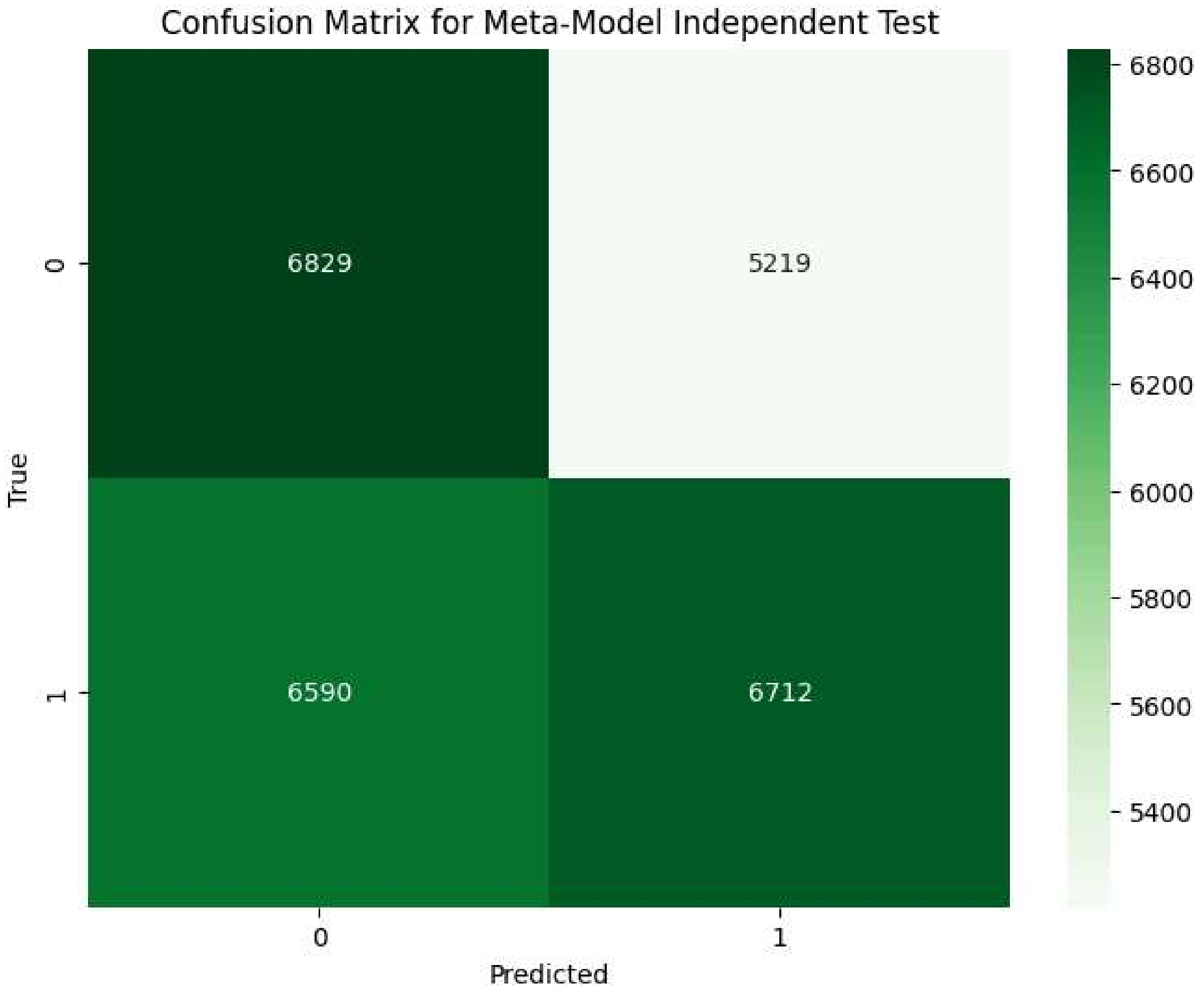
```
feature_subsets, scaler =\
_Data_Processing.preprocess_data(
        file_name   =           'spx_full.csv'
        ,indp_size  =           0.01
        ,test_size  =           0.01
        ,shfl_splt      =               False
        ,t_start    =           645
        ,t_end          =               800
        ,mod_type       =               'Area_Classification'
        ,target_t       =               45
        ,num_class      =               2
        ,split_val      =               5
        ,verbose        =               1
        ,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()
```

```
Trying to load CSV file into DataFrame...
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
Trying to drop unused targets...index location of "ToD" feature: (5, np.int16(1439))
Success.
Trying to collect indices of wanted times...Success.
Trying to drop price features...Success...

# of Samples:   226576

# of Features:  436

Target:         tc_2a_45m

Trying to split DataFrame into X and y...<class 'numpy.ndarray'> <class 'numpy.ndarr
ay'> <class 'numpy.float64'>
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.
        201226 Samples Dropped.

Trying to split X and y into Train/Validation/Independent...Success.
Trying to clean up...Success.
X_train:        (24843, 436).
y_train:        (24843,).
X_val:          (253, 436).
y_val:          (253,).
X_ind:          (254, 436).
y_ind:          (254,).
Collecting garbage...Success.
Terminating.
793/793 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step
793/793 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step
793/793 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step
793/793 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step
793/793 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step
793/793 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step
793/793 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step
793/793 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step
(25350, 8)
```

```
793/793 ━━━━━━━━━━━━━━━━━━━━ 1s 930us/step
0.5341617357001972
```



In [30]:

```
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         =               'None'
        ,cstm_scale     =               joblib.load('scaler/tmp.joblib')
        ,frmt_lstm      =               lstm_format
        ,time_steps =           5
        ,keep_price =           False
        ,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
index location of "ToD" feature: (5, np.int16(1439))
<class 'numpy.ndarray'> <class 'numpy.ndarray'> <class 'numpy.float64'>
        201226 Samples Dropped.
```

In [31]:

```
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)
```

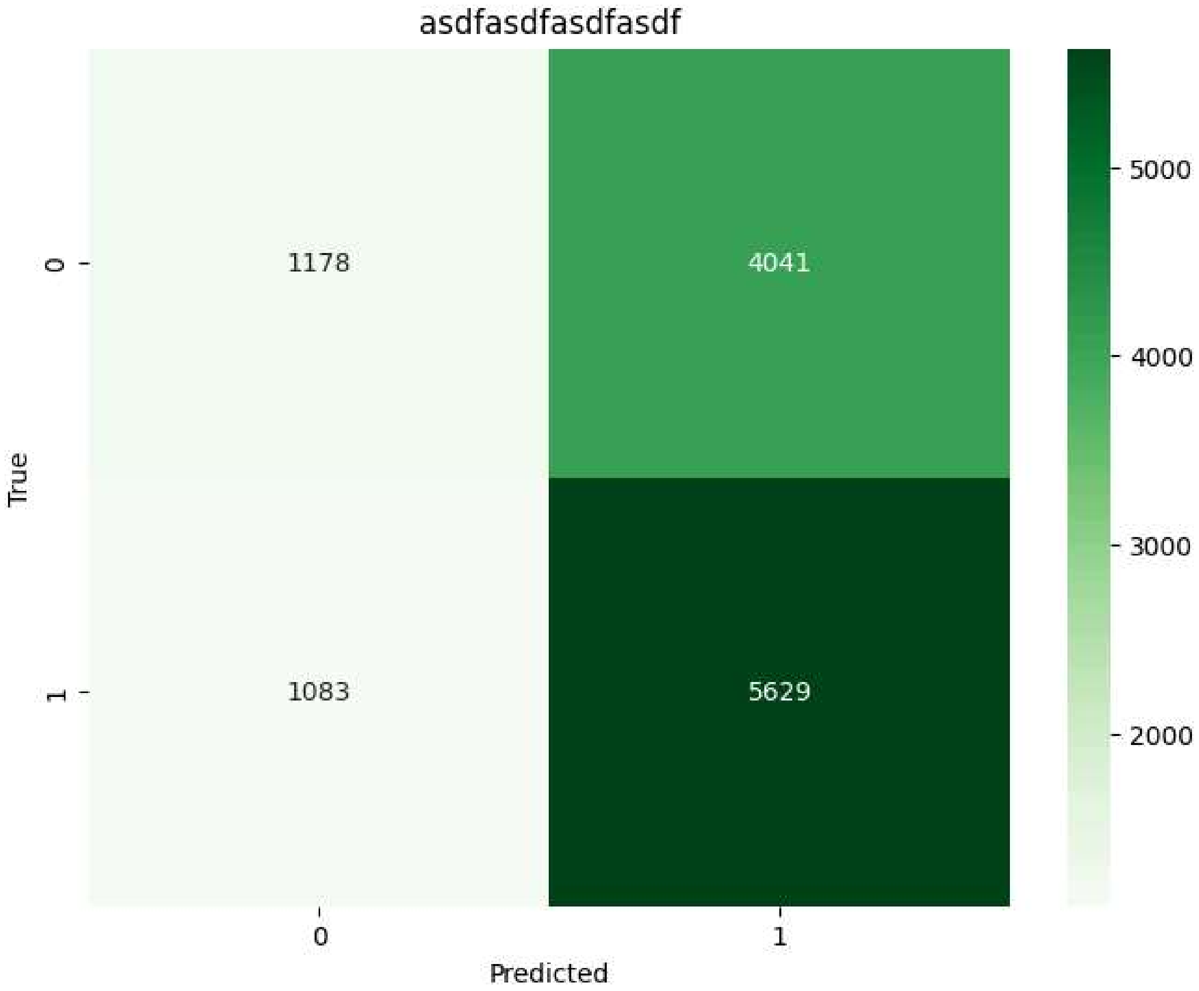
In [32]:

```
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
df = df.drop(df[df['score']%2==0].index).reset_index(drop=True)

#pd.set_option('display.max_rows',None)
co = df.corr()['target']
print(co.sort_values())
```

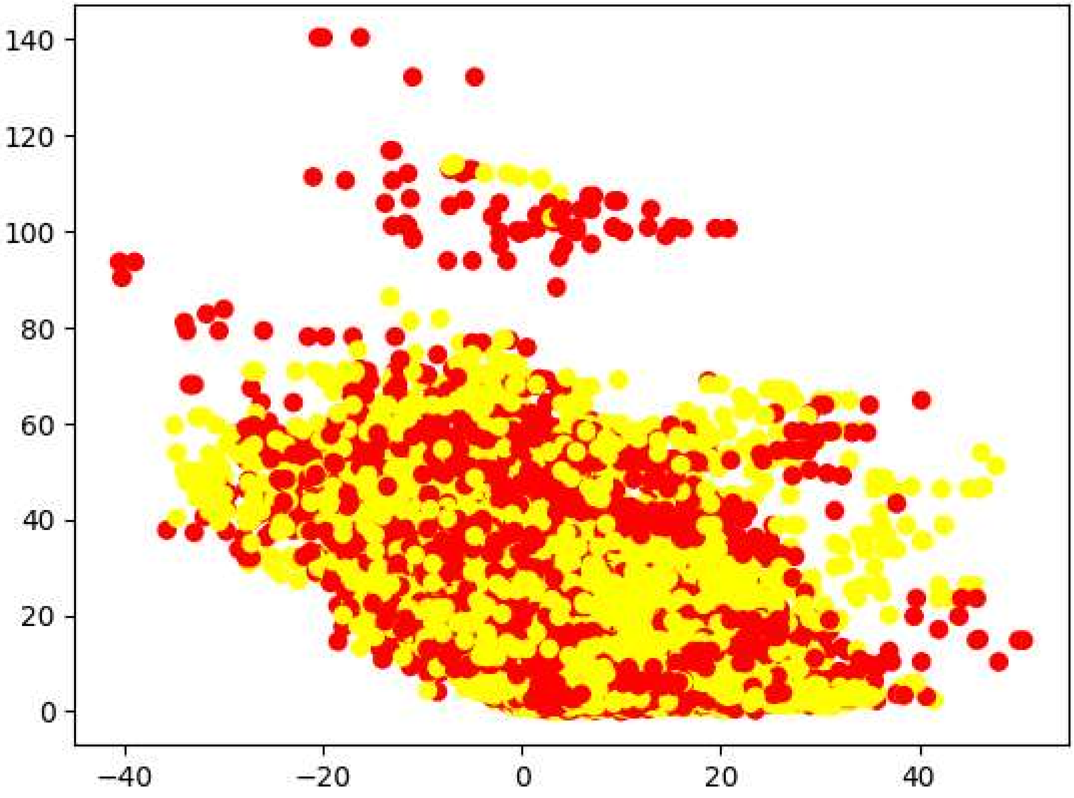
```
394      -0.072224
395      -0.063864
396      -0.056401
3        -0.047741
382      -0.046398
            ...
51        0.077344
52        0.078332
target    1.000000
score     1.000000
mpred          NaN
Name: target, Length: 439, dtype: float64
```

In [33]:

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

df_pair = pd.DataFrame()
#df_pair['ft'] = df[518].values
df_pair['ft2'] = df[394].values
df_pair['ft3'] = df[55].values
#df_pair['model_prediction'] = df['score']

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 [34]:
```
import matplotlib.pyplot as plt

plt.scatter(df_pair['ft3'], df_pair['ft2'],c=df['target'],cmap='autumn')
plt.show()
```



In [35]:

```
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         =               'None'
        ,cstm_scale     =               joblib.load('scaler/tmp.joblib')
        ,frmt_lstm      =               lstm_format
        ,time_steps =           5
        ,keep_price =           False
        ,indices        =               0
)
```

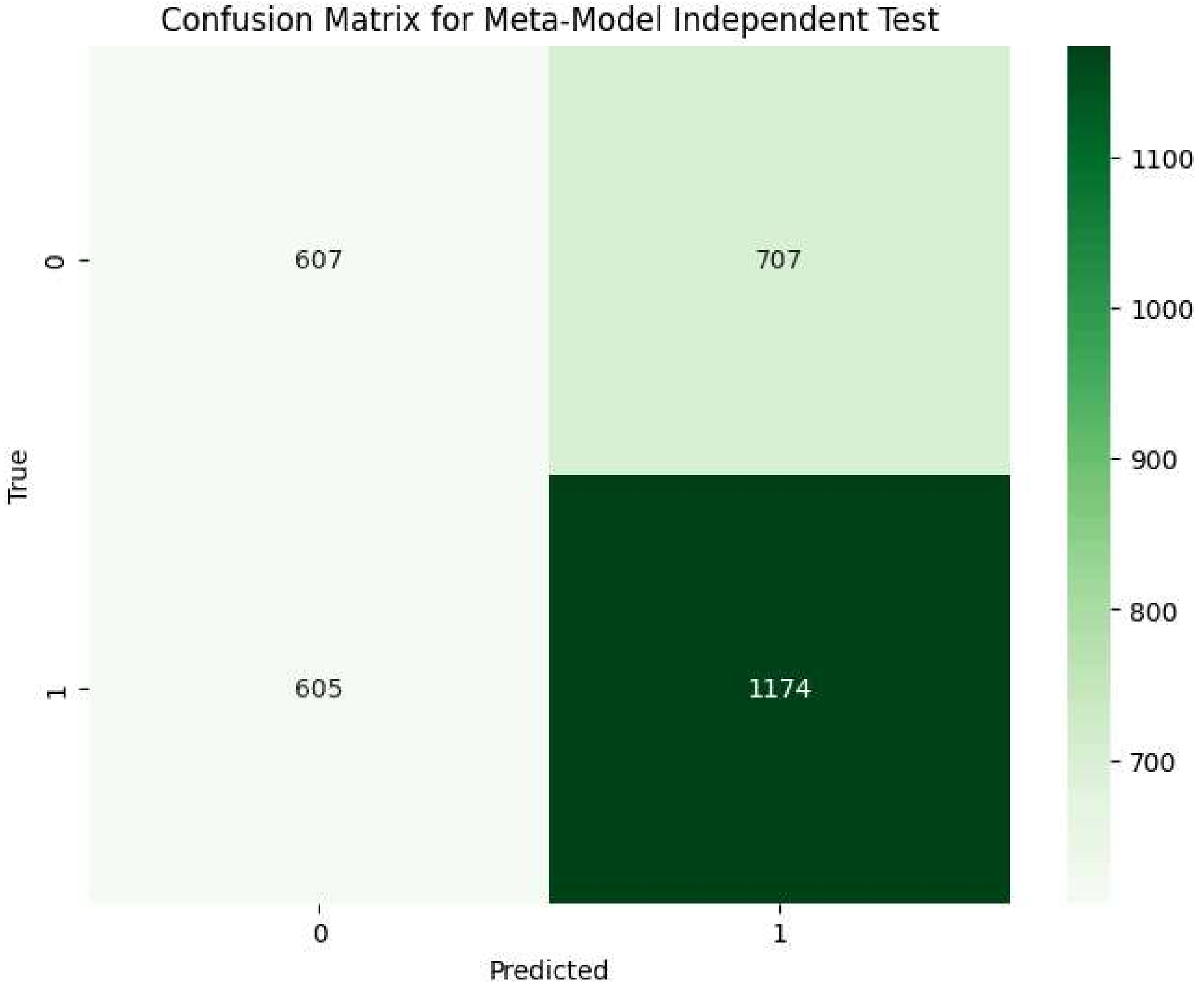
```
loaded chunk 1 of size: 125400164 -> 64600164
loaded chunk 2 of size: 18408884 -> 9395364
concat chunks
concatted chunks
Success.
Size of dataset:        74083444
index location of "ToD" feature: (5, np.int16(1439))
<class 'numpy.ndarray'> <class 'numpy.ndarray'> <class 'numpy.float64'>
        25577 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
index location of "ToD" feature: (5, np.int16(1439))
<class 'numpy.ndarray'> <class 'numpy.ndarray'> <class 'numpy.float64'>
        25577 Samples Dropped.
```

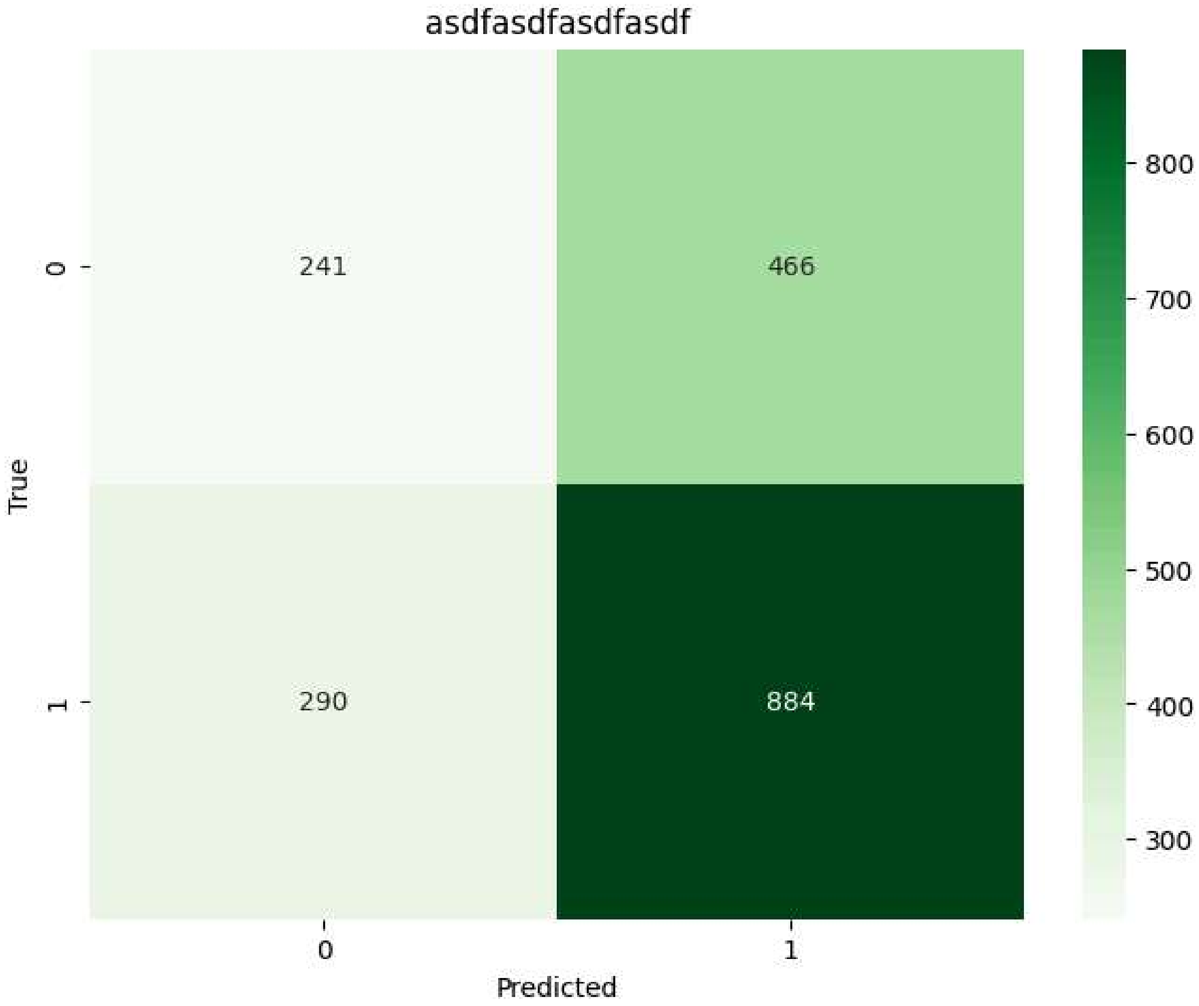
In [36]:

```
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
df = df.drop(df[df['score']%2==0].index).reset_index(drop=True)

df_pair = pd.DataFrame()
#df_pair['ft'] = df[518].values
df_pair['ft2'] = df[394].values
df_pair['ft3'] = df[55].values
#df_pair['model_prediction'] = df['mpred']

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

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()
```

```
97/97 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step
97/97 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
97/97 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
97/97 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
97/97 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
97/97 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step
97/97 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
97/97 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
(3093, 8)
97/97 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
0.5758163595215001
```


Confusion Matrix for Meta-Model Independent Test



In [ ]:

```
import mplfinance as mpf
import pandas as pd
import numpy as np


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

num_candles = 155

for section in range(int(len(X_test[:,0])/num_candles)):
        section*=num_candles
        section_end = section+num_candles
        X_thold = X_test[section:section_end,:]

        #custom coloring
        color_map = {
                0: 'white',
                1: 'red',
                2: 'white',
                3: 'green'
        }
        colors = [color_map[condition] for condition in cm_vals[section:section_end
        mc = mpf.make_marketcolors(up='g',down='r')
        custom_style = mpf.make_mpf_style(marketcolors=mc)#, gridcolor='lightgray')
```