# **Data preprocessing/augmentation**

## Data cleaning

Fetch one-min stock data for trading hours only - 09:30 EST to 16:00 EST

Handle missing samples

## Data augment

Build DateTime column as dataframe index

Calculate one-min returns

Calculate cum returns for the interested period

Calculate daily RSI for the interested period

Concatenate sector etf stock price and returns (needed for RNN time-series forecasting)

Concatenate S&P500 etf stock price and returns (needed for RNN time-series forecasting)

In [187]:

import pandas as pd

import glob

**# read daily stock data for single stock under consideration (MSFT)**

stockPath ='C:/cygwin64/home/mpatnam/CSCIE63/CSCI E-89 Deep Learning/Project/stocks'

stockFiles = glob.glob(stockPath + "/allstocks\_2014\*/table\_msft.csv")

stockdata = pd.concat([pd.read\_csv(f, header=None , names=['Date', 'Time', 'Open', 'High', 'Low', 'Close', 'Volume', 'SplitFactor', 'Earnings', 'Dividends']) for f in stockFiles], axis=0)

stockdata

Out[187]:

|  | **Date** | **Time** | **Open** | **High** | **Low** | **Close** | **Volume** | **SplitFactor** | **Earnings** | **Dividends** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 20140102 | 700 | 33.6842 | 33.6842 | 33.6842 | 33.6842 | 2.217100e+00 | 1 | 0 | 0.0 |
| **1** | 20140102 | 705 | 33.6300 | 33.6300 | 33.6300 | 33.6300 | 3.769010e+01 | 1 | 0 | 0.0 |
| **432** | 20141231 | 1958 | 43.1611 | 43.1611 | 43.1611 | 43.1611 | 1.078980e+02 | 1 | 0 | 0.0 |

114842 rows × 10 columns

In [188]:

**#**

**# Read daily stock data for single etf under consideration (XLK)**

**# Note the single stock chosen (MSFT) is one of the biggest holdings of this ETF.**

**#**

etfPath = 'C:/cygwin64/home/mpatnam/CSCIE63/CSCI E-89 Deep Learning/Project/etfs'

etfFiles = glob.glob(etfPath + "/allstocks\_2014\*/table\_xlk.csv")

etfdata = pd.concat([pd.read\_csv(f, header=None , names=['etf.Date', 'etf.Time', 'etf.Open', 'etf.High', 'etf.Low', 'etf.Close', 'etf.Volume',

'etf.SplitFactor', 'etf.Earnings', 'etf.Dividends']) for f in etfFiles], axis=0)

print('Total rows x columns:', etfdata.shape)

etfdata

Total rows x columns: (99969, 10)

Out[188]:

|  | **etf.Date** | **etf.Time** | **etf.Open** | **etf.High** | **etf.Low** | **etf.Close** | **etf.Volume** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 20140102 | 800 | 33.4127 | 33.4127 | 33.4127 | 33.4127 | 234.3850 |
| **1** | 20140102 | 841 | 33.3940 | 33.3940 | 33.3940 | 33.3940 | 57.7937 |
| **2** | 20140102 | 913 | 33.3846 | 33.3846 | 33.3846 | 33.3846 | 149.8350 |
| **397** | 20141231 | 1659 | 39.4072 | 39.4072 | 39.4072 | 39.4072 | 52.5158 |

99969 rows × 10 columns

In [189]:

**#**

**# Read daily stock data for the etf that represents the entire S&P500 index (SPX)**

**# Note the single stock chosen (MSFT) is one of the biggest holdings of this ETF.**

**#**

spyPath = 'C:/cygwin64/home/mpatnam/CSCIE63/CSCI E-89 Deep Learning/Project/etfs'

spyFiles = glob.glob(etfPath + "/allstocks\_2014\*/table\_spy.csv")

spydata = pd.concat([pd.read\_csv(f, header=None , names=['spy.Date', 'spy.Time', 'spy.Open', 'spy.High', 'spy.Low', 'spy.Close', 'spy.Volume',

'spy.SplitFactor', 'spy.Earnings', 'spy.Dividends']) for f in spyFiles], axis=0)

print('Total rows x columns:', spydata.shape)

spydata

Total rows x columns: (171924, 10)

Out[189]:

|  | **spy.Date** | **spy.Time** | **spy.Open** | **spy.High** | **spy.Low** | **spy.Close** | **spy.Volume** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 20140102 | 700 | 170.728 | 170.747 | 170.700 | 170.700 | 53056.4000 |
| **1** | 20140102 | 701 | 170.663 | 170.700 | 170.663 | 170.672 | 8090.0200 |
| **2** | 20140102 | 702 | 170.672 | 170.700 | 170.672 | 170.700 | 4313.5300 |
| **629** | 20141231 | 1959 | 194.443 | 194.462 | 194.443 | 194.452 | 2379.6900 |

171924 rows × 10 columns

In [195]:

**# Calculate daily return**

stockdata['stock.Return'] = (stockdata['Close']-stockdata['Open'])\*100/stockdata['Open']

stockdata

Out[195]:

|  | **Date** | **Time** | **Open** | **High** | **Low** | **Close** | **Volume** | **stock.Return** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 20140102 | 700 | 33.6842 | 33.6842 | 33.6842 | 33.6842 | 2.217100e+00 | 0.000000 |
| **1** | 20140102 | 705 | 33.6300 | 33.6300 | 33.6300 | 33.6300 | 3.769010e+01 | 0.000000 |
| **2** | 20140102 | 721 | 33.6300 | 33.6300 | 33.6300 | 33.6300 | 1.108530e+02 | 0.000000 |

114842 rows × 11 columns

In [196]:

**# Calculate daily return of etf**

etfdata['etf.Return'] = (etfdata['etf.Close']-etfdata['etf.Open'])\*100/etfdata['etf.Open']

etfdata

Out[196]:

|  | **etf.Date** | **etf.Time** | **etf.Open** | **etf.High** | **etf.Low** | **etf.Close** | **etf.Volume** | **etf.Return** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 20140102 | 800 | 33.4127 | 33.4127 | 33.4127 | 33.4127 | 234.3850 | 0.000000 |
| **1** | 20140102 | 841 | 33.3940 | 33.3940 | 33.3940 | 33.3940 | 57.7937 | 0.000000 |
| **2** | 20140102 | 913 | 33.3846 | 33.3846 | 33.3846 | 33.3846 | 149.8350 | 0.000000 |

99969 rows × 11 columns

In [197]:

**# Calculate one-min/cumulative returns of spy**

spydata['spy.Return'] = (spydata['spy.Close']-spydata['spy.Open'])\*100/spydata['spy.Open']

spydata

Out[197]:

|  | **spy.Date** | **spy.Time** | **spy.Close** | **spy.Volume** | **spy.Return** |
| --- | --- | --- | --- | --- | --- |
| **0** | 20140102 | 700 | 170.700 | 53056.4000 | -0.016400 |
| **1** | 20140102 | 701 | 170.672 | 8090.0200 | 0.005274 |
| **2** | 20140102 | 702 | 170.700 | 4313.5300 | 0.016406 |
| **629** | 20141231 | 1959 | 194.452 | 2379.6900 | 0.004629 |

171924 rows × 11 columns

In [198]:

**#**

**# Calculate average-gain and loss for each day using one-min samples**

**#**

arr1=stockdata['Close'][:-1].values

arr2=stockdata['Close'][1:].values

arr\_gain=arr2-arr1

arr\_loss=arr2-arr1

print(arr\_gain.shape)

print(arr\_loss.shape)

arr\_gain[arr\_gain<0]=0

arr\_loss[arr\_loss>0]=0

arr\_loss=abs(arr\_loss)

arr\_gain=np.insert(arr\_gain,0, 0)

arr\_loss=np.insert(arr\_loss,0, 0)

**# Add intermediate Gain/Loss columns - used later for RSI**

stockdata['Gain']=arr\_gain

stockdata['Loss']=arr\_loss

stockdata['RSI']=0.0

stockdata

Out[198]:

|  | **Date** | **Time** | **Open** | **High** | **Low** | **Close** | **Volume** | **stock.Return** | **Gain** | **Loss** | **RSI** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 20140102 | 700 | 33.6842 | 33.6842 | 33.6842 | 33.6842 | 2.217100e+00 | 0.000000 | 0.0000 | 0.0000 | 0.0 |
| **1** | 20140102 | 705 | 33.6300 | 33.6300 | 33.6300 | 33.6300 | 3.769010e+01 | 0.000000 | 0.0000 | 0.0542 | 0.0 |
| **2** | 20140102 | 721 | 33.6300 | 33.6300 | 33.6300 | 33.6300 | 1.108530e+02 | 0.000000 | 0.0000 | 0.0000 | 0.0 |

114842 rows × 14 columns

In [199]:

**#**

**# calculate RSI:**

**# RSI = 100 - 100 / (1 + RS)**

**# where,RS = Average Gain / Average Loss**

**#**

groupby\_date = stockdata['Gain'].groupby(stockdata['Date'])

agg\_gain=groupby\_date.apply(sum)

groupby\_date = stockdata['Loss'].groupby(stockdata['Date'])

agg\_loss=groupby\_date.apply(sum)

for idx, date in enumerate(agg\_gain.index.values):

RSI = 100 - (100 / (1 + (agg\_gain.values[idx] / agg\_loss.values[idx])))

print(date, agg\_gain.values[idx], agg\_loss.values[idx], RSI)

stockdata.loc[(stockdata['Date'] == date), 'RSI'] = RSI

stockdata

Out[199]:

|  | **Date** | **Time** | **Open** | **High** | **Low** | **Close** | **Volume** | **stock.Return** | **Gain** | **Loss** | **RSI** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 20140102 | 700 | 33.6842 | 33.6842 | 33.6842 | 33.6842 | 2.217100e+00 | 0.000000 | 0.0000 | 0.0000 | 48.474178 |
| **1** | 20140102 | 705 | 33.6300 | 33.6300 | 33.6300 | 33.6300 | 3.769010e+01 | 0.000000 | 0.0000 | 0.0542 | 48.474178 |
| **2** | 20140102 | 721 | 33.6300 | 33.6300 | 33.6300 | 33.6300 | 1.108530e+02 | 0.000000 | 0.0000 | 0.0000 | 48.474178 |

114842 rows × 14 columns

In [200]:

**# Use samples from trading hours only**

stockdata=stockdata[(stockdata['Time']>=930) & (stockdata['Time']<=1600)]

etfdata=etfdata[(etfdata['etf.Time']>=930) & (etfdata['etf.Time']<=1600)]

spydata=spydata[(spydata['spy.Time']>=930) & (spydata['spy.Time']<=1600)]

In [208]:

**# Data Wrangling -**

**# convert Date/Time columns into native datetime type**

**# set DateTime as table index column**

stockdata['DateTimeStr'] = stockdata['Date'].map(str)+' '+stockdata['Time'].map(str)

stockdata['DateTime'] = pd.to\_datetime(stockdata['DateTimeStr'], format="%Y%m%d %H%M")

stockdata=stockdata.set\_index(pd.DatetimeIndex(stockdata['DateTime']))

stockdata

Out[208]:

|  | **Close** | **Volume** | **stock.Return** | **Gain** | **Loss** | **RSI** | **DateTimeStr** | **DateTime** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **DateTime** |  |  |  |  |  |  |  |  |
| **2014-01-02 09:30:00** | 33.6842 | 1319510.0 | -0.026712 | 0.0000 | 0.0090 | 48.474178 | 20140102 930 | 2014-01-02 09:30:00 |
| **2014-01-02 09:31:00** | 33.7157 | 206817.0 | 0.066779 | 0.0315 | 0.0000 | 48.474178 | 20140102 931 | 2014-01-02 09:31:00 |

98061 rows × 16 columns

In [209]:

**# Do the same data wrangling for etfdata and spydata**

Out[209]:

In [212]:

**# calculate cumulative returns**

stockdata['stock.CumReturn'] = (stockdata['Close']-stockdata['Close'][0])\*100/stockdata['Close'][0]

etfdata['etf.CumReturn'] = (etfdata['etf.Close']-etfdata['etf.Close'][0])\*100/etfdata['etf.Close'][0]

spydata['spy.CumReturn'] = (spydata['spy.Close']-spydata['spy.Close'][0])\*100/spydata['spy.Close'][0]

In [216]:

**# Address missing samples of MSFT/XLK/SPY data by doing an inner join**

stock\_combine\_data=pd.concat([stockdata,

etfdata['etf.Open'], etfdata['etf.Close'], etfdata['etf.Return'], etfdata['etf.CumReturn'],

spydata['spy.Open'], spydata['spy.Close'], spydata['spy.Return'], spydata['spy.CumReturn']],

axis=1, join='inner')

Out[216]:

In [218]:

**# plot cum returns of stock vs etf vs index**

import matplotlib.pyplot as plt

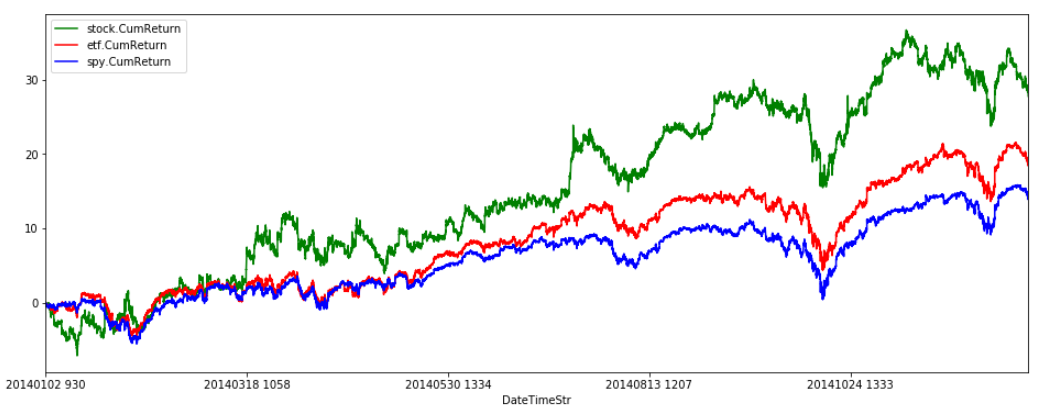
plt.figure(figsize=(16,6))

stock\_combine\_data.plot(x='DateTimeStr', y='stock.CumReturn', color='green', ax=plt.gca())

stock\_combine\_data.plot(x='DateTimeStr', y='etf.CumReturn', color='red', ax=plt.gca())

stock\_combine\_data.plot(x='DateTimeStr', y='spy.CumReturn', color='blue', ax=plt.gca())

plt.show()



In [219]:

**# plot returns for each period (one-min) of stock vs etf vs index**

import matplotlib.pyplot as plt

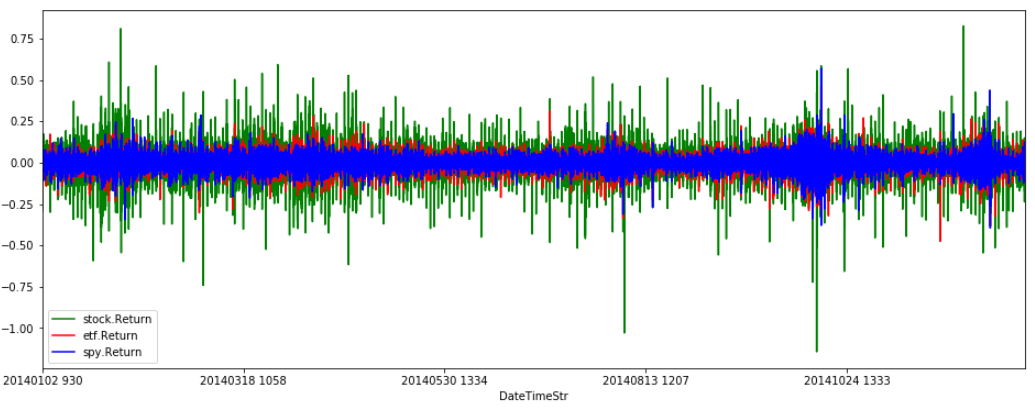
plt.figure(figsize=(16,6))

stock\_combine\_data.plot(x='DateTimeStr', y='stock.Return', color='green', ax=plt.gca())

stock\_combine\_data.plot(x='DateTimeStr', y='etf.Return', color='red', ax=plt.gca())

stock\_combine\_data.plot(x='DateTimeStr', y='spy.Return', color='blue', ax=plt.gca())

plt.show()



#### **OBSERVATION**: We can see the volatility of MSFT stock compared to its sector etf (XLK) and the S&P index (SPY). As expected, volatility of MSFT is higher compared to the sector etf, which in turn is relatively higher compared to SPY. In general, individual stocks tend move larger compared to the indices representing the whole sector or market.

In [224]:

**# save the processed data to a csv file**

file\_name='Processed/msft\_stock.2014.csv'

stock\_combine\_data.to\_csv(file\_name, index=False)