Problem 1

$r_t = 1.25 + a_t \qquad a_t = \sigma_t \epsilon_t$

$$\sigma_t^2 = \begin{cases} 0.2 a_{t-1}^2 + 0.9 \sigma_{t-1}^2 & \text{if } s_t = 1 \\ 4.25 + 0.1 a_{t-1}^2 + 0.75 \sigma_{t-1}^2 & \text{if } s_t = 2 \end{cases}$$

$P(s_t = 2 \mid s_{t-1} = 1) = 0.2 \qquad P(s_t = 1 \mid s_{t-1} = 2) = 0.2$

$a_{100} = 6 \qquad \sigma_{100}^2 = 50 \qquad s_{100} = 2$

1) $\sigma_{100}^2(1) = \omega_2 \times (0.2 \times a_{100}^2 + 0.9 \times \sigma_{100}^2) + (1-\omega_2)(4.25 + 0.1 \times a_{100}^2 + 0.75 \times \sigma_{100}^2)$

$$\sigma_{100}^2(1) = \underbrace{0.1 \times (0.2 \times 6^2 + 0.9 \times 50)}_{5.22} + \underbrace{0.9 \times (4.25 + 0.1 \times 6^2 + 0.75 \times 50)}_{40.815}$$

$\sigma_{100}^2(1) = 46.035$

ii)

$$\sigma^2_{100}(1) = P(S_{100}=2) \times 46.035 + 1 - P(S_{100}=2) \times \left[w_1 \times (4.25 + 0.1 \times 6^2 + 0.75 \times 50) \right.$$

$$\left. + (1-w_1) \times (0.2 \times 6^2 + 0.9 \times 50) \right]$$

$$\sigma^2_{100}(1) = 0.75 \times 46.035 + 0.25 \left[\underbrace{0.2 \times (4.25 + 0.1 \times 6^2 + 0.75 \times 50)}_{9.07} + \right.$$

$$\left. \underbrace{0.80 \times (0.2 \times 6^2 + 0.9 \times 50)}_{41.76} \right]$$

$$\sigma^2_{100}(1) = 0.75 * 46.035 + 12.7075 = 47.23375$$

HW4- FE542

M.Furkan Isik

4/16/2021

Promlem 2

In R create a report in pdf format using RMarkdown (or, if you choose to use Python instead,create a Jupyter notebook) to:

(i) Download daily price data for January 1, 2000 through December 31, 2020 of Amazonstock from Yahoo Finance. Compute theweeklylogarithmic returnsrt. You may usethe quantmod package in R for this purpose.

```
library(quantmod)

## Loading required package: xts

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: TTR

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

getSymbols('AMZN',src='yahoo',from='2000-01-01',to='2020-12-31')

## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.

## [1] "AMZN"
```

```r
r = weeklyReturn(AMZN,type="log")
length(r)

## [1] 1096
```

ii) Using lagged returnsrt−1,rt−2,rt−3as input, build a 3-2-1 feed-forward neural networkto forecast 1-step-ahead returns. Use data up to December 31, 2018 as the trainingdata set and the remainder as the testing data. Calculate the mean squared error onthe test data.

```r
library(nnet)
#Predict returns
r = as.numeric(r)
y_train = r[4:991]
x_train = cbind(r[3:990],r[2:989],r[1:988])
nn = nnet(x_train,y_train,size=2,linout=TRUE) #3-2-1 NN (output is real valued)

## # weights:  11
## initial  value 740.434815
## iter  10 value 4.891887
## iter  20 value 4.890944
## iter  30 value 4.889078
## iter  40 value 4.880880
## iter  50 value 4.873608
## iter  60 value 4.867152
## iter  70 value 4.864457
## iter  80 value 4.863125
## iter  90 value 4.861374
## iter 100 value 4.858287
## final  value 4.858287
## stopped after 100 iterations

summary(nn)

## a 3-2-1 network with 11 weights
## options were - linear output units
##  b->h1 i1->h1 i2->h1 i3->h1
##  -0.09   0.99   0.31   1.18
##  b->h2 i1->h2 i2->h2 i3->h2
##  -0.68  -2.34  -0.91  -2.93
##  b->o h1->o h2->o
## -1.16  1.77  0.92
```

- the mean squared error is 0.00146048

```r
#the mean squared error
ytest = r[992:1096]
xtest = cbind(r[991:1095],r[990:1094],r[989:1093])
ypred = predict(nn,xtest)
```

```r
prednet = mean((ytest - ypred)^2)
print(prednet)

## [1] 0.001464757
```

(iii) Using lagged returnsrt−1,rt−2,rt−3and their signs (directions) to build a 6-5-1 feed-forward neural network to forecast the 1-step-aheaddirectionof Microsoft stock pricemovement (with 1 denoting upward movement and 0 downward movement). Use dataup to December 31, 2018 as the training data set and the remainder as the testing data.Calculate the mean squared error on the test data

```r
#Predict direction of returns
dy_train = ifelse(r[4:991] > 0,1,0)
x_train = cbind(r[3:990],r[2:989],r[1:988])
x_train = cbind(x_train , ifelse(r[3:990]>0,1,0))
x_train = cbind(x_train , ifelse(r[2:989]>0,1,0))
x_train = cbind(x_train , ifelse(r[1:988]>0,1,0))
nn = nnet(x_train,dy_train,size=5,linout=F)

## # weights:  41
## initial  value 278.327987
## iter  10 value 246.104974
## iter  20 value 245.534744
## iter  30 value 244.588412
## iter  40 value 243.912904
## iter  50 value 243.394100
## iter  60 value 242.449556
## iter  70 value 241.728567
## iter  80 value 240.706439
## iter  90 value 240.470131
## iter 100 value 240.394586
## final  value 240.394586
## stopped after 100 iterations

summary(nn)

## a 6-5-1 network with 41 weights
## options were -
##   b->h1  i1->h1  i2->h1  i3->h1  i4->h1  i5->h1  i6->h1
##  -46.47  -37.03  -19.39   57.78   45.20  -17.49  -45.57
##   b->h2  i1->h2  i2->h2  i3->h2  i4->h2  i5->h2  i6->h2
## -141.59  -98.83  -17.97 -297.80   -8.72 -192.43  135.60
##   b->h3  i1->h3  i2->h3  i3->h3  i4->h3  i5->h3  i6->h3
##  -41.75 -132.27 -129.49  340.61 -166.91 -150.26 -292.31
##   b->h4  i1->h4  i2->h4  i3->h4  i4->h4  i5->h4  i6->h4
##  -82.87  -36.75 -125.28 -171.88 -133.79  -54.17 -142.65
##   b->h5  i1->h5  i2->h5  i3->h5  i4->h5  i5->h5  i6->h5
##  -63.97 -113.46 -214.56 -140.48  -60.03   32.99 -116.85
```

```
##    b->o   h1->o   h2->o   h3->o   h4->o   h5->o
##    0.04    3.85   73.34 -141.09  119.79   62.65
```

- mean squared error is 0.2504239
- accuracy is 0.6 which is not quite high

```r
#mean squared error

dy_test = ifelse(r[992:1096] > 0,1,0)
x_test = cbind(r[991:1095],r[990:1094],r[989:1093])
x_test = cbind(x_test , ifelse(r[991:1095]>0,1,0))
x_test = cbind(x_test , ifelse(r[990:1094]>0,1,0))
x_test = cbind(x_test , ifelse(r[989:1093]>0,1,0))

dy_pred = predict(nn,x_test)
msfe = mean((dy_test - dy_pred)^2)
print(msfe)

## [1] 0.2442915

acc = 1 - sum(abs(dy_test - ifelse(dy_pred > 0.5,1,0)))/length(dy_test)
print(acc)

## [1] 0.6190476
```