jx1021_Jiangguangyu_Xue_Final Project 7831

July 9, 2019

In [1]:
```
import json
import datetime as dt
import urllib.request
import pandas as pd
import numpy as np

from sqlalchemy import Column, Integer, Float, String
from sqlalchemy import create_engine
from sqlalchemy import MetaData
from sqlalchemy import Table
from sqlalchemy import inspect
from sqlalchemy import select
from sqlalchemy import ForeignKey
```

In [2]:
```
requestURL = "https://eodhistoricaldata.com/api/eod/"
myEodKey = "5ba84ea974ab42.45160048"
startDate = dt.datetime(2017,12,31)
endDate = dt.datetime(2019,5,3)
```

In [27]:
```
stock_list = pd.read_csv("PairTrading_demo.csv")
```

In [28]:
```
stock_list.head()
```

Out[28]:

	Stock1	Stock2
0	AAPL	HPQ
1	APC	CHK
2	CVX	XOM
3	DAL	UAL
4	T	VZ

In [29]:
```
stock1 = []
for stock in stock_list["Stock1"]:
    if stock not in stock1:
        stock1.append(stock)

stock2 = []
for stock in stock_list["Stock2"]:
    if stock not in stock2:
        stock2.append(stock)
```

```
In [30]: def get_daily_data(symbol, start=startDate, end=endDate, requestType=requestURL, apiK
             symbolURL = str(symbol) + ".US?"
             startURL = "from=" + str(start)
             endURL = "to=" + str(end)
             apiKeyURL = "api_token=" + myEodKey
             completeURL = requestURL + symbolURL + startURL + '&' + endURL + '&' + apiKeyURL
             print(completeURL)
             with urllib.request.urlopen(completeURL) as req:
                 data = json.load(req)
                 return data


In [31]: def create_pair1_table(name, metadata, engine):
                 tables = metadata.tables.keys()
                 if name not in tables:
                         table = Table(name, metadata,
                                                 Column('symbol_1', String(50), primary_key=Tru
                                                 Column('date', String(50), primary_key=True, n
                                                 Column('open_1', Float, nullable=False),
                                                 Column('high_1', Float, nullable=False),
                                                 Column('low_1', Float, nullable=False),
                                                 Column('close_1', Float, nullable=False),
                                     Column('adjusted_close_1', Float, nullable=False),
                                                 Column('volume_1', Integer, nullable=False))
                         table.create(engine)
                         return table

         def create_pair2_table(name, metadata, engine):
                 tables = metadata.tables.keys()
                 if name not in tables:
                         table = Table(name, metadata,
                                                 Column('symbol_2', String(50), primary_key=Tru
                                                 Column('date', String(50), primary_key=True, n
                                                 Column('open_2', Float, nullable=False),
                                                 Column('high_2', Float, nullable=False),
                                                 Column('low_2', Float, nullable=False),
                                                 Column('close_2', Float, nullable=False),
                                                 Column('adjusted_close_2', Float, nullable=Fal
                                                 Column('volume_2', Integer, nullable=False))
                         table.create(engine)
                         return table


         def populate_stock1_data(tickers, metadata, engine, table_name):
             conn = engine.connect()
             table = metadata.tables[table_name]
             for ticker in tickers:
                 stock = get_daily_data(ticker)
```

```
            #print(stock)
            for stock_data in stock:
                #print(k, v)
                trading_date = stock_data['date']
                trading_open = stock_data['open']
                trading_high = stock_data['high']
                trading_low = stock_data['low']
                trading_close = stock_data['close']
                trading_adjusted_close = stock_data['adjusted_close']
                trading_volume = stock_data['volume']
                insert_st = table.insert().values(symbol_1=ticker, date=trading_date,
                                                  open_1 = trading_open, high_1 = trading_high,
                                                  close_1 = trading_close, adjusted_close_1 = tr
                                  volume_1 = trading_volume)
                conn.execute(insert_st)

        def populate_stock2_data(tickers, metadata, engine, table_name):
            conn = engine.connect()
            table = metadata.tables[table_name]
            for ticker in tickers:
                stock = get_daily_data(ticker)
                #print(stock)
                for stock_data in stock:
                    #print(k, v)
                    trading_date = stock_data['date']
                    trading_open = stock_data['open']
                    trading_high = stock_data['high']
                    trading_low = stock_data['low']
                    trading_close = stock_data['close']
                    trading_adjusted_close = stock_data['adjusted_close']
                    trading_volume = stock_data['volume']
                    insert_st = table.insert().values(symbol_2=ticker, date=trading_date,
                                                      open_2 = trading_open, high_2 = trading_high,
                                                      close_2 = trading_close, adjusted_close_2 = tr
                                  volume_2 = trading_volume)
                    conn.execute(insert_st)

In [32]: def create_PairPrices_table(name, metadata, engine):
            tables = metadata.tables.keys()
            if name not in tables:
                    table = Table(name, metadata,
                                                      Column('symbol_1', String(50),ForeignKey("Pair
                                                      Column('symbol_2',String(50),ForeignKey("Pair2
                                                      Column('date', String(50), primary_key=True, n
                                                      Column('open_1', Float, nullable=False),
                                                      Column('close_1', Float, nullable=False),
                                                      Column('open_2', Float, nullable=False),
                                                      Column('close_2', Float, nullable=False))
```

```
                table.create(engine)
        return table

    def populate_PairsPrices_data(stock_list, metadata, engine, table_name,stock1_table,st
        conn = engine.connect()
        table = metadata.tables[table_name]

        for row in np.array(stock_list):
            ticker_1 =row[0]
            ticker_2 =row[1]

            s = select([stock1_table.c.date,stock1_table.c.open_1,stock1_table.c.close_1])
            result = conn.execute(s)
            res_1 = result.fetchall()

            s = select([stock2_table.c.open_2,stock2_table.c.close_2]).where(stock2_table
            result = conn.execute(s)
            res_2 = result.fetchall()

            for i,item in enumerate(res_1):
                insert_st = table.insert().values(symbol_1=ticker_1, symbol_2=ticker_2,
                                                  date = res_1[i][0] , open_1 = res_1[i][
                                                  open_2 = res_2[i][0],close_2 = res_2[i]
                conn.execute(insert_st)

In [33]: def create_Pairs_table(name, metadata, engine):
            tables = metadata.tables.keys()
            if name not in tables:
                table = Table(name, metadata,
                                        Column('symbol_1',String(50), primary_key=True
                                        Column('symbol_2',String(50), primary_key=True
                                        Column('volatility', Float, nullable=False),
                                        Column('profit_loss', Float, nullable=False))
                table.create(engine)
            return table


    def populate_Pairs_data(stock_list, metadata, engine, table_name,stock1_table,stock2_t
        conn = engine.connect()
        table = metadata.tables[table_name]


        for row in np.array(stock_list):
            ticker_1 =row[0]
            ticker_2 =row[1]


            s = select([stock1_table.c.close_1]).where(stock1_table.c.symbol_1==ticker_1)
```

```
            result = conn.execute(s)
            close_1 = result.fetchall()
            close_1 = np.array(close_1).flatten()


            s = select([stock2_table.c.close_2]).where(stock2_table.c.symbol_2==ticker_2)
            result = conn.execute(s)
            close_2 = result.fetchall()
            close_2 = np.array(close_2).flatten()

            volatility = np.std(close_1/close_2)
            insert_st = table.insert().values(symbol_1=ticker_1, symbol_2=ticker_2,
                                              volatility = volatility,profit_loss = 0.0)

            conn.execute(insert_st)

    #def populate_Pairs_data(tickers, metadata, engine, table_name):
```

In [34]:

```
    def create_Trades_table(name, metadata, engine):
            tables = metadata.tables.keys()
            if name not in tables:
                    table = Table(name, metadata,
                                          Column('symbol_1',String(50), primary_key=True
                        Column('symbol_2',String(50), primary_key=True, nullable=False),
                                          Column('date', String(50), primary_key=True, n
                                          Column('profit_loss_d', Float, nullable=False)
                    table.create(engine)
            return table
```

In [35]:

```
def populate_Trades_data(tickers, metadata, engine, table_name,Pairs_table,stock1_table

        conn = engine.connect()
        table = metadata.tables[table_name]

        s = select([Pairs_table.c.volatility])
        result = conn.execute(s)
        volatility = result.fetchall()
        volatility = np.array(volatility).flatten()

        s = select([stock1_table.c.date]).where(stock1_table.c.symbol_1=="AAPL").where(sto
        result = conn.execute(s)
        dates= result.fetchall()
```

```
    dates = np.array(dates).flatten()


    for i,row in enumerate(np.array(stock_list)):
        ticker_1 =row[0]
        ticker_2 =row[1]

        s = select([stock1_table.c.open_1,stock1_table.c.close_1]).where(stock1_table
        result = conn.execute(s)
        result = result.fetchall()
        open_1,close_1 = np.array(result).transpose()

        s = select([stock2_table.c.open_2,stock2_table.c.close_2]).where(stock2_table
        result = conn.execute(s)
        result = result.fetchall()
        open_2,close_2 = np.array(result).transpose()

        condition = close_1[:-1]/close_2[:-1]-open_1[1:]/open_2[1:]-k*volatility[i]
        PnL_sum = 0

        for j,date in enumerate(dates):
            if j ==0:
                continue
            if (condition[j-1]>0):
                n1 = -10000
            if(condition[j-1]<0):
                n1 = 10000

            n2 = - np.floor(n1*(open_1[j]/open_2[j]))
            PnL = n1*(close_1[j]-open_1[j]) + n2*(close_2[j]-open_2[j])

            insert_st = table.insert().values(symbol_1=ticker_1, symbol_2=ticker_2,
                                              date = date,profit_loss_d = PnL)
            conn.execute(insert_st)

            PnL_sum += PnL
        print(PnL)
        table_2 = metadata.tables["Pairs_table"]

        stmt = table_2.update().values(profit_loss = PnL_sum).where(table_2.c.symbol_
        conn.execute(stmt)




def clear_a_table(table_name, metadata, engine):
```

```
            conn = engine.connect()
            table = metadata.tables[table_name]
            delete_st = table.delete()
            conn.execute(delete_st)



        def execute_sql_statement(sql_st, engine):
            result = engine.execute(sql_st)
            return result

        def build_pair_trading_model():
            return 0
```

In [36]:
```
metadata = MetaData()
engine =  create_engine('sqlite:///:memory:')
conn = engine.connect()
```

In [37]:
```
stock1_table = create_pair1_table("Pair1Stocks", metadata, engine)
```

In [38]:
```
populate_stock1_data(stock1, metadata, engine,"Pair1Stocks")
```

```
https://eodhistoricaldata.com/api/eod/AAPL.US?from=2017-12-31 00:00:00&to=2019-05-03 00:00:00&a
https://eodhistoricaldata.com/api/eod/APC.US?from=2017-12-31 00:00:00&to=2019-05-03 00:00:00&ap
https://eodhistoricaldata.com/api/eod/CVX.US?from=2017-12-31 00:00:00&to=2019-05-03 00:00:00&ap
https://eodhistoricaldata.com/api/eod/DAL.US?from=2017-12-31 00:00:00&to=2019-05-03 00:00:00&ap
https://eodhistoricaldata.com/api/eod/T.US?from=2017-12-31 00:00:00&to=2019-05-03 00:00:00&api_
```

In [39]:
```
stock2_table = create_pair2_table("Pair2Stocks", metadata, engine)
populate_stock2_data(stock2, metadata, engine,"Pair2Stocks")
```

```
https://eodhistoricaldata.com/api/eod/HPQ.US?from=2017-12-31 00:00:00&to=2019-05-03 00:00:00&ap
https://eodhistoricaldata.com/api/eod/CHK.US?from=2017-12-31 00:00:00&to=2019-05-03 00:00:00&ap
https://eodhistoricaldata.com/api/eod/XOM.US?from=2017-12-31 00:00:00&to=2019-05-03 00:00:00&ap
https://eodhistoricaldata.com/api/eod/UAL.US?from=2017-12-31 00:00:00&to=2019-05-03 00:00:00&ap
https://eodhistoricaldata.com/api/eod/VZ.US?from=2017-12-31 00:00:00&to=2019-05-03 00:00:00&api
```

In [40]:
```
PairPrices_table = create_PairPrices_table("PairPrices", metadata, engine)
populate_PairsPrices_data(stock_list, metadata, engine, "PairPrices",stock1_table,stoc
```

In [41]:
```
Pairs_table = create_Pairs_table("Pairs_table", metadata, engine)#volatility,PnL
populate_Pairs_data(stock_list, metadata, engine, "Pairs_table",stock1_table,stock2_ta
```

In [42]:
```
k = 1
Trades_table = create_Trades_table("Trades_table", metadata, engine)
```

In [43]:
```
populate_Trades_data(stock_list, metadata, engine,"Trades_table",Pairs_table,stock1_ta
```

```
16020.079999999703
43120.82999999975
-19448.22000000004
8801.29999999996
-27739.320000000316
```

```
In [44]: s=select([Pairs_table])
         result = conn.execute(s)
         result.fetchall()
```

```
Out[44]: [('AAPL', 'HPQ', 0.5553570579532815, -29696.490000000478),
          ('APC', 'CHK', 2.777387301509632, -111562.48000000039),
          ('CVX', 'XOM', 0.053188692413880884, 20660.009999999307),
          ('DAL', 'UAL', 0.07489218195313094, -66522.47999999927),
          ('T', 'VZ', 0.07777551601082226, 147149.81999999922)]
```

```
In [56]: s=select([stock1_table]).where(stock1_table.c.open_1>0)
         result = conn.execute(s)
         q =  result.fetchall()
```

```
In [57]: np.array(q).shape
```

```
Out[57]: (1680, 8)
```

```
In [46]: conn = engine.connect()
         s = select([stock1_table.c.open_1,stock1_table.c.close_1]).where(stock1_table.c.symbol
         result = conn.execute(s)
         result= result.fetchall()
         open_1_d1 =result[0][0]
         close_1_d1 = result[0][1]
```

```
In [47]: def real_time_trading():
             print("Real time trade start:")
             print("Please enter parameter k:")
             k = float(input())

             #input the tickers for the pair
             print("Please input ticker for the first stock of the pair:")
             ticker_1 = input()
             print("Please enter a open price for it")
             open_1_d2 = float(input())
             print("Please enter a close price for it")
             close_1_d2 = float(input())


             print("Please input ticker for the second stock of the pair:")
             ticker_2 = input()
             print("Please enter a open price for it")
```

```
open_2_d2 = float(input())
print("Please enter a close price for it")
close_2_d2 = float(input())


print("Which day you want to choose as yesterday?(Please enter as yyyy-mm-dd)")
date_2 = input()

conn = engine.connect()

#get selected date open and close prices
s = select([stock1_table.c.open_1,stock1_table.c.close_1]).where(stock1_table.c.sy
result = conn.execute(s)
result= result.fetchall()
open_1_d1 =result[0][0]
close_1_d1 = result[0][1]

s = select([stock2_table.c.open_2,stock2_table.c.close_2]).where(stock2_table.c.sy
result = conn.execute(s)
result= result.fetchall()
open_2_d1 =  result[0][0]
close_2_d1 = result[0][1]

s=select([Pairs_table.c.volatility]).where(Pairs_table.c.symbol_1 == ticker_1).whe
result = conn.execute(s)
volatility = result.fetchall()
volatility = volatility[0][0]

if((close_1_d1/close_2_d1 - open_1_d2/open_2_d2)>= (k * volatility)):
    N1 = -10000
    print("")
    print("We accomplish result shown below:")
    print("Short this pair")
    print("Historical volatility:{}".format(volatility))
    print("Short {s1} of 10000 shares".format(s1 = ticker_1,s2 = N1))
    N2 = -np.floor(N1 * (open_1_d2/open_2_d2))
    print("long {s1} of {s2} shares".format(s1 =ticker_2 ,s2 = N2))

    PnL =  N1 * (close_1_d2 - open_1_d2) + N2 * (close_2_d2 - open_2_d2)

    print("Overall PnL :{}".format(PnL))

else:
    N1 = 10000
    print("")
    print("We accomplish result shown below:")
    print("Long this pair")
    print("Historical volatility:{}".format(volatility))
```

```
                print("Long {s1} of 10000 shares".format(s1 = ticker_1,s2 = N1))
                N2 = -N1 * (open_1_d2/open_2_d2)
                print("short {s1} of {s2} shares".format(s1 =ticker_2 ,s2 = N2))

                PnL =  N1 * (close_1_d2 - open_1_d2) + N2 * (close_2_d2 - open_2_d2)

                print("Overall PnL :{}".format(PnL))

                return 12138
```

```
In [48]: real_time_trading()

Real time trade start:
Please enter parameter k:
1
Please input ticker for the first stock of the pair:
AAPL
Please enter a open price for it
160
Please enter a close price for it
161
Please input ticker for the second stock of the pair:
HPQ
Please enter a open price for it
20
Please enter a close price for it
21
Which day you want to choose as yesterday?(Please enter as yyyy-mm-dd)
2018-12-31

We accomplish result shown below:
Long this pair
Historical volatility:0.5553570579532815
Long AAPL of 10000 shares
short HPQ of -80000.0 shares
Overall PnL :-70000.0


Out[48]: 12138
```