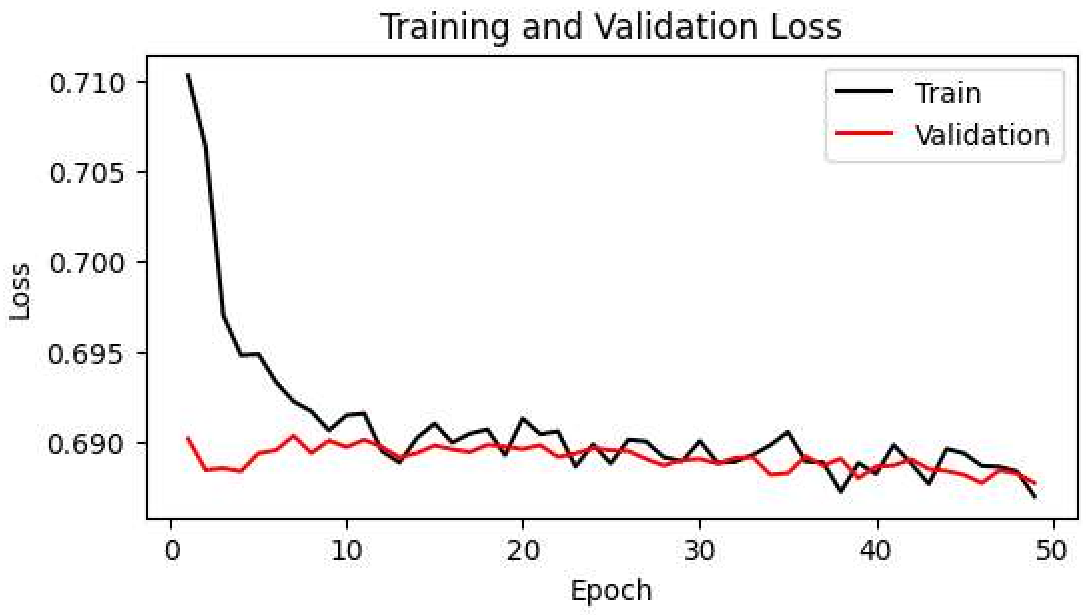
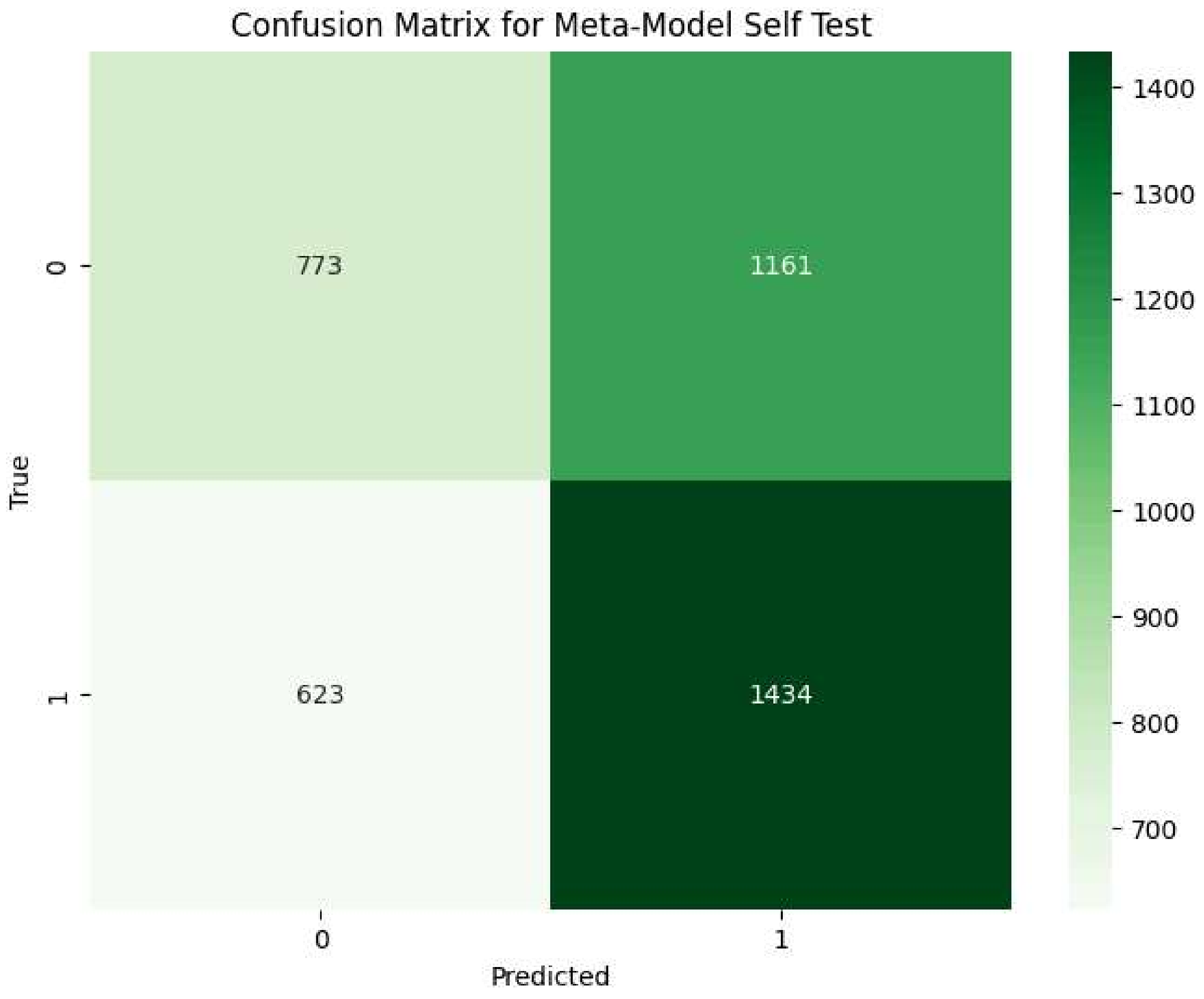
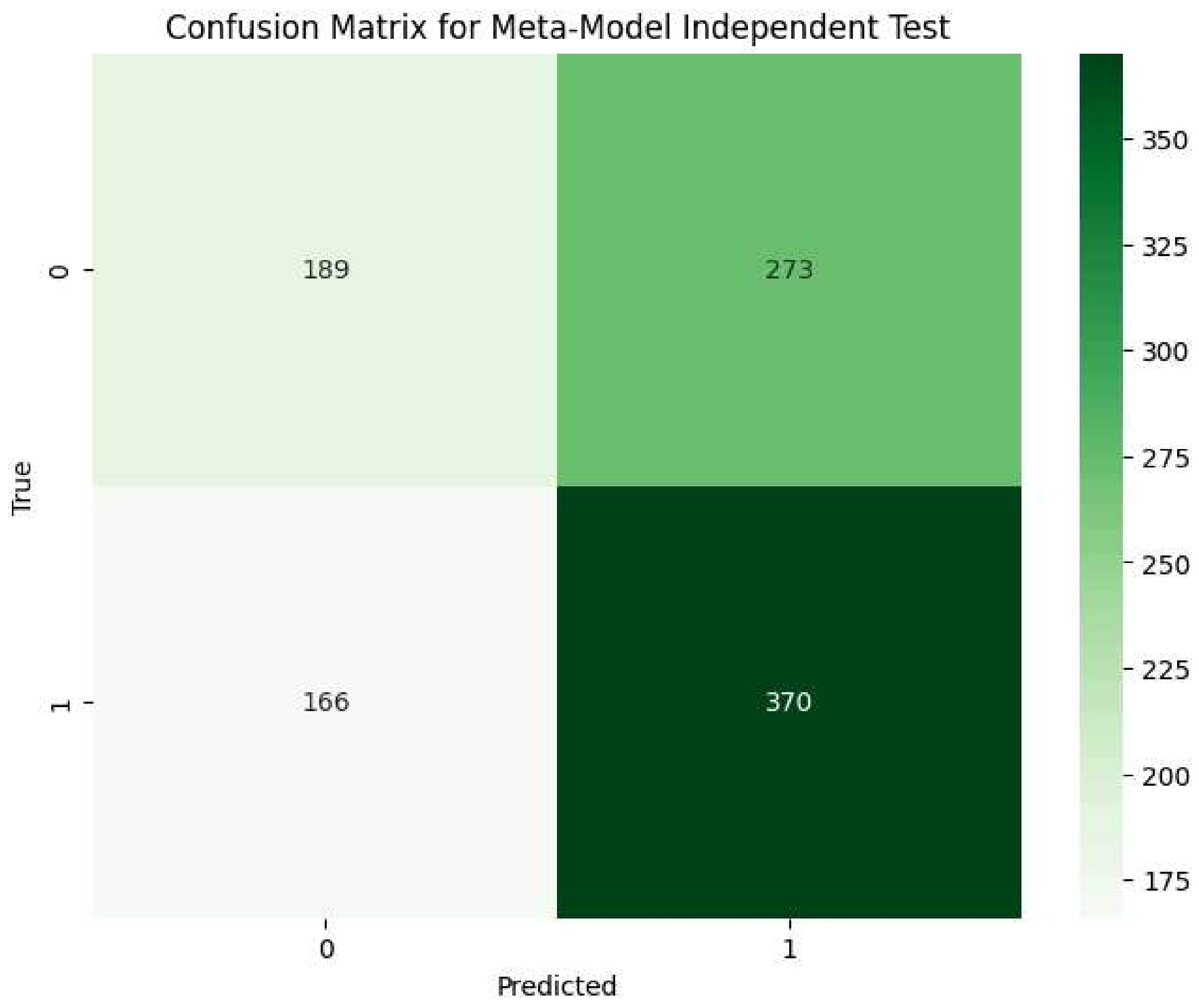


```
125/125 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
META-MODEL SELF TEST:
        Accuracy:       0.55
        Precision:      0.55
        Recall:         0.7
```



```
META-MODEL INDEPENDENT TEST:
        Accuracy:       0.56
        Precision:      0.58
        Recall:         0.69
```



In [ ]:

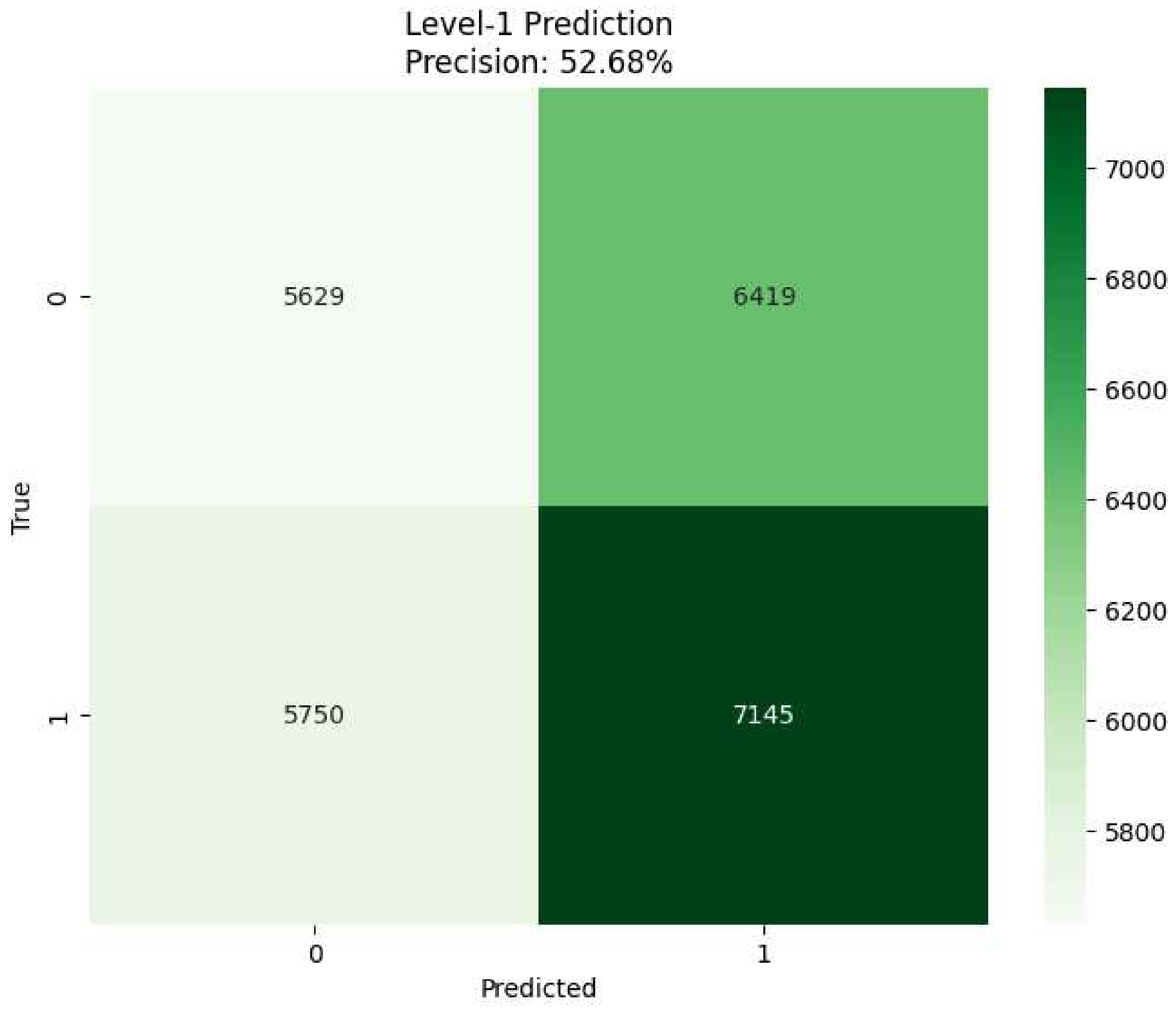
In [6]:
```
'''NOTE load in data for testing quality of level-1/2 models'''

import _Data_Processing
from importlib import reload
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       =               15
        ,num_class      =               2
        ,split_val      =               5
        ,verbose        =               1
        ,scaler         =               'Custom'
    ,cstm_scale =               joblib.load('scaler/tmp.joblib')
        ,frmt_lstm      =               lstm_format
        ,keep_price =           True
    ,indices    =               0
)
```

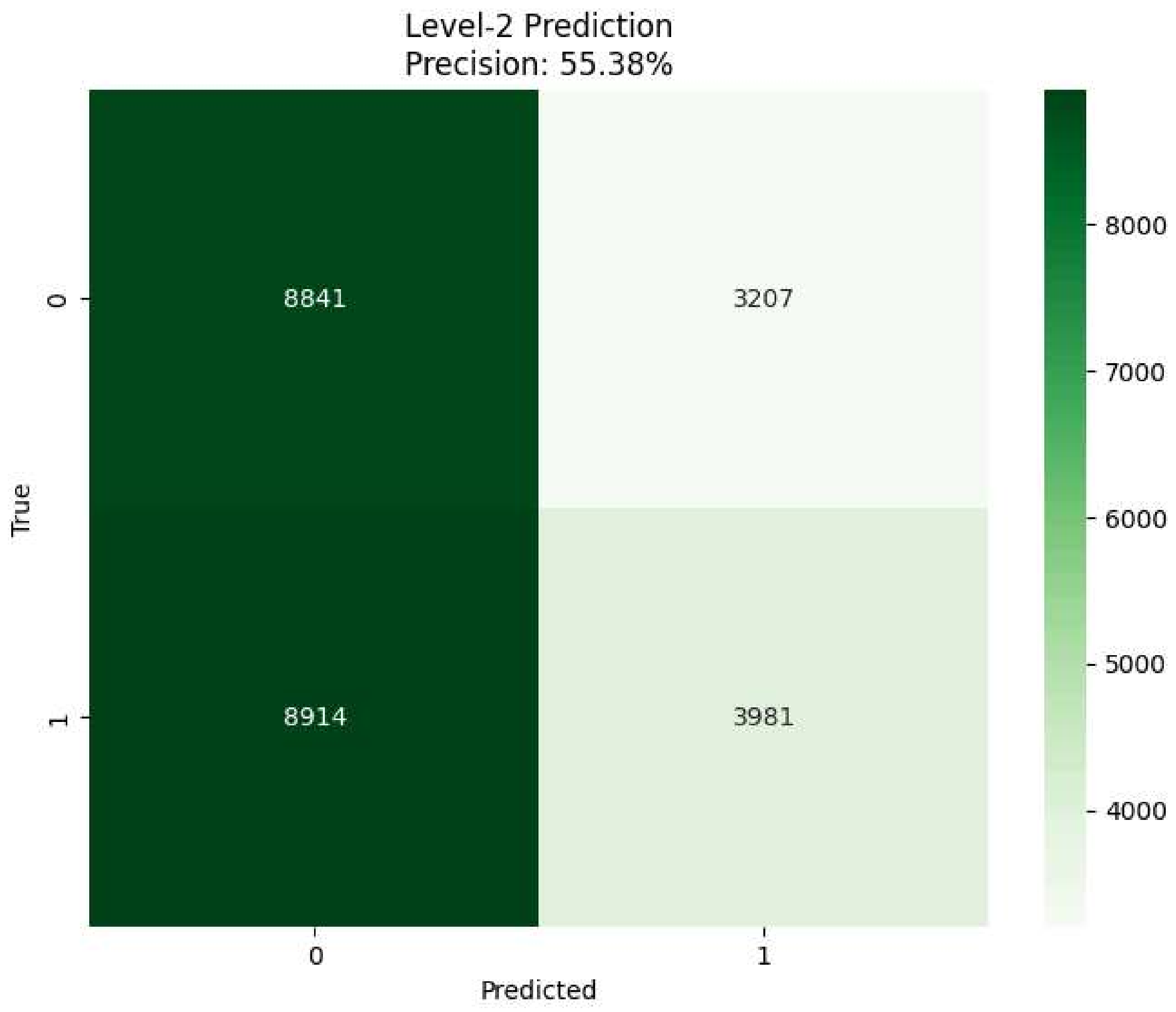
```
Trying to load CSV file into DataFrame...
loaded chunk 1 of size: 125400164 -> 64600164
loaded chunk 2 of size: 18408884 -> 9395364
concat chunks
concatted chunks
Success.
Size of dataset:        74083444
Trying to drop unused targets...Success.
Trying to collect indices of wanted times...Success...

# of Samples:   28670

# of Features:  519

Target:         tc_2a_15m

Trying to split DataFrame into X and y...Success.
Trying to collect all feature names and indices...Success.
Trying to clean up...Success.
Trying to encode y and make class weights...Failed [NON-FATAL: NOT IMPLEMENTED]
Trying to standardize all featurespace from training featurespace...Success.
Trying to drop unwanted time-range samples...Success.
        25577 Samples Dropped.

Trying to split X and y into Train/Validation/Independent...Success.
Trying to clean up...Success.
X_train:        (3031, 519).
y_train:        (3031,).
X_val:          (31, 519).
y_val:          (31,).
X_ind:          (31, 519).
y_ind:          (31,).
Collecting garbage...Success.
Terminating.
```

In [7]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import _Master_Model
reload(_Master_Model)
reload(_Utility)
from importlib import reload
from sklearn.svm import SVC
from _Utility import get_class_weights
```

```
shallow_master = _Master_Model.Master(
        model_depth             =       2
        ,all_models     =       [models, metamodel]
        ,lvl0_formatters=       [X_find_parts, X_trans_parts]
)

pred_2 = shallow_master.master_predict(X)
cm_vals = _Utility.get_cm_values(y, pred_2)

_Utility.show_confusion_matrix(y,pred_2,title=f'Level-1 Prediction\nPrecision: {rou

df = pd.DataFrame(X)
df['score'] = cm_vals
df['target'] = y
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

df_pair = pd.DataFrame(X)
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=_Utility.get_class_weights(df['target'

y_svmpred = pred_2#clf.predict(X_svm)
#'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[p,feats].reshape(1, -1))

#_Utility.show_confusion_matrix(df['target'],y_svmpred,title=f'Level-2 Prediction\n
_Utility.show_confusion_matrix(y,y_svmpred,title=f'Level-2 Prediction\nPrecision: {
```

```
780/780 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step
780/780 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step
780/780 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step
780/780 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
780/780 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step
780/780 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
780/780 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
780/780 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
(24943, 8)
780/780 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step
```


Level-1 Prediction
Precision: 52.68%
True
0
1
5629
6419
5750
7145
Predicted
0
1
7000
6800
6600
6400
6200
6000
5800

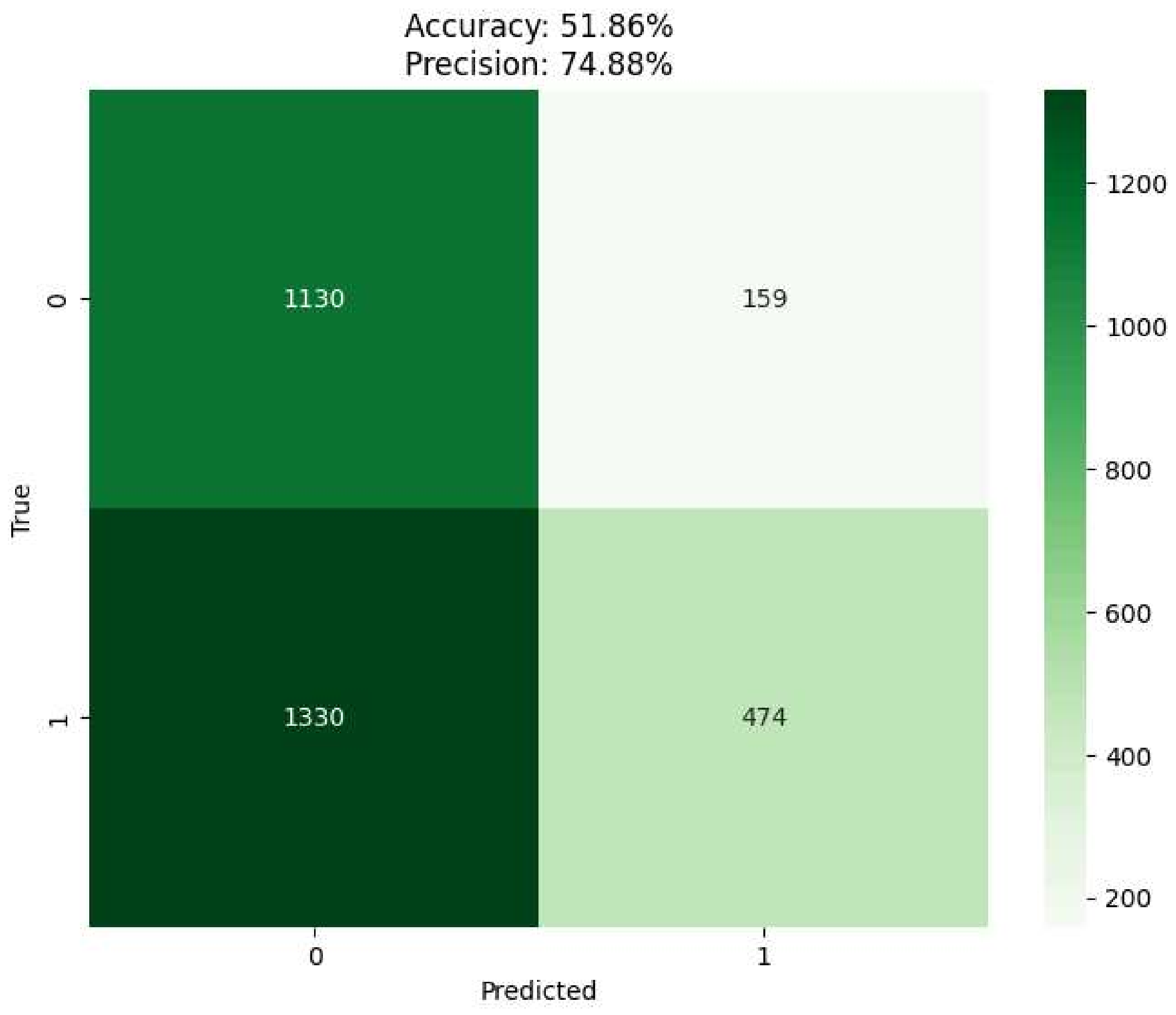


```
In [ ]: reload(_Master_Model)
        reload(_Utility)
        deep_master = _Master_Model.Master(
                model_depth             =       3
                ,all_models     =       [models, metamodel.model, clf]
                ,lvl0_formatters=       [X_find_parts, X_trans_parts]
            ,lvl2_formatters=    [feats]
        )

        y_3pred = deep_master.master_predict(X_test)

        _Utility.show_confusion_matrix(y_test, y_3pred, title=f'Accuracy: {_Utility.get_acc
```

```
In [ ]: import _Master_Model
        from importlib import reload
        reload(_Master_Model)
        #deep_master = _Master_Model.Master(model_depth=3)
        #deep_master.load_model('pre63p2-645-800')
        deep_master.save_model('models/m15_241')
```

```
In [ ]: import _Time_Ensemble
        import _Master_Model
        import joblib
import _Utility

'''NOTE load in data for testing quality of level-1/2 models'''

import _Data_Processing
from importlib import reload
reload(_Data_Processing)
reload(_Master_Model)
reload(_Time_Ensemble)
lstm_format = False
X_te, _, _, __,\
y_te, _, ___, ____,\
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       =               60
        ,num_class      =               2
        ,split_val      =               5
        ,verbose        =               0
        ,scaler         =               'Custom'
    ,cstm_scale =               joblib.load('scaler/tmp.joblib')
        ,frmt_lstm      =               lstm_format
        ,keep_price =           True
    ,indices    =               0
)

chronos_array = _Time_Ensemble.chronos_predict(X_te, \
['pre63p2-645-800','models/m55','models/m60','models/m50'])
```

In [2]:
```
reload(_Utility)
reload(_Time_Ensemble)
t_pred  =       _Time_Ensemble.chronos_fusion(master_predictions=chronos_array, fus
_Utility.show_confusion_matrix(y_te, t_pred, title=f'Accuracy: {_Utility.get_accura
```

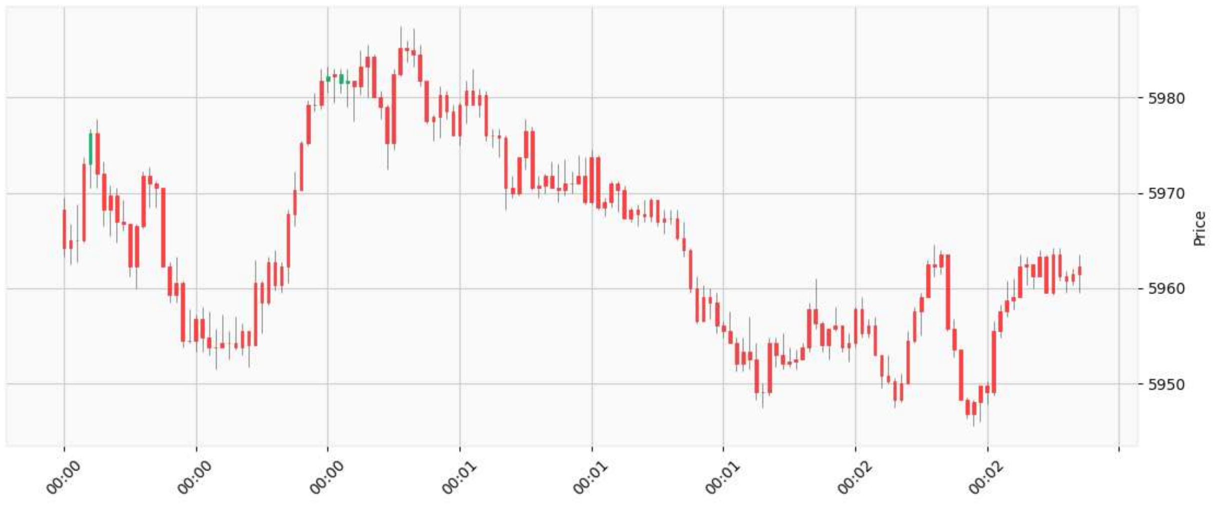


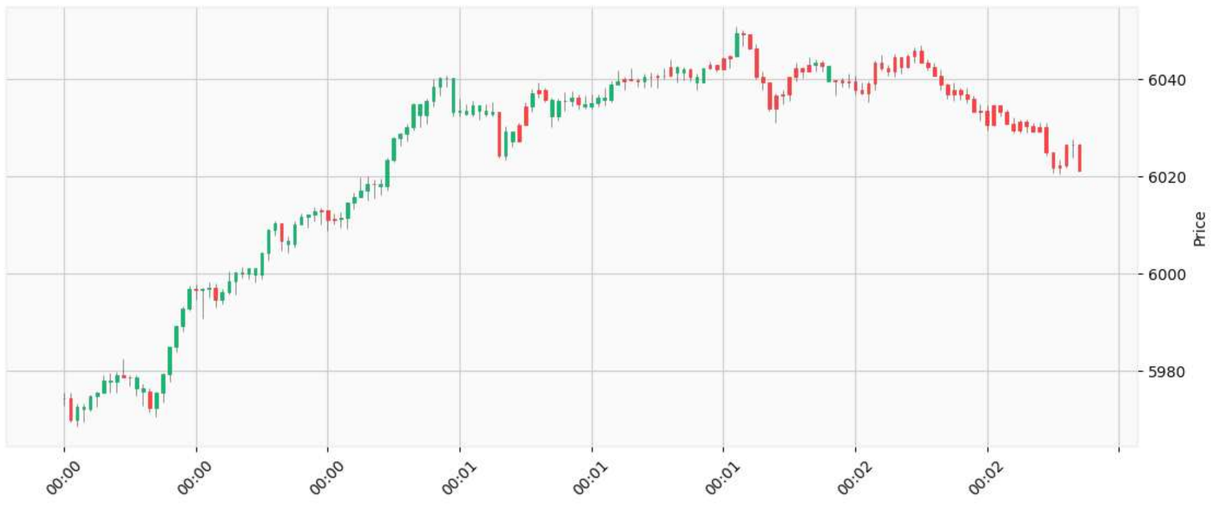
In [3]:
```
#reload(_Utility)
#reload(_Time_Ensemble)
#vals = _Utility.graph_range(_Time_Ensemble.chronos_fusion, kw='vote_var', kw_range
#scores = [precision_score(y_te, vals[i]) for i in range(0,5)]
#_Utility.plot_standard_line(scores, range(0,5))
```

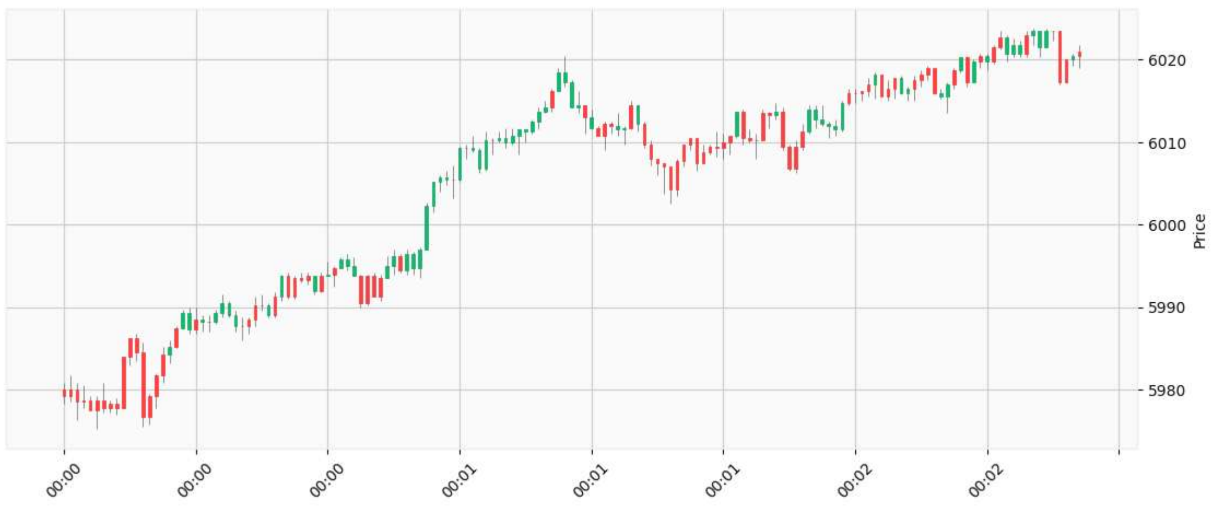
In [4]:
```
#showing chart of predictions
X_raw, _, _, __,\
y_raw, _, ___, ____,\
feature_subsets, scaler =\
_Data_Processing.preprocess_data(
        file_name   =           'spx_test.csv'
        ,indp_size  =           0.01
        ,test_size  =           0.01
        ,shfl_splt      =               False
        ,t_start    =           645
        ,t_end          =               800
        ,mod_type       =               'Area_Classification'
        ,target_t       =               60
        ,num_class      =               2
        ,split_val      =               5
        ,verbose        =               0
        ,scaler         =               'None'
    ,cstm_scale =               joblib.load('scaler/tmp.joblib')
```

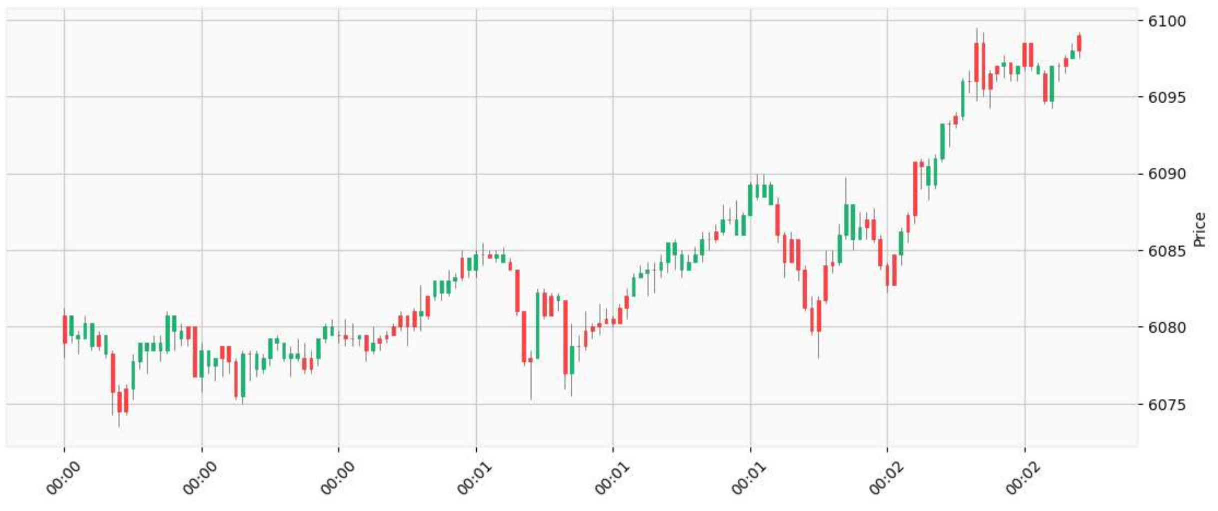
```
        ,frmt_lstm      =               lstm_format
        ,keep_price =           True
    ,indices    =               0
)

reload(_Utility)
_Utility.show_predictions_chart(X_raw=X_raw,predictions=t_pred, t_start=645, t_end=
```

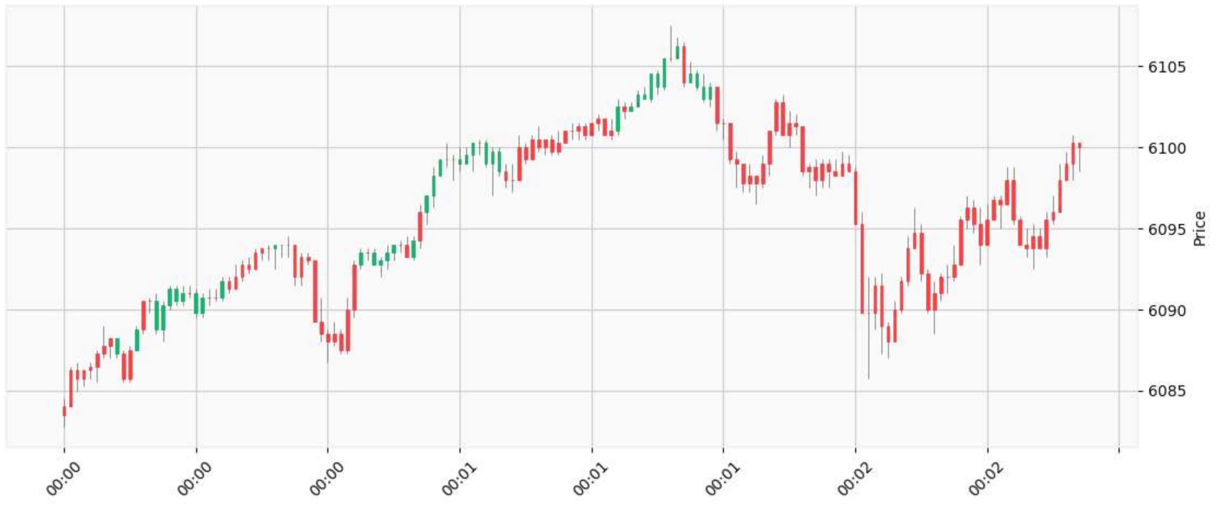
```
loaded chunk 1 of size: 125400164 -> 64600164
loaded chunk 2 of size: 18408884 -> 9395364
concat chunks
concatted chunks
Success.
Size of dataset:        74083444
        25577 Samples Dropped.
```

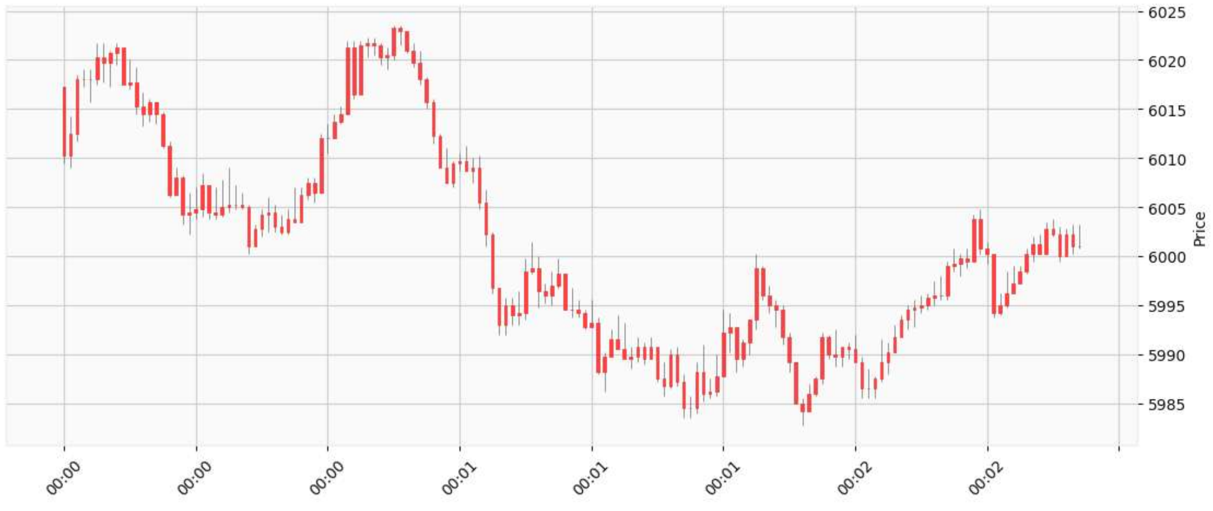




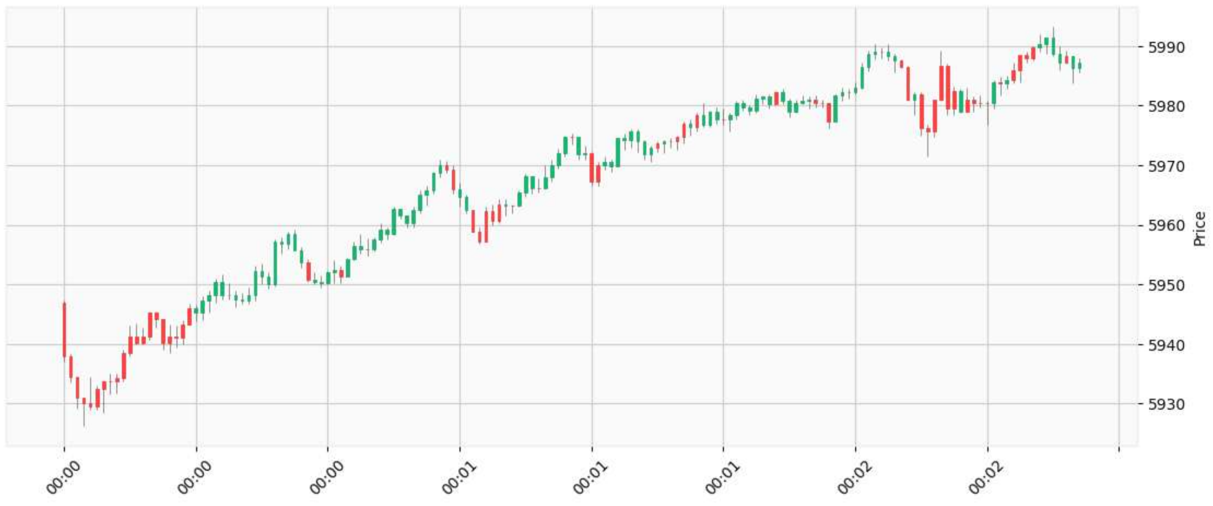

6020
6010
6000
5990
5980
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02
00:02


6100
6095
6090
6085
6080
6075
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02
00:02

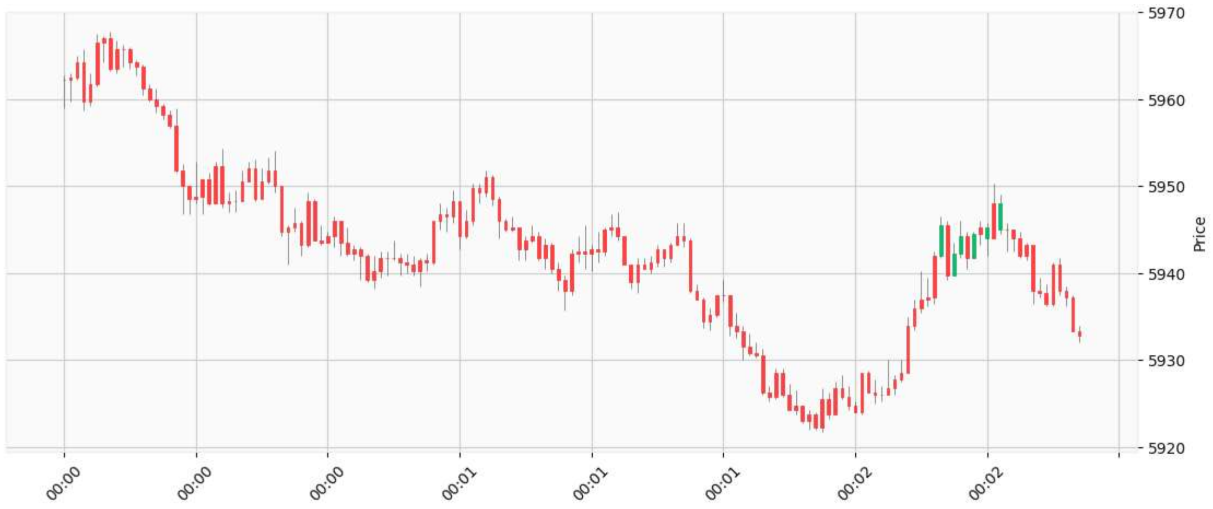

6105
6100
6095
6090
6085
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02
00:02



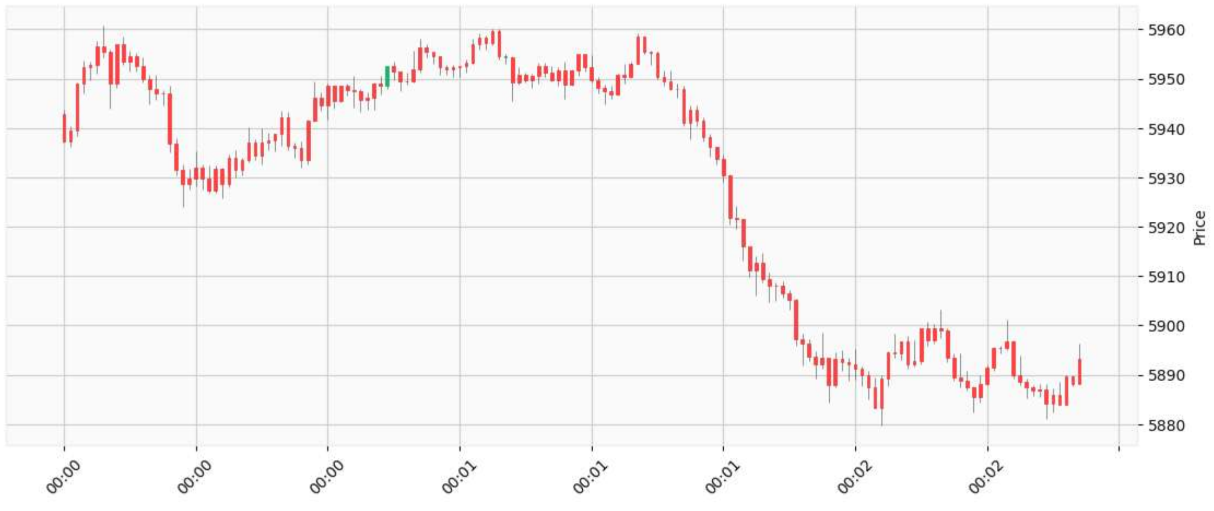
6025
6020
6015
6010
6005
6000
5995
5990
5985
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02
00:02

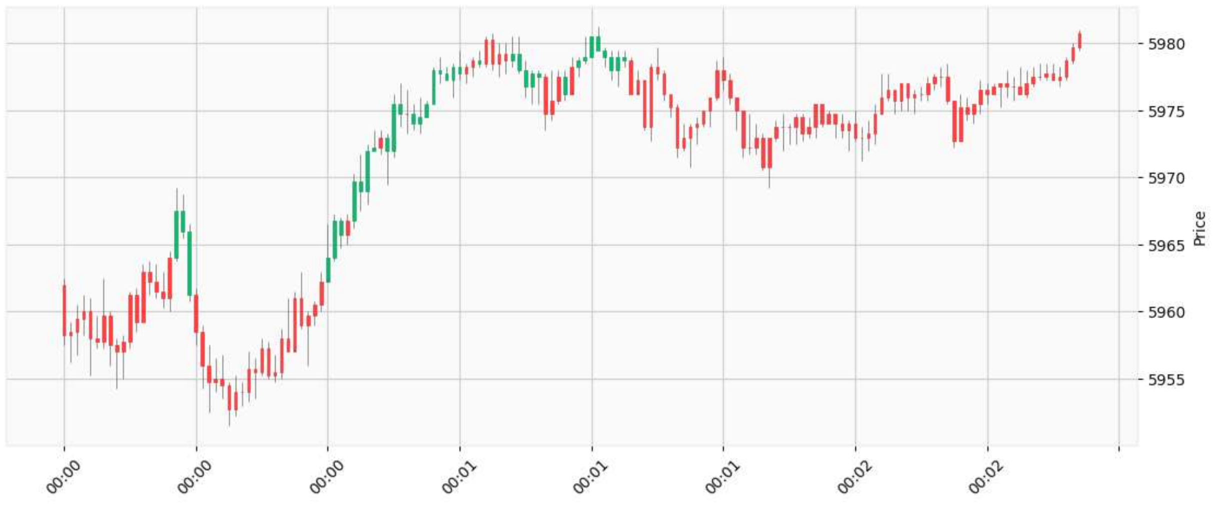


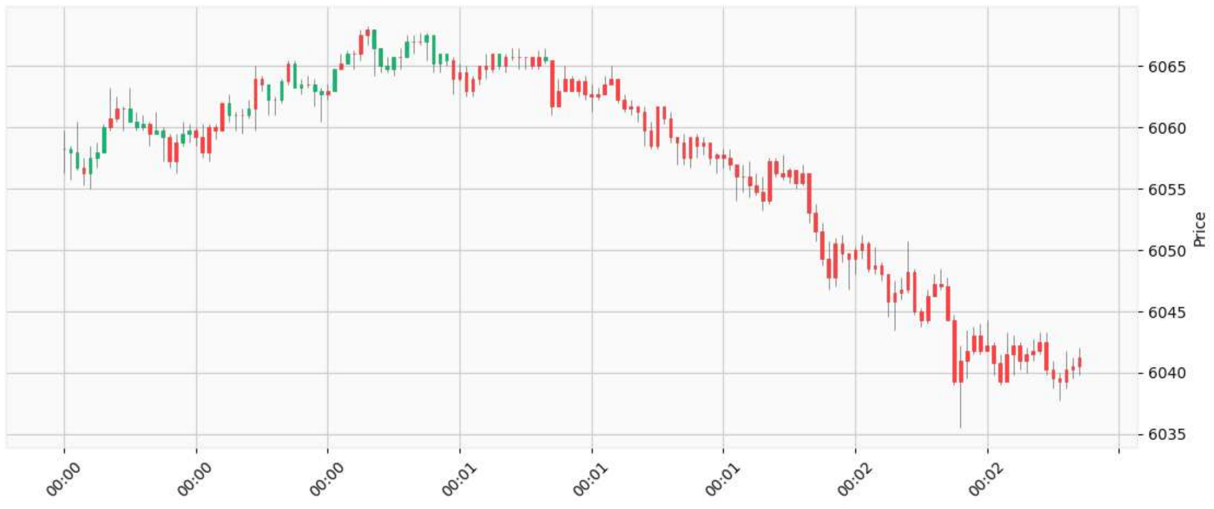
5990
5980
5970
5960
5950
5940
5930
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02
00:02

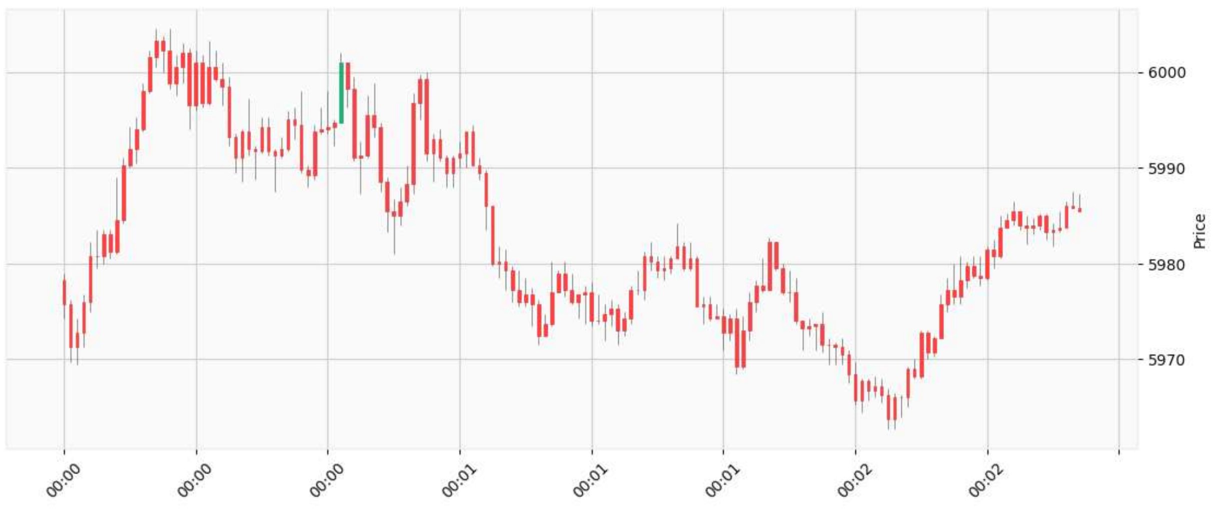


5970
5960
5950
5940
5930
5920
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02
00:02

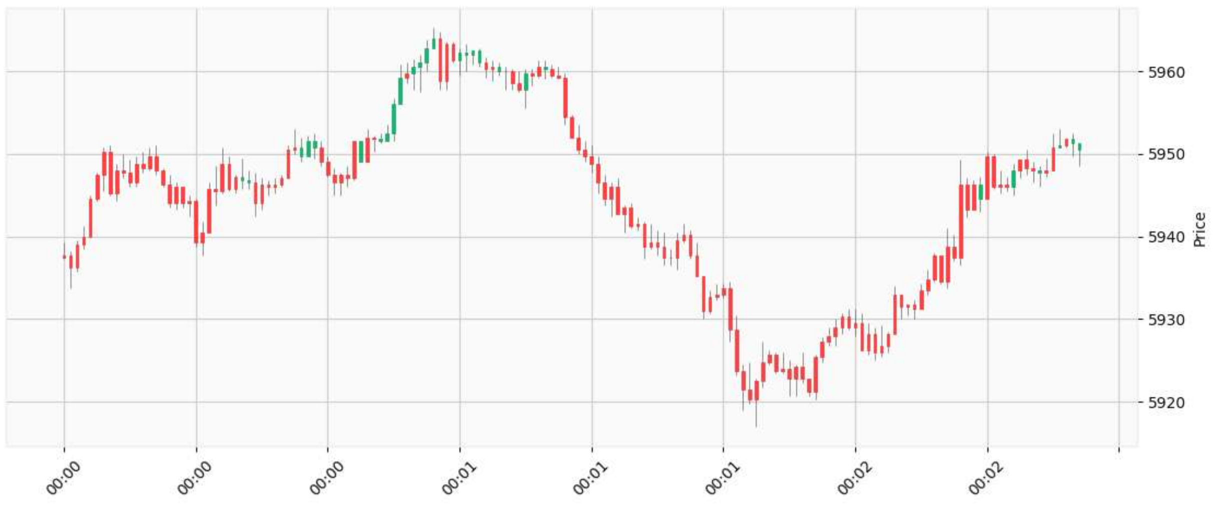

5960
5950
5940
5930
5920
5910
5900
5890
5880
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02
00:02


5980
5975
5970
5965
5960
5955
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02
00:02

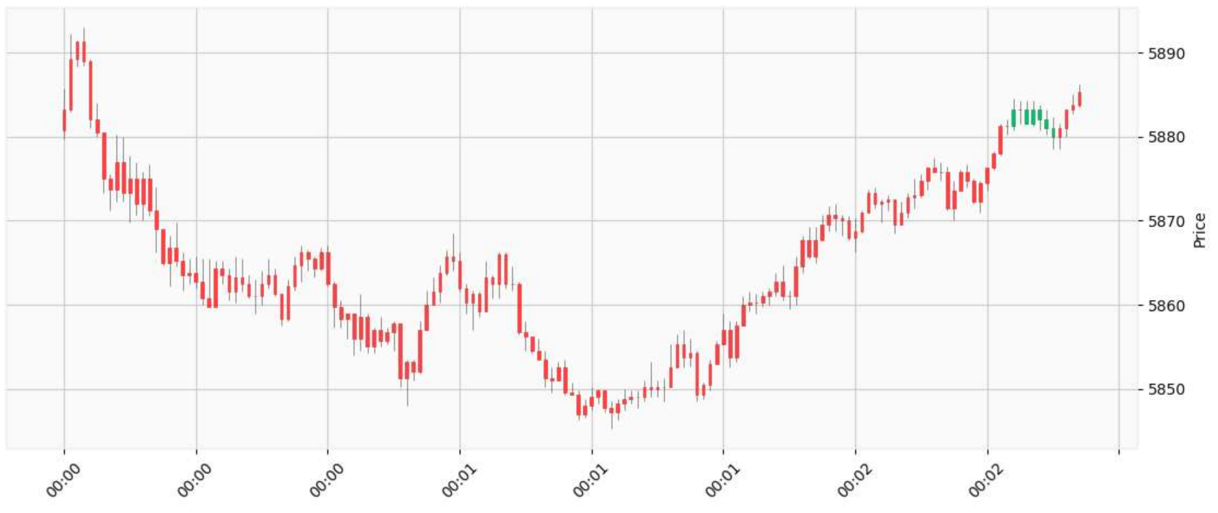

6065
6060
6055
6050
6045
6040
6035
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02
00:02



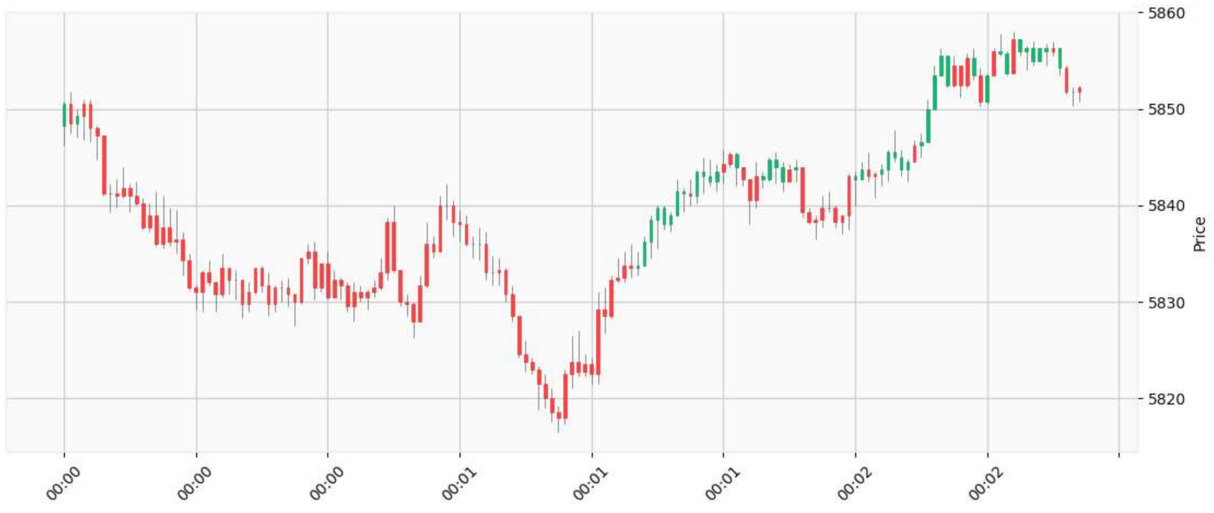
6000
5990
5980
5970
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02
00:02

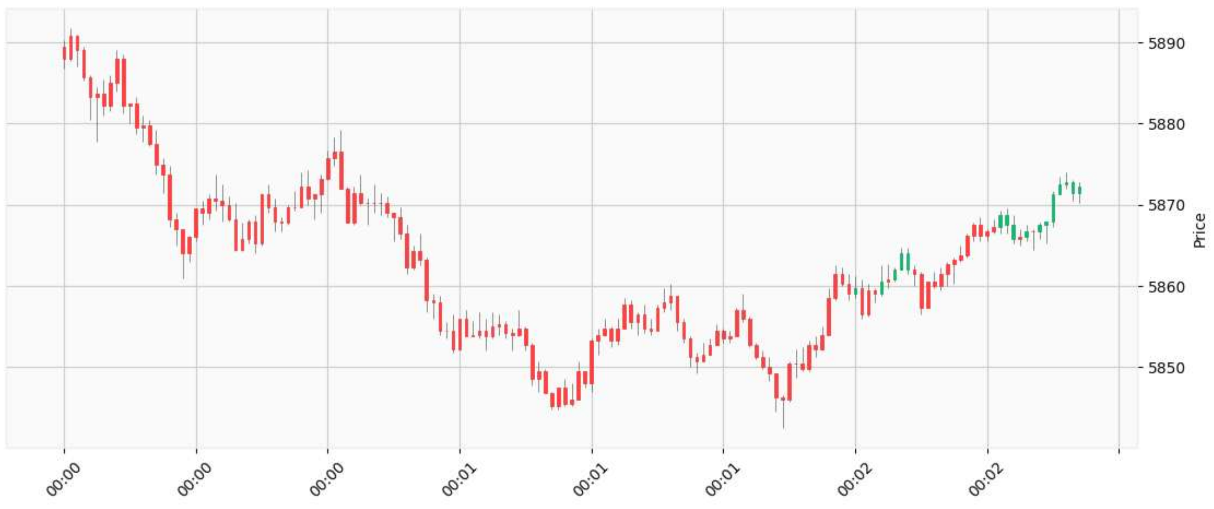


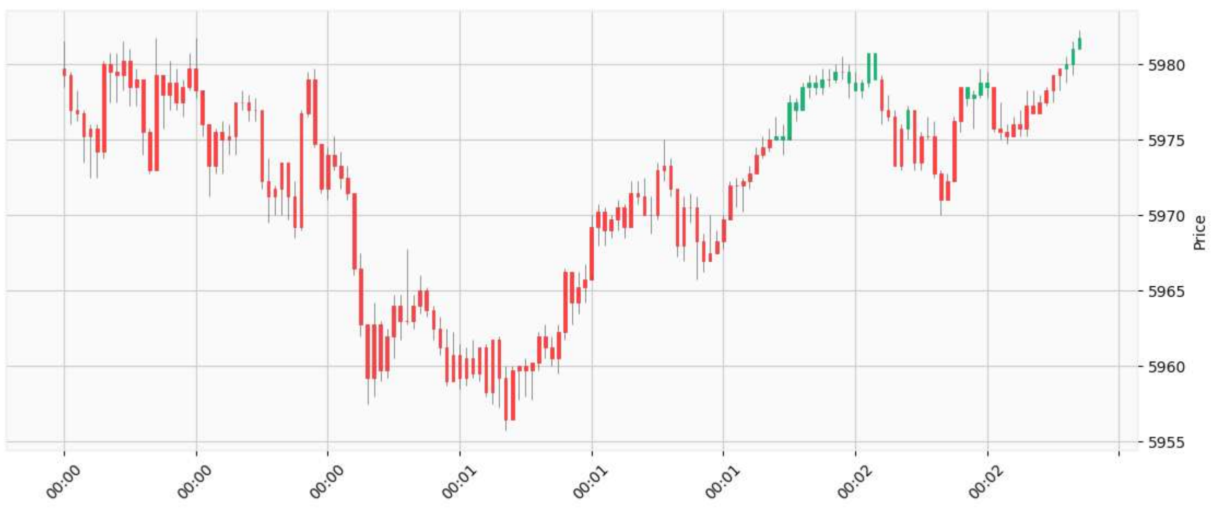
5960
5950
5940
5930
5920
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02
00:02

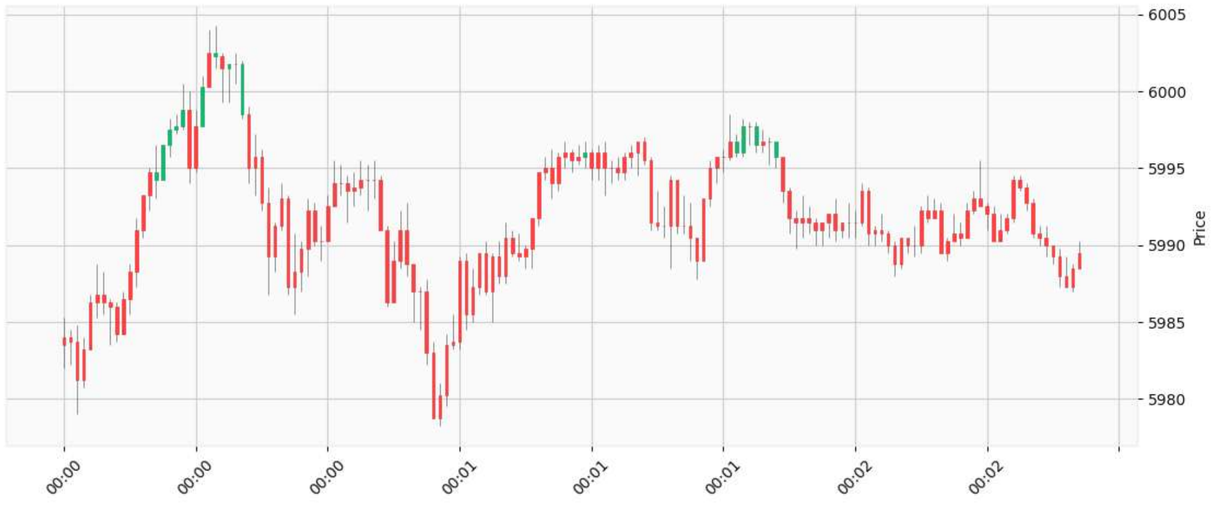


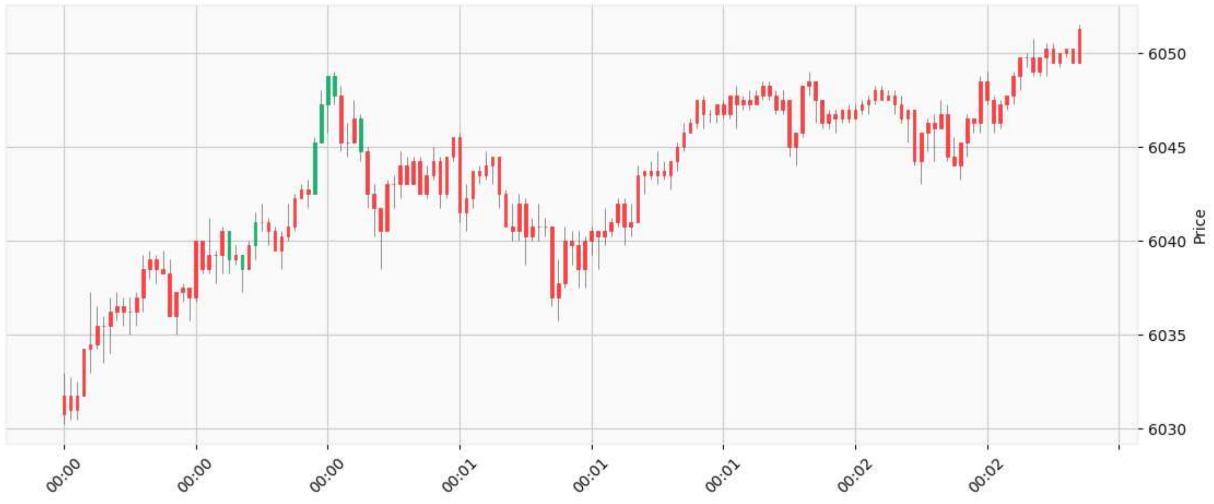
5890
5880
5870
5860
5850
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02
00:02

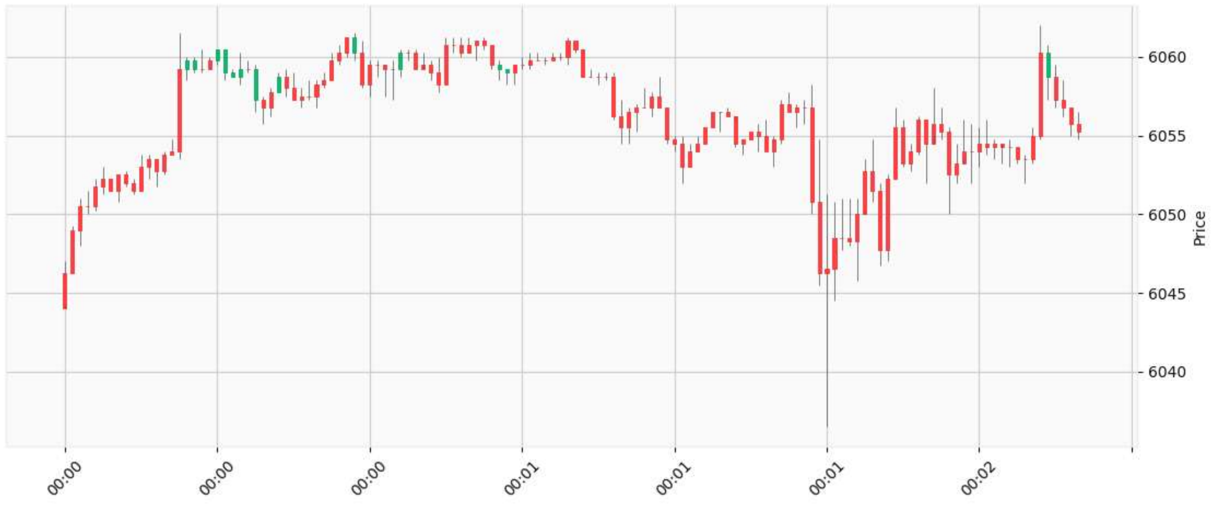

5860
5850
5840
5830
5820
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02
00:02


5890
5880
5870
5860
5850
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02
00:02


5980
5975
5970
5965
5960
5955
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02
00:02


6005
6000
5995
5990
5985
5980
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02
00:02


6050
6045
6040
6035
6030
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02
00:02


6060
6055
6050
6045
6040
Price
00:00
00:00
00:00
00:01
00:01
00:01
00:02