market_features

December 11, 2019

https://github.com/QuantCS109/TrumpTweets/blob/master/notebooks_features/market_features.ipynb

1 Overview

This notebook uses cleans and aggregates futures market data file 'futures.csv'

We will use the notebook to create market predictors & perform some EDA used in our report(Boxplot, Correlation Matrix, etc.)

[1]:
```
import numpy as np
import pandas as pd
import re
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import scatter_matrix
from sklearn.model_selection import train_test_split
```

[2]:
```
# create futures dataframe
futures_df = pd.read_csv('../data/input_data/futures.csv')
```

[3]:
```
# quick check of df
display(futures_df.head())
display(futures_df.shape)
display(futures_df.dtypes)

assets_key = futures_df['symbol'].unique()
display(len(futures_df['name'].unique()))

# display(len(assets_name))
display(futures_df['name'].unique())
```

```
   Unnamed: 0                                          name symbol  \
0         472  CBOT 10-year US Treasury Note Futures #1 (TY1)     TY
1         473  CBOT 10-year US Treasury Note Futures #1 (TY1)     TY
2         474  CBOT 10-year US Treasury Note Futures #1 (TY1)     TY
3         475  CBOT 10-year US Treasury Note Futures #1 (TY1)     TY
4         476  CBOT 10-year US Treasury Note Futures #1 (TY1)     TY
```

```
         date        open        high         low      settle   volume
0  2015-11-16  123.125000  123.125000  122.812500  122.859375   910514
1  2015-11-17  122.921875  123.031250  122.500000  122.921875  1042810
2  2015-11-18  122.875000  122.968750  122.609375  122.843750   861285
3  2015-11-19  122.796875  123.078125  122.734375  122.968750   939993
4  2015-11-20  122.921875  123.140625  122.828125  122.859375   916842
```

```
(13052, 9)
```

```
Unnamed: 0      int64
name           object
symbol         object
date           object
open          float64
high          float64
low           float64
settle        float64
volume          int64
dtype: object
```

```
13
```

```
array(['CBOT 10-year US Treasury Note Futures #1 (TY1)',
       'CBOT 30-year US Treasury Bond Futures #1 (US1)',
       'CBOT Corn Futures #2 (C2)', 'CBOT Soybeans Futures #2 (S2)',
       'CBOT Wheat Futures #2 (W2)',
       'CME Canadian Dollar CAD Futures #1 (CD1)',
       'CME Euro FX Futures #1 (EC1)',
       'CME Japanese Yen JPY Futures #1 (JY1)',
       'CME Mexican Peso Futures #1 (MP1)',
       'CME NASDAQ 100 Index Mini Futures #1 (NQ1)',
       'CME S&P 500 Index E-Mini Futures #1 (ES1)',
       'NYMEX Gold Futures #1 (GC1)',
       'NYMEX WTI Crude Oil Futures #1 (CL1)'], dtype=object)
```

[4]:
```
# drop reduntant column
futures_df = futures_df.drop(['Unnamed: 0'],axis=1)
display(futures_df.head())
```

```
                                             name symbol        date  \
0  CBOT 10-year US Treasury Note Futures #1 (TY1)     TY  2015-11-16
1  CBOT 10-year US Treasury Note Futures #1 (TY1)     TY  2015-11-17
2  CBOT 10-year US Treasury Note Futures #1 (TY1)     TY  2015-11-18
3  CBOT 10-year US Treasury Note Futures #1 (TY1)     TY  2015-11-19
4  CBOT 10-year US Treasury Note Futures #1 (TY1)     TY  2015-11-20
```

```
         open        high         low      settle   volume
0  123.125000  123.125000  122.812500  122.859375   910514
1  122.921875  123.031250  122.500000  122.921875  1042810
2  122.875000  122.968750  122.609375  122.843750   861285
3  122.796875  123.078125  122.734375  122.968750   939993
4  122.921875  123.140625  122.828125  122.859375   916842
```

[5]:
```python
# to keep new names in case we need them
new_names = []
key_names = futures_df['name'].unique()

for name in key_names:
    clean_name = name[0:name.find('#')]
    new_names.append(clean_name)

symbol_names = {}
for i in range(len(key_names)):
    symbol_names[assets_key[i]] = new_names[i]
print(symbol_names)
```

```
{'TY': 'CBOT 10-year US Treasury Note Futures ', 'US': 'CBOT 30-year US Treasury
Bond Futures ', 'C': 'CBOT Corn Futures ', 'S': 'CBOT Soybeans Futures ', 'W':
'CBOT Wheat Futures ', 'CD': 'CME Canadian Dollar CAD Futures ', 'EC': 'CME Euro
FX Futures ', 'JY': 'CME Japanese Yen JPY Futures ', 'MP': 'CME Mexican Peso
Futures ', 'NQ': 'CME NASDAQ 100 Index Mini Futures ', 'ES': 'CME S&P 500 Index
E-Mini Futures ', 'GC': 'NYMEX Gold Futures ', 'CL': 'NYMEX WTI Crude Oil
Futures '}
```

[6]:
```python
# to initialize new dict of lists
assets_list = futures_df['symbol'].unique()

col_dict = {inst:[key for key in futures_df.columns if re.match(r"{}_+".
 ↪format(inst),key) ]
                for inst in assets_list}

new_col_dict = {inst:[key for key in futures_df.columns if re.match(r"{}_+".
 ↪format(inst),key) ]
                for inst in assets_list}
```

[7]:
```python
# to create new column names by asset info
asset_info = ['open', 'high', 'low', 'settle', 'volume']
print(len(col_dict))

for key in new_col_dict:
    new_col_names = []
    for info in asset_info:
```

```
        new_col_names.append(f'{key}_{info}')
    new_col_dict[key] = new_col_names
    col_dict[key] = asset_info

print(new_col_dict)
```

```
13
{'TY': ['TY_open', 'TY_high', 'TY_low', 'TY_settle', 'TY_volume'], 'US':
['US_open', 'US_high', 'US_low', 'US_settle', 'US_volume'], 'C': ['C_open',
'C_high', 'C_low', 'C_settle', 'C_volume'], 'S': ['S_open', 'S_high', 'S_low',
'S_settle', 'S_volume'], 'W': ['W_open', 'W_high', 'W_low', 'W_settle',
'W_volume'], 'CD': ['CD_open', 'CD_high', 'CD_low', 'CD_settle', 'CD_volume'],
'EC': ['EC_open', 'EC_high', 'EC_low', 'EC_settle', 'EC_volume'], 'JY':
['JY_open', 'JY_high', 'JY_low', 'JY_settle', 'JY_volume'], 'MP': ['MP_open',
'MP_high', 'MP_low', 'MP_settle', 'MP_volume'], 'NQ': ['NQ_open', 'NQ_high',
'NQ_low', 'NQ_settle', 'NQ_volume'], 'ES': ['ES_open', 'ES_high', 'ES_low',
'ES_settle', 'ES_volume'], 'GC': ['GC_open', 'GC_high', 'GC_low', 'GC_settle',
'GC_volume'], 'CL': ['CL_open', 'CL_high', 'CL_low', 'CL_settle', 'CL_volume']}
```

[8]:

```
# to modify original dataframe with new columns renamed
futures_chg = futures_df.copy()
for key_new, value_new in new_col_dict.items():
    for key_old, value_old in col_dict.items():
        if key_new ==  key_old and len(value_new) == len(value_old):
            for i in range(len(value_new)):
                futures_chg[value_new[i]] =␣
 ↪futures_chg[value_old[i]][(futures_chg['symbol'] == key_new)]
```

[9]:

```
# to create list of df with each data and merge
df_list = []
for key, values in new_col_dict.items():
    if 'date' in values:
        values = values
    else:
        values.append('date')

    df_name = f'df_{key}'
    df_name = pd.DataFrame(futures_chg[values].dropna())
    df_name.set_index('date', inplace = True)
    df_name.index = pd.to_datetime(df_name.index)
    df_list.append(df_name)

# merge all individual asset df into a single final
df_merged = pd.concat(df_list, join='outer', axis=1).dropna()
```

[10]:

```
# check shape of new dataframe
display(df_merged.shape)
```

```
display(df_merged.columns)
```

```
(1004, 65)
```

```
Index(['TY_open', 'TY_high', 'TY_low', 'TY_settle', 'TY_volume', 'US_open',
       'US_high', 'US_low', 'US_settle', 'US_volume', 'C_open', 'C_high',
       'C_low', 'C_settle', 'C_volume', 'S_open', 'S_high', 'S_low',
       'S_settle', 'S_volume', 'W_open', 'W_high', 'W_low', 'W_settle',
       'W_volume', 'CD_open', 'CD_high', 'CD_low', 'CD_settle', 'CD_volume',
       'EC_open', 'EC_high', 'EC_low', 'EC_settle', 'EC_volume', 'JY_open',
       'JY_high', 'JY_low', 'JY_settle', 'JY_volume', 'MP_open', 'MP_high',
       'MP_low', 'MP_settle', 'MP_volume', 'NQ_open', 'NQ_high', 'NQ_low',
       'NQ_settle', 'NQ_volume', 'ES_open', 'ES_high', 'ES_low', 'ES_settle',
       'ES_volume', 'GC_open', 'GC_high', 'GC_low', 'GC_settle', 'GC_volume',
       'CL_open', 'CL_high', 'CL_low', 'CL_settle', 'CL_volume'],
      dtype='object')
```

[11]:
```
# create new df to add changes
df_chg = df_merged.copy()
```

[12]:
```
# to group keys for new columns
settle_list = [value[3] for value in new_col_dict.values()]

chg_list = []
for key in new_col_dict:
    chg_list.append(f'{key}_chg')

# info to calculate chg
chg_dict = {}
for i in range(len(settle_list)):
    chg_dict[chg_list[i]] = settle_list[i]
print(chg_dict)
```

```
{'TY_chg': 'TY_settle', 'US_chg': 'US_settle', 'C_chg': 'C_settle', 'S_chg':
'S_settle', 'W_chg': 'W_settle', 'CD_chg': 'CD_settle', 'EC_chg': 'EC_settle',
'JY_chg': 'JY_settle', 'MP_chg': 'MP_settle', 'NQ_chg': 'NQ_settle', 'ES_chg':
'ES_settle', 'GC_chg': 'GC_settle', 'CL_chg': 'CL_settle'}
```

[13]:
```
# to calculate daily returns
start = 0
end = 1
for new_col, col in chg_dict.items():
    df_chg[new_col] = ((df_chg[col].shift(-end) - df_chg[col].shift(-start) ) /␣
 ↪df_chg[col].shift(-start))*100
```

[14]:
```
# check calculated returns make sense
display(df_chg['ES_settle'][:5])
display(df_chg['ES_chg'][:5])
display(df_chg['ES_chg'].head())
display(df_chg['CL_chg'].max())
```

```
date
2015-11-16    2033.25
2015-11-17    2034.25
2015-11-18    2065.00
2015-11-19    2064.50
2015-11-20    2074.00
Name: ES_settle, dtype: float64
```

```
date
2015-11-16    0.049182
2015-11-17    1.511614
2015-11-18   -0.024213
2015-11-19    0.460160
2015-11-20   -0.216972
Name: ES_chg, dtype: float64
```


```
date
2015-11-16    0.049182
2015-11-17    1.511614
2015-11-18   -0.024213
2015-11-19    0.460160
2015-11-20   -0.216972
Name: ES_chg, dtype: float64
```


```
14.356074425392187
```

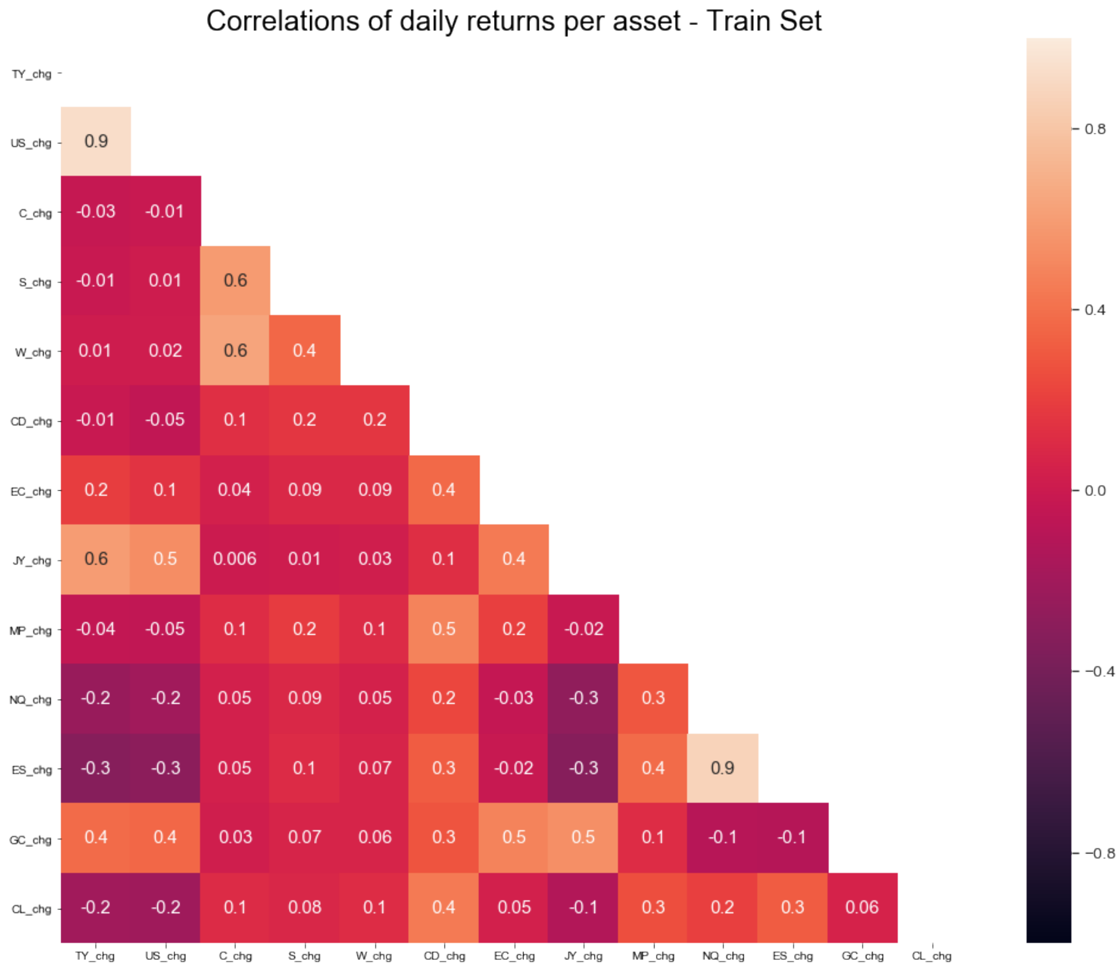
[15]:
```
# correlation matrix for train set
corr = df_chg[['TY_chg',
       'US_chg', 'C_chg', 'S_chg', 'W_chg', 'CD_chg', 'EC_chg', 'JY_chg',
       'MP_chg', 'NQ_chg', 'ES_chg', 'GC_chg', 'CL_chg']].
 ↪corr(method='spearman')

train_chg, test_chg = train_test_split(df_chg[['TY_chg', 'US_chg', 'C_chg',␣
 ↪'S_chg',
                                               'W_chg', 'CD_chg', 'EC_chg',␣
 ↪'JY_chg',
                                               'MP_chg', 'NQ_chg', 'ES_chg',␣
 ↪'GC_chg',
```

```
                                        'CL_chg']], test_size=0.2,␣
↪shuffle=False)
display(corr)

corr_train = train_chg[['TY_chg', 'US_chg', 'C_chg', 'S_chg',
                        'W_chg', 'CD_chg', 'EC_chg', 'JY_chg',
                        'MP_chg', 'NQ_chg', 'ES_chg', 'GC_chg', 'CL_chg']].
↪corr(method='spearman')
```

```
          TY_chg    US_chg     C_chg     S_chg     W_chg    CD_chg    EC_chg  \
TY_chg  1.000000  0.929813 -0.037894 -0.033128 -0.009360 -0.029599  0.185025
US_chg  0.929813  1.000000 -0.021297 -0.017051 -0.001817 -0.066555  0.137872
C_chg  -0.037894 -0.021297  1.000000  0.590967  0.635705  0.123561  0.031685
S_chg  -0.033128 -0.017051  0.590967  1.000000  0.377231  0.165789  0.081151
W_chg  -0.009360 -0.001817  0.635705  0.377231  1.000000  0.154748  0.085512
CD_chg -0.029599 -0.066555  0.123561  0.165789  0.154748  1.000000  0.386823
EC_chg  0.185025  0.137872  0.031685  0.081151  0.085512  0.386823  1.000000
JY_chg  0.615694  0.546382 -0.000670  0.003932  0.010033  0.114173  0.428219
MP_chg -0.052900 -0.067062  0.103884  0.180698  0.090852  0.466071  0.204817
NQ_chg -0.253752 -0.223282  0.050750  0.112248  0.031669  0.237212 -0.027786
ES_chg -0.334398 -0.307023  0.058807  0.133042  0.046745  0.311372 -0.017914
GC_chg  0.427560  0.407059  0.017868  0.058155  0.046955  0.259988  0.457338
CL_chg -0.210198 -0.214595  0.096303  0.101937  0.113049  0.432394  0.049678

          JY_chg    MP_chg    NQ_chg    ES_chg    GC_chg    CL_chg
TY_chg  0.615694 -0.052900 -0.253752 -0.334398  0.427560 -0.210198
US_chg  0.546382 -0.067062 -0.223282 -0.307023  0.407059 -0.214595
C_chg  -0.000670  0.103884  0.050750  0.058807  0.017868  0.096303
S_chg   0.003932  0.180698  0.112248  0.133042  0.058155  0.101937
W_chg   0.010033  0.090852  0.031669  0.046745  0.046955  0.113049
CD_chg  0.114173  0.466071  0.237212  0.311372  0.259988  0.432394
EC_chg  0.428219  0.204817 -0.027786 -0.017914  0.457338  0.049678
JY_chg  1.000000 -0.026159 -0.295784 -0.341942  0.529157 -0.129834
MP_chg -0.026159  1.000000  0.309020  0.376872  0.094373  0.239515
NQ_chg -0.295784  0.309020  1.000000  0.885055 -0.104344  0.208172
ES_chg -0.341942  0.376872  0.885055  1.000000 -0.113819  0.305280
GC_chg  0.529157  0.094373 -0.104344 -0.113819  1.000000  0.024639
CL_chg -0.129834  0.239515  0.208172  0.305280  0.024639  1.000000
```

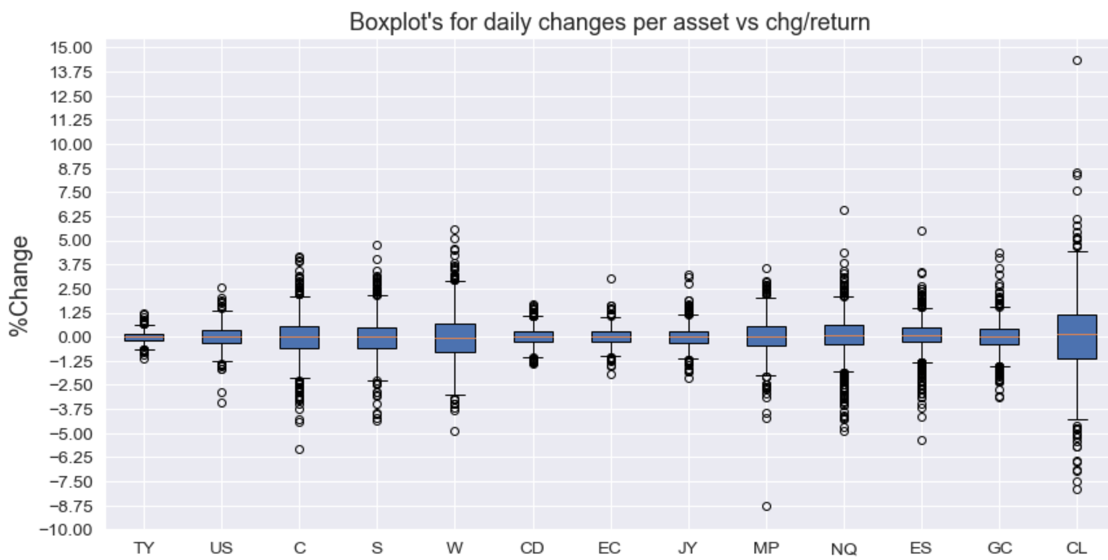
[16]:
```
# correlation heatmap
fig, ax = plt.subplots(1, 1, figsize=(18,14))
# to hide right side
mask = np.triu(corr_train.corr())

sns.set(font_scale=1.25)
```

```
sns.heatmap(corr_train, annot=True, square=True, fmt='.01', vmin=-1, vmax=1,␣
 ↪mask=mask)

b, t = plt.ylim()
b += 0.5 # add 0.5 to the bottom ylim value
t -= 0.5 # subtract 0.5 from the top ylim value
plt.ylim(b, t)
# set plot & labels
ax.set_yticklabels(labels=corr_train.columns, rotation=1)
ax.set_title("Correlations of daily returns per asset - Train Set", fontsize=24)

plt.show()
```



[17]:
```
# boxplot for daily returns
df_chgs = df_chg.dropna()
df_chgs.index = pd.to_datetime(df_chgs.index)
labels = new_col_dict.keys()
```

```
box_plot_data = [df_chgs.TY_chg, df_chgs.US_chg, df_chgs.C_chg, df_chgs.S_chg,
                 df_chgs.W_chg, df_chgs.CD_chg, df_chgs.EC_chg, df_chgs.JY_chg,
                 df_chgs.MP_chg, df_chgs.NQ_chg, df_chgs.ES_chg, df_chgs.GC_chg,
                 df_chgs.CL_chg]

fig, ax = plt.subplots(1, 1, figsize=(14,7))

ax.boxplot(box_plot_data, patch_artist=True, labels=labels)
ax.set_title("Boxplot's for daily changes per asset vs chg/return", fontsize=18)
ax.set_ylabel('%Change', fontsize=18)
plt.yticks(np.arange(-10, 15.25, step=1.25))
plt.show()
```



[18]:
```
# to see top 5 up/down potential outliers
for asset_chg in chg_list:
    # for outliers print
    chg_q3 = np.percentile(df_chgs[asset_chg], 75)
    chg_q1 = np.percentile(df_chgs[asset_chg], 25)
    chg_iqr = chg_q3 - chg_q1
    outliers_up = pd.DataFrame(df_chgs[asset_chg].loc[(df_chgs[asset_chg] >␣
 ↪(chg_q3 + chg_iqr*1.5))])

    outliers_down = pd.DataFrame(df_chgs[asset_chg].loc[(df_chgs[asset_chg] <␣
 ↪(chg_q1 - chg_iqr*1.5))])

    if len(outliers_up) > 5:
        print(outliers_up.sort_values(by=[asset_chg], ascending=False)[0:5])
```

```python
    else:
        print(outliers_up.sort_values(by=[asset_chg], ascending=False))

    print()

    if len(outliers_down) > 5:
        print(outliers_down.sort_values(by=[asset_chg], ascending=True)[0:5])
    else:
        print(outliers_down.sort_values(by=[asset_chg], ascending=True))
    print()
```

```
              TY_chg
date
2016-06-23  1.220554
2018-05-25  1.166105
2019-07-31  0.988166
2016-06-02  0.860691
2019-01-02  0.784413

              TY_chg
date
2016-11-08 -1.150410
2015-12-02 -0.949247
2016-05-17 -0.808724
2016-02-11 -0.752336
2019-09-04 -0.744153

              US_chg
date
2016-06-23  2.529064
2016-06-24  2.029043
2016-02-05  1.883598
2018-05-25  1.837741
2016-02-01  1.724881

              US_chg
date
2016-11-08 -3.401078
2015-12-02 -2.846975
2015-12-28 -1.673734
2016-05-17 -1.650165
2017-02-28 -1.553271

               C_chg
date
2019-10-10  4.150702
```

```
2019-09-27  4.104235
2019-05-24  3.992954
2019-05-29  3.909348
2019-05-23  3.400121

               C_chg
date
2019-08-09 -5.837712
2016-06-20 -4.428698
2019-03-28 -4.313001
2019-06-27 -3.776683
2019-10-09 -3.452528

               S_chg
date
2016-05-09  4.764903
2018-07-05  4.012589
2019-05-13  3.401760
2019-09-11  3.192383
2018-10-31  3.111702

               S_chg
date
2016-07-01 -4.316682
2018-08-09 -4.185351
2016-07-06 -4.020598
2017-07-12 -3.915789
2016-07-29 -3.444654

               W_chg
date
2018-07-24  5.579399
2019-03-11  5.104782
2019-05-29  4.536680
2017-06-29  4.519341
2017-06-30  4.250641

               W_chg
date
2019-08-09 -4.911591
2016-04-21 -3.811734
2017-07-12 -3.654604
2018-06-29 -3.508772
2018-07-10 -3.462051

              CD_chg
date
2016-02-02  1.676400
```

```
2017-11-30  1.603257
2016-01-20  1.519131
2017-07-11  1.475170
2016-03-15  1.440262

              CD_chg
date
2017-01-17 -1.365735
2016-05-02 -1.355431
2015-12-16 -1.349073
2016-06-23 -1.294051
2016-01-14 -1.250966

              EC_chg
date
2015-12-02  3.054289
2016-03-09  1.639960
2016-06-02  1.611437
2016-02-02  1.478370
2017-04-21  1.413166

              EC_chg
date
2016-06-23 -1.910336
2018-06-13 -1.515275
2015-12-16 -1.485066
2016-12-07 -1.259348
2018-08-09 -1.196931

              JY_chg
date
2016-06-23  3.246625
2016-07-28  3.117164
2016-04-27  2.749549
2016-06-02  1.912375
2016-02-02  1.877416

              JY_chg
date
2016-07-08 -2.136953
2016-04-21 -1.808851
2016-07-11 -1.785286
2016-01-28 -1.765703
2016-11-29 -1.651203

              MP_chg
date
2016-02-16  3.530778
```

```
2016-11-14  2.885345
2017-03-02  2.814885
2017-02-17  2.722590
2017-03-14  2.691318

              MP_chg
date
2016-11-08 -8.775779
2016-11-09 -4.208754
2016-06-23 -3.971756
2018-10-26 -3.152873
2019-05-30 -2.916831

              NQ_chg
date
2018-12-24  6.590186
2019-01-03  4.358276
2018-10-15  3.863304
2018-02-05  3.396755
2018-03-23  3.267876

              NQ_chg
date
2018-10-09 -4.893986
2018-02-02 -4.707494
2016-06-23 -4.280447
2018-10-23 -4.198750
2018-12-20 -4.157395

              ES_chg
date
2018-12-24  5.491821
2019-01-03  3.394999
2018-02-05  3.280865
2016-01-28  2.626667
2016-02-29  2.521118

              ES_chg
date
2018-02-02 -5.349129
2016-06-23 -4.118480
2018-10-09 -3.694136
2019-08-02 -3.492334
2018-12-03 -3.172057

              GC_chg
date
2016-06-23  4.371867
```

```
2016-02-10  4.114462
2019-06-19  3.532349
2016-02-05  3.200382
2018-10-10  2.781166

              GC_chg
date
2016-11-10 -3.114136
2016-10-03 -3.075383
2016-12-14 -2.720051
2016-02-12 -2.332187
2015-12-16 -2.314500

               CL_chg
date
2019-09-13  14.356074
2016-11-29   8.523993
2018-12-24   8.405467
2016-02-11   7.558282
2016-01-21   6.092533

              CL_chg
date
2019-07-31 -7.922656
2018-11-21 -7.492436
2018-12-17 -6.980803
2018-11-12 -6.878650
2018-12-21 -6.516184
```

[19]:
```
# to create new predictors names
vol_list = ['TY_volume', 'US_volume',  'C_volume', 'S_volume',
            'W_volume', 'CD_volume', 'EC_volume', 'JY_volume',
            'MP_volume', 'NQ_volume', 'ES_volume', 'GC_volume', 'CL_volume']

vol_chg_dict = {inst:[key for key in futures_df.columns if re.match(r"{}_+".
 ↪format(inst),key) ]
                for inst in assets_list}

for key in vol_chg_dict:
    new_col_avg = f'{key}_volume_avg'
    new_col_op_cl = f'{key}_OP_v_CL'
    vol_chg_dict[key] = [new_col_avg, new_col_op_cl]
```

[20]:
```
# new df to create predictors
df_new_pred = df_chg.copy()
```

```
start = -1
end = 0

count = 0
for col, new_col in vol_chg_dict.items():
    df_new_pred[new_col[0]] = df_new_pred[vol_list[count]].rolling(5,␣
 ↪min_periods=1).mean()
    df_new_pred[f'{col}_volume_chg'] = ((df_new_pred[new_col[0]].shift(-end) -␣
 ↪df_new_pred[new_col[0]].shift(-start)
                                        ) / df_new_pred[new_col[0]].
 ↪shift(-start))*100
    df_new_pred[new_col[1]] = (df_new_pred[f'{col}_open'].shift(-end) -␣
 ↪df_new_pred[f'{col}_settle'].shift(-start))
    df_new_pred[f'{col}_opening'] = [('up' if op_v_cl > 0 else ('down' if␣
 ↪op_v_cl < 0 else 'unch'))
                               for op_v_cl in df_new_pred[new_col[1]]]

    count +=1
```

[21]:
```
# validate new feature make sense
display(df_new_pred[['TY_volume', 'TY_volume_avg',␣
 ↪'TY_volume_chg','TY_OP_v_CL', 'TY_opening', 'TY_chg']].head())
```

date	TY_volume	TY_volume_avg	TY_volume_chg	TY_OP_v_CL	TY_opening
2015-11-16	910514.0	910514.0	NaN	NaN	unch
2015-11-17	1042810.0	976662.0	7.264908	0.062500	up
2015-11-18	861285.0	938203.0	-3.937800	-0.046875	down
2015-11-19	939993.0	938650.5	0.047698	-0.046875	down
2015-11-20	916842.0	934288.8	-0.464678	-0.046875	down

date	TY_chg
2015-11-16	0.050871
2015-11-17	-0.063557
2015-11-18	0.101755
2015-11-19	-0.088945
2015-11-20	0.114460

[22]:
```
# new df for adj predictors
temp_cols = vol_chg_dict.values()

df_preds = df_new_pred.copy()

for cols in temp_cols:
    df_preds = df_preds.drop(cols, axis=1)
```

```
df_preds = df_preds.drop(df_merged.columns, axis=1)
display(df_preds.head())
```

```
              TY_chg    US_chg     C_chg     S_chg     W_chg    CD_chg  \
date
2015-11-16  0.050871  0.435680  0.145773  0.410034 -1.058201  0.130796
2015-11-17 -0.063557  0.000000  0.000000 -0.408359 -0.200535 -0.143688
2015-11-18  0.101755  0.616438  0.436681  0.217077  0.837240  0.327033
2015-11-19 -0.088945 -0.181529 -0.048309 -0.240674 -0.365327 -0.365082
2015-11-20  0.114460  0.363719  0.579990  0.554885  1.166667 -0.222469

              EC_chg    JY_chg    MP_chg    NQ_chg  ...  MP_volume_chg  \
date                                                ...
2015-11-16 -0.248181 -0.112918  0.263745  0.205101  ...            NaN
2015-11-17 -0.017159 -0.113045  0.060704  1.742416  ...      -9.487984
2015-11-18  0.737944  0.520598  1.071790  0.154751  ...      -1.474846
2015-11-19 -0.681431  0.016888  0.720288  0.607746  ...      14.791535
2015-11-20 -0.248714  0.005628 -0.198649 -0.281560  ...       1.550716

            MP_opening  NQ_volume_chg  NQ_opening ES_volume_chg  ES_opening  \
date
2015-11-16        unch            NaN        unch           NaN        unch
2015-11-17          up      -2.480323          up     -2.869230          up
2015-11-18          up      -4.076211          up     -2.553725          up
2015-11-19          up      -4.872332        down     -5.156960        down
2015-11-20        down      -5.892286        down     13.056758        down

           GC_volume_chg  GC_opening CL_volume_chg  CL_opening
date
2015-11-16           NaN        unch           NaN        unch
2015-11-17      8.504745        down    -10.880246          up
2015-11-18     -3.044936          up      7.026649          up
2015-11-19      2.525782          up      7.071820          up
2015-11-20    -14.641963          up      4.785334          up

[5 rows x 39 columns]
```

[23]:
```
# dummies for opening
df_preds = pd.get_dummies(df_preds, columns=['TY_opening', 'US_opening',
                                             'C_opening', 'S_opening',
                                             'W_opening', 'CD_opening',
                                             'EC_opening', 'JY_opening',
                                             'MP_opening', 'NQ_opening',
                                             'ES_opening',␣
 ↪'GC_opening','CL_opening'])
```

[24]:
```
display(df_preds.tail())
```

```
                TY_chg    US_chg     C_chg     S_chg     W_chg    CD_chg  \
date
2019-11-04 -0.506451 -0.999412 -0.444727 -0.420499  0.774818 -0.065742
2019-11-05  0.315113  0.692795 -1.021059 -0.686197  0.240269 -0.197355
2019-11-06 -0.724900 -1.454688 -1.031593  0.850385 -0.814957  0.013183
2019-11-07  0.000000 -0.159585  0.716612 -0.500659 -0.579990 -0.349305
2019-11-08       NaN       NaN       NaN       NaN       NaN       NaN

                EC_chg    JY_chg    MP_chg    NQ_chg  ...  NQ_opening_up  \
date                                                  ...
2019-11-04 -0.573605 -0.552756 -0.096544 -0.060859  ...              1
2019-11-05  0.027043  0.261566  0.231929 -0.042627  ...              1
2019-11-06 -0.216284 -0.369585  0.250675  0.283286  ...              0
2019-11-07 -0.216753  0.147291  0.153876  0.325011  ...              1
2019-11-08       NaN       NaN       NaN       NaN  ...              0

            ES_opening_down  ES_opening_unch  ES_opening_up  GC_opening_down  \
date
2019-11-04                0                0              1                0
2019-11-05                0                0              1                0
2019-11-06                1                0              0                0
2019-11-07                0                0              1                1
2019-11-08                0                0              1                0

            GC_opening_unch  GC_opening_up  CL_opening_down  CL_opening_unch  \
date
2019-11-04                0              1                0                0
2019-11-05                0              1                0                0
2019-11-06                0              1                0                0
2019-11-07                0              0                0                1
2019-11-08                0              1                1                0

            CL_opening_up
date
2019-11-04              1
2019-11-05              1
2019-11-06              1
2019-11-07              0
2019-11-08              0

[5 rows x 65 columns]
```

[25]:
```
# clean nan & only predictos
df_preds = df_preds.dropna()
```

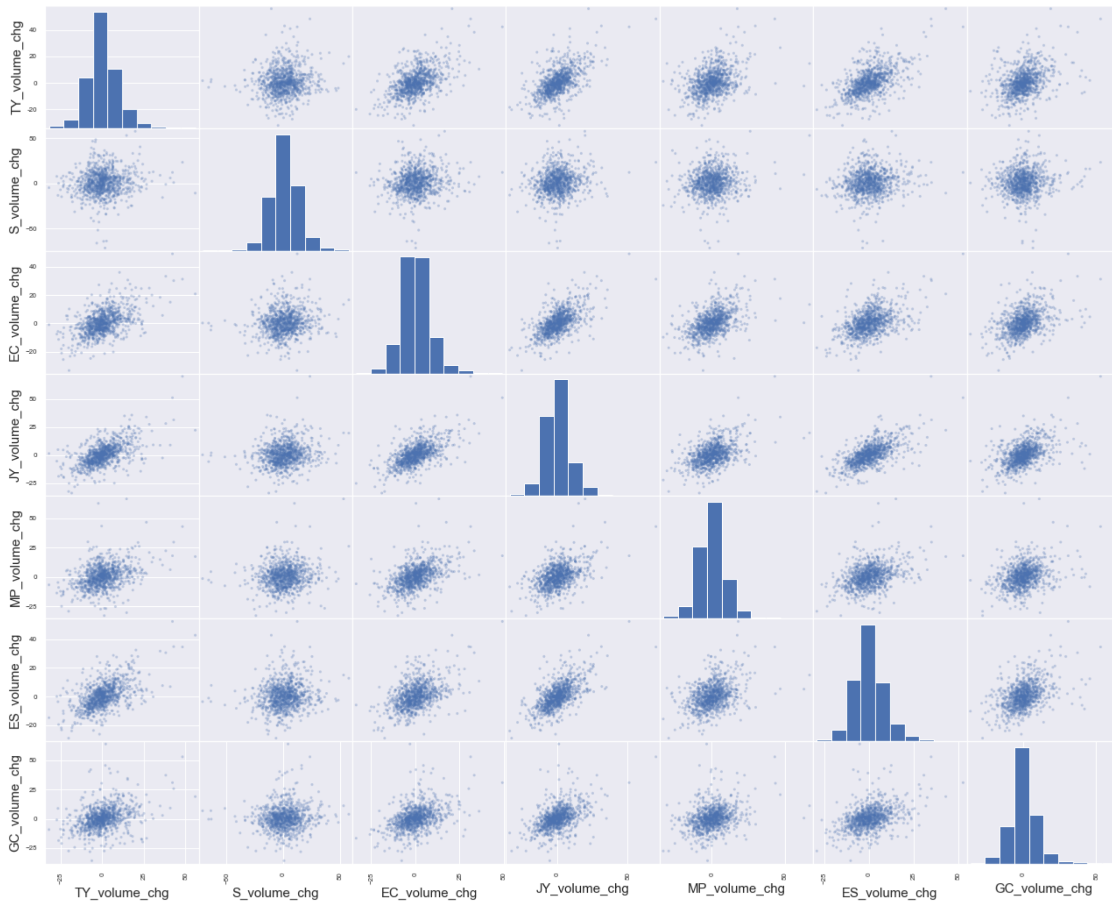
```python
df_preds = df_preds.drop(columns=['TY_chg', 'US_chg', 'C_chg',
                                   'S_chg', 'W_chg', 'CD_chg',
                                   'EC_chg', 'JY_chg', 'MP_chg',
                                   'NQ_chg', 'ES_chg', 'GC_chg',
                                   'CL_chg'], axis=1)
```

[26]:
```python
display(df_preds.head())
display(df_preds.tail())
```

```
            TY_volume_chg  US_volume_chg  C_volume_chg  S_volume_chg  \
date
2015-11-17       7.264908      15.490918     24.506945     -0.360762
2015-11-18      -3.937800      -2.338013     14.887758     48.598815
2015-11-19       0.047698      28.648659      1.152503      2.666122
2015-11-20      -0.464678      -2.639304     -1.979218      6.434186
2015-11-23      15.506875      10.638416      4.286497     14.352737

            W_volume_chg  CD_volume_chg  EC_volume_chg  JY_volume_chg  \
date
2015-11-17    110.707511      -1.582925       2.775035     -25.097245
2015-11-18      7.521267       2.253567       3.006541       8.761078
2015-11-19      0.050234       0.956822       8.022800      -8.684613
2015-11-20     -6.863598       7.343141       1.632575      -9.503479
2015-11-23     15.793635       5.520806       3.223812     -17.977841

            MP_volume_chg  NQ_volume_chg  ...  NQ_opening_up  ES_opening_down  \
date                                      ...
2015-11-17      -9.487984      -2.480323  ...              1                0
2015-11-18      -1.474846      -4.076211  ...              1                0
2015-11-19      14.791535      -4.872332  ...              0                1
2015-11-20       1.550716      -5.892286  ...              0                1
2015-11-23      -3.448337     -10.153613  ...              0                0

            ES_opening_unch  ES_opening_up  GC_opening_down  GC_opening_unch  \
date
2015-11-17                0              1                1                0
2015-11-18                0              1                0                0
2015-11-19                0              0                0                0
2015-11-20                0              0                0                0
2015-11-23                0              1                1                0

            GC_opening_up  CL_opening_down  CL_opening_unch  CL_opening_up
date
2015-11-17              0                0                0              1
2015-11-18              1                0                0              1
2015-11-19              1                0                0              1
2015-11-20              1                0                0              1
```

```
2015-11-23              0               1               0             0

[5 rows x 52 columns]


            TY_volume_chg  US_volume_chg  C_volume_chg  S_volume_chg  \
date
2019-11-01       5.444218       3.559422      2.346276     -7.204511
2019-11-04      -0.249927       0.744727      8.269223     -2.750233
2019-11-05       9.582721       7.085331      2.704347    -11.998904
2019-11-06       0.769120       0.529222      4.371096     -3.834591
2019-11-07       5.457695       4.377160     12.496372      1.452101

            W_volume_chg  CD_volume_chg  EC_volume_chg  JY_volume_chg  \
date
2019-11-01      6.994584       5.248811       8.418831       8.156451
2019-11-04     -3.320639      -1.795057       6.518063       0.180374
2019-11-05      2.290130      -1.928930       4.955569       9.126471
2019-11-06     15.326201     -21.129020      -7.754343      -0.602006
2019-11-07     18.185768      -6.346015      -2.141936       2.863428

            MP_volume_chg  NQ_volume_chg  ...  NQ_opening_up  ES_opening_down  \
date                                      ...
2019-11-01       6.478965       0.135277  ...              1                0
2019-11-04       1.368208       0.815442  ...              1                0
2019-11-05       0.091712      -2.284713  ...              1                0
2019-11-06      -6.454871      -3.647642  ...              0                1
2019-11-07      -5.892271      -4.716510  ...              1                0

            ES_opening_unch  ES_opening_up  GC_opening_down  GC_opening_unch  \
date
2019-11-01                0              1                0                0
2019-11-04                0              1                0                0
2019-11-05                0              1                0                0
2019-11-06                0              0                0                0
2019-11-07                0              1                1                0

            GC_opening_up  CL_opening_down  CL_opening_unch  CL_opening_up
date
2019-11-01              1                1                0              0
2019-11-04              1                0                0              1
2019-11-05              1                0                0              1
2019-11-06              1                0                0              1
2019-11-07              0                0                1              0

[5 rows x 52 columns]
```

[27]:
```
# scatter_matrix for some volume predictors
cor_columns = ['TY_volume_chg', 'S_volume_chg', 'EC_volume_chg',␣
↪'JY_volume_chg',
               'MP_volume_chg', 'ES_volume_chg', 'GC_volume_chg']

scatter_matrix(df_preds.loc[:, cor_columns] , figsize = (22,18), alpha=0.3);
```



[28]:
```
# create csv_file of only predictors
df_preds.to_csv('../data/features/market_features.csv')
```

[ ]: