In [1]:
```
# !pip install plotly
import pandas as pd
# Join postal codes to
import pandas_ta as ta
import numpy as np
import seaborn as sns
import datetime
from datetime import timedelta
import math
import requests
import plotly as py
import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objects as go
import matplotlib.pyplot as plt

import sklearn as sk
import sklearn.preprocessing
from sklearn import metrics

from sklearn.preprocessing import   (StandardScaler, MinMaxScaler, LabelBinarizer)
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split

# import chart_studio.tools as tls
```

In [2]:
```
#manipulatable variables
numDaysBack = str(36545) #for daily you can go back multiple years worth, for daily you can only go back 90 days
myInterval = 'daily' # options are daily or hourly
theCoins = 'ethereum' #can add more than one coin if you like
window_length = 14
mycom = 0.4
lower_macd_ema = 12
upper_macd_ema = 26
trigger_macd_ema = 9

def df_builder_clean(days, interval, coins):
    #manipulatable variables
    numDaysBack = days #for daily you can go back multiple years worth, for daily you can only go back 90 days
    myInterval = interval # options are daily or hourly
    theCoins = coins

    #builds initial dataframe with ethereum as first market but just to log the dates we are working with
    geckoReq = 'https://api.coingecko.com/api/v3/coins/ethereum/market_chart?vs_currency=usd&days='+numDaysBack+'&interval='+myInterval
    r = requests.get(geckoReq).json()
    ts = r['prices'][0][0]
    tx = ts/1000
    HistPricesList = []
    for i in range(len(r['prices'])):
        currentUnix = r['prices'][i][0]
        price = r['prices'][i][1]
        currentUnix = currentUnix/1000
#         currentTS = datetime.datetime.fromtimestamp(currentUnix).strftime("%d-%m-%Y %H:%M:%S") #adding dd-mm-yyyy hh:mm:ss
        currentTS = datetime.datetime.fromtimestamp(currentUnix).strftime("%m-%d-%Y") #just the date mm-dd-yyyy
        HistPricesList.append([currentTS, price])
    global df
    df = pd.DataFrame(HistPricesList, columns = ['date'])

    #looping through each coin and adding in all data points and input variables
    for coin in theCoins:
        price_data(coin)
#         add_ewm(coin, mycom)
#         add_rsi(coin, window_length)
#         add_macd(coin, lower_macd_ema, upper_macd_ema, trigger_macd_ema)
        print('just added: ', coin)
    df['date'] = pd.to_datetime(df['date'])
    df = df.set_index('date')
#     print('c')
#     display(c)
    return df


def price_data(coin):
    global df
    # geckoReq = 'https://api.coingecko.com/api/v3/coins/ethereum/market_chart?vs_currency=usd&days=60&interval=daily'
    geckoReq = 'https://api.coingecko.com/api/v3/coins/'+coin+'/market_chart?vs_currency=usd&days='+numDaysBack+'&interval='+myInterval

    r = requests.get(geckoReq).json()
    ts = r['prices'][0][0]
    # print(ts)
    tx = ts/1000
    print(datetime.datetime.fromtimestamp(ts).strftime("%m-%d-%Y"))
    # print('prices length',len(r['prices']))
    HistPricesList = []
    for i in range(len(r['prices'])):
        currentUnix = r['prices'][i][0]
        price = r['prices'][i][1]
        volume = r['total_volumes'][i][1]
        currentUnix = currentUnix/1000
        currentTS = datetime.datetime.fromtimestamp(currentUnix).strftime("%m-%d-%Y")
#         print('price:', price, 'TS: ', currentTS)
        HistPricesList.append([currentTS, price, volume])
#         currentUnix = currentTS
#         print(HistPricesList)
    dfCoin = pd.DataFrame(HistPricesList, columns = ['date', coin, coin+'_volume'])
#     print('dfCoin')
#     display(dfCoin)
    df = pd.merge(df, dfCoin[coin], left_on = df['date'], right_on=dfCoin['date']).drop(['key_0'], axis = 1)
    df = pd.merge(df, dfCoin[coin+'_volume'], left_on = df['date'], right_on=dfCoin['date']).drop(['key_0'], axis = 1)

def add_ewm(coin, mycom):
    df[coin+'_ewm'] = df[coin].ewm(com=mycom).mean()

def add_rsi(coin, window_length):
    global df
    df['diff'] = df[coin].diff(1)
    df['gain'] = df['diff'].clip(lower=0).round(2)
    df['loss'] = df['diff'].clip(upper=0).abs().round(2)

    # Get initial Averages
    df['avg_gain'] = df['gain'].rolling(window=window_length, min_periods=window_length).mean()[:window_length+1]
    df['avg_loss'] = df['loss'].rolling(window=window_length, min_periods=window_length).mean()[:window_length+1]

    # Get WMS averages
    # Average Gains
    for i, row in enumerate(df['avg_gain'].iloc[window_length+1:]):
        df['avg_gain'].iloc[i + window_length + 1] =\
            (df['avg_gain'].iloc[i + window_length] *
                (window_length - 1) +
                df['gain'].iloc[i + window_length + 1])\
                / window_length
    # Average Losses
    for i, row in enumerate(df['avg_loss'].iloc[window_length+1:]):
        df['avg_loss'].iloc[i + window_length + 1] =\
            (df['avg_loss'].iloc[i + window_length] *
                (window_length - 1) +
                df['loss'].iloc[i + window_length + 1])\
                / window_length

    df['rs'] = df['avg_gain'] / df['avg_loss']

    df['rsi'] = 100 - (100 / (1.0 + df['rs']))
    df = pd.DataFrame(df)
    df = df.drop(['gain', 'loss', 'avg_loss', 'avg_gain', 'rs'], axis = 1)

    #renaming diff and rsi columns
    dict = {'diff': coin+'_diff',
            'rsi': coin+'_rsi'}
    df.rename(columns=dict,
              inplace=True)

def add_macd(coin, lower_macd_ema, upper_macd_ema, trigger_macd_ema):
    global df
    #get ema for lower
    k = df[coin].ewm(span=lower_macd_ema, adjust = False, min_periods = lower_macd_ema).mean()

    #get ema for upper
    d = df[coin].ewm(span=upper_macd_ema, adjust=False, min_periods=upper_macd_ema).mean()

    macd = k-d

    # Get the 9-Day EMA of the MACD for the Trigger line
    macd_s = macd.ewm(span=trigger_macd_ema, adjust=False, min_periods=trigger_macd_ema).mean()
    # Calculate the difference between the MACD - Trigger for the Convergence/Divergence value
    macd_h = macd - macd_s
    # Add all of our new values for the MACD to the dataframe
    df['macd'] = df.index.map(macd)
    df['macd_h'] = df.index.map(macd_h)
    df['macd_s'] = df.index.map(macd_s)


myDf = pd.DataFrame(df_builder_clean(numDaysBack, myInterval, theCoins))
display(myDf)
```

```
08-11-2017
just added:  ethereum
```

date	ethereum	ethereum_volume
2017-08-17	296.114635	5.904704e+08
2017-08-18	296.622090	5.537022e+08
2017-08-19	295.171577	3.428236e+08
2017-08-20	322.201220	1.743916e+09
2017-08-21	312.174471	8.983443e+08
...	...	...
2022-08-12	1959.330925	1.584926e+10
2022-08-13	1982.411828	1.481675e+10
2022-08-14	1936.701164	1.217227e+10
2022-08-15	1908.277642	1.831087e+10
2022-08-16	1883.678469	1.465477e+10

1826 rows × 2 columns

In [3]:
```
myDf = myDf.dropna()
# display(myDf)

# display(my_data.tail(10))
# myDf.drop(myDf.tail(10).index,
#           inplace = True)
display(myDf)
myDf.corr()
```

date	ethereum	ethereum_volume
2017-08-17	296.114635	5.904704e+08
2017-08-18	296.622090	5.537022e+08
2017-08-19	295.171577	3.428236e+08
2017-08-20	322.201220	1.743916e+09
2017-08-21	312.174471	8.983443e+08
...	...	...
2022-08-12	1959.330925	1.584926e+10
2022-08-13	1982.411828	1.481675e+10
2022-08-14	1936.701164	1.217227e+10
2022-08-15	1908.277642	1.831087e+10
2022-08-16	1883.678469	1.465477e+10

1826 rows × 2 columns

Out[3]:

	ethereum	ethereum_volume
ethereum	1.000000	0.533854
ethereum_volume	0.533854	1.000000

In [4]:
```
import statsmodels.api
import statsmodels as sm
x = pd.DataFrame(myDf['ethereum'])
res = sm.tsa.seasonal_decompose(x,model='additive')
fig = res.plot()
fig.set_size_inches((16, 9))
fig.tight_layout()
plt.show()
```


In [5]:
```
seq=list(myDf['ethereum'])
# display(seq)
scaler = StandardScaler()
window_size = 1
slidingWindow = []
for i in range(len(seq) - window_size + 1):
    slidingWindow.append(seq[i : i + window_size])

# print(sliding_window)

sliding_df = pd.DataFrame(sliding_window)
sliding_df=sliding_df.rename(columns={0:'3DaysPrior', 1:'2DaysPrior',2:'1DayPrior',3:'Target'})

# sliding_df = sliding_df.drop(columns=[1,2,3,4,5])
temp_df = myDf.tail(1823)
temp_df=temp_df.reset_index()
display(myDf.tail(1823))
sliding_df['date'] = temp_df['date']
sliding_df=sliding_df.set_index('date')
display(sliding_df)
```

date	ethereum	ethereum_volume
2017-08-20	322.201220	1.743916e+09
2017-08-21	312.174471	8.983443e+08
2017-08-22	316.788920	4.664746e+08
2017-08-23	321.785298	3.977527e+08
2017-08-24	329.025281	4.519507e+08
...	...	...
2022-08-12	1959.330925	1.584926e+10
2022-08-13	1982.411828	1.481675e+10
2022-08-14	1936.701164	1.217227e+10
2022-08-15	1908.277642	1.831087e+10
2022-08-16	1883.678469	1.465477e+10

1823 rows × 2 columns

date	3DaysPrior	2DaysPrior	1DayPrior	Target
2017-08-20	296.114635	296.622090	295.171577	322.201220
2017-08-21	296.622090	295.171577	322.201220	312.174471
2017-08-22	295.171577	322.201220	312.174471	316.788920
2017-08-23	322.201220	312.174471	316.788920	321.785298
2017-08-24	312.174471	316.788920	321.785298	329.025281
...	...	...	...	...
2022-08-12	1698.964139	1852.878055	1881.437405	1959.330925
2022-08-13	1852.878055	1881.437405	1959.330925	1982.411828
2022-08-14	1881.437405	1959.330925	1982.411828	1936.701164
2022-08-15	1959.330925	1982.411828	1936.701164	1908.277642
2022-08-16	1982.411828	1936.701164	1908.277642	1883.678469

1823 rows × 4 columns

In [38]:
```
sliding_df_corr = sliding_df.corr()
sns.set(rc = {'figure.figsize':(15,8)})

sns.heatmap(sliding_df_corr, cmap='Blues')
```

Out[38]:
```
<AxesSubplot:>
```