COMP336 Assignment1 Part2 Report

1. **Load the data into a pandas data frame.**
   - The shape of the data frame: [619040 rows x 7 columns].
2. **Identify the set of all names in the data, and sort these names in alphabetical order.**
   - The number of the different names: 505
   - The first 5 names are: ['A', 'AAL', 'AAP', 'AAPL', 'ABBV']
     The last 5 names are: ['XYL', 'YUM', 'ZBH', 'ZION', 'ZTS']
3. **Filter out all names for which the first date is after 1st Jan 2014 or the last date is before 31st Dec 2017.**
   - The number of the removed names: 22
     They are: ['APTV', 'BHF', 'BHGE', 'CFG', 'CSRA', 'DWDP', 'DXC', 'EVHC', 'FTV', 'GOOG', 'HLT', 'HPE', 'HPQ', 'INFO', 'KHC', 'NAVI', 'PYPL', 'QRVO', 'SYF', 'UA', 'WLTW', 'WRK']
   - The number of the remaining names: 483
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.**
   - The number of the remaining dates: 994
   - The first 5 dates: ['2014-01-02', '2014-01-03', '2014-01-06', '2014-01-07', '2014-01-08']
     The last 5 dates: ['2017-12-22', '2017-12-26', '2017-12-27', '2017-12-28', '2017-12-29']
5. **Build a new pandas data frame with "close" values.**
   - The shape of the new data frame: [994 rows x 483 columns].
6. **Create a data frame containing returns.**
   - The shape of the returns data frame: [993 rows x 483 columns].
7. **Calculate the principal components of the returns.**
   - The number of the components: 483

original

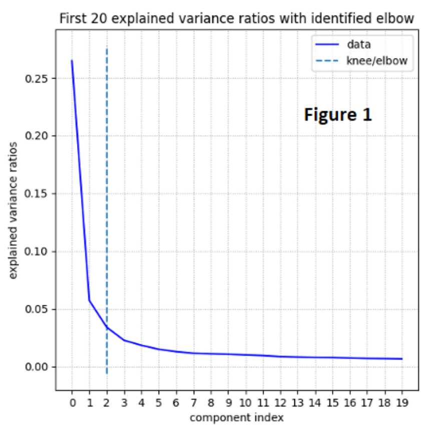

Figure 1

normalized

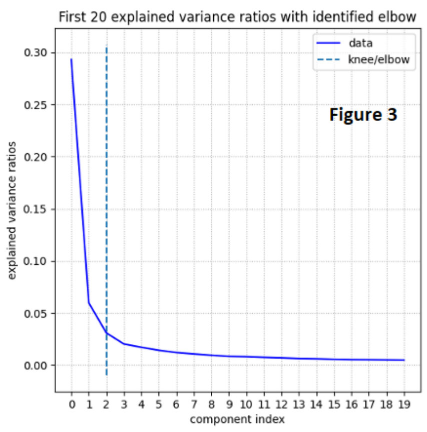

Figure 3

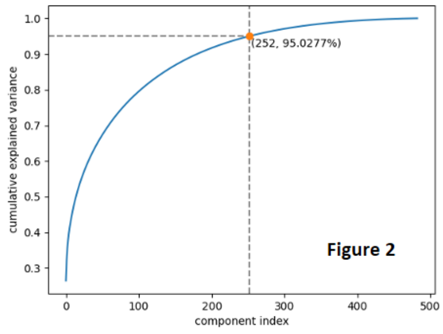

Figure 2

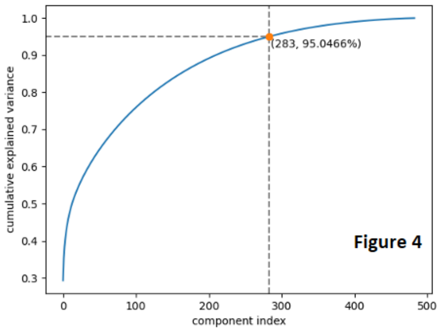

Figure 4

8. **Extract the explained variance ratios.**
   - The percentage of variance explained by the first principal component: 26.4890%
   - **Figure 1** plots the first 20 explained variance ratios with the elbow at x=2.

   Code Implementation:

```
from kneed import KneeLocator
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

pca = PCA()
pca.fit(returns)
# extract the explained variance ratios
explained_variance_ratio = pca.explained_variance_ratio_
# check the first ratio
print ("The first principle component explains {percentage_:.4%} of
variance.".format(percentage_=explained_variance_ratio[0]))
# Plot the first 20 explained variance ratios. Identify an elbow and mark it on the
plot
x = range(20)
y = explained_variance_ratio[:20]
kl = KneeLocator(x, y, curve="convex", direction='decreasing')
kl.plot_knee()
plt.xticks(range(0,20,1))
plt.xlabel('component index')
plt.ylabel('explained variance ratios')
plt.grid(linestyle=':')
plt.show()
```

   Code Explanation:
   The explained variance ratios are stored in 'explained_variance_ratio_' of sklearn PCA() class. The percentage of variance explained by the first principal component is exactly the first explained variance ratio with index 0. Then the first 20 ratios are extracted as y-axis to draw the graph, with the x-axis being the indices from 0 to 19. The elbow is identified by kneed.KneeLocator via feeding x, y to it.

9. **Plot the cumulative variance ratios.**
   - As marked in **Figure 2** above, the orange point shows that the first principal component for which the cumulative variance ratio is greater than or equal to 95% is the 253rd component (index=252).

10. **Normalize the returns data frame and repeat steps 7~9.**
   - The normalized returns is calculated by: (returns-returns.mean()) / returns.std()
   - The shape of the data frame and the number of the principal components stays the same.
   - The percentage of variance explained by the first principal component rise to: 29.2997%.
   - **Figure 3** shows the first 20 explained variance ratios and with an elbow at x=2.
   - **Figure 4** shows the cumulative variance ratios which exceed 95% at x=283.

Appendix – python code

```
import pandas as pd
import numpy as np
from functools import reduce
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from kneed import KneeLocator
from sklearn import preprocessing


print("\n -- 1) --")
stock_data_file="../data/stock_data.csv"
# load datafrom csv file
stock_data=pd.read_csv(stock_data_file, sep=',', header=0)
print (stock_data)

# stock_data.to_pickle('../output/stock_data.pkl')
# stock_data=pd.read_pickle('../output/stock_data.pkl')


print("\n -- 2) --")
# a sorted list of "name" without duplications
Name=sorted(set(stock_data['Name'].tolist()))
# print (Name)
print ("There are altogether {len_} names in the data.".format(len_=len(Name)))
print("The first 5 names are: {fisrt_}".format(fisrt_=Name[:5]))
print("The last 5 names are: {last_}".format(last_=Name[-5:]))


print("\n -- 3) --")
# convert the data in "date" into datetime dtype
stock_data['date'] = pd.to_datetime(stock_data['date'])

first_date=pd.to_datetime('2014-01-01')
# a dataframe containing the first date of all companies
first=stock_data.loc[stock_data.groupby('Name')['date'].idxmin()]
# a list of names that need to be deleted based on first date
first_filter = first[first["date"]>first_date]['Name'].tolist()
print("There're {len_} companies that need to be deleted based on first 
date".format(len_=len(first_filter)))

last_date=pd.to_datetime('2017-12-31')
# a dataframe containing the last date of all companies
last=stock_data.loc[stock_data.groupby('Name')['date'].idxmax()]
# a list of names that need to be deleted based on first date
last_filter = last[last["date"]<last_date]['Name'].tolist()
print("There're {len_} companies that need to be deleted based on last 
date".format(len_=len(last_filter)))

# the set of all the names that needs to be deleted
name_filter=sorted(set(first_filter).union(set(last_filter)))
print ("There're altogether {len_} companies that needs to be deleted. They 
are:".format(len_=len(name_filter)))
print (name_filter)

# a list of all the names after deletion
name_remaining = sorted(set(Name) - set(name_filter))
print ("After deleting the above names, there are {len_} names 
remaining.".format(len_=len(name_remaining)))
```

```
print("\n -- 4) --")
# delete the companies in (3) from the original dataset
stock_data = stock_data[~stock_data['Name'].isin(name_filter)]
print ("There're {len_} entries left in the dataset after deleting the unwanted
companies.".format(len_=stock_data.shape[0]))
# a dataframe containing a list of all dates for each company
all_dates = stock_data.groupby('Name')['date'].apply(list).reset_index(name="date")
# find the common dates (dtype=np.ndarray)
common_dates=np.unique(reduce(np.intersect1d, all_dates['date'])) # no duplicates here
actually
print ("There're {len_} common dates for the remaining companies, from {start_} to
{end_}.".format(len_=common_dates.shape[0], start_=common_dates[0].strftime('%Y-%m-%d'),
end_=common_dates[-1].strftime('%Y-%m-%d')))
# turn the common dates back into a dataframe
common_dates_df = pd.DataFrame(common_dates, columns=['date'])
# filter out the unwanted dates
date = common_dates_df[(common_dates_df['date'] >= first_date) & (common_dates_df['date']
<= last_date)]['date']
print ("There're {len_} valid dates after deleting at last.".format(len_=date.shape[0]))
print ("The first 5 dates: {first_}".format(first_=date.head(5).dt.strftime('%Y-%m-%d').tolist()))
print ("The last 5 dates:    {last_}".format(last_=date.tail(5).dt.strftime('%Y-%m-%d').tolist()))


print("\n -- 5) --")
stock_data = stock_data[stock_data['date'].isin(date)] # renew stock_data by valid common
dates, 480102 rows
dataset = pd.DataFrame(index=date, columns=name_remaining) # 994 * 483 = 480102
def fill_dataset(x):
        dataset.loc[x['date'],x['Name']] = x['close']
        return 0
stock_data.apply(lambda x: fill_dataset(x), axis=1)
# print (dataset.isnull().values.sum()) # 0


print("\n -- 6) --")
# calculate the return and delete the first row
returns = dataset.pct_change()[1:] # (current-previous)/previous

# returns.to_pickle('../output/returns.pkl')
# returns=pd.read_pickle('../output/returns.pkl')


print("\n -- 7) --")
pca = PCA()
# alculate the principal components of the returns
pca.fit(returns)
principle_components = pca.n_components_
print (principle_components) # 483


print("\n -- 8) --")
# extract the explained variance ratios
explained_variance_ratio = pca.explained_variance_ratio_
print ("The first principle component explains {percentage_:.4%} of
variance.".format(percentage_=explained_variance_ratio[0]))

# Plot the first 20 explained variance ratios. Identify an elbow and mark it on the plot
x = range(20)
y = explained_variance_ratio[:20]
```

```
kl = KneeLocator(x, y, curve="convex", direction='decreasing')
kl.plot_knee()
plt.xticks(range(0,20,1))
plt.xlabel('component index')
plt.ylabel('explained variance ratios')
plt.title('First 20 explained variance ratios with identified elbow')
plt.grid(linestyle=':')
# plt.savefig("../output/explained_variance_ratios_original.png", format="png")
plt.show()
plt.close()


print("\n -- 9) --")
# Calculate the cumulative variance ratios using numpy.cumsum
cumulative_variance_ratios = np.cumsum(pca.explained_variance_ratio_)

# find the first cumulative ratio that is greater or equal to 95%
x1 = 0
y1 = 0
for x1 in range(len(cumulative_variance_ratios)):
	if cumulative_variance_ratios[x1] >= 0.95:
		y1 = cumulative_variance_ratios[x1]
		break
	x1 += 1

plt.figure()
plt.plot(cumulative_variance_ratios)
plt.axhline(y=y1, xmax=x1/len(cumulative_variance_ratios),linestyle="--", color='0.5')
plt.axvline(x=x1,linestyle="--", color='0.5')
plt.plot(x1,y1,"8") # x1=252, actually is the 253th component
plt.text(x1+2, y1-0.03, '({}, {:.4%})'.format(x1,y1)) # adjust the position of text
plt.xlabel('component index')
plt.ylabel('cumulative explained variance')
# plt.savefig("../output/cumulative_explained_variance_original.png", format="png")
plt.show()
plt.close()


print("\n -- 10) --")
# standardize the values in each column
returns_normalized=(returns-returns.mean())/returns.std()
# print (returns_normalized)

# returns_normalized.to_pickle('../output/returns_normalized.pkl')
# returns_normalized=pd.read_pickle('../output/returns_normalized.pkl')


print("\n -- 10.7) --")
pca = PCA()
# alculate the principal components of the returns
pca.fit(returns_normalized)
principle_components = pca.n_components_
print (principle_components) # 483


print("\n -- 10.8) --")
# extract the explained variance ratios
explained_variance_ratio = pca.explained_variance_ratio_
print ("The first principle component explains {percentage_:.4%} of
variance.".format(percentage_=explained_variance_ratio[0]))
```

```python
# Plot the first 20 explained variance ratios. Identify an elbow and mark it on the plot
x = range(20)
y = explained_variance_ratio[:20]
kl = KneeLocator(x, y, curve="convex", direction='decreasing')
kl.plot_knee()
plt.xticks(range(0,20,1))
plt.xlabel('component index')
plt.ylabel('explained variance ratios')
plt.title('First 20 explained variance ratios with identified elbow')
plt.grid(linestyle=':')
# plt.savefig("../output/explained_variance_ratios_normalized.png", format="png")
plt.show()
plt.close()


print("\n -- 10.9) --")
# Calculate the cumulative variance ratios using numpy.cumsum
cumulative_variance_ratios = np.cumsum(pca.explained_variance_ratio_)

# find the first cumulative ratio that is greater or equal to 95%
x1 = 0
y1 = 0
for x1 in range(len(cumulative_variance_ratios)):
	if cumulative_variance_ratios[x1] >= 0.95:
		y1 = cumulative_variance_ratios[x1]
		break
	x1 += 1

plt.figure()
plt.plot(cumulative_variance_ratios)
plt.axhline(y=y1, xmax=x1/len(cumulative_variance_ratios),linestyle="--", color='0.5')
plt.axvline(x=x1,linestyle="--", color='0.5')
plt.plot(x1,y1,"8") # x1=283, actually is the 284th component
plt.text(x1+2, y1-0.03, '({}, {:.4%})'.format(x1,y1)) # adjust the position of text
plt.xlabel('component index')
plt.ylabel('cumulative explained variance')
# plt.savefig("../output/cumulative_explained_variance_normalized.png", format="png")
plt.show()
plt.close()
```