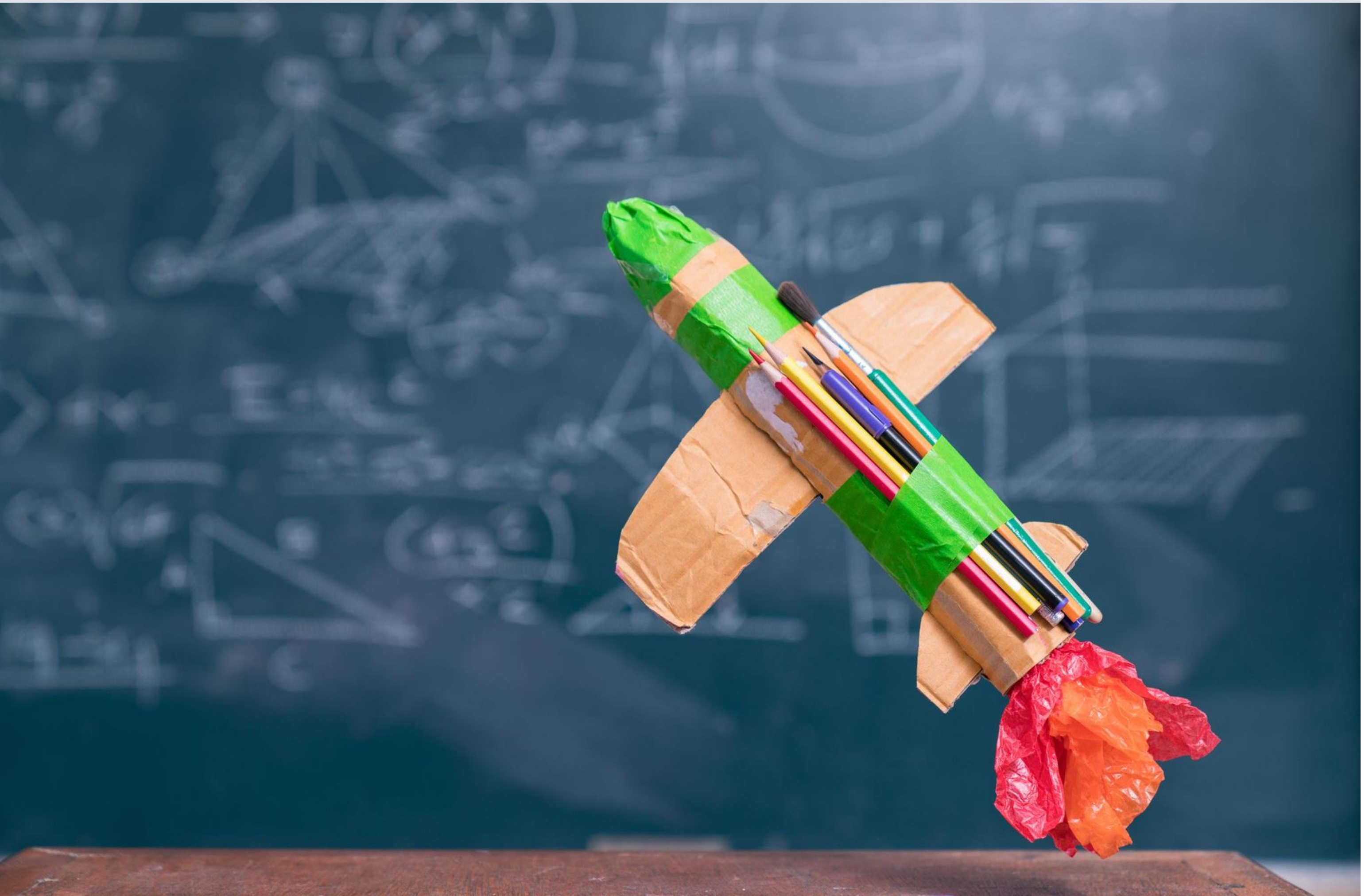
Falcon 9: Data Science Analysis

FRANCESCO COCCIRO

Executive Summary



- This project focuses on predicting the landing success of the first stage of SpaceX's Falcon 9 rocket. By leveraging data analysis and Data Science models, the primary objective is to provide the competing startup with tools and insights to make more competitive bids for rocket launches. The outcomes of this project will enable the startup to make informed decisions based on the predictions of Falcon 9 landing success.
- The methodology involving data collection, data wrangling, exploratory data analysis, data visualization, model development, model evaluation, and reporting the results.

Table of content

- DATA COLLECTING
- EXPLORATORY DATA ANALYSIS
- DATA VISUALIZATION
- PREDICTIVE ANALYSIS

Welcome to the world of commercial space travel.

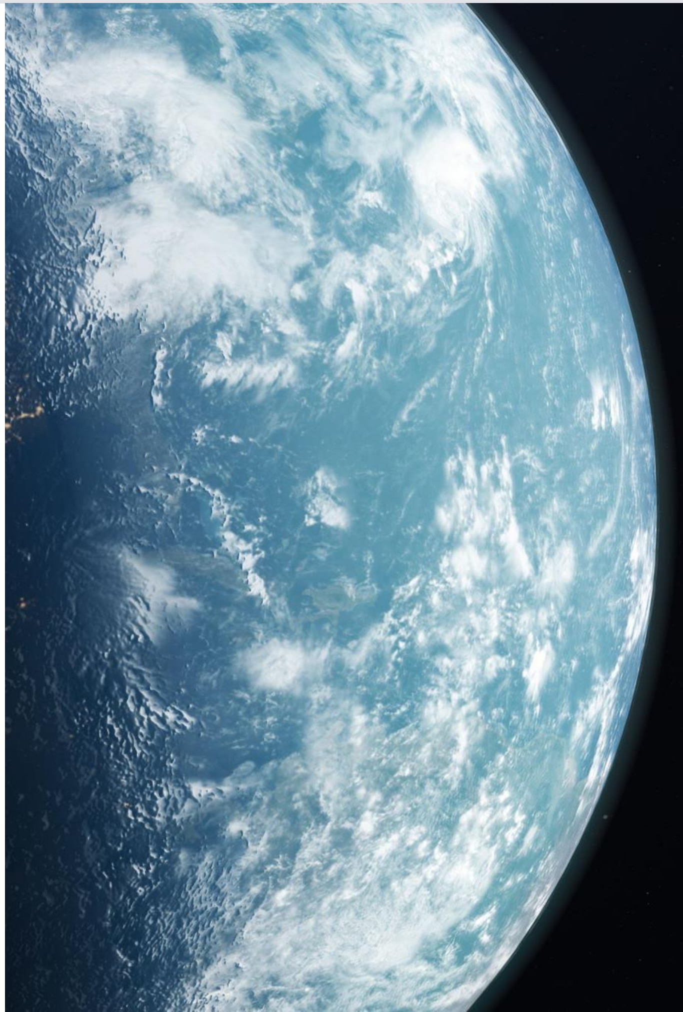
In an era where companies are revolutionizing space exploration and making it more accessible, SpaceX has emerged as a frontrunner in the industry. With remarkable achievements like sending spacecraft to the International Space Station, deploying the Starlink satellite internet constellation, and conducting manned missions to space, SpaceX has proven its capabilities.

One key factor behind SpaceX's success is the cost-effectiveness of its Falcon 9 rocket launches. While other providers charge significantly higher prices, SpaceX stands out by reusing the first stage of its rockets, resulting in substantial savings.

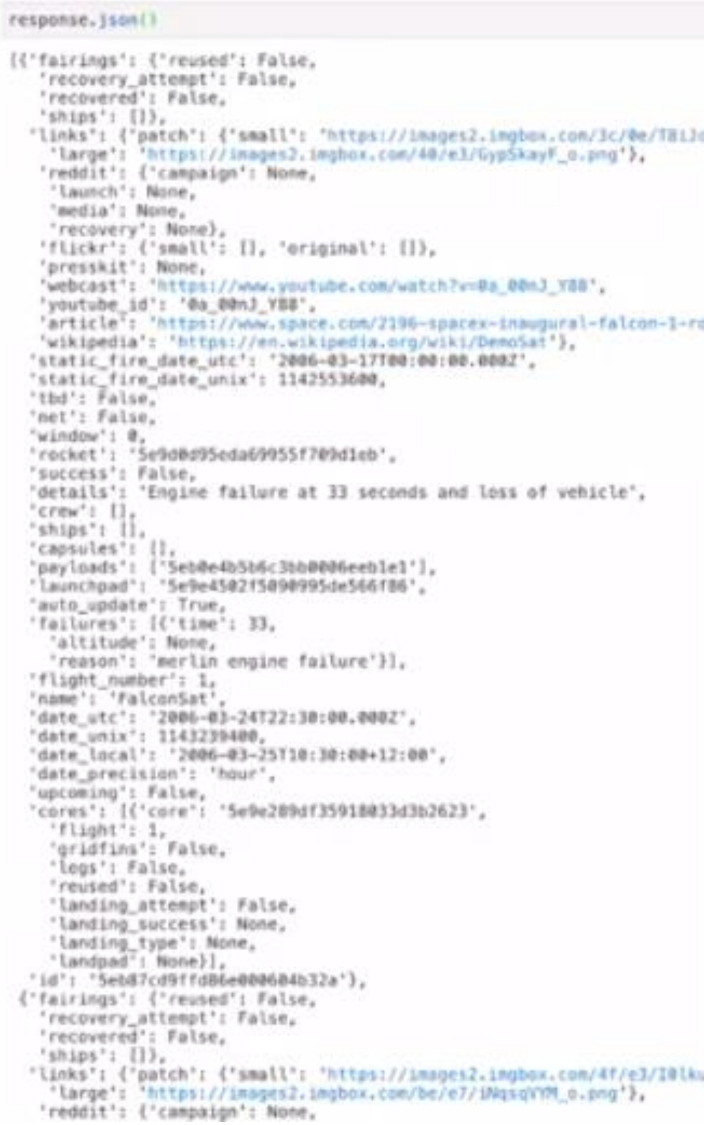
As data scientists working for the aspiring rocket company Space Y, our mission is to compete with SpaceX and establish ourselves in the industry.

Our task involves gathering information about SpaceX, creating insightful dashboards, and training machine learning models using public data.

Through this capstone project, we aim to forecast whether SpaceX will successfully reuse the first stage, ultimately determining the cost-effectiveness of our launches.



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

response =requests.get(url)

response.json()
```

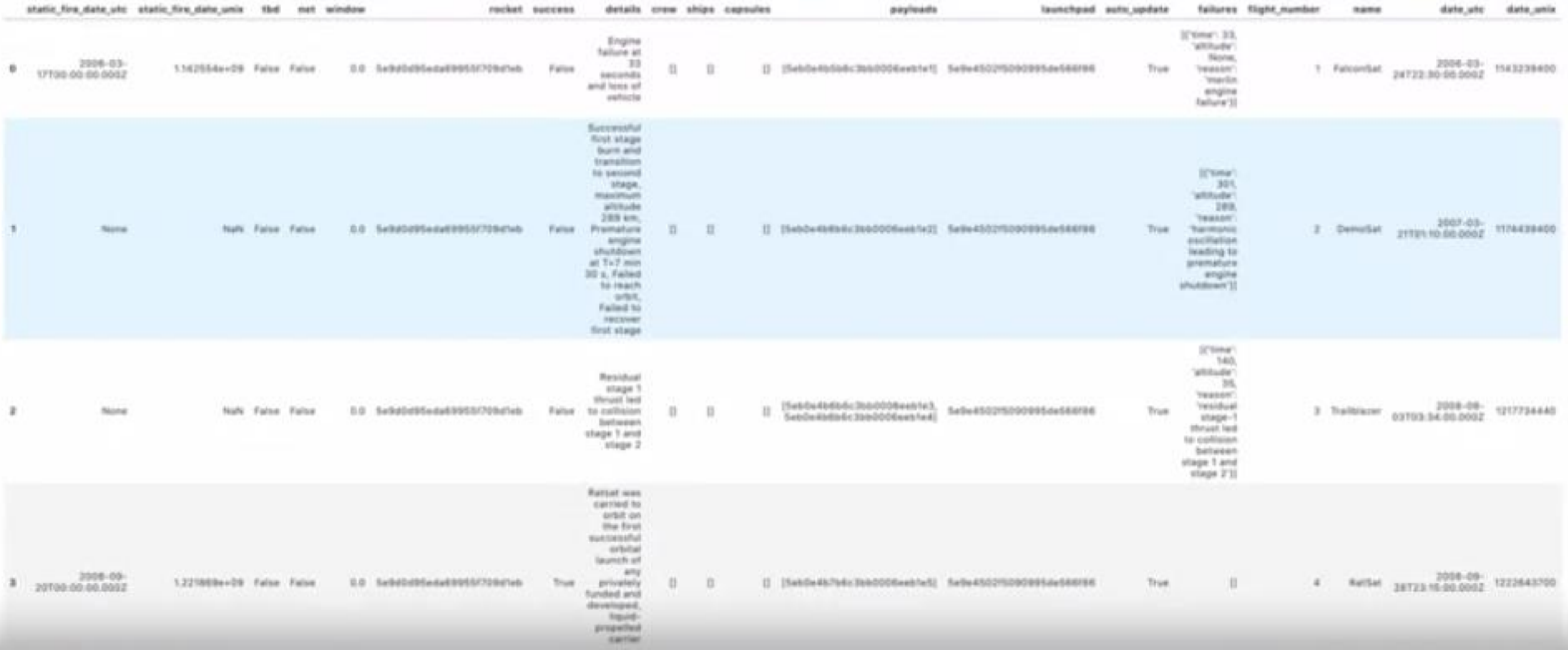


```python
response.json()
```

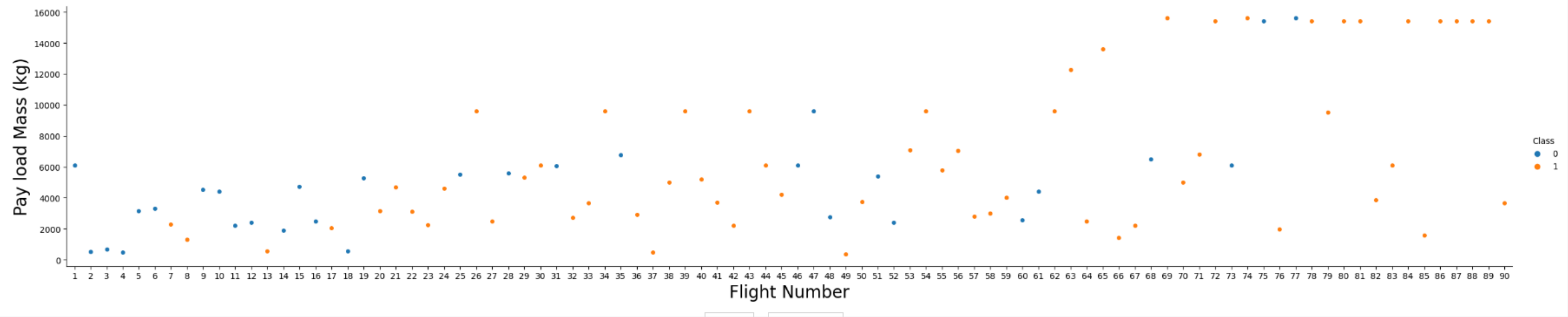
```
[{'fairings': {'reused': False,
   'recovery_attempt': False,
   'recovered': False,
   'ships': []},
  'links': {'patch': {'small': 'https://images2.imgbox.com/3c/0e/T8iJc
    'large': 'https://images2.imgbox.com/40/e3/GypSkayF_o.png'},
   'reddit': {'campaign': None,
    'launch': None,
    'media': None,
    'recovery': None},
   'flickr': {'small': [], 'original': []},
   'presskit': None,
   'webcast': 'https://www.youtube.com/watch?v=0a_00nJ_Y88',
   'youtube_id': '0a_00nJ_Y88',
   'article': 'https://www.space.com/2196-spacex-inaugural-falcon-1-ro
   'wikipedia': 'https://en.wikipedia.org/wiki/DemoSat'},
  'static_fire_date_utc': '2006-03-17T00:00:00.000Z',
  'static_fire_date_unix': 1142553600,
  'tbd': False,
  'net': False,
  'window': 0,
  'rocket': '5e9d0d95eda69955f709d1eb',
  'success': False,
  'details': 'Engine failure at 33 seconds and loss of vehicle',
  'crew': [],
  'ships': [],
  'capsules': [],
  'payloads': ['5eb0e4b5b6c3bb0006eeb1e1'],
  'launchpad': '5e9e4502f5090995de566f86',
  'auto_update': True,
  'failures': [{'time': 33,
    'altitude': None,
    'reason': 'merlin engine failure'}],
  'flight_number': 1,
  'name': 'FalconSat',
  'date_utc': '2006-03-24T22:30:00.000Z',
  'date_unix': 1143239400,
  'date_local': '2006-03-25T10:30:00+12:00',
  'date_precision': 'hour',
  'upcoming': False,
  'cores': [{'core': '5e9e289df35918033d3b2623',
    'flight': 1,
    'gridfins': False,
    'legs': False,
    'reused': False,
    'landing_attempt': False,
    'landing_success': None,
    'landing_type': None,
    'landpad': None}],
  'id': '5eb87cd9ffd86e000604b32a'},
 {'fairings': {'reused': False,
   'recovery_attempt': False,
   'recovered': False,
   'ships': []},
  'links': {'patch': {'small': 'https://images2.imgbox.com/4f/e3/I0lku
    'large': 'https://images2.imgbox.com/be/e7/iNqsqVYM_o.png'},
   'reddit': {'campaign': None,
```

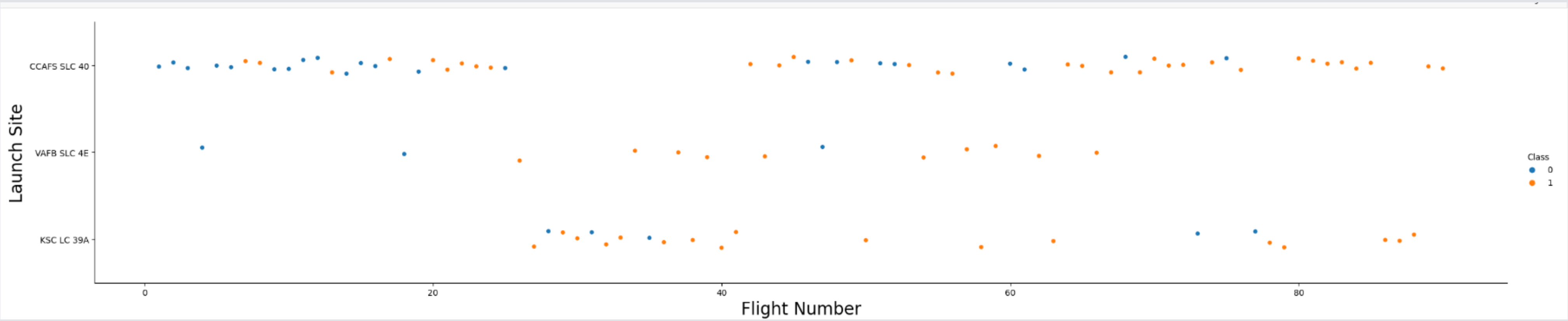
Wrangling Data using an API

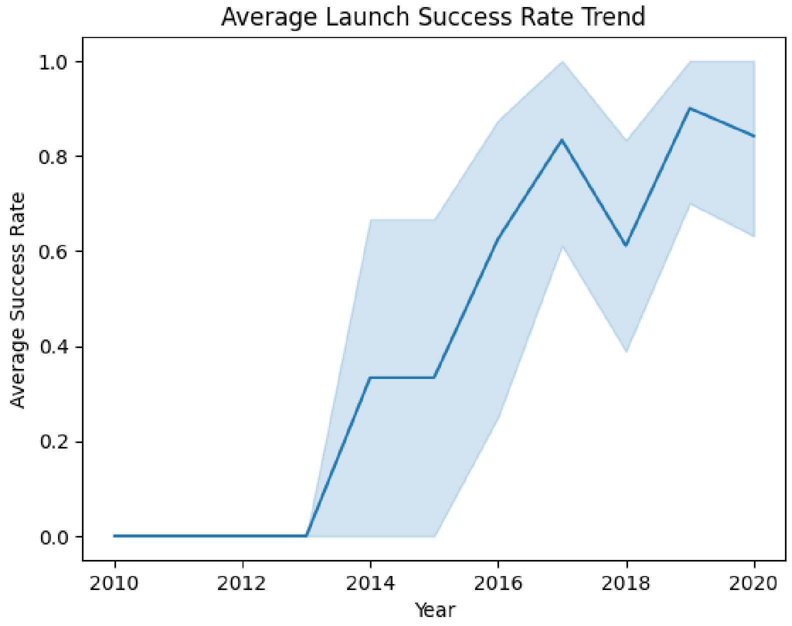
```
data = pd.json_normalize(response.json())
```

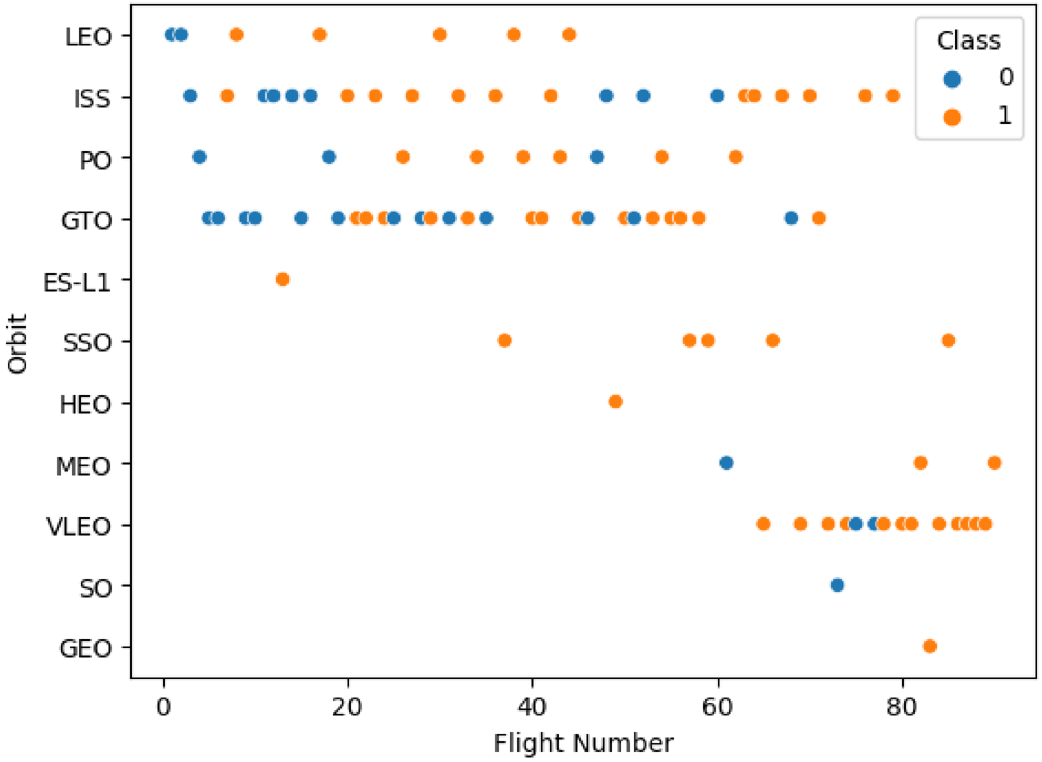


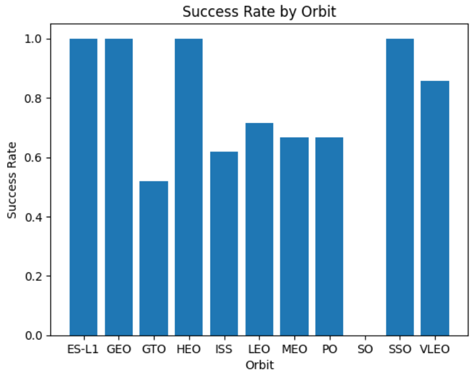
	static_fire_date_utc	static_fire_date_unix	tbd	net	window	rocket	success	details	crew	ships	capsules	payloads	launchpad	auto_update	failures	flight_number	name	date_utc	date_unix
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	False	0.0	5e9d0d95eda69955f709d1eb	False	Engine failure at 33 seconds and loss of vehicle	[]	[]	[]	[5eb0e4b5b6c3bb0006eeb1e1]	5e9e4502f5090995de566f86	True	[{'time': 33, 'altitude': None, 'reason': 'merlin engine failure'}]	1	FalconSat	2006-03-24T22:30:00.000Z	1143239400
1	None	NaN	False	False	0.0	5e9d0d95eda69955f709d1eb	False	Successful first stage burn and transition to second stage, maximum altitude 289 km, Premature engine shutdown at T+7 min 30 s, Failed to reach orbit, Failed to recover first stage	[]	[]	[]	[5eb0e4b6b6c3bb0006eeb1e2]	5e9e4502f5090995de566f86	True	[{'time': 301, 'altitude': 289, 'reason': 'harmonic oscillation leading to premature engine shutdown'}]	2	DemoSat	2007-03-21T01:10:00.000Z	1174439400
2	None	NaN	False	False	0.0	5e9d0d95eda69955f709d1eb	False	Residual stage 1 thrust led to collision between stage 1 and stage 2	[]	[]	[]	[5eb0e4b6b6c3bb0006eeb1e3, 5eb0e4b6b6c3bb0006eeb1e4]	5e9e4502f5090995de566f86	True	[{'time': 140, 'altitude': 35, 'reason': 'residual stage-1 thrust led to collision between stage 1 and stage 2'}]	3	Trailblazer	2008-08-03T03:34:00.000Z	1217734440
3	2008-09-20T00:00:00.000Z	1.221869e+09	False	False	0.0	5e9d0d95eda69955f709d1eb	True	Ratsat was carried to orbit on the first successful orbital launch of any privately funded and developed, liquid-propelled carrier	[]	[]	[]	[5eb0e4b7b6c3bb0006eeb1e5]	5e9e4502f5090995de566f86	True	[]	4	RatSat	2008-09-28T23:15:00.000Z	1222643700

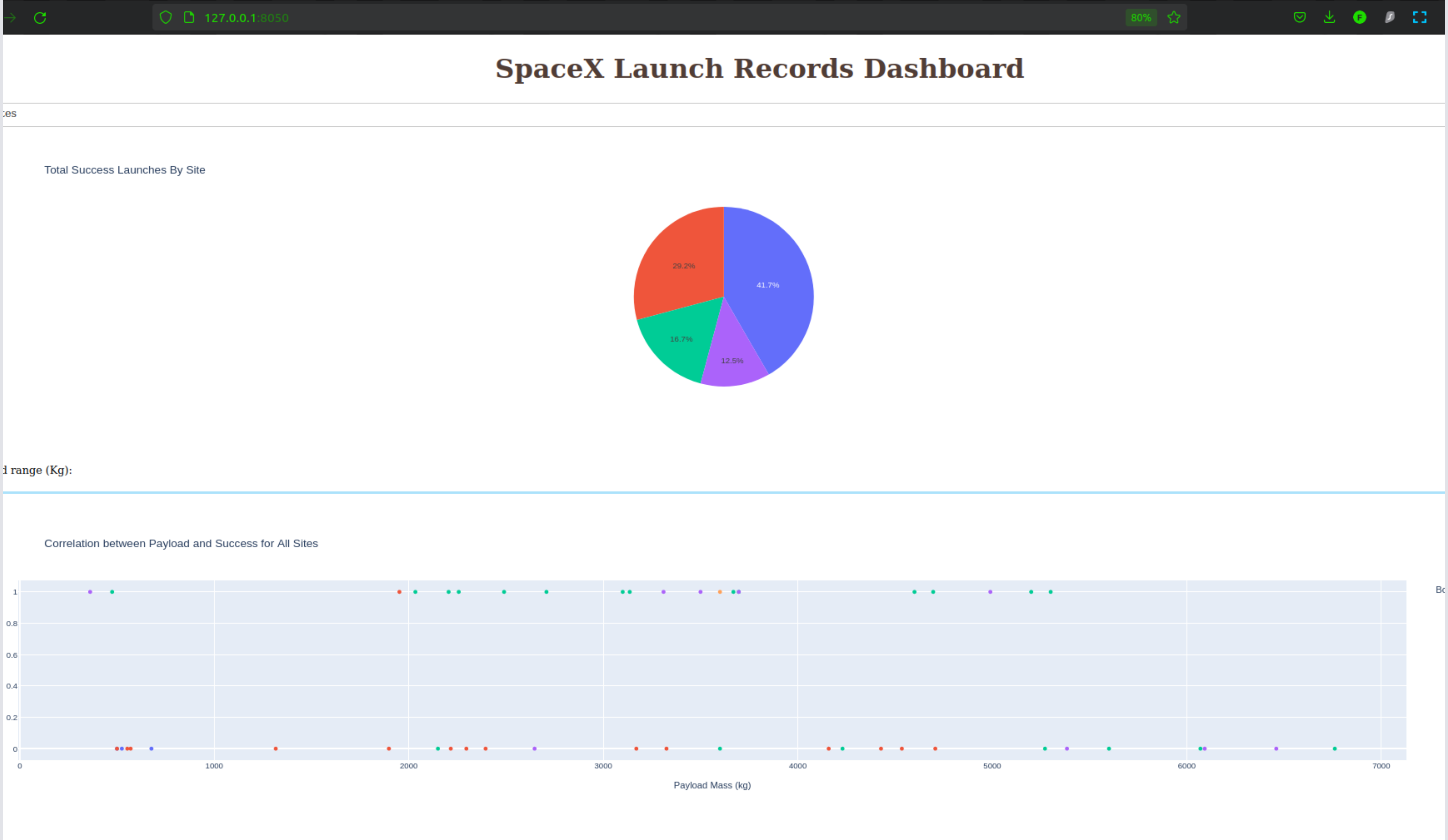

Pay load Mass (kg)
0
2000
4000
6000
8000
10000
12000
14000
16000
Flight Number
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90
Class
● 0
● 1


Launch Site
CCAFS SLC 40
VAFB SLC 4E
KSC LC 39A
0
20
40
60
80
Flight Number
Class
● 0
● 1


Average Launch Success Rate Trend
Average Success Rate
0.0
0.2
0.4
0.6
0.8
1.0
2010
2012
2014
2016
2018
2020
Year


Orbit
LEO
ISS
PO
GTO
ES-L1
SSO
HEO
MEO
VLEO
SO
GEO
0
20
40
60
80
Flight Number
Class
0
1


Success Rate by Orbit
Success Rate
0.0
0.2
0.4
0.6
0.8
1.0
ES-L1
GEO
GTO
HEO
ISS
LEO
MEO
PO
SO
SSO
VLEO
Orbit



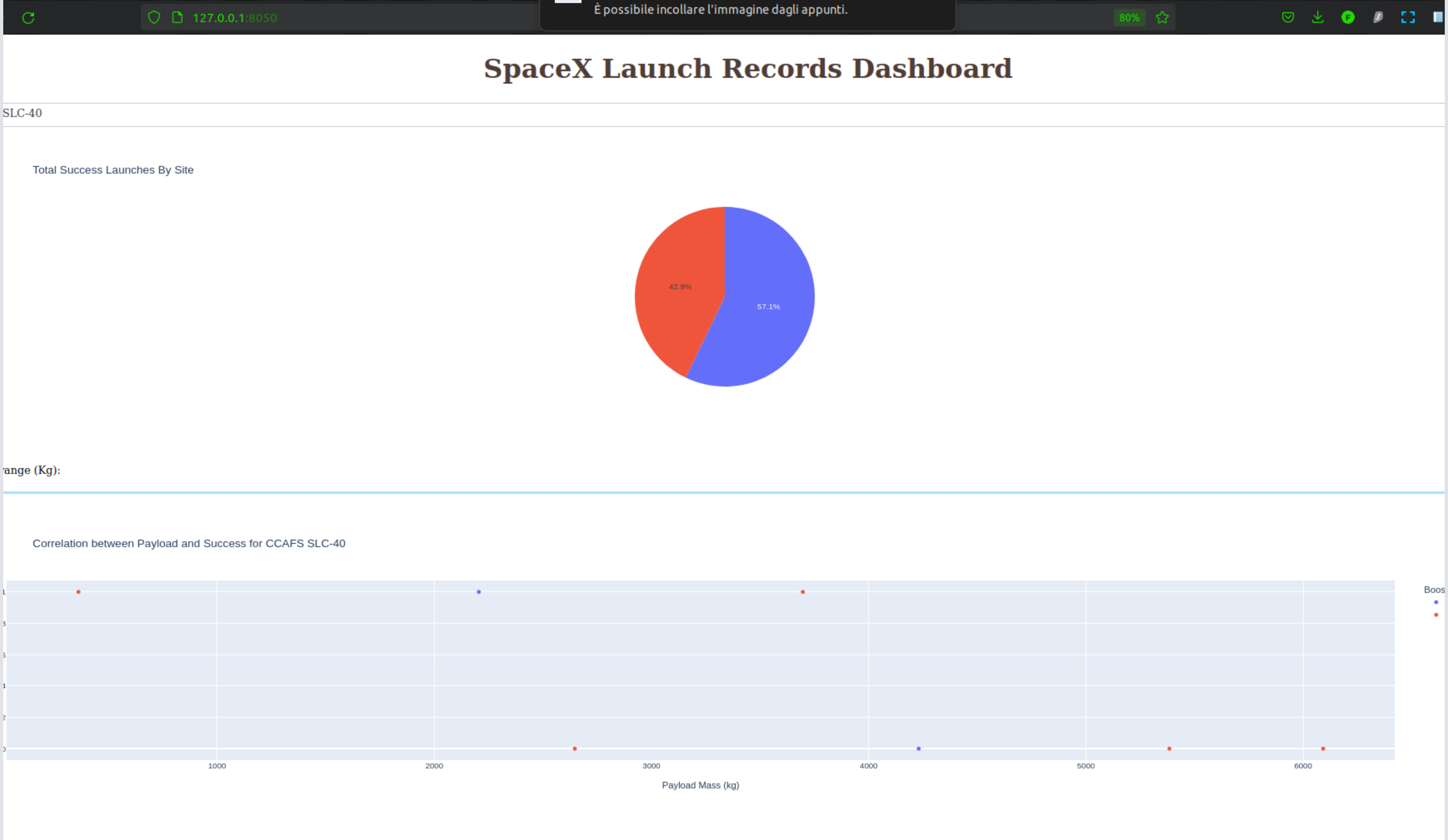
SpaceX Launch Records Dashboard

ces

Total Success Launches By Site

d range (Kg):

Correlation between Payload and Success for All Sites



È possibile incollare l'immagine dagli appunti.

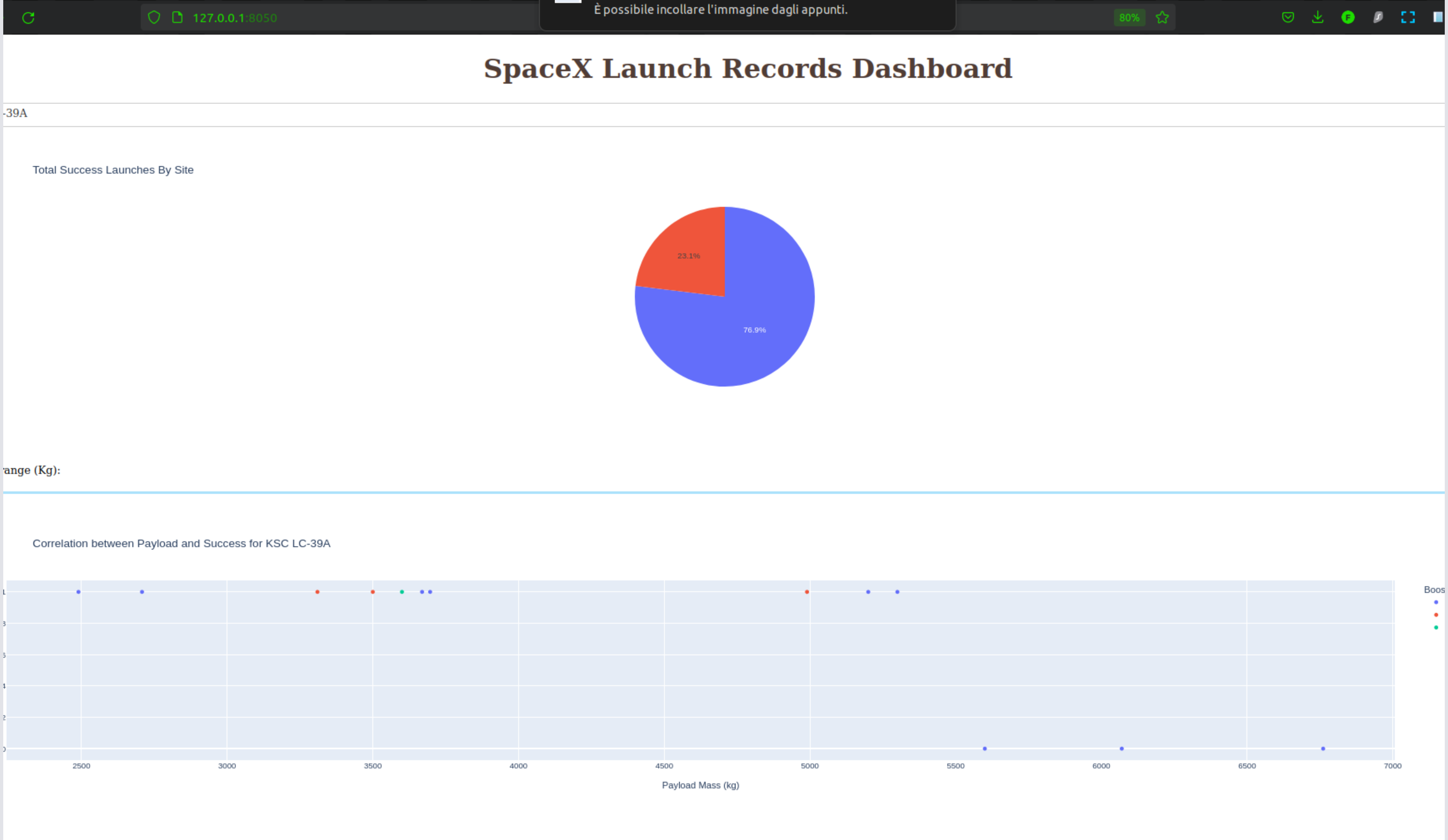
SpaceX Launch Records Dashboard

SLC-40

Total Success Launches By Site

ange (Kg):

Correlation between Payload and Success for CCAFS SLC-40



È possibile incollare l'immagine dagli appunti.

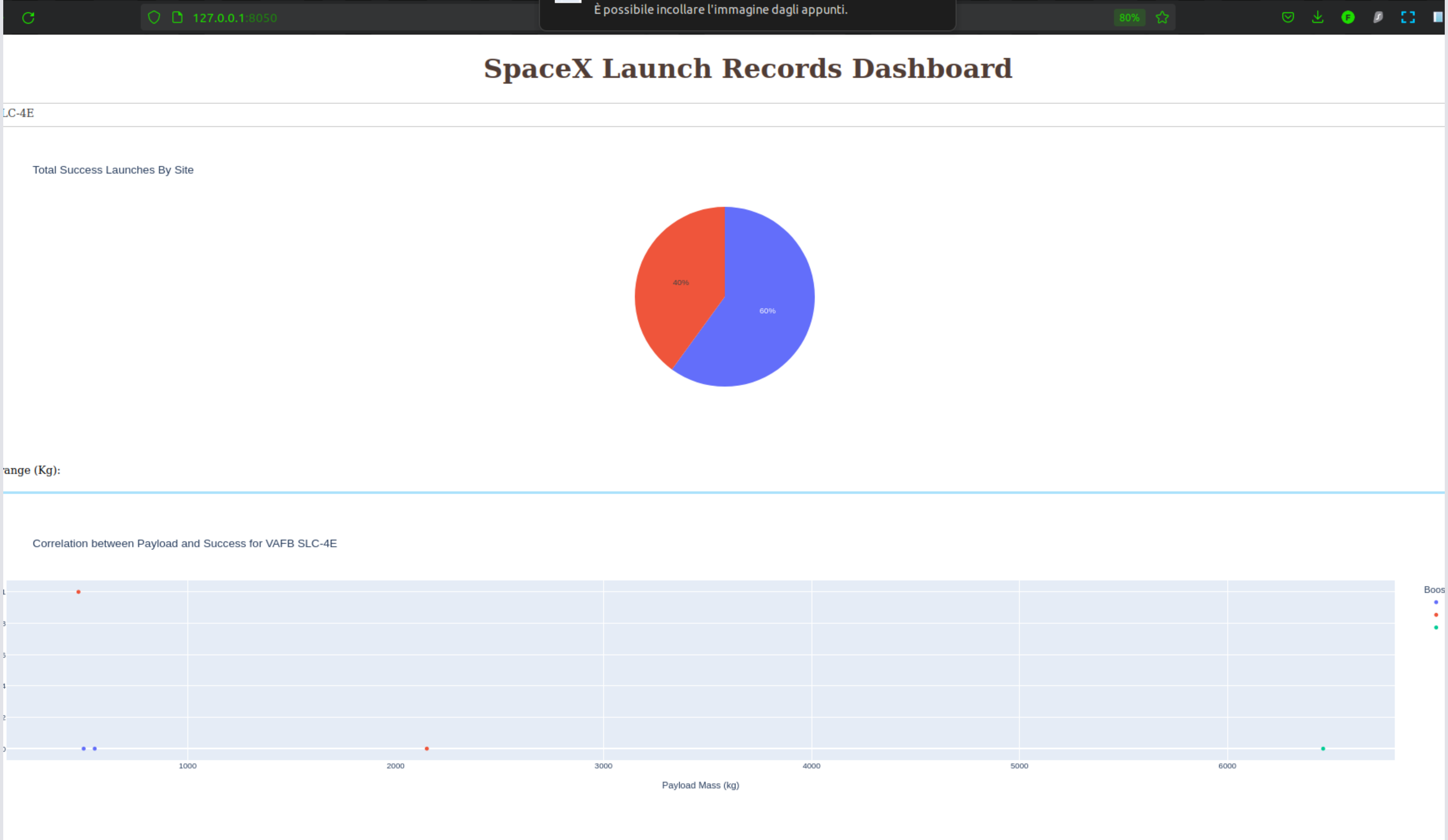
SpaceX Launch Records Dashboard

-39A

Total Success Launches By Site

ange (Kg):

Correlation between Payload and Success for KSC LC-39A



127.0.0.1:8050

È possibile incollare l'immagine dagli appunti.

80%

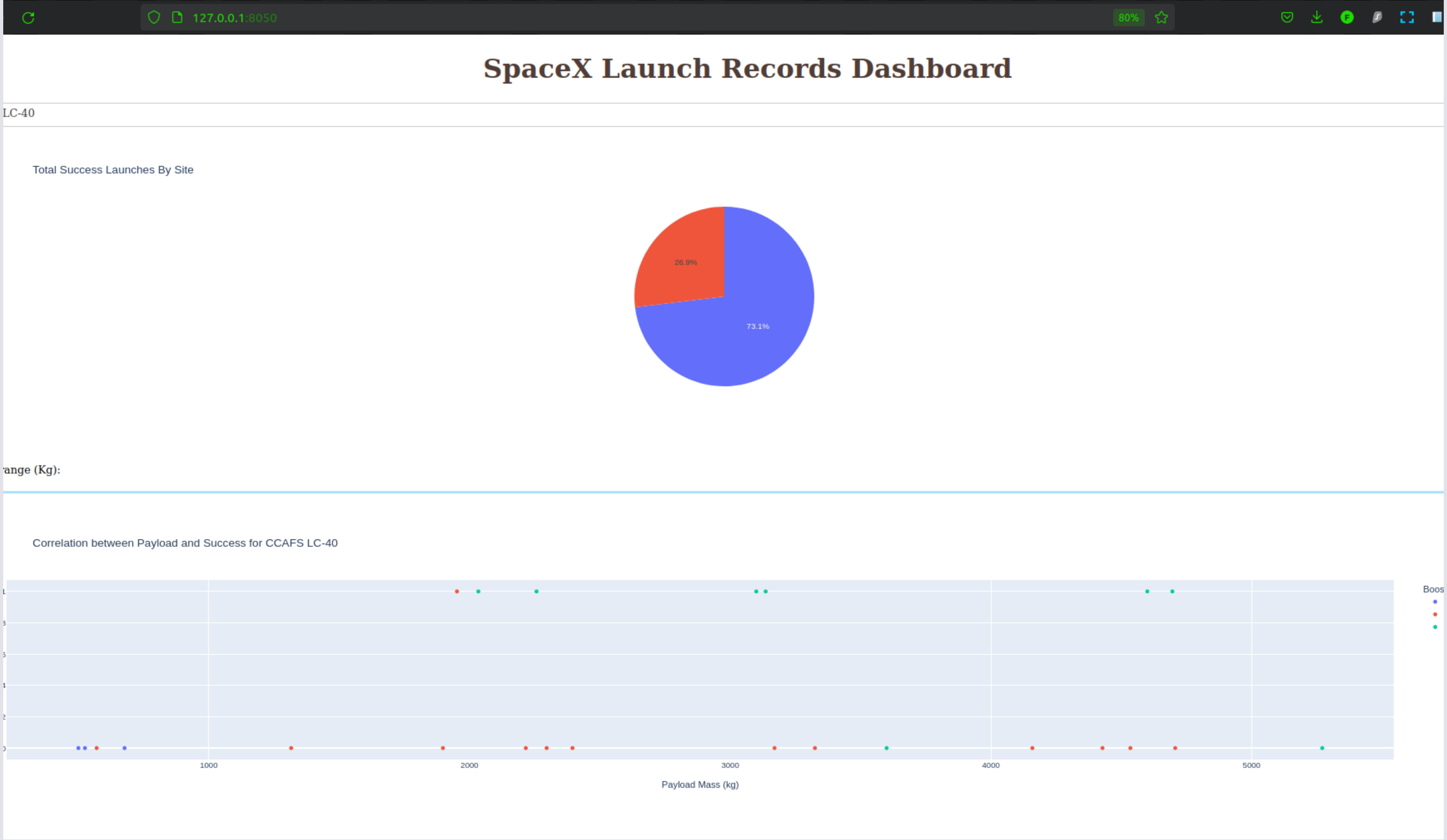
SpaceX Launch Records Dashboard

LC-4E

Total Success Launches By Site

ange (Kg):

Correlation between Payload and Success for VAFB SLC-4E



127.0.0.1:8050

80%

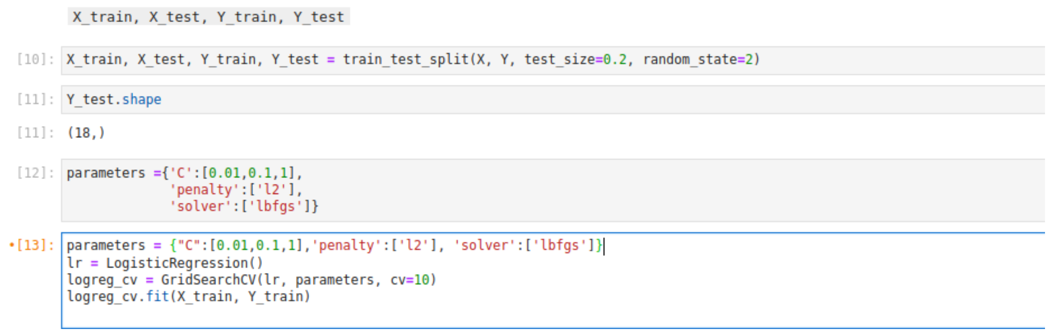
SpaceX Launch Records Dashboard

LC-40

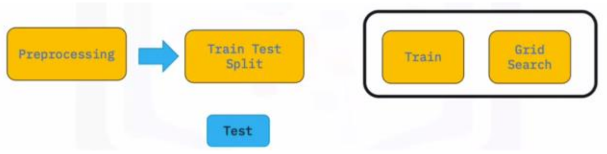
Total Success Launches By Site

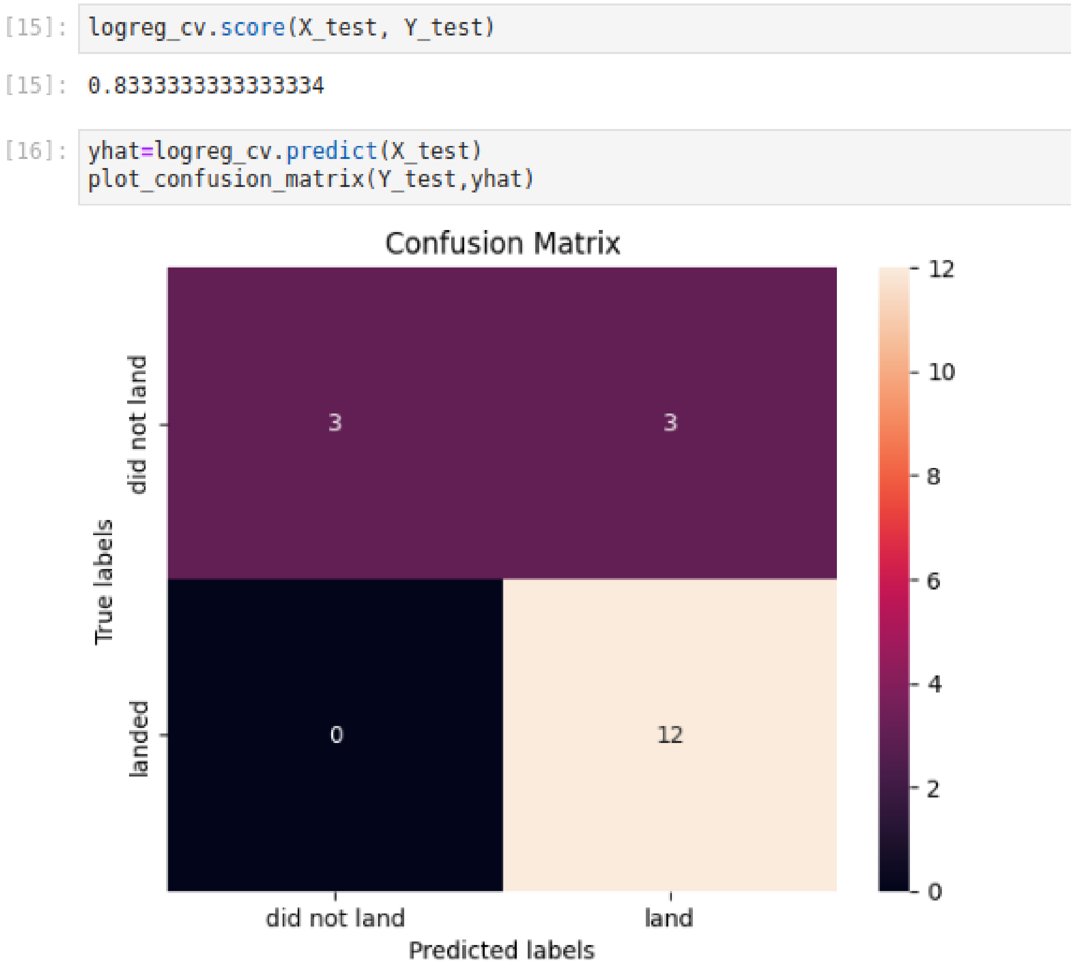
ange (Kg):

Correlation between Payload and Success for CCAFS LC-40

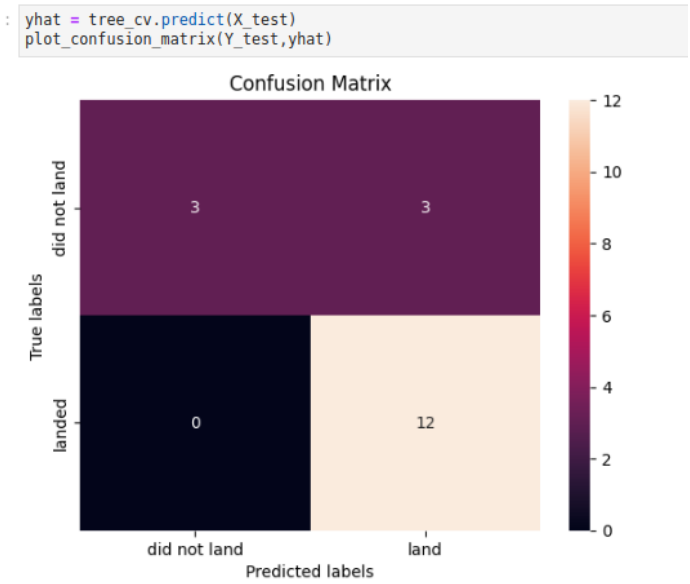


```
X_train, X_test, Y_train, Y_test

[10]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)

[11]: Y_test.shape

[11]: (18,)

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

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

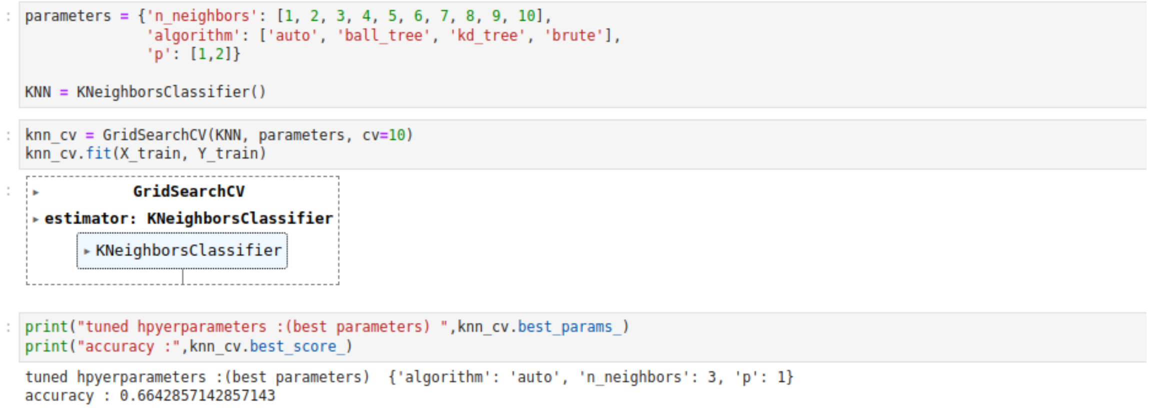




```
[15]: logreg_cv.score(X_test, Y_test)

[15]: 0.8333333333333334

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



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



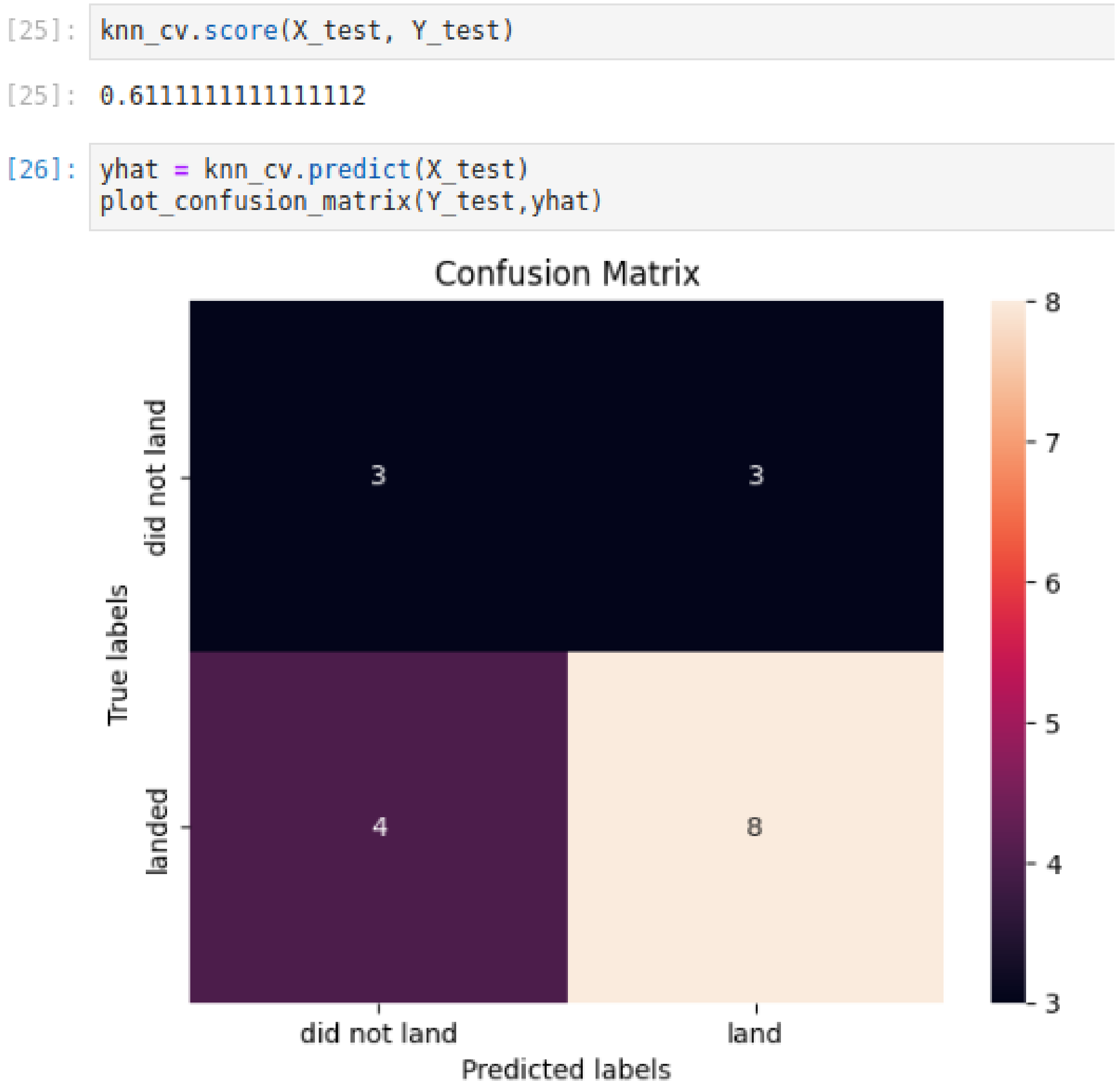
```
parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
              'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
              'p': [1,2]}

KNN = KNeighborsClassifier()
```

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

```
GridSearchCV
estimator: KNeighborsClassifier
KNeighborsClassifier
```

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

```
tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 3, 'p': 1}
accuracy : 0.6642857142857143
```



```
[25]: knn_cv.score(X_test, Y_test)
```

```
[25]: 0.6111111111111112
```

```
[26]: yhat = knn_cv.predict(X_test)
      plot_confusion_matrix(Y_test,yhat)
```

Thanks