Main

December 16, 2020

1 Import the necessary packages

1.0.1 Run pip install statsmodels and pip install pandas-datareader if not already installed

[1]:
```
import pandas as pd
import numpy as np
from datetime import date, datetime
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.formula.api import ols
from sklearn.linear_model import LogisticRegression
import pandas_datareader.data as web
```

2 Load data sets & process data

S&P 500 : index that measures the stock performance of 500 large companies listed on stock exchanges in the United States, used as a benchmark. * Average annual return: 10% (6% after inflation) * Traded from 9.30 a.m. to 4 p.m. EST

SSE : index of all stocks traded at the Shanghai Stock Exchange * Traded from 9.30 a.m. to 4 p.m. GMT +8

FXI : track the investment results of the FTSE China 50 Index composed of large-capitalization Chinese equities * Available to international investors * Traded on the Hong Kong Stock Exchange * Traded from 9.30 a.m. to 4 p.m. EST

2.0.1 Load S&P 500, Shanghai Composite Index & iShares China large-cap ETF data from 1/2/2018 to 12/31/2019 (trade war period)

[2]:
```
start_date = '2018-01-02'
end_date = '2019-12-31'

#S&P is traded in EST time: 9.30 a.m. to 4p.m.
spx = web.DataReader('^GSPC', data_source = 'yahoo', start = start_date, end =␣
↪end_date)
spx = spx.reset_index()
```

```
#Shanghai Composite Index is in China time: GMT+8
sse = web.DataReader('000001.SS', data_source = 'yahoo', start = start_date,␣
↪end = end_date)
sse = sse.reset_index()

#FXI: iShares China large-cap ETF, traded in EST
fxi = web.DataReader('FXI', data_source = 'yahoo', start = start_date, end =␣
↪end_date)
fxi = fxi.reset_index()
```

2.0.2 Calculate change in prices between Close (previous day) to Open, then Open - Close (same day)

[3]:
```
spx['CloseOpen'] = np.log(spx['Open']) - np.log(spx['Close'].shift(1))
spx['OpenClose'] = np.log(spx['Close'])- np.log(spx['Open'])
spx = spx.dropna()

sse['CloseOpen'] = np.log(sse['Open']) - np.log(sse['Close'].shift(1))
sse['OpenClose'] = np.log(sse['Close'])- np.log(sse['Open'])
sse = sse.dropna()

fxi['CloseOpen'] = np.log(fxi['Open']) - np.log(fxi['Close'].shift(1))
fxi['OpenClose'] = np.log(fxi['Close'])- np.log(fxi['Open'])
fxi = fxi.dropna()
```

2.0.3 Load lstm balanced data set: data set of Trump's Trade war related tweets with sentiment categorized using LSTM algorithm: 1 being positive and -1 being negative sentiment

The models were trained by a data set by the Crowdflower's Data for Everyone Library. This data set included tweets about the GOP 2016 Debate and each tweet was labeled postive neutral or negative. The data set was highly imbalanced so the majority class was undersampled to yield our LSTM model.

Our code for training the LSTM model is in LSTM_balanced_training and LSTM_unbalanced_training.

Our code for data preprocessing and predicting for the LSTM model is in LSTM_sentiment_analysis_prediction.

[4]:
```
lstm_bal = pd.read_csv('../output/results/LSTM_balanced_spm_results.csv')
lstm_bal['Date'] = [datetime.strptime(x,'%Y-%m-%d') for x in␣
↪lstm_bal['real_Date']]

lstm_sse = pd.read_csv('../output/results/LSTM_balanced_sse_results.csv')
lstm_sse['Date'] = [datetime.strptime(x,'%Y-%m-%d') for x in␣
↪lstm_sse['real_Date']]
```

```
# A is sentiment aggregated over Open-Close, B is  sentiment aggregated over␣
↪Close-Open
```

2.0.4 Load textblob data set: data set of Trump's tweets Trade war related tweets with sentiment categorized using textblob algorithm: 1 being positive, -1 being negative, and 0 being neutral sentiment

The textblob is trained using the textblob models as a sentiment analysis model. We need to do natural language processing before training and predicting the model. After predicting, we save the prediction results in the file textblob_prediction_SPM and textblob_prediction_SSE.

Our code for data preprocessing, training and predicting for the textblob model is in textblob_sentiment_analysis.

[5]:
```
textblob_spm = pd.read_csv('../output/results/textblob_prediction_data_spm.
↪csv', index_col = 0)
textblob_spm['Date'] = [datetime.strptime(x,'%Y-%m-%d') for x in␣
↪textblob_spm['real_Date']]

textblob_sse = pd.read_csv('../output/results/textblob_prediction_data_sse.
↪csv', index_col = 0)
textblob_sse['Date'] = [datetime.strptime(x,'%Y-%m-%d') for x in␣
↪textblob_sse['real_Date']]
# A is sentiment aggregated over Open-Close, B is  sentiment aggregated over␣
↪Close-Open
```

3 Merging financial data with sentiment data

Neutral sentiments are dropped, along with periods of Open-Close or Close-Open with no tweets

3.1 1. Merging S&P 500 with lstm data set

[6]:
```
spx_reg = spx.merge(lstm_bal, how = 'outer', on = ['Date'])

#Split data set into Close-Open and Open-Close
spx_reg_CO = spx_reg[['Date','CloseOpen','B']]
spx_reg_CO = spx_reg_CO.dropna()

spx_reg_OC = spx_reg[['Date','OpenClose','A']]
spx_reg_OC = spx_reg_OC.dropna()

#Sort into Positive and Negative Columns
spx_reg_CO['Positive'] = (spx_reg_CO['B'] == 1)
spx_reg_CO['Negative'] = (spx_reg_CO['B'] == -1)
spx_reg_OC['Positive'] = (spx_reg_OC['A'] == 1)
spx_reg_OC['Negative'] = (spx_reg_OC['A'] == -1)
```

```
spx_reg_CO['Positive'] = [int(x==True) for x in spx_reg_CO['Positive']]
spx_reg_CO['Negative'] = [int(x==True) for x in spx_reg_CO['Negative']]
spx_reg_OC['Positive'] = [int(x==True) for x in spx_reg_OC['Positive']]
spx_reg_OC['Negative'] = [int(x==True) for x in spx_reg_OC['Negative']]
```

3.2 2. Merging S&P 500 with textblob data set

[7]:
```
spx_textblob = spx.merge(textblob_spm, how = 'outer', on = ['Date'])

#Split data set into Close-Open and Open-Close
spx_textblob_CO = spx_textblob[['Date','CloseOpen','B']]
spx_textblob_CO = spx_textblob_CO.dropna()

spx_textblob_OC = spx_textblob[['Date','OpenClose','A']]
spx_textblob_OC = spx_textblob_OC.dropna()

#Sort into Positive and Negative Columns
spx_textblob_CO['Positive'] = (spx_textblob_CO['B'] == 1)
spx_textblob_CO['Negative'] = (spx_textblob_CO['B'] == -1)
spx_textblob_OC['Positive'] = (spx_textblob_OC['A'] == 1)
spx_textblob_OC['Negative'] = (spx_textblob_OC['A'] == -1)
spx_textblob_CO['Positive'] = [int(x==True) for x in␣
↪spx_textblob_CO['Positive']]
spx_textblob_CO['Negative'] = [int(x==True) for x in␣
↪spx_textblob_CO['Negative']]
spx_textblob_OC['Positive'] = [int(x==True) for x in␣
↪spx_textblob_OC['Positive']]
spx_textblob_OC['Negative'] = [int(x==True) for x in␣
↪spx_textblob_OC['Negative']]
```

3.3 3. Merging Shanghai Composite Index with lstm data set

[8]:
```
sse_reg = sse.merge(lstm_sse, how = 'outer', on = ['Date'])
sse_reg_CO = sse_reg[['Date','CloseOpen','B']]
sse_reg_CO = sse_reg_CO.dropna()

sse_reg_OC = sse_reg[['Date','OpenClose','A']]
sse_reg_OC = sse_reg_OC.dropna()

sse_reg_CO['Positive'] = (sse_reg_CO['B'] == 1)
sse_reg_CO['Negative'] = (sse_reg_CO['B'] == -1)
sse_reg_OC['Positive'] = (sse_reg_OC['A'] == 1)
sse_reg_OC['Negative'] = (sse_reg_OC['A'] == -1)
sse_reg_CO['Positive'] = [int(x==True) for x in sse_reg_CO['Positive']]
sse_reg_CO['Negative'] = [int(x==True) for x in sse_reg_CO['Negative']]
sse_reg_OC['Positive'] = [int(x==True) for x in sse_reg_OC['Positive']]
```

```
sse_reg_OC['Negative'] = [int(x==True) for x in sse_reg_OC['Negative']]
```

3.4 4. Merging Shanghai Composite Index with textblob data set

[9]:
```
sse_textblob = sse.merge(textblob_sse, how = 'outer', on = ['Date'])
sse_textblob_CO = sse_textblob[['Date','CloseOpen','B']]
sse_textblob_CO = sse_textblob_CO.dropna()

sse_textblob_OC = sse_textblob[['Date','OpenClose','A']]
sse_textblob_OC = sse_textblob_OC.dropna()

sse_textblob_CO['Positive'] = (sse_textblob_CO['B'] == 1)
sse_textblob_CO['Negative'] = (sse_textblob_CO['B'] == -1)
sse_textblob_OC['Positive'] = (sse_textblob_OC['A'] == 1)
sse_textblob_OC['Negative'] = (sse_textblob_OC['A'] == -1)
sse_textblob_CO['Positive'] = [int(x==True) for x in␣
↪sse_textblob_CO['Positive']]
sse_textblob_CO['Negative'] = [int(x==True) for x in␣
↪sse_textblob_CO['Negative']]
sse_textblob_OC['Positive'] = [int(x==True) for x in␣
↪sse_textblob_OC['Positive']]
sse_textblob_OC['Negative'] = [int(x==True) for x in␣
↪sse_textblob_OC['Negative']]
```

3.5 5. Merging FXI with lstm data set

[10]:
```
fxi_reg = fxi.merge(lstm_bal, how = 'outer', on = ['Date'])
fxi_reg_CO = fxi_reg[['Date','CloseOpen','B']]
fxi_reg_CO = fxi_reg_CO.dropna()

fxi_reg_OC = fxi_reg[['Date','OpenClose','A']]
fxi_reg_OC = fxi_reg_OC.dropna()

fxi_reg_CO['Positive'] = (fxi_reg_CO['B'] == 1)
fxi_reg_CO['Negative'] = (fxi_reg_CO['B'] == -1)
fxi_reg_OC['Positive'] = (fxi_reg_OC['A'] == 1)
fxi_reg_OC['Negative'] = (fxi_reg_OC['A'] == -1)
fxi_reg_CO['Positive'] = [int(x==True) for x in fxi_reg_CO['Positive']]
fxi_reg_CO['Negative'] = [int(x==True) for x in fxi_reg_CO['Negative']]
fxi_reg_OC['Positive'] = [int(x==True) for x in fxi_reg_OC['Positive']]
fxi_reg_OC['Negative'] = [int(x==True) for x in fxi_reg_OC['Negative']]
```

3.6 6. Merging FXI with textblob data set¶

[11]:
```
fxi_textblob = fxi.merge(textblob_spm, how = 'outer', on = ['Date'])
fxi_textblob_CO = fxi_textblob[['Date','CloseOpen','B']]
fxi_textblob_CO = fxi_textblob_CO.dropna()

fxi_textblob_OC = fxi_textblob[['Date','OpenClose','A']]
fxi_textblob_OC = fxi_textblob_OC.dropna()

fxi_textblob_CO['Positive'] = (fxi_textblob_CO['B'] == 1)
fxi_textblob_CO['Negative'] = (fxi_textblob_CO['B'] == -1)
fxi_textblob_OC['Positive'] = (fxi_textblob_OC['A'] == 1)
fxi_textblob_OC['Negative'] = (fxi_textblob_OC['A'] == -1)
fxi_textblob_CO['Positive'] = [int(x==True) for x in␣
 ↪fxi_textblob_CO['Positive']]
fxi_textblob_CO['Negative'] = [int(x==True) for x in␣
 ↪fxi_textblob_CO['Negative']]
fxi_textblob_OC['Positive'] = [int(x==True) for x in␣
 ↪fxi_textblob_OC['Positive']]
fxi_textblob_OC['Negative'] = [int(x==True) for x in␣
 ↪fxi_textblob_OC['Negative']]
```

4 Run Regression & Calculate Point-biserial correlation coefficient on merged data set

4.0.1 Point-biserial correlation coefficient:

The point biserial correlation coefficient (rpb) is a correlation coefficient used when one variable (e.g. Y) is dichotomous. The point-biserial correlation is mathematically equivalent to the Pearson (product moment) correlation, that is, if we have one continuously measured variable X and a dichotomous variable Y, rXY = rpb.

$$r_{pb} = \frac{M_1 - M_0}{S_n} * \sqrt{\frac{n_1 * n_0}{n^2}}$$

4.1 1. S&P 500 vs. lstm sentiment

4.1.1 a. Regression

Close-Open

[12]:
```
fit_spx_CO = ols('CloseOpen ~ C(Positive)', data=spx_reg_CO).fit()
print(fit_spx_CO.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              CloseOpen   R-squared:                       0.001
Model:                            OLS   Adj. R-squared:                 -0.003
```

```
Method:                 Least Squares   F-statistic:                    0.2204
Date:                Wed, 16 Dec 2020   Prob (F-statistic):              0.639
Time:                        18:19:14   Log-Likelihood:                 1098.2
No. Observations:                 271   AIC:                            -2192.
Df Residuals:                     269   BIC:                            -2185.
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
====
                     coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
----
Intercept          0.0004      0.000      1.198      0.232      -0.000
0.001
C(Positive)[T.1]  -0.0002      0.001     -0.470      0.639      -0.001
0.001
==============================================================================
Omnibus:                       28.822   Durbin-Watson:                   2.144
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               42.000
Skew:                          -0.682   Prob(JB):                     7.58e-10
Kurtosis:                       4.363   Cond. No.                         2.69
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

Open-Close

[13]:
```python
fit_spx_OC = ols('OpenClose ~ C(Positive)', data=spx_reg_OC).fit()
print(fit_spx_OC.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              OpenClose   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                 -0.008
Method:                 Least Squares   F-statistic:                   0.05105
Date:                Wed, 16 Dec 2020   Prob (F-statistic):              0.822
Time:                        18:19:15   Log-Likelihood:                 421.03
No. Observations:                 125   AIC:                            -838.1
Df Residuals:                     123   BIC:                            -832.4
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
====
                     coef    std err          t      P>|t|      [0.025
0.975]
```

```
--------------------------------------------------------------------------------
----
Intercept            -0.0008      0.001     -0.632      0.529      -0.003
0.002
C(Positive)[T.1]     -0.0003      0.002     -0.226      0.822      -0.003
0.003
==============================================================================
Omnibus:                       34.583   Durbin-Watson:                   2.273
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               62.282
Skew:                          -1.232   Prob(JB):                     2.99e-14
Kurtosis:                       5.427   Cond. No.                         2.99
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

4.1.2 b. Calculate Point-biserial correlation coefficient

[14]:
```
x = spx_reg_CO['CloseOpen']
y = spx_reg_CO['Positive']
std_x = np.std(x)
M1 = np.mean(spx_reg_CO[spx_reg_CO['Positive']==1]['CloseOpen'])
n1 = spx_reg_CO[spx_reg_CO['Positive']==1].shape[0]
M0 = np.mean(spx_reg_CO[spx_reg_CO['Negative']==1]['CloseOpen'])
n0 = spx_reg_CO[spx_reg_CO['Negative']==1].shape[0]
n = n1+n0
rpb = (M1-M0)*np.sqrt(n1*n0/(n*(n-1)))/std_x
print('Point-biserial correlation coefficient for Close-Open is:', rpb)
```

```
Point-biserial correlation coefficient for Close-Open is: -0.004854826222243692
```

[15]:
```
x = spx_reg_OC['OpenClose']
y = spx_reg_OC['Positive']
std_x = np.std(x)
M1 = np.mean(spx_reg_OC[spx_reg_OC['Positive']==1]['OpenClose'])
n1 = spx_reg_OC[spx_reg_OC['Positive']==1].shape[0]
M0 = np.mean(spx_reg_OC[spx_reg_OC['Negative']==1]['OpenClose'])
n0 = spx_reg_OC[spx_reg_OC['Negative']==1].shape[0]
n = n1+n0
rpb = (M1-M0)*np.sqrt(n1*n0/(n*(n-1)))/std_x
print('Point-biserial correlation coefficient for Open-Close is:', rpb)
```

```
Point-biserial correlation coefficient for Open-Close is: -0.019320563119210717
```

4.2 2. S&P 500 vs. textblob sentiment

4.2.1 a. Regression

Close-Open

[16]:
```python
fit_spx_textblob_CO = ols('CloseOpen ~ C(Positive)', data=spx_textblob_CO).fit()
print(fit_spx_textblob_CO.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              CloseOpen   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                 -0.004
Method:                 Least Squares   F-statistic:                   0.06211
Date:                Wed, 16 Dec 2020   Prob (F-statistic):              0.803
Time:                        18:19:23   Log-Likelihood:                 909.04
No. Observations:                 227   AIC:                            -1814.
Df Residuals:                     225   BIC:                            -1807.
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
====
                       coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
----
Intercept            0.0002      0.000      0.486      0.627      -0.001
0.001
C(Positive)[T.1]    -0.0001      0.001     -0.249      0.803      -0.001
0.001
==============================================================================
Omnibus:                       20.040   Durbin-Watson:                   2.252
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               25.740
Skew:                          -0.616   Prob(JB):                     2.57e-06
Kurtosis:                       4.096   Cond. No.                         2.84
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

Open-Close

[17]:
```python
fit_spx_textblob_OC = ols('OpenClose ~ C(Positive)', data=spx_textblob_OC).fit()
print(fit_spx_textblob_OC.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              OpenClose   R-squared:                       0.005
Model:                            OLS   Adj. R-squared:                 -0.005
```

```
Method:                 Least Squares   F-statistic:                    0.5259
Date:                Wed, 16 Dec 2020   Prob (F-statistic):              0.470
Time:                        18:19:27   Log-Likelihood:                 324.17
No. Observations:                  98   AIC:                            -644.3
Df Residuals:                      96   BIC:                            -639.2
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
====
                     coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
----
Intercept         -0.0003      0.001     -0.237      0.813      -0.003
0.002
C(Positive)[T.1]  -0.0013      0.002     -0.725      0.470      -0.005
0.002
==============================================================================
Omnibus:                       26.030   Durbin-Watson:                   2.177
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               40.024
Skew:                          -1.183   Prob(JB):                     2.04e-09
Kurtosis:                       5.051   Cond. No.                         2.89
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

4.2.2 b. Calculate Point-biserial correlation coefficient

[18]:
```python
x = spx_textblob_CO['CloseOpen']
y = spx_textblob_CO['Positive']
std_x = np.std(x)
M1 = np.mean(spx_textblob_CO[spx_textblob_CO['Positive']==1]['CloseOpen'])
n1 = spx_textblob_CO[spx_textblob_CO['Positive']==1].shape[0]
M0 = np.mean(spx_textblob_CO[spx_textblob_CO['Negative']==1]['CloseOpen'])
n0 = spx_textblob_CO[spx_textblob_CO['Negative']==1].shape[0]
n = n1+n0
rpb = (M1-M0)*np.sqrt(n1*n0/(n*(n-1)))/std_x
print('Point-biserial correlation coefficient for Close-Open is:', rpb)
```

```
Point-biserial correlation coefficient for Close-Open is: 0.027757194506961l5
```

[19]:
```python
x = spx_textblob_OC['OpenClose']
y = spx_textblob_OC['Positive']
std_x = np.std(x)
M1 = np.mean(spx_textblob_OC[spx_textblob_OC['Positive']==1]['OpenClose'])
n1 = spx_textblob_OC[spx_textblob_OC['Positive']==1].shape[0]
```

```
M0 = np.mean(spx_textblob_OC[spx_textblob_OC['Negative']==1]['OpenClose'])
n0 = spx_textblob_OC[spx_textblob_OC['Negative']==1].shape[0]
n = n1+n0
rpb = (M1-M0)*np.sqrt(n1*n0/(n*(n-1)))/std_x
print('Point-biserial correlation coefficient for Open-Close is:', rpb)
```

```
Point-biserial correlation coefficient for Open-Close is: 0.013366760182372414
```

4.3 3. Shanghai Composite Index vs. lstm

4.3.1 a. Regression

Close-Open

[20]:
```
fit_sse_CO = ols('CloseOpen ~ C(Positive)', data=sse_reg_CO).fit()
print(fit_sse_CO.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              CloseOpen   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                 -0.006
Method:                 Least Squares   F-statistic:                   0.02554
Date:                Wed, 16 Dec 2020   Prob (F-statistic):              0.873
Time:                        18:19:43   Log-Likelihood:                 626.36
No. Observations:                 170   AIC:                            -1249.
Df Residuals:                     168   BIC:                            -1242.
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
====
                       coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
----
Intercept           -0.0010      0.001     -1.299      0.196      -0.003
0.001
C(Positive)[T.1]     0.0002      0.001      0.160      0.873      -0.002
0.002
==============================================================================
Omnibus:                       51.207   Durbin-Watson:                   1.848
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              292.448
Skew:                          -0.939   Prob(JB):                     3.13e-64
Kurtosis:                       9.145   Cond. No.                         3.10
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

Open-Close

[21]:
```
fit_sse_OC = ols('OpenClose ~ C(Positive)', data=sse_reg_OC).fit()
print(fit_sse_OC.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              OpenClose   R-squared:                       0.003
Model:                            OLS   Adj. R-squared:                 -0.002
Method:                 Least Squares   F-statistic:                    0.6628
Date:                Wed, 16 Dec 2020   Prob (F-statistic):              0.416
Time:                        18:19:44   Log-Likelihood:                 703.43
No. Observations:                 221   AIC:                            -1403.
Df Residuals:                     219   BIC:                            -1396.
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
====
                       coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
----
Intercept            0.0020      0.001      2.070      0.040    9.62e-05
0.004
C(Positive)[T.1]   -0.0011      0.001     -0.814      0.416      -0.004
0.002
==============================================================================
Omnibus:                        3.785   Durbin-Watson:                   2.043
Prob(Omnibus):                  0.151   Jarque-Bera (JB):                3.935
Skew:                           0.159   Prob(JB):                        0.140
Kurtosis:                       3.571   Cond. No.                         2.66
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

4.3.2 b. Calculate Point-biserial correlation coefficient

[22]:
```
x = sse_reg_CO['CloseOpen']
y = sse_reg_CO['Positive']
std_x = np.std(x)
M1 = np.mean(sse_reg_CO[sse_reg_CO['Positive']==1]['CloseOpen'])
n1 = sse_reg_CO[sse_reg_CO['Positive']==1].shape[0]
M0 = np.mean(sse_reg_CO[sse_reg_CO['Negative']==1]['CloseOpen'])
n0 = sse_reg_CO[sse_reg_CO['Negative']==1].shape[0]
n = n1+n0
rpb = (M1-M0)*np.sqrt(n1*n0/(n*(n-1)))/std_x
```

```
print('Point-biserial correlation coefficient for Close-Open is:', rpb)
```

```
Point-biserial correlation coefficient for Close-Open is: 0.018178598827927814
```

[23]:
```
x = sse_reg_OC['OpenClose']
y = sse_reg_OC['Positive']
std_x = np.std(x)
M1 = np.mean(sse_reg_OC[sse_reg_OC['Positive']==1]['OpenClose'])
n1 = sse_reg_OC[sse_reg_OC['Positive']==1].shape[0]
M0 = np.mean(sse_reg_OC[sse_reg_OC['Negative']==1]['OpenClose'])
n0 = sse_reg_OC[sse_reg_OC['Negative']==1].shape[0]
n = n1+n0
rpb = (M1-M0)*np.sqrt(n1*n0/(n*(n-1)))/std_x
print('Point-biserial correlation coefficient for Open-Close is:', rpb)
```

```
Point-biserial correlation coefficient for Open-Close is: -0.07985055685344604
```

4.4 4. Shanghai Composite Index vs. textblob sentiment

4.4.1 a. Regression

Close-Open

[24]:
```
fit_sse_textblob_CO = ols('CloseOpen ~ C(Positive)', data=sse_textblob_CO).fit()
print(fit_sse_textblob_CO.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              CloseOpen   R-squared:                       0.035
Model:                            OLS   Adj. R-squared:                  0.027
Method:                 Least Squares   F-statistic:                     4.659
Date:                Wed, 16 Dec 2020   Prob (F-statistic):             0.0327
Time:                        18:19:49   Log-Likelihood:                 489.99
No. Observations:                 132   AIC:                            -976.0
Df Residuals:                     130   BIC:                            -970.2
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
====
                       coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
----
Intercept            0.0006      0.001      0.702      0.484      -0.001
0.002
C(Positive)[T.1]    -0.0023      0.001     -2.159      0.033      -0.004
-0.000
==============================================================================
Omnibus:                       11.255   Durbin-Watson:                   2.009
```

```
Prob(Omnibus):                  0.004   Jarque-Bera (JB):               28.330
Skew:                          -0.130   Prob(JB):                     7.05e-07
Kurtosis:                       5.255   Cond. No.                         2.95
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

Open-Close

```
[25]: fit_sse_textblob_OC = ols('OpenClose ~ C(Positive)', data=sse_textblob_OC).fit()
      print(fit_sse_textblob_OC.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              OpenClose   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                 -0.005
Method:                 Least Squares   F-statistic:                   0.02868
Date:                Wed, 16 Dec 2020   Prob (F-statistic):              0.866
Time:                        18:19:50   Log-Likelihood:                 585.38
No. Observations:                 182   AIC:                            -1167.
Df Residuals:                     180   BIC:                            -1160.
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
====
                       coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
----
Intercept            0.0016      0.001      1.427      0.155      -0.001
0.004
C(Positive)[T.1]    -0.0002      0.001     -0.169      0.866      -0.003
0.003
==============================================================================
Omnibus:                        1.398   Durbin-Watson:                   1.933
Prob(Omnibus):                  0.497   Jarque-Bera (JB):                1.064
Skew:                           0.025   Prob(JB):                        0.587
Kurtosis:                       3.371   Cond. No.                         2.86
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

4.4.2 b. Calculate Point-biserial correlation coefficient

[26]:
```
x = sse_textblob_CO['CloseOpen']
y = sse_textblob_CO['Positive']
std_x = np.std(x)
M1 = np.mean(sse_textblob_CO[sse_textblob_CO['Positive']==1]['CloseOpen'])
n1 = sse_textblob_CO[sse_textblob_CO['Positive']==1].shape[0]
M0 = np.mean(sse_textblob_CO[sse_textblob_CO['Negative']==1]['CloseOpen'])
n0 = sse_textblob_CO[sse_textblob_CO['Negative']==1].shape[0]
n = n1+n0
rpb = (M1-M0)*np.sqrt(n1*n0/(n*(n-1)))/std_x
print('Point-biserial correlation coefficient for Close-Open is:', rpb)
```

```
Point-biserial correlation coefficient for Close-Open is: -0.07790471887144577
```

[27]:
```
x = sse_textblob_OC['OpenClose']
y = sse_textblob_OC['Positive']
std_x = np.std(x)
M1 = np.mean(sse_textblob_OC[sse_textblob_OC['Positive']==1]['OpenClose'])
n1 = sse_textblob_OC[sse_textblob_OC['Positive']==1].shape[0]
M0 = np.mean(sse_textblob_OC[sse_textblob_OC['Negative']==1]['OpenClose'])
n0 = sse_textblob_OC[sse_textblob_OC['Negative']==1].shape[0]
n = n1+n0
rpb = (M1-M0)*np.sqrt(n1*n0/(n*(n-1)))/std_x
print('Point-biserial correlation coefficient for Open-Close is:', rpb)
```

```
Point-biserial correlation coefficient for Open-Close is: -0.02791541200934203
```

4.5 5. FXI vs. lstm

4.5.1 a. Regression

Close-Open

[28]:
```
fit_fxi_CO = ols('CloseOpen ~ C(Positive)', data=fxi_reg_CO).fit()
print(fit_fxi_CO.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              CloseOpen   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                 -0.004
Method:                 Least Squares   F-statistic:                   0.03736
Date:                Wed, 16 Dec 2020   Prob (F-statistic):              0.847
Time:                        18:19:53   Log-Likelihood:                 813.27
No. Observations:                 271   AIC:                            -1623.
Df Residuals:                     269   BIC:                            -1615.
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
```

```
====
                        coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
----
Intercept             0.0002      0.001      0.182      0.856      -0.002
0.002
C(Positive)[T.1]      0.0003      0.001      0.193      0.847      -0.003
0.003
==============================================================================
Omnibus:                       10.804   Durbin-Watson:                   2.108
Prob(Omnibus):                  0.005   Jarque-Bera (JB):               11.157
Skew:                          -0.433   Prob(JB):                      0.00378
Kurtosis:                       3.489   Cond. No.                         2.69
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

Open-Close

[29]:
```python
fit_fxi_OC = ols('OpenClose ~ C(Positive)', data=fxi_reg_OC).fit()
print(fit_fxi_OC.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              OpenClose   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                 -0.008
Method:                 Least Squares   F-statistic:                 5.350e-07
Date:                Wed, 16 Dec 2020   Prob (F-statistic):              0.999
Time:                        18:19:54   Log-Likelihood:                 426.31
No. Observations:                 125   AIC:                            -848.6
Df Residuals:                     123   BIC:                            -843.0
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
====
                        coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
----
Intercept            -0.0007      0.001     -0.638      0.525      -0.003
0.002
C(Positive)[T.1]   1.084e-06      0.001      0.001      0.999      -0.003
0.003
==============================================================================
Omnibus:                       14.980   Durbin-Watson:                   2.295
```

```
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               37.041
Skew:                          -0.375   Prob(JB):                     9.05e-09
Kurtosis:                       5.559   Cond. No.                         2.99
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

4.5.2 b. Calculate Point-biserial correlation coefficient

[30]:
```python
x = fxi_reg_CO['CloseOpen']
y = fxi_reg_CO['Positive']
std_x = np.std(x)
M1 = np.mean(fxi_reg_CO[fxi_reg_CO['Positive']==1]['CloseOpen'])
n1 = fxi_reg_CO[fxi_reg_CO['Positive']==1].shape[0]
M0 = np.mean(fxi_reg_CO[fxi_reg_CO['Negative']==1]['CloseOpen'])
n0 = fxi_reg_CO[fxi_reg_CO['Negative']==1].shape[0]
n = n1+n0
rpb = (M1-M0)*np.sqrt(n1*n0/(n*(n-1)))/std_x
print('Point-biserial correlation coefficient for Close-Open is:', rpb)
```

```
Point-biserial correlation coefficient for Close-Open is: 0.028838620300368312
```

[31]:
```python
x = sse_reg_OC['OpenClose']
y = sse_reg_OC['Positive']
std_x = np.std(x)
M1 = np.mean(sse_reg_OC[sse_reg_OC['Positive']==1]['OpenClose'])
n1 = sse_reg_OC[sse_reg_OC['Positive']==1].shape[0]
M0 = np.mean(sse_reg_OC[sse_reg_OC['Negative']==1]['OpenClose'])
n0 = sse_reg_OC[sse_reg_OC['Negative']==1].shape[0]
n = n1+n0
rpb = (M1-M0)*np.sqrt(n1*n0/(n*(n-1)))/std_x
print('Point-biserial correlation coefficient for Open-Close is:', rpb)
```

```
Point-biserial correlation coefficient for Open-Close is: -0.07985055685344604
```

4.6 6. FXI vs. textblob

4.6.1 a. Regression

Close-Open

[32]:
```python
fit_fxi_textblob_CO = ols('CloseOpen ~ C(Positive)', data=fxi_textblob_CO).fit()
print(fit_fxi_textblob_CO.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              CloseOpen   R-squared:                       0.002
```

```
Model:                            OLS   Adj. R-squared:                 -0.003
Method:                 Least Squares   F-statistic:                    0.3506
Date:                Wed, 16 Dec 2020   Prob (F-statistic):              0.554
Time:                        18:20:00   Log-Likelihood:                 680.48
No. Observations:                 227   AIC:                            -1357.
Df Residuals:                     225   BIC:                            -1350.
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
====
                      coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
----
Intercept           0.0009      0.001      0.693      0.489      -0.002
0.003
C(Positive)[T.1]   -0.0010      0.002     -0.592      0.554      -0.004
0.002
==============================================================================
Omnibus:                       14.843   Durbin-Watson:                   2.127
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               16.551
Skew:                          -0.552   Prob(JB):                     0.000255
Kurtosis:                       3.729   Cond. No.                         2.84
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

Open-Close

[33]:
```python
fit_fxi_textblob_OC = ols('OpenClose ~ C(Positive)', data=fxi_textblob_OC).fit()
print(fit_fxi_textblob_OC.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              OpenClose   R-squared:                       0.011
Model:                            OLS   Adj. R-squared:                  0.001
Method:                 Least Squares   F-statistic:                     1.059
Date:                Wed, 16 Dec 2020   Prob (F-statistic):              0.306
Time:                        18:20:02   Log-Likelihood:                 330.08
No. Observations:                  98   AIC:                            -656.2
Df Residuals:                      96   BIC:                            -651.0
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
====
                      coef    std err          t      P>|t|      [0.025
```

```
0.975]
------------------------------------------------------------------------------------
----
Intercept             0.0004      0.001      0.263      0.793      -0.002
0.003
C(Positive)[T.1]    -0.0018      0.002     -1.029      0.306      -0.005
0.002
==============================================================================
Omnibus:                       10.815   Durbin-Watson:                   2.263
Prob(Omnibus):                  0.004   Jarque-Bera (JB):               23.084
Skew:                          -0.297   Prob(JB):                     9.71e-06
Kurtosis:                       5.302   Cond. No.                         2.89
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

4.6.2 b. Calculate Point-biserial correlation coefficient

[34]:
```python
x = fxi_textblob_CO['CloseOpen']
y = fxi_textblob_CO['Positive']
std_x = np.std(x)
M1 = np.mean(fxi_textblob_CO[fxi_textblob_CO['Positive']==1]['CloseOpen'])
n1 = fxi_textblob_CO[fxi_textblob_CO['Positive']==1].shape[0]
M0 = np.mean(fxi_textblob_CO[fxi_textblob_CO['Negative']==1]['CloseOpen'])
n0 = fxi_textblob_CO[fxi_textblob_CO['Negative']==1].shape[0]
n = n1+n0
rpb = (M1-M0)*np.sqrt(n1*n0/(n*(n-1)))/std_x
print('Point-biserial correlation coefficient for Close-Open is:', rpb)
```

```
Point-biserial correlation coefficient for Close-Open is: -0.044188713120793345
```

[35]:
```python
x = fxi_textblob_OC['OpenClose']
y = fxi_textblob_OC['Positive']
std_x = np.std(x)
M1 = np.mean(fxi_textblob_OC[fxi_textblob_OC['Positive']==1]['OpenClose'])
n1 = fxi_textblob_OC[fxi_textblob_OC['Positive']==1].shape[0]
M0 = np.mean(fxi_textblob_OC[fxi_textblob_OC['Negative']==1]['OpenClose'])
n0 = fxi_textblob_OC[fxi_textblob_OC['Negative']==1].shape[0]
n = n1+n0
rpb = (M1-M0)*np.sqrt(n1*n0/(n*(n-1)))/std_x
print('Point-biserial correlation coefficient for Open-Close is:', rpb)
```

```
Point-biserial correlation coefficient for Open-Close is: -0.016016097171625734
```

5 References

https://medium.com/@outside2SDs/an-overview-of-correlation-measures-between-categorical-and-continuous-variables-4c7f85610365

https://en.wikipedia.org/wiki/Point-biserial_correlation_coefficient

https://topforeignstocks.com/foreign-adrs-list/the-full-list-of-chinese-adrs/

https://finance.yahoo.com/

https://www.barchart.com/

[ ]:

```

```