Mid-term Report

Hou LAN
41423018

1. Description
   This project is going to do a python project related to stock data based on dataframe: read files, do the preprocessing, calculate price changes and get the correlation and cluster (K-means and DBSCAN).
2. Data
   We use yahoo finance API to get the stock data of 73 firms from Dec. 31, 2015 to Jan. 08, 2016. (Tab.1 sample of stock data from yahoo finance)

Tab.1 first 5 lines of stock remote data

Date	**Open**	**High**	**Low**	**Close**	**Volume**	**Adj Close**	**tic**
2015/12/31	2.4	2.55	2.39	2.5	107100	2.5	COVS
2016/1/4	2.51	2.59	2.41	2.5	84200	2.5	COVS
2016/1/5	2.52	2.52	2.32	2.36	83900	2.36	COVS
2016/1/6	2.341	2.38	2.31	2.34	46800	2.34	COVS
2016/1/7	2.3	2.33	2.25	2.28	47100	2.28	COVS

```
start = datetime.datetime(2015,12,31)
end = datetime.datetime(2016,1,8)

lst_tic = list(firm_lst.COMN)
stock_df = pd.DataFrame()

for tic in lst_tic:
    try:
        f = web.DataReader(tic,'yahoo',start,end)
        f.reset_index(level=0,inplace=True)
        f['tic'] = tic
        stock_df = pd.concat([stock_df,f])
    except Exception:
    pass
```

3. Tasks

3.1. Calculate the Price Changes

The key function is pct_change() in dataframe.

```
stock_ret = pd.DataFrame()
for t in lst_tic:
        df_tmp = stock_df[stock_df.tic==t]
        df_tmp['ret'] = df_tmp['Adj Close'].pct_change()
        stock_ret = pd.concat([stock_ret,df_tmp])
```

3.2 Preprocessing

This step is designed for normalizing price data and return data in order to visualize the data in a beautiful way.

```
#preprocessing
stock_ret.index = stock_ret.Date
nullret_date = stock_ret.ix[stock_ret.ret.isnull(),'Date'].unique()
stock_ret.drop(nullret_date,axis=0,inplace=True)

#get normalized price
price_df = stock_df[["Adj Close", "tic"]].set_index("tic", append=True)['Adj
Close'].unstack("tic")
price_matrix = price_df.T.as_matrix()
price_matrix = preprocessing.scale(price_matrix.T).T
norm_price=pd.DataFrame(price_matrix.T,columns=price_df.columns,index=pri
ce_df.index)

#get normalized return
ret_df = stock_ret[["ret", "tic"]].set_index("tic", append=True).ret.unstack("tic")
ret_matrix = ret_df.T.as_matrix()
ret_matrix = preprocessing.scale(ret_matrix.T).T
norm_ret=pd.DataFrame(ret_matrix.T,columns=ret_df.columns,index=ret_df.ind
ex)
```

3.3 Visualization

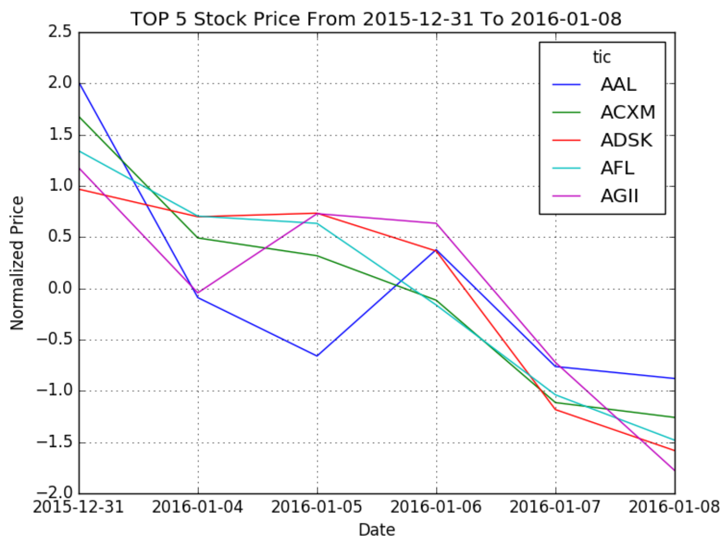

Fig.1 Normalized Prices of Top 5 Stocks

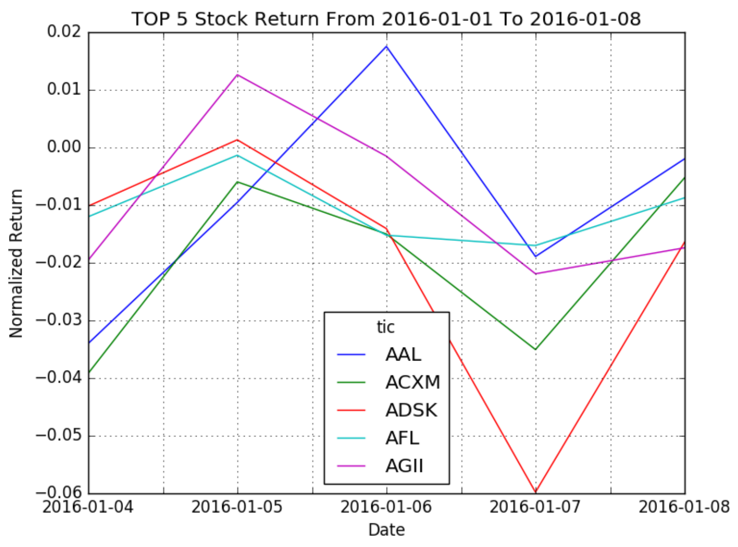

Fig.2 Normalized Returns of Top 5 Stocks

3.4 Get the Correleation Matrix

The most effective way to get the most related stocks is to use THE MAGIC dataframe's function "pd.corr". 'UFCS' and 'CHFC' is what we want.

```
#calculate the related matrix and find the most related two firms
corr_ret = norm_ret.corr(method='pearson')
corr_ret[corr_ret==1] = 0
conm = corr_ret.columns
idy, idx = np.where(corr_ret == corr_ret.max().max())
conm_1,conm_2 = conm[idx] # find most related two stocks
```

3.5 Cluster

We suppose to use normalized return to cluster.

3.5.1 K-means

① find the elbow point

Acorroding to Fig.3, we use 4 as the optimal K in K-means.

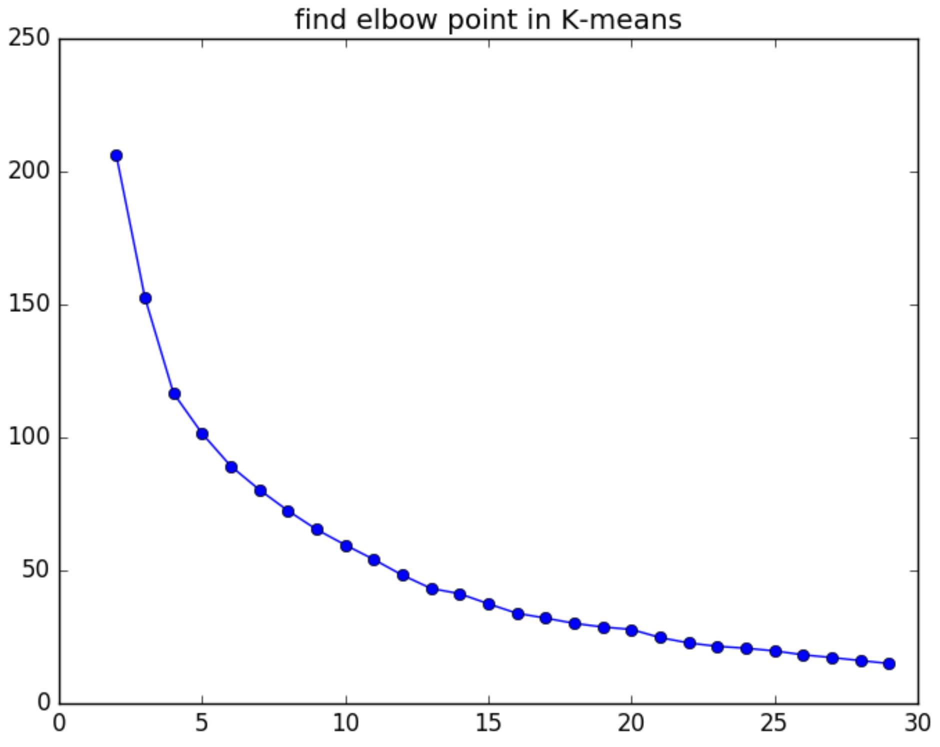

Fig.3 Finding the Elbow Point

②Use K-means from sklearn

3.5.2 DBSCAN

Based on the optimal K we find from last step, we use DBSCAN to cluster again, but the outcome isn't great since DBSCAN isn't quite efficent for clustering stock data.

① find the optimal eps.

We use the NearestNeighbors function and find 0.8 as optimal eps approximately.

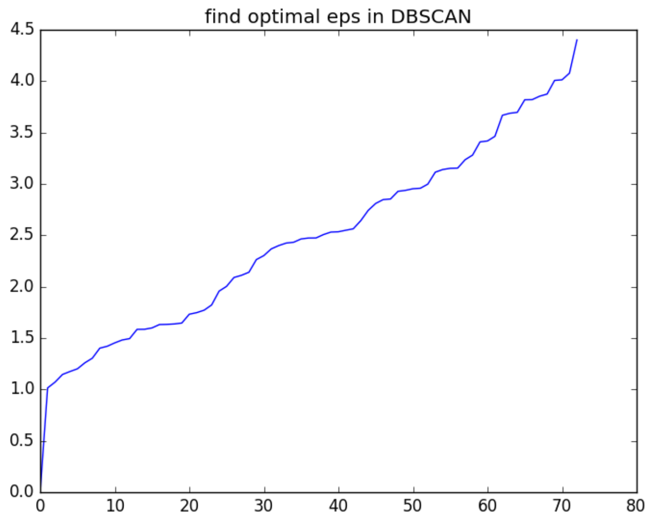

Fig.4 Finding the Optimal eps

```
nbrs = NearestNeighbors(n_neighbors=len(ret_matrix)).fit(ret_matrix)
distances, indices = nbrs.kneighbors(ret_matrix)
```

② Use DBSACN from sklearn

CONCLUSION：

According to silhouette_score, of course K-means is a better way to cluster stock data.