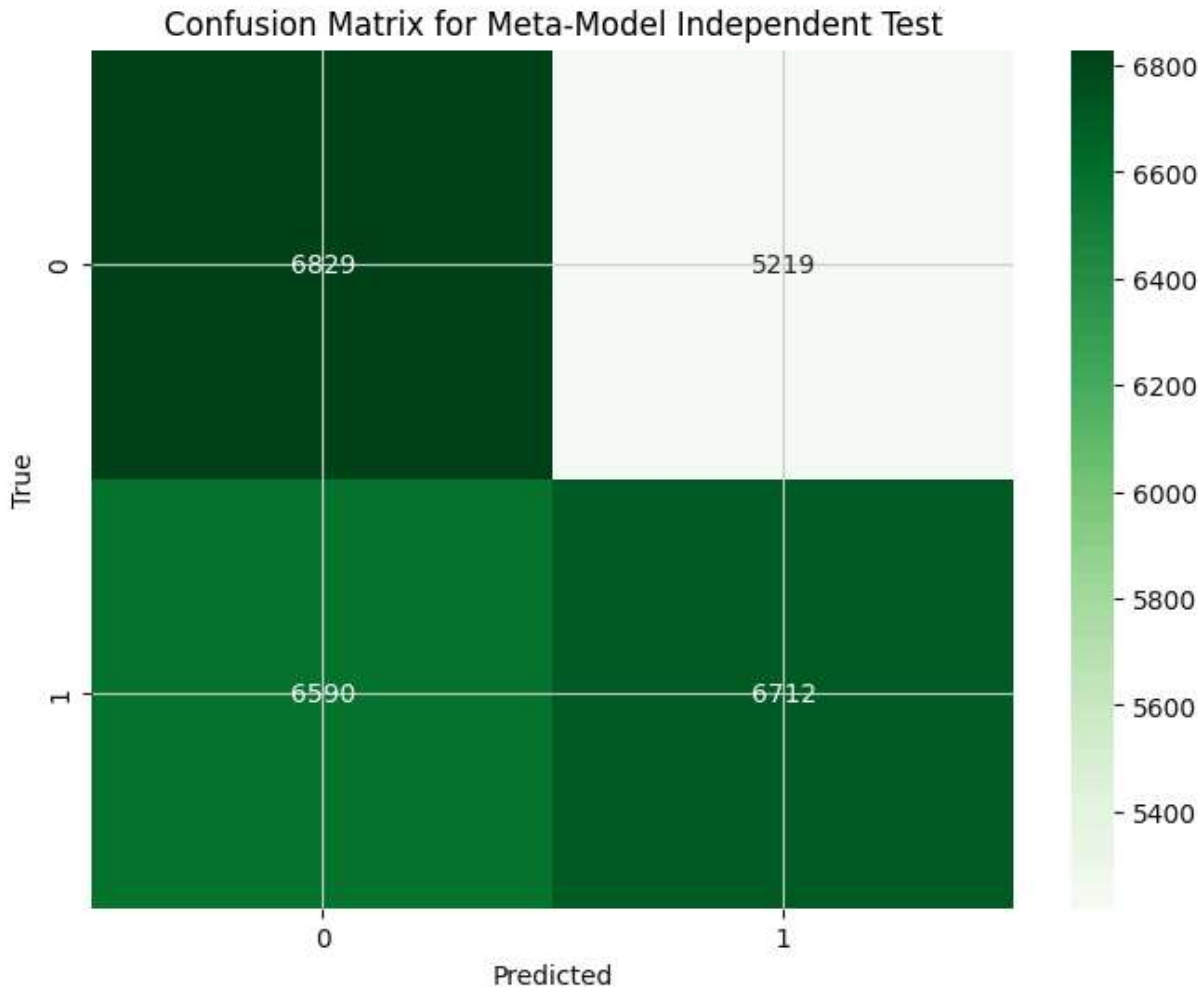
```
feature_subsets, scaler =\
_Data_Processing.preprocess_data(
        file_name   =           'spx_full.csv'
        ,indp_size  =           0.01
        ,test_size  =           0.01
        ,shfl_splt      =               False
        ,t_start    =           645
        ,t_end          =               800
        ,mod_type       =               'Area_Classification'
        ,target_t       =               45
        ,num_class      =               2
        ,split_val      =               5
        ,verbose        =               0
        ,scaler         =               'Custom'
        ,cstm_scale     =               joblib.load('scaler/tmp.joblib')
        ,frmt_lstm      =               lstm_format
        ,time_steps =           5
        ,keep_price =           False
        ,indices        =               0
)

from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
import _Master_Model
from importlib import reload
reload(_Master_Model)

test_on_X = X_load
test_on_y = y_load

loadmodel = _Master_Model.Master(
        model_depth=2
)
loadmodel.load_model('pred1_63p2_acc-645-800')
m_pred = loadmodel.master_predict(test_on_X, threshold=0.5)
print(accuracy_score(test_on_y, m_pred))

#Create the confusion matrix
cm = confusion_matrix(test_on_y, m_pred)
# Plot the confusion matrix using seaborn
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Greens', \
                        xticklabels=range(2), yticklabels=range(2))
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title(f'Confusion Matrix for Meta-Model Independent Test')
plt.show()
```

```
loaded chunk 1 of size: 125400164 -> 64675164
loaded chunk 2 of size: 125400164 -> 64575164
loaded chunk 3 of size: 125400164 -> 64675164
loaded chunk 4 of size: 125400164 -> 64675164
loaded chunk 5 of size: 125400164 -> 64675164
loaded chunk 6 of size: 125400164 -> 64675164
loaded chunk 7 of size: 125400164 -> 64675164
loaded chunk 8 of size: 125400164 -> 64575164
loaded chunk 9 of size: 125400164 -> 64675164
loaded chunk 10 of size: 7905380 -> 3951196
concat chunks
concatted chunks
Success.
Size of dataset:        586152276
<class 'numpy.ndarray'> <class 'numpy.ndarray'> <class 'numpy.float64'>
        201226 Samples Dropped.

793/793 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step
793/793 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step
793/793 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
793/793 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
793/793 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
793/793 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
793/793 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
793/793 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step
(25350, 8)
793/793 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step
0.5341617357001972
```



In [27]:
```
import matplotlib.pyplot as plt

import _Data_Processing
from importlib import reload
import joblib
reload(_Data_Processing)
lstm_format = False
X_loadraw, _, _, __,\
y_loadraw, _, ___, ____,\
feature_subsets, scaler =\
_Data_Processing.preprocess_data(
        file_name   =           'spx_full.csv'
        ,indp_size  =           0.01
        ,test_size  =           0.01
        ,shfl_splt      =               False
        ,t_start    =           645
        ,t_end          =               800
        ,mod_type       =               'Area_Classification'
        ,target_t       =               45
        ,num_class      =               2
        ,split_val      =               5
        ,verbose        =               0
        ,scaler         =               'Standard'
        ,cstm_scale     =               joblib.load('scaler/tmp.joblib')
        ,frmt_lstm      =               lstm_format
```

```
        ,time_steps =           5
        ,keep_price =           True
        ,indices        =               0
)
```

```
loaded chunk 1 of size: 125400164 -> 64675164
loaded chunk 2 of size: 125400164 -> 64575164
loaded chunk 3 of size: 125400164 -> 64675164
loaded chunk 4 of size: 125400164 -> 64675164
loaded chunk 5 of size: 125400164 -> 64675164
loaded chunk 6 of size: 125400164 -> 64675164
loaded chunk 7 of size: 125400164 -> 64675164
loaded chunk 8 of size: 125400164 -> 64575164
loaded chunk 9 of size: 125400164 -> 64675164
loaded chunk 10 of size: 7905380 -> 3951196
concat chunks
concatted chunks
Success.
Size of dataset:        586152276
<class 'numpy.ndarray'> <class 'numpy.ndarray'> <class 'numpy.float64'>
        201226 Samples Dropped.
```

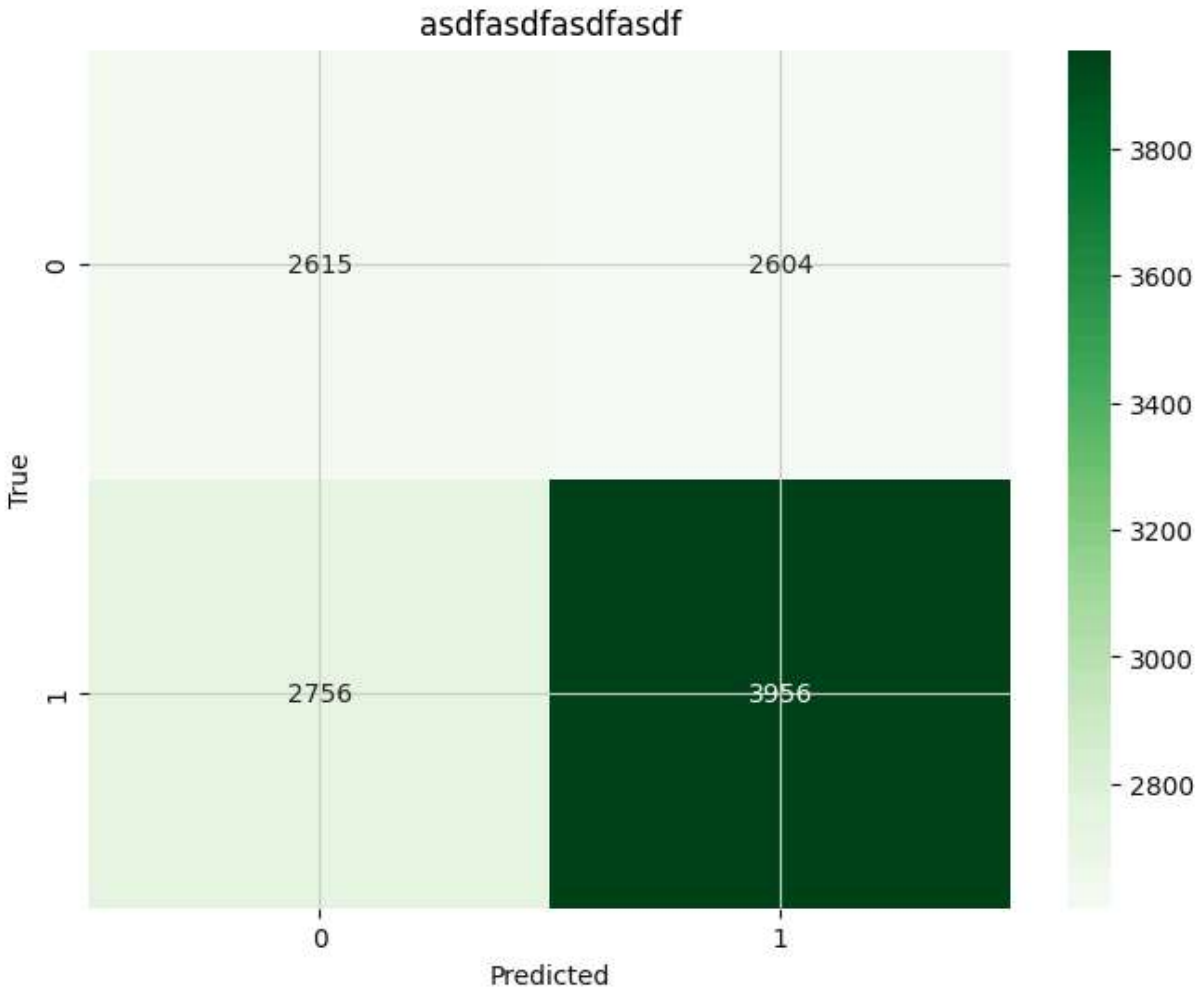
In [28]:
```
cm_vals = []
for i in range(len(m_pred)):
        if(test_on_y[i] == 0):
                if(m_pred[i] == 0):
                        cm_vals.append(0)
                if(m_pred[i] == 1):
                        cm_vals.append(1)
        if(test_on_y[i] == 1):
                if(m_pred[i] == 0):
                        cm_vals.append(2)
                if(m_pred[i] == 1):
                        cm_vals.append(3)
```

In [29]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

df = pd.DataFrame(X_loadraw)
df['score'] = cm_vals
#df['mpred'] = m_pred
df['target'] = y_loadraw
kept_indices = df.index[~(df['score'] % 2 == 0)].tolist()
df = df.drop(df[df['score']%2==0].index).reset_index(drop=True)
df = df.drop(columns=['score']).reset_index(drop=True)
#pd.set_option('display.max_rows',None)
co = df.corr()['target'].drop('target')
print(co.sort_values())
p = co.nlargest(5).index.tolist()
n = co.nsmallest(5).index.tolist()
feats = p+n
```

```
477   -0.072224
478   -0.063864
479   -0.056401
6     -0.047741
465   -0.046398
         ...
53     0.075912
58     0.075932
56     0.076819
54     0.077344
55     0.078332
Name: target, Length: 519, dtype: float64
```

In [30]:

```
from sklearn.svm import SVC
from _Utility import get_class_weights

df_pair = pd.DataFrame(X_loadraw)
df_pair = df_pair.iloc[kept_indices].reset_index(drop=True)
df_pair = df_pair.iloc[:, feats]

X_svm = df_pair.values
y_svm = df['target'].values

clf = SVC(kernel='linear',C=1.0,class_weight=get_class_weights(df['target'])).fit(X

y_svmpred = clf.predict(X_svm)

#Create the confusion matrix
cm = confusion_matrix(df['target'], y_svmpred)
# Plot the confusion matrix using seaborn
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Greens', \
                        xticklabels=range(2), yticklabels=range(2))
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title(f'asdfasdfasdfasdf')
plt.show()
```



In [31]:
```
import matplotlib.pyplot as plt

import _Data_Processing
from importlib import reload
import joblib
reload(_Data_Processing)
lstm_format = False
X_test, _, _, __,\
y_test, _, ___, ____,\
feature_subsets, scaler =\
_Data_Processing.preprocess_data(
        file_name   =           'spx_test.csv'
        ,indp_size  =           0.01
        ,test_size  =           0.01
        ,shfl_splt      =               False
        ,t_start    =           645
        ,t_end          =               800
        ,mod_type       =               'Area_Classification'
        ,target_t       =               45
        ,num_class      =               2
        ,split_val      =               5
        ,verbose        =               0
        ,scaler         =               'Custom'
        ,cstm_scale     =               joblib.load('scaler/tmp.joblib')
        ,frmt_lstm      =               lstm_format
```

```
        ,time_steps =           5
        ,keep_price =           False
        ,indices        =               0
)

import matplotlib.pyplot as plt

import _Data_Processing
from importlib import reload
import joblib
reload(_Data_Processing)
lstm_format = False
X_testraw, _, _, __,\
y_testraw, _, ___, ____,\
feature_subsets, scaler =\
_Data_Processing.preprocess_data(
        file_name   =           'spx_test.csv'
        ,indp_size  =           0.01
        ,test_size  =           0.01
        ,shfl_splt      =               False
        ,t_start    =           645
        ,t_end          =               800
        ,mod_type       =               'Area_Classification'
        ,target_t       =               45
        ,num_class      =               2
        ,split_val      =               5
        ,verbose        =               0
        ,scaler         =               'Standard'
        ,cstm_scale     =               joblib.load('scaler/tmp.joblib')
        ,frmt_lstm      =               lstm_format
        ,time_steps =           5
        ,keep_price =           True
        ,indices        =               0
)
```

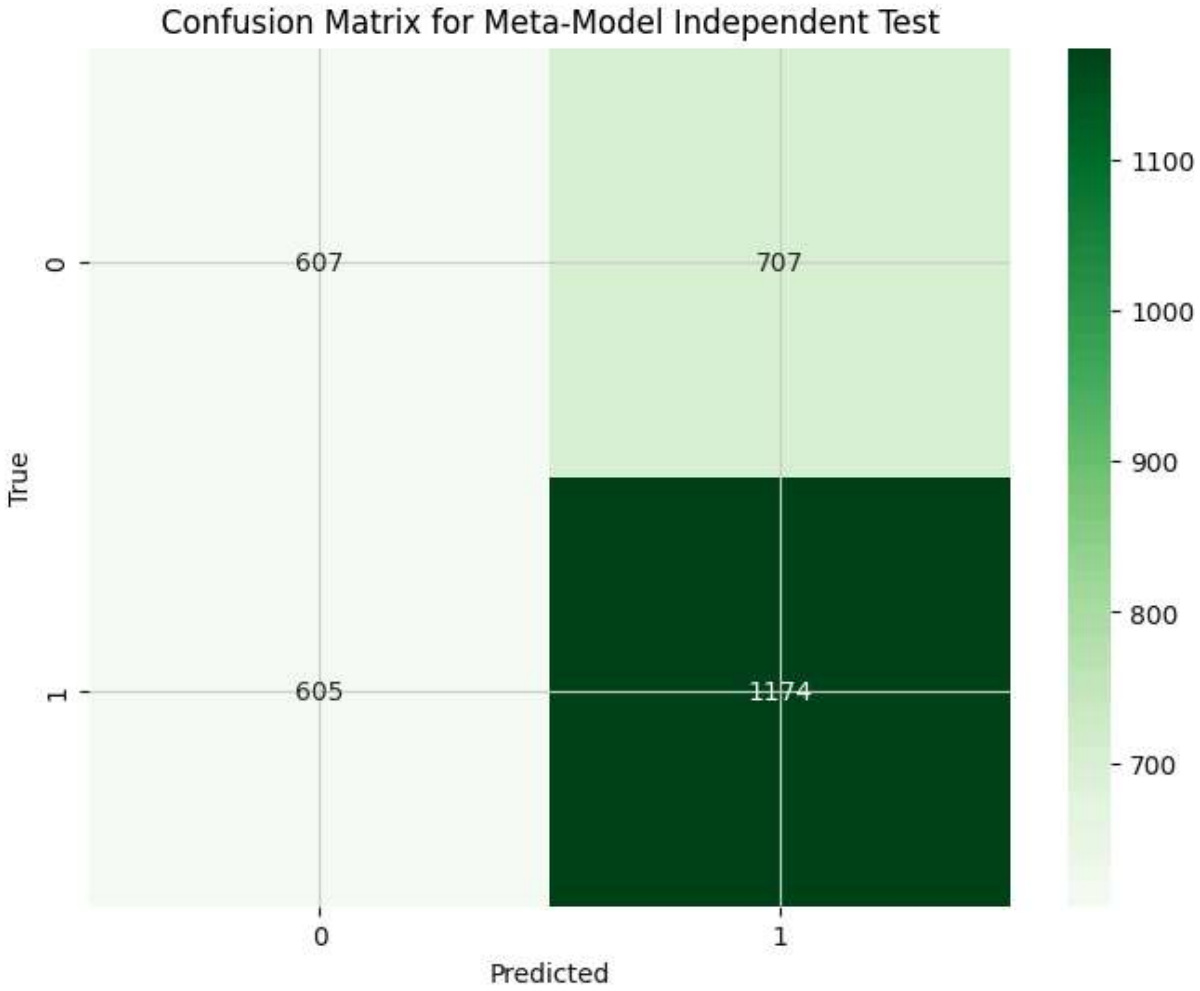
```
loaded chunk 1 of size: 125400164 -> 64600164
loaded chunk 2 of size: 18408884 -> 9395364
concat chunks
concatted chunks
Success.
Size of dataset:        74083444
<class 'numpy.ndarray'> <class 'numpy.ndarray'> <class 'numpy.float64'>
        25577 Samples Dropped.

loaded chunk 1 of size: 125400164 -> 64600164
loaded chunk 2 of size: 18408884 -> 9395364
concat chunks
concatted chunks
Success.
Size of dataset:        74083444
<class 'numpy.ndarray'> <class 'numpy.ndarray'> <class 'numpy.float64'>
        25577 Samples Dropped.
```

In [32]:
```
from sklearn.metrics import precision_score
import numpy as np
```

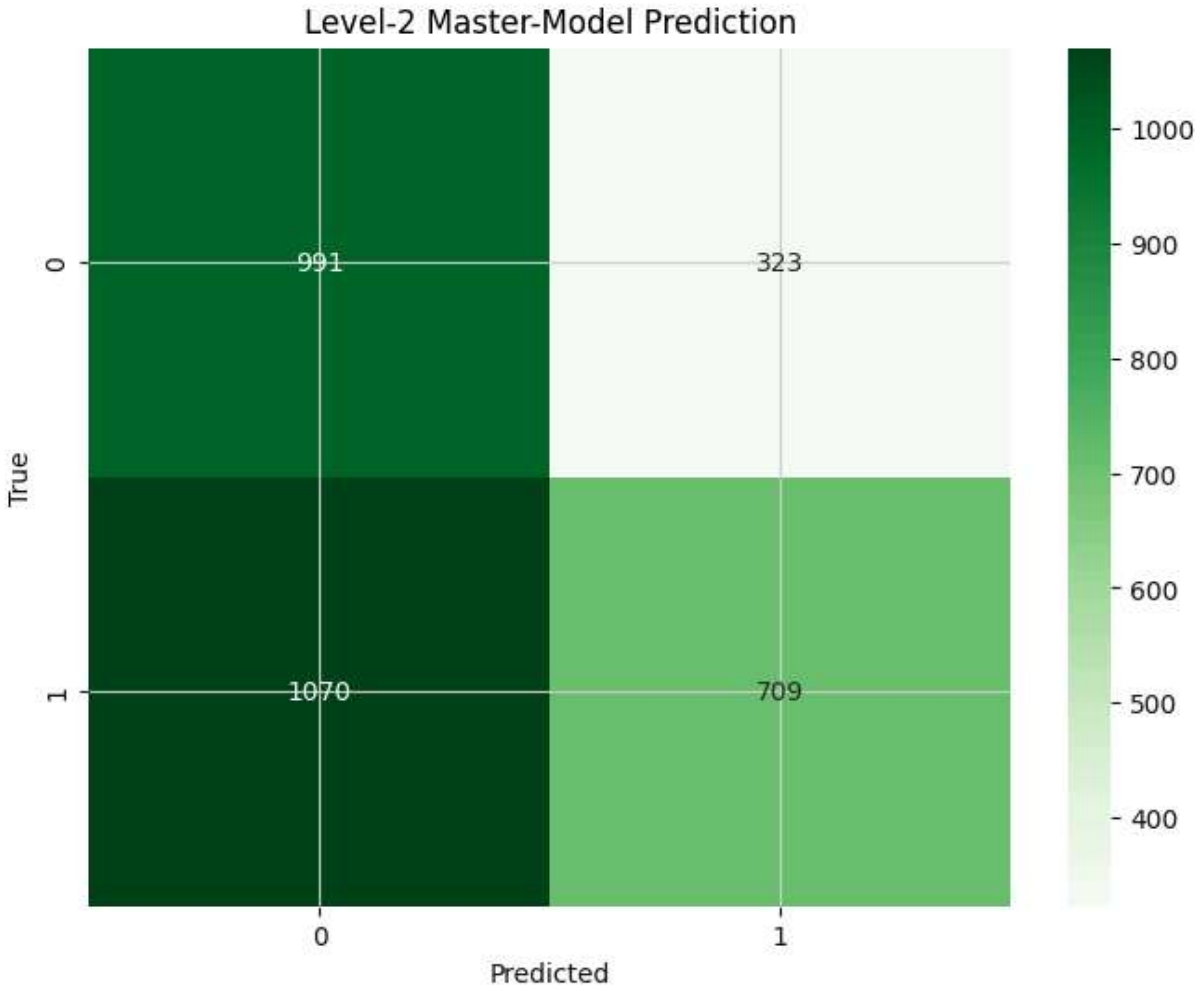
```
loadmodel = _Master_Model.Master(
        model_depth=2
)
loadmodel.load_model('pred1_63p2_acc-645-800')
m_pred = loadmodel.master_predict(X_test, threshold=0.5)
print(accuracy_score(y_test, m_pred))

#Create the confusion matrix
cm = confusion_matrix(y_test, m_pred)
# Plot the confusion matrix using seaborn
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Greens', \
                        xticklabels=range(2), yticklabels=range(2))
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title(f'Confusion Matrix for Meta-Model Independent Test')
plt.show()

cm_vals = []
for i in range(len(m_pred)):
        if(test_on_y[i] == 0):
                if(m_pred[i] == 0):
                        cm_vals.append(0)
                if(m_pred[i] == 1):
                        cm_vals.append(1)
        if(test_on_y[i] == 1):
                if(m_pred[i] == 0):
                        cm_vals.append(2)
                if(m_pred[i] == 1):
                        cm_vals.append(3)

df = pd.DataFrame(X_testraw)
df['score'] = cm_vals
df['mpred'] = m_pred
df['target'] = y_testraw
kept_indices = df.index[~(df['score'] % 2 == 0)].tolist()
#df = df.drop(df[df['score']%2==0].index).reset_index(drop=True)
df = df.drop(columns=['score']).reset_index(drop=True)

df_test = pd.DataFrame(X_testraw)
df_test = df_test.iloc[:, feats]

X_svm = df_test.values
y_svm = df['target'].values

y_svmpred = m_pred

#'polishing' predictions based on if level-1 predicted (1)
for p in range(len(y_svmpred)):
        if(y_svmpred[p] == 1):
                y_svmpred[p] = clf.predict(X_svm[p].reshape(1, -1))

print(f'INDEPENDENT SET LEVEL-2 PRECISION: {precision_score(df['target'], y_svmpred
#Create the confusion matrix
cm = confusion_matrix(df['target'], y_svmpred)
```

```
# Plot the confusion matrix using seaborn
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Greens', \
                        xticklabels=range(2), yticklabels=range(2))
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title(f'Level-2 Master-Model Prediction')
plt.show()
```

```
97/97 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step
97/97 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step
97/97 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step
97/97 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step
97/97 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step
97/97 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step
97/97 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step
97/97 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step
(3093, 8)
97/97 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
0.5758163595215001
```



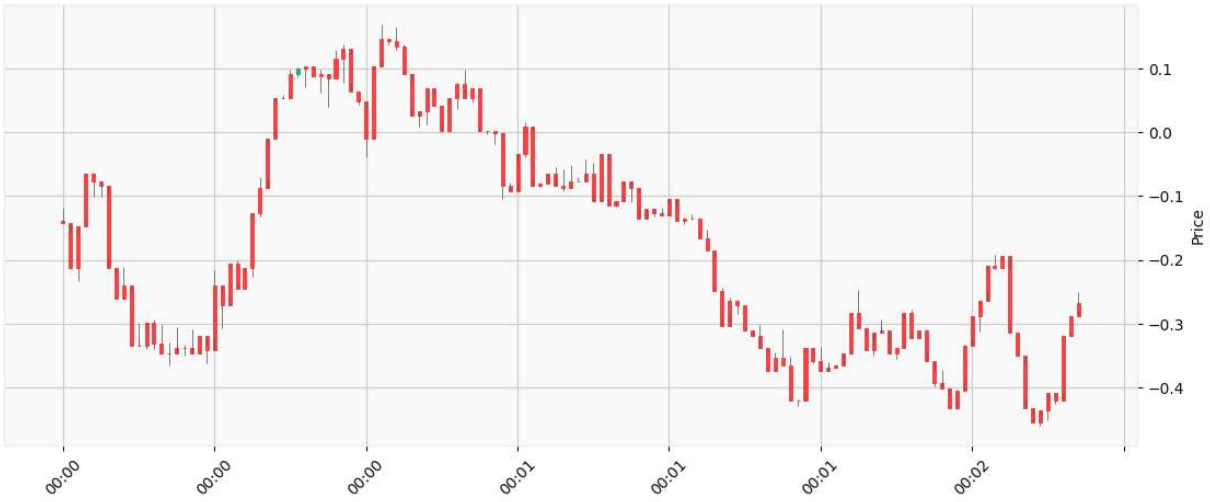

```
INDEPENDENT SET LEVEL-2 PRECISION: 0.687015503875969
```

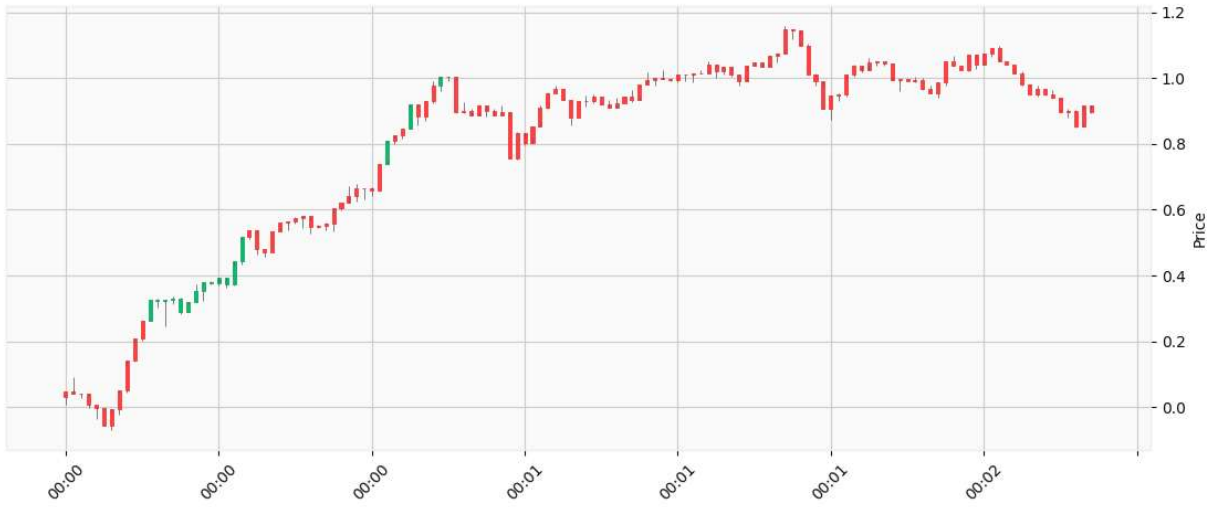


In [33]:
```
import mplfinance as mpf
import pandas as pd
import numpy as np
import copy
#from _Utility import swap

y_true = y_svm
y_pred = m_pred
y_pred2= y_svmpred

X_raw  = X_testraw

print(int(len(X_raw[:,0])/155))

num_candles = 155

iter = 0

for section in range(int(len(X_raw[:,0])/num_candles)):
        section*=num_candles
        section_end = section+num_candles
        X_thold = copy.deepcopy(X_testraw[section:section_end,:])

        h = X_thold[:,0]
        l = X_thold[:,1]
```

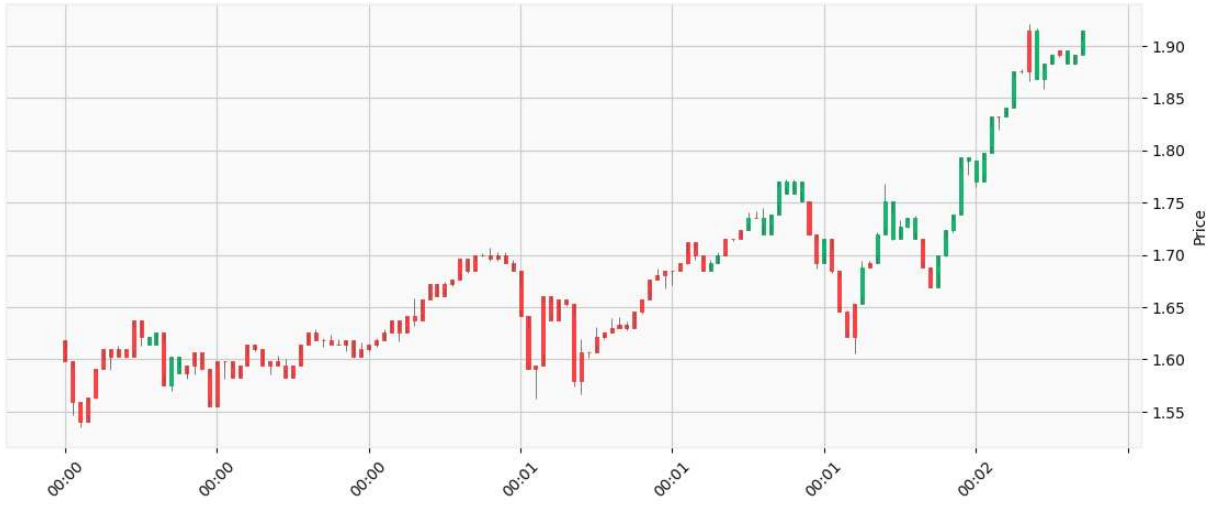
```
        c = X_thold[:,2]
        o = np.roll(c, shift=1)

        #small for loop to force direction of candle based on prediction of model
        for i in range(len(c)):
                #if predicts 1
                if(y_pred[section+i]==1):
                        if(y_pred2[iter]==1):
                                if(c[i]<o[i]):#force green
                                        c[i],o[i] = o[i],c[i]
                        else:
                                if(c[i]>o[i]):#force red
                                        c[i],o[i] = o[i],c[i]
                        iter = iter + 1
                else:
                        if(c[i]>o[i]):#force red
                                c[i],o[i] = o[i],c[i]


        data = {
                'Date':range(0,len(X_thold[:])*1000000000,1000000000),
                'Open':o,
                'High':h,
                'Low':l,
                'Close':c
        }
        df = pd.DataFrame(data)
        #df['color'] = colors
        df['Date'] = pd.to_datetime(df['Date'])
        df.set_index('Date',inplace=True)

        mpf.plot(df[10:-10], type='candle',style='yahoo',figratio=(20,8))
        #mpf.plot(df, addplot=plot,style=custom_style,figratio=(20,8))
```

19



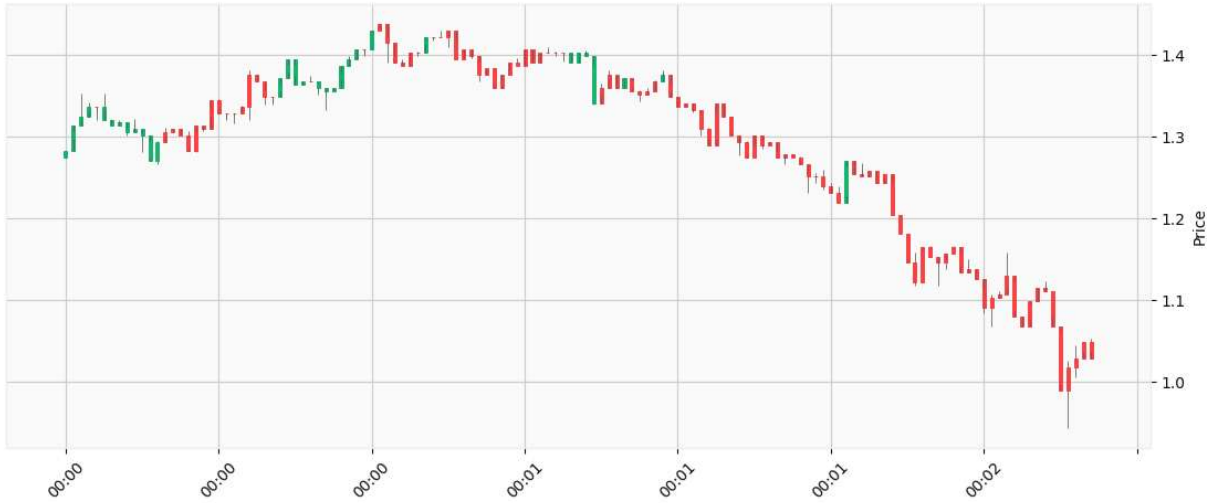

1.2
1.0
0.8
0.6
0.4
0.2
0.0
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02

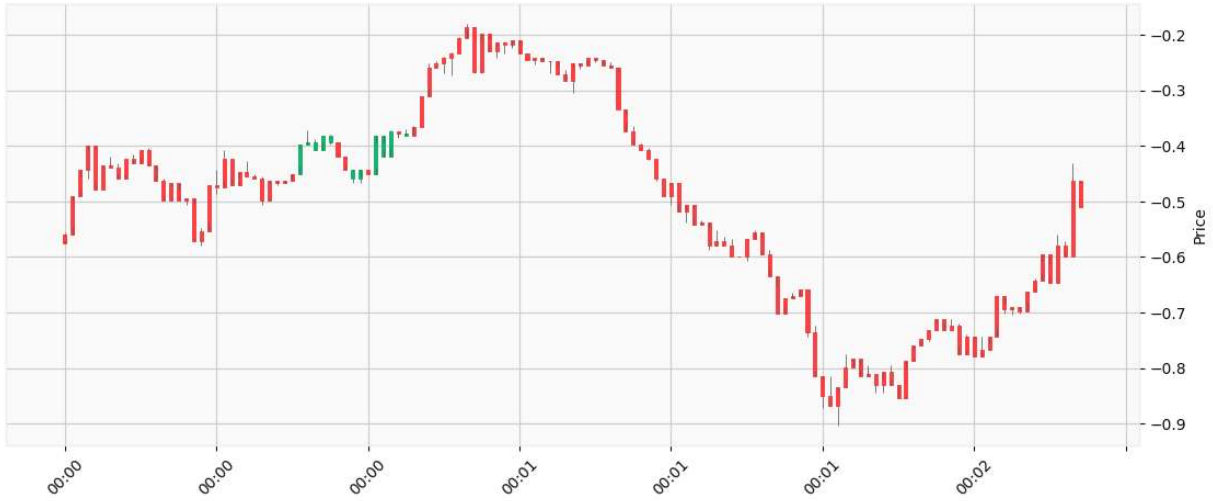

1.90
1.85
1.80
1.75
1.70
1.65
1.60
1.55
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02

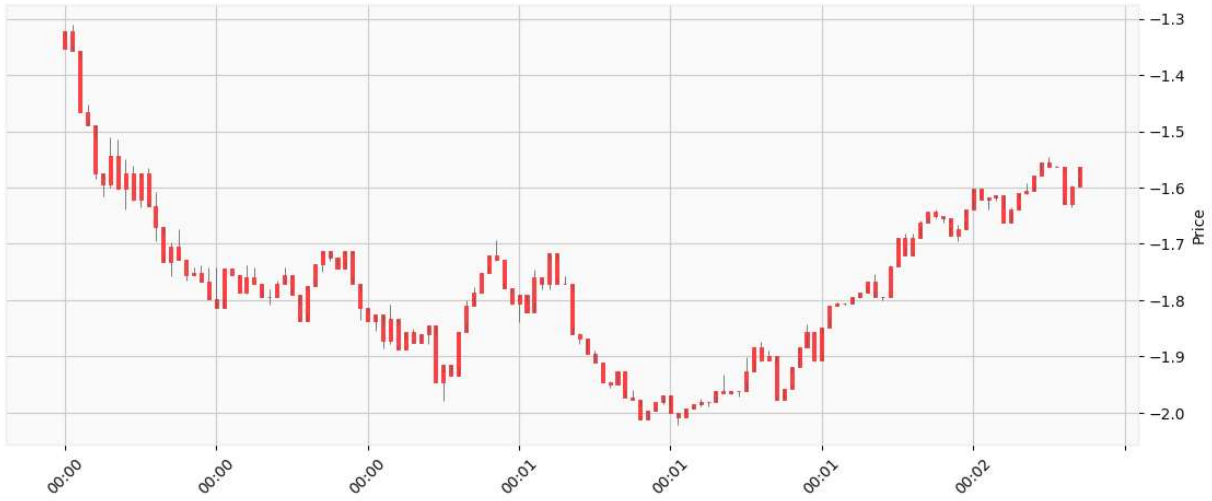

2.05
2.00
1.95
1.90
1.85
1.80
1.75
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02
0.7
0.6
0.5
0.4
0.3
0.2
0.1
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02
0.2
0.0
−0.2
−0.4
−0.6
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02

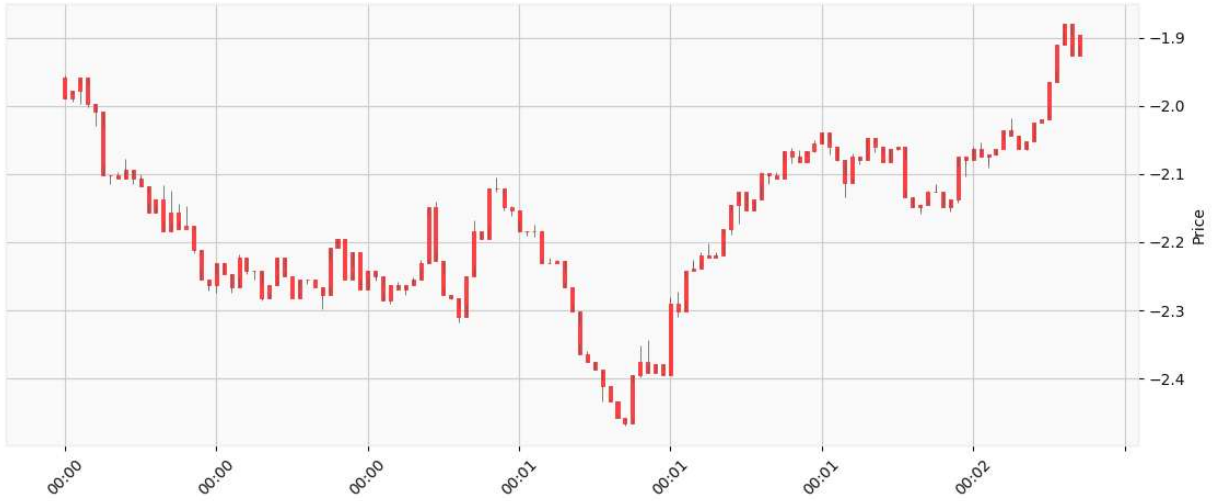

Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02

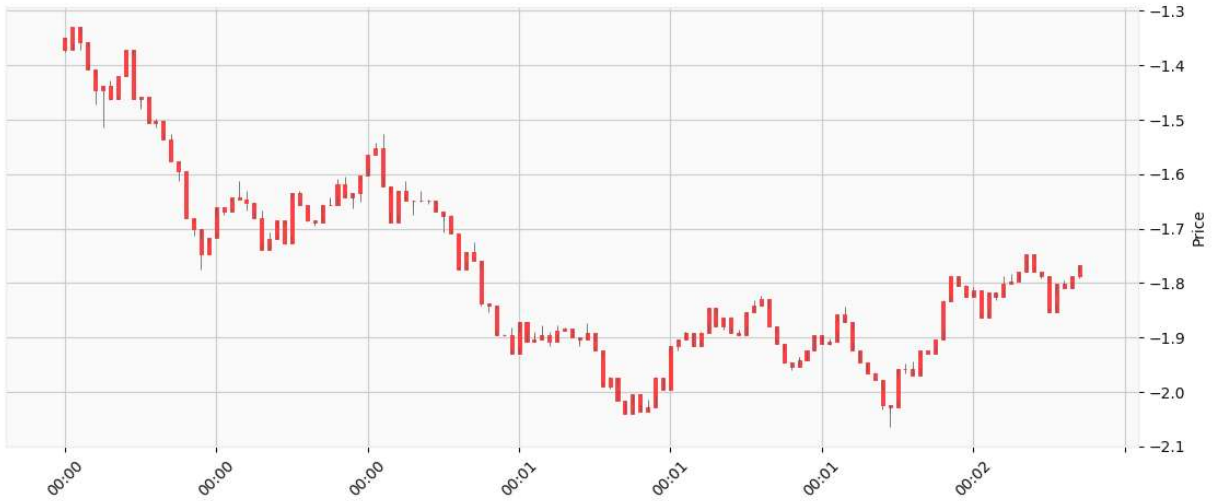

1.4
1.3
1.2
1.1
1.0
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02

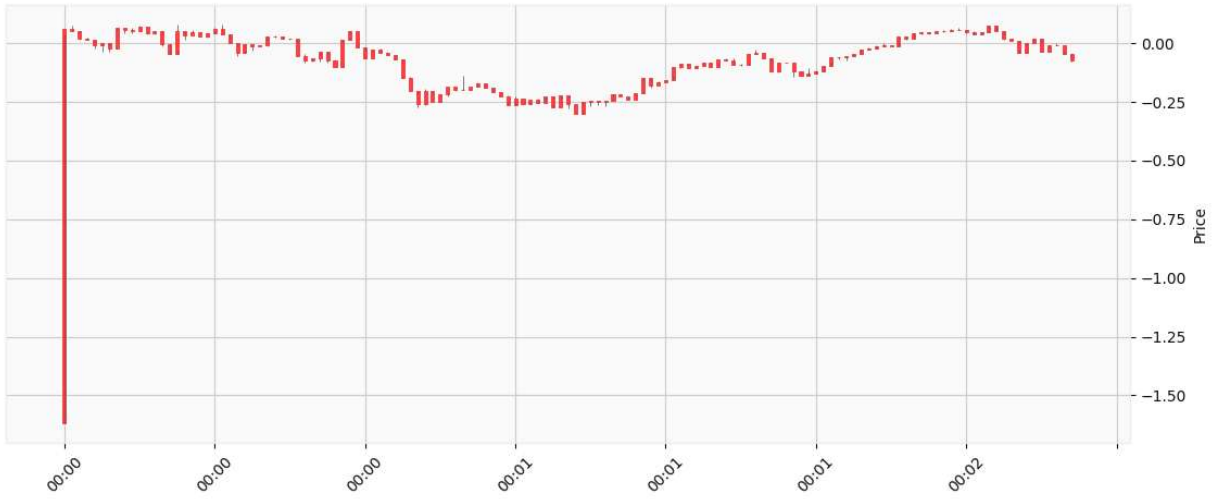

0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02

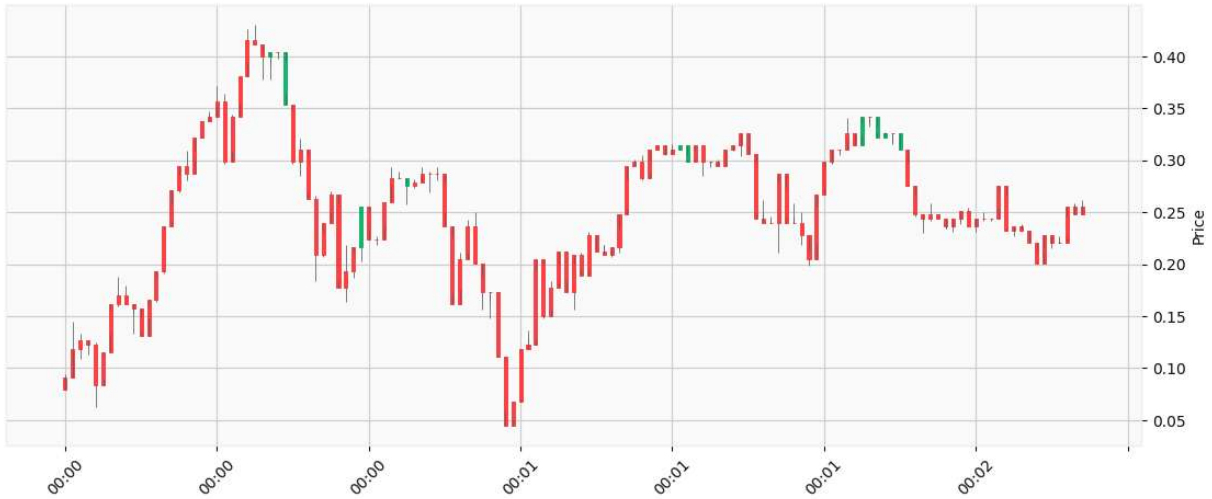

−0.2
−0.3
−0.4
−0.5
−0.6
−0.7
−0.8
−0.9
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02

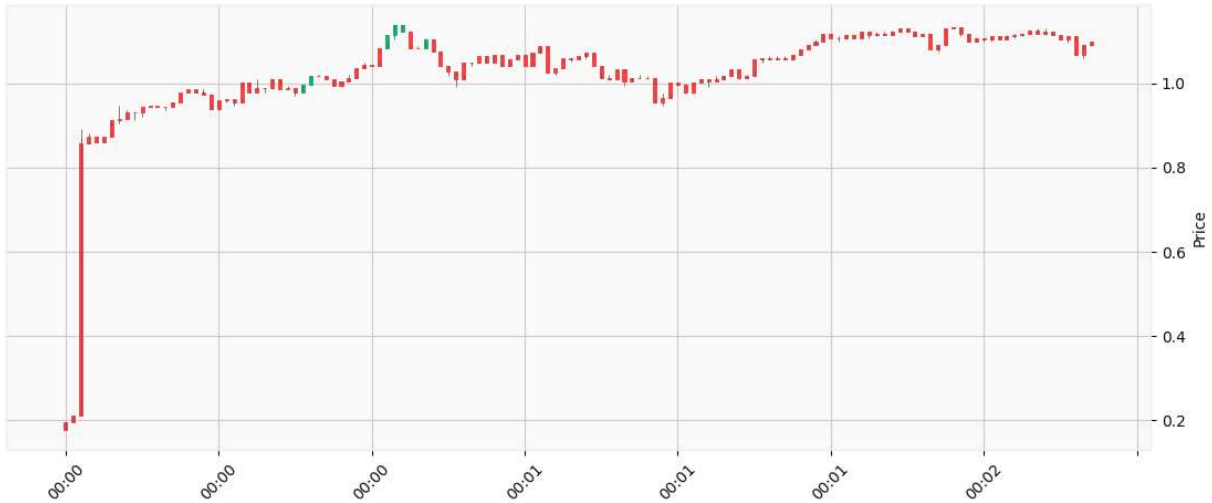

−1.3
−1.4
−1.5
−1.6
−1.7
−1.8
−1.9
−2.0
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02


−1.9
−2.0
−2.1
−2.2
−2.3
−2.4
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02


−1.3
−1.4
−1.5
−1.6
−1.7
−1.8
−1.9
−2.0
−2.1
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02


0.00
−0.25
−0.50
−0.75
−1.00
−1.25
−1.50
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02


0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02


1.0
0.8
0.6
0.4
0.2
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02