Comp 7405 Assignment 2 (10 marks)

1. (2 marks) Implement the Black-Scholes formulas for C(S; t) and P(S; t), and calculate the values of both call and put options with following parameters:

import numpy as np

from scipy.stats import norm

# 1. (2 marks) Implement the Black-Scholes formulas for C(S; t) and P(S; t),

# and calculate the values of both call and put options with following parameters:

# Data

# S = 50, K = 50, t = 0, T = 0:5, sigma = 20%, and r = 1%.

# S = 50, K = 60, t = 0, T = 0:5, sigma = 20%, and r = 1%.

# S = 50, K = 50, t = 0, T = 1:0, sigma = 20%, and r = 1%.

# S = 50, K = 50, t = 0, T = 0:5, sigma = 30%, and r = 1%.

# S = 50, K = 50, t = 0, T = 0:5, sigma = 20%, and r = 2%.

data = [

{

"S": 50,

"K": 50,

"t": 0,

"T": 0.5,

"sigma": 0.2,

"r": 0.01

},

{

"S": 50,

"K": 60,

"t": 0,

"T": 0.5,

"sigma": 0.2,

"r": 0.01

},

{

"S": 50,

"K": 50,

"t": 0,

"T": 1,

"sigma": 0.2,

"r": 0.01

},

{

"S": 50,

"K": 50,

"t": 0,

"T": 0.5,

"sigma": 0.3,

"r": 0.01

},

{

"S": 50,

"K": 50,

"t": 0,

"T": 0.5,

"sigma": 0.2,

"r": 0.02

},

]

# σ = sigma

# δ = delta

# S = Current stock price

# N(d1) and N(d2) = Cumulative density function

# K = Exercise price

# r = Annualised risk free rate

# d = Annual dividend yield of underlying stock

# T = Time to expiry

# ln(S / X) = Natural logarithmic value of (S / X)

# e = 2.71828

# delta = Annual dividend yield of underlying stock

# sigma = Annualised standard deviation of share returns or Volatility

def d1(S, K, t, T, r, sigma):

# d1 = ((ln(S / K) + (r + (sigma^2 / 2)) \* (T - t)) / (sigma \* √T-t)

time = T - t

lnSK = np.log(S / K)

rate = (r + (np.power(sigma, 2) / 2)) \* (time)

denominator = sigma \* (np.sqrt(time))

return (lnSK + rate) / denominator

def d2(S, K, t, T, r, sigma):

# d2 = d1 - sigma \* √T-t

return d1(S, K, t, T, r, sigma) - (sigma \* np.sqrt(T - t))

def black\_scholes\_call(S, K, t, T, r, sigma):

# Call Option

# C(S,t) = SN(d1) - Ke^(-r(T - t)) \* N(d2)

return (S \* norm.cdf(d1(S, K, t, T, r, sigma))) - ((K \* np.exp(-r \* (T - t))) \* norm.cdf(d2(S, K, t, T, r, sigma)))

def black\_scholes\_put(S, K, t, T, r, sigma):

# Put Option

# P(S,t) = Ke^(-r(T - t)) \* N(-d2) - SN(-d1)

return K \* np.exp(-r \* (T - t)) \* norm.cdf(-(d2(S, K, t, T, r, sigma))) - (S \* norm.cdf(-(d1(S, K, t, T, r, sigma))))

for stock in data:

print("Stock data: {}".format(stock))

S = stock["S"]

K = stock["K"]

t = stock["t"]

T = stock["T"]

r = stock["r"]

sigma = stock["sigma"]

print("Call option price: {}".format(

black\_scholes\_call(S, K, t, T, r, sigma)))

print("Put option price: {}".format(

black\_scholes\_put(S, K, t, T, r, sigma)))

Result of (1)

**Stock data**: {'S': 50, 'K': 50, 't': 0, 'T': 0.5, 'sigma': 0.2, 'r': 0.01}

Call option price: 2.9380121169138036

Put option price: 2.6886360765479225

**Stock data**: {'S': 50, 'K': 60, 't': 0, 'T': 0.5, 'sigma': 0.2, 'r': 0.01}

Call option price: 0.3870694028577839

Put option price: 10.08781815441872

**Stock data**: {'S': 50, 'K': 50, 't': 0, 'T': 1, 'sigma': 0.2, 'r': 0.01}

Call option price: 4.216659345054804

Put option price: 3.7191510325132064

**Stock data**: {'S': 50, 'K': 50, 't': 0, 'T': 0.5, 'sigma': 0.3, 'r': 0.01}

Call option price: 4.338822781168002

Put option price: 4.089446740802124

**Stock data**: {'S': 50, 'K': 50, 't': 0, 'T': 0.5, 'sigma': 0.2, 'r': 0.02}

Call option price: 3.060327056727921

Put option price: 2.5628187441863233

Base on the result, you will see that

**Strike** price is increased (50 -> 60)

Call option price will be decreased

Put option price will be increased

**Maturity** is increased (0.5 -> 1)

Call option will be increased

Put option will be increased

**Volatility** is increased (0.2 -> 0.3)

Call option will be increased

Put option will be increased

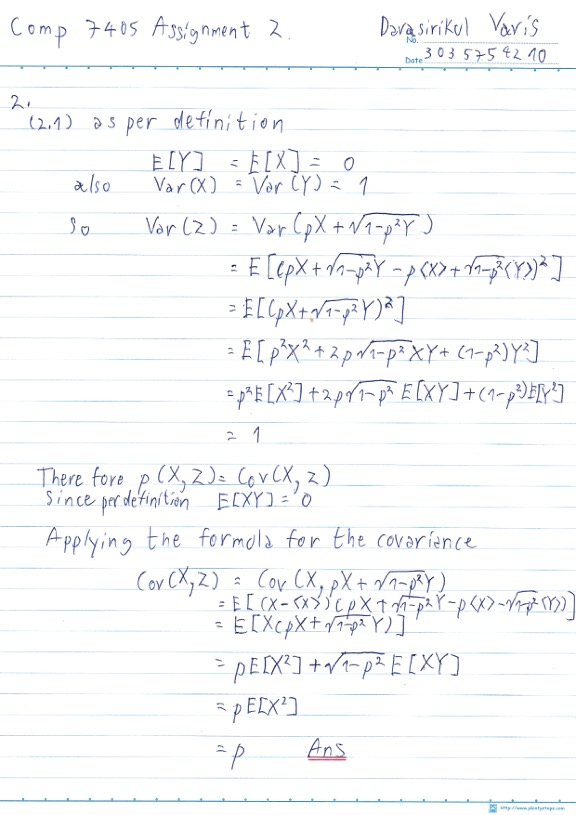
**Risk free rate** is increased (0.01 -> 0.02)

Call option will be increased

Put option will be decreased

2.

(2.1)



(2.2)

import numpy as np

# (2.2) Write a short program to numerically verify ρ(X,Z) = ρ

def generate\_standard\_normal\_random\_variable(size):

# (a) write a standard normal random variable generator.

return np.random.standard\_normal(size=(size, 2))

def generate\_Z(X, Y):

# Please use 0.5 as the correlation ρ.

p = 0.5

# Z formula Z = ρX + ((√1-ρ^2) \* Y)

return (p \* X) + (np.sqrt(1 - np.power(p, 2)) \* Y)

def calculate\_correlation\_coefficient(X\_list, Z\_list):

# snr\_list = [[X, Y]]

# Z\_list = [Z]

# Calculate ρ(X,Z)

# ρ(X,Z) = Cov(X,Z) / √Var(X)Var(Z)

# X\_list = snr\_list[:,0]

return np.cov(X\_list, Z\_list, bias=True)[0][1] / np.sqrt(np.var(X\_list) \* np.var(Z\_list))

def correlated\_normal\_random\_variables():

# (b) generate 200 samples of X and Y.

# standard normal random variable size 200

# snr = [[X,Y]]

snr\_list = generate\_standard\_normal\_random\_variable(200)

# (c) generate the samples of Z using the formula and the samples of X and Y .

Z\_list = [generate\_Z(snr[0], snr[1]) for snr in snr\_list]

# (d) calculate the sample correlation coefficient ρ(X,Z) based on the samples

# of X and Z, and compare it with the theoretical value 0.5.

X\_list = snr\_list[:, 0]

print("p(X, Z): {}".format(calculate\_correlation\_coefficient(X\_list, Z\_list)))

correlated\_normal\_random\_variables()

Result of 2.2:

**p(X, Z)** : 0.5315036877607728

3.

(3.1)

import numpy as np

from scipy.stats import norm

# σ = sigma

# δ = delta

def calculate\_d1\_d2(S, K, t, T, r, q, sigma):

time = T - t

lnSK = np.log(S / K)

rate = r - q

denominator = sigma \* (np.sqrt(time))

plus\_sigma = (1 / 2) \* sigma \* np.sqrt(time)

# d1 = ((ln(S / K) + (r - q)(T - t)) / (sigma \* √T-t)) + ((1/2) \* sigma \* √T-t)

d1 = ((lnSK + (rate \* time)) / denominator) + plus\_sigma

# d1 = ((ln(S / K) + (r - q)(T - t)) / (sigma \* √T-t)) - ((1/2) \* sigma \* √T-t)

d2 = ((lnSK + (rate \* time)) / denominator) - plus\_sigma

return d1, d2

def black\_scholes\_call(S, K, t, T, r, q, sigma):

time = T - t

d1, d2 = calculate\_d1\_d2(S, K, t, T, r, q, sigma)

# Call Option

# C(S,t) = Se^(-q(T - t))N(d1) - Ke^(-r(T - t)) \* N(d2)

return (S \* np.exp(-q \* (time)) \* norm.cdf(d1)) - ((K \* np.exp(-r \* (time))) \* norm.cdf(d2))

def black\_scholes\_put(S, K, t, T, r, q, sigma):

time = T - t

d1, d2 = calculate\_d1\_d2(S, K, t, T, r, q, sigma)

# Call Option

# P(S,t) = Ke^(-r(T - t)) \* N(-d2) - Se^(-q(T - t))N(-d1)

return ((K \* np.exp(-r \* (time))) \* norm.cdf(-d2)) - (S \* np.exp(-q \* (time)) \* norm.cdf(-d1))

def black\_scholes\_vega(S, K, t, T, r, q, sigma):

d1, d2 = calculate\_d1\_d2(S, K, t, T, r, q, sigma)

time = T - t

# Note that the formulas for ∂C(σ) / ∂σ and ∂P(σ) / ∂σ also need to change to:

# ∂C(σ) / ∂σ = ∂P(σ) / ∂σ = Se^(-q(T - t)) \* √T-t \* N'(d1)

return S \* np.exp(-q \* (time)) \* np.sqrt(time) \* norm.pdf(d1)

# (3.1)

# Implement the algorithm presented in Lecture 4 to calculate implied volatilities

# with the extended Black-Scholes formulas (1)-(2).

# ==========================================================================

def calculate\_implied\_volatility(S, K, t, T, r, q, option\_type, C\_true):

time = T - t

# The initial guess σ^ changes to:

#

# σ^ = √2|(lnS0 / K + (r - q)(T - t)) / T - t |

sigma\_hat = np.sqrt(2 \* np.abs(np.log(S / K) + ((r - q) \* (time))))

tol = 1e-8

nmax = 100

sigma\_diff = 1

n = 1

sigma = sigma\_hat

# C\_true = black\_scholes\_call(S, K, t, T, r, q, sigma\_true) if option\_type == 'C' else black\_scholes\_put(S, K, t, T, r, q, sigma\_true)

while (sigma\_diff >= tol and n < nmax):

C = black\_scholes\_call(S, K, t, T, r, q, sigma) if option\_type == 'C' else black\_scholes\_put(

S, K, t, T, r, q, sigma)

Cvega = black\_scholes\_vega(S, K, t, T, r, q, sigma)

if Cvega == 0:

return np.nan

increment = (C - C\_true) / Cvega

sigma = sigma - increment

n = n+1

sigmadiff = abs(increment)

return sigma

(3.2)

import csv

import numpy as np

import pandas as pd

from datetime import datetime

import matplotlib.pyplot as plt

from question\_3\_1 import calculate\_implied\_volatility

market\_data = []

instruments\_data = []

def convert\_local\_time(date\_time\_string):

return datetime.strptime(

date\_time\_string, '%Y-%b-%d %H:%M:%S.%f').time().strftime("%H:%M:%S")

def create\_date(date\_time\_string):

return datetime.strptime(

date\_time\_string, '%Y-%b-%d %H:%M:%S.%f')

with open('./instruments.csv') as csv\_file:

for row in csv.DictReader(csv\_file, skipinitialspace=True):

d\_row = {}

for key, value in row.items():

if (key != 'Type' and key != 'OptionType' and key != 'Symbol' and value != ''):

d\_row[key] = float(value)

else:

d\_row[key] = value

instruments\_data.append(d\_row)

with open('./marketdata.csv') as csv\_file:

for row in csv.DictReader(csv\_file, skipinitialspace=True):

local\_time = convert\_local\_time(row['LocalTime'])

d\_row = {}

for key, value in row.items():

if (key != 'LocalTime' and key != 'Symbol'):

d\_row[key] = float(value)

else:

d\_row[key] = value

market\_data.append(d\_row)

(3.2.1)

# calculate the bid/ask implied volatilities of all instruments at

# 09:31:00, 09:32:00, 09:33:00.

# Specifically, you take snapshots of the given market data at

# 09:31:00, 09:32:00, 09:33:00, respectively.

# Then for each snapshot,

# you calculate the bid implied volatility and ask implied volatility of each instrument.

# Put your results in three separate csv files

# using the names "31.csv", "32.csv", and "33.csv"

# (this is to make our tutor's life easier, thank you).

# The csv files should have the following format:

# ----------------------------------------------

# Strike | BidVolP | AskVolP | BidVolC | AskVolC

# 1.9 | .... | .... | .... | ....

def compute\_latest\_data(data\_list, exit\_key, exit\_value, spot\_key, spot\_value):

result = []

for index, data in enumerate(data\_list):

exit\_time = convert\_local\_time(data[exit\_key])

result\_index = next((index for (index, d) in enumerate(

result) if d["Symbol"] == data['Symbol']), None)

if result\_index is None:

result.append(data)

else:

result.pop(result\_index)

result.append(data)

if exit\_time == exit\_value:

break

last\_spot\_index = 0

for index, value in enumerate(result):

if value[spot\_key] == spot\_value:

last\_spot\_index = index

result = result[:(last\_spot\_index + 1)]

return result

def compute\_implied\_volatility(data, instruments, T, r, q):

# Sample {'LocalTime': '2016-Feb-16 09:32:00.907981', 'Symbol': 10000566.0, 'Last': 0.0027, 'Bid1': 0.0026, 'BidQty1': 1.0, 'Ask1': 0.0035, 'AskQty1': 6.0}

implied\_volatility = []

equity\_price = data[-1]

for index, market in enumerate(data):

instrument = next(

filter(lambda v: v['Symbol'] == market['Symbol'], instruments), None)

if instrument['Type'] != 'Option':

continue

computed\_data = {}

K = instrument['Strike']

bid\_implied\_volatility = calculate\_implied\_volatility(

equity\_price['Last'], K, 0, T, r, q, instrument['OptionType'], market['Bid1'])

ask\_implied\_volatility = calculate\_implied\_volatility(

equity\_price['Last'], K, 0, T, r, q, instrument['OptionType'], market['Ask1'])

bid\_implied\_volatility = 'NaN' if np.isnan(

bid\_implied\_volatility) else bid\_implied\_volatility

ask\_implied\_volatility = 'NaN' if np.isnan(

ask\_implied\_volatility) else ask\_implied\_volatility

if instrument['OptionType'] == 'P':

# Calculate Implied Volatility

computed\_data = {

'Strike': K,

'BidVolP': bid\_implied\_volatility,

'AskVolP': ask\_implied\_volatility,

'BidVolC': '',

'AskVolC': '',

'Symbol': market['Symbol'],

'LocalTime': market['LocalTime'],

}

if instrument['OptionType'] == 'C':

# Calculate Implied Volatility

computed\_data = {

'Strike': K,

'BidVolP': '',

'AskVolP': '',

'BidVolC': bid\_implied\_volatility,

'AskVolC': ask\_implied\_volatility,

'Symbol': market['Symbol'],

'LocalTime': market['LocalTime'],

}

implied\_volatility.append(computed\_data)

return implied\_volatility

def create\_result\_file(result\_file\_name, iv\_data):

with open(result\_file\_name, "w") as f:

wr = csv.DictWriter(

f, delimiter=",", fieldnames=list(iv\_data[0].keys()))

wr.writeheader()

wr.writerows(iv\_data)

def calculate\_bid\_ask\_implied\_volatilities\_all\_instruments():

# Compute 09:31:00 Data

options\_31 = compute\_latest\_data(

market\_data, 'LocalTime', '09:31:00', 'Symbol', '510050')

# Compute 09:32:00 Data

options\_32 = compute\_latest\_data(

market\_data, 'LocalTime', '09:32:00', 'Symbol', '510050')

# Compute 09:33:00 Data

options\_33 = compute\_latest\_data(

market\_data, 'LocalTime', '09:33:00', 'Symbol', '510050')

q = 0.2 # 20%

r = 0.04 # 4%

# Time to maturity

T = (24 - 16) / 365

iv\_31 = compute\_implied\_volatility(options\_31, instruments\_data, T, r, q)

iv\_32 = compute\_implied\_volatility(options\_32, instruments\_data, T, r, q)

iv\_33 = compute\_implied\_volatility(options\_33, instruments\_data, T, r, q)

# Create implied volatility result

create\_result\_file("31.csv", iv\_31)

create\_result\_file("32.csv", iv\_32)

create\_result\_file("33.csv", iv\_33)

return iv\_31, iv\_32, iv\_33

iv\_31, iv\_32, iv\_33 = calculate\_bid\_ask\_implied\_volatilities\_all\_instruments()

(3.2.2)

# (3.2.2)

# Put the results into three di  
erent plots one for each time point.

# For each plot, the x-axis should be the strike levels,

# and the y-axis should be implied volatilities.

def compute\_x\_y(iv\_data):

x = []

y = []

for iv in iv\_data:

x.append(iv['Strike'])

vola = 0

counter = 0

if iv['BidVolP'] != '' and iv['BidVolP'] != 'NaN':

vola += iv['BidVolP']

counter += 1

if iv['AskVolP'] != '' and iv['AskVolP'] != 'NaN':

vola += iv['AskVolP']

counter += 1

if iv['BidVolC'] != '' and iv['BidVolC'] != 'NaN':

vola += iv['BidVolC']

counter += 1

if iv['AskVolC'] != '' and iv['AskVolC'] != 'NaN':

vola += iv['AskVolC']

counter += 1

if (counter != 0):

vola = vola / counter

y.append(vola)

return x, y

def plot\_iv\_data(iv\_31, iv\_32, iv\_33):

print(pd.DataFrame(iv\_31))

x\_31, y\_31 = compute\_x\_y(iv\_31)

x\_32, y\_32 = compute\_x\_y(iv\_32)

x\_33, y\_33 = compute\_x\_y(iv\_33)

# For 31.csv

plt.plot(x\_31, y\_31)

# naming the x-axis

plt.xlabel('Strike Level')

# naming the y-axis

plt.ylabel('Implied volatilities.')

# plot title

plt.title('31.csv Data')

plt.show()

# For 32.csv

plt.plot(x\_32, y\_32)

# naming the x-axis

plt.xlabel('Strike Level')

# naming the y-axis

plt.ylabel('Implied volatilities.')

# plot title

plt.title('32.csv Data')

plt.show()

# For 33.csv

plt.plot(x\_33, y\_33)

# naming the x-axis

plt.xlabel('Strike Level')

# naming the y-axis

plt.ylabel('Implied volatilities.')

# plot title

plt.title('33.csv Data')

plt.show()

plot\_iv\_data(iv\_31, iv\_32, iv\_33)

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

(3.3)

import csv

import numpy as np

from datetime import datetime

from question\_1 import black\_scholes\_call, black\_scholes\_put

market\_data = []

instruments\_data = []

with open('./marketdata.csv') as csv\_file:

for row in csv.DictReader(csv\_file, skipinitialspace=True):

d\_row = {}

for key, value in row.items():

if (key != 'LocalTime' and key != 'Symbol'):

d\_row[key] = float(value)

else:

d\_row[key] = value

market\_data.append(d\_row)

with open('./instruments.csv') as csv\_file:

for row in csv.DictReader(csv\_file, skipinitialspace=True):

d\_row = {}

for key, value in row.items():

if (key != 'Type' and key != 'OptionType' and key != 'Symbol' and value != ''):

d\_row[key] = float(value)

else:

d\_row[key] = value

instruments\_data.append(d\_row)

r = 0.04 # 4%

q = 0.2 # 20%

# (3.3)

# The trading unit for buying/selling an option is 10000, and the transaction

# cost is about 3.3 RMBs per unit. Using the non-arbitrage conditions you

# have learned so far, check whether you see any arbitrage opportunities in

# the data. You can consider two cases: one without any transaction cost, and

# the other one with the real transaction cost. You can assume there is no

# transaction cost for A50ETF. Write down your findings and submit them.

def call\_put\_parity(S, K, t, T, r, q):

# C(S,t) - P(S,t) = Se^(-q(T - t)) - Ke^(-r(T - t))

time = T - t

return (S \* np.exp(-q \* (time))) - (K \* np.exp(-r \* (time)))

def write\_result\_to\_text\_file(text, file\_name):

# Open a file with access mode 'a'

with open(file\_name, "a") as file\_object:

# Append 'hello' at the end of file

file\_object.write(text)

file\_object.write("\n")

# Calculate call-put parity from 31.csv ()

def portfolio\_estimation():

filtered\_data = []

temp\_object = {}

for market in market\_data:

instrument = next(

filter(lambda v: v['Symbol'] == market['Symbol'], instruments\_data), None)

if instrument['Symbol'] == '510050':

temp\_object['Last'] = market['Last']

temp\_object['Bid1'] = market['Bid1']

temp\_object['BidQty1'] = market['BidQty1']

temp\_object['Ask1'] = market['Ask1']

temp\_object['AskQty1'] = market['AskQty1']

temp\_object['LocalTime'] = market['LocalTime']

filtered\_data.append(temp\_object)

temp\_object = {}

continue

if instrument['OptionType'] == 'C':

temp\_object['callLast'] = market['Last']

temp\_object['callBid'] = market['Bid1']

temp\_object['callBidQty'] = market['BidQty1']

temp\_object['callAsk'] = market['Ask1']

temp\_object['callAskQty'] = market['AskQty1']

temp\_object['callStrike'] = instrument['Strike']

if instrument['OptionType'] == 'P':

temp\_object['putLast'] = market['Last']

temp\_object['putBid'] = market['Bid1']

temp\_object['putBidQty'] = market['BidQty1']

temp\_object['putAsk'] = market['Ask1']

temp\_object['putAskQty'] = market['AskQty1']

temp\_object['putStrike'] = instrument['Strike']

with open("question\_3\_3\_generated\_data\_table.csv", "w") as f:

wr = csv.DictWriter(

f, delimiter=",", fieldnames=list(filtered\_data[0].keys()))

wr.writeheader()

wr.writerows(filtered\_data)

# Time to maturity

T = (24 - 16) / 365

sigma = 0.2

# Clean up portfolio compare result file

portfolio\_compare\_result\_file = 'question\_3\_3\_portfolio\_compare.txt'

open(portfolio\_compare\_result\_file, 'w').close()

for data in filtered\_data:

# Call - Put parity formula

# C(S,t) - P(S,t) = Se^(-q(T - t)) - Ke^(-r(T - t))

#

# Portfolio A = C(S,t) + Ke^(-r(T - t))

port\_a = black\_scholes\_call(

data['Last'], data['callStrike'], 0, T, r, sigma) + (data['callStrike'] \* np.exp(-r \* T))

# Portfolio B = P(S,t) + Se^(-q(T - t))

port\_b = black\_scholes\_put(

data['Last'], data['putStrike'], 0, T, r, sigma) + (data['Last'] \* np.exp(-q \* T))

write\_result\_to\_text\_file('Equity 510050 at: {}'.format(

data['LocalTime']), portfolio\_compare\_result\_file)

write\_result\_to\_text\_file(

'Without Transaction cost', portfolio\_compare\_result\_file)

write\_result\_to\_text\_file('Portfolio A: {}'.format(

port\_a), portfolio\_compare\_result\_file)

write\_result\_to\_text\_file('Portfolio B: {}'.format(

port\_b), portfolio\_compare\_result\_file)

if port\_a == port\_b:

write\_result\_to\_text\_file(

'Portfolio A == Portfolio B: No Arbitrage oppotunity', portfolio\_compare\_result\_file)

write\_result\_to\_text\_file(

'---------------------------------------------------', portfolio\_compare\_result\_file)

continue

if port\_a > port\_b:

write\_result\_to\_text\_file(

'Portfolio A > Portfolio B: So we should Buy Portfolio A and Sell Portfolio B', portfolio\_compare\_result\_file)

if port\_a < port\_b:

write\_result\_to\_text\_file(

'Portfolio A < Portfolio B: So we should Buy Portfolio B and Sell Portfolio A', portfolio\_compare\_result\_file)

write\_result\_to\_text\_file(

'--------------------------------------------------', portfolio\_compare\_result\_file)

return None

portfolio\_estimation()

Results and analysis

Equity 510050 at: 2016-Feb-16 09:30:04.520358

Without Transaction cost

Portfolio A: 2.0983792311040372

Portfolio B: 1.951456550880307

Portfolio A > Portfolio B: So we should Buy Portfolio A and Sell Portfolio B

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:30:09.587622

Without Transaction cost

Portfolio A: 1.9600295278877207

Portfolio B: 1.9691259088059465

Portfolio A < Portfolio B: So we should Buy Portfolio B and Sell Portfolio A

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:30:14.591004

Without Transaction cost

Portfolio A: 2.1481324145450515

Portfolio B: 1.9691259088059465

Portfolio A > Portfolio B: So we should Buy Portfolio A and Sell Portfolio B

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:30:19.560765

Without Transaction cost

Portfolio A: 1.9632180168490452

Portfolio B: 1.9546494138094925

Portfolio A > Portfolio B: So we should Buy Portfolio A and Sell Portfolio B

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:30:24.641089

Without Transaction cost

Portfolio A: 1.9771162866168404

Portfolio B: 1.950959207062523

Portfolio A > Portfolio B: So we should Buy Portfolio A and Sell Portfolio B

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:30:29.631013

Without Transaction cost

Portfolio A: 1.9640792353729952

Portfolio B: 1.951456550880307

Portfolio A > Portfolio B: So we should Buy Portfolio A and Sell Portfolio B

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:30:34.698456

Without Transaction cost

Portfolio A: 1.9590313246078548

Portfolio B: 1.9685476835772877

Portfolio A < Portfolio B: So we should Buy Portfolio B and Sell Portfolio A

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:30:39.675425

Without Transaction cost

Portfolio A: 1.9632180168490452

Portfolio B: 1.9685476835772877

Portfolio A < Portfolio B: So we should Buy Portfolio B and Sell Portfolio A

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:30:44.746451

Without Transaction cost

Portfolio A: 1.9615072038776593

Portfolio B: 1.9484753732585354

Portfolio A > Portfolio B: So we should Buy Portfolio A and Sell Portfolio B

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:30:49.732882

Without Transaction cost

Portfolio A: 2.007051577260538

Portfolio B: 1.9546494138094925

Portfolio A > Portfolio B: So we should Buy Portfolio A and Sell Portfolio B

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:30:54.724653

Without Transaction cost

Portfolio A: 1.9580332230131245

Portfolio B: 1.9679761998561436

Portfolio A < Portfolio B: So we should Buy Portfolio B and Sell Portfolio A

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:30:59.765946

Without Transaction cost

Portfolio A: 1.9580332230131245

Portfolio B: 1.9494689939414327

Portfolio A > Portfolio B: So we should Buy Portfolio A and Sell Portfolio B

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:31:04.765544

Without Transaction cost

Portfolio A: 1.957035228362366

Portfolio B: 1.952947348773829

Portfolio A > Portfolio B: So we should Buy Portfolio A and Sell Portfolio B

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:31:09.865762

Without Transaction cost

Portfolio A: 1.9590313246078548

Portfolio B: 1.9546494138094925

Portfolio A > Portfolio B: So we should Buy Portfolio A and Sell Portfolio B

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:31:14.821262

Without Transaction cost

Portfolio A: 1.9580332230131245

Portfolio B: 1.9537964194108959

Portfolio A > Portfolio B: So we should Buy Portfolio A and Sell Portfolio B

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:31:19.801187

Without Transaction cost

Portfolio A: 1.9615072038776593

Portfolio B: 1.952947348773829

Portfolio A > Portfolio B: So we should Buy Portfolio A and Sell Portfolio B

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:31:24.801608

Without Transaction cost

Portfolio A: 1.9615072038776593

Portfolio B: 1.9674114833724583

Portfolio A < Portfolio B: So we should Buy Portfolio B and Sell Portfolio A

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:31:29.852563

Without Transaction cost

Portfolio A: 1.9560373461558171

Portfolio B: 2.139573245259623

Portfolio A < Portfolio B: So we should Buy Portfolio B and Sell Portfolio A

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:31:34.849650

Without Transaction cost

Portfolio A: 1.9585522734594123

Portfolio B: 1.9494689939414327

Portfolio A > Portfolio B: So we should Buy Portfolio A and Sell Portfolio B

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:31:39.930647

Without Transaction cost

Portfolio A: 2.1481295690748246

Portfolio B: 1.952947348773829

Portfolio A > Portfolio B: So we should Buy Portfolio A and Sell Portfolio B

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:31:44.908538

Without Transaction cost

Portfolio A: 1.957035228362366

Portfolio B: 1.9484753732585354

Portfolio A > Portfolio B: So we should Buy Portfolio A and Sell Portfolio B

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:31:49.882024

Without Transaction cost

Portfolio A: 1.9585522734594123

Portfolio B: 1.9679761998561436

Portfolio A < Portfolio B: So we should Buy Portfolio B and Sell Portfolio A

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:31:54.837643

Without Transaction cost

Portfolio A: 1.9606577568942747

Portfolio B: 1.9474818650198475

Portfolio A > Portfolio B: So we should Buy Portfolio A and Sell Portfolio B

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:31:59.935368

Without Transaction cost

Portfolio A: 1.9606577568942747

Portfolio B: 1.997752220218177

Portfolio A < Portfolio B: So we should Buy Portfolio B and Sell Portfolio A

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:32:04.992453

Without Transaction cost

Portfolio A: 1.955039582144415

Portfolio B: 1.9512612744403928

Portfolio A > Portfolio B: So we should Buy Portfolio A and Sell Portfolio B

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:32:09.877191

Without Transaction cost

Portfolio A: 1.9560373461558171

Portfolio B: 1.9521022757583049

Portfolio A > Portfolio B: So we should Buy Portfolio A and Sell Portfolio B

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:32:14.936166

Without Transaction cost

Portfolio A: 1.9765404289278352

Portfolio B: 1.9494689939414327

Portfolio A > Portfolio B: So we should Buy Portfolio A and Sell Portfolio B

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:32:20.032478

Without Transaction cost

Portfolio A: 1.9623606484825875

Portfolio B: 1.9537964194108959

Portfolio A > Portfolio B: So we should Buy Portfolio A and Sell Portfolio B

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:32:25.002629

Without Transaction cost

Portfolio A: 1.9590313246078548

Portfolio B: 1.9546494138094925

Portfolio A > Portfolio B: So we should Buy Portfolio A and Sell Portfolio B

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:32:29.991008

Without Transaction cost

Portfolio A: 1.9771162866168404

Portfolio B: 1.9546494138094925

Portfolio A > Portfolio B: So we should Buy Portfolio A and Sell Portfolio B

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:32:35.061747

Without Transaction cost

Portfolio A: 1.9623606484825875

Portfolio B: 1.9679761998561436

Portfolio A < Portfolio B: So we should Buy Portfolio B and Sell Portfolio A

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:32:40.038588

Without Transaction cost

Portfolio A: 2.148130462772213

Portfolio B: 1.9679761998561436

Portfolio A > Portfolio B: So we should Buy Portfolio A and Sell Portfolio B

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:32:45.100994

Without Transaction cost

Portfolio A: 1.9623606484825875

Portfolio B: 1.9679761998561436

Portfolio A < Portfolio B: So we should Buy Portfolio B and Sell Portfolio A

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:32:50.067310

Without Transaction cost

Portfolio A: 1.9765404289278352

Portfolio B: 2.0412949321062444

Portfolio A < Portfolio B: So we should Buy Portfolio B and Sell Portfolio A

--------------------------------------------------

Equity 510050 at: 2016-Feb-16 09:32:55.065505

Without Transaction cost

Portfolio A: 1.9580332230131245

Portfolio B: 1.9537964194108959

Portfolio A > Portfolio B: So we should Buy Portfolio A and Sell Portfolio B

--------------------------------------------------

(Result file is at question\_3\_3\_portfolio\_compare.txt)