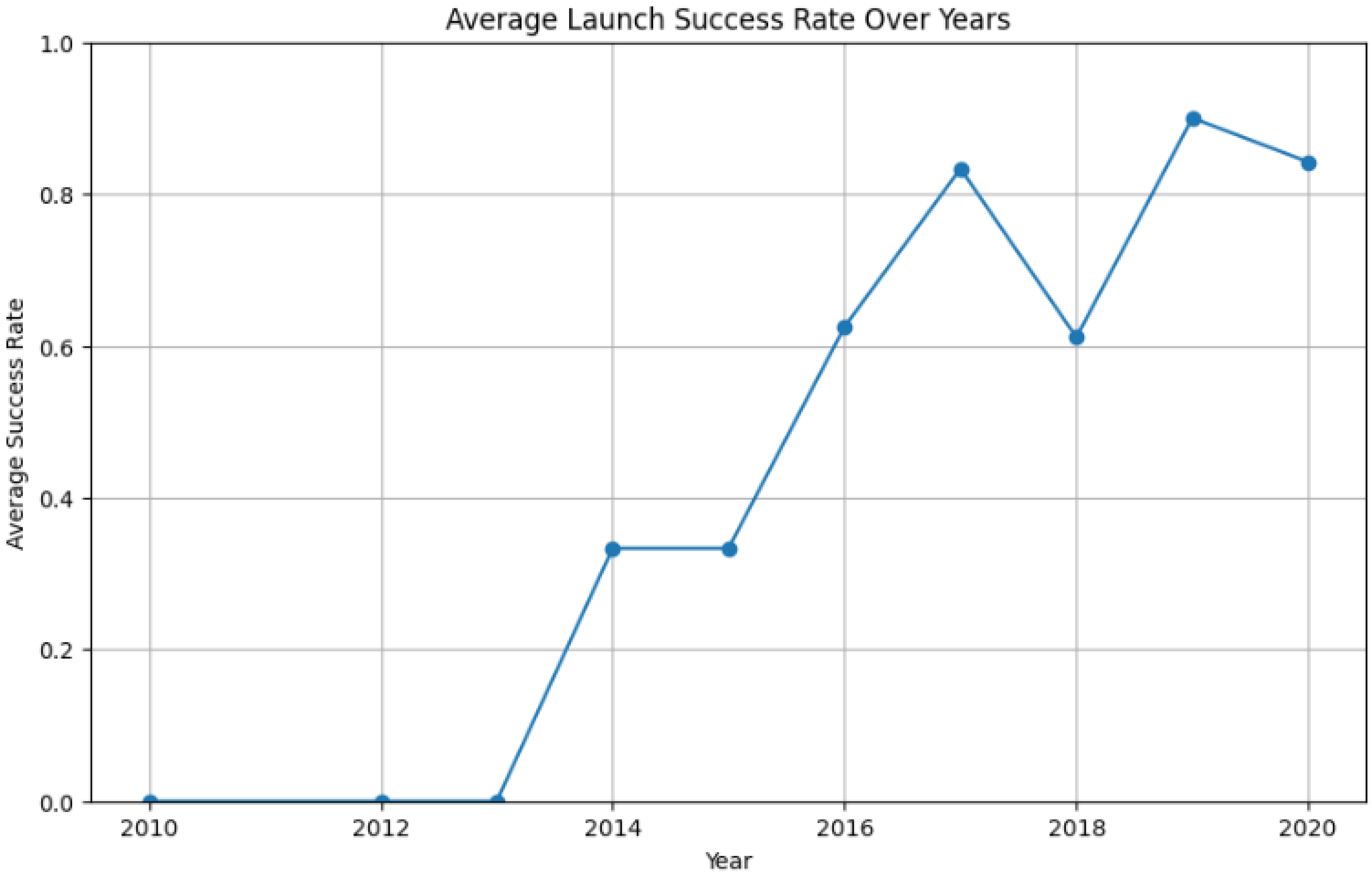
EDA with visualization results 5



Payload vs. Orbit type scatter chart

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

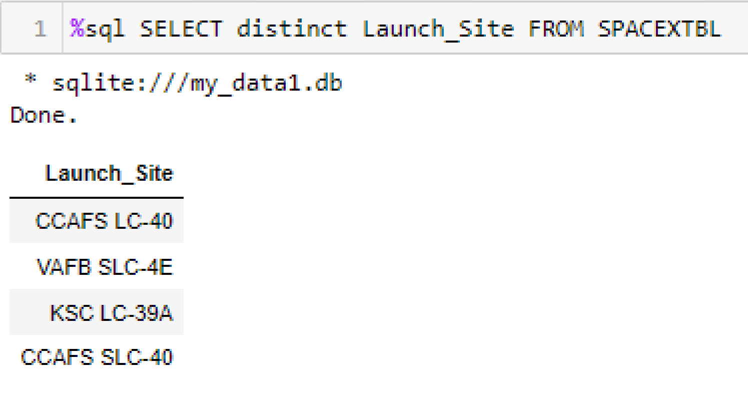
EDA with visualization results 6



Launch success yearly trend line

Rates kept increasing

EDA with SQL results 1



```
%sql SELECT distinct Launch_Site FROM SPACEXTBL
```

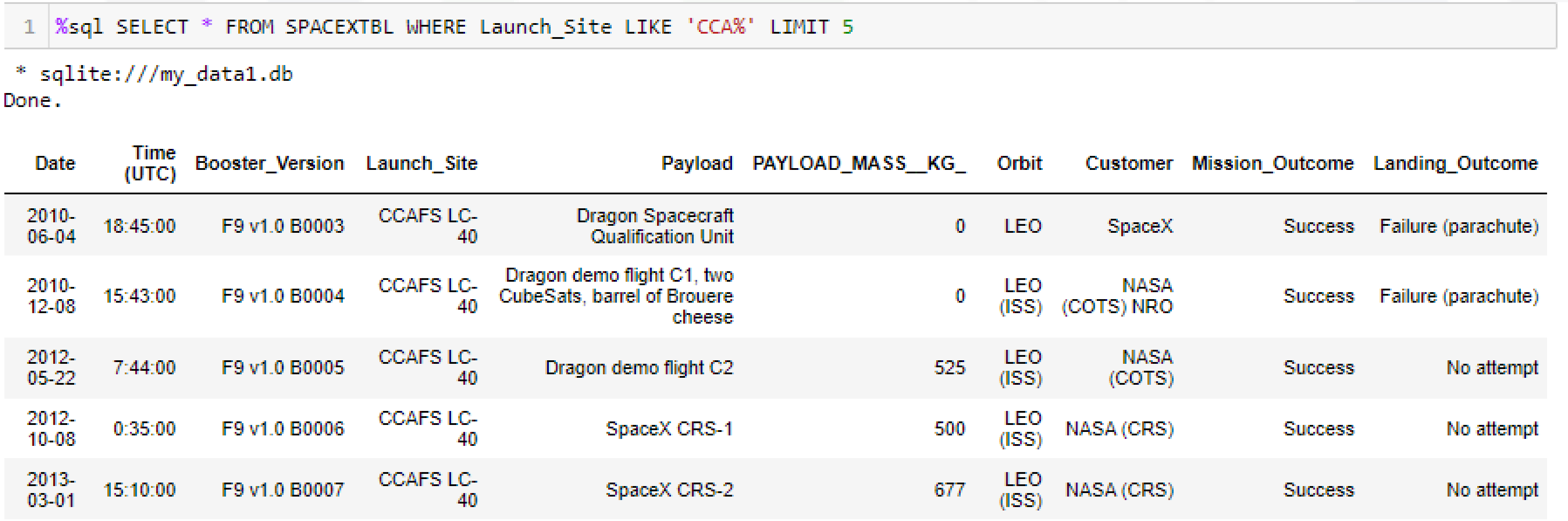
```
 * sqlite:///my_data1.db
Done.
```

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

All launch site names: Find the names of the unique sites

EDA with SQL results 2

Launch site names begin with `CCA`: Find all launch site names begin with `CCA`



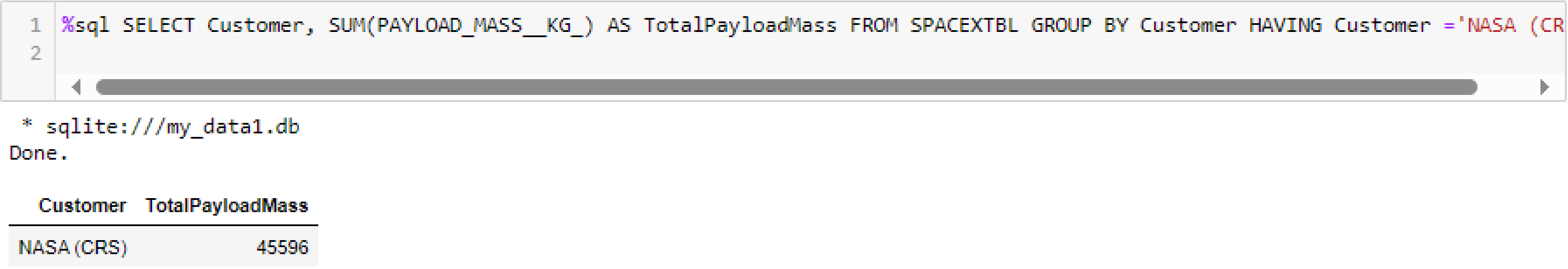
```
%sql SELECT * FROM SPACEXTBL WHERE Launch_Site LIKE 'CCA%' LIMIT 5
```

```
 * 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

EDA with SQL results 3

Total payload mass: Calculate the total payload carried by booster from NASA



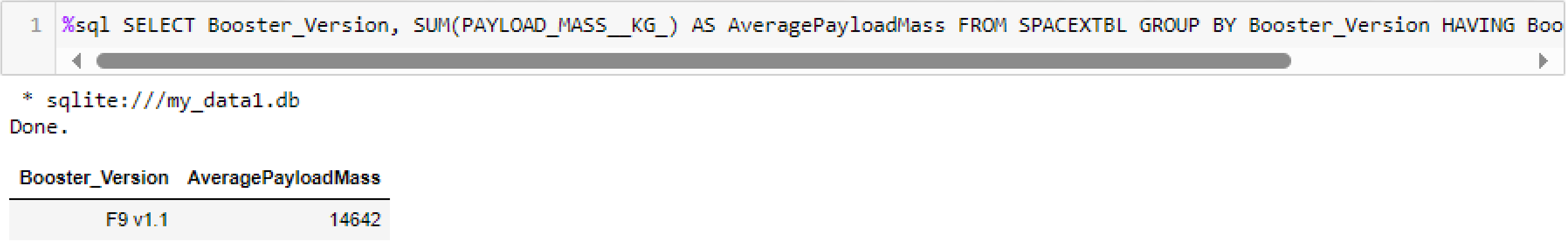
```
%sql SELECT Customer, SUM(PAYLOAD_MASS__KG_) AS TotalPayloadMass FROM SPACEXTBL GROUP BY Customer HAVING Customer ='NASA (CR
```

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

Customer	TotalPayloadMass
NASA (CRS)	45596

EDA with SQL results 4

Average payload mass by F9 v1.1: Calculate the average payload mass carried by booster F9 v1.1



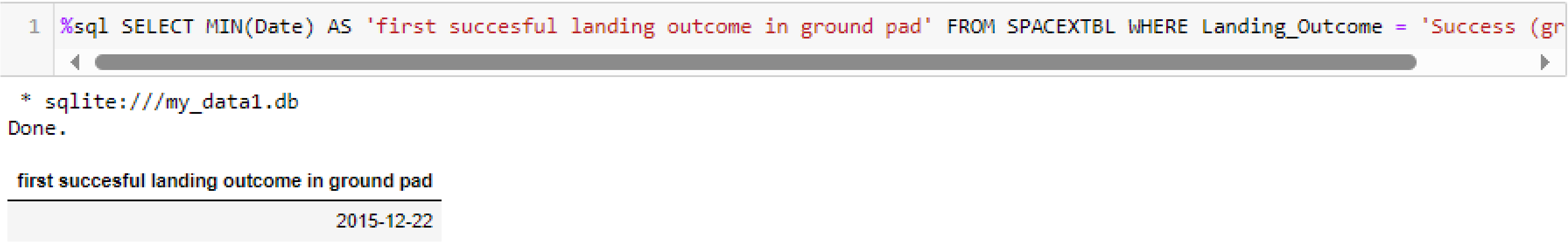
```
%sql SELECT Booster_Version, SUM(PAYLOAD_MASS__KG_) AS AveragePayloadMass FROM SPACEXTBL GROUP BY Booster_Version HAVING Boo
```

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

Booster_Version	AveragePayloadMass
F9 v1.1	14642

EDA with SQL results 5

First successful ground landing date: find the date when the first successful landing outcome in ground pad



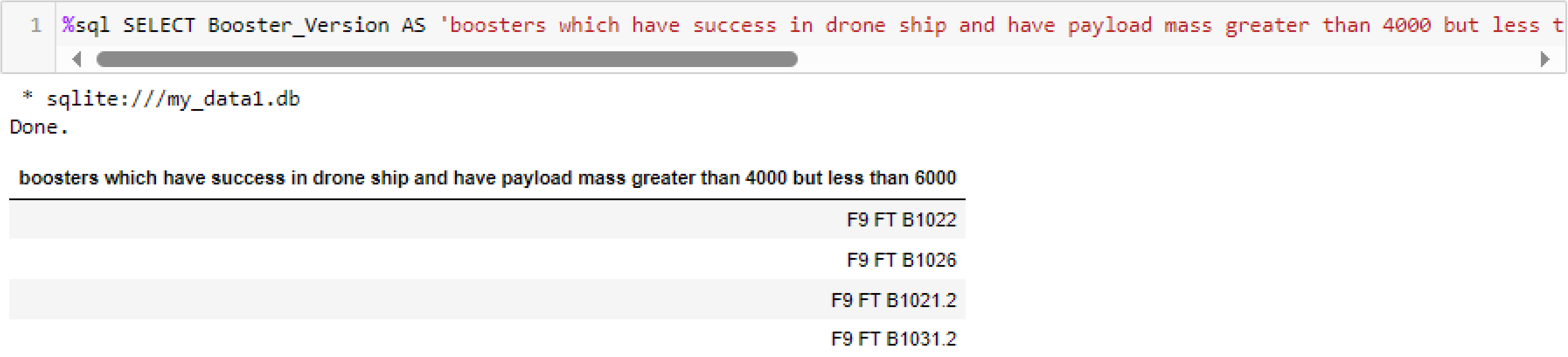
```
%sql SELECT MIN(Date) AS 'first succesful landing outcome in ground pad' FROM SPACEXTBL WHERE Landing_Outcome = 'Success (gr
```

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

first succesful landing outcome in ground pad
2015-12-22

EDA with SQL results 6

Successful drone ship landing with payload between 4000 and 6000: List the names of boosters which have success in drone ship and have payload greater than 4000 but less than 6000



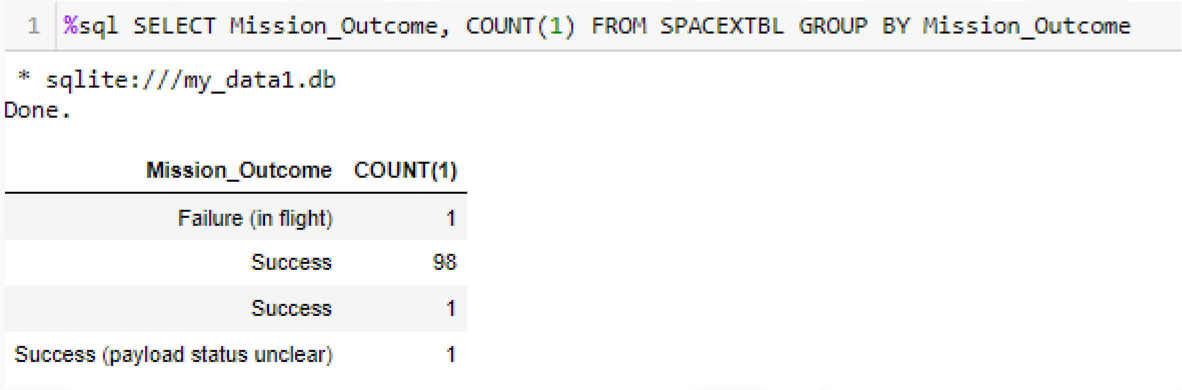
```
%sql SELECT Booster_Version AS 'boosters which have success in drone ship and have payload mass greater than 4000 but less t
```

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

boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

EDA with SQL results 7

Total number of successful and failure mission outcomes: Calculate the total number of successful and failure mission outcomes



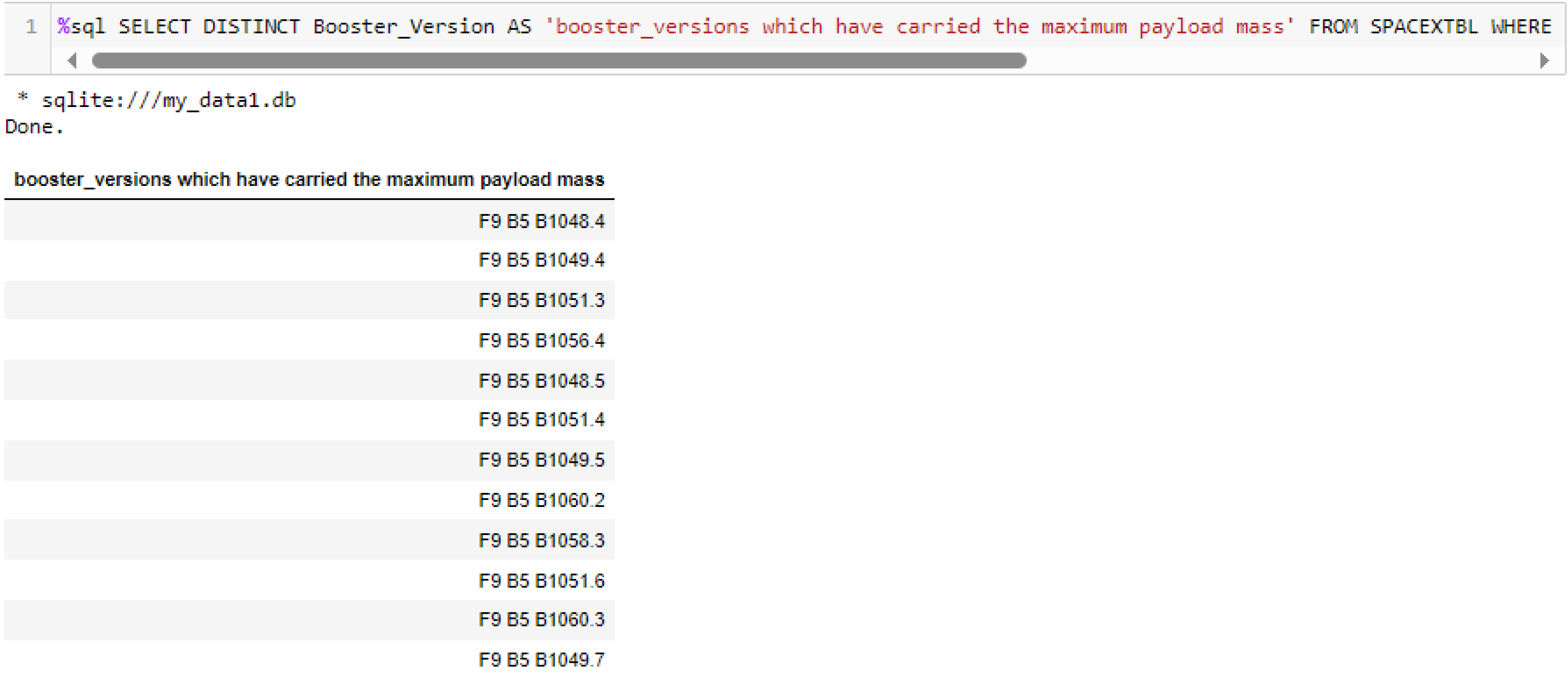
```
%sql SELECT Mission_Outcome, COUNT(1) FROM SPACEXTBL GROUP BY Mission_Outcome
```

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

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

EDA with SQL results 8

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



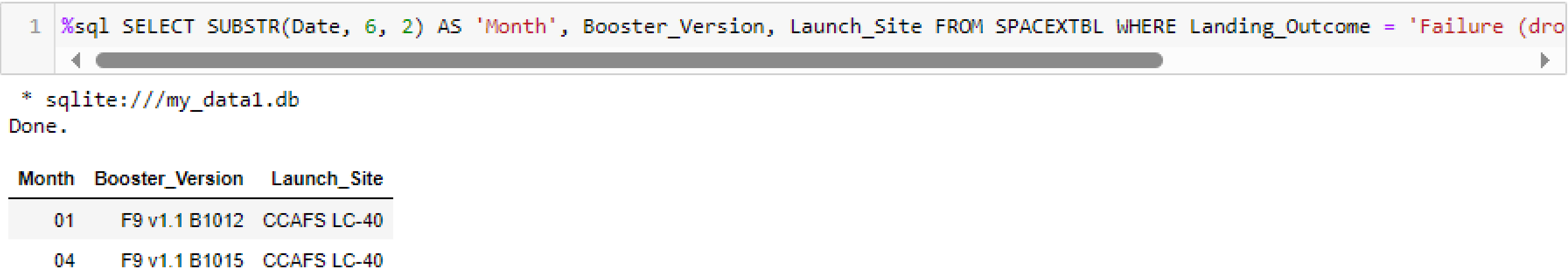
```
%sql SELECT DISTINCT Booster_Version AS 'booster_versions which have carried the maximum payload mass' FROM SPACEXTBL WHERE
```

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

booster_versions which have carried the maximum payload mass
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

EDA with SQL results 9

2015 launch records: List the launch records for months in 2015



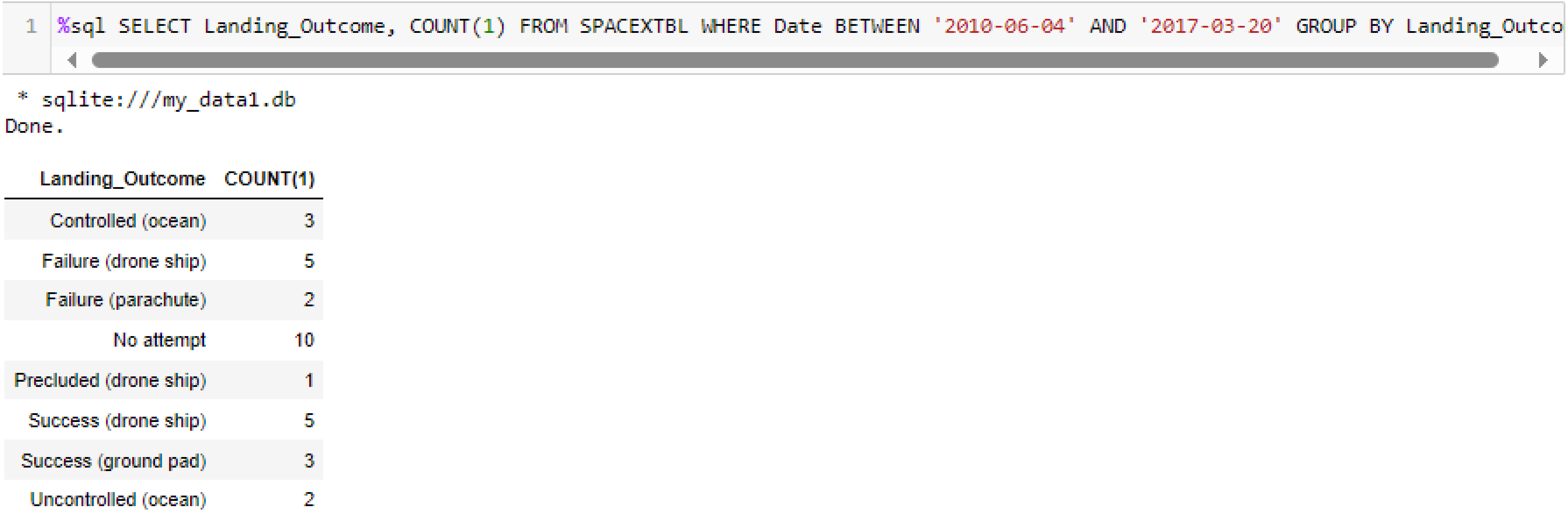
```
%sql SELECT SUBSTR(Date, 6, 2) AS 'Month', Booster_Version, Launch_Site FROM SPACEXTBL WHERE Landing_Outcome = 'Failure (dro
```

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

Month	Booster_Version	Launch_Site
01	F9 v1.1 B1012	CCAFS LC-40
04	F9 v1.1 B1015	CCAFS LC-40

EDA with SQL results 10

Rank success count between 2010-06-04 and 2017-03-20: Rank the count of successful landings between 2010-06-04 and 2017-03-20



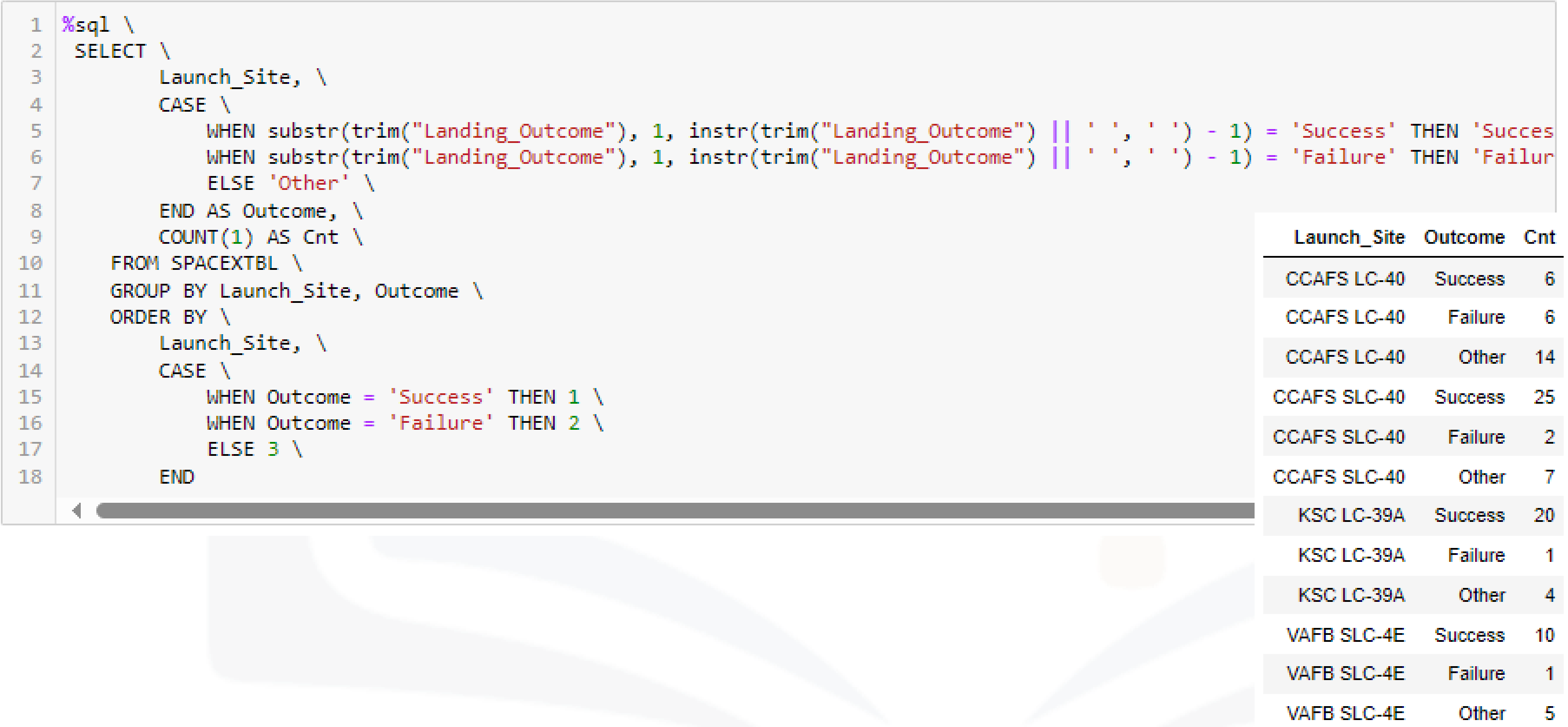
```
%sql SELECT Landing_Outcome, COUNT(1) FROM SPACEXTBL WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY Landing_Outco
```

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

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

EDA with SQL results - Extra

KSC LC-39A has the highest success rate.

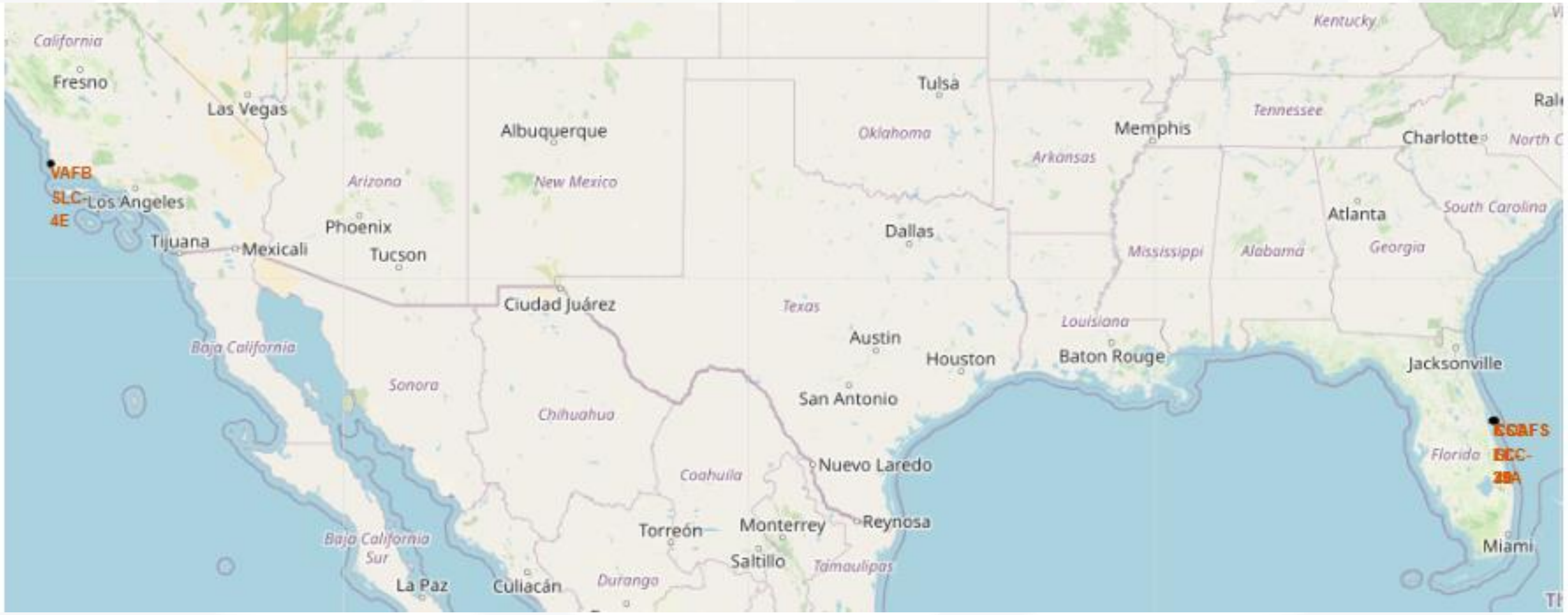


```
%sql \
 SELECT \
        Launch_Site, \
        CASE \
            WHEN substr(trim("Landing_Outcome"), 1, instr(trim("Landing_Outcome") || ' ', ' ') - 1) = 'Success' THEN 'Succes
            WHEN substr(trim("Landing_Outcome"), 1, instr(trim("Landing_Outcome") || ' ', ' ') - 1) = 'Failure' THEN 'Failur
            ELSE 'Other' \
        END AS Outcome, \
        COUNT(1) AS Cnt \
    FROM SPACEXTBL \
    GROUP BY Launch_Site, Outcome \
    ORDER BY \
        Launch_Site, \
        CASE \
            WHEN Outcome = 'Success' THEN 1 \
            WHEN Outcome = 'Failure' THEN 2 \
            ELSE 3 \
        END
```

Launch_Site	Outcome	Cnt
CCAFS LC-40	Success	6
CCAFS LC-40	Failure	6
CCAFS LC-40	Other	14
CCAFS SLC-40	Success	25
CCAFS SLC-40	Failure	2
CCAFS SLC-40	Other	7
KSC LC-39A	Success	20
KSC LC-39A	Failure	1
KSC LC-39A	Other	4
VAFB SLC-4E	Success	10
VAFB SLC-4E	Failure	1
VAFB SLC-4E	Other	5

Interactive map with Folium results 1

all launch sites’ markers on a global map

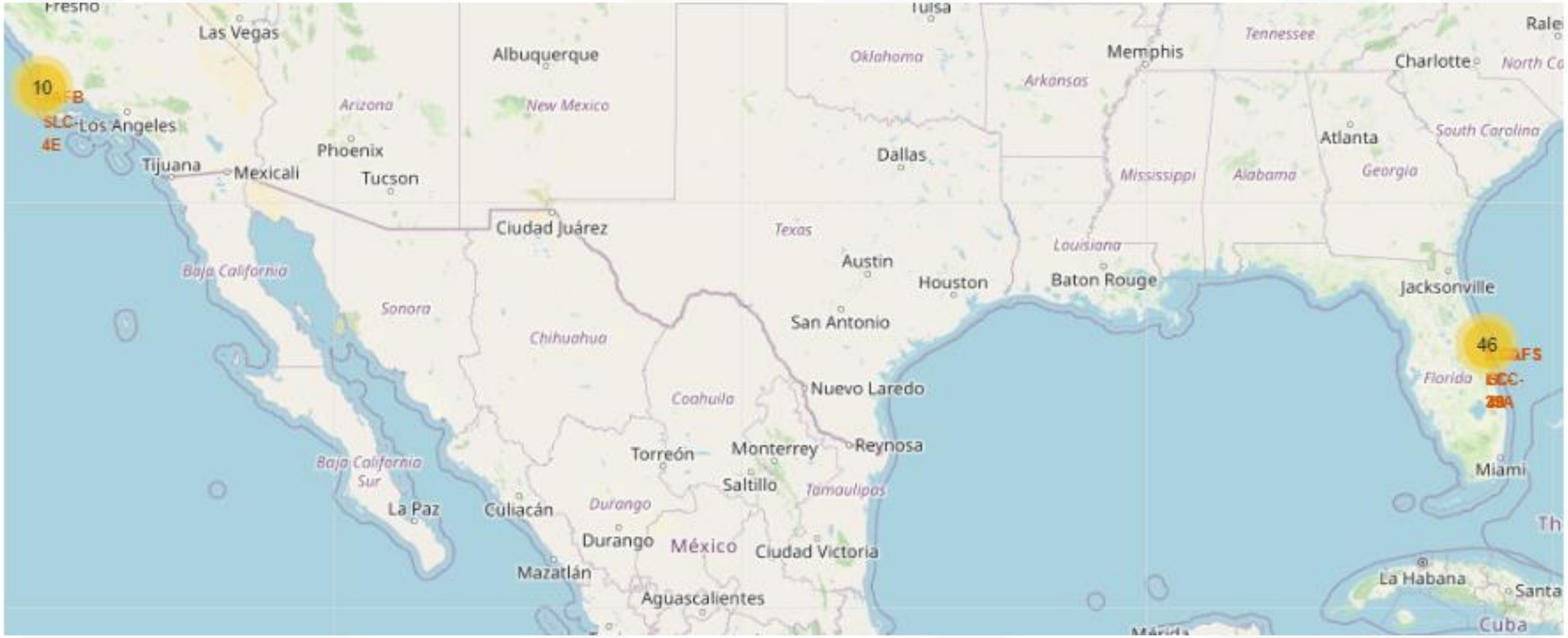


IBM Developer

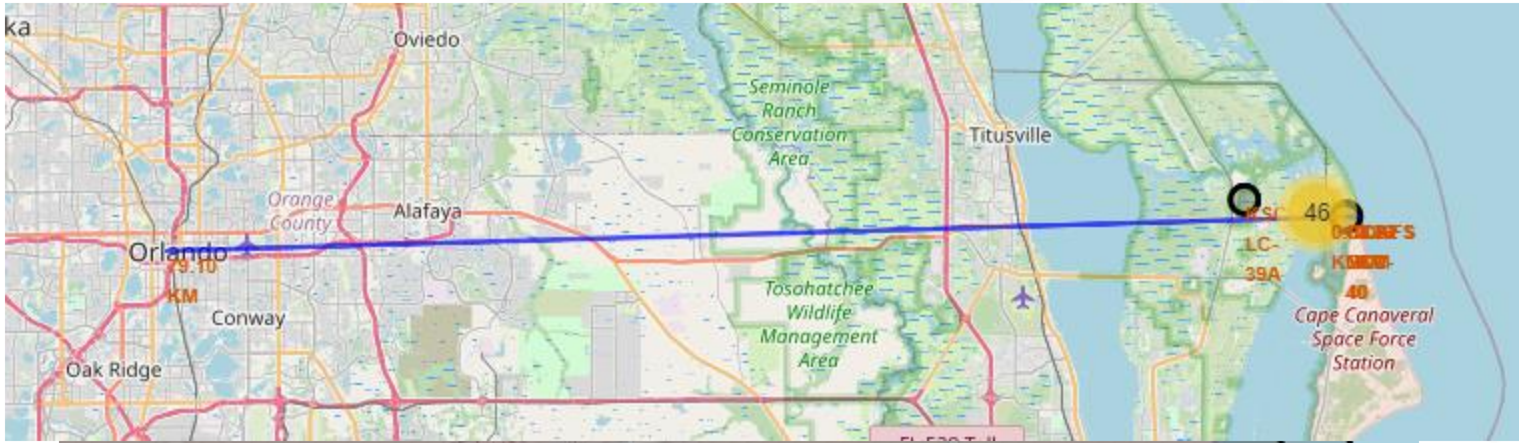
SKILLS NETWORK

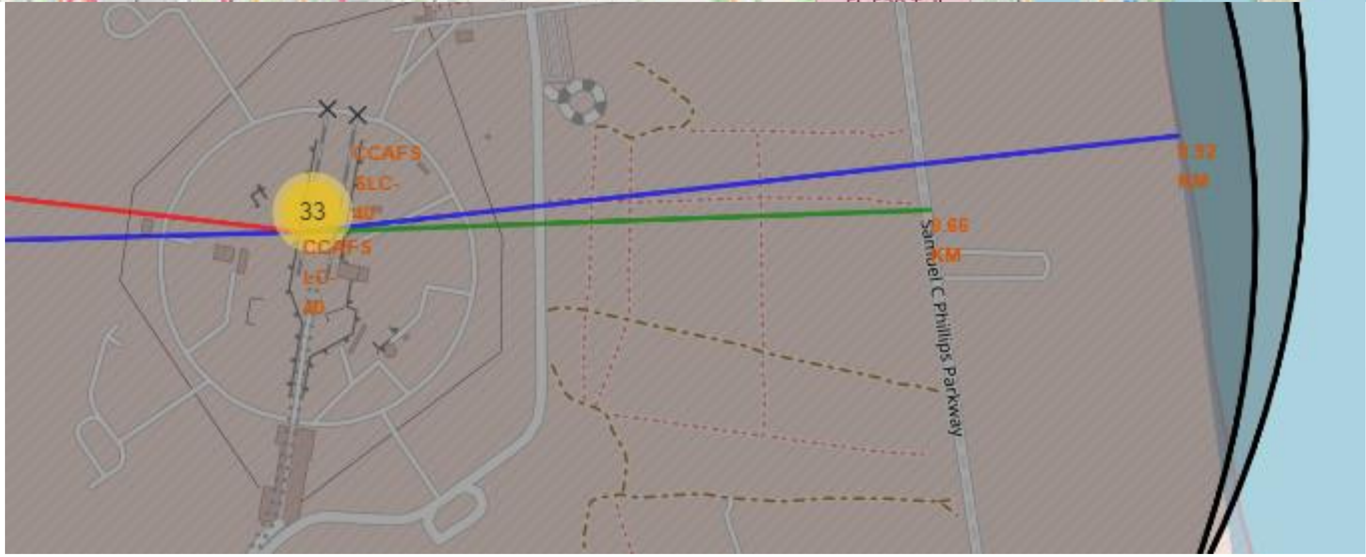
Interactive map with Folium results 2

all launch records per site on the map



Interactive map with Folium results 3

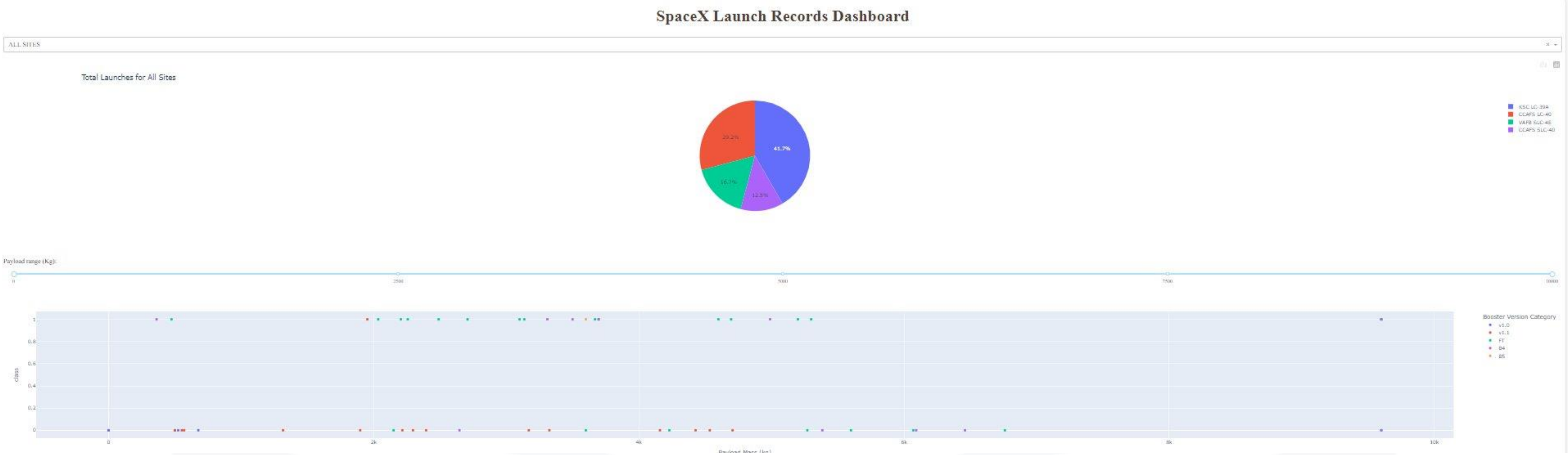

Oviedo
Seminole Ranch Conservation Area
Titusville
Orange County
Alafaya
Orlando
9.10 KM
Conway
Oak Ridge
Tosohatchee Wildlife Management Area
46
LC-39A
CCAFS
40
Cape Canaveral Space Force Station



launch sites’ proximities

- Distances from CCAFS LC 40
    - To the closest shore: 0.92km
    - To the closest highway: 0.66km
    - To the closest city (Orlando, FL): 9.10km

plotly Dash dashboard results



SpaceX Launch Records Dashboard

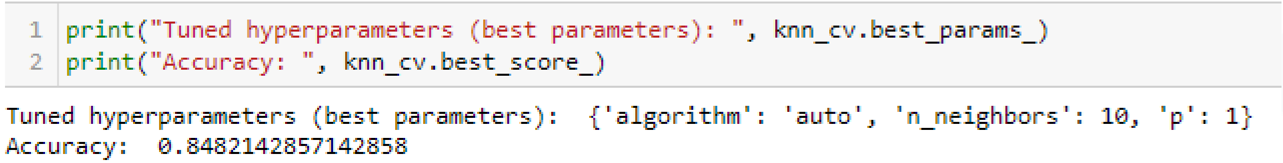
ALL SITES

Total Launches for All Sites

Payload range (Kg):

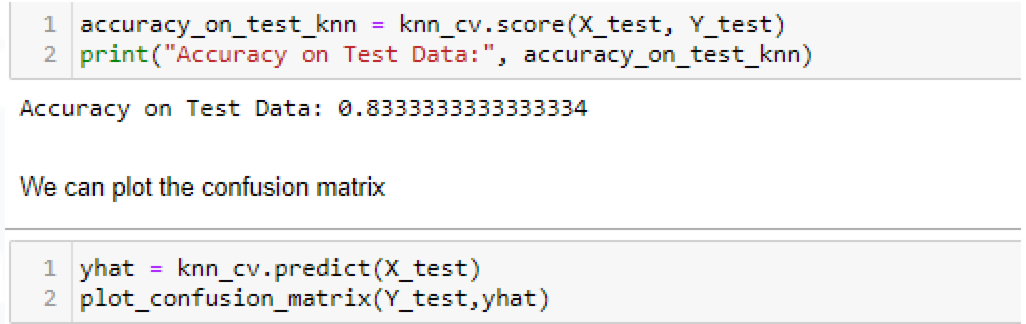
Predictive analysis (classification) results 1

KNN model performance



```
print("Tuned hyperparameters (best parameters): ", knn_cv.best_params_)
print("Accuracy: ", knn_cv.best_score_)
```

```
Tuned hyperparameters (best parameters):  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
Accuracy:  0.8482142857142858
```

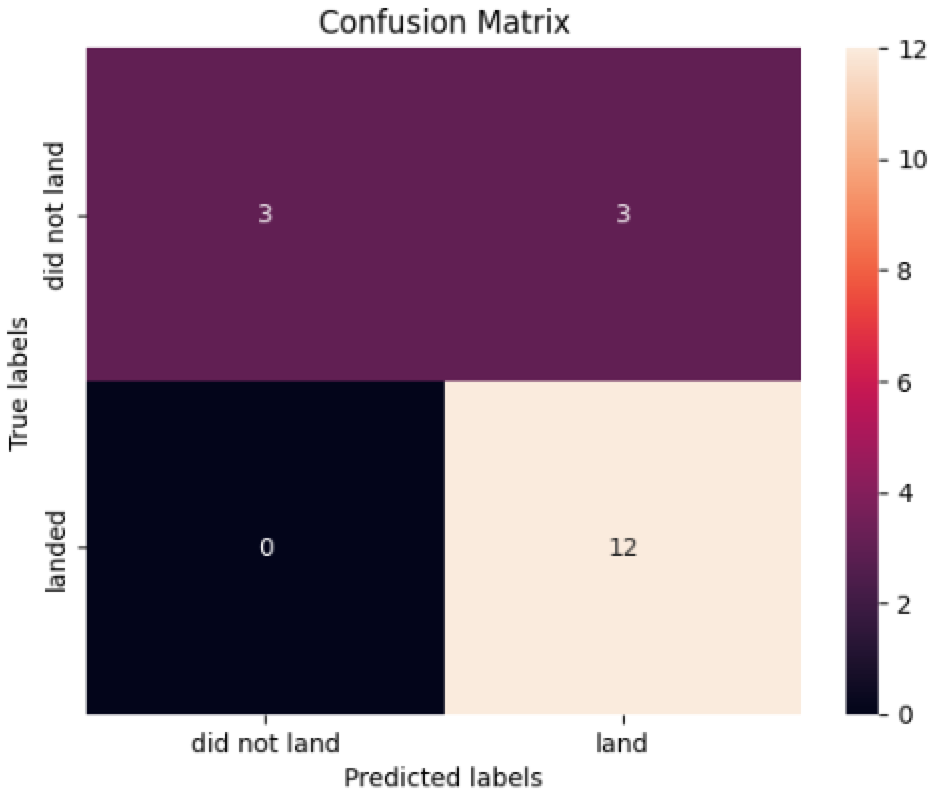


```
accuracy_on_test_knn = knn_cv.score(X_test, Y_test)
print("Accuracy on Test Data:", accuracy_on_test_knn)
```

```
Accuracy on Test Data: 0.8333333333333334
```

We can plot the confusion matrix

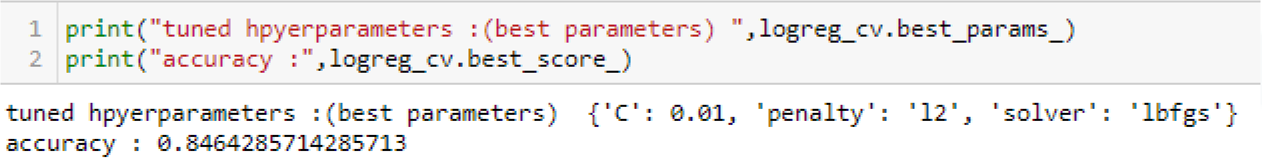
```
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



IBM Developer

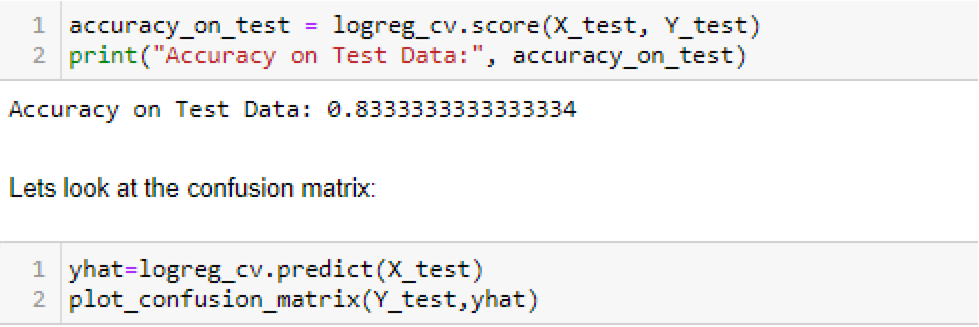
Predictive analysis (classification) results 2

Logistic regression model performance



```
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
```

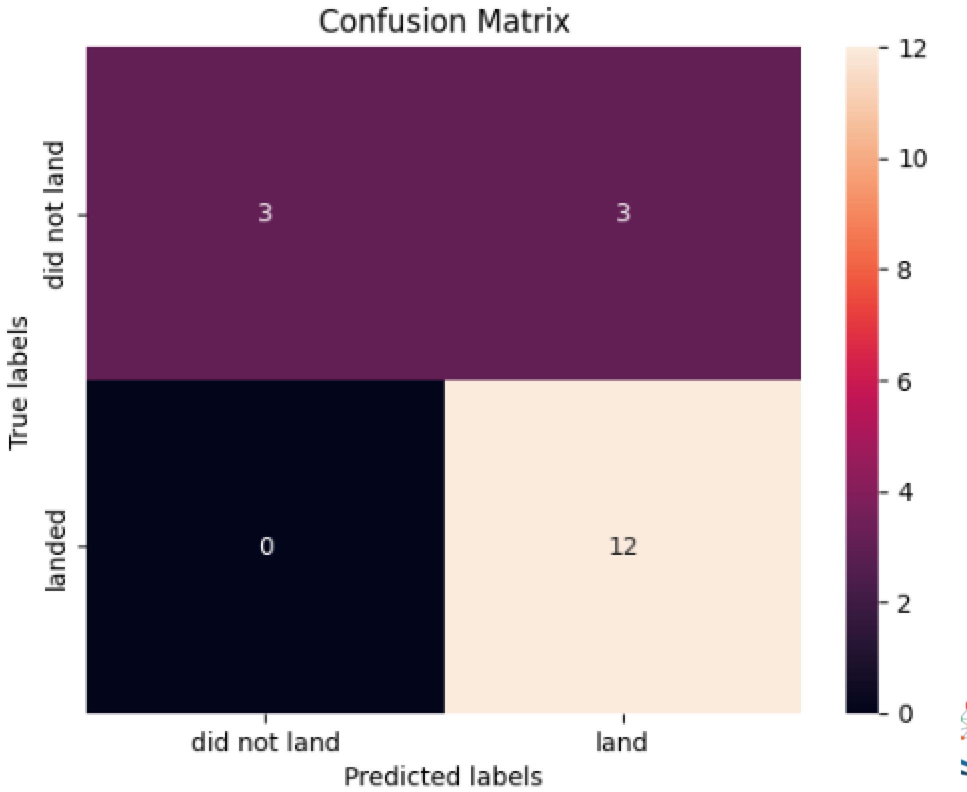


```
accuracy_on_test = logreg_cv.score(X_test, Y_test)
print("Accuracy on Test Data:", accuracy_on_test)
```

```
Accuracy on Test Data: 0.8333333333333334
```

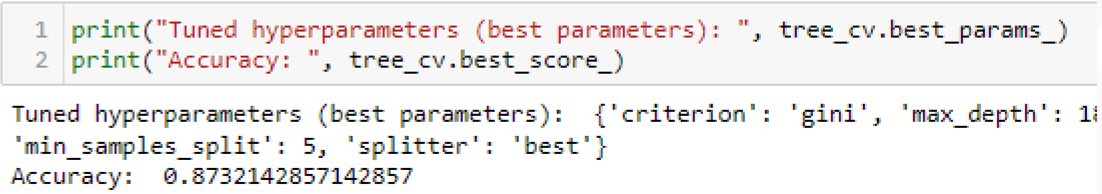
Lets look at the confusion matrix:

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



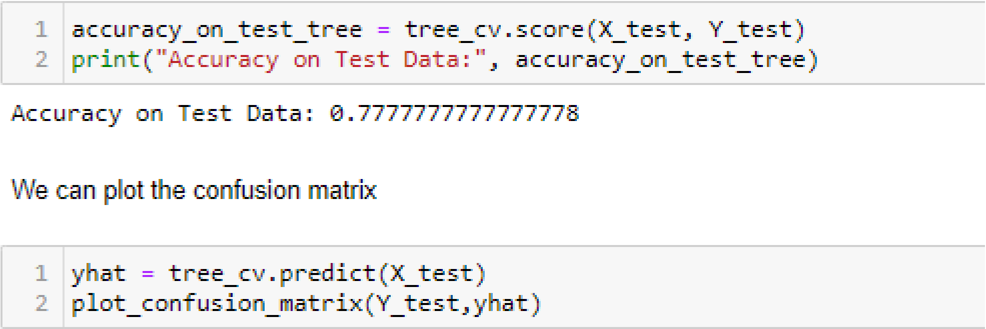
Predictive analysis (classification) results 3

Decision tree model performance



```
print("Tuned hyperparameters (best parameters): ", tree_cv.best_params_)
print("Accuracy: ", tree_cv.best_score_)
```

```
Tuned hyperparameters (best parameters):  {'criterion': 'gini', 'max_depth': 18
'min_samples_split': 5, 'splitter': 'best'}
Accuracy:  0.8732142857142857
```

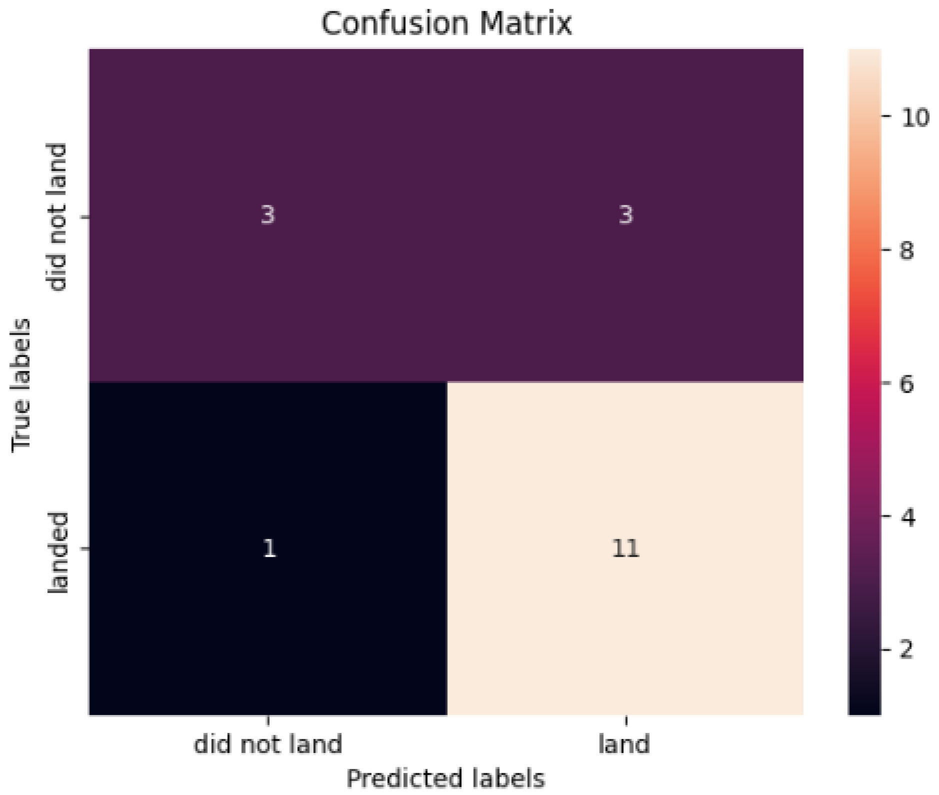


```
accuracy_on_test_tree = tree_cv.score(X_test, Y_test)
print("Accuracy on Test Data:", accuracy_on_test_tree)
```

```
Accuracy on Test Data: 0.7777777777777778
```

We can plot the confusion matrix

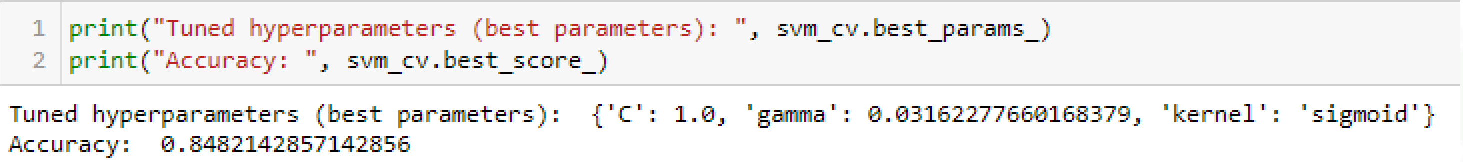
```
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



IBM Developer

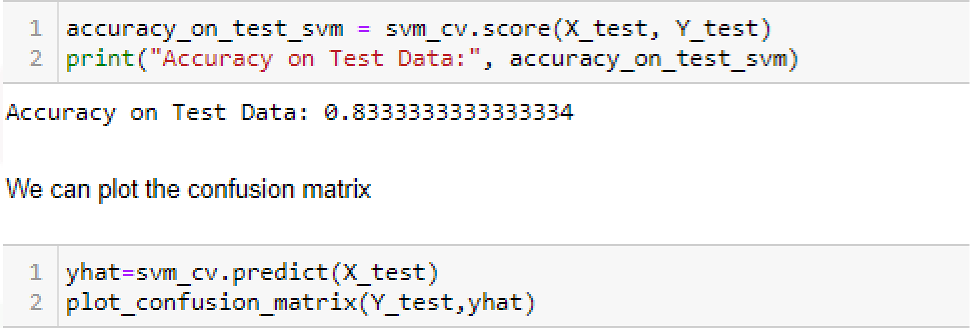
Predictive analysis (classification) results 4

SVM model performance



```
print("Tuned hyperparameters (best parameters): ", svm_cv.best_params_)
print("Accuracy: ", svm_cv.best_score_)
```

```
Tuned hyperparameters (best parameters):  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
Accuracy:  0.8482142857142856
```

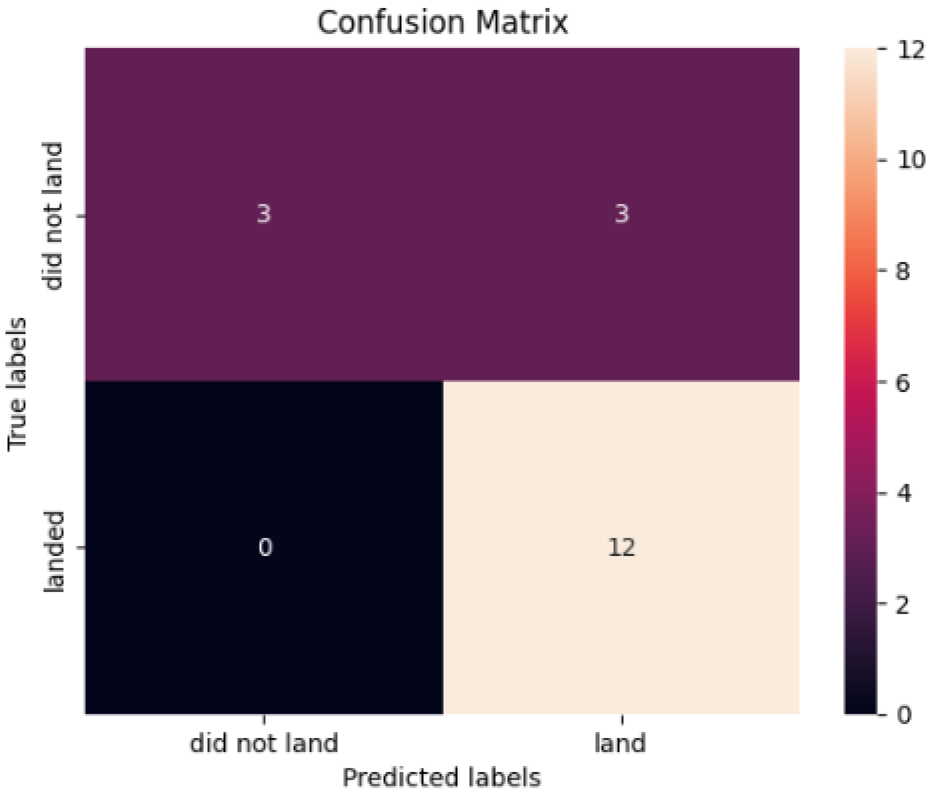


```
accuracy_on_test_svm = svm_cv.score(X_test, Y_test)
print("Accuracy on Test Data:", accuracy_on_test_svm)
```

```
Accuracy on Test Data: 0.8333333333333334
```

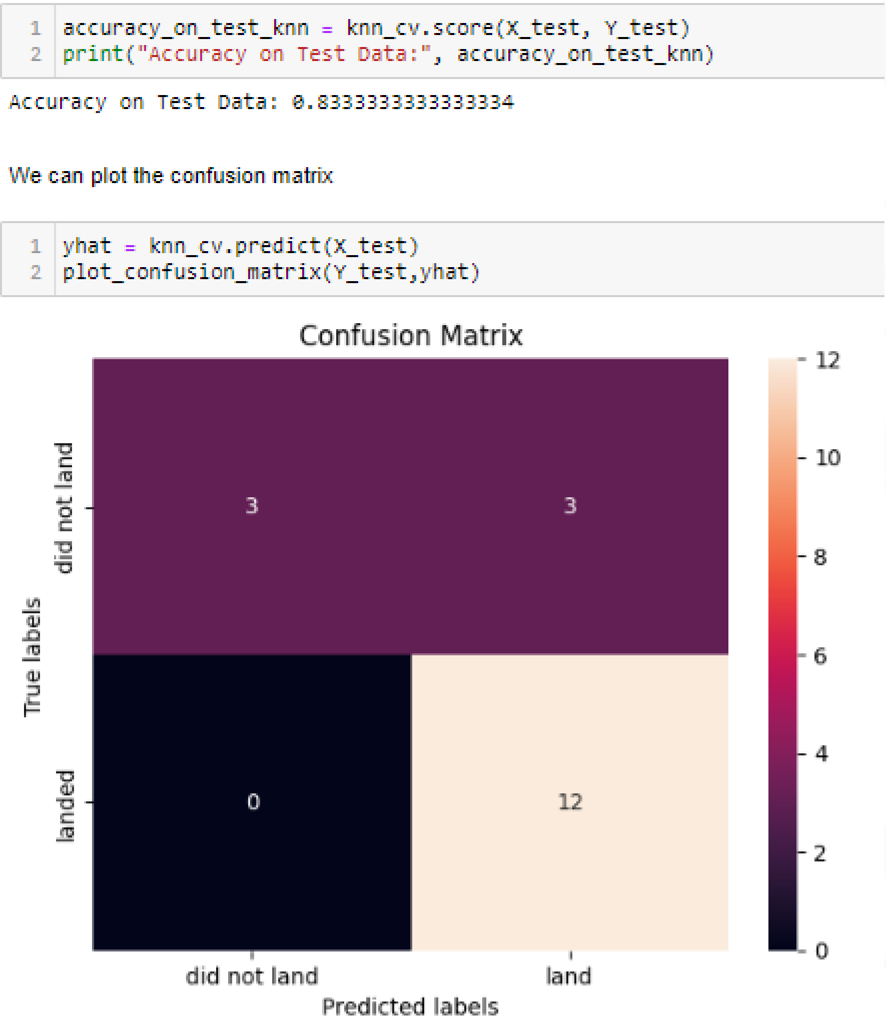
We can plot the confusion matrix

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



IBM Developer

Predictive analysis (classification) results 5

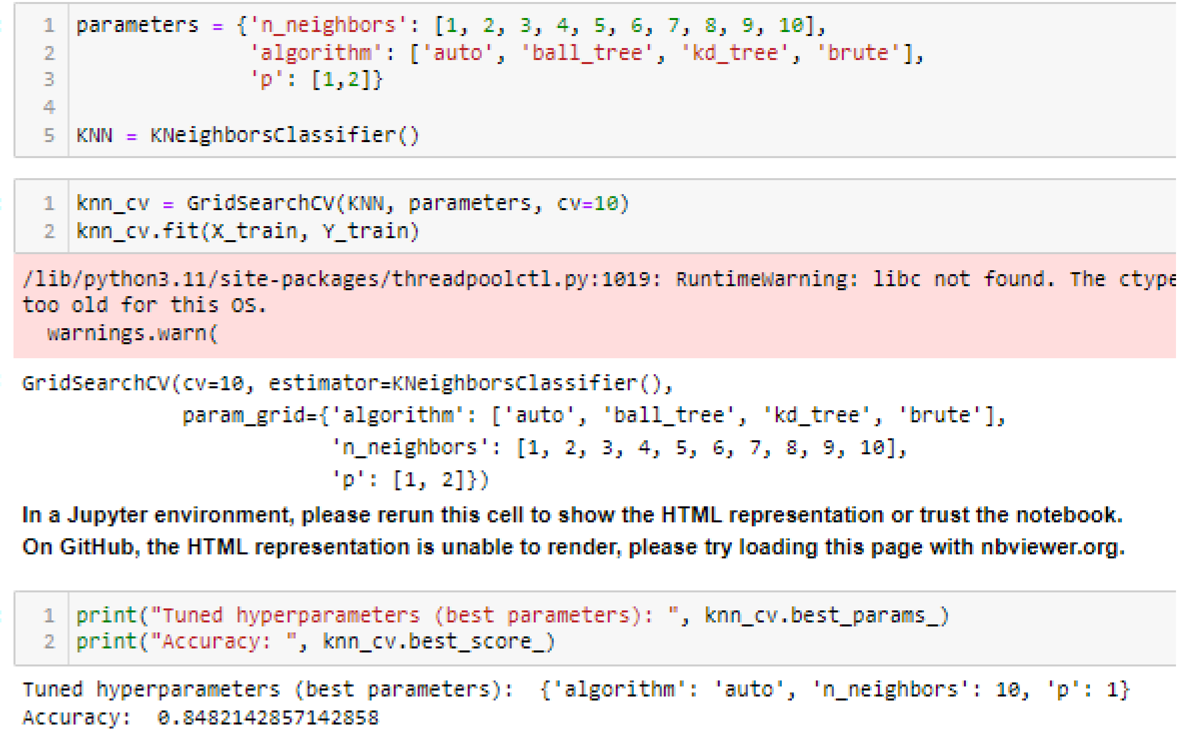


```
accuracy_on_test_knn = knn_cv.score(X_test, Y_test)
print("Accuracy on Test Data:", accuracy_on_test_knn)
```

```
Accuracy on Test Data: 0.8333333333333334
```

We can plot the confusion matrix

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



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

KNN = KNeighborsClassifier()
```

```
knn_cv = GridSearchCV(KNN, parameters, cv=10)
knn_cv.fit(X_train, Y_train)
```

```
/lib/python3.11/site-packages/threadpoolctl.py:1019: RuntimeWarning: libc not found. The ctype
too old for this OS.
  warnings.warn(
```

```
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]})
```

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

```
print("Tuned hyperparameters (best parameters): ", knn_cv.best_params_)
print("Accuracy: ", knn_cv.best_score_)
```

```
Tuned hyperparameters (best parameters):  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
Accuracy:  0.8482142857142858
```

All models tested had three false-negative but a decision tree classifier had one more false-negative than the other models.

K-nearest neighbors(knn) had the highest accuracy score among all models.

DISCUSSION

- Decision-tree classifier
  - The 'max_features' parameter should be set to an integer, a float, a string ('log2' or 'sqrt'), or None. 'auto' that were taught and provided from the instruction is replaced with None.

- Accuracy vs run-time
  - The accuracy of the model was not noticeably different among the models used.
  - Decision tree and kNN took more resource and time for fitting the models to the test data than the logistic regression and SVM.
  - Assuming the quality of dataset stay similar, the trade-off between accuracy and resource consumption should be considered when volume of data is significant.

OVERALL FINDINGS & IMPLICATIONS

Findings

- For the launch site CCAFS SLC 40 only, high success rate for the payload mass over 10,000.
- The heavier Payload mass, the higher success rate for the three orbits: LEO, ISS, and PO.
- KSC LC-39A has the highest success rate.

→

Put more resources to the combination of the launch site, payload and the orbit that has had higher success rate.

CONCLUSION

- Among many factors (features), some were indifferent with respects to the success rate.
- However, some had definitely influenced more on the success rate of the landing scenario.
- So, the company should focus more on the combination of the success factors by putting more resources to them to increase a chance of reusing the rocket and reduce its cost.

APPENDIX

- Rocket launch data from SpaceX API
  - https://api.spacexdata.com/v4/launches/past
- List of Falcon 9 and Falcon Heavy launches Wikipage updated on 9th June 2021
  - https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922