MLStratChallenge-Final

April 6, 2021

1 Algothon 2021 - ML Strategy Challenge

For : http://www.algothon.org/challenges.html

Joint work between Sanjit Neelam, and Chris Chia

Preliminaries : requires Python 3.8, the `numpy`, `pandas`, `lightgbm` `matplotlib` , `seaborn`, `statsmodels`, `scikit-learn`, `shap` packages. The `yfinance`, `quandl` packages are optional. Requires the data provided by the challenge organisers from https://drive.google.com/drive/folders/180FaVThDIFtmrCZ2cGiYskvlyvyMv5Au, which is included in the `data` directory.

2 Table of Contents

[2]:
```
import numpy as np
import pandas as pd
import lightgbm as lgb
from numpy import random
import shap

import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from sklearn.metrics import confusion_matrix

from ml_strat import *

import warnings
warnings.filterwarnings('ignore')

plt.style.use("fivethirtyeight")
```

```
<IPython.core.display.HTML object>
```

3 Task

- In short, the task is to come up with a **meta-model** that can predict the performance of the trade, and what position between $[-1, 1]$ to take in it. The evaluation metric is the *Sharpe Ratio* from the prediction and the actual return $\hat{y}_i \cdot r_i$.

4 Preprocessing

We read in all the provided data, define all transformations and apply them.

- For now, we only encode the Trading Date by Month, Date, Year
- The time stamp is time on and time off is fixed 8pm every day.

5 External Data

We select some external variables that might be general enough to help us predict performance

- U.S. Treasury Bond Futures, Continuous Contract #1 (US1) (Front Month). Link: https://www.quandl.com/data/CHRIS/CME_US1 #
- Crude Oil Futures, Continuous Contract (CL1) (Front Month). Link: https://www.quandl.com/data/CHRIS/CME_CL1 # #1 (
- Bond: U.S. Treasury Bond Futures, Continuous Contract #1 (US1) (Front Month) from Quandl. Link: https://www.quandl.com/data/CHRIS/CME_GC1
- USD index futures. Link: https://www.quandl.com/data/CHRIS/ICE_DX1
- S&P 500 Futures, Continuous Contract #1 (SP1)
- Gold Futures, Continuous Contract #1 (GC1) (Front Month)
- Commodity: S&P GSCI Index (^SPGSCI) from yfinance
- Currency: USD Index from investing.com
- Stock: https://www.quandl.com/data/CHRIS/CME_SP1
- US 10 Year T Note: https://uk.investing.com/rates-bonds/us-10-yr-t-note-historical-data

We use primarily the current **Open**, and previous **Close** and **Volume** prices.

[47]:
```
assets = ['bond_1', 'bond_2', 'commodity_1', 'commodity_2', 'currency_1',
          'currency_2', 'stock_1', 'stock_2']
symbols = [("-TBond Futures", "CHRIS/CME_US1"), ("Crude Futures", "CHRIS/
 ↪CME_CL1"),
           ("Gold Futures", "CHRIS/CME_CL2"), ("USD Futures", "CHRIS/ICE_DX1"),
           ("SP500 Futures","CHRIS/CME_SP1")]
tickers = ["^VIX", "^GSPC","^FTSE","USDGBP=X","EURGBP=X"]


#### Collect Data from APIs. If running locally, skip this step. ####
df = pd.DataFrame()
```

```python
#### join all the data ####
path = "data"
for a in assets:
    try:
        temp = pd.read_csv(f"{path}/{a}.csv")
    except:
        try:
            temp = pd.read_csv(f"{a}.csv")
        except:
            pass
    if temp.shape[0] != 0: #if temp is non-empty
        try:
            df = pd.concat([df, temp], axis=0).reset_index(drop=True)
        except:
            print("Some error")

df3 = preprocess(df)

#### Comment these lines if running locally. ####
# quandl_dfs = get_quandl_data(symbols, "pkLbjb4QQUmszgP48_jC", path)
dfs = get_y_finance_feats(tickers, path)

#### Read in saved data, rather than using API ####
quandl_dfs = {}
yf_dfs = {}
for x in symbols:
    quandl_dfs[x[0]] = pd.read_csv(f"{path}/{x[0]}.csv")
for x in tickers:
    yf_dfs[x] = pd.read_csv(f"{path}/{x}.csv")

us10yr = pd.read_csv(f"{path}/US 10 YR T-Note Futures Historical Data.csv")
us10yr['Date'] = pd.to_datetime(us10yr["Date"])
us10yr = us10yr[['Date','Open']]
us10yr.columns = ["Date","US10Yr-Open"]

df2  = df3.copy()
for x in yf_dfs:
    yf_dfs[x]['Date'] = pd.to_datetime(yf_dfs[x]['Date'])
    df2 = pd.merge(df2, yf_dfs[x], on ='Date', how='left')

for x in quandl_dfs:
    quandl_dfs[x]['Date'] = pd.to_datetime(quandl_dfs[x]['Date'])
    df2 = pd.merge(df2, quandl_dfs[x], on ='Date', how='left')

df2 = pd.merge(df2, us10yr, how='left',on='Date')

cat_feats =['Market']
```

```
df2[cat_feats]  = df2[cat_feats].astype("category")
display(df2)
```

```
           Date   Market    Return  Entry  Year  Month  Day  ^VIX-Open  \
0    2012-11-07   bond_1 -0.257833      0  2012     11    7  17.719999
1    2012-11-20   bond_1  0.022210      1  2012     11   20  15.110000
2    2012-11-28   bond_1 -0.436010      0  2012     11   28  16.430000
3    2012-12-03   bond_1 -0.209771      0  2012     12    3  15.810000
4    2012-12-05   bond_1 -0.313398      0  2012     12    5  16.950001
...         ...      ...       ...    ...   ...    ...  ...        ...
8767 2021-03-01  stock_2  1.345958      0  2021      3    1  25.200001
8768 2021-03-04  stock_2 -0.123132      7  2021      3    4  26.520000
8769 2021-03-05  stock_2  0.388406      9  2021      3    5  29.480000
8770 2021-03-09  stock_2  0.220305      1  2021      3    9  25.110001
8771 2021-03-11  stock_2 -0.214449      4  2021      3   11  22.500000

      ^VIX-Close  ^VIX-Rets  ...  Gold Futures-Volume  \
0      17.580000   0.007932  ...              72502.0
1      15.240000  -0.008567  ...              49831.0
2      15.920000   0.031533  ...              37492.0
3      15.870000  -0.003788  ...              56936.0
4      17.120001  -0.009980  ...              38194.0
...          ...        ...  ...                  ...
8767   27.950001  -0.103573  ...             120240.0
8768   26.670000  -0.005640  ...             163978.0
8769   28.570000   0.031355  ...             229281.0
8770   25.469999  -0.014235  ...             193133.0
8771   22.559999  -0.002663  ...             214240.0

      Gold Futures-Previous Day Open Interest  Gold Futures-Open  \
0                                    225676.0              88.76
1                                    134660.0              89.60
2                                    158521.0              87.94
3                                    174864.0              89.45
4                                    185398.0              89.03
...                                       ...                ...
8767                                 284806.0              61.42
8768                                 289658.0              60.93
8769                                 299696.0              63.94
8770                                 315940.0              64.60
8771                                 345254.0              64.62

      USD Futures-Volume  USD Futures-Prev. Day Open Interest  \
0                24315.0                              35614.0
1                17425.0                              36993.0
2                18571.0                              32940.0
3                15892.0                              32939.0
4                21297.0                              38796.0
```

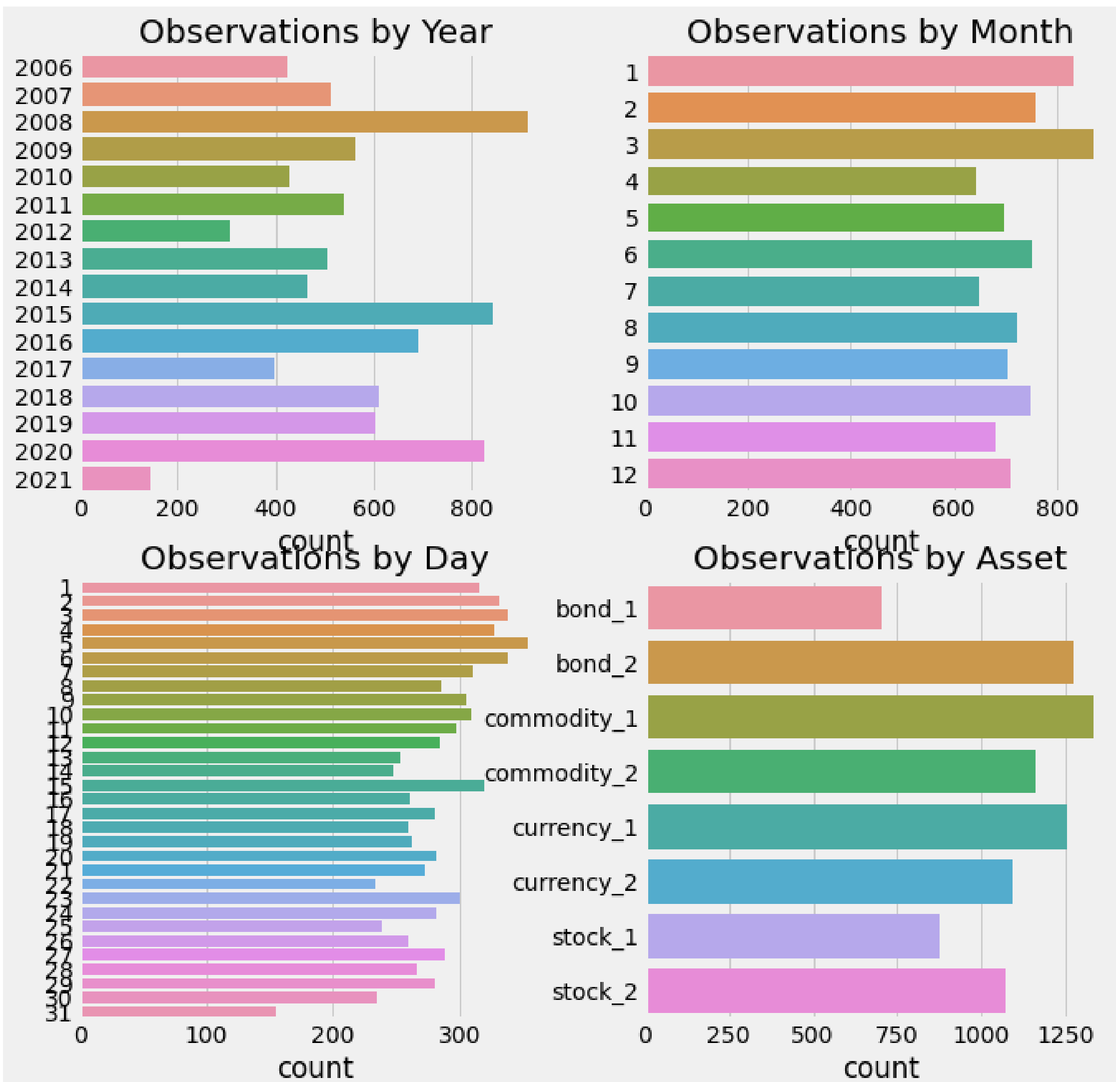
```
...               ...                                     ...
8767          49289.0                                 35802.0
8768          28972.0                                 33745.0
8769          36468.0                                 35911.0
8770          41680.0                                 33927.0
8771          48926.0                                 25550.0

      USD Futures-Open  SP500 Futures-Volume  \
0               80.785                5703.0
1               81.025               11007.0
2               80.390                8179.0
3               80.195                8357.0
4               79.620                7556.0
...                ...                   ...
8767            90.900                5078.0
8768            91.025                1550.0
8769            91.640                4114.0
8770            92.445                1425.0
8771            91.855                3450.0

      SP500 Futures-Previous Day Open Interest  SP500 Futures-Open  \
0                                     204290.0              1424.9
1                                     220451.0              1379.9
2                                     216620.0              1397.4
3                                     211908.0              1413.2
4                                     208499.0              1401.8
...                                        ...                 ...
8767                                   30661.0              3846.7
8768                                   32429.0              3803.4
8769                                   31598.0                 NaN
8770                                   29699.0                 NaN
8771                                   31631.0                 NaN

      US10Yr-Open
0          132.66
1          133.95
2          133.77
3          134.05
4          134.09
...           ...
8767       134.27
8768       134.06
8769       133.44
8770       132.97
8771       133.58

[8772 rows x 38 columns]
```

6 Exploratory Data Analysis

Number of observations by Year, Month, Day, Asset

[45]:
```
fig, ax = plt.subplots(figsize=(10, 10), ncols = 2, nrows = 2)
sns.countplot(y=df2['Year'].values, ax = ax[0, 0])
ax[0, 0].set_title("Observations by Year");
sns.countplot(y=df2['Month'].values, ax = ax[0, 1])
ax[0, 1].set_title("Observations by Month");
sns.countplot(y=df2['Day'].values, ax = ax[1, 0])
ax[1, 0].set_title("Observations by Day");
sns.countplot(y=df2['Market'].values, ax = ax[1, 1])
ax[1, 1].set_title("Observations by Asset");
```

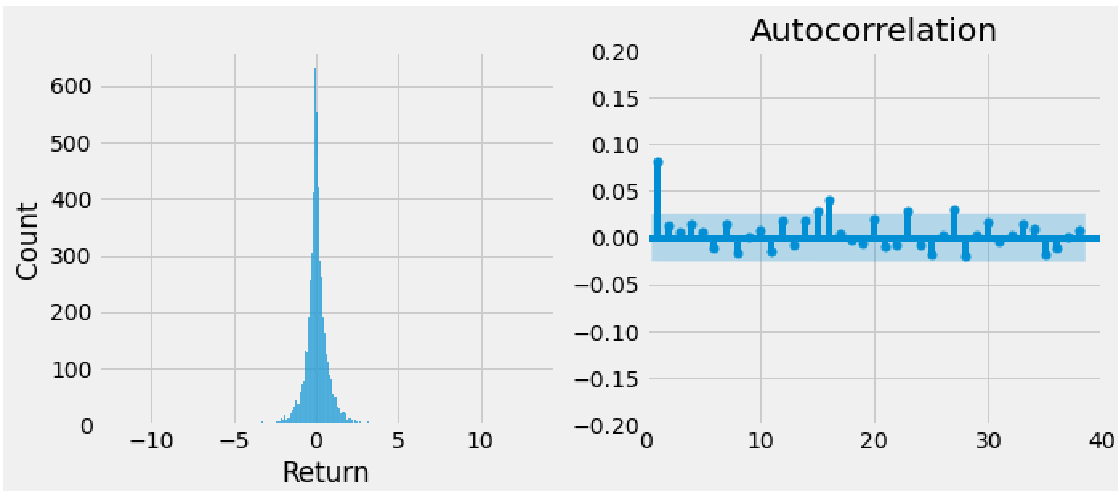
[45]:
```
Text(0.5, 1.0, 'Observations by Asset')
```



Distribution of Returns; Returns look leptokurtif / fat-tailed. Although Autocorrelation is likely computed incorrectly, there seems to be some levels of clustering in trading returns.

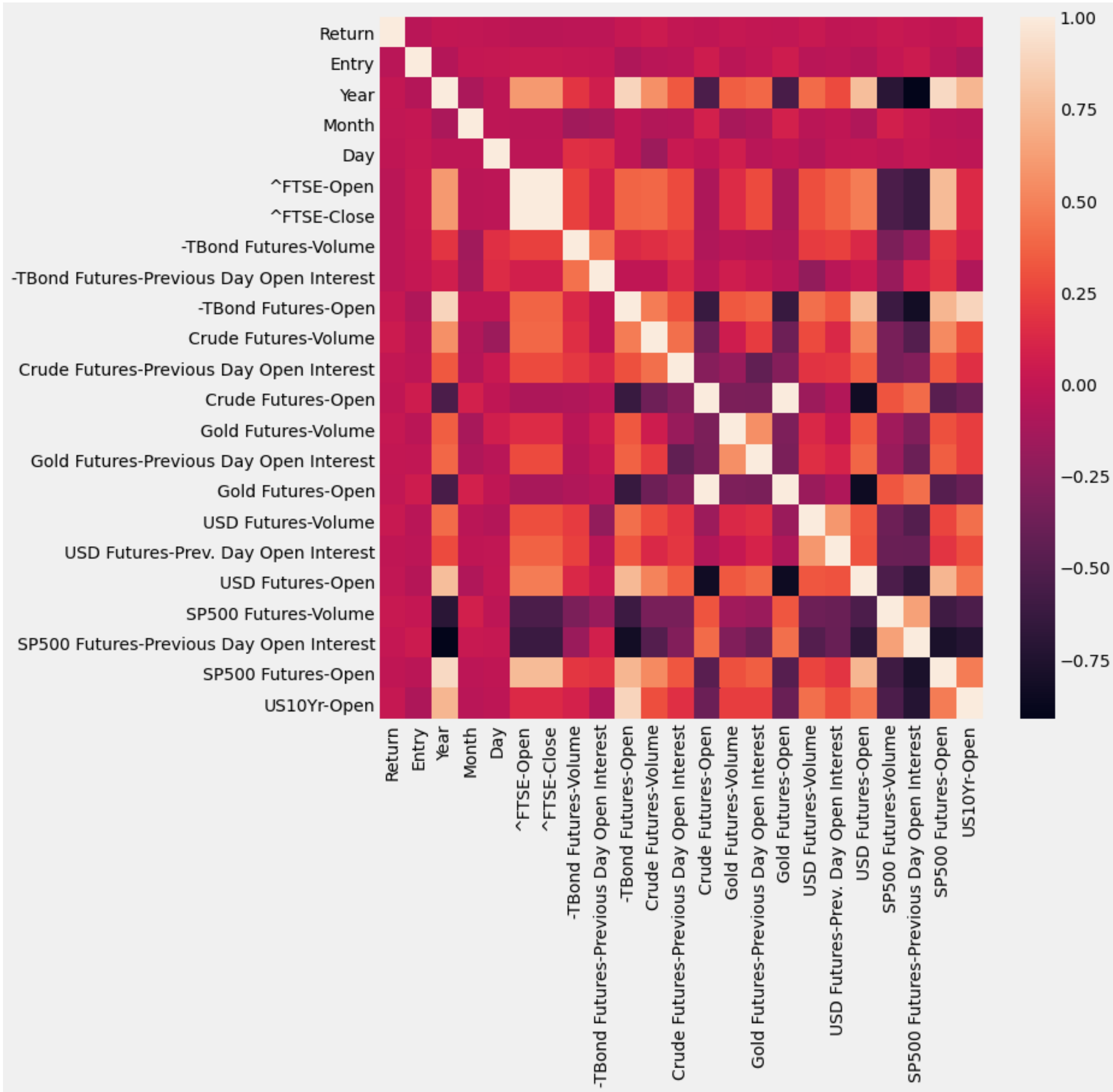
[101]:
```
fig, ax = plt.subplots(ncols = 2, figsize=(10, 4))
sns.histplot(df2['Return'], ax = ax[0])
rets = df.groupby(pd.to_datetime(df['Position_Open']))['Return'].mean()
plot_acf(rets, ax = ax[1]);
ax[1].set_xlim([0, 40]);
ax[1].set_ylim([-0.2, 0.2]);
```

[101]:
```
(-0.2, 0.2)
```



[81]:
```
fig, ax = plt.subplots(figsize=(10, 10))
sns.heatmap(df2.corr())
```

[81]:
```
<AxesSubplot:>
```



[31]:
```
df2.columns
```

[31]:
```
Index(['Date', 'Market', 'Return', 'Entry', 'Year', 'Month', 'Day',
       '^FTSE-Open', '^FTSE-Close', '-TBond Futures-Volume',
       '-TBond Futures-Previous Day Open Interest', '-TBond Futures-Open',
       'Crude Futures-Volume', 'Crude Futures-Previous Day Open Interest',
       'Crude Futures-Open', 'Gold Futures-Volume',
       'Gold Futures-Previous Day Open Interest', 'Gold Futures-Open',
       'USD Futures-Volume', 'USD Futures-Prev. Day Open Interest',
       'USD Futures-Open', 'SP500 Futures-Volume',
       'SP500 Futures-Previous Day Open Interest', 'SP500 Futures-Open',
       'US10Yr-Open'],
      dtype='object')
```

Features have some, but limited signal with trading return

[48]:
```
df2.corrwith(df2['Return']).abs().sort_values(ascending=False)
```

[48]:
```
Return                                         1.000000
^VIX-Close                                     0.082373
^VIX-Open                                      0.080933
Crude Futures-Volume                           0.050153
Entry                                          0.039781
^FTSE-Close                                    0.037595
^FTSE-Open                                     0.037567
USDGBP=X-Rets                                  0.034474
EURGBP=X-Rets                                  0.033328
-TBond Futures-Volume                          0.028739
USD Futures-Volume                             0.027646
SP500 Futures-Volume                           0.026102
-TBond Futures-Previous Day Open Interest      0.025513
US10Yr-Open                                    0.020462
Gold Futures-Volume                            0.016130
-TBond Futures-Open                            0.015148
Day                                            0.014485
^FTSE-Rets                                     0.013847
^GSPC-Open                                     0.012689
^GSPC-Close                                    0.012655
USD Futures-Prev. Day Open Interest            0.012611
SP500 Futures-Previous Day Open Interest       0.011384
SP500 Futures-Open                             0.010007
Crude Futures-Open                             0.009365
Gold Futures-Open                              0.008974
USD Futures-Open                               0.007598
EURGBP=X-Open                                  0.005811
Crude Futures-Previous Day Open Interest       0.004457
Month                                          0.004252
EURGBP=X-Close                                 0.003736
^VIX-Rets                                      0.002473
USDGBP=X-Open                                  0.001289
^GSPC-Rets                                     0.000603
USDGBP=X-Close                                 0.000309
Year                                           0.000163
Gold Futures-Previous Day Open Interest        0.000027
dtype: float64
```

[56]:
```
np.unique(list(map(str,pd.to_datetime(df['Position_Open']).dt.time)))
```

[56]:
```
array(['15:00:00', '15:30:00', '16:00:00', '16:30:00', '17:00:00',
       '17:30:00', '18:00:00', '18:30:00', '19:00:00', '19:30:00'],
      dtype='<U8')
```

That VIX is *very* correlated with the trade returns, suggests that there might be data leakage - since the data is from yahoo finance it has the potential to be fairly unclean. However, if indeed the yahoo finance open corresponds to US market opens, there should be no data leakage since all trades in the dataset open at the earliest 15:00.

For additional features, We could also potenitally include further lags of OI, volume and other features

7 Modelling: LightGBM

We use Gradient Tree Boosting because of the lack of knowledge of the *time series structure* of the data. LightGBM is a popular variant of gradient boosted trees that allows for relatively fast training

We test multiple position-sizing strategies, which could be something that is further investigated:

- Long-short, predict a value between [-1, 1]
- Hard-classification Long-Short, predict -1, 0, 1
- Long-only, predict a value between [0, 1]
- Hard-classification Long only: predict or 0, 1

[57]:
```
sharpe = lambda x: np.mean(x / np.std(x))
# position sizing strategies
identity = lambda p: p
ls_pos = lambda p: 2 * p - 1 # scale prediction p so that it is between [-1, 1]
hard_l = lambda p: p > 0.5 # hard boundary, so the signal is either [0, 1]
hard_ls = lambda p: np.rint(2 * (p > 0.5) - 1) # hard boundary, the signal is␣
↪either -1, 1
```

[67]:
```
drop_feats = ['Date','Year','Return']
X_train, X_val, X_test, y_train, y_val, y_test, train_ind, val_ind, test_ind =␣
↪split(df2, drop_feats = drop_feats)

### LightGBM Model ###
train_data = lgb.Dataset(X_train, label = y_train > 0, categorical_feature =␣
↪['Market','Entry','Month','Day'])
val_data = lgb.Dataset(X_val, label = y_val > 0,categorical_feature =␣
↪['Market','Entry','Month','Day'])

SEED = random.randint(0, 999999)
print("SEED", SEED, "\n")

param = {"num_leaves": 64,
           "max_depth":8,
           "objective": "binary",
           "num_round": 500,
           "verbose": -1,
         "early_stopping_rounds":20,
```

```
          "seed": SEED}

evals_result = {}
bst = lgb.train(param, train_data,␣
 ↪valid_sets=[val_data],evals_result=evals_result,verbose_eval=False)

train_preds = bst.predict(X_train)
val_preds = bst.predict(X_val)
test_preds = bst.predict(X_test)

#### Write Results ####
try:
    results = pd.read_csv(f"{path}/backtest_log.csv")
except:
    baseline = list(map(sharpe, [y_train, y_val, y_test])) + ['always long']
    perfect = list(map(lambda x: sharpe(abs(x)), [y_train, y_val, y_test])) +␣
 ↪['perfect sharpe']
    results = pd.DataFrame([baseline, perfect],␣
 ↪columns=['train_sharpe','val_sharpe','test_sharpe','description'])

for description, f in zip(["LGB Classification (probability)", "LGB␣
 ↪Classification (2x - 1 scaling)",
              "LGB Hard Classification Long-only", "LGB Hard Classification␣
 ↪Long-Short"],
             [identity, ls_pos, hard_l, hard_ls]):

    temp = pd.DataFrame([[sharpe(f(train_preds) * y_train), sharpe(f(val_preds)␣
 ↪* y_val),
                      sharpe(f(test_preds) * y_test), description, str(param)]])
    temp.columns = ['train_sharpe','val_sharpe','test_sharpe','description',␣
 ↪'hyperparameters']
    results = pd.concat([results, temp], axis = 0)

results = results.
 ↪drop_duplicates(subset=['train_sharpe','test_sharpe','val_sharpe'])
results.reset_index(drop=True).to_csv(f"{path}/backtest_log.csv", index=False)
display(results)
```

```
(5497, 35) (5497,)
(1089, 35) (1089,)
(1214, 35) (1214,)

SEED 14481


    train_sharpe  val_sharpe  test_sharpe  \
0       0.087039   -0.003976     0.066723
1       0.695343    0.826045     0.809421
```

2	0.094210	-0.003556	0.067892
3	0.195920	0.003650	0.078369
4	0.125941	0.009718	0.079376
5	0.152040	0.022515	0.085299
6	0.094210	-0.003556	0.067892
7	0.195920	0.003650	0.078369
8	0.094210	-0.003556	0.067892
9	0.195920	0.003650	0.078369
10	0.125941	0.009718	0.079376
11	0.152040	0.022515	0.085299
12	0.101515	-0.002767	0.068175
13	0.257908	0.017378	0.034970
14	0.184922	0.014136	0.072440
15	0.243896	0.025946	0.036663
16	0.090049	-0.003145	0.066260
17	0.139830	0.017840	0.031175
18	0.103940	-0.000008	0.051410
19	0.116588	0.003964	0.019999
20	0.087757	-0.003981	0.066486
21	0.102373	-0.003815	0.055379
22	0.087039	-0.003976	0.066723
0	0.184922	0.014136	0.072440
0	0.243896	0.025946	0.036663

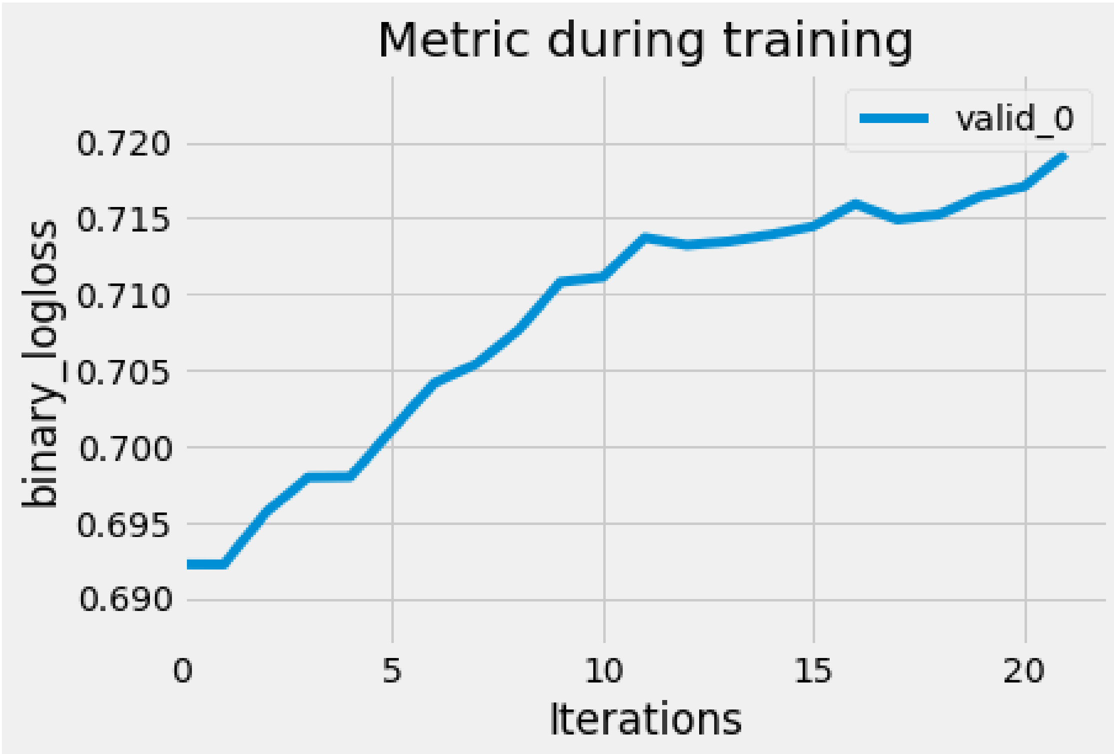
	description \
0	always long
1	perfect sharpe
2	LGB Classification (probability)
3	LGB Classification (2x - 1 scaling)
4	LGB Hard Classification Long-only
5	LGB Hard Classification Long-Short
6	LGB Classification (probability)
7	LGB Classification (2x - 1 scaling)
8	LGB Classification (probability)
9	LGB Classification (2x - 1 scaling)
10	LGB Hard Classification Long-only
11	LGB Hard Classification Long-Short
12	LGB Classification (probability)
13	LGB Classification (2x - 1 scaling)
14	LGB Hard Classification Long-only
15	LGB Hard Classification Long-Short
16	LGB Classification (probability)
17	LGB Classification (2x - 1 scaling)
18	LGB Hard Classification Long-only
19	LGB Hard Classification Long-Short
20	LGB Classification (probability)
21	LGB Classification (2x - 1 scaling)
22	LGB Hard Classification Long-only

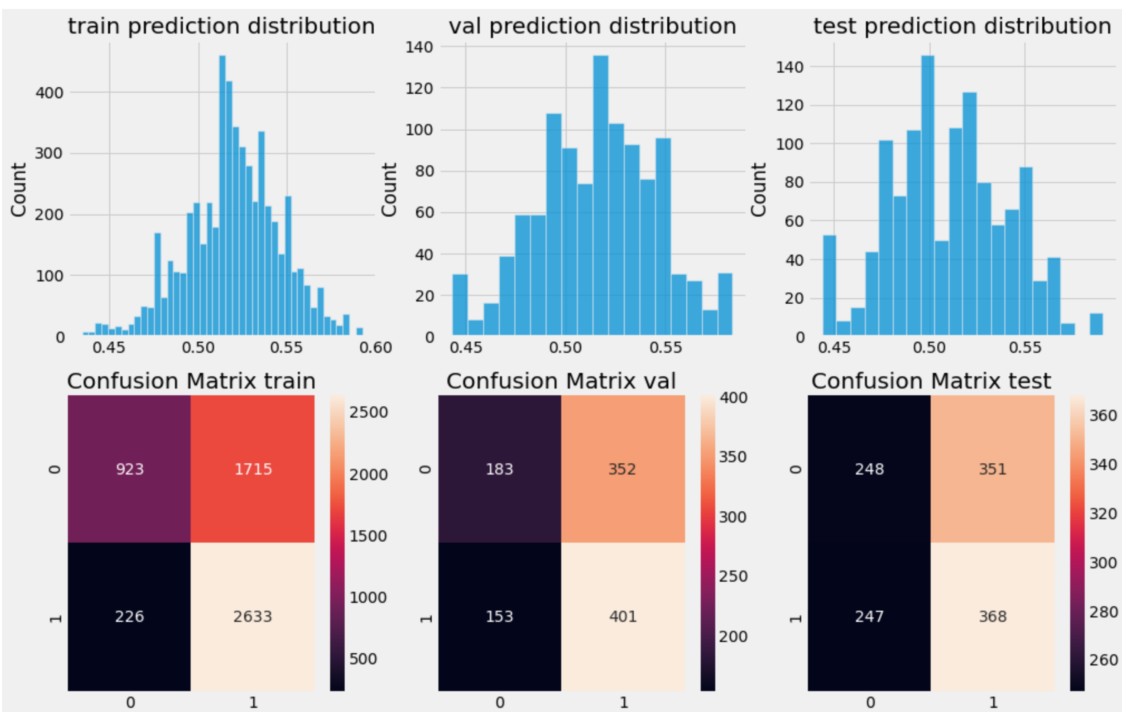
```
0     LGB Hard Classification Long-only
0    LGB Hard Classification Long-Short

                                      hyperparameters
0                                                 NaN
1                                                 NaN
2   {'num_leaves': 64, 'max_depth': 8, 'objective'...
3   {'num_leaves': 64, 'max_depth': 8, 'objective'...
4   {'num_leaves': 64, 'max_depth': 8, 'objective'...
5   {'num_leaves': 64, 'max_depth': 8, 'objective'...
6   {'num_leaves': 64, 'max_depth': 8, 'objective'...
7   {'num_leaves': 64, 'max_depth': 8, 'objective'...
8   {'num_leaves': 64, 'max_depth': 8, 'objective'...
9   {'num_leaves': 64, 'max_depth': 8, 'objective'...
10  {'num_leaves': 64, 'max_depth': 8, 'objective'...
11  {'num_leaves': 64, 'max_depth': 8, 'objective'...
12  {'num_leaves': 64, 'max_depth': 8, 'objective'...
13  {'num_leaves': 64, 'max_depth': 8, 'objective'...
14  {'num_leaves': 64, 'max_depth': 8, 'objective'...
15  {'num_leaves': 64, 'max_depth': 8, 'objective'...
16  {'num_leaves': 16, 'max_depth': 5, 'objective'...
17  {'num_leaves': 16, 'max_depth': 5, 'objective'...
18  {'num_leaves': 16, 'max_depth': 5, 'objective'...
19  {'num_leaves': 16, 'max_depth': 5, 'objective'...
20  {'num_leaves': 16, 'max_depth': 2, 'objective'...
21  {'num_leaves': 16, 'max_depth': 2, 'objective'...
22  {'num_leaves': 16, 'max_depth': 2, 'objective'...
0   {'num_leaves': 64, 'max_depth': 8, 'objective'...
0   {'num_leaves': 64, 'max_depth': 8, 'objective'...
```

7.1 Evaluation: Predictions

In addition to evaluating the performance of the model in terms of Sharpe, we can evaluate the performance of the model in terms of its *machine learning* performance. FOr many of our runs, we find that the validation error consistently increases.

[68]:
```
lgb.plot_metric(evals_result)
fig, ax = plt.subplots(ncols = 3, figsize=(15, 10), nrows = 2)
names = ["train","val","test"]
for i, x in enumerate(zip([train_preds, val_preds, test_preds], [y_train,␣
 ↪y_val, y_test])):
    sns.histplot(x[0], ax = ax[0, i])
    ax[0, i].set_title(f"{names[i]} prediction distribution")
    sns.heatmap(confusion_matrix(x[1] > 0, x[0] > 0.5), ax=ax[1, i],annot=True,␣
 ↪fmt='g')
    ax[1, i].set_title(f"Confusion Matrix {names[i]}")
```


Metric during training
valid_0
binary_logloss
0.720
0.715
0.710
0.705
0.700
0.695
0.690
0
5
10
15
20
Iterations

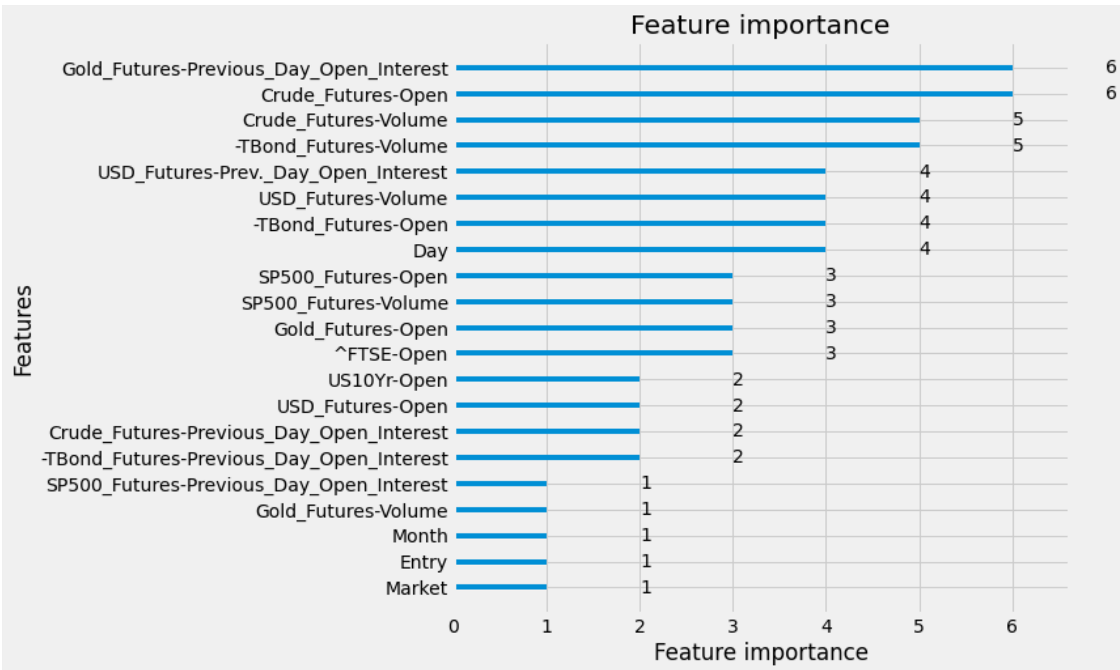

train prediction distribution
val prediction distribution
test prediction distribution
Count
Confusion Matrix train
923
1715
226
2633
Confusion Matrix val
183
352
153
401
Confusion Matrix test
248
351
247
368

7.2 Evaluation: Interpretability

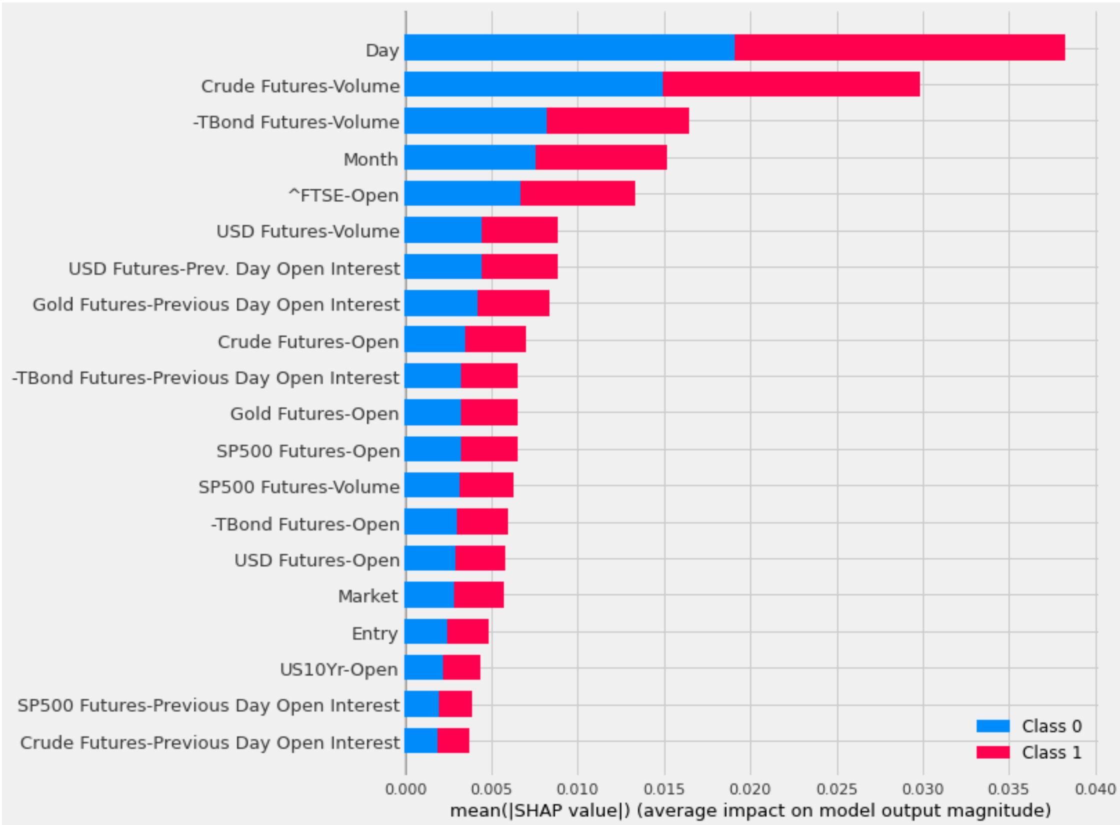
Finally, we should consider *model intepretability*; in short, how the model uses the features to come to a prediction. We do this in two ways: 1) using `lightgbm`'s in-built feature importance, which is based on mean decrease in gini impurity, and the `shap` library's *Shapley Values*, which measures how each feature contributes to the final prediction.

[117]:
```
#### Diagnosis ####
fig, ax = plt.subplots(figsize=(7.5, 7.5))
lgb.plot_importance(bst, ax)
```

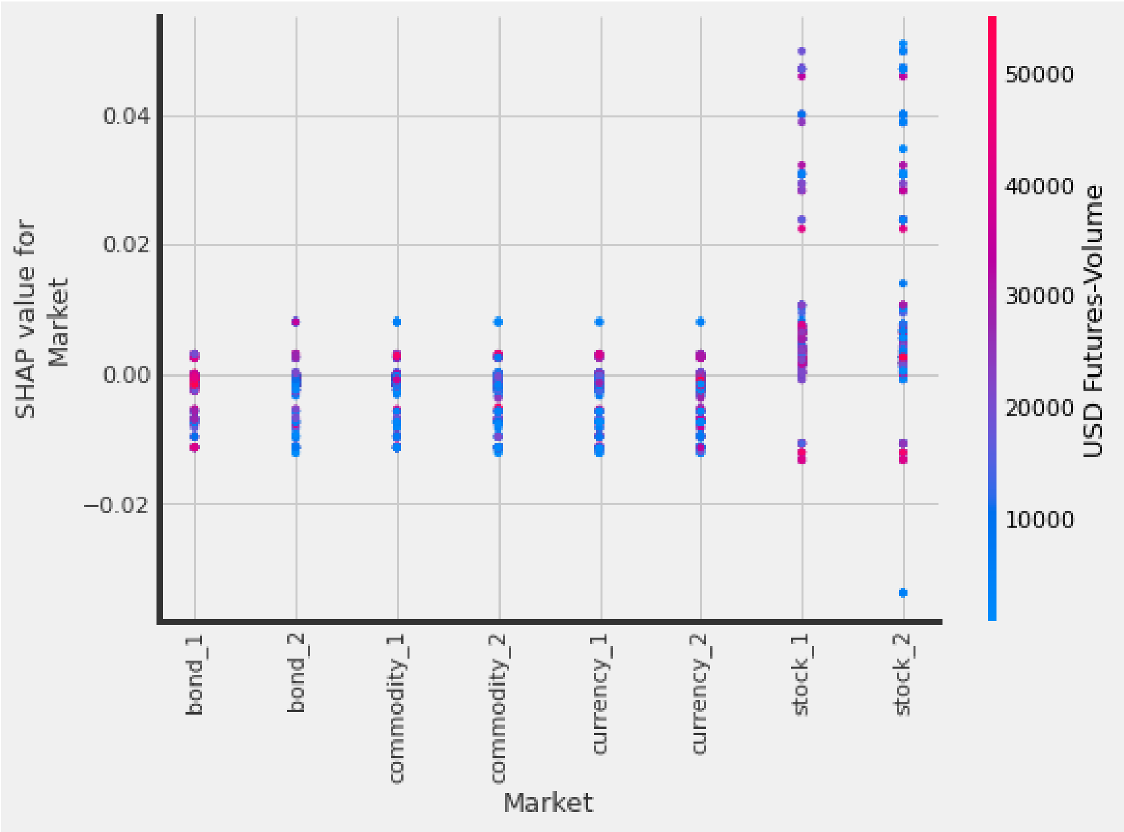
[117]:
```
<AxesSubplot:title={'center':'Feature importance'}, xlabel='Feature importance',
ylabel='Features'>
```



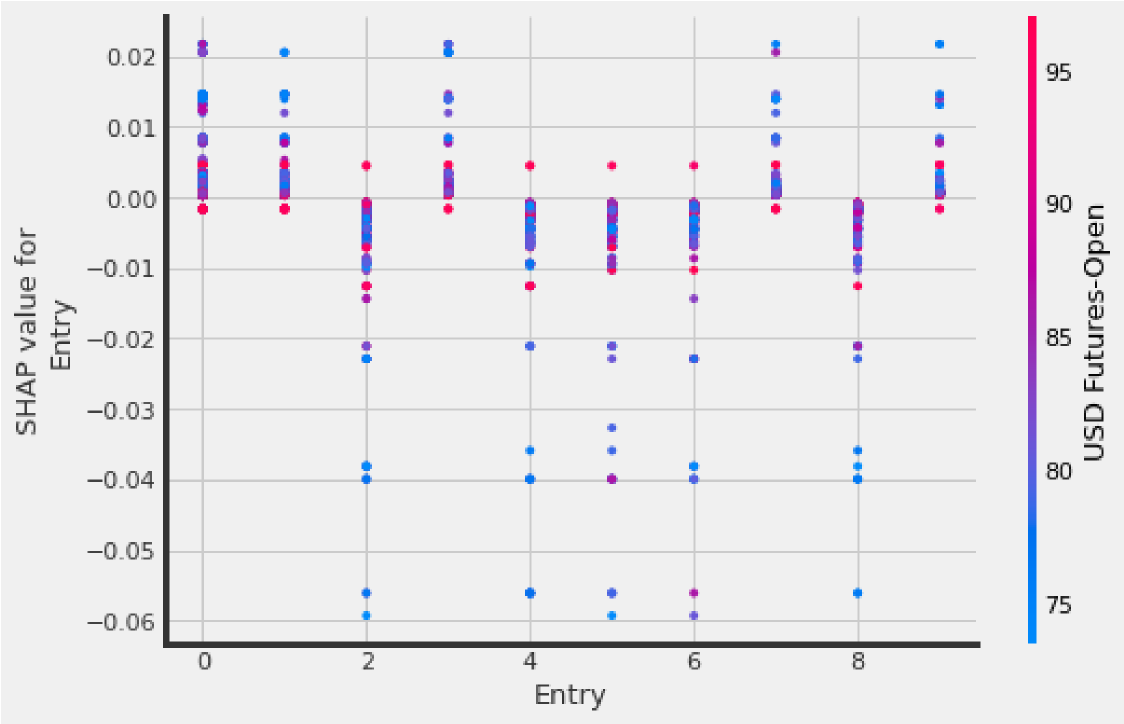
[129]:
```
explainer = shap.TreeExplainer(bst)
shap_values = explainer.shap_values(X_train)
shap.summary_plot(shap_values, feature_names = X_train.columns)
```



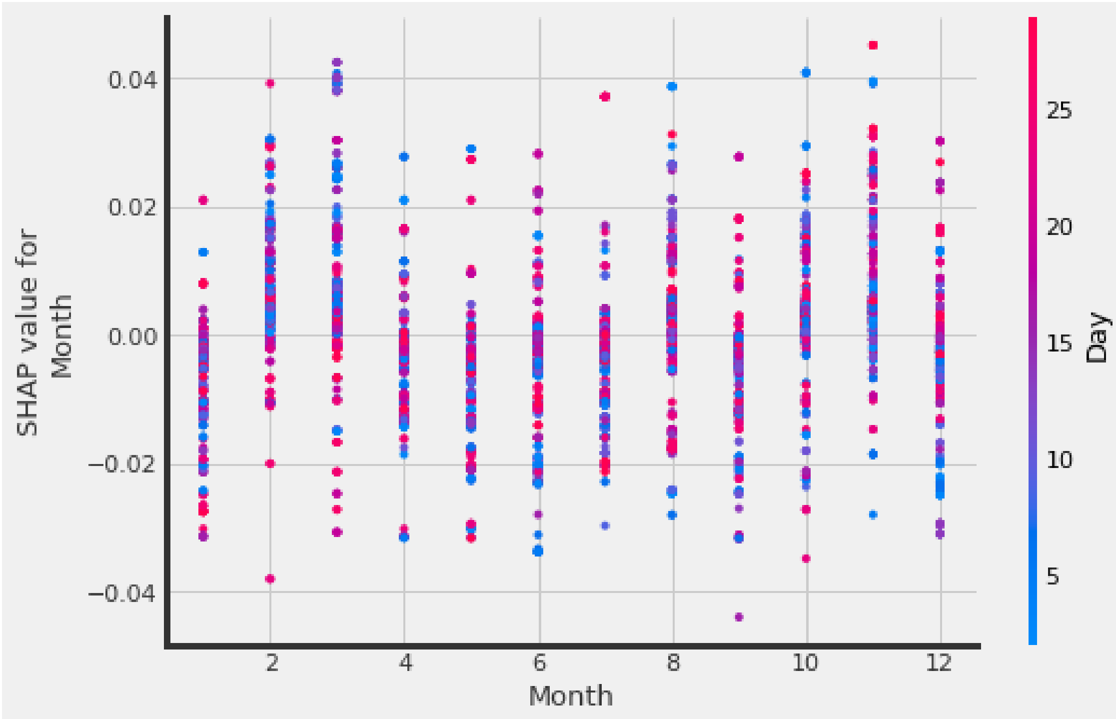
[135]:
```
for i,x in enumerate(X_train.columns):
    shap.dependence_plot(i, shap_values[1], X_train)
```

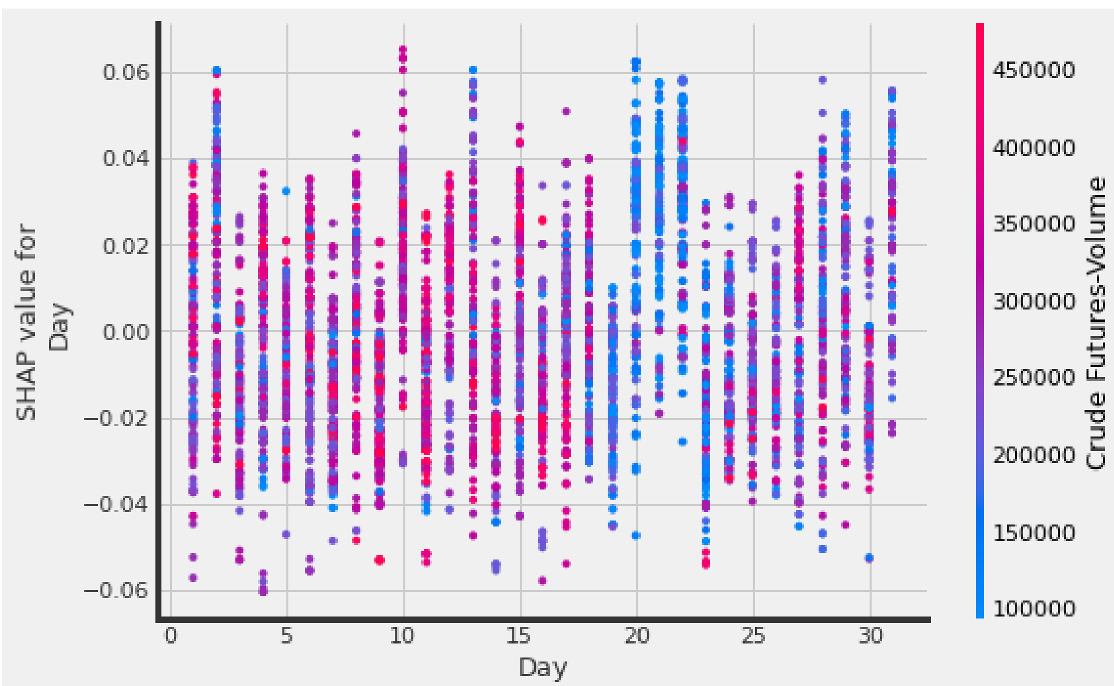


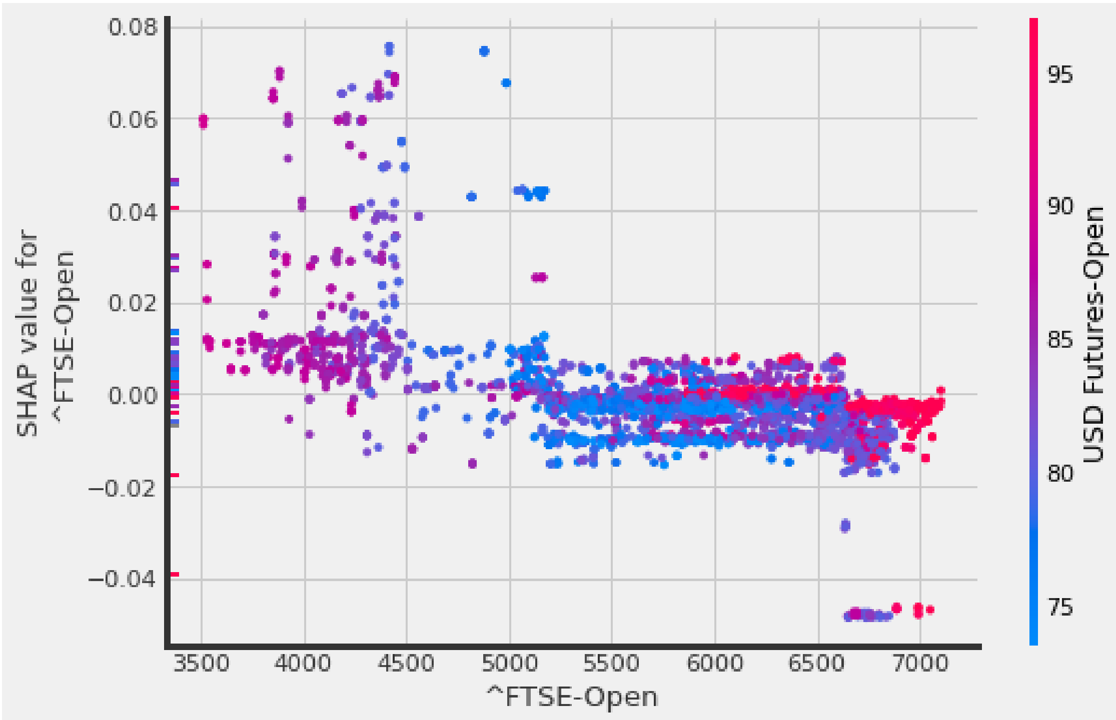
SHAP value for
Market
0.04
0.02
0.00
−0.02
bond_1
bond_2
commodity_1
commodity_2
currency_1
currency_2
stock_1
stock_2
Market
50000
40000
30000
20000
10000
USD Futures-Volume



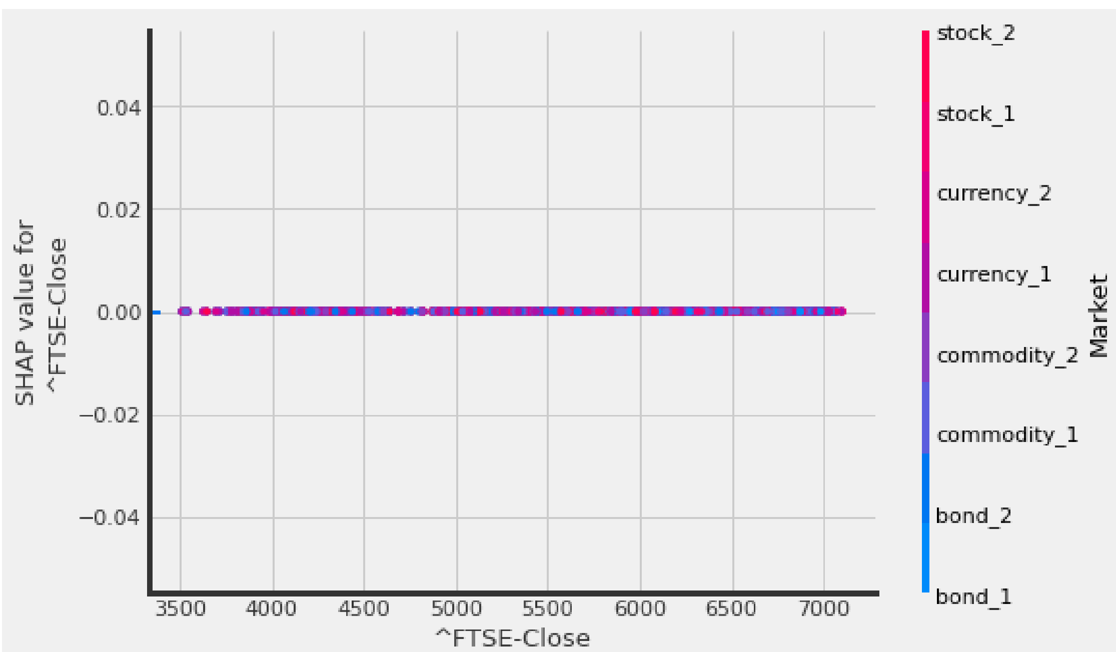
SHAP value for
Entry
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
0
2
4
6
8
Entry
95
90
85
80
75
USD Futures-Open


SHAP value for
Month
Month
Day
0.04
0.02
0.00
−0.02
−0.04
2
4
6
8
10
12
25
20
15
10
5

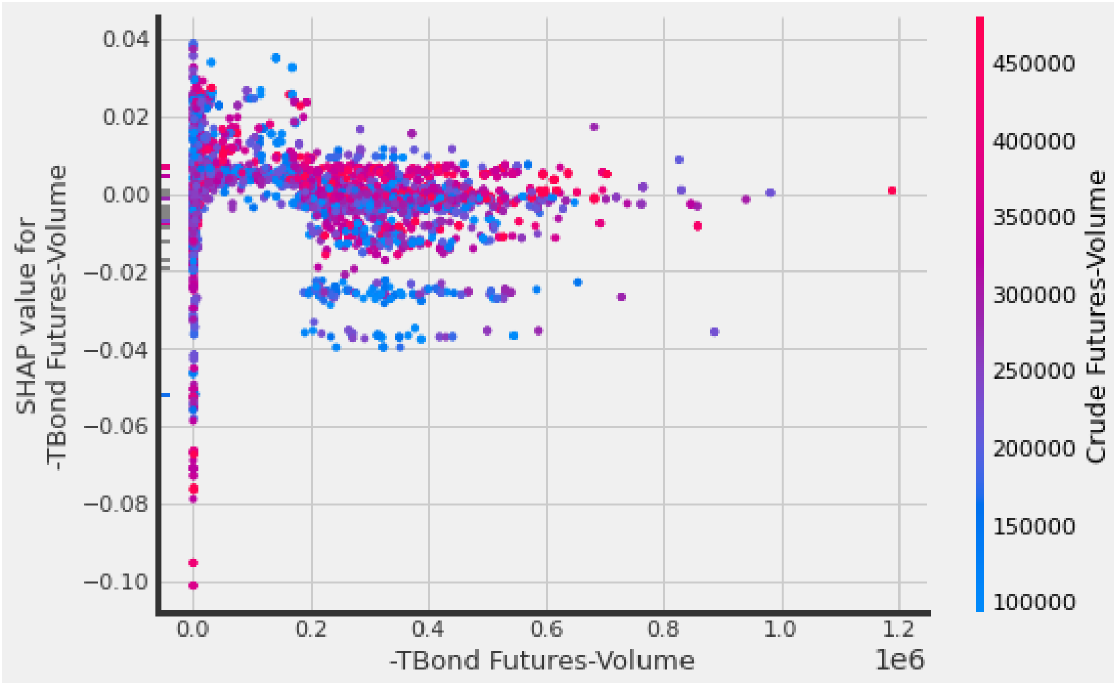

SHAP value for
Day
Day
Crude Futures-Volume
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
0
5
10
15
20
25
30
450000
400000
350000
300000
250000
200000
150000
100000

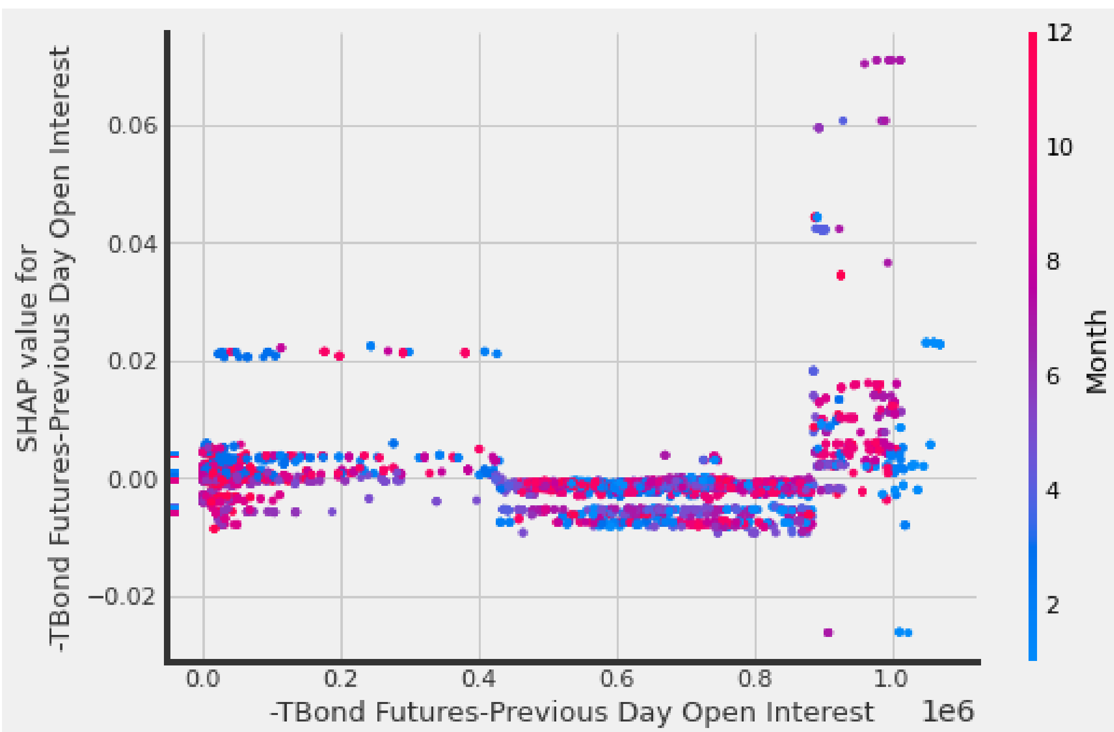


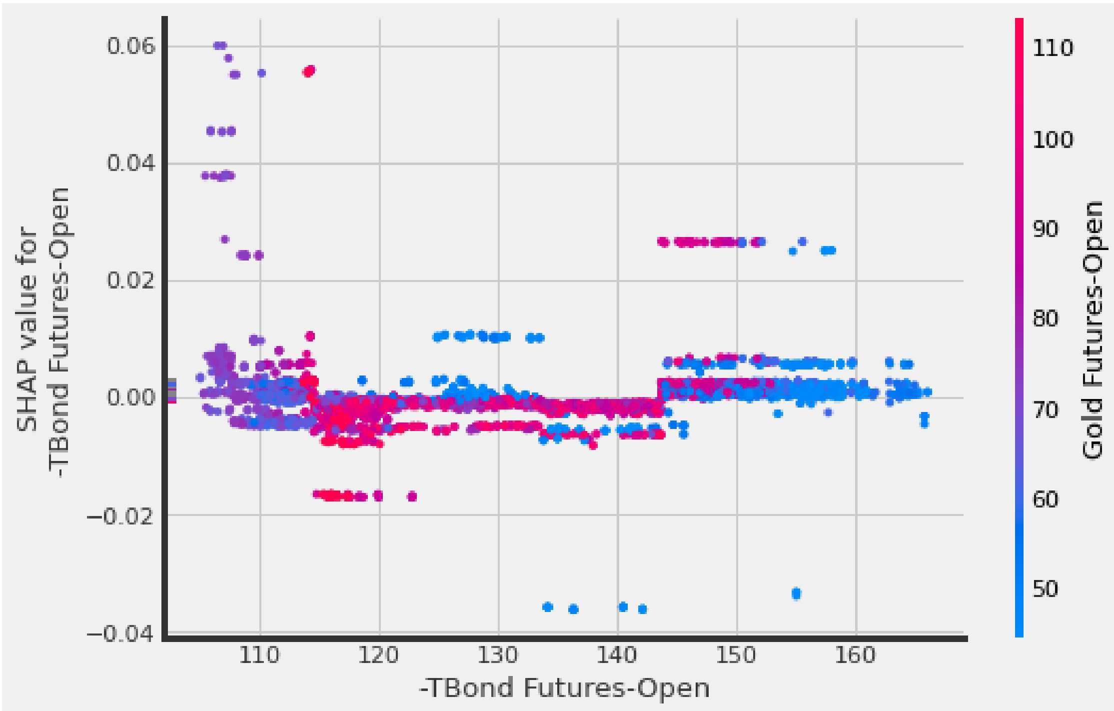
SHAP value for ^FTSE-Open
^FTSE-Open
USD Futures-Open

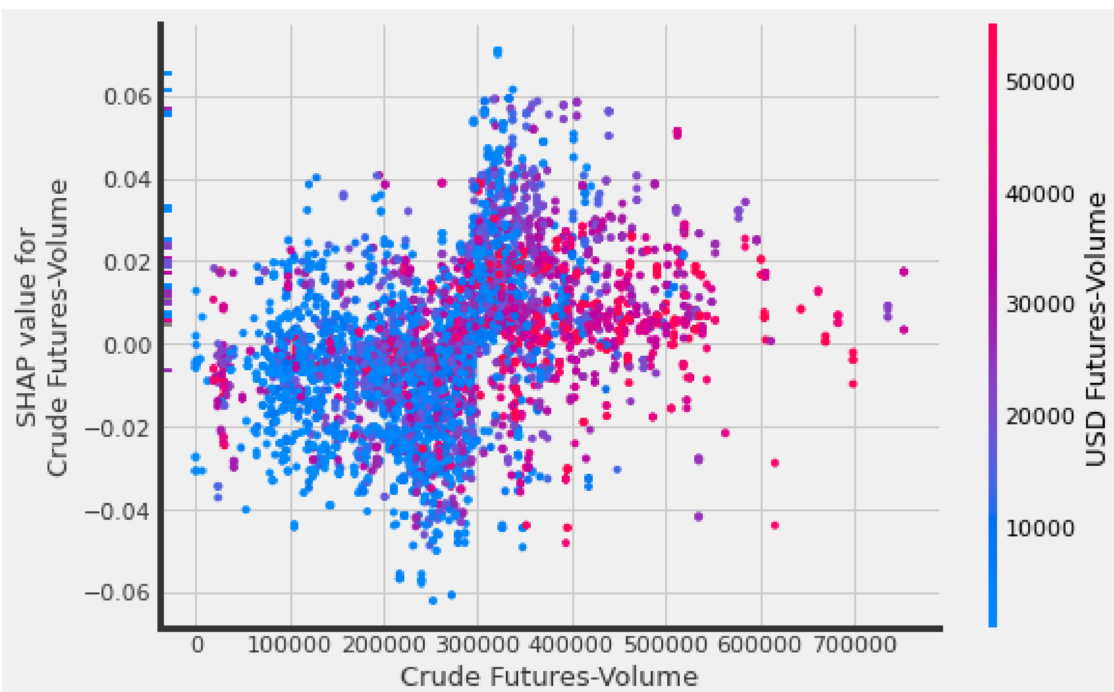


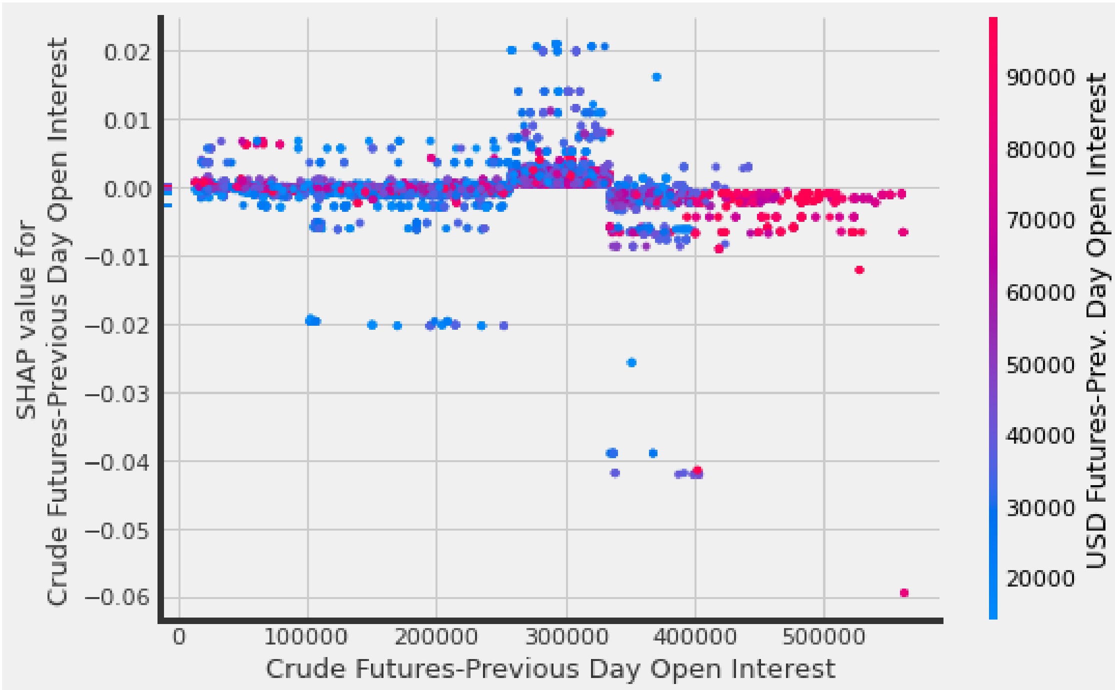
SHAP value for ^FTSE-Close
^FTSE-Close
Market
stock_2
stock_1
currency_2
currency_1
commodity_2
commodity_1
bond_2
bond_1

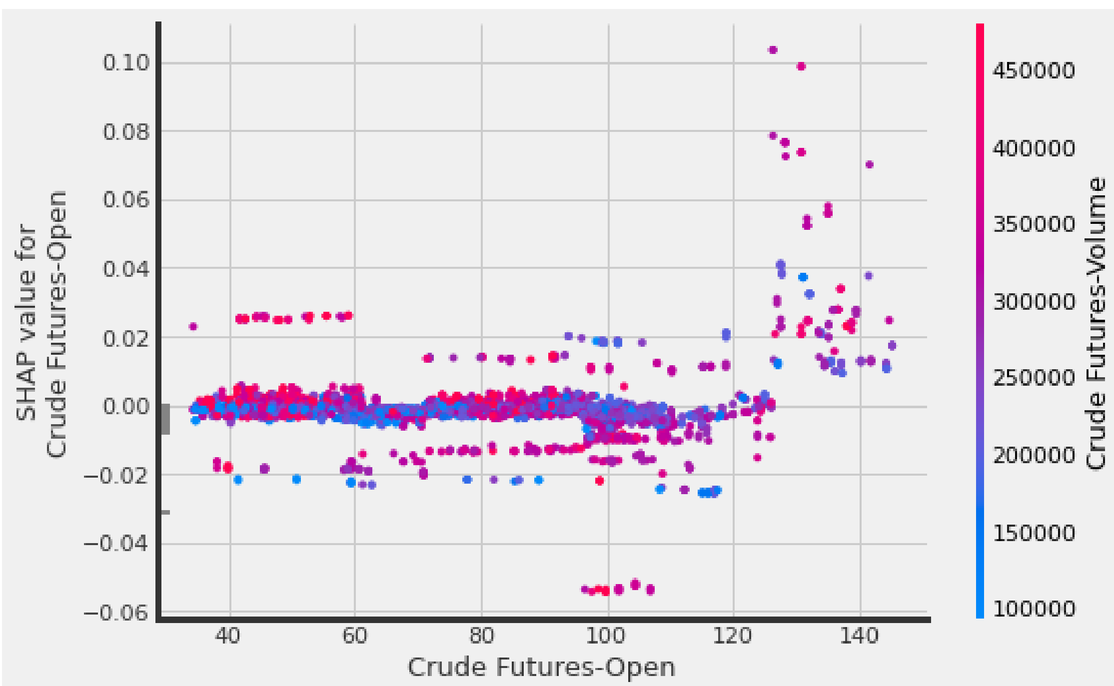

SHAP value for
-TBond Futures-Volume
-TBond Futures-Volume
1e6
Crude Futures-Volume
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
−0.10
0.0
0.2
0.4
0.6
0.8
1.0
1.2
450000
400000
350000
300000
250000
200000
150000
100000

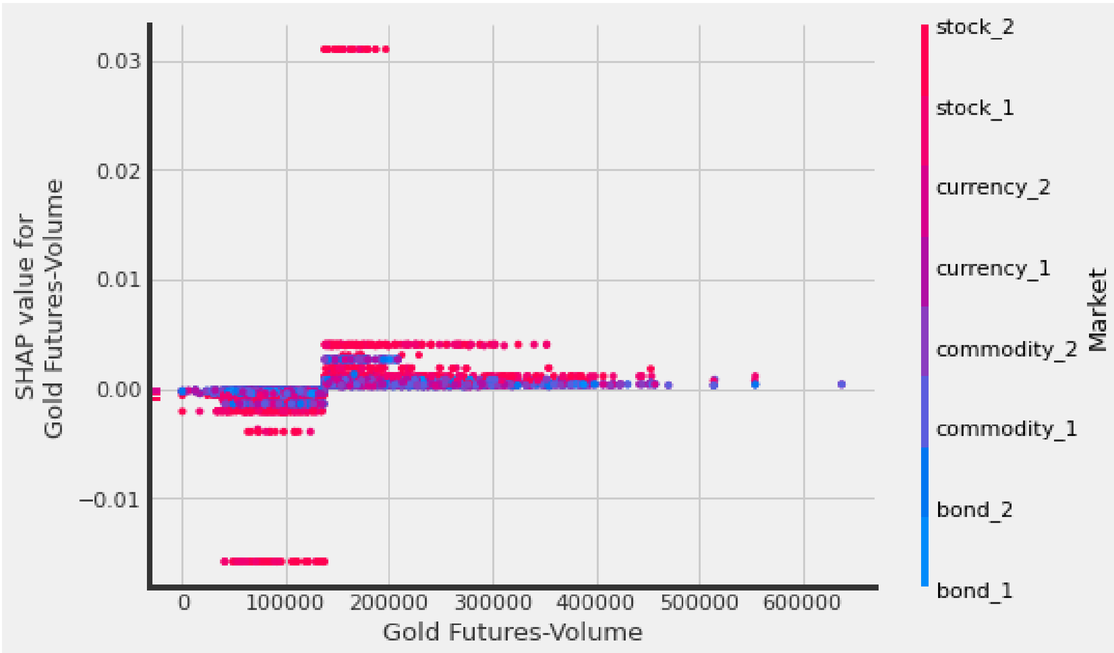

SHAP value for
-TBond Futures-Previous Day Open Interest
-TBond Futures-Previous Day Open Interest
1e6
Month
0.06
0.04
0.02
0.00
−0.02
0.0
0.2
0.4
0.6
0.8
1.0
12
10
8
6
4
2


0.06
0.04
0.02
0.00
−0.02
−0.04
SHAP value for
-TBond Futures-Open
110
120
130
140
150
160
-TBond Futures-Open
110
100
90
80
70
60
50
Gold Futures-Open

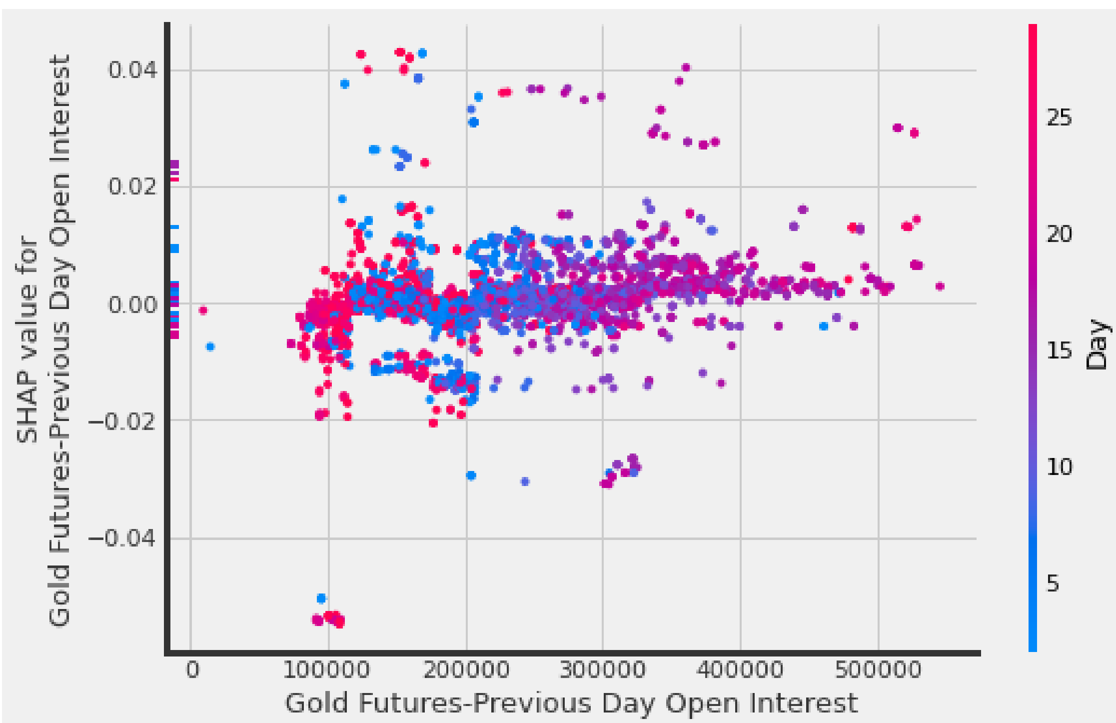

0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
SHAP value for
Crude Futures-Volume
0
100000
200000
300000
400000
500000
600000
700000
Crude Futures-Volume
50000
40000
30000
20000
10000
USD Futures-Volume


SHAP value for
Crude Futures-Previous Day Open Interest
Crude Futures-Previous Day Open Interest
USD Futures-Prev. Day Open Interest
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
−0.05
−0.06
0
100000
200000
300000
400000
500000
90000
80000
70000
60000
50000
40000
30000
20000

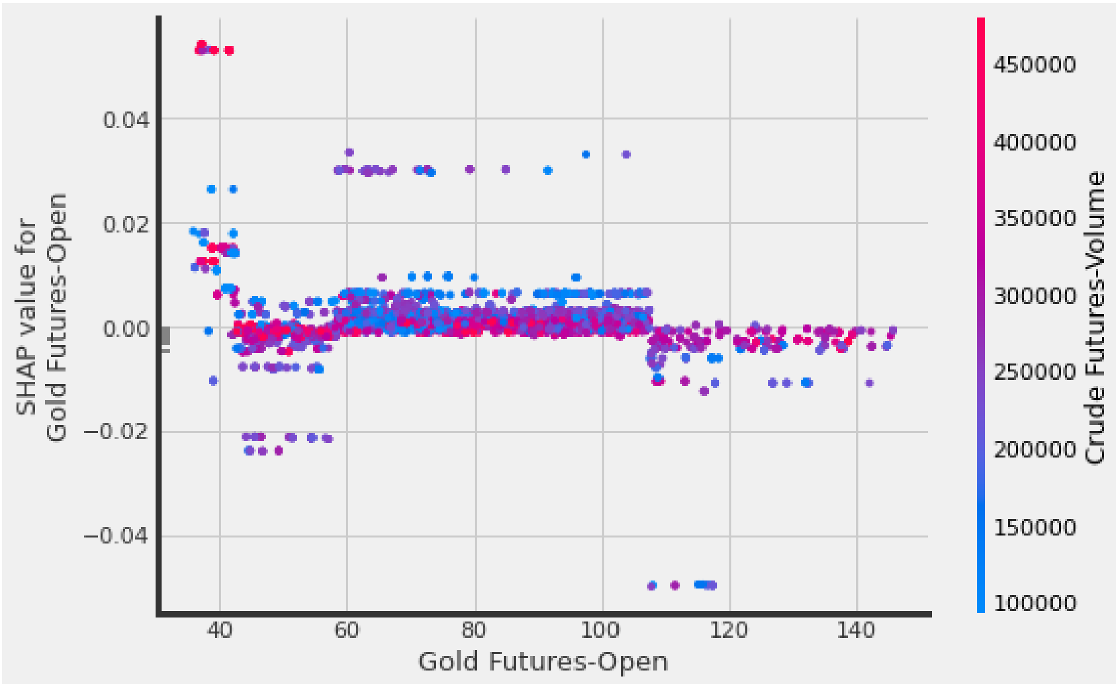

SHAP value for
Crude Futures-Open
Crude Futures-Open
Crude Futures-Volume
0.10
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
40
60
80
100
120
140
450000
400000
350000
300000
250000
200000
150000
100000



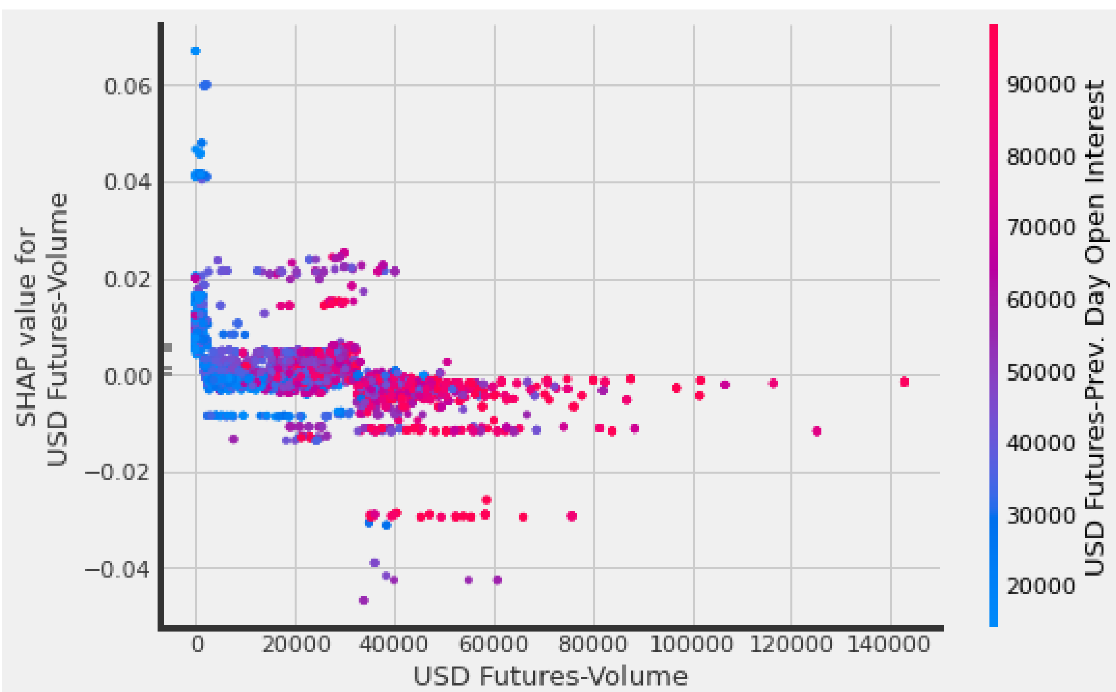
SHAP value for
Gold Futures-Volume
Gold Futures-Volume
0.03
0.02
0.01
0.00
−0.01
0
100000
200000
300000
400000
500000
600000
stock_2
stock_1
currency_2
currency_1
commodity_2
commodity_1
bond_2
bond_1
Market



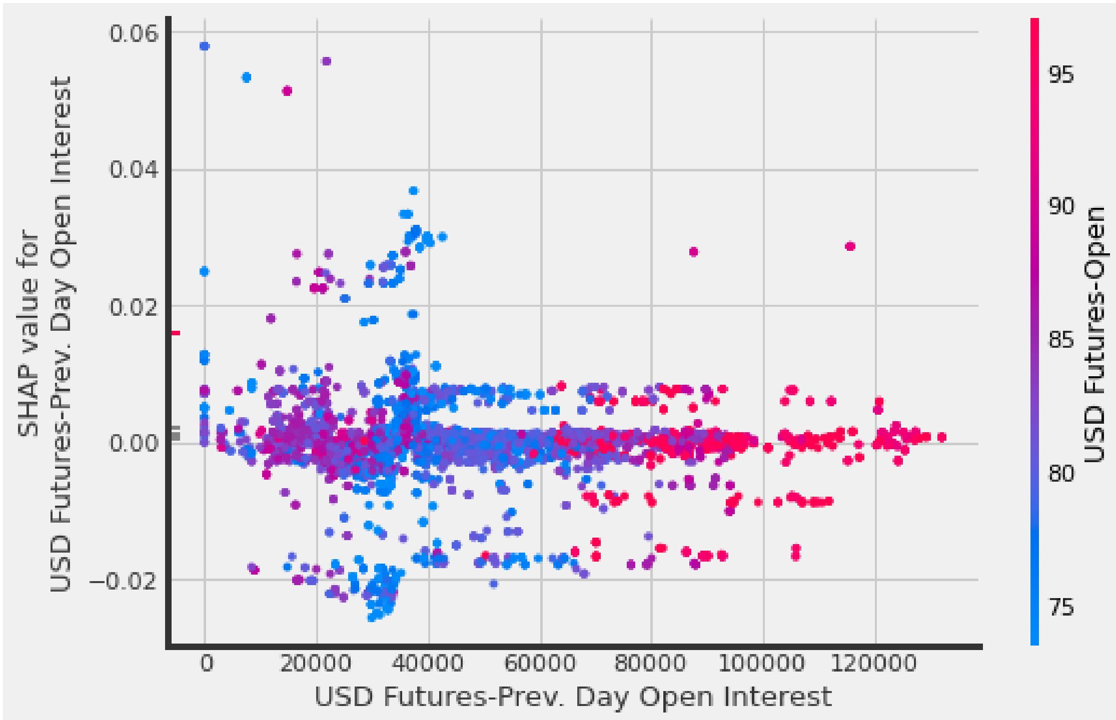
SHAP value for
Gold Futures-Previous Day Open Interest
Gold Futures-Previous Day Open Interest
0.04
0.02
0.00
−0.02
−0.04
0
100000
200000
300000
400000
500000
25
20
15
10
5
Day

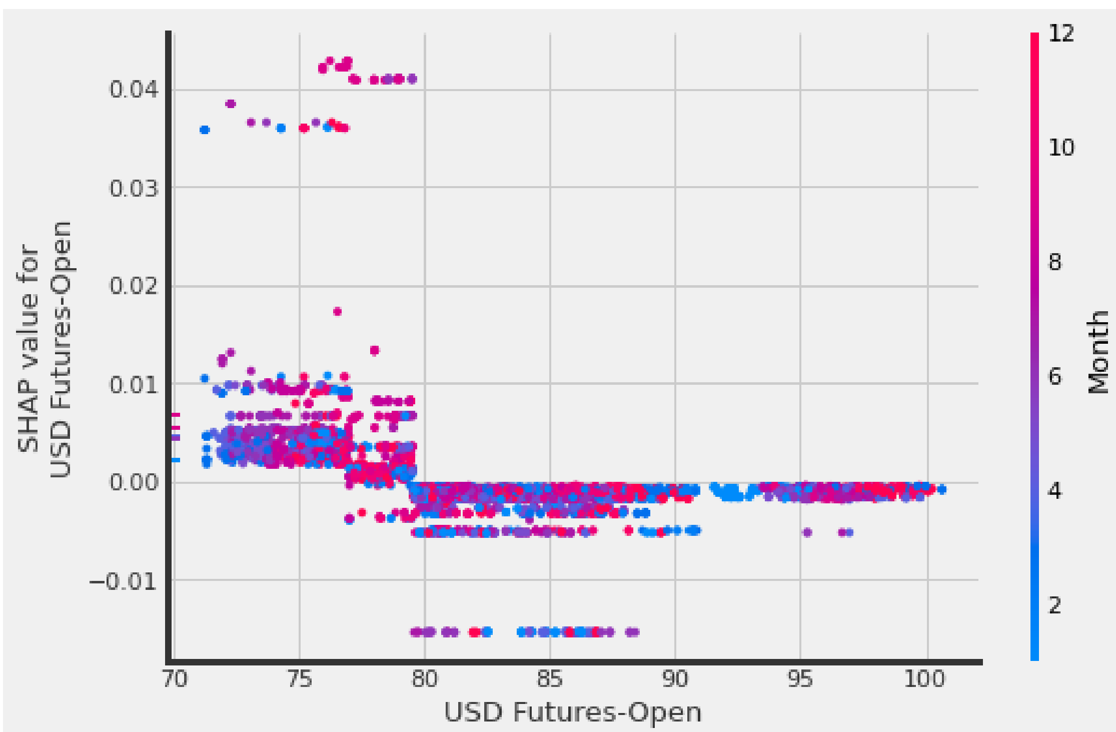


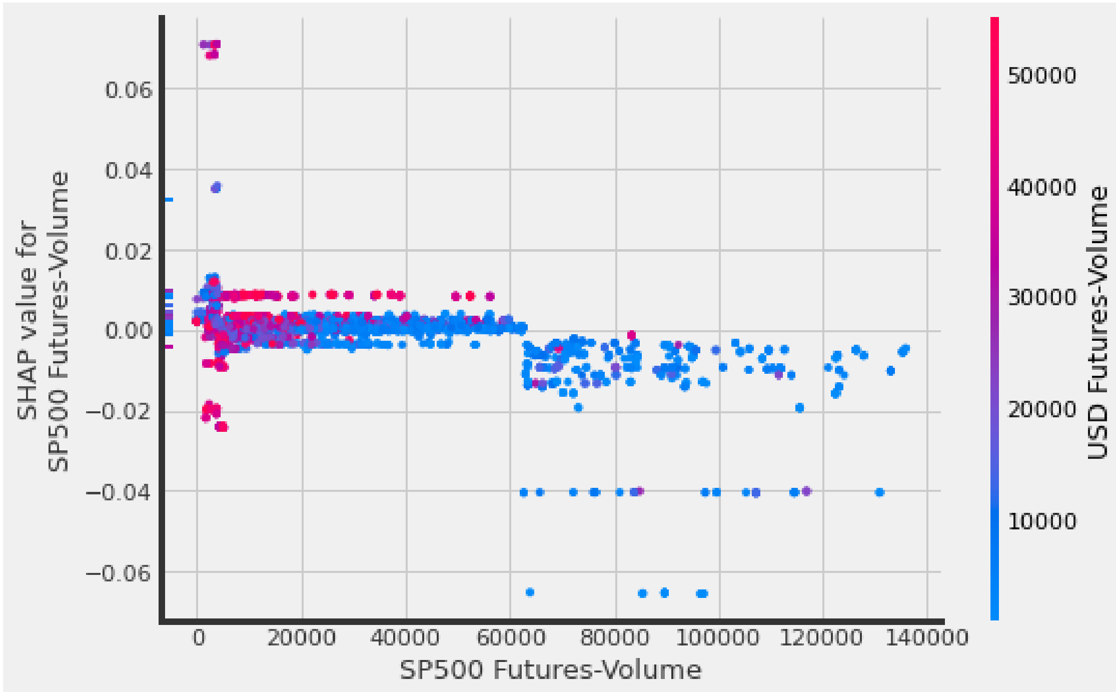
SHAP value for
Gold Futures-Open
0.04
0.02
0.00
−0.02
−0.04
40
60
80
100
120
140
Gold Futures-Open
450000
400000
350000
300000
250000
200000
150000
100000
Crude Futures-Volume



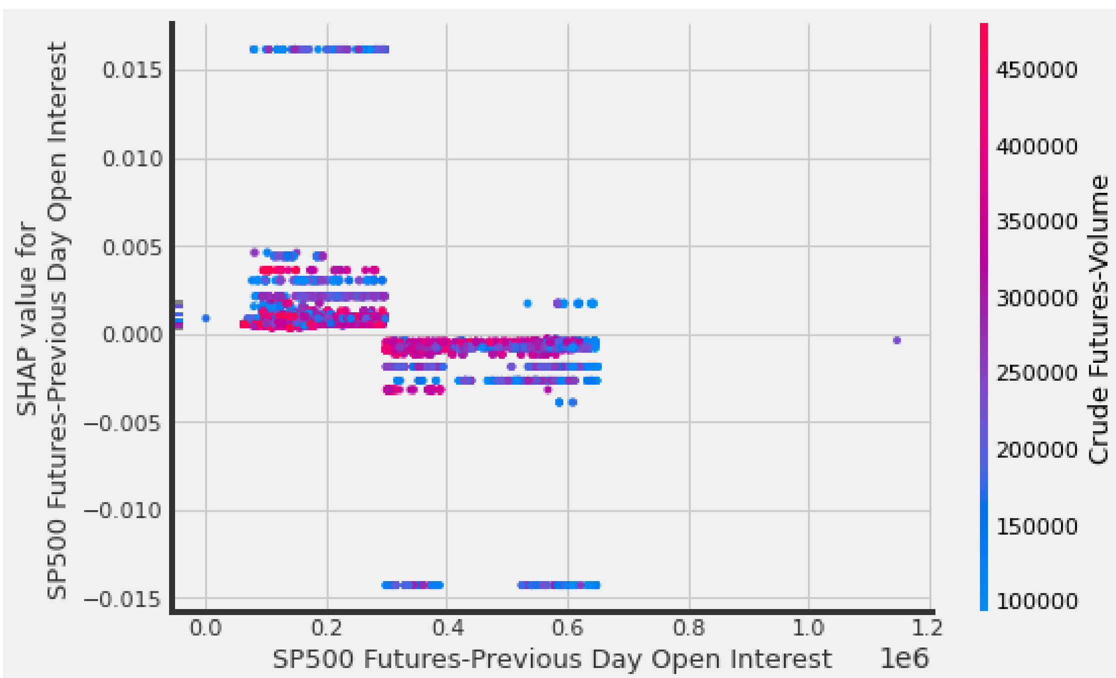
SHAP value for
USD Futures-Volume
0.06
0.04
0.02
0.00
−0.02
−0.04
0
20000
40000
60000
80000
100000
120000
140000
USD Futures-Volume
90000
80000
70000
60000
50000
40000
30000
20000
USD Futures-Prev. Day Open Interest


0.06
0.04
0.02
0.00
−0.02
SHAP value for
USD Futures-Prev. Day Open Interest
0
20000
40000
60000
80000
100000
120000
USD Futures-Prev. Day Open Interest
95
90
85
80
75
USD Futures-Open

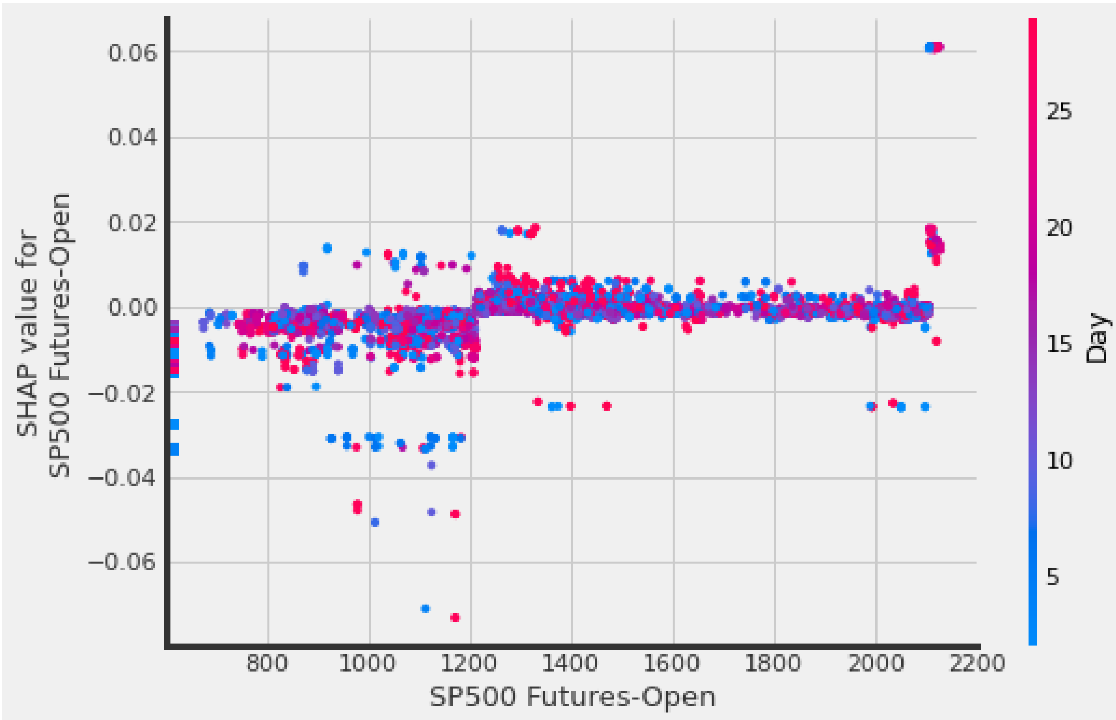

0.04
0.03
0.02
0.01
0.00
−0.01
SHAP value for
USD Futures-Open
70
75
80
85
90
95
100
USD Futures-Open
12
10
8
6
4
2
Month

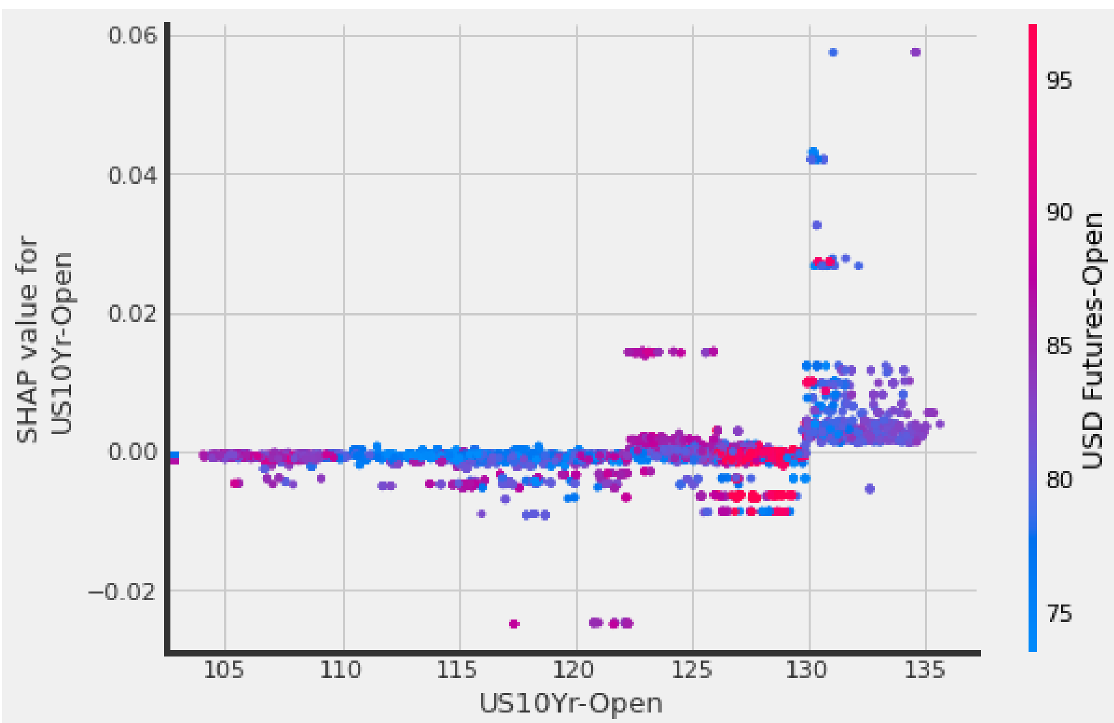


SHAP value for
SP500 Futures-Volume
SP500 Futures-Volume
USD Futures-Volume
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
0
20000
40000
60000
80000
100000
120000
140000
50000
40000
30000
20000
10000



SHAP value for
SP500 Futures-Previous Day Open Interest
SP500 Futures-Previous Day Open Interest
1e6
Crude Futures-Volume
0.015
0.010
0.005
0.000
−0.005
−0.010
−0.015
0.0
0.2
0.4
0.6
0.8
1.0
1.2
450000
400000
350000
300000
250000
200000
150000
100000


SHAP value for
SP500 Futures-Open
SP500 Futures-Open
Day
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
800
1000
1200
1400
1600
1800
2000
2200
25
20
15
10
5


SHAP value for
US10Yr-Open
US10Yr-Open
USD Futures-Open
0.06
0.04
0.02
0.00
−0.02
105
110
115
120
125
130
135
95
90
85
80
75