Collaborative Review Task 2

In this module, you are required to complete a Collaborative review task, which is designed to test your ability to apply and analyze the knowledge you have learned during the week.

Questions

1. Returns

```
- Download 1-2 years of price history of a stock.
- Compute its log return.
- Compute the mean, standard deviation, skewness, and excess kurtosis of its
log return.
- Repeat for a second stock.
- Compute the covariance and the correlation. Explain their difference. How
 do you convert one to the other?
```

2. Build your own transition

```
- Divide the data into 2 uneven parts: the first part is 80% of your data, a
nd the second part is 20%.
- Categorize each day in the 1-2 year price history as belonging to one of f
our categories:
    - Both stocks up
    - Stock #1 up, stock #2 down.
    - Stock #1 down, stock #2 up.
    - Both stocks down
- Build a transition matrix of portfolio direction that shows your portfolio
in four scenerios:
    - From moving together to moving together. That means starting from uu o
r dd & going to uu or dd.
    - From moving together to moving apart. That means starting from uu or d
d & going to ud or du.
    - From moving apart to moving together. That means starting from ud or d
u & going to uu or dd.
    - From moving apart to moving apart. That means starting from ud or du &
going to ud or du.
- How similar is the transition matrix from the first group to the second gr
oup?
- Is the process Markovian?
```

Answers:

```
In [1]: import numpy as np
        import pandas as pd
        import datetime
        import warnings
        warnings.filterwarnings('ignore')
```

1. Return

- Download 2 years data from Tesla and Exxon Mobile as below

```
In [2]: tesla = pd.read_csv("TSLA.csv",
                            delimiter=',')
        tesla['Date'] = pd.to_datetime(tesla['Date'])
        tesla.set_index('Date', inplace=True)

        exxon = pd.read_csv("XOM.csv",
                            delimiter=',')
        exxon['Date'] = pd.to_datetime(exxon['Date'])
        exxon.set_index('Date', inplace=True)
```

- Calculate daily log return

```
In [3]: # Calculate daily log return
        tesla["Log_Return"] = np.log(tesla['Price']).diff()
        exxon["Log_Return"] = np.log(exxon['Price']).diff()
```

```
In [4]: tesla.head()
```

Out[4]:

Date	Price	Log_Return
2015-03-09	190.880005	NaN
2015-03-10	190.320007	-0.002938
2015-03-11	193.740005	0.017810
2015-03-12	191.070007	-0.013877
2015-03-13	188.679993	-0.012587

In [5]:
```
exxon.head()
```

Out[5]:

Date	Price	Log_Return
2015-03-09	69.726906	NaN
2015-03-10	68.990021	-0.010624
2015-03-11	68.793503	-0.002853
2015-03-12	68.957260	0.002378
2015-03-13	68.670692	-0.004164

- Calculate mean, standard deviation, skewness, excess kurtosis. Note that we do both stocks at the same time

In [6]:
```
# Calculate mean, standard deviation, skewness, excess kurtosis
import scipy.stats as stats

tesla_mean = tesla['Log_Return'].mean()
tesla_std = tesla['Log_Return'].std()
tesla_skew = stats.skew(tesla['Log_Return'].dropna())
tesla_kurtosis = stats.kurtosis(tesla['Log_Return'].dropna())

exxon_mean = exxon['Log_Return'].mean()
exxon_std = exxon['Log_Return'].std()
exxon_skew = stats.skew(exxon['Log_Return'].dropna())
exxon_kurtosis = stats.kurtosis(exxon['Log_Return'].dropna())

print('Tesla')
print('The mean of Tesla Log Return is: {}'. format(tesla_mean))
print('The standard deviation of Tesla Log Return is: {}'. format(tesl
a_std))
print('The skewness of Tesla Log Return is: {}'. format(tesla_skew))
print('The kurtosis of Tesla Log Return is: {}'. format(tesla_kurtosis
))

print('\nExxon Mobile')
print('The mean of Exxon Mobile Log Return is: {}'. format(exxon_mean
))
print('The standard deviation of Exxon Mobile Log Return is: {}'. form
at(exxon_std))
print('The skewness of Exxon Mobile Log Return is: {}'. format(exxon_s
kew))
print('The kurtosis of Exxon Mobile Log Return is: {}'. format(exxon_k
urtosis))
```

```
Tesla
The mean of Tesla Log Return is: 0.000658917718912885
The standard deviation of Tesla Log Return is: 0.023921911654999784
The skewness of Tesla Log Return is: -0.3268473880026572
The kurtosis of Tesla Log Return is: 2.21193789165081

Exxon Mobile
The mean of Exxon Mobile Log Return is: -1.1141112389291308e-05
The standard deviation of Exxon Mobile Log Return is: 0.01167961025524
1375
The skewness of Exxon Mobile Log Return is: -0.0792199851483688
The kurtosis of Exxon Mobile Log Return is: 4.031992509924429
```

- Calculate Covariance and Correlation

In [7]:
```
# Calculate Covariance and Correlation
covariance_tesla_exxon = np.cov(tesla['Log_Return'].dropna(),
                                exxon['Log_Return'].dropna())[0][1]
correlation_tesla_exxon = np.corrcoef(tesla['Log_Return'].dropna(),
                                      exxon['Log_Return'].dropna())[0]
[1]
```

In [8]:
```
print("The Covariance between Tesla and Exxon Mobile is: {}".format(co
variance_tesla_exxon))
```

```
The Covariance between Tesla and Exxon Mobile is: 5.2462135978720254e-
05
```

In [9]:
```
print("The Correlation between Tesla and Exxon Mobile is: {}".format(c
orrelation_tesla_exxon))
```

```
The Correlation between Tesla and Exxon Mobile is: 0.18776806719129588
```

- Explanation at how to calculate one another and its difference

In order to calculate Correlation by Covariance, we use the following equation

$$Correlation = Covariance/(\sigma_{Telsa} * \sigma_{ExxonMobile})$$

Covariance describes how two variables move with respect to each other.

Correlation describes the link between two variables.

In [10]:
```
corr = covariance_tesla_exxon / (tesla_std * exxon_std)
print("The Correlation between Tesla and Exxon Mobile calculated based
on Covariance is: {}"
      .format(correlation_tesla_exxon))
```

```
The Correlation between Tesla and Exxon Mobile calculated based on Cov
ariance is: 0.18776806719129588
```

2. Build your own transition

```
- Divide the data into 2 uneven parts: the first part is 80% of your data, a
nd the second part is 20%.
```

In [11]:
```
# Divide data into 2 uneven part: 80-20
from sklearn.model_selection import train_test_split
tesla_80, tesla_20 = train_test_split(tesla, test_size=0.2)
exxon_80, exxon_20 = train_test_split(exxon, test_size=0.2)
```

- Categorize

In [12]:
```
# Create a new columns decide which direction the return go based on l
og return
def decide_up_down(log_r):
    if (log_r >0):
        result = "U"
    else:
        result = "D"
    return result

# Vectorize:
v_decide_up_down = np.vectorize(decide_up_down)

tesla_80["UD"] = v_decide_up_down(tesla_80["Log_Return"])
tesla_20["UD"] = v_decide_up_down(tesla_20["Log_Return"])
exxon_80["UD"] = v_decide_up_down(exxon_80["Log_Return"])
exxon_20["UD"] = v_decide_up_down(exxon_20["Log_Return"])

# We create portfolio consist of 2 data set: 80 and 20
def create_portfolio_ud(tesla, exxon):
    tesla["EX_UD"] = exxon["UD"]
    tesla.rename(columns={"UD": "TL_UD"}, inplace=True)
    portfolio = tesla[["TL_UD",'EX_UD']]
    portfolio.dropna(inplace=True)
    portfolio['Status'] = portfolio['TL_UD'] + portfolio['EX_UD']
    portfolio['Previous_Status'] = portfolio['Status'].shift(1)
    portfolio.dropna(inplace=True)
    return portfolio

portfolio_80 = create_portfolio_ud(tesla_80,exxon_80)
portfolio_20 = create_portfolio_ud(tesla_20,exxon_20)
```

In [13]:
```
portfolio_80.head()
```

Out[13]:

Date	TL_UD	EX_UD	Status	Previous_Status
2017-10-30	D	D	DD	UU
2016-12-15	D	U	DU	DD
2017-08-04	U	D	UD	DU
2016-09-28	U	U	UU	UD
2015-12-17	D	D	DD	UU

In [14]:
```
portfolio_20.head()
```

Out[14]:

	TL_UD	EX_UD	Status	Previous_Status
Date				
2015-08-14	U	D	UD	DD
2017-11-01	D	U	DU	UD
2016-04-12	D	U	DU	DU
2016-09-22	U	U	UU	DU
2015-12-09	D	U	DU	UU

- Build a transition matrix of portfolio direction

In [15]:
```
def movement(status, p_status):
    if (status == "UU" or status == "DD"):
        if (p_status == "UU" or p_status == "DD"):
            result = "mt_mt"
        else:
            result = "mt_ma"
    else:
        if (p_status == "UU" or p_status == "DD"):
            result = "ma_mt"
        else:
            result = "ma_ma"
    return result

v_movement = np.vectorize(movement)
portfolio_20["Movement"] = v_movement(portfolio_20['Status'],
                                      portfolio_20['Previous_Status'])
portfolio_80["Movement"] = v_movement(portfolio_80['Status'],
                                      portfolio_80['Previous_Status'])

matrix_20 = portfolio_20['Movement'].value_counts()
matrix_80 = portfolio_80['Movement'].value_counts()

def probablity(self, total):
    result = self / total
    return result

def transition_table(matrix):
    result = pd.DataFrame(columns=["Moving Together", "Moving Apart"])
    result.at['Moving Together', 'Moving Together'] = probablity(matri
x['mt_mt'], matrix.sum())
    result.at['Moving Together', 'Moving Apart'] = probablity(matrix[
'mt_ma'], matrix.sum())
    result.at['Moving Apart', 'Moving Together'] = probablity(matrix[
'ma_mt'], matrix.sum())
    result.at['Moving Apart', 'Moving Apart'] = probablity(matrix['ma_
ma'], matrix.sum())
    return result
```

In [16]:
```
transition_table(matrix_20)
```

Out[16]:

	Moving Together	Moving Apart
Moving Together	0.181818	0.212121
Moving Apart	0.212121	0.393939

In [17]:
```
transition_table(matrix_80)
```

Out[17]:

	Moving Together	Moving Apart
Moving Together	0.287815	0.254202
Moving Apart	0.254202	0.203782

- How similar is the transition matrix from the first group to the second group?

In both groups, the transition matrix show us that the total probability of the transition table at each row is not equal to 1, thus making the transition matrix less helpful.

- Is the process Markovian?

No. Because the sum of each row of the Moving Together and Moving Apart is not equal to 1, thus making it no Markovian process.

Also, the probablity was calculated from different periods of time, thus making its probability is not a whole set at a certain point/ total time.

In [ ]:
```

```