TASK 2

1. Unzip stock_data.zip, which contains historical market data for stocks in the S&P 500 index. The S&P 500 is a stock market index tracking the performance of 500 large companies listed on stock exchanges in the United States.

CODE

```
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import MinMaxScaler
import networkx as nx
import numpy as np
import matplotlib.pyplot as plt

print("-----------------1st---------------- \n")
# 1st part
dataframe=pd.read_csv("stock_data.csv")
dataframe['date']=pd.to_datetime(dataframe['date'])
dataframe2=dataframe[['date','open','high','low','close','volume','Name']]
```

2. Identify the set of all names in the data, and sort these names in alphabetical order. How many are there? List the first and last 5 names.

CODE

```
print("\n-----------------2nd---------------- \n")
#2nd part
dataframe2['date']=pd.to_datetime(dataframe2['date'])
#dataframe2['date']=pd.dt.date(dataframe2['date'])
df3=dataframe2
a=dataframe2.sort_values('Name')
a=a['Name'].unique()
print(a[:5])
print(a[-5:])
print("Total number of names",len(a))
```

OUTPUT

```
-----------------1st-----------------


-----------------2nd-----------------

['A' 'AAL' 'AAP' 'AAPL' 'ABBV']
['XYL' 'YUM' 'ZBH' 'ZION' 'ZTS']
Total number of names 505
```

3. Filter out all names for which the first date is after 1st Jan 2014 or the last date is before 31st Dec 2017. Which names were removed? How many are left?

CODE

```python
print("\n-----------------3rd-----------------\n")

#3rd part
df3=df3.sort_values(['Name','date'])
removed_stocks=df3.groupby('Name')['date'].agg(['first','last'])
#removed_stocks.reset_index()
removed_stocks=removed_stocks.loc[(removed_stocks['first']>'2014-01-01') | (removed_stocks['last']<'2017-12-31')]
removed_stocks=removed_stocks.reset_index()
print("Removed stocks ",removed_stocks)

removed_stock_list=[]
df4=dataframe.copy()

print("Dataframe 3 ", df3)

#Iterating through the length of stocks that are to be removed
for i in range(len(removed_stocks)):
    removed_stock_list.append(removed_stocks.loc[i,'Name'])


for i in range(len(removed_stock_list)):
    same_index=dataframe[dataframe['Name']==removed_stock_list[i]].index
    dataframe.drop(same_index,inplace=True)
```

```python
for i in range(len(removed_stock_list)):
    same_index=dataframe[dataframe['Name']==removed_stock_list[i]].index
    dataframe.drop(same_index,inplace=True)

print("Remaining stocks number",len(dataframe['Name'].unique()))
#print("Removed stock list", len(removed_stock_list))
print("Removed stock list", removed_stock_list)
```

OUTPUT

```
-----------------3rd-----------------

Removed stocks       Name      first       last
0    APTV 2017-12-05 2018-02-07
1     BHF 2017-07-17 2018-02-07
2    BHGE 2017-07-03 2018-02-07
3     CFG 2014-09-24 2018-02-07
4    CSRA 2015-11-16 2018-02-07
5    DWDP 2017-09-01 2018-02-07
6     DXC 2017-04-03 2018-02-07
7    EVHC 2016-12-02 2018-02-07
8     FTV 2016-07-01 2018-02-07
9    GOOG 2014-03-27 2018-02-07
10    HLT 2017-01-04 2018-02-07
11    HPE 2015-10-19 2018-02-07
12    HPQ 2015-10-19 2018-02-07
13   INFO 2014-06-19 2018-02-07
14    KHC 2015-07-06 2018-02-07
15   NAVI 2014-04-17 2018-02-07
16   PYPL 2015-07-06 2018-02-07
17   QRVO 2015-01-02 2018-02-07
18    SYF 2014-07-31 2018-02-07
19     UA 2016-04-07 2018-02-07
20   WLTW 2016-01-05 2018-02-07
21    WRK 2015-06-24 2018-02-07
```

```
Remaining stocks number 483
Removed stock list ['APTV', 'BHF', 'BHGE', 'CFG', 'CSRA', 'DWDP', 'DXC', 'EVHC', 'FTV', 'GOOG', 'HLT', 'HPE', 'HPQ', 'INFO', 'K
HC', 'NAVI', 'PYPL', 'QRVO', 'SYF', 'UA', 'WLTW', 'WRK']
```

4. Identify the set of dates that are common to all the remaining names. Remove all the dates that are before 1st Jan 2014 or after 31st Dec 2017. How many dates are there? What are the first and last 5 dates? (BELOW)

CODE

```python
print("\n----------------4th----------------\n")
#4th Part
first_names=dataframe2.groupby('Name').first()
removed_names=first_names[first_names['date']>pd.to_datetime('2014-01-01')].reset_index('Name')
removed_names_array=np.array(removed_names['Name'])
print(removed_names_array)

remaining_names=dataframe2.loc[~dataframe2['Name'].isin(removed_names_array)]
filtered_data=remaining_names[(remaining_names['date']>='2014-01-01')&(remaining_names['date']<='2017-12-31')]

grouped_data=filtered_data.groupby('Name')
indexed_group=grouped_data.size().index.tolist()
print(indexed_group)

similar_dates=pd.DataFrame()
prev_similar_dates=pd.DataFrame()

for index in indexed_group:
    present_similar_dates=grouped_data.get_group(index).date
    if not similar_dates.empty:
        prev_similar_dates=similar_dates
    if not prev_similar_dates.empty:
        similar_dates=pd.merge(prev_similar_dates,present_similar_dates,how='inner')
    prev_similar_dates=grouped_data.get_group(index).date
print("The length of common dates ", len(similar_dates))
print("The first five dates ", similar_dates.head(5))
print("The last five dates ", similar_dates.tail(5))
```

OUTPUT

```
----------------4th----------------

['APTV' 'BHF' 'BHGE' 'CFG' 'CSRA' 'DWDP' 'DXC' 'EVHC' 'FTV' 'GOOG' 'HLT'
 'HPE' 'HPQ' 'INFO' 'KHC' 'NAVI' 'PYPL' 'QRVO' 'SYF' 'UA' 'WLTW' 'WRK']
['A', 'AAL', 'AAP', 'AAPL', 'ABBV', 'ABC', 'ABT', 'ACN', 'ADBE', 'ADI', 'ADM', 'ADP', 'ADS', 'ADSK', 'AEE', 'AEP', 'AES', 'AE
T', 'AFL', 'AGN', 'AIG', 'AIV', 'AIZ', 'AJG', 'AKAM', 'ALB', 'ALGN', 'ALK', 'ALL', 'ALLE', 'ALXN', 'AMAT', 'AMD', 'AME', 'AM
G', 'AMGN', 'AMP', 'AMT', 'AMZN', 'ANDV', 'ANSS', 'ANTM', 'AON', 'AOS', 'APA', 'APC', 'APD', 'APH', 'ARE', 'ARNC', 'ATVI', 'A
VB', 'AVGO', 'AVY', 'AWK', 'AXP', 'AYI', 'AZO', 'BA', 'BAC', 'BAX', 'BBT', 'BBY', 'BDX', 'BEN', 'BF.B', 'BIIB', 'BK', 'BLK',
'BLL', 'BMY', 'BRK.B', 'BSX', 'BWA', 'BXP', 'C', 'CA', 'CAG', 'CAH', 'CAT', 'CB', 'CBG', 'CBOE', 'CBS', 'CCI', 'CCL', 'CDNS',
'CELG', 'CERN', 'CF', 'CHD', 'CHK', 'CHRW', 'CHTR', 'CI', 'CINF', 'CL', 'CLX', 'CMA', 'CMCSA', 'CME', 'CMG', 'CMI', 'CMS', 'C
NC', 'CNP', 'COF', 'COG', 'COL', 'COO', 'COP', 'COST', 'COTY', 'CPB', 'CRM', 'CSCO', 'CSX', 'CTAS', 'CTL', 'CTSH', 'CTXS', 'C
VS', 'CVX', 'CXO', 'D', 'DAL', 'DE', 'DFS', 'DG', 'DGX', 'DHI', 'DHR', 'DIS', 'DISCA', 'DISCK', 'DISH', 'DLR', 'DLTR', 'DOV',
'DPS', 'DRE', 'DRI', 'DTE', 'DUK', 'DVA', 'DVN', 'EA', 'EBAY', 'ECL', 'ED', 'EFX', 'EIX', 'EL', 'EMN', 'EMR', 'EOG', 'EQIX',
'EQR', 'EQT', 'ES', 'ESRX', 'ESS', 'ETFC', 'ETN', 'ETR', 'EW', 'EXC', 'EXPD', 'EXPE', 'EXR', 'F', 'FAST', 'FB', 'FBHS', 'FC
X', 'FDX', 'FE', 'FFIV', 'FIS', 'FISV', 'FITB', 'FL', 'FLIR', 'FLR', 'FLS', 'FMC', 'FOX', 'FOXA', 'FRT', 'FTI', 'GD', 'GE',
'GGP', 'GILD', 'GIS', 'GLW', 'GM', 'GOOGL', 'GPC', 'GPN', 'GPS', 'GRMN', 'GS', 'GT', 'GWW', 'HAL', 'HAS', 'HBAN', 'HBI', 'HC
A', 'HCN', 'HCP', 'HD', 'HES', 'HIG', 'HII', 'HOG', 'HOLX', 'HON', 'HP', 'HRB', 'HRL', 'HRS', 'HSIC', 'HST', 'HSY', 'HUM', 'I
BM', 'ICE', 'IDXX', 'IFF', 'ILMN', 'INCY', 'INTC', 'INTU', 'IP', 'IPG', 'IQV', 'IR', 'IRM', 'ISRG', 'IT', 'ITW', 'IVZ', 'JBH
T', 'JCI', 'JEC', 'JNJ', 'JNPR', 'JPM', 'JWN', 'K', 'KEY', 'KIM', 'KLAC', 'KMB', 'KMI', 'KMX', 'KO', 'KORS', 'KR', 'KSS', 'KS
```

```
BM', 'ICE', 'IDXX', 'IFF', 'ILMN', 'INCY', 'INTC', 'INTU', 'IP', 'IPG', 'IQV', 'IR', 'IRM', 'ISRG', 'IT', 'ITW', 'IVZ', 'JBH
T', 'JCI', 'JEC', 'JNJ', 'JNPR', 'JPM', 'JWN', 'K', 'KEY', 'KIM', 'KLAC', 'KMB', 'KMI', 'KMX', 'KO', 'KORS', 'KR', 'KSS', 'KS
U', 'L', 'LB', 'LEG', 'LEN', 'LH', 'LKQ', 'LLL', 'LLY', 'LMT', 'LNC', 'LNT', 'LOW', 'LRCX', 'LUK', 'LUV', 'LYB', 'M', 'MA',
'MAA', 'MAC', 'MAR', 'MAS', 'MAT', 'MCD', 'MCHP', 'MCK', 'MCO', 'MDLZ', 'MDT', 'MET', 'MGM', 'MHK', 'MKC', 'MLM', 'MMC', 'MM
M', 'MNST', 'MO', 'MON', 'MOS', 'MPC', 'MRK', 'MRO', 'MS', 'MSFT', 'MSI', 'MTB', 'MTD', 'MU', 'MYL', 'NBL', 'NCLH', 'NDAQ',
'NEE', 'NEM', 'NFLX', 'NFX', 'NI', 'NKE', 'NLSN', 'NOC', 'NOV', 'NRG', 'NSC', 'NTAP', 'NTRS', 'NUE', 'NVDA', 'NWL', 'NWS', 'N
WSA', 'O', 'OKE', 'OMC', 'ORCL', 'ORLY', 'OXY', 'PAYX', 'PBCT', 'PCAR', 'PCG', 'PCLN', 'PDCO', 'PEG', 'PEP', 'PFE', 'PFG', 'P
G', 'PGR', 'PH', 'PHM', 'PKG', 'PKI', 'PLD', 'PM', 'PNC', 'PNR', 'PNW', 'PPG', 'PPL', 'PRGO', 'PRU', 'PSA', 'PSX', 'PVH', 'PW
R', 'PX', 'PXD', 'QCOM', 'RCL', 'RE', 'REG', 'REGN', 'RF', 'RHI', 'RHT', 'RJF', 'RL', 'RMD', 'ROK', 'ROP', 'ROST', 'RRC', 'RS
G', 'RTN', 'SBAC', 'SBUX', 'SCG', 'SCHW', 'SEE', 'SHW', 'SIG', 'SJM', 'SLB', 'SLG', 'SNA', 'SNI', 'SNPS', 'SO', 'SPG', 'SPG
I', 'SRCL', 'SRE', 'STI', 'STT', 'STX', 'STZ', 'SWK', 'SWKS', 'SYK', 'SYMC', 'SYY', 'T', 'TAP', 'TDG', 'TEL', 'TGT', 'TIF',
'TJX', 'TMK', 'TMO', 'TPR', 'TRIP', 'TROW', 'TRV', 'TSCO', 'TSN', 'TSS', 'TWX', 'TXN', 'TXT', 'UAA', 'UAL', 'UDR', 'UHS', 'UL
TA', 'UNH', 'UNM', 'UNP', 'UPS', 'URI', 'USB', 'UTX', 'V', 'VAR', 'VFC', 'VIAB', 'VLO', 'VMC', 'VNO', 'VRSK', 'VRSN', 'VRTX',
'VTR', 'VZ', 'WAT', 'WBA', 'WDC', 'WEC', 'WFC', 'WHR', 'WM', 'WMB', 'WMT', 'WU', 'WY', 'WYN', 'WYNN', 'XEC', 'XEL', 'XL', 'XL
NX', 'XOM', 'XRAY', 'XRX', 'XYL', 'YUM', 'ZBH', 'ZION', 'ZTS']
The length of common dates  994
```

```
The length of common dates  994
The first five dates         date
0 2014-01-02
1 2014-01-03
2 2014-01-06
3 2014-01-07
4 2014-01-08
The last five dates           date
989 2017-12-22
990 2017-12-26
991 2017-12-27
992 2017-12-28
993 2017-12-29
```

5. Build a new pandas dataframe which has a column for each of the names from step (3) and a row for each of the dates from step (4). The dataframe should contain the “close” values for each corresponding name and date.

CODE

```
print("\n----------------5th-------------\n")
##5th Part
filtered_data=filtered_data.loc[filtered_data['date'].isin(similar_dates['date'])]
new_df=filtered_data.pivot(index='date',columns='Name', values='close')
print(new_df)
```

OUTPUT

```
----------------5th-------------

Name            A     AAL     AAP      AAPL   ABBV    ABC    ABT     ACN  \
date
2014-01-02  56.21  25.360  109.74   79.0185  51.98  69.89  38.23   81.13
2014-01-03  56.92  26.540  112.88   77.2828  52.30  69.94  38.64   81.40
2014-01-06  56.64  27.030  111.80   77.7042  50.39  69.69  39.15   80.54
2014-01-07  57.45  26.905  113.18   77.1481  50.49  70.45  38.85   81.52
2014-01-08  58.39  27.630  112.30   77.6371  50.36  71.14  39.20   82.15
...           ...     ...     ...       ...    ...    ...    ...     ...
2017-12-22  67.35  52.590  100.55  175.0100  98.21  92.46  56.93  153.89
2017-12-26  67.25  52.850  101.96  170.5700  97.75  93.25  57.00  152.99
2017-12-27  67.30  52.400   99.77  170.6000  98.09  92.60  57.47  153.32
2017-12-28  67.45  52.460   99.71  171.0800  97.79  92.59  57.46  153.57
2017-12-29  66.97  52.030   99.69  169.2300  96.71  91.82  57.07  153.09
```

```
Name          ADBE    ADI  ...     XL   XLNX     XOM   XRAY    XRX    XYL  \
date                       ...
2014-01-02   59.29  49.28  ...  31.25  45.97   99.75  47.96  47.64  34.16
2014-01-03   59.16  49.61  ...  30.81  45.62   99.51  48.19  47.96  34.47
2014-01-06   58.12  49.33  ...  30.37  45.42   99.66  47.90  48.36  34.41
2014-01-07   58.97  49.59  ...  30.37  45.52  101.07  48.64  48.76  34.51
2014-01-08   58.90  49.71  ...  30.39  45.91  100.74  48.73  48.32  34.49
...            ...    ...  ...    ...    ...     ...    ...    ...    ...
2017-12-22  175.00  88.85  ...  35.17  67.92   83.97  65.82  29.58  67.58
2017-12-26  174.44  88.63  ...  35.24  67.60   83.98  66.26  29.41  67.50
2017-12-27  175.36  89.10  ...  35.20  67.91   83.90  66.03  29.48  68.23
2017-12-28  175.55  89.38  ...  35.37  68.50   84.02  66.43  29.45  68.25
2017-12-29  175.24  89.03  ...  35.16  67.42   83.64  65.83  29.15  68.20
```

```
Name          YUM     ZBH   ZION    ZTS
date
2014-01-02  75.09   92.24  29.65  32.36
2014-01-03  75.56   92.64  29.86  32.05
2014-01-06  75.50   93.24  29.65  31.98
2014-01-07  76.56   95.10  29.74  32.10
2014-01-08  76.53   97.43  30.00  31.74
...           ...     ...    ...    ...
2017-12-22  82.40  120.12  51.33  71.99
2017-12-26  82.19  119.96  50.86  72.34
2017-12-27  82.40  120.14  50.71  72.45
2017-12-28  82.67  121.75  51.34  72.39
2017-12-29  81.61  120.67  50.83  72.04

[994 rows x 483 columns]
```

6. Create another dataframe containing returns calculated as:
   return(name, date) = (close(name, date) - close(name, previous date)) / close(name, previous date)
   Note that this dataframe should have one less row than the dataframe from step (5), because you can’t calculate returns for the first date (there’s no previous date).

CODE

```
print("\n-----------------6th-----------------\n")
#6th part
return_value=new_df.diff()/new_df.shift()
#return_value=return_value.drop(return_value.index[0])
return_value=return_value.drop(return_value.index[0]).fillna(0)
print(return_value)
```

OUTPUT

```
-----------------6th-----------------

Name               A       AAL       AAP      AAPL      ABBV       ABC  \
date
2014-01-03  0.012631  0.046530  0.028613 -0.021966  0.006156  0.000715
2014-01-06 -0.004919  0.018463 -0.009568  0.005453 -0.036520 -0.003574
2014-01-07  0.014301 -0.004624  0.012343 -0.007157  0.001985  0.010905
2014-01-08  0.016362  0.026947 -0.007775  0.006338 -0.002575  0.009794
2014-01-09  0.000343  0.064785  0.011131 -0.012772  0.017077  0.003374
...              ...       ...       ...       ...       ...       ...
2017-12-22 -0.002518 -0.003789  0.004195  0.000000  0.003064 -0.005700
2017-12-26 -0.001485  0.004944  0.014023 -0.025370 -0.004684  0.008544
2017-12-27  0.000743 -0.008515 -0.021479  0.000176  0.003478 -0.006971
2017-12-28  0.002229  0.001145 -0.000601  0.002814 -0.003058 -0.000108
2017-12-29 -0.007116 -0.008197 -0.000201 -0.010814 -0.011044 -0.008316
```

```
Name              ABT       ACN      ADBE       ADI  ...        XL      XLNX  \
date                                                 ...
2014-01-03  0.010725  0.003328 -0.002193  0.006696  ... -0.014080 -0.007614
2014-01-06  0.013199 -0.010565 -0.017579 -0.005644  ... -0.014281 -0.004384
2014-01-07 -0.007663  0.012168  0.014625  0.005271  ...  0.000000  0.002202
2014-01-08  0.009009  0.007728 -0.001187  0.002420  ...  0.000659  0.008568
2014-01-09  0.001786  0.009738  0.003226 -0.003822  ...  0.000329 -0.002832
...              ...       ...       ...       ...  ...       ...       ...
2017-12-22  0.000000 -0.002010  0.002521  0.002256  ...  0.003137 -0.006582
2017-12-26  0.001230 -0.005848 -0.003200 -0.002476  ...  0.001990 -0.004711
2017-12-27  0.008246  0.002157  0.005274  0.005303  ... -0.001135  0.004586
2017-12-28 -0.000174  0.001631  0.001083  0.003143  ...  0.004830  0.008688
2017-12-29 -0.006787 -0.003126 -0.001766 -0.003916  ... -0.005937 -0.015766
```

```
Name              XOM      XRAY       XRX       XYL       YUM       ZBH  \
date
2014-01-03 -0.002406  0.004796  0.006717  0.009075  0.006259  0.004337
2014-01-06  0.001507 -0.006018  0.008340 -0.001741 -0.000794  0.006477
2014-01-07  0.014148  0.015449  0.008271  0.002906  0.014040  0.019949
2014-01-08 -0.003265  0.001850 -0.009024 -0.000580 -0.000392  0.024501
2014-01-09 -0.009728  0.003694 -0.002483  0.004059 -0.019339 -0.009853
...              ...       ...       ...       ...       ...       ...
2017-12-22  0.001431  0.004885 -0.004376 -0.002509 -0.001212  0.001417
2017-12-26  0.000119  0.006685 -0.005747 -0.001184 -0.002549 -0.001332
2017-12-27 -0.000953 -0.003471  0.002380  0.010815  0.002555  0.001501
2017-12-28  0.001430  0.006058 -0.001018  0.000293  0.003277  0.013401
2017-12-29 -0.004523 -0.009032 -0.010187 -0.000733 -0.012822 -0.008871

Name             ZION       ZTS
date
2014-01-03  0.007083 -0.009580
2014-01-06 -0.007033 -0.002184
2014-01-07  0.003035  0.003752
2014-01-08  0.008742 -0.011215
2014-01-09  0.007333  0.006931
...              ...       ...
2017-12-22 -0.002526 -0.004012
2017-12-26 -0.009156  0.004862
2017-12-27 -0.002949  0.001521
2017-12-28  0.012424 -0.000828
2017-12-29 -0.009934 -0.004835

[993 rows x 483 columns]
```

7. Use the class sklearn.decomposition.PCA to calculate the principal components of the returns from step (6).

CODE

```
print ("------------7th-----------")
#7th part

pca=PCA()
pca.fit(return_value.values)
explained_v=pca.explained_variance_
print("explained variance ",explained_v[0:20])
explained_v_r=pca.explained_variance_ratio_
print("Explained variance ratio " ,explained_v_r[0:20])
```

OUTPUT

```
------------7th-----------
explained variance  [0.03404849 0.00734829 0.00437188 0.0029135  0.00235598 0.00190996
 0.00164795 0.00147048 0.00140845 0.00135982 0.00128138 0.00121279
 0.00109579 0.00104291 0.00100619 0.00099093 0.00093837 0.00089089
 0.00087131 0.00084314]
Explained variance ratio  [0.26488994 0.05716809 0.03401231 0.02266639 0.01832904 0.0148591
 0.01282072 0.01144004 0.01095745 0.01057907 0.00996888 0.00943521
 0.00852503 0.00811363 0.00782795 0.00770922 0.00730028 0.00693095
 0.00677864 0.00655948]
```

8. For the principal components calculated in step (7), extract the explained variance ratios (you can get these from the PCA object). What percentage of variance is explained by the first principal component? Plot the first 20 explained variance ratios. Identify an elbow and mark it on the plot. List your code for this question and provide a 1 paragraph description of it.

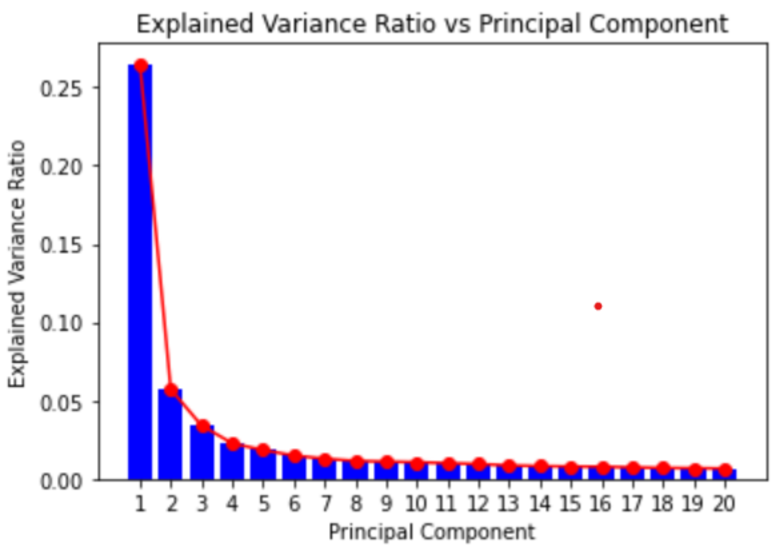
CODE

From the above result, we get to know for the first principal component, the **explained variance** is **0.03404849** and **explained variance ratio** is **0.26488994** which is about **26.489%.** From the result below we know that there is no significant changes **between 3rd and 4th** and hence the **elbow** might be present between them.

```
print("--------------8th-----------")
#8th part

x= range(1,len(explained_v_r[0:20])+1)
y= explained_v_r[0:20]

plt.bar(x,y,color='blue')
plt.plot(x,y,'o-',color='r')
plt.xlabel("Principal Component")
plt.ylabel("Explained Variance Ratio")
plt.title("Explained Variance Ratio vs Principal Component",fontsize=12)
plt.xticks(range(1,21))
plt.show()
```

OUTPUT

```
--------------8th-----------
```



9. Calculate the cumulative variance ratios using numpy.cumsum on the list of explained variance ratios from step (8). Plot all these cumulative variance ratios (x axis = principal

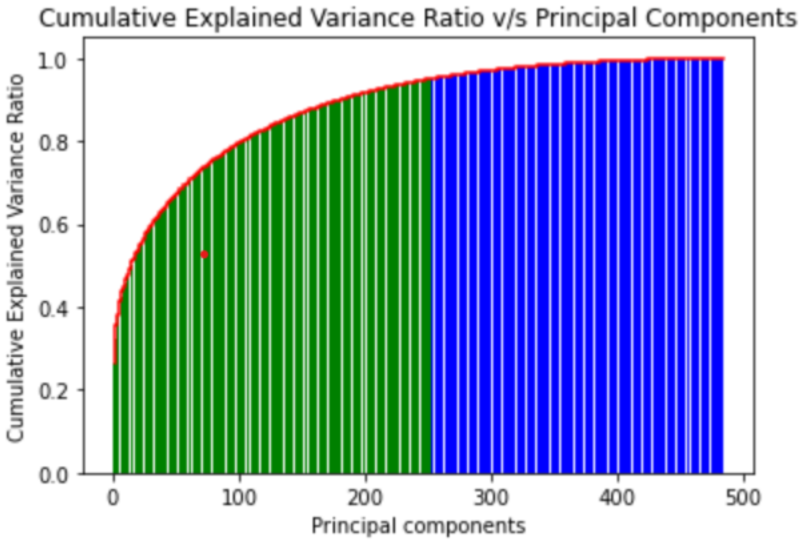
component, y axis = cumulative variance ratio). Mark on your plot the principal component for which the cumulative variance ratio is greater than or equal to 95%

CODE

```
print("-------------9th--------------")


x2=range(1,len(explained_v_r)+1)
y2=np.cumsum(explained_v_r)
percent=np.where(y2>=0.95,'blue','green')
plt.bar(x2,y2,color=percent)
plt.step(x2,y2,'-',color='r')
plt.title("Cumulative Explained Variance Ratio v/s Principal Components")
plt.xlabel("Principal components")
plt.ylabel("Cumulative Explained Variance Ratio")
plt.show()
```

OUTPUT

```
-------------9th--------------
```



10. Normalise your dataframe from step (6) so that the columns have zero mean and unit variance. Repeat steps (7) - (9) for this new dataframe.

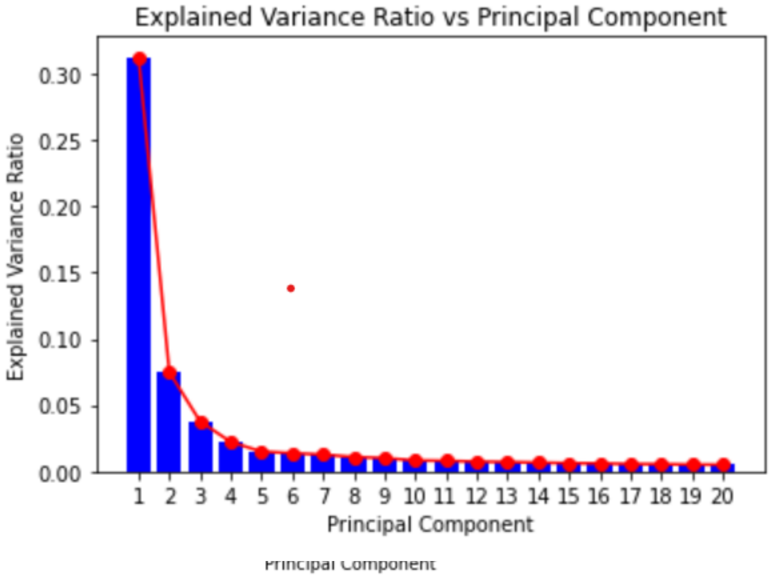
CODE

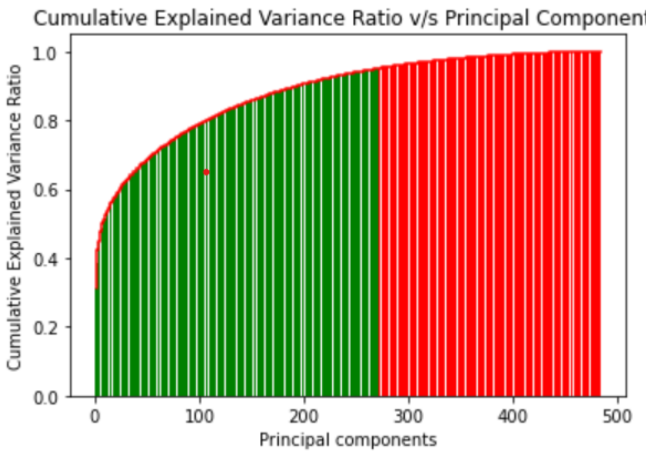
```
print("----------------10th-----------")


minmax=MinMaxScaler()
norm_df=minmax.fit_transform(return_value)

normalised_pca=PCA()
normalised_pca.fit(norm_df)
explained_normalised_v=normalised_pca.explained_variance_
print("explained variance ",explained_normalised_v[0:20])
explained_normalised_v_r=normalised_pca.explained_variance_ratio_
print("Explained variance ratio " ,explained_normalised_v_r[0:20])

#Bar Graph
x= range(1,len(explained_normalised_v_r[0:20])+1)
y= explained_normalised_v_r[0:20]

plt.bar(x,y,color='blue')
#plt.step(x,y,'--',color='g',where='mid')
plt.plot(x,y,'o-',color='r')
plt.xlabel("Principal Component")
plt.ylabel("Explained Variance Ratio")
plt.title("Explained Variance Ratio vs Principal Component",fontsize=12)
plt.xticks(range(1,21))
plt.show()
```

```
x2=range(1,len(explained_normalised_v_r)+1)
y2=np.cumsum(explained_normalised_v_r)
percent=np.where(y2>=0.95,'red','green')
plt.bar(x2,y2,color=percent)
plt.step(x2,y2,'-',color='r')
plt.title("Cumulative Explained Variance Ratio v/s Principal Components")
plt.xlabel("Principal components")
plt.ylabel("Cumulative Explained Variance Ratio")
plt.show()
```

OUTPUT

```
----------------10th-----------
explained variance  [1.05744371 0.25345163 0.12767582 0.07519056 0.05061481 0.04641487
 0.04302188 0.0366073  0.03411343 0.02793899 0.02709253 0.02492391
 0.02396995 0.02245957 0.02102452 0.02028507 0.01887098 0.01855235
 0.01736346 0.0171222 ]
Explained variance ratio  [0.31241237 0.07488004 0.03772069 0.02221438 0.0149537  0.01371286
 0.01271043 0.0108153  0.01007851 0.00825433 0.00800425 0.00736355
 0.00708171 0.00663548 0.00621151 0.00599304 0.00557526 0.00548113
 0.00512988 0.0050586 ]
```





FULL CODE

```
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import MinMaxScaler
import networkx as nx
import numpy as np
import matplotlib.pyplot as plt

print("-----------------1st----------------- \n")
# 1st part
dataframe=pd.read_csv("stock_data.csv")
dataframe['date']=pd.to_datetime(dataframe['date'])
dataframe2=dataframe[['date','open','high','low','close','volume','Name']]

print("\n-----------------2nd----------------- \n")
```

```
#2nd part
dataframe2['date']=pd.to_datetime(dataframe2['date'])
#dataframe2['date']=pd.dt.date(dataframe2['date'])
df3=dataframe2
a=dataframe2.sort_values('Name')
a=a['Name'].unique()
print(a[:5])
print(a[-5:])
print("Total number of names",len(a))

print("\n-----------------3rd-----------------\n")

#3rd part
df3=df3.sort_values(['Name','date'])
removed_stocks=df3.groupby('Name')['date'].agg(['first','last'])
#removed_stocks.reset_index()
removed_stocks=removed_stocks.loc[(removed_stocks['first']>'2014-01-01') |
(removed_stocks['last']<'2017-12-31')]
removed_stocks=removed_stocks.reset_index()
print("Removed stocks ",removed_stocks)

removed_stock_list=[]
df4=dataframe.copy()

print("Dataframe 3 ", df3)

#Iterating through the length of stocks that are to be removed
for i in range(len(removed_stocks)):
    removed_stock_list.append(removed_stocks.loc[i,'Name'])


for i in range(len(removed_stock_list)):
    same_index=dataframe[dataframe['Name']==removed_stock_list[i]].index
    dataframe.drop(same_index,inplace=True)

print("Remaining stocks number",len(dataframe['Name'].unique()))
#print("Removed stock list", len(removed_stock_list))
print("Removed stock list", removed_stock_list)


print("\n-----------------4th-----------------\n")
#4th Part
first_names=dataframe2.groupby('Name').first()
removed_names=first_names[first_names['date']>pd.to_datetime('2014-01-
01')].reset_index('Name')
removed_names_array=np.array(removed_names['Name'])
print(removed_names_array)

remaining_names=dataframe2.loc[~dataframe2['Name'].isin(removed_names_array)]
```

```
filtered_data=remaining_names[(remaining_names['date']>='2014-01-
01')&(remaining_names['date']<='2017-12-31')]

grouped_data=filtered_data.groupby('Name')
indexed_group=grouped_data.size().index.tolist()
print(indexed_group)

similar_dates=pd.DataFrame()
prev_similar_dates=pd.DataFrame()

for index in indexed_group:
    present_similar_dates=grouped_data.get_group(index).date
    if not similar_dates.empty:
        prev_similar_dates=similar_dates
    if not prev_similar_dates.empty:
        similar_dates=pd.merge(prev_similar_dates,present_similar_dates,how='i
nner')
    prev_similar_dates=grouped_data.get_group(index).date
print("The length of common dates ", len(similar_dates))
print("The first five dates ", similar_dates.head(5))
print("The last five dates ", similar_dates.tail(5))


print("\n-----------------5th--------------\n")
##5th Part
filtered_data=filtered_data.loc[filtered_data['date'].isin(similar_dates['date
'])]
new_df=filtered_data.pivot(index='date',columns='Name', values='close')
print(new_df)


print("\n-----------------6th-----------------\n")
#6th part
return_value=new_df.diff()/new_df.shift()
#return_value=return_value.drop(return_value.index[0])
return_value=return_value.drop(return_value.index[0]).fillna(0)
print(return_value)

print ("-------------7th-----------")
#7th part

pca=PCA()
pca.fit(return_value.values)
explained_v=pca.explained_variance_
print("explained variance ",explained_v[0:20])
explained_v_r=pca.explained_variance_ratio_
print("Explained variance ratio " ,explained_v_r[0:20])
```

```
print("--------------8th-----------")
#8th part

x= range(1,len(explained_v_r[0:20])+1)
y= explained_v_r[0:20]

plt.bar(x,y,color='blue')
plt.plot(x,y,'o-',color='r')
plt.xlabel("Principal Component")
plt.ylabel("Explained Variance Ratio")
plt.title("Explained Variance Ratio vs Principal Component",fontsize=12)
plt.xticks(range(1,21))
plt.show()

print("-------------9th--------------")


x2=range(1,len(explained_v_r)+1)
y2=np.cumsum(explained_v_r)
percent=np.where(y2>=0.95,'blue','green')
plt.bar(x2,y2,color=percent)
plt.step(x2,y2,'-',color='r')
plt.title("Cumulative Explained Variance Ratio v/s Principal Components")
plt.xlabel("Principal components")
plt.ylabel("Cumulative Explained Variance Ratio")
plt.show()

print("---------------10th-----------")


minmax=MinMaxScaler()
norm_df=minmax.fit_transform(return_value)

normalised_pca=PCA()
normalised_pca.fit(norm_df)
explained_normalised_v=normalised_pca.explained_variance_
print("explained variance ",explained_normalised_v[0:20])
explained_normalised_v_r=normalised_pca.explained_variance_ratio_
print("Explained variance ratio "  ,explained_normalised_v_r[0:20])

#Bar Graph
x= range(1,len(explained_normalised_v_r[0:20])+1)
y= explained_normalised_v_r[0:20]

plt.bar(x,y,color='blue')
#plt.step(x,y,'--',color='g',where='mid')
plt.plot(x,y,'o-',color='r')
plt.xlabel("Principal Component")
plt.ylabel("Explained Variance Ratio")
```

```
plt.title("Explained Variance Ratio vs Principal Component",fontsize=12)
plt.xticks(range(1,21))
plt.show()


x2=range(1,len(explained_normalised_v_r)+1)
y2=np.cumsum(explained_normalised_v_r)
percent=np.where(y2>=0.95,'red','green')
plt.bar(x2,y2,color=percent)
plt.step(x2,y2,'-',color='r')
plt.title("Cumulative Explained Variance Ratio v/s Principal Components")
plt.xlabel("Principal components")
plt.ylabel("Cumulative Explained Variance Ratio")
plt.show()
```

WEBSITE USED

TASK 2

https://stackoverflow.com/questions/37787698/how-to-sort-pandas-dataframe-from-one-column

https://stackoverflow.com/questions/32072076/find-the-unique-values-in-a-column-and-then-sort-them

https://www.marsja.se/pandas-convert-column-to-datetime/

https://stackoverflow.com/questions/13851535/how-to-delete-rows-from-a-pandas-dataframe-based-on-a-conditional-expression

https://stackoverflow.com/questions/37313691/how-to-remove-a-pandas-dataframe-from-another-dataframe

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop_duplicates.html

https://www.geeksforgeeks.org/different-ways-to-create-pandas-dataframe/

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.pivot.html

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.shift.html

https://www.geeksforgeeks.org/replace-nan-values-with-zeros-in-pandas-dataframe/

https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

https://www.interviewqs.com/ddi-code-snippets/rows-cols-python

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html