```python
#loading the basic necessary imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

#loading the data files
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')

#checking for correlation between the features and the target
train.corr()
```

```
                        timestamp      open      high       low
close  \
timestamp                1.000000  0.534376  0.534272  0.534501
0.534386
open                     0.534376  1.000000  0.999994  0.999994
0.999992
high                     0.534272  0.999994  1.000000  0.999989
0.999995
low                      0.534501  0.999994  0.999989  1.000000
0.999995
close                    0.534386  0.999992  0.999995  0.999995
1.000000
volume                   0.237101  0.205516  0.206893  0.203911
0.205389
quote_asset_volume       0.272706  0.427065  0.428565  0.425314
0.426929
number_of_trades         0.286741  0.390715  0.392173  0.389095
0.390649
taker_buy_base_volume    0.231311  0.203129  0.204721  0.201869
0.203450
taker_buy_quote_volume   0.265828  0.419227  0.420968  0.417871
0.419604
target                   0.005468 -0.004030 -0.003995 -0.004087 -
0.004100

                          volume  quote_asset_volume  number_of_trades
\
timestamp               0.237101            0.272706          0.286741

open                    0.205516            0.427065          0.390715

high                    0.206893            0.428565          0.392173

low                     0.203911            0.425314          0.389095
```

```
close                   0.205389             0.426929             0.390649
volume                  1.000000             0.848345             0.795536
quote_asset_volume      0.848345             1.000000             0.895895
number_of_trades        0.795536             0.895895             1.000000
taker_buy_base_volume   0.963362             0.818933             0.769869
taker_buy_quote_volume  0.814487             0.963957             0.866840
target                  0.015103             0.012075             0.014019

                        taker_buy_base_volume  taker_buy_quote_volume  \
timestamp                            0.231311                0.265828
open                                 0.203129                0.419227
high                                 0.204721                0.420968
low                                  0.201869                0.417871
close                                0.203450                0.419604
volume                               0.963362                0.814487
quote_asset_volume                   0.818933                0.963957
number_of_trades                     0.769869                0.866840
taker_buy_base_volume                1.000000                0.848061
taker_buy_quote_volume               0.848061                1.000000
target                               0.013395                0.010717

                          target
timestamp               0.005468
open                   -0.004030
high                   -0.003995
low                    -0.004087
close                  -0.004100
volume                  0.015103
quote_asset_volume      0.012075
number_of_trades        0.014019
taker_buy_base_volume   0.013395
```

```
taker_buy_quote_volume  0.010717
target                  1.000000

#creating a new calculated column
train['price_range'] = train['high'] - train['low']/train['open']

#droping these columns cause of high correlation with other features
train.drop(['high','low','open','taker_buy_base_volume'],axis = 1,
inplace = True)

train.head()

    timestamp    close   volume  quote_asset_volume  number_of_trades  \
0  1525471260  0.90130   134.98          121.646459               4.0

1  1525471320  0.90195  1070.54          965.505313              12.0

2  1525471380  0.90139  2293.06         2066.963991               5.0

3  1525471440  0.90139  6850.59         6175.000909              19.0

4  1525471500  0.90130   832.30          750.222624               3.0


   taker_buy_quote_volume  target  price_range
0              112.723589     1.0    -0.098700
1              793.612703     0.0    -0.098050
2                0.000000     0.0    -0.098589
3             1610.149485     0.0    -0.098589
4              707.428900     0.0    -0.098510

#converting the format for the feature
train['timestamp'] = pd.to_datetime(train['timestamp'])

train.head()

                      timestamp    close   volume  quote_asset_volume  \
0 1970-01-01 00:00:01.525471260  0.90130   134.98          121.646459

1 1970-01-01 00:00:01.525471320  0.90195  1070.54          965.505313

2 1970-01-01 00:00:01.525471380  0.90139  2293.06         2066.963991

3 1970-01-01 00:00:01.525471440  0.90139  6850.59         6175.000909

4 1970-01-01 00:00:01.525471500  0.90130   832.30          750.222624


   number_of_trades  taker_buy_quote_volume  target  price_range
0               4.0              112.723589     1.0    -0.098700
```

```
1               12.0                 793.612703     0.0    -0.098050
2                5.0                   0.000000     0.0    -0.098589
3               19.0                1610.149485     0.0    -0.098589
4                3.0                 707.428900     0.0    -0.098510
```

```
train.tail()
```

```
                            timestamp   close    volume
quote_asset_volume  \
2122433 1970-01-01 00:00:01.652817240  0.4304  136274.0
58630.1628
2122434 1970-01-01 00:00:01.652817300  0.4305  104478.0
44967.8376
2122435 1970-01-01 00:00:01.652817360  0.4309  212396.0
91526.9872
2122436 1970-01-01 00:00:01.652817420  0.4306  131047.0
56443.0038
2122437 1970-01-01 00:00:01.652817480  0.4301  101150.0
43542.2629

         number_of_trades  taker_buy_quote_volume  target  price_range

2122433             144.0              23325.9277     1.0    -0.567774

2122434              99.0              22484.0304     1.0    -0.569300

2122435             177.0              46673.0616     0.0    -0.568800

2122436             107.0              14097.1489     0.0    -0.567276

2122437             105.0              19851.7237     1.0    -0.568039
```

```
#making sure that the dataset is not imbalanced
train['target'].value_counts()
```

```
0.0    1112614
1.0    1009824
Name: target, dtype: int64
```

```
#creating moving average windows
for window in [5,10,20,30]:
  train[f'close_ma_{window}'] = train['close'].rolling(window =
window).mean()
  train[f'close_std_{window}'] = train['close'].rolling(window =
window).std()

#creating lags in order to capture recent trends
for lag in range(1,6):
  train[f'close_lag_{lag}'] = train['close'].shift(lag)
  train[f'volume_lag_{lag}'] = train['volume'].shift(lag)
```

```
  train[f'quote_volume_lag_{lag}'] =
train['quote_asset_volume'].shift(lag)

train.head()

                      timestamp    close   volume  quote_asset_volume
\
0 1970-01-01 00:00:01.525471260  0.90130   134.98          121.646459

1 1970-01-01 00:00:01.525471320  0.90195  1070.54          965.505313

2 1970-01-01 00:00:01.525471380  0.90139  2293.06         2066.963991

3 1970-01-01 00:00:01.525471440  0.90139  6850.59         6175.000909

4 1970-01-01 00:00:01.525471500  0.90130   832.30          750.222624


   number_of_trades  taker_buy_quote_volume  target  price_range
close_ma_5  \
0               4.0              112.723589     1.0    -0.098700
NaN
1              12.0              793.612703     0.0    -0.098050
NaN
2               5.0                0.000000     0.0    -0.098589
NaN
3              19.0             1610.149485     0.0    -0.098589
NaN
4               3.0              707.428900     0.0    -0.098510
0.901466

   close_std_5  ...  quote_volume_lag_2  close_lag_3  volume_lag_3  \
0          NaN  ...                 NaN          NaN           NaN
1          NaN  ...                 NaN          NaN           NaN
2          NaN  ...          121.646459          NaN           NaN
3          NaN  ...          965.505313      0.90130        134.98
4     0.000274  ...         2066.963991      0.90195       1070.54

   quote_volume_lag_3  close_lag_4  volume_lag_4
quote_volume_lag_4  \
0                 NaN          NaN           NaN                 NaN

1                 NaN          NaN           NaN                 NaN

2                 NaN          NaN           NaN                 NaN

3          121.646459          NaN           NaN                 NaN

4          965.505313       0.9013        134.98          121.646459
```

```
   close_lag_5  volume_lag_5  quote_volume_lag_5
0          NaN           NaN                 NaN
1          NaN           NaN                 NaN
2          NaN           NaN                 NaN
3          NaN           NaN                 NaN
4          NaN           NaN                 NaN

[5 rows x 31 columns]
```

```
#making sure there are no nulls
train.dropna(inplace = True)
```

```
train.head()
```

```
                       timestamp    close    volume
quote_asset_volume  \
29 1970-01-01 00:00:01.525473000  0.89571    446.13
400.511026
30 1970-01-01 00:00:01.525473060  0.89787  14254.75
12796.739759
31 1970-01-01 00:00:01.525473120  0.89572   1318.45
1183.339181
32 1970-01-01 00:00:01.525473180  0.89570   1604.75
1437.376947
33 1970-01-01 00:00:01.525473240  0.89445      6.92
6.197552

    number_of_trades  taker_buy_quote_volume  target  price_range
close_ma_5  \
29               7.0              393.479663     1.0    -0.102135
0.896284
30              18.0             1261.653247     0.0    -0.102130
0.896284
31               5.0                0.000000     0.0    -0.099735
0.895854
32               9.0                0.000000     0.0    -0.104258
0.896140
33               3.0                5.795050     1.0    -0.102947
0.895890

    close_std_5  ...  quote_volume_lag_2  close_lag_3  volume_lag_3  \
29     0.001562  ...        13430.793297      0.89787      10914.10
30     0.001562  ...        10638.856804      0.89427      15007.25
31     0.001288  ...          400.511026      0.89570      11877.72
32     0.000967  ...        12796.739759      0.89571        446.13
33     0.001234  ...         1183.339181      0.89787      14254.75

    quote_volume_lag_3  close_lag_4  volume_lag_4  quote_volume_lag_4
\
29         9799.465432      0.89787        941.71          845.490906
```

```
30        13430.793297      0.89787      10914.10          9799.465432
31        10638.856804      0.89427      15007.25         13430.793297
32          400.511026      0.89570      11877.72         10638.856804
33        12796.739759      0.89571        446.13           400.511026

    close_lag_5  volume_lag_5  quote_volume_lag_5
29      0.89790        522.25          468.824943
30      0.89787        941.71          845.490906
31      0.89787      10914.10         9799.465432
32      0.89427      15007.25        13430.793297
33      0.89570      11877.72        10638.856804

[5 rows x 31 columns]
```

```
train.shape
```

```
(2122409, 31)
```

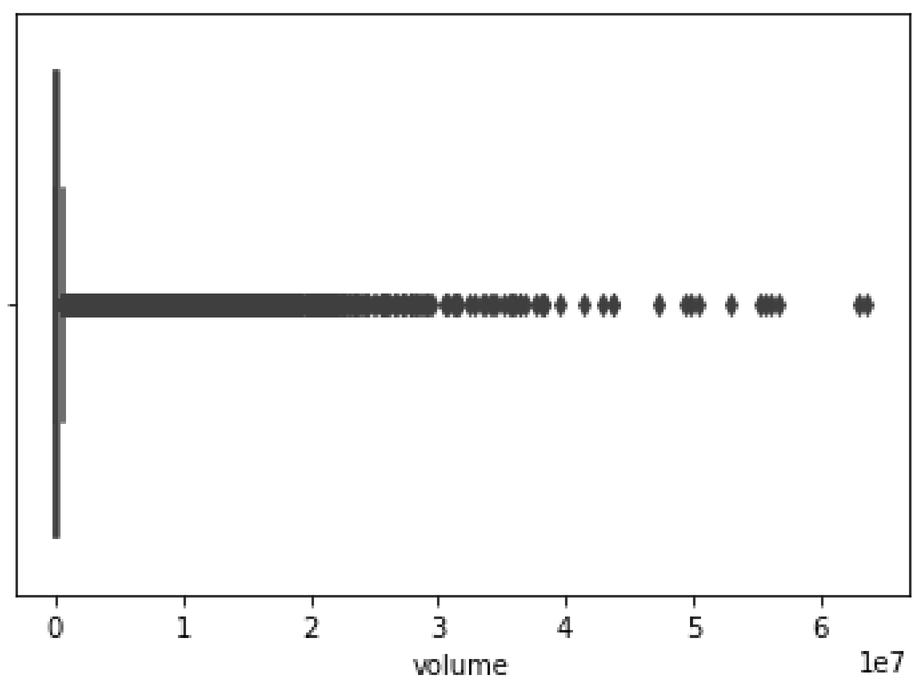
```
!pip install ta
```

```
Defaulting to user installation because normal site-packages is not
writeable
Requirement already satisfied: ta in ./.local/lib/python3.8/site-
packages (0.11.0)
Requirement already satisfied: numpy in ./.local/lib/python3.8/site-
packages (from ta) (1.22.4)
Requirement already satisfied: pandas in
/shared/centos7/anaconda3/2021.05/lib/python3.8/site-packages (from
ta) (1.2.4)
Requirement already satisfied: python-dateutil>=2.7.3 in
/shared/centos7/anaconda3/2021.05/lib/python3.8/site-packages (from
pandas->ta) (2.8.1)
Requirement already satisfied: pytz>=2017.3 in
/shared/centos7/anaconda3/2021.05/lib/python3.8/site-packages (from
pandas->ta) (2021.1)
Requirement already satisfied: six>=1.5 in
/shared/centos7/anaconda3/2021.05/lib/python3.8/site-packages (from
python-dateutil>=2.7.3->pandas->ta) (1.15.0)
```

```
#importing technical indicator libraries
from ta.momentum import RSIIndicator
from ta.trend import MACD, EMAIndicator
from ta.volatility import BollingerBands

#adding these technical indicators as columns for our dataset after
calculations
```

```python
rsi = RSIIndicator(close=train['close'], window=14)
train['rsi'] = rsi.rsi()

macd = MACD(close=train['close'], window_slow=26, window_fast=12,
window_sign=9)
train['macd'] = macd.macd()
train['macd_signal'] = macd.macd_signal()
train['macd_diff'] = macd.macd_diff()

bb = BollingerBands(close=train['close'], window=20, window_dev=2)
train['bb_mavg'] = bb.bollinger_mavg()
train['bb_high'] = bb.bollinger_hband()
train['bb_low'] = bb.bollinger_lband()
train['bb_width'] = train['bb_high'] - train['bb_low']

ema_10 = EMAIndicator(close=train['close'], window=10)
ema_30 = EMAIndicator(close=train['close'], window=30)
train['ema_10'] = ema_10.ema_indicator()
train['ema_30'] = ema_30.ema_indicator()
train['ema_diff'] = train['ema_10'] - train['ema_30']

train.fillna(method='ffill', inplace=True)


sns.boxplot(train['volume'])
```

```
<AxesSubplot:xlabel='volume'>
```



```
#importing scaling libraries
from sklearn.preprocessing import RobustScaler, MinMaxScaler,
QuantileTransformer, StandardScaler

#segragating the columns in order to use different scaling methods
based on the feature
price_cols = ['close', 'close_ma_5', 'close_std_5', 'close_lag_3',
'close_lag_4', 'close_lag_5']
volume_cols = ['volume', 'quote_asset_volume',
'taker_buy_quote_volume', 'volume_lag_2', 'quote_volume_lag_2',
'volume_lag_3', 'quote_volume_lag_3', 'volume_lag_4',
'quote_volume_lag_4', 'volume_lag_5', 'quote_volume_lag_5']
count_cols = ['number_of_trades']
derived_cols = ['price_range']

scaler_price = RobustScaler()
train[price_cols] = scaler_price.fit_transform(train[price_cols])

scaler_volume = QuantileTransformer(output_distribution='uniform')
train[volume_cols] = scaler_volume.fit_transform(train[volume_cols])

scaler = RobustScaler()

# Scale the technical indicators
indicator_cols = ['rsi', 'macd', 'macd_signal', 'macd_diff',
                  'bb_mavg', 'bb_high', 'bb_low', 'bb_width',
                  'ema_10', 'ema_30', 'ema_diff']

# Fit and transform the indicators
train[indicator_cols] = scaler.fit_transform(train[indicator_cols])

scaler_count = RobustScaler()
train[count_cols] = scaler_count.fit_transform(train[count_cols])

scaler_derived = StandardScaler()
train[derived_cols] =
scaler_derived.fit_transform(train[derived_cols])

train.corr()
```

	close	volume	quote_asset_volume \
close	1.000000	0.426095	0.621662
volume	0.426095	1.000000	0.961909
quote_asset_volume	0.621662	0.961909	1.000000
number_of_trades	0.390660	0.455200	0.466815
taker_buy_quote_volume	0.607932	0.914741	0.955486
target	-0.004094	0.023198	0.018436
price_range	0.999973	0.428296	0.623475
close_ma_5	0.999991	0.426111	0.621678
close_std_5	0.518567	0.440379	0.484346
close_ma_10	0.999979	0.426118	0.621683

```
close_std_10            0.532959  0.447505          0.494101
close_ma_20             0.999956  0.426122          0.621686
close_std_20            0.541721  0.451679          0.500157
close_ma_30             0.999934  0.426122          0.621685
close_std_30            0.545720  0.452857          0.502359
close_lag_1             0.999992  0.426105          0.621671
volume_lag_1            0.205432  0.427237          0.417133
quote_volume_lag_1      0.426977  0.407578          0.430979
close_lag_2             0.999985  0.426112          0.621677
volume_lag_2            0.426332  0.773082          0.771883
quote_volume_lag_2      0.622662  0.772327          0.837376
close_lag_3             0.999977  0.426116          0.621679
volume_lag_3            0.426086  0.761633          0.762371
quote_volume_lag_3      0.621671  0.762631          0.829013
close_lag_4             0.999970  0.426116          0.621679
volume_lag_4            0.426404  0.754912          0.756749
quote_volume_lag_4      0.621727  0.756752          0.824015
close_lag_5             0.999963  0.426117          0.621679
volume_lag_5            0.426142  0.750243          0.752897
quote_volume_lag_5      0.622046  0.753286          0.821145
rsi                     0.006706  0.001523          0.001759
macd                    0.010645  0.000660          0.001724
macd_signal             0.011124  0.001540          0.002618
macd_diff               0.000672 -0.002535         -0.002364
bb_mavg                 0.999956  0.426148          0.621712
bb_high                 0.999903  0.428730          0.623606
bb_low                  0.999911  0.423489          0.619731
bb_width                0.541722  0.451688          0.500165
ema_10                  0.999985  0.426129          0.621696
ema_30                  0.999952  0.426166          0.621734
ema_diff                0.010903  0.000707          0.001791

                        number_of_trades  taker_buy_quote_volume
target  \
close                           0.390660                0.607932 -
0.004094
volume                          0.455200                0.914741
0.023198
quote_asset_volume              0.466815                0.955486
0.018436
number_of_trades                1.000000                0.460050
0.014017
taker_buy_quote_volume          0.460050                1.000000
0.014626
target                          0.014017                0.014626
1.000000
price_range                     0.394248                0.609449 -
0.003894
close_ma_5                      0.390778                0.607825 -
```

```
0.003996
close_std_5                        0.725221                0.477097
0.012693
close_ma_10                        0.390831                0.607820 -
0.003942
close_std_10                       0.706030                0.487240
0.011987
close_ma_20                        0.390878                0.607821 -
0.003876
close_std_20                       0.678464                0.493314
0.011289
close_ma_30                        0.390910                0.607820 -
0.003827
close_std_30                       0.659731                0.495473
0.011261
close_lag_1                        0.390750                0.607780 -
0.004008
volume_lag_1                       0.669235                0.411873
0.014080
quote_volume_lag_1                 0.756043                0.426160
0.011087
close_lag_2                        0.390792                0.607792 -
0.003974
volume_lag_2                       0.414006                0.749095
0.021078
quote_volume_lag_2                 0.434261                0.813887
0.016634
close_lag_3                        0.390824                0.607799 -
0.003959
volume_lag_3                       0.407725                0.739849
0.021339
quote_volume_lag_3                 0.428672                0.805779
0.016945
close_lag_4                        0.390853                0.607802 -
0.003944
volume_lag_4                       0.405098                0.734608
0.020645
quote_volume_lag_4                 0.425952                0.801083
0.016366
close_lag_5                        0.390861                0.607803 -
0.003932
volume_lag_5                       0.401313                0.731018
0.020371
quote_volume_lag_5                 0.424179                0.798491
0.016133
rsi                                0.013391                0.041953 -
0.025348
macd                              -0.012388                0.003401 -
0.017848
```

```
macd_signal                    -0.008248                0.002354 -
0.014565
macd_diff                      -0.015024                0.003856 -
0.013515
bb_mavg                         0.390886                0.607847 -
0.003876
bb_high                         0.396371                0.609750 -
0.003718
bb_low                          0.385290                0.605859 -
0.004036
bb_width                        0.678465                0.493323
0.011289
ema_10                          0.390815                0.607844 -
0.003957
ema_30                          0.390908                0.607870 -
0.003852
ema_diff                       -0.012610                0.003623 -
0.018077

                        price_range  close_ma_5  close_std_5
close_ma_10  \
close                      0.999973    0.999991     0.518567
0.999979
volume                     0.428296    0.426111     0.440379
0.426118
quote_asset_volume         0.623475    0.621678     0.484346
0.621683
number_of_trades           0.394248    0.390778     0.725221
0.390831
taker_buy_quote_volume     0.609449    0.607825     0.477097
0.607820
target                    -0.003894   -0.003996     0.012693    -
0.003942
price_range                1.000000    0.999979     0.522157
0.999970
close_ma_5                 0.999979    1.000000     0.518711
0.999994
close_std_5                0.522157    0.518711     1.000000
0.518984
close_ma_10                0.999970    0.999994     0.518984
1.000000
close_std_10               0.536311    0.533031     0.889734
0.533205
close_ma_20                0.999950    0.999974     0.519431
0.999988
close_std_20               0.544901    0.541739     0.817567
0.541840
close_ma_30                0.999930    0.999952     0.519772
0.999969
```

close_std_30	0.548802	0.545714	0.789294	0.545767
close_lag_1	0.999982	0.999995	0.518612	0.999985
volume_lag_1	0.208662	0.205558	0.602039	0.205716
quote_volume_lag_1	0.429863	0.427107	0.763338	0.427258
close_lag_2	0.999976	0.999997	0.518695	0.999989
volume_lag_2	0.428073	0.426333	0.431776	0.426342
quote_volume_lag_2	0.624085	0.622666	0.478171	0.622675
close_lag_3	0.999969	0.999995	0.518785	0.999992
volume_lag_3	0.427784	0.426083	0.417914	0.426092
quote_volume_lag_3	0.623056	0.621672	0.466502	0.621681
close_lag_4	0.999963	0.999991	0.518882	0.999993
volume_lag_4	0.428079	0.426399	0.405770	0.426406
quote_volume_lag_4	0.623088	0.621726	0.456202	0.621735
close_lag_5	0.999956	0.999984	0.519014	0.999993
volume_lag_5	0.427790	0.426137	0.401225	0.426140
quote_volume_lag_5	0.623391	0.622045	0.453709	0.622050
rsi	0.005200	0.005166	-0.009784	0.003699
macd	0.009706	0.009809	-0.115706	0.007914
macd_signal	0.010495	0.010915	-0.108376	0.009875
macd_diff	-0.000457	-0.001397	-0.045317	-0.004367
bb_mavg	0.999950	0.999974	0.519426	0.999988
bb_high	0.999934	0.999920	0.525725	0.999936
bb_low	0.999866	0.999927	0.512994	0.999941
bb_width	0.544902	0.541740	0.817569	0.541842
ema_10	0.999974	0.999996	0.518965	

```
0.999999
ema_30                           0.999947    0.999968     0.519652
0.999982
ema_diff                         0.009936    0.010027    -0.116949
0.008097

                        ...      macd  macd_signal  macd_diff
bb_mavg  \
close                   ...  0.010645     0.011124   0.000672
0.999956
volume                  ...  0.000660     0.001540  -0.002535
0.426148
quote_asset_volume      ...  0.001724     0.002618  -0.002364
0.621712
number_of_trades        ... -0.012388    -0.008248  -0.015024
0.390886
taker_buy_quote_volume  ...  0.003401     0.002354   0.003856
0.607847
target                  ... -0.017848    -0.014565  -0.013515 -
0.003876
price_range             ...  0.009706     0.010495  -0.000457
0.999950
close_ma_5              ...  0.009809     0.010915  -0.001397
0.999974
close_std_5             ... -0.115706    -0.108376  -0.045317
0.519426
close_ma_10             ...  0.007914     0.009875  -0.004367
0.999988
close_std_10            ... -0.113127    -0.114604  -0.018102
0.533634
close_ma_20             ...  0.003818     0.006415  -0.007113
1.000000
close_std_20            ... -0.092383    -0.102269   0.011532
0.542120
close_ma_30             ...  0.000512     0.002892  -0.007114
0.999992
close_std_30            ... -0.072788    -0.085354   0.023566
0.545965
close_lag_1             ...  0.010424     0.011121  -0.000033
0.999963
volume_lag_1            ... -0.037668    -0.027435  -0.038540
0.205870
quote_volume_lag_1      ... -0.032810    -0.022586  -0.037542
0.427393
close_lag_2             ...  0.009978     0.011024  -0.001180
0.999968
volume_lag_2            ...  0.000301     0.001117  -0.002413
0.426373
quote_volume_lag_2      ...  0.001274     0.002194  -0.002535
```

```
0.622705
close_lag_3             ...  0.009366     0.010815  -0.002527
0.999973
volume_lag_3            ...  0.000271     0.000957  -0.002024
0.426125
quote_volume_lag_3      ...  0.001152     0.002006  -0.002356
0.621713
close_lag_4             ...  0.008632     0.010492  -0.003918
0.999977
volume_lag_4            ...  0.000289     0.000817  -0.001542
0.426440
quote_volume_lag_4      ...  0.001070     0.001831  -0.002092
0.621767
close_lag_5             ...  0.007813     0.010059  -0.005253
0.999981
volume_lag_5            ...  0.000393     0.000748  -0.000996
0.426174
quote_volume_lag_5      ...  0.001041     0.001688  -0.001752
0.622084
rsi                     ...  0.467648     0.374633   0.375323
0.001695
macd                    ...  1.000000     0.950990   0.348172
0.003818
macd_signal             ...  0.950990     1.000000   0.041233
0.006415
macd_diff               ...  0.348172     0.041233   1.000000 -
0.007113
bb_mavg                 ...  0.003818     0.006415  -0.007113
1.000000
bb_high                 ...  0.002703     0.005167  -0.006931
0.999951
bb_low                  ...  0.004947     0.007679  -0.007297
0.999950
bb_width                ... -0.092383    -0.102269   0.011532
0.542120
ema_10                  ...  0.007816     0.009483  -0.003495
0.999989
ema_30                  ...  0.002042     0.004001  -0.005530
0.999995
ema_diff                ...  0.999581     0.949190   0.352274
0.004025

                         bb_high    bb_low  bb_width    ema_10
ema_30  \
close                   0.999903  0.999911  0.541722  0.999985
0.999952
volume                  0.428730  0.423489  0.451688  0.426129
0.426166
quote_asset_volume      0.623606  0.619731  0.500165  0.621696
```

```
0.621734
number_of_trades        0.396371  0.385290  0.678465  0.390815
0.390908
taker_buy_quote_volume  0.609750  0.605859  0.493323  0.607844
0.607870
target                 -0.003718 -0.004036  0.011289 -0.003957 -
0.003852
price_range             0.999934  0.999866  0.544902  0.999974
0.999947
close_ma_5              0.999920  0.999927  0.541740  0.999996
0.999968
close_std_5             0.525725  0.512994  0.817569  0.518965
0.519652
close_ma_10             0.999936  0.999941  0.541842  0.999999
0.999982
close_std_10            0.540877  0.526244  0.905376  0.533231
0.533904
close_ma_20             0.999951  0.999950  0.542120  0.999989
0.999995
close_std_20            0.550425  0.533653  1.000000  0.541870
0.542427
close_ma_30             0.999947  0.999938  0.542446  0.999971
0.999997
close_std_30            0.553677  0.538098  0.951868  0.545799
0.546247
close_lag_1             0.999909  0.999917  0.541711  0.999990
0.999958
volume_lag_1            0.211484  0.200163  0.588225  0.205671
0.205904
quote_volume_lag_1      0.433502  0.421161  0.751266  0.427212
0.427425
close_lag_2             0.999915  0.999922  0.541721  0.999992
0.999963
volume_lag_2            0.429026  0.423643  0.457773  0.426353
0.426391
quote_volume_lag_2      0.624655  0.620666  0.505507  0.622686
0.622725
close_lag_3             0.999920  0.999927  0.541747  0.999993
0.999967
volume_lag_3            0.428778  0.423395  0.457659  0.426105
0.426143
quote_volume_lag_3      0.623660  0.619678  0.504681  0.621693
0.621733
close_lag_4             0.999924  0.999931  0.541784  0.999992
0.999971
volume_lag_4            0.429104  0.423699  0.458719  0.426420
0.426458
quote_volume_lag_4      0.623715  0.619731  0.504769  0.621747
0.621788
```

```
close_lag_5                0.999928  0.999933  0.541827  0.999990
0.999974
volume_lag_5               0.428815  0.423455  0.456691  0.426155
0.426192
quote_volume_lag_5         0.624027  0.620053  0.504511  0.622064
0.622105
rsi                        0.001699  0.001692  0.001212  0.003992
0.001272
macd                       0.002703  0.004947 -0.092383  0.007816
0.002042
macd_signal                0.005167  0.007679 -0.102269  0.009483
0.004001
macd_diff                 -0.006931 -0.007297  0.011532 -0.003495 -
0.005530
bb_mavg                    0.999951  0.999950  0.542120  0.999989
0.999995
bb_high                    1.000000  0.999801  0.550425  0.999937
0.999950
bb_low                     0.999801  1.000000  0.533653  0.999941
0.999941
bb_width                   0.550425  0.533653  1.000000  0.541871
0.542427
ema_10                     0.999937  0.999941  0.541871  1.000000
0.999983
ema_30                     0.999950  0.999941  0.542427  0.999983
1.000000
ema_diff                   0.002896  0.005168 -0.093481  0.008024
0.002248

                           ema_diff
close                      0.010903
volume                     0.000707
quote_asset_volume         0.001791
number_of_trades          -0.012610
taker_buy_quote_volume     0.003623
target                    -0.018077
price_range                0.009936
close_ma_5                 0.010027
close_std_5               -0.116949
close_ma_10                0.008097
close_std_10              -0.114038
close_ma_20                0.004025
close_std_20              -0.093481
close_ma_30                0.000756
close_std_30              -0.074179
close_lag_1                0.010667
volume_lag_1              -0.038300
quote_volume_lag_1        -0.033359
close_lag_2                0.010198
```

```
volume_lag_2                0.000360
quote_volume_lag_2          0.001347
close_lag_3                 0.009561
volume_lag_3                0.000342
quote_volume_lag_3          0.001234
close_lag_4                 0.008806
volume_lag_4                0.000372
quote_volume_lag_4          0.001160
close_lag_5                 0.007970
volume_lag_5                0.000488
quote_volume_lag_5          0.001141
rsi                         0.470734
macd                        0.999581
macd_signal                 0.949190
macd_diff                   0.352274
bb_mavg                     0.004025
bb_high                     0.002896
bb_low                      0.005168
bb_width                   -0.093481
ema_10                      0.008024
ema_30                      0.002248
ema_diff                    1.000000

[41 rows x 41 columns]

#importing libraries required for random forest and metrics for
evaluation
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import classification_report, precision_score,
recall_score
train.dropna(inplace = True)

#extract features from the DateTime
train['hour'] = train['timestamp'].dt.hour
train['minute'] = train['timestamp'].dt.minute
train['day'] = train['timestamp'].dt.dayofweek  # 0=Monday, 6=Sunday

#drop the original DateTime column if it is not needed
train.drop(columns=['timestamp'], inplace=True)

#splitting the training data into predictors and target
x = train.drop('target', axis = 1)
y = train['target']

#splitting x and y into training and validation sets
from sklearn.model_selection import train_test_split, GridSearchCV
train_size = 0.8
train_index = int(len(train) * train_size)
```

```
x_train, x_val = x[:train_index], x[train_index:]
y_train, y_val = y[:train_index], y[train_index:]

x_train.shape, x_val.shape, y_train.shape, y_val.shape
```

```
((1697900, 43), (424476, 43), (1697900,), (424476,))
```

```
y_train.head()
```

```
62    0.0
63    0.0
64    0.0
65    0.0
66    0.0
Name: target, dtype: float64
```

```
#making sure there are no nulls
x_train.isna().sum()
```

```
close                       0
volume                      0
quote_asset_volume          0
number_of_trades            0
taker_buy_quote_volume      0
price_range                 0
close_ma_5                  0
close_std_5                 0
close_ma_10                 0
close_std_10                0
close_ma_20                 0
close_std_20                0
close_ma_30                 0
close_std_30                0
close_lag_1                 0
volume_lag_1                0
quote_volume_lag_1          0
close_lag_2                 0
volume_lag_2                0
quote_volume_lag_2          0
close_lag_3                 0
volume_lag_3                0
quote_volume_lag_3          0
close_lag_4                 0
volume_lag_4                0
quote_volume_lag_4          0
close_lag_5                 0
volume_lag_5                0
quote_volume_lag_5          0
rsi                         0
macd                        0
macd_signal                 0
```

```
macd_diff                 0
bb_mavg                   0
bb_high                   0
bb_low                    0
bb_width                  0
ema_10                    0
ema_30                    0
ema_diff                  0
hour                      0
minute                    0
day                       0
dtype: int64
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV,
TimeSeriesSplit
from sklearn.metrics import make_scorer, f1_score
import numpy as np
import pandas as pd
from time import time
import gc
from sklearn.utils.class_weight import compute_class_weight

#calculating class wrights
unique_classes = np.unique(y_train)
class_weights = compute_class_weight('balanced',
classes=unique_classes, y=y_train)
class_weight_dict = dict(zip(unique_classes, class_weights))

#hyper parameter grid
param_grid = {
    'n_estimators': [800, 1000, 1200, 1500, 2000],
    'max_depth': [15, 20, 25, 30, 40, 50, None],
    'min_samples_split': [2, 5, 10, 15, 20, 30],
    'min_samples_leaf': [1, 2, 4, 6, 8, 10],
    'max_features': [0.2, 0.3, 0.4, 0.5, 0.6, 'sqrt', 'log2'],
    'criterion': ['gini', 'entropy'],
    'class_weight': ['balanced', 'balanced_subsample',
class_weight_dict],
    'max_samples': [0.3, 0.4, 0.5, 0.6, 0.7, 0.8],
    'bootstrap': [True]
}

#base RF model
rf = RandomForestClassifier(
    random_state=42,
    n_jobs=-1,
    verbose=1,
    oob_score=True
)
```

```python
#scoring based on f1
scorer = make_scorer(f1_score, average='macro')

#time series split with gap
tscv = TimeSeriesSplit(n_splits=5, gap=100)

print("Starting extensive hyperparameter search...")
start_time = time()

#randomised search through hyper parameters
random_search = RandomizedSearchCV(
    estimator=rf,
    param_distributions=param_grid,
    n_iter=35,  #couldnt look for more iterations than 20 cause of
computation load
    cv=tscv,
    scoring=scorer,
    verbose=2,
    random_state=42,
    n_jobs=-1,
    return_train_score=True
)

#fit on subset
subset_size = 500000
x_train_subset = x_train[-subset_size:] if isinstance(x_train,
pd.DataFrame) else x_train[-subset_size:]
y_train_subset = y_train[-subset_size:] if isinstance(y_train,
pd.Series) else y_train[-subset_size:]

random_search.fit(x_train_subset, y_train_subset)
print(f"\nOptimization completed in {(time() - start_time) / 60:.2f}
minutes")

#best parameters
best_params = random_search.best_params_
print("\nBest Parameters found:")
for param, value in best_params.items():
    print(f"{param}: {value}")

#clean up memory
del random_search, x_train_subset, y_train_subset
gc.collect()

#training the final model on the entire dataset
print("\nTraining final model on complete dataset...")
start_time = time()

#cobining the entire data
```

```
x_combined = pd.concat([x_train, x_val]) if isinstance(x_train,
pd.DataFrame) else np.concatenate([x_train, x_val])
y_combined = pd.concat([y_train, y_val]) if isinstance(y_train,
pd.Series) else np.concatenate([y_train, y_val])

#creating sample weights (temporal + class weights)
sample_weights = np.linspace(0.5, 1, len(y_combined))  # Time-based
weights
class_weight_map = {k: v for k, v in zip(unique_classes,
class_weights)}
class_weights_array = np.array([class_weight_map[y] for y in
y_combined])
final_sample_weights = sample_weights * class_weights_array

#training the final model on best paras obtained
final_model = RandomForestClassifier(
    **best_params,
    random_state=42,
    n_jobs=-1,
    verbose=1,
    oob_score=True
)

final_model.fit(x_combined, y_combined,
sample_weight=final_sample_weights)

print(f"\nFinal training completed in {(time() - start_time) / 60:.2f}
minutes")
print(f"Out of bag score: {final_model.oob_score_:.4f}")

#saving the model
import joblib
model_filename = f'rf_model_extensive_{time():.0f}.joblib'
joblib.dump(final_model, model_filename)
print(f"\nModel saved as {model_filename}")
```

```
Starting extensive hyperparameter search...
Fitting 5 folds for each of 35 candidates, totalling 175 fits
```

```
x_train.head()

#converting the test data as per the feature engineering of the train
data
test = pd.read_csv('test.csv')

test['price_range'] = test['high'] - test['low'] / test['open']
test.drop(['high', 'low', 'open', 'taker_buy_base_volume'], axis=1,
inplace=True)
test['timestamp'] = pd.to_datetime(test['timestamp'])

for window in [5, 10, 20, 30]:
```

```
    test[f'close_ma_{window}'] =
test['close'].rolling(window=window).mean()
    test[f'close_std_{window}'] =
test['close'].rolling(window=window).std()

for lag in range(1, 6):
    test[f'close_lag_{lag}'] = test['close'].shift(lag)
    test[f'volume_lag_{lag}'] = test['volume'].shift(lag)
    test[f'quote_volume_lag_{lag}'] =
test['quote_asset_volume'].shift(lag)

test.interpolate(method='linear', inplace=True)

rsi = RSIIndicator(close=test['close'], window=14)
test['rsi'] = rsi.rsi()

macd = MACD(close=test['close'], window_slow=26, window_fast=12,
window_sign=9)
test['macd'] = macd.macd()
test['macd_signal'] = macd.macd_signal()
test['macd_diff'] = macd.macd_diff()

bb = BollingerBands(close=test['close'], window=20, window_dev=2)
test['bb_mavg'] = bb.bollinger_mavg()
test['bb_high'] = bb.bollinger_hband()
test['bb_low'] = bb.bollinger_lband()
test['bb_width'] = test['bb_high'] - test['bb_low']

ema_10 = EMAIndicator(close=test['close'], window=10)
ema_30 = EMAIndicator(close=test['close'], window=30)
test['ema_10'] = ema_10.ema_indicator()
test['ema_30'] = ema_30.ema_indicator()
test['ema_diff'] = test['ema_10'] - test['ema_30']

test.fillna(method='ffill', inplace=True)


test.fillna(method='bfill', inplace=True)

test.fillna(test.mean(), inplace=True)


test['hour'] = test['timestamp'].dt.hour
test['minute'] = test['timestamp'].dt.minute
test['day'] = test['timestamp'].dt.dayofweek


test.drop(columns=['timestamp'], inplace=True)


price_cols = ['close', 'close_ma_5', 'close_std_5', 'close_lag_3',
```

```
'close_lag_4', 'close_lag_5']
volume_cols = ['volume', 'quote_asset_volume',
'taker_buy_quote_volume', 'volume_lag_2', 'quote_volume_lag_2',
'volume_lag_3', 'quote_volume_lag_3', 'volume_lag_4',
'quote_volume_lag_4', 'volume_lag_5', 'quote_volume_lag_5']
count_cols = ['number_of_trades']
derived_cols = ['price_range']


scaler_price = RobustScaler()
test[price_cols] = scaler_price.fit_transform(test[price_cols])

scaler_volume = QuantileTransformer(output_distribution='uniform')
test[volume_cols] = scaler_volume.fit_transform(test[volume_cols])

scaler_indicator = RobustScaler()
indicator_cols = ['rsi', 'macd', 'macd_signal', 'macd_diff',
'bb_mavg', 'bb_high', 'bb_low', 'bb_width', 'ema_10', 'ema_30',
'ema_diff']
test[indicator_cols] =
scaler_indicator.fit_transform(test[indicator_cols])

scaler_count = RobustScaler()
test[count_cols] = scaler_count.fit_transform(test[count_cols])

scaler_derived = StandardScaler()
test[derived_cols] = scaler_derived.fit_transform(test[derived_cols])

test.drop('row_id',axis = 1, inplace = True)

test_2 = pd.read_csv("test.csv")

#making predictions
predictions = model.predict(test)

#making sure that the predictions are of the correct shape
predictions.shape

#creating a dataframe based on the predictions
predictions_df = pd.DataFrame({
    'row_id': test_2['row_id'],
    'target': predictions
})

predictions_df.to_csv('predictions.csv', index=False)

print("Predictions saved to predictions.csv")
```