Program reskilling Data/AI PO
Specific session for SOCIETE GENERALE

4: Unsupervised learning: similarities

Eric Benhamou

PSL UNIVERSITÉ PARIS Dauphine UNIVERSITÉ PARIS ENS ÉCOLE NORMALE SUPÉRIEURE MINES ParisTech

PR[AI]RIE
PaRis Artificial Intelligence Research InstitutE

Agenda

Session 4: Clustering

Motivation with some examples

- Stock Classifications
- Investor classification
- Database issue

Clustering Models

- Clustering intuition
- Distance
- Models
- Case of K-means

LAB

Summary

Recall from previous session

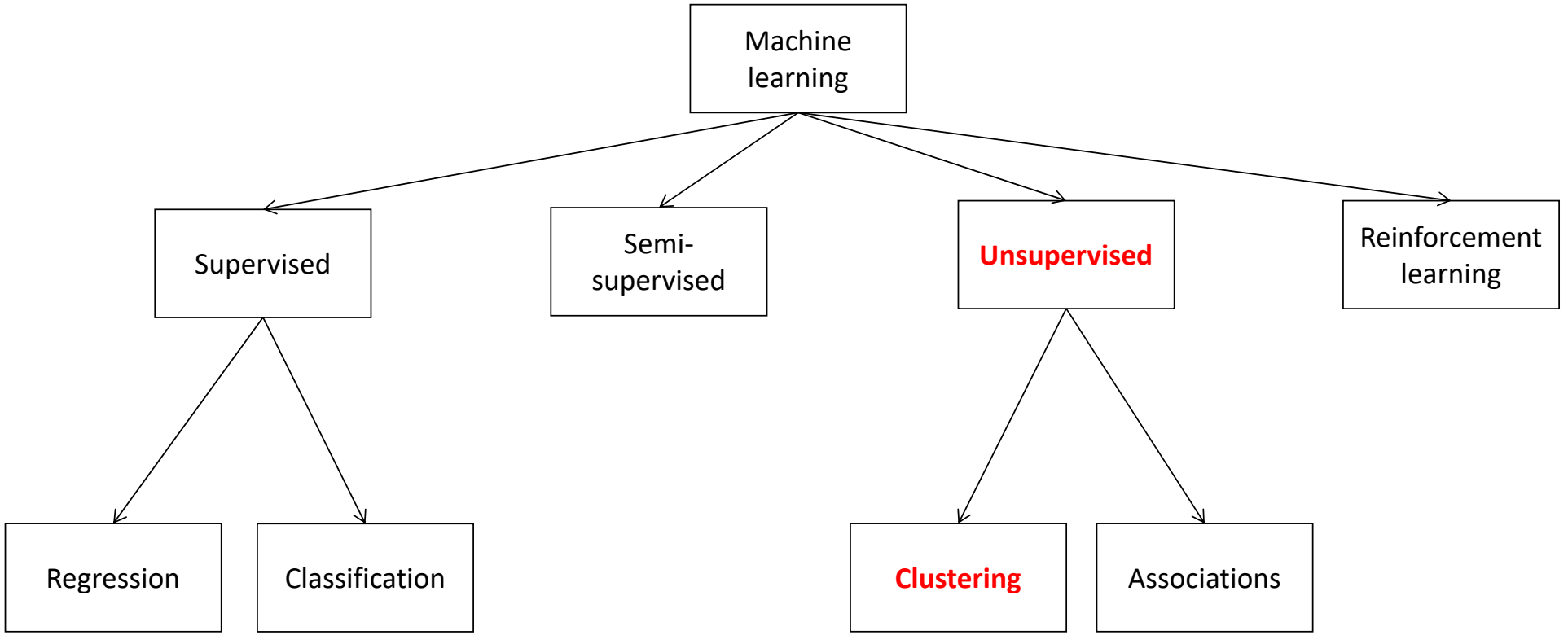
- In the previous session, you learned the different AI modeling techniques:
  - logistic regression model,
  - decision trees,
  - and deep learning.

- We also learned the various metrics of model performance :
  - ROC Curve
  - Confusion matrix
  - Classification report

Goals

- The goal of this chapter is to introduce **additional techniques** that can be used in banking namely clustering.
- This is a different type of machine learning that enables to rapidly identify and find **similarities** between data
- It is referred to as **unsupervised learning**

ML taxonomy


Machine learning
Supervised
Semi-supervised
Unsupervised
Reinforcement learning
Regression
Classification
Clustering
Associations

Various examples:

- Stock classification – style
- Investor classification
- Mergers and acquisitions identifications

- We will develop the first two examples to explain how AI can help.

Stock classification – style

Stock classification – style

There are two schools of thought when it comes to classifying stocks: one based on qualitative features and another based on quantitative features. We will be focusing on the qualitative approach, which is called **style**. An example of such a scheme is *Morningstar Style Box* (http://news.morningstar.com/pdfs/FactSheet_StyleBox_Final.pdf).

Here, we can look at the sector/industry, the size of the stocks, the riskiness of the stock, the potential of the stock, and so on. There are many ways to create features and classify stocks. We will use **sector and size as the features for qualitative classification** in this session.

The quantitative approach (for example, **arbitrage pricing theory** (APT)) groups stocks that contain similar factors together analytically.

Investor classification

- Like stock classification, there are both quantitative and qualitative approaches.
- Qualitative could be based on the type of money (pension, sovereign wealth, insurance, and so on), strategies (long-short, global macro, and so on), underlying holdings (futures, commodities, equities, bonds, and private equities), riskiness, and so on.
- Quantitative could be based on proximate factors that these investors are based on. In the first example of this session, we will use investment riskiness and return as the features for qualitative classification.

Database issue

- We are going to manage a large amount of data through the examples in this chapter. Due to this, it is critical to understand the underlying data technologies that we will use. These data technologies are related to storing varying types of data and information. There are two challenges related to information storage – first is the physical medium that we use to store the information, while the second is the format in which the information is stored.
- Hadoop is one such solution that allows stored files to be physically distributed. This helps us to deal with various issues such as storing a large amount of data in one place, backup, recovery, and so on. In our case, we store the data on one computer as the size does not justify using this technology, but the following NoSQL databases could support this storage option.
- In Python, there is another file format called HDF5, which also supports distributed filesystems.

SQL vs NoSQL

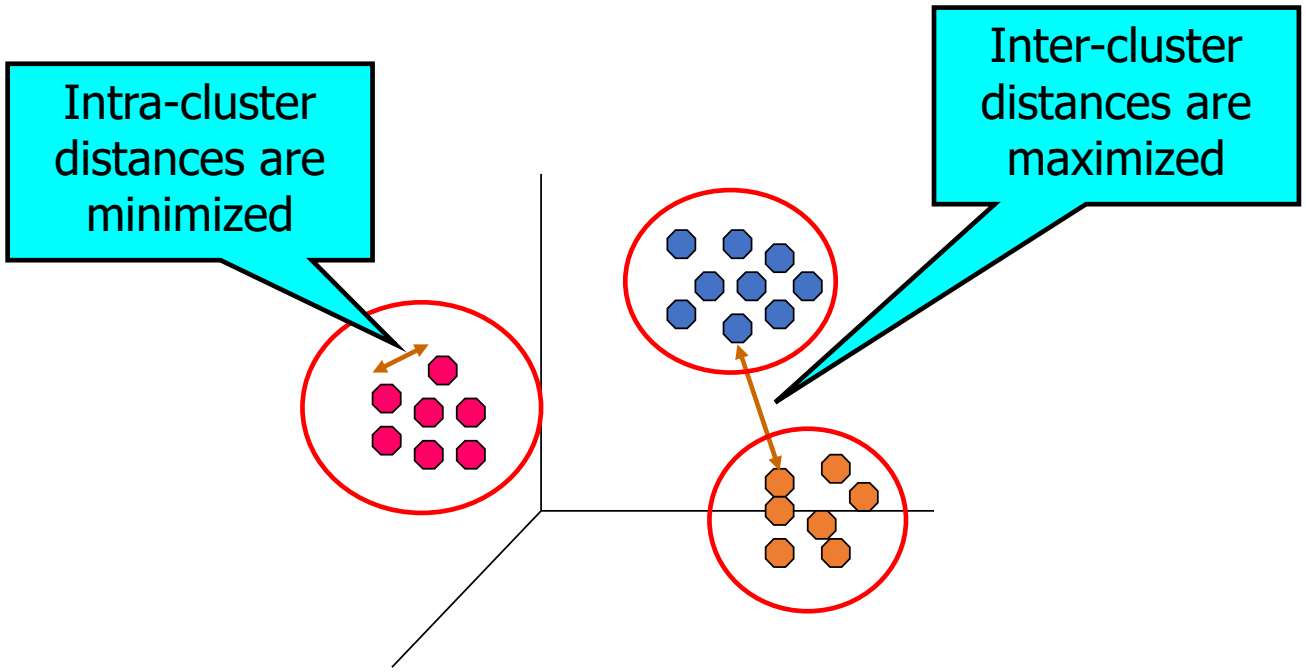
- While NoSQL databases can be used, the reason why I am not using them in this chapter can be explained with the help of the following table, which compares SQLite, Cassandra, and MongoDB side by side:

	Pros	Cons	Conclusions
SQLite	Structured data format, compatible with DataFrames	Cannot save unstructured data.	We need this for simplicity.
Cassandra	Can run at distributed computing and can put in structured data (with fields as items)	When dealing with structured data, the syntax is not straightforward to insert.	We can't use these for our case as we aim to cluster similar investors and predict who will buy our new issues in IPO.
MongoDB	Can handle huge data sizes and parallel processing of different records at scale	Not suitable for fully structured data such as trading records; still need to convert it into a DataFrame before running any machine learning algorithm.	

- Through this analysis, we see that it may not be necessary to have a NoSQL database for the sake of being cutting-edge. In the case of capital markets, where data is quite structured, it could be more efficient to use a SQL database that fits this purpose.

Clustering intuition

- Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups



What is a good clustering?

- A **good clustering** method will produce high quality clusters with
  - high **intra-class similarity**
  - low **inter-class similarity**
- The **quality** of a clustering result depends on both the similarity measure used by the method and its implementation
- The **quality** of a clustering method is also measured by its ability to discover some or all of the hidden patterns

Measuring the quality of clustering

- **Dissimilarity/Similarity** metric: Similarity is expressed in terms of a distance function, typically metric: *d(i, j)*

- There is a separate “quality” function that measures the “goodness” of a cluster.
- The definitions of **distance functions** are usually very different for interval-scaled, boolean, categorical, ordinal ratio, and vector variables.
- Weights should be associated with different variables based on applications and data semantics.
- It is hard to define “similar enough” or “good enough”. The answer is typically highly subjective.

Notion of clusters ambiguous

- How many clusters?

Two Clusters

Four Clusters

Six Clusters

Importance of standardizing data

Standardize data

Calculate the mean absolute deviation:

$$s_f = \frac{1}{n}(|x_{1f} - m_f| + |x_{2f} - m_f| + ... + |x_{nf} - m_f|)$$

where $$m_f = \frac{1}{n}(x_{1f} + x_{2f} + ... + x_{nf})$$

Calculate the standardized measurement (z-score)

$$z_{if} = \frac{x_{if} - m_f}{s_f}$$

Using mean absolute deviation is more robust than using standard deviation

Similarity and Dissimilarity Between Objects

- **Distances** are normally used to measure the similarity or dissimilarity between two data objects
- Some popular ones include: **Minkowski distance**

$$d(i,j)=\sqrt[q]{(|x_{i_1}-x_{j_1}|^q+|x_{i_2}-x_{j_2}|^q+...+|x_{i_p}-x_{j_p}|^q)}$$

where i = (xi1, xi2, …, xip) and j = (xj1, xj2, …, xjp) are two p-dimensional data objects, and q is a positive integer

- If q = 1, d is **Manhattan distance**

$$d(i,j)=|x_{i_1}-x_{j_1}|+|x_{i_2}-x_{j_2}|+...+|x_{i_p}-x_{j_p}|$$

And the Euclidean distance

- *If q = 2, d* is Euclidean distance

$$d(i,j) = \sqrt{(|x_{i_1} - x_{j_1}|^2 + |x_{i_2} - x_{j_2}|^2 + ... + |x_{i_p} - x_{j_p}|^2)}$$

  - Properties
    - $d(i,j) \geq 0$
    - $d(i,i) = 0$
    - $d(i,j) = d(j,i)$
    - $d(i,j) \leq d(i,k) + d(k,j)$

- Also, one can use weighted distance, parametric Pearson product moment correlation, or other dissimilarity measures

Clustering techniques

- Partitional
  - K-Means
  - Bisecting K-Means
  - K-Medoids
  - CLARA
  - CLARANS
- Hierarchical
  - Agglomerative
  - Divisive
- Density
  - DBSCAN

Clustering challenges

- One of the key challenges of adopting clustering in banking is that it leads to clusters that are too large, which reduces the true positive rate if all the clusters are targeted. As per my experience, I would use it for preliminary data analysis to understand the major dynamics of the target populations, not necessarily to draw actionable insights that make economic sense in a wholesale banking setting.

- In our example, we will create lots of clusters with the very stringent requirement that the distance of each data point from the centroid averages a 5% deviation

Clustering tips

- Another key question regarding the clustering algorithm is determining **how many features we feed it**.

- We could commit bias clustering by overweighing certain types of financial ratios (for example, using two different kinds of profitability ratios, such as return on equity and return on asset) for clustering.

- One of the solution to this is to run principle component analysis, which removes similar features by merging them into the same feature.

K means in more details

- The basic algorithm is very simple
- Number of clusters, K, must be specified
- Each cluster is associated with a centroid (mean or center point)
- Each point is assigned to the cluster with the closest centroid

1: Select K points as the initial centroids.
2: **repeat**
3: Form K clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: **until** The centroids don't change

K-means Clustering – Details

- Initial centroids are often chosen randomly. Clusters produced vary from one run to another.
- The centroid is (typically) the mean of the points in the cluster.
- ‘Closeness’ is measured by Euclidean distance, cosine similarity, correlation, etc.
- K-means will converge for common similarity measures mentioned above.
- Most of the convergence happens in the first few iterations.
  - Often the stopping condition is changed to ‘Until relatively few points change clusters’ or some measure of clustering doesn’t change.
- Complexity is O(n * K * I * d)
  - n = number of points,
  - K = number of clusters,

    I = number of iterations, d = number of attributes

Evaluating K-means Clusters

- Most common measure is Sum of Squared Error (SSE)
  - For each point, the error is the distance to the nearest cluster
  - To get SSE, we square these errors and sum them.

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

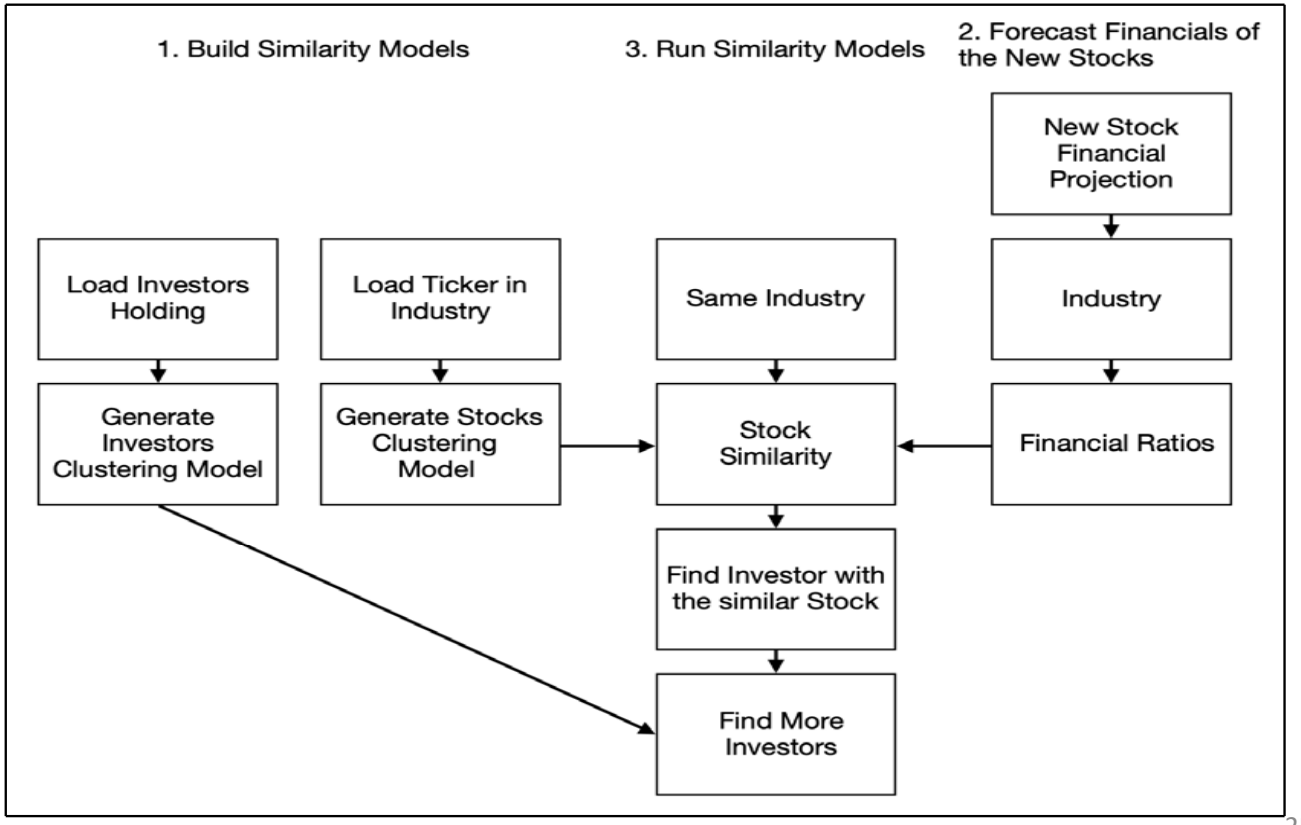
  - x is a data point in cluster C_i and m_i is the representative point for cluster C_i
    - can show that m_i corresponds to the center (mean) of the cluster
  - Given two clusters, we can choose the one with the smallest error
- One easy way to reduce SSE is to increase K, i.e. the number of clusters
  - A good clustering with smaller K can have a lower SSE than a poor clustering with higher K

Lab: Auto syndication for new issues

- If there are issues, there are investors behind them. Traditional investment banks will hire a group of professionals called the syndication desk to handle the allocation of security issues to investors who can buy these shares and bonds.
- If we consider the role of the syndication desk of the investment bank, our work will be to identify the cornerstone investors of the upcoming new issues with Duke Energy, as the CFO has the funding needs in equities. To do so, we will use the institutional holding data of US stocks from SEC filing via Quandl/Sharadar, which will help us find out the investment preferences of investors who share similar interests and match those with the investors who also hold similar stocks, such as Duke Energy.
- With regard to who to sell to, we will take the largest investors of US stocks as our universe of investors. The syndicated desk's job is to sell the major position of any equity issues to these investors. Using the unsupervised learning method, we recommend the relevant stocks to the right investors as an initial public offering. This can be done using securities similarities (called holding similarities) and investment styles (called investor similarities).

Solving the problem

- The following diagram shows the steps involved in solving the problem at hand:



Building similarity models

- Here, we will build two similarity models – one on stock similarity and another on finding similar investors. Both models are clustering models, and they belong to the last type of machine learning approach – unsupervised learning. We have picked 21 financial ratios to build the clustering model at the stock level, while for the investor model, we have a maximum of 60 features (*six capitalization sizes * five investment decisions * two types of indicators*):
  - Six capitalization scales: Nano, Micro, Small, Medium, Large, and Mega
  - Five investment decisions: Two for Buy (New, or Partial), one for Hold, and two for Sell (All or Partial)
  - Seven indicators: Quarterly return (total return, realized, unrealized), new money changing rate's mean and standard deviation, and current value

- Import all the relevant libraries and then load the ticker's universe by reading the CSV files together with the scale fields that describe the stocks. To reduce the processing time, load the investor lists instead of all the investors. For each investor, calculate the direction per market segment stock (that is, we use scale as the only market segment, but in reality, we should use country × industry × scale).

Code: import the required libraries and data

```
'''**********************
1a) Load Data
'''
#import relevant libraries
import quandl
from datetime import date,timedelta
import pandas as pd
import os
```

To avoid calling multiple times save data

```
def quandl_get_table(csv_filename, datatable_code, paginate=True, investorname='', calendardate='YYYY-MM-DD'):
    if not os.path.exists('quandl//'):
        os.makedirs('quandl//')

    if os.path.exists('quandl//' + csv_filename):
        return pd.read_csv('quandl//' + csv_filename, index_col=0, parse_dates=True)
    else:
        df = quandl.get_table(datatable_code, paginate=paginate,investorname=investorname,calendardate=calendardate)
        df.to_csv('quandl//' + csv_filename)
        return df
```

Code: load data 1/4

```
#load tickers universe and description field (scale)
print('load ticker universe')

df_tkr = pd.read_csv('industry_tickers_list.csv')


dict_scale_tkr = {}
for index, row in df_tkr.iterrows():
    scale = row['scalemarketcap']
    tkr = row['ticker']
    dict_scale_tkr[tkr] = [scale]

start_d =date(2018,1,1)
end_d =date(2018,1,5)

#loop through investors
quandl.ApiConfig.api_key = QUANDLKEY
#comment this out if you prefer the longer list
f_name = open('investors_select.txt','r')
#use this if you prefe the full list
#f_name = open('investors.txt','r')

investorNameList = f_name.readlines()

st_yr = 2013
end_yr = 2019
qtr_mmdd_list= ['-03-31','-06-30','-09-30','-12-31']
prev_investor = ""
prev_data_df = pd.DataFrame()
current_file_dir = os.path.dirname(__file__)
delta = timedelta(days=1)
print('prep investor movement')
```

Code: load data 2/4

```
for investor in investorNameList:
    investor = investor.rstrip('\n')
    print(investor)
    if os.path.exists('data//' + investor+'.csv'):
        print('already done, skipping!')
        continue

    curr_d = start_d
    investor_df = pd.DataFrame()
    data_df = pd.DataFrame()
    prev_investor_df = pd.DataFrame()
    prev_investor = ''
    #calculate the change in position by ticker on Quarter-to-quarter basis
    for yr_num in range(st_yr,end_yr):
        yr = str(yr_num)
        for mmdd in qtr_mmdd_list:
            dte_str = yr + mmdd
            print(dte_str)
            try:
                data_df = quandl_get_table(f"{investor}_data_df_{yr_num}{mmdd}.csv", "SHARADAR/SF3",
paginate=True,investorname=investor,calendardate=dte_str)
            except Exception:
                print('no data')
                continue
```

Code: load data 3/4

- ```
  if (len(data_df)>0 and len(prev_data_df)>0):
          df_combined = data_df.merge(prev_data_df, on='ticker')
          #fld_y is prev, fld_x is current
          df_combined['units_chg'] = df_combined['units_x'] - df_combined['units_y']
          df_combined['price_chg'] = df_combined['price_x'] - df_combined['price_y']
          if len(investor_df)==0:
             investor_df = df_combined
          else:
             investor_df = investor_df.append(df_combined)
       prev_data_df = data_df
  ```
  ```

Code: load data 4/4

```
#qualify investor's activities
print('classify investor decision for', investor)
investor_df['action'] =''
investor_df['scale'] = ''
i = 0
for index, row in investor_df.iterrows():
    try:
        this_scale = dict_scale_tkr[row['ticker']][0]
    except Exception:
        continue
    #is_scale = (this_scale in list_scale)
    if row['units_chg'] < 0:
        if row['units_x'] == 0:
            investor_df.at[index,'action']='SELL-ALL'
        else:
            investor_df.at[index,'action']='SELL-PARTIAL'
    elif row['units_chg'] > 0:
        if row['units_y'] == 0:
            investor_df.at[index,'action']='BUY-NEW'
        else:
            investor_df.at[index,'action']='BUY-MORE'
    else:
        investor_df.at[index,'action']='HOLD'
    investor_df.at[index,'scale']=this_scale
    i += 1
    if i == 1 or round(i/len(investor_df) *100)>round((i-1)/len(investor_df) *100):
        print(f'did {i/len(investor_df):.0%}')
#output the ticker's activities of the investor
output_path = os.path.join(current_file_dir,'data',investor+'.csv')
investor_df.to_csv(output_path)
print('saved', output_path)
```

Code: prepare data (investor profile) 1/2

```
'''
''''**********************
1b) Prepare investor Profile
'''
#load relevant libraries
import os
import pandas as pd
import numpy as np
from time import time
from sklearn import metrics
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import pickle

np.random.seed(42)

list_fld = ['investorname_x','calendardate_x','scale','action']
measure_fld =['value_x','value_y','realized_return','unrealized_return','new_money']
current_file_dir = os.path.dirname(__file__)

#Summarize quarterly performance of investors per quarter
input_path = os.path.join(current_file_dir,'data','investor_data')
file_list = os.listdir(input_path)
investor_pd = pd.DataFrame()
for file in file_list:
    print(file)
    if not file.endswith('.csv'):
        continue
    file_path = os.path.join(input_path,file)
    tmp_pd = pd.read_csv(file_path)
    tmp_pd['unrealized_return']=0
    tmp_pd['realized_return']=0
    tmp_pd['new_money']=0
    for index, row in tmp_pd.iterrows():
        #units_chg = row['units_x']-row['units_y']

        if row['units_chg'] > 0:
            realized_return = 0
            unrealized_return = row['units_y']*row['price_chg']
            new_money = row['units_chg'] * row['price_x']
        else: #sell off or hold
            realized_return = (-row['units_chg'])*(row['price_chg'])
            unrealized_return = row['units_y']*(row['price_chg'])
            new_money = 0
        tmp_pd.loc[index,'unrealized_return']=unrealized_return
        tmp_pd.loc[index,'realized_return']=realized_return
        tmp_pd.loc[index,'new_money']=realized_return
```

Code: prepare data (investor profile) 2/2

```
    #calculate return (realized, unrealized and new money)
    if len(tmp_pd)>0:
        tmp_pd_group = tmp_pd.groupby(list_fld)[measure_fld].sum()
        tmp_pd_group['return'] =
(tmp_pd_group['realized_return']+tmp_pd_group['unrealized_return'])/tmp_pd_group['value_y']
        tmp_pd_group['unrealized_return'] = (tmp_pd_group['unrealized_return'])/tmp_pd_group['value_y']
        tmp_pd_group['realized_return'] = (tmp_pd_group['realized_return'])/tmp_pd_group['value_y']
        tmp_pd_group['new_money'] = (tmp_pd_group['new_money'])/tmp_pd_group['value_y']
        tmp_pd_pivot =
tmp_pd_group.pivot_table(values=['value_x','realized_return','return','unrealized_return','new_money']
            ,index =['investorname_x'],columns=['scale','action'],aggfunc={'return':np.mean,'return':np.std,
            'realized_return':np.mean,'realized_return':np.std,
            'unrealized_return':np.mean,'unrealized_return':np.std,
            'value_x':np.sum})
        #tmp_pd_pivot = tmp_pd_group.pivot_table(values=['return'],index
=['investorname_x'],columns=['scale','action'],aggfunc={'return':np.mean})
        investor_pd = investor_pd.append(tmp_pd_pivot)

investor_pd.to_csv('investor_summary.csv')
```

Code: cluster data

```
'''************************
1c) Cluster investors
'''
#cleansed and transform data for clustering
investor_pd = investor_pd.replace([np.inf, ], 999999999)
investor_pd = investor_pd.fillna(0)
investor_pd = investor_pd.dropna()

sc_X = StandardScaler()
X = sc_X.fit_transform(investor_pd)

#define the k means function
def bench_k_means(estimator, name, data):
    t0 = time()
    cluster_labels = estimator.fit_predict(data)
    score = metrics.silhouette_score(data, cluster_labels, metric='euclidean')
    t1 = time()
    print('time spent :' +str(t1-t0))
    return score,cluster_labels

#try out different K means parameters and find out the best parameters
best_score = 1
best_cluster = 0
best_labels = pd.DataFrame()
track ={}
best_KMeans_model = KMeans()
for num_cluster in range(5, 500):
    KMeans_model = KMeans(init='k-means++', n_clusters=num_cluster, n_init=10)
    this_score,this_labels = bench_k_means(KMeans_model,
            name="k-means++", data=X)
    track[num_cluster] = this_score
    if this_score < best_score:
        best_score = this_score
        best_cluster = num_cluster
        best_KMeans_model = KMeans_model
        best_labels = this_labels
        print(num_cluster)
```

Code: output results

- '''**********
  1d) Output the results
  '''

```
#TO RUN
best_labels_pd = pd.DataFrame(best_labels)
best_labels_pd.columns = ['cluster']
X_pd = pd.DataFrame(X)
best_labels_data = pd.concat([X_pd,best_labels_pd],axis=1)

#Output clusters
f_cluster=open('investor_cluster_'+str(best_cluster)+'.pkl',"wb+")
pickle.dump(best_KMeans_model, f_cluster)
f_cluster.close()

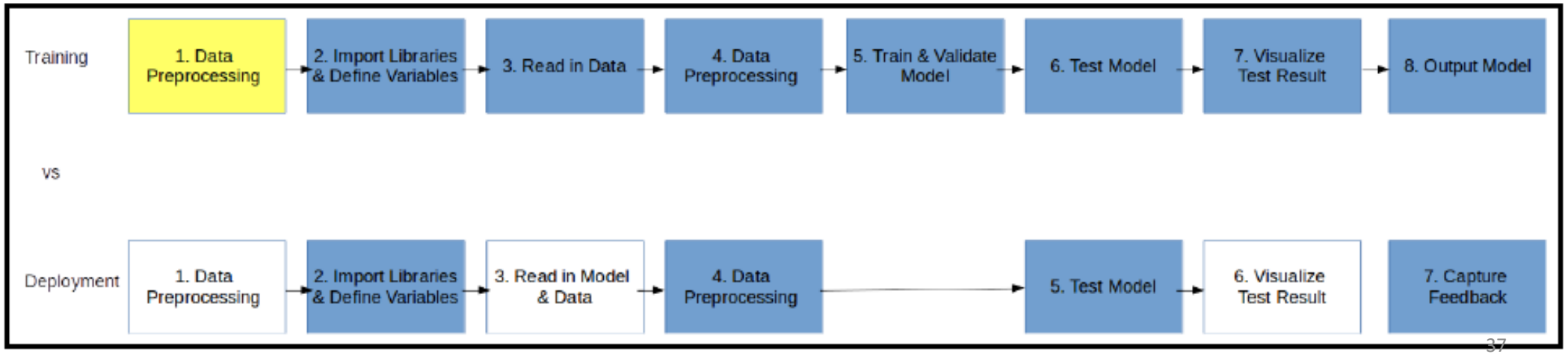
f_SC=open('investor_SC_'+str(best_cluster)+'.pkl',"wb+")
pickle.dump(sc_X, f_SC)
f_SC.close()

f_labels=open('investor_labels_'+str(best_cluster)+'.pkl',"wb+")
pickle.dump(best_labels_data, f_labels)
f_labels.close()
```

Well done!

Congrats!

- You **have now delivered a model** that can be used at the operational level for clustering stock types



Summary

- In this lecture, we learned about supervised learning and clustering in particular to help find similarities between data

- We discuss the concept of distance and the subjectivity of number of clusters

- In the lab, we presented KMeans