Statistical Machine Learning with Python Week #1

Ching-Shih Tsou (Ph.D.), Distinguished Prof. at the Department of Mechanical Engineering/Director of the Center for Artifical Intelligence & Data Science, Ming Chi University of Technology

August/2020 at MCUT

Outline

Week #1

- Business Problems and Data Mining Tasks
- Dimensionality Reduction and Principal Component Analysis
- Clustering Analysis

Week #2

- Association Rule Mining
- k Nearest Neighbors
- Tree-Based Models (Classification Trees, Regression Trees, and Model Trees incl.)

Week #3

- Naïve Bayes Classification (Text Mining incl.)
- Support Vector Machines
- *Bagging, Boosting, and Random Forest
- Confusion Matrix [Slides from ROC]
- ROC Curve, AUC, Lift Chart

Prob.&Stats. + Data Mining + Machine Learning = Data Analysis + Computer Programming

Business Problems and Data Mining Tasks

- Each data-driven business decision-making problem is unique, comprising its own combination of goals, desires, constraints, and even personalities.
- Similar to engineering problems, though, there are sets of common tasks that underlie the business problems.
- Data scientists usually decompose a business problem into subtasks. The solutions to the subtasks can then be composed to solve the overall problem. (Decomposition of business problems and re-composition of solutions)
- Some of these subtasks are unique to the particular business problem, but others are common data mining tasks.
- Despite the large number of specific data mining algorithms developed over the years, there are only a handful of fundamentally different types of tasks these algorithms address.The following will explain these basic tasks.

Task 1. Data reduction (橫向精簡與縱向降維，後者又稱為屬性挑選與萃取)

```
 take a large set of data and replace it with a smaller set of data that conta
ins much of the important information in the larger set.
```

- The smaller dataset may be easier to deal with or to process.
- For example, a massive dataset on consumer movie-viewing preferences may be reduced to a much smaller dataset revealing the consumer taste preferences that are latent in the viewing data (for example, viewer genre preferences).
- Data reduction usually involves loss of information. What is important is the trade-off for improved insight.

Task 2. Similarity matching

attempts to identify similar individuals based on data known about them. Similarity matching can be used directly to find similar entities. (The basic logic of problem-solving: find the similarity among different objects and discover the dissimilarity from similar things 問題解決的基本邏輯：異中求同、同中求異)

- For example, IBM is interested in finding companies similar to their best business customers, in order to focus their sales force on the best opportunities. They use similarity matching based on “firmographic” data describing characteristics of the companies.
- Similarity matching is the basis for one of the most popular methods for making product recommendations (finding people who are similar to you in terms of the products they have liked or have purchased).
- Similarity measures underlie certain solutions to other data mining tasks, such as classification, regression, and clustering. 。

Task 3. Profiling (aka. behavior description)

```
attempts to characterize the typical behavior of an individual, group, or pop
ulation.
```

- An example profiling question would be: “What is the typical cell phone usage of this customer segment?”
- Behavior may not have a simple description; profiling cell phone usage might require a complex description of night and weekend airtime averages, international usage, roaming charges, text minutes, and so on.
- Profiling is often used to establish behavioral norms for anomaly detection applications such as fraud detection and monitoring for intrusions to computer systems.
- For example, if we know what kind of purchases a person typically makes on a credit card, we can determine whether a new charge on the card fits that profile or not. We can use the degree of mismatch as a suspicion score and issue an alarm if it is too high.

Task 4. Clustering

attempts to group individuals in a population together by their similarity

- An example clustering question would be: “Do our customers form natural groups or segments?”
  - Clustering is useful in preliminary domain exploration to see which natural groups exist because these groups in turn may suggest other data mining tasks or approaches.
- Clustering also is used as input to decision-making processes.
  - for example:What (example:What) products should we offer or develop? How should our customer care teams (or sales teams) be structured?

Task 5. Co-occurrence grouping (or affinity grouping)

also known as frequent itemset mining, association rule discovery, and market-basket analysis, attempts to find associations between entities based on transactions involving them.

- For example: what items are commonly purchased together?
- While clustering looks at similarity between objects based on the objects’ attributes, co-occurrence grouping considers similarity of objects based on their appearing together (by frequency counting) in transactions.
- Deciding how to act upon this discovery might require some creativity, but it could suggest a special promotion, product display, or combination offer.
- Some recommendation systems also perform a type of affinity grouping by finding, for example, pairs of books that are purchased frequently by the same people.
- Co-occurrence situation include co-occurrence frequency(support of association rule), and surprisingness(novelty degree of association rule)

Task 6. Classification and class probability estimation

```
attempt to predict, for each individual in a population, which of a (small) se
t of classes this individual belongs to. Usually the classes are mutually excl
usive.
```

- Classification problem : “Among all the customers of MegaTelCo, which are likely to respond to a given offer?” In this example the two classes could be called will respond and will not respond.
- For a classification task, a data mining procedure produces a model that, given a new individual, determines which class that individual belongs to. 例如：流失或不會流失、違約或不會違約、購買或不會購買
- A closely related task is scoring or class probability estimation. A scoring model applied to an indi- vidual produces, instead of a class prediction, a score representing the probability (or some other quantification of likelihood) that that individual belongs to each class.
- Classification and scoring are very closely related; as we shall see, a model that can do one can usually be modified to do the other.

Task 7. Value estimation

```
 attempts to estimate or predict, for each individual, the numerical value of
some variable for that individual.
```

- regression question : How much will a given customer use the service? The property (variable) to be predicted here is service usage, and a model could be generated by looking at other or similar individuals in the population and their historical usage.
- Regression is related to classification, but the two are different.
    - Classification : predicts **whether** something will happen
    - Regression : predicts **how much** something will happen

Task 8. Link prediction (one of the applications of graph mining 圖形資料探勘的用途之一)

```
 attempts to predict connections between data items, usually by suggesting tha
t a link should **exist**, and possibly also estimating the **strength** of th
e link.
```

- Link prediction is common in social networking systems: “Since you and Karen share 10 friends, maybe you’d like to be Karen’s friend?” 。
- Link prediction can also estimate the strength of a link. For example, for recommending movies to customers one can think of a graph between customers and the movies they’ve watched or rated. Within the graph, we search for links that do not exist between customers and movies, but that we predict should exist and should be strong. These links form the basis for recommendations.

Task 9. Causal modeling

```
  attempts to help us understand what events or actions actually influence oth
ers.
```

- Use predictive modeling to target advertisements to consumers, and we observe that indeed the targeted consumers purchase at a higher rate subsequent to having been targeted.
- Was this because the advertisements influenced the consumers to purchase? Or did the predictive models simply do a good job of identifying those consumers who would have purchased **anyway**? (ha ha!)
- Causal modeling include experiments (A/B tests) and observational method; they attempt to understand what would be the difference between the situations—which cannot both happen—where the "treatment" event were to happen, and were not to happen.
- In all cases, a careful data scientist should always include with a **causal** conclusion the exact assumptions that must be made in order for the causal conclusion to hold (there always are such assumptions—always ask). (許多模型背後通常有假設，資料科學家要了解假設可能使得分析結論是無效的。因為你的資料與情境，很可能不符合模型的假設！)

Statistics & Industrial Control

- Open-loop PID (proportion-integral-derivative) control
    - $u(t) = K_P \left(e(t) + \frac{1}{T_t} \int_0^t e(t)dt + T_D \frac{de(t)}{dt} \right)$
- From PID control to data-driven closed-loop control

Capability for Analyzing Big Data (大數據之回歸基本面)

What do we need to have?

- Data sensitive (資料有感)
    - What kind of computation and visualization can we do under nominal scale, order scale, interval scale, and ratio scale? Under structural and ill-structural data? One more step towards modeling approach and algorithms.
- Data mashups (資料混搭)
    - Get the right meaning of different kind of data - record, graph, sequence, text, audio, video, and try to mix them together in your analysis. (記錄資料、圖形資料、有序資料、文字、聲音、影片)
- Models mashups (模型混搭)
    - From Statistics and Machine Learning to the backdrop hung by Operations Research. (統計、機器學習、作業研究或稱運籌學)
- Prototyping tools (雛形化工具)
    - Hands-on through R, Python, Julia … Learning by doing, doing along with learning. (學中做、做中學)
- Fusion with other information technologies (其它IT技術)
    - Linux, Web, Cloud, Hadoop, Spark, NoSQL … Learning will never end up. (不斷超越)
- *All built on business understanding* (商業理解)！

Three Types of Model

Model Category	Form of $f(\cdot)$	Independent Variables	Common Techniques
Prescriptive (or Normative)	known, well-defined	known or under decision maker's control	LP, Networks, IP, CPM, EOQ, NLP, GP, MOLP
Predictive	unknown, ill-defined	known or under decision maker's control	Regression Analysis, Time Series Analysis, Discriminant Analysis
Descriptive	known, well-defined	unknown or uncertain	Simulation, PERT, Queueing, Inventory Models

Hands-on Cases Oriented Course Design (案例實作導向的課程設計)

Motivation

- The data mining (or machine learning, or predictive modeling process) is inherently hands-on. 資料探勘(或是機器學習、預測建模)本質上是實作導向的。
- An article about computational science in a scientific publication is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures. 在科學的出版品中的文章通常不是學術，比較像是學者的廣告。真正的學術是完整的軟體開發環境，與產生圖形的完整指令集。
- from Buckheit, J. and Donoho, D.L. (1995), “WaveLab and Reproducible Research”, in A. Antoniadis, G. Oppenheim (eds.), “Wavelets in Statistics”, pp. 55-82, Springer-Verlag, New York.

Programming Languages in Computer Science

> Fourth Generation Language

- PHP, R, Python, …
- They are dynamic and evolvable. Don’t be surprised, there may have new packages iploaded just after our classes end.

Third Generation Language

- Fortran, Basic, Pascal, C/C++, Java, …
- Static and not so much changes after ten years.

Second Generation Language

- Assembler
- Whole groups of bit operations are assembled.

First Generation Language

- Machine code
- Programming a computer at 0 and 1 states or bits is possible
- You must highlight the differences between the fourth and the third generation programming languages during the learning of data-driven programming.

Object-Oriented Programming in Python

Most Python scripts look like as follows:

Import relevant class function first.

- from sklearn.naive_bayes import MultinomialNB

Define the model you want to build. It’s a model with unknown parameters.

- clf = MultinomialNB()

Input training data and Python will fit the model. We are going to have a parameterized model.

- clf.fit(sms_dtm_train, sms_raw_train[‘type’])

Use the model to find the predictions.

- pred = clf.predict(sms_dtm_test)

Please carefully differentiate the following

things for each line of Python script. (仔細區辨下列名稱)

- Class name or type name 類別名
- Object name defined by ourselve (ususally located at the left hand side of assignment operators, such as “=”) 自訂物件名(通常是<-的左方, 或函數內=的右方)
- Method name 方法名
- Function name 函數名
- Argument names in a function are usually omitted, but it is not a good practice for you. Attention to the argument values please. 引數名稱(經常省略！建議初學者勿省略)與引數值
- Other reserved names (保留字)

Functional Programming in R

Most R scripts look like as follows:

- kmeans.results <- kmeans(x = iris2, centers = 3)
- Object <- function(argument1=value1, argument1=value1)
- 自定物件名 <- 函數名(引數名1＝引數值1或自訂物件名1,引數名2＝引數值2或自訂物件名2,…)

Please carefully differentiate the following things for each line of R script (仔細區辨下列名稱)

- Object name defined by ourselve (ususally located at the left hand side of assignment operators, such as “<-” or “=”) 自訂物件名(通常是<-的左方, 或函數內=的右方)
- Function name 函數名
- Argument names in a function are usually omitted, but it is not a good practice for you. Attention to the argument values please. 引數名稱(經常省略！建議初學者勿省略)與引數值
- Other reserved names (保留字)

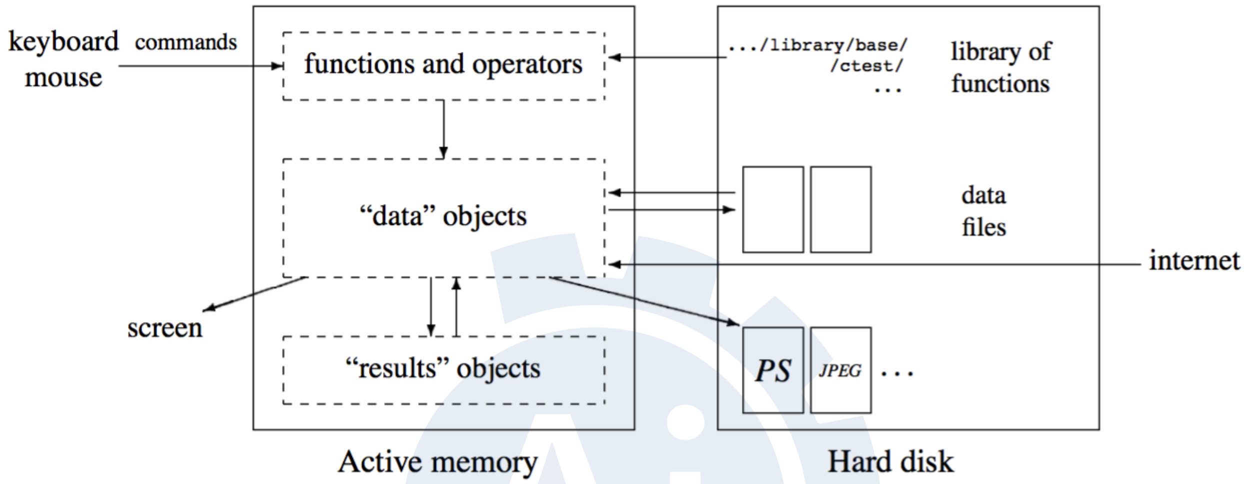
```
?reserved
```

Data-Driven Programming in R and Python

- What is the class (or type) and structure of input object? 投入物件(類別為何？結構是什麼？)
- Attention to the arguments setting variations 轉換過程(有何引數arguments?)
    - Different settings mean different types of transformation. 不同的引數設定值表示以不同的方式完成轉換
    - Always look up the documentation to understand different settings. 需參閱線上使用說明文件以了解各種變化
- What is the class (or type) and structure of output object? 產出物件(類別為何？結構是什麼？)
    - Usually encapsulate all results (or output) by a list in R or with a suffix _ in Python. 通常

以串列(list)結構封裝函數計算的所有結果(R語言S3物件導向)，Python的計算結果則多帶有下底線 _
    - The class (or type) name of the output is usually same as the function name which creates that object. 類別名稱通常與函數名稱相同

Reminders Before Hands-On Practice 實作前的提醒



- How R and Python works? There are some key concepts.
    - Object-oriented, especially the data objects 物件導向(尤其注意資料物件)
    - Package encloses the data, function, and documentation. 套件=資料+函數+說明文件(套件要先下載到硬碟並載入到此次對話中)
    - Both R and Python have functional programming as mentioned above 函數式語法 (input(s) -> arguments -> output(s))
- As a data science tool, data objects are, of course, the main focus. 運作對象是資料物件
    - So, always attention to the class of data object and its dimension. 隨時注意資料物件為何類別？維度與元素個數是多少？
- Most R scripts involve several functions, please comprehend the script parts by parts and read it inside out. 注意分解動作，也就是合成函數的觀念，由內而外逐步理解
- Do not neglect the error message, read it and understand that will improve you a lot. 錯誤訊息是學習過程中的至寶，如果習慣性忽略錯誤訊息，學習效果會非常差！
- It's better to type the script by yourself. 最好自己敲，而非copy -> paste.
- R is case-sensitive and all parenthses should be in pairs. 大小寫有差，半/全形不同，注意空格與括弧的對應
- Continuously improve your English and self-learning ability. 英語與自學能力不斷地提升

The Steps of Using Machine Learning to Analyze Data

- Collecting Data
- Exploring and preparing the data
- Training Model
- Evaluating model performance
- Improving model performance

Algorithms Selection according to the Characteristics of Data

- Supervised learning: Supervised learning starts with a set of observations containing values for both the predictor variables and the outcome. The dataset is then divided into a training sample and a validation sample. A predictive model is developed using the data in the training sample and tested for accuracy using the data in the validation sample.

Model	Task
Nearest Neighbor Method	Classification
Naive Bayes	Classification
Decision Tree	Classification
Classification rules learning algorithm	Classification
Linear regression	Numerical Prediction
Regression Tree	Numerical Prediction
Model Tree	Numerical Prediction
Neural Network	Both
支援向量機	Both

- Unsupervised learning: As opposed to predictive models (i.e. supervised learning) that predict a target of interest, in unsupervised learning, no single feature is more important than any other. In fact, because there is no target to learn, the process of training a descriptive model is called unsupervised learning.

Model	Task
Association Rules	Patterns Detection
k-means	Clustering

Prologue - An Interesting Observation Related to Clustering

- Have you ever spent time watching a large crowd? If so, you are likely to have seen some recurring personalities.
    - Perhaps a certain type of person, identified by a freshly pressed suit and a briefcase, comes to typify the “fat cat” business executive.
    - A twenty-something wearing skinny jeans, a flannel shirt, and sunglasses might be dubbed a “hipster,” while a woman unloading children from a minivan may be labeled a “soccer mom.”
- Of course, applying stereotypes to individuals are dangerous, as no two people are exactly alike. Yet understood as ***a way to describe a collective*** (that’s what clustering want to do), the

labels capture some underlying aspect of ***similarity*** among the individuals within the group.
- Clustering is an unsupervised machine learning task that automatically divides the data into clusters, or groups of similar items. It does this without having been told how the groups should look ahead of time.
- As we may not even know what we're looking for, clustering is used for ***knowledge discovery*** rather than prediction. It provides an insight into the natural groupings found within data.

Clustering - 1

- Clustering is guided by the principle that items or objects inside a cluster should be very similar to each other, but very different from those outside.
- The definition of ***similarity*** might vary across applications, but the basic idea is always the same— group the data so that the related elements are placed together.
- Clustering methods employed in the following applications:
    - Segmenting customers into groups with similar demographics or buying patterns for targeted marketing campaigns
    - Detecting anomalous behavior, such as unauthorized network intrusions, by identifying patterns of use falling outside the known clusters
    - Simplifying extremely large datasets by grouping features with similar values into a smaller number of homogeneous categories

Clustering - 2

- Type of Clusterings
    - Partitional clustering
    - Hierarchical clustering
- Types of Clusters
    - Well-separated
    - Center-based
    - Contiguous
    - Density-based
    - Property or conceptual
    - Described by an Objective Function
- Clustering Algorithms
    - K-means and its variants
    - Hierarchical clustering
    - Density-based clustering
    - Graph-based clustering

The Optimization Problem

$$arg\min_{S} \sum_{j=1}^{k} \sum_{\mathbf{x}_i \subseteq S_j} \|\mathbf{x}_i - \overline{\mathbf{x}}_j\|^2,$$

K-means Clustering

Pseudocode of K-means

- Select K points as the initial centroids
- ***repeat*** Form K clusters by assigning all points to the closest centroid (corresponds to Expectation) Recompute the centroid of each cluster (corresponds to Maximization)
- ***until*** The centroids don't change

Limitations of K-means

- Sizes
- Densities
- Non-globular shapes
- Outliers
- K-means algorithm uses a heuristic process that finds locally optimal solutions. Put simply, this means that it starts with an initial guess for the cluster assignments, and then modifies the assignments slightly to see whether the changes improve the homogeneity within the clusters.

Animation Demostration for K-Means Algorithm

```
# not run here, please execute the following by yourself
library(animation)
kmeans.ani()
```

Hands-Ons Case: Teenage Market Segmentation

- Interacting with friends on a social networking service (SNS), such as Facebook has become a rite of passage (成年禮) for teenagers around the world.
- These teenagers are a coveted demographic for businesses hoping to sell snacks, beverages, electronics, and hygiene products.
- One way to gain this edge is to identify segments of teenagers who share similar tastes, so that marketer can avoid targeting advertisements to teens with no interest in the product being sold.
- Given the text of teenagers' SNS pages, we can identify groups that share common interests such as sports, religion, or music.
- Clustering can automate the process of discovering the natural segments in this population.
- However, it will be up to us to decide whether or not the clusters are interesting and how we can use them for advertising.

- Download a dataset representing a random sample of 30,000 U.S. high school students who had profiles on a well-known SNS, and each teen's gender, age, and number of SNS friends was recorded.
- A text mining tool was used to divide the remaining SNS page content into words (i.e. word segmentation). From the top 500 words appearing across all the pages, 36 words were chosen to represent five categories of interests: namely extracurricular activities, fashion, religion, romance, and antisocial behavior.
    - Such as football, sexy, kissed, bible, shopping, death, and drugs…

Collecting Data

- Set up the configuration for us to program in R and Python interchangeable

```
import numpy as np
import pandas as pd
teens = pd.read_csv('./snsdata.csv', encoding = 'utf-8')
```

- Data understanding

```
print(teens.dtypes)
```

```
## gradyear           int64
## gender             object
## age               float64
## friends             int64
## basketball          int64
## football            int64
## soccer              int64
## softball            int64
## volleyball          int64
## swimming            int64
## cheerleading        int64
## baseball            int64
## tennis              int64
## sports              int64
## cute                int64
## sex                 int64
## sexy                int64
## hot                 int64
## kissed              int64
## dance               int64
## band                int64
## marching            int64
## music               int64
## rock                int64
## god                 int64
## church              int64
## jesus               int64
## bible               int64
## hair                int64
## dress               int64
## blonde              int64
## mall                int64
## shopping            int64
## clothes             int64
## hollister           int64
## abercrombie         int64
## die                 int64
## death               int64
## drunk               int64
## drugs               int64
## dtype: object
```

Exploring and Preparing the Data - 1

- For data.frame, generic function summary() tells us NAs appeared only in ‘gender’ and ‘age’

```
print(teens.describe(include='all'))
```

```
##              gradyear gender  ...        drunk        drugs
## count   30000.000000  27276  ...  30000.000000  30000.000000
## unique           NaN      2  ...           NaN           NaN
## top              NaN      F  ...           NaN           NaN
## freq             NaN  22054  ...           NaN           NaN
## mean     2007.500000    NaN  ...      0.087967      0.060433
## std         1.118053    NaN  ...      0.399125      0.345522
## min      2006.000000    NaN  ...      0.000000      0.000000
## 25%      2006.750000    NaN  ...      0.000000      0.000000
## 50%      2007.500000    NaN  ...      0.000000      0.000000
## 75%      2008.250000    NaN  ...      0.000000      0.000000
## max      2009.000000    NaN  ...      8.000000     16.000000
##
## [11 rows x 40 columns]
```

Exploring and Preparing the Data - 2

- Check the nullity for whole dataset.

```
teens.isnull().sum()
```

```
## gradyear          0
## gender         2724
## age            5086
## friends           0
## basketball        0
## football          0
## soccer            0
## softball          0
## volleyball        0
## swimming          0
## cheerleading      0
## baseball          0
## tennis            0
## sports            0
## cute              0
## sex               0
## sexy              0
## hot               0
## kissed            0
## dance             0
## band              0
## marching          0
## music             0
## rock              0
## god               0
## church            0
## jesus             0
## bible             0
## hair              0
## dress             0
## blonde            0
## mall              0
## shopping          0
## clothes           0
## hollister         0
## abercrombie       0
## die               0
## death             0
## drunk             0
## drugs             0
## dtype: int64
```

Training Model - 1

- Feature selection : cluster analysis by 36 keywords.

```
interests=teens.loc[:, 'basketball':'drugs']
```

Training Model - 2

- The scales of term frequency differ, so normalization (centering & scaling) can help to avoid some feature dominate other features.

```
from sklearn import preprocessing

teens_z=preprocessing.scale(interests)
teens_z = pd.DataFrame(teens_z)
teens_z.head(6)
```

```
##          0         1         2   ...        33        34        35
## 0 -0.332217 -0.357697 -0.242874  ... -0.261530 -0.220403 -0.174908
## 1 -0.332217  1.060049 -0.242874  ... -0.261530 -0.220403 -0.174908
## 2 -0.332217  1.060049 -0.242874  ...  2.027908 -0.220403 -0.174908
## 3 -0.332217 -0.357697 -0.242874  ... -0.261530 -0.220403 -0.174908
## 4 -0.332217 -0.357697 -0.242874  ... -0.261530  2.285122  2.719316
## 5 -0.332217 -0.357697 -0.242874  ... -0.261530  2.285122 -0.174908
##
## [6 rows x 36 columns]
```

Training Model - 3

- k-means clustering，why centers = 5 ?
    - Five stereotypes in the Breakfast Club by John Hughes (1985) - a Brain, an Athelete, a Basket Case, a Princess, and a Criminal, so let’s start with k=5

```
from sklearn.cluster import KMeans

mdl = KMeans(n_clusters = 5)
```

- tot.withinss is the sum of five withinss, totss is the sum of tot.withinss and betweenss(how to check it?)

```
mdl.fit(teens_z)

# Get the attributes and methods before fitting
```

```
## KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
##        n_clusters=5, n_init=10, n_jobs=None, precompute_distances='auto',
##        random_state=None, tol=0.0001, verbose=0)
```

```
pre = dir(mdl)

# Show a few of them
print(pre[51:56])

# Input standardized document-term matrix for model fitting
```

```
## ['score', 'set_params', 'tol', 'transform', 'verbose']
```

```
mdl.fit(teens_z)

# Get the attributes and methods after fitting
```

```
## KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
##        n_clusters=5, n_init=10, n_jobs=None, precompute_distances='auto',
##        random_state=None, tol=0.0001, verbose=0)
```

```
post = dir(mdl)

# # Show again
print(post[51:56])

# Difference set between 'post' and 'pre'
```

```
## ['score', 'set_params', 'tol', 'transform', 'verbose']
```

```
print(list(set(post) - set(pre)))
```

```
## []
```

Model Performance Evaluation

- Unsupervised learning results can be somewhat subjective, so it’s difficult to evaluate the results.
- Quantify vs Qualitative evaluating (cluster validity, Sum of Squares Within /Sum of Squares Between)
- If the groups are too large or too small, they are not likely to be very useful.
- Saving sklearn model and read in again

```
# not run here
import pickle
filename = './_data/kmeans.sav'
# pickle.dump(mdl, open(filename, 'wb'))
res = pickle.load(open(filename, 'rb'))
```

```
pd.Series(mdl.labels_).value_counts()
```

```
## 1    22440
## 3     5827
## 2     1124
## 0      608
## 4        1
## dtype: int64
```

- Cluster analysis results need human interpretation

```
mdl.labels_[:10]
```

```
## array([1, 3, 1, 1, 2, 1, 1, 1, 1, 3], dtype=int32)
```

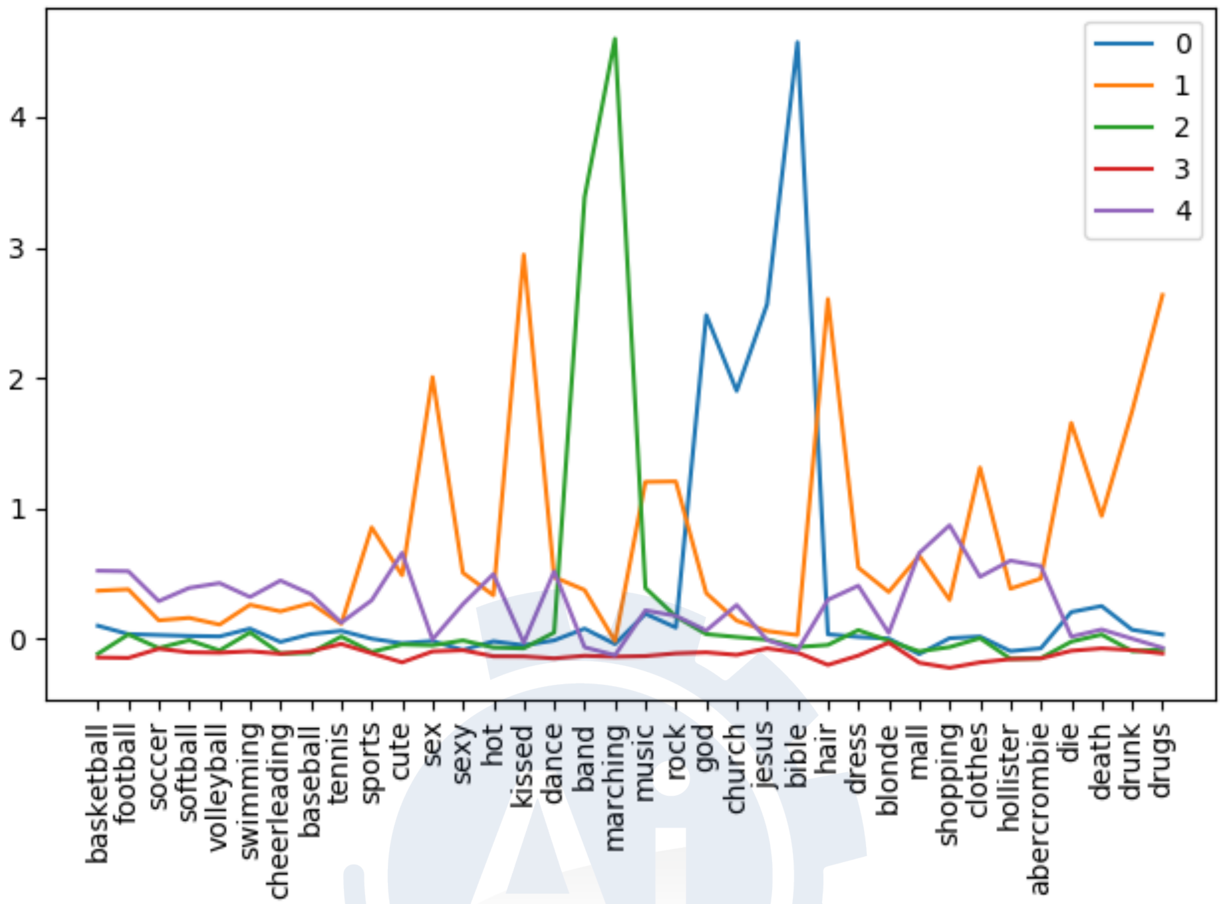
```
# Check the shape of cluster centers matrix
print(mdl.cluster_centers_.shape)

# Create a pandas DataFrame with keyworda for better presentation
```

```
## (5, 36)
```

```
cen = pd.DataFrame(mdl.cluster_centers_, index = range(5),
columns=teens.iloc[:,4:40].columns)
print(cen)
```

```
##    basketball  football    soccer  ...      death      drunk     drugs
## 0   -0.086946  0.062030 -0.101212  ...   0.054774  -0.088533 -0.065422
## 1   -0.147462 -0.149321 -0.080005  ...  -0.077514  -0.084582 -0.111042
## 2    0.362595  0.378270  0.135084  ...   0.906592   1.748383  2.580394
## 3    0.506195  0.494311  0.292165  ...   0.115203  -0.005238 -0.063782
## 4   -0.332217  2.477795 -0.242874  ...  13.475099  14.812744 -0.174908
##
## [5 rows x 36 columns]
```



Line Plot of Term Frequency for Five Clusters

Interpretation of the Clustering

- 36 words' line plot of each cluster

```
# Transpose the cluster centers matrix for plotting
ax = cen.T.plot()
# x-axis ticks position setting
ax.set_xticks(list(range(36)))
# x-axis labels setting (low-level plotting)
```

```
## [<matplotlib.axis.XTick object at 0x7ffe3888a890>, <matplotlib.axis.XTick o
bject at 0x7ffe3888eb50>, <matplotlib.axis.XTick object at 0x7ffe38886cd0>, <m
atplotlib.axis.XTick object at 0x7ffe30b31610>, <matplotlib.axis.XTick object
at 0x7ffe30b424d0>, <matplotlib.axis.XTick object at 0x7ffe30b42d10>, <matplot
lib.axis.XTick object at 0x7ffe30b503d0>, <matplotlib.axis.XTick object at 0x7
ffe30b50a10>, <matplotlib.axis.XTick object at 0x7ffe3888e810>, <matplotlib.ax
is.XTick object at 0x7ffe30b6ac10>, <matplotlib.axis.XTick object at 0x7ffe600
d0390>, <matplotlib.axis.XTick object at 0x7ffe600d08d0>, <matplotlib.axis.XTi
ck object at 0x7ffe600d0f10>, <matplotlib.axis.XTick object at 0x7ffe600d0210
>, <matplotlib.axis.XTick object at 0x7ffe30b6a5d0>, <matplotlib.axis.XTick ob
ject at 0x7ffe30b50310>, <matplotlib.axis.XTick object at 0x7ffe600d6950>, <ma
tplotlib.axis.XTick object at 0x7ffe600d6710>, <matplotlib.axis.XTick object a
t 0x7ffe600de610>, <matplotlib.axis.XTick object at 0x7ffe600dec50>, <matplotl
ib.axis.XTick object at 0x7ffe600e42d0>, <matplotlib.axis.XTick object at 0x7f
fe600e4910>, <matplotlib.axis.XTick object at 0x7ffe600e4790>, <matplotlib.axi
s.XTick object at 0x7ffe600e43d0>, <matplotlib.axis.XTick object at 0x7ffe600d
6690>, <matplotlib.axis.XTick object at 0x7ffe600eb090>, <matplotlib.axis.XTic
k object at 0x7ffe600ebb10>, <matplotlib.axis.XTick object at 0x7ffe600f31d0>,
<matplotlib.axis.XTick object at 0x7ffe600f37d0>, <matplotlib.axis.XTick objec
t at 0x7ffe600f3e10>, <matplotlib.axis.XTick object at 0x7ffe600f9490>, <matpl
otlib.axis.XTick object at 0x7ffe600f9ad0>, <matplotlib.axis.XTick object at 0
x7ffe600f93d0>, <matplotlib.axis.XTick object at 0x7ffe600f3090>, <matplotlib.
axis.XTick object at 0x7ffe600e4750>, <matplotlib.axis.XTick object at 0x7ffe6
0101710>]
```

```
ax.set_xticklabels(list(cen.T.index), rotation=90)
```

```
## [Text(0, 0, 'basketball'), Text(0, 0, 'football'), Text(0, 0, 'soccer'), Te
xt(0, 0, 'softball'), Text(0, 0, 'volleyball'), Text(0, 0, 'swimming'), Text
(0, 0, 'cheerleading'), Text(0, 0, 'baseball'), Text(0, 0, 'tennis'), Text(0,
0, 'sports'), Text(0, 0, 'cute'), Text(0, 0, 'sex'), Text(0, 0, 'sexy'), Text
(0, 0, 'hot'), Text(0, 0, 'kissed'), Text(0, 0, 'dance'), Text(0, 0, 'band'),
Text(0, 0, 'marching'), Text(0, 0, 'music'), Text(0, 0, 'rock'), Text(0, 0, 'g
od'), Text(0, 0, 'church'), Text(0, 0, 'jesus'), Text(0, 0, 'bible'), Text(0,
0, 'hair'), Text(0, 0, 'dress'), Text(0, 0, 'blonde'), Text(0, 0, 'mall'), Tex
t(0, 0, 'shopping'), Text(0, 0, 'clothes'), Text(0, 0, 'hollister'), Text(0,
0, 'abercrombie'), Text(0, 0, 'die'), Text(0, 0, 'death'), Text(0, 0, 'drunk
'), Text(0, 0, 'drugs')]
```

```
fig = ax.get_figure()
fig.tight_layout()
# fig.savefig('./_img/sns_lineplot.png')
```

Model Performance Improvement - 1

- Append clustering results to original data frame 'teens'.

```
teens = pd.concat([teens,pd.Series(mdl.labels_).rename('cluster')], axis=1)
```

- Some columns of new table

```
teens[['gender','age','friends','cluster']][0:5]
```

```
##   gender     age  friends  cluster
## 0      M  18.982        7        1
## 1      F  18.801        0        3
## 2      M  18.335       69        1
## 3      F  18.875        0        1
## 4    NaN  18.995       10        2
```

- Look at the mean age of each cluster, the difference of them is little, because valid age for teenagers are between 13 to 20.

```
teens.groupby('cluster').aggregate({'age': np.mean})
```

```
##                age
## cluster
## 0        18.137230
## 1        18.129906
## 2        17.533893
## 3        17.567499
## 4        18.119000
```

Model Performance Improvement - 2

- Find the female ratio of each cluster.
- Notice the Princesses cluster.

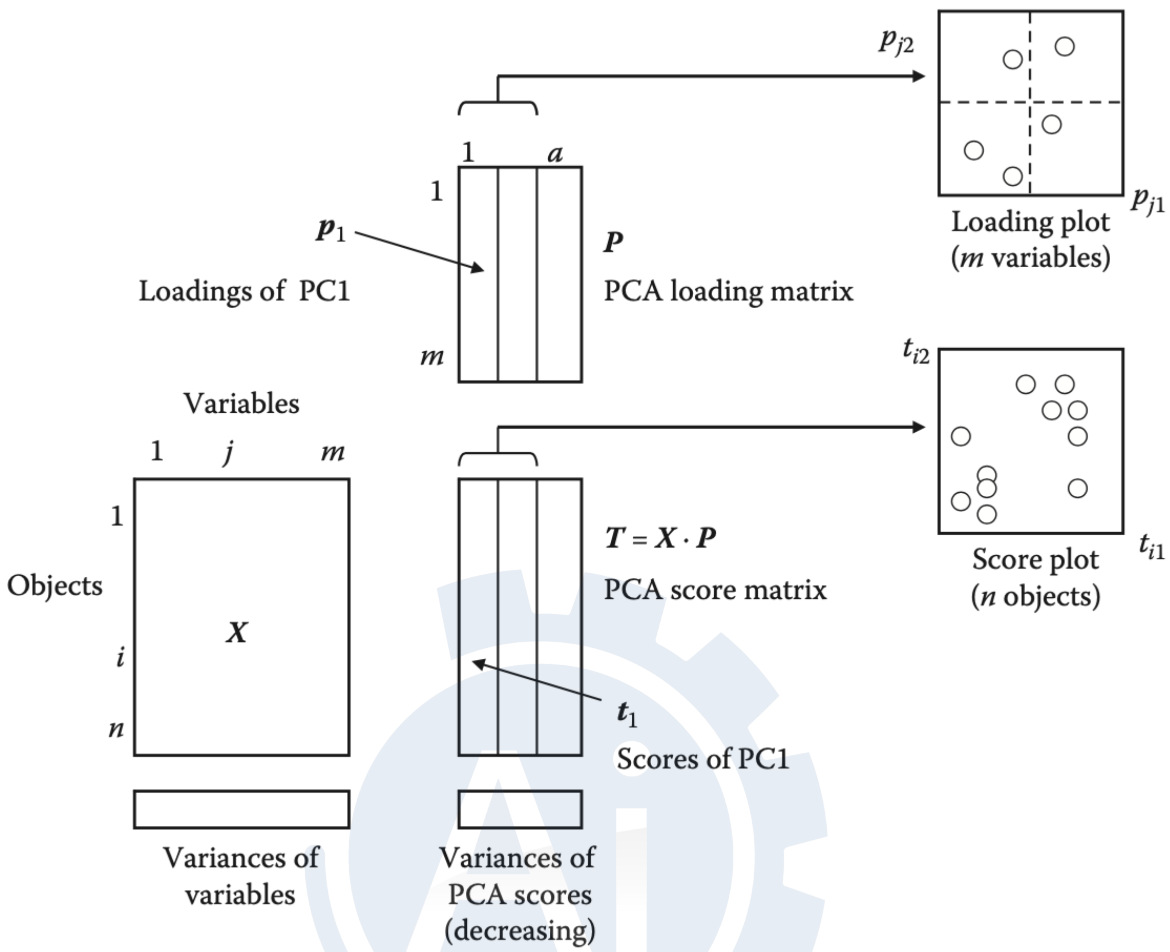
```
teens.groupby('cluster').aggregate({'female': np.mean})
```

- Calculate average number of friends for each cluster.
- Notice the Criminals and Basket Cases.

```
teens.groupby('cluster').aggregate({'friends': np.mean})
```

```
##            friends
## cluster
## 0        32.804276
## 1        27.843048
## 2        30.250000
## 3        38.887249
## 4        44.000000
```

Dimensionality Reduction and Principal Component Analysis



Matrix scheme for PCA

- Cell Segmentation Case

```
import pandas as pd
import numpy as np
cell = pd.read_csv('segmentationOriginal.csv')
```

- Data Understanding and Missing Values Identifying

```
cell.head(2)
```

```
##          Cell   Case Class  ...  WidthStatusCh1  XCentroid  YCentroid
## 0  207827637   Test    PS  ...               2         42         14
## 1  207932307  Train    PS  ...               1        215        347
##
## [2 rows x 119 columns]
```

```
cell.info() # RangeIndex, Columns, dtypes, memory type
```

```
## <class 'pandas.core.frame.DataFrame'>
## RangeIndex: 2019 entries, 0 to 2018
## Columns: 119 entries, Cell to YCentroid
## dtypes: float64(49), int64(68), object(2)
## memory usage: 1.8+ MB
```

```
cell.shape
```

```
## (2019, 119)
```

```
cell.columns.values # 119 variable names
```

```
cell.dtypes
```

```
## Cell                              int64
## Case                             object
## Class                            object
## AngleCh1                        float64
## AngleStatusCh1                    int64
## AreaCh1                           int64
## AreaStatusCh1                     int64
## AvgIntenCh1                     float64
## AvgIntenCh2                     float64
## AvgIntenCh3                     float64
## AvgIntenCh4                     float64
## AvgIntenStatusCh1                 int64
## AvgIntenStatusCh2                 int64
## AvgIntenStatusCh3                 int64
## AvgIntenStatusCh4                 int64
## ConvexHullAreaRatioCh1          float64
## ConvexHullAreaRatioStatusCh1      int64
## ConvexHullPerimRatioCh1         float64
## ConvexHullPerimRatioStatusCh1     int64
## DiffIntenDensityCh1             float64
## DiffIntenDensityCh3             float64
## DiffIntenDensityCh4             float64
## DiffIntenDensityStatusCh1         int64
## DiffIntenDensityStatusCh3         int64
## DiffIntenDensityStatusCh4         int64
## EntropyIntenCh1                 float64
## EntropyIntenCh3                 float64
## EntropyIntenCh4                 float64
## EntropyIntenStatusCh1             int64
## EntropyIntenStatusCh3             int64
##                                  ...
## ShapeP2ACh1                     float64
## ShapeP2AStatusCh1                 int64
## SkewIntenCh1                    float64
## SkewIntenCh3                    float64
## SkewIntenCh4                    float64
## SkewIntenStatusCh1                int64
## SkewIntenStatusCh3                int64
## SkewIntenStatusCh4                int64
## SpotFiberCountCh3                 int64
## SpotFiberCountCh4                 int64
## SpotFiberCountStatusCh3           int64
## SpotFiberCountStatusCh4           int64
## TotalIntenCh1                     int64
## TotalIntenCh2                     int64
## TotalIntenCh3                     int64
## TotalIntenCh4                     int64
## TotalIntenStatusCh1               int64
## TotalIntenStatusCh2               int64
## TotalIntenStatusCh3               int64
## TotalIntenStatusCh4               int64
## VarIntenCh1                     float64
```

```
## VarIntenCh3                     float64
## VarIntenCh4                     float64
## VarIntenStatusCh1                 int64
## VarIntenStatusCh3                 int64
## VarIntenStatusCh4                 int64
## WidthCh1                        float64
## WidthStatusCh1                    int64
## XCentroid                         int64
## YCentroid                         int64
## Length: 119, dtype: object
```

```
cell.describe(include = "all")
```

```
##                 Cell  Case Class  ...  WidthStatusCh1    XCentroid    YCent
roid
## count   2.019000e+03  2019  2019  ...     2019.000000  2019.000000  2019.00
0000
## unique           NaN     2     2  ...             NaN          NaN
NaN
## top              NaN  Test    PS  ...             NaN          NaN
NaN
## freq             NaN  1010  1300  ...             NaN          NaN
NaN
## mean    2.084024e+08   NaN   NaN  ...        0.271421   260.727093   177.34
3239
## std     2.790457e+05   NaN   NaN  ...        0.607706   140.365593   107.72
0132
## min     2.078276e+08   NaN   NaN  ...        0.000000     9.000000     8.00
0000
## 25%     2.083325e+08   NaN   NaN  ...        0.000000   142.000000    88.00
0000
## 50%     2.083843e+08   NaN   NaN  ...        0.000000   262.000000   165.00
0000
## 75%     2.084052e+08   NaN   NaN  ...        0.000000   382.000000   253.00
0000
## max     2.109641e+08   NaN   NaN  ...        2.000000   501.000000   501.00
0000
##
## [11 rows x 119 columns]
```

```
cell.isnull().any() # check NA by column
```

```
cell.isnull().values.any() # False, means no missing value ! Check the differe
nce between above two !!!!

#cell.isnull()
#type(cell.isnull()) # pandas.core.frame.DataFrame, so .index, .column, and .v
alues three important attributes

#cell.isnull().values
#type(cell.isnull().values) # numpy.ndarray
```

```
## False
```

```
cell.isnull().sum()
```

- Select the training set

```
#cell['Case'].nunique()
cell['Case'].unique()
```

```
## array(['Test', 'Train'], dtype=object)
```

```
cell.Case.value_counts()
#select the training set
```

```
## Test     1010
## Train    1009
## Name: Case, dtype: int64
```

```
cell_train = cell.loc[cell['Case']=='Train'] # same as cell[cell['Case']=='Tra
in']
cell_train.head()
```

```
##          Cell   Case Class  ...  WidthStatusCh1  XCentroid  YCentroid
## 1   207932307  Train    PS  ...               1        215        347
## 2   207932463  Train    WS  ...               0        371        252
## 3   207932470  Train    PS  ...               0        487        295
## 11  207932484  Train    WS  ...               0        211        495
## 14  207932459  Train    PS  ...               0        172        207
##
## [5 rows x 119 columns]
```

```
cell['Case'][:10]
```

```
## 0     Test
## 1    Train
## 2    Train
## 3    Train
## 4     Test
## 5     Test
## 6     Test
## 7     Test
## 8     Test
## 9     Test
## Name: Case, dtype: object
```

```
type(cell['Case']) # <class 'pandas.core.series.Series'>
```

```
## <class 'pandas.core.series.Series'>
```

```
cell[['Case']][:10]
```

```
##     Case
## 0   Test
## 1  Train
## 2  Train
## 3  Train
## 4   Test
## 5   Test
## 6   Test
## 7   Test
## 8   Test
## 9   Test
```

```
type(cell[['Case']]) # <class 'pandas.core.frame.DataFrame'>
```

```
## <class 'pandas.core.frame.DataFrame'>
```

- Create feature matrix (X)

```
cell_data = cell_train.drop(['Cell','Class','Case'], axis=1)
cell_data.head()

# alternative way to do the same thing
```

```
##       AngleCh1  AngleStatusCh1  AreaCh1  ...  WidthStatusCh1  XCentroid  YC
entroid
## 1   133.752037               0      819  ...               1        215
347
## 2   106.646387               0      431  ...               0        371
252
## 3    69.150325               0      298  ...               0        487
295
## 11  109.416426               0      256  ...               0        211
495
## 14  104.278654               0      258  ...               0        172
207
##
## [5 rows x 116 columns]
```

```
cell_data = cell_train.drop(cell_train.columns[0:3], 1)
cell_data.head()
```

```
##       AngleCh1  AngleStatusCh1  AreaCh1  ...  WidthStatusCh1  XCentroid  YC
entroid
## 1   133.752037               0      819  ...               1        215
347
## 2   106.646387               0      431  ...               0        371
252
## 3    69.150325               0      298  ...               0        487
295
## 11  109.416426               0      256  ...               0        211
495
## 14  104.278654               0      258  ...               0        172
207
##
## [5 rows x 116 columns]
```

- Create class label vector (y) (label encoding and one-hot encoding)

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder # Encode labels with value betw
een 0 and n_classes-1.

# label encoding
le_class = LabelEncoder().fit(cell['Class']) # 'PS': 0, 'WS': 1
Class_label = le_class.transform(cell['Class']) # 0: PS, 1: WS
Class_label.shape # (2019,)

# one-hot encoding
```

```
## (2019,)
```

```
ohe_class = OneHotEncoder(sparse=False).fit(Class_label.reshape(-1,1)) # spars
e : boolean, default=True Will return sparse matrix if set True else will retu
rn an array.
#help(OneHotEncoder)
```

```
## /opt/anaconda3/lib/python3.7/site-packages/sklearn/preprocessing/_encoders.
py:414: FutureWarning: The handling of integer data will change in version 0.2
2. Currently, the categories are determined based on the range [0, max(value
s)], while in the future they will be determined based on the unique values.
## If you want the future behaviour and silence this warning, you can specify
"categories='auto'".
## In case you used a LabelEncoder before this OneHotEncoder to convert the ca
tegories to integers, then you can now use the OneHotEncoder directly.
##   warnings.warn(msg, FutureWarning)
```

```
ohe_class.get_params()
#{'categorical_features': 'all',
# 'dtype': float,
# 'handle_unknown': 'error',
# 'n_values': 'auto',
# 'sparse': False}
#ohe_class.categorical_features
```

```
## {'categorical_features': None, 'categories': None, 'drop': None, 'dtype':
<class 'numpy.float64'>, 'handle_unknown': 'error', 'n_values': None, 'sparse
': False}
```

```
Class_ohe=ohe_class.transform(Class_label.reshape(-1,1)) # (2019, 2)

Class_label.reshape(-1,1).shape # (2019, 1) different to 1darray (2019,)
```

```
## (2019, 1)
```

```
Class_ohe.shape # (2019, 2) 2darray
```

```
## (2019, 2)
```

```
Class_ohe
```

```
## array([[1., 0.],
##        [1., 0.],
##        [0., 1.],
##        ...,
##        [1., 0.],
##        [0., 1.],
##        [0., 1.]])
```

```
# Fast way to do one-hot encoding or dummy encoding
Class_dum = pd.get_dummies(cell['Class'])
print (Class_dum.head())
```

```
##    PS  WS
## 0   1   0
## 1   1   0
## 2   0   1
## 3   1   0
## 4   1   0
```

- Differentiate categorical features from numeric features

```
print(cell_data.columns)
```

```
## Index(['AngleCh1', 'AngleStatusCh1', 'AreaCh1', 'AreaStatusCh1', 'AvgIntenC
h1',
##        'AvgIntenCh2', 'AvgIntenCh3', 'AvgIntenCh4', 'AvgIntenStatusCh1',
##        'AvgIntenStatusCh2',
##        ...
##        'VarIntenCh1', 'VarIntenCh3', 'VarIntenCh4', 'VarIntenStatusCh1',
##        'VarIntenStatusCh3', 'VarIntenStatusCh4', 'WidthCh1', 'WidthStatusCh
1',
##        'XCentroid', 'YCentroid'],
##       dtype='object', length=116)
```

```
type(cell_data.columns) # pandas.core.indexes.base.Index
```

```
## <class 'pandas.core.indexes.base.Index'>
```

```
dir(pd.Series.str)
```

```
## ['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '_
_format__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__',
'__init__', '__init_subclass__', '__iter__', '__le__', '__lt__', '__module__',
'__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__',
'__sizeof__', '__str__', '__subclasshook__', '__weakref__', '_freeze', '_get_s
eries_list', '_make_accessor', '_validate', '_wrap_result', 'capitalize', 'cat
', 'center', 'contains', 'count', 'decode', 'encode', 'endswith', 'extract', '
extractall', 'find', 'findall', 'get', 'get_dummies', 'index', 'isalnum', 'isa
lpha', 'isdecimal', 'isdigit', 'islower', 'isnumeric', 'isspace', 'istitle', '
isupper', 'join', 'len', 'ljust', 'lower', 'lstrip', 'match', 'normalize', 'pa
d', 'partition', 'repeat', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition
', 'rsplit', 'rstrip', 'slice', 'slice_replace', 'split', 'startswith', 'strip
', 'swapcase', 'title', 'translate', 'upper', 'wrap', 'zfill']
```

```
pd.Series(cell_data.columns).str.contains("Status").head() # logical indices a
fter making cell_data.columns as pandas.Series
#type(pd.Series(cell_data.columns).str.contains("Status")) # pandas.core.serie
s.Series
```

```
## 0    False
## 1     True
## 2    False
## 3     True
## 4    False
## dtype: bool
```

```
cell_data.columns[pd.Series(cell_data.columns).str.contains("Status")] # again
pandas.core.indexes.base.Index
#type(cell_data.columns[pd.Series(cell_data.columns).str.contains("Status")])
# pandas.core.indexes.base.Index
```

```
## Index(['AngleStatusCh1', 'AreaStatusCh1', 'AvgIntenStatusCh1',
##        'AvgIntenStatusCh2', 'AvgIntenStatusCh3', 'AvgIntenStatusCh4',
##        'ConvexHullAreaRatioStatusCh1', 'ConvexHullPerimRatioStatusCh1',
##        'DiffIntenDensityStatusCh1', 'DiffIntenDensityStatusCh3',
##        'DiffIntenDensityStatusCh4', 'EntropyIntenStatusCh1',
##        'EntropyIntenStatusCh3', 'EntropyIntenStatusCh4', 'EqCircDiamStatusC
h1',
##        'EqEllipseLWRStatusCh1', 'EqEllipseOblateVolStatusCh1',
##        'EqEllipseProlateVolStatusCh1', 'EqSphereAreaStatusCh1',
##        'EqSphereVolStatusCh1', 'FiberAlign2StatusCh3', 'FiberAlign2StatusCh
4',
##        'FiberLengthStatusCh1', 'FiberWidthStatusCh1', 'IntenCoocASMStatusCh
3',
##        'IntenCoocASMStatusCh4', 'IntenCoocContrastStatusCh3',
##        'IntenCoocContrastStatusCh4', 'IntenCoocEntropyStatusCh3',
##        'IntenCoocEntropyStatusCh4', 'IntenCoocMaxStatusCh3',
##        'IntenCoocMaxStatusCh4', 'KurtIntenStatusCh1', 'KurtIntenStatusCh3',
##        'KurtIntenStatusCh4', 'LengthStatusCh1', 'MemberAvgAvgIntenStatusCh2
',
##        'MemberAvgTotalIntenStatusCh2', 'NeighborAvgDistStatusCh1',
##        'NeighborMinDistStatusCh1', 'NeighborVarDistStatusCh1',
##        'PerimStatusCh1', 'ShapeBFRStatusCh1', 'ShapeLWRStatusCh1',
##        'ShapeP2AStatusCh1', 'SkewIntenStatusCh1', 'SkewIntenStatusCh3',
##        'SkewIntenStatusCh4', 'SpotFiberCountStatusCh3',
##        'SpotFiberCountStatusCh4', 'TotalIntenStatusCh1', 'TotalIntenStatusC
h2',
##        'TotalIntenStatusCh3', 'TotalIntenStatusCh4', 'VarIntenStatusCh1',
##        'VarIntenStatusCh3', 'VarIntenStatusCh4', 'WidthStatusCh1'],
##       dtype='object')
```

```
len(cell_data.columns[pd.Series(cell_data.columns).str.contains("Status")]) #
58 features with "Status"
```

```
## 58
```

```
cell_num = cell_data.drop(cell_data.columns[pd.Series(cell_data.columns).str.c
ontains("Status")],axis=1)
cell_num.head()
```

```
##       AngleCh1  AreaCh1  AvgIntenCh1  ...   WidthCh1  XCentroid  YCentroid
## 1   133.752037      819    31.923274  ...  32.161261        215        347
## 2   106.646387      431    28.038835  ...  21.185525        371        252
## 3    69.150325      298    19.456140  ...  13.392830        487        295
## 11  109.416426      256    18.828571  ...  17.546861        211        495
## 14  104.278654      258    17.570850  ...  17.660339        172        207
##
## [5 rows x 58 columns]
```

- Dimensionality Reduction (dr) by PCA

```
from sklearn.decomposition import PCA
dr = PCA() # Principal Components Analysis


cell_pca = dr.fit_transform(cell_num) # PCA only for numeric
cell_pca
```

```
## array([[ 8.47983436e+04, -1.09206431e+05,  3.08230196e+04, ...,
##         -1.85584989e-02, -1.86032858e-02,  1.47400180e-11],
##        [-2.14595263e+04, -1.77417260e+04, -1.78468558e+03, ...,
##         -2.00628855e-02,  5.54777551e-03,  1.46113095e-11],
##        [-5.35770250e+04,  1.03968298e+04,  9.15969476e+03, ...,
##         -1.08847740e-02,  2.82838575e-02,  6.42386175e-14],
##        ...,
##        [-2.57298792e+04, -2.43260560e+04, -5.99996296e+04, ...,
##         -1.03950386e-02,  1.48297048e-02, -1.54181159e-12],
##        [-2.71587740e+04, -1.94760869e+04, -4.94019505e+04, ...,
##         -1.42084059e-02,  3.19415826e-03, -6.61852643e-12],
##        [ 1.14504120e+03, -4.19702178e+04, -2.30591893e+04, ...,
##         -5.69891526e-03,  2.15411583e-03,  9.98910549e-13]])
```

```
dir(dr)
```

```
## ['__abstractmethods__', '__class__', '__delattr__', '__dict__', '__dir__',
'__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getstate__
', '__gt__', '__hash__', '__init__', '__init_subclass__', '__le__', '__lt__',
'__module__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__',
'__setattr__', '__setstate__', '__sizeof__', '__str__', '__subclasshook__', '_
_weakref__', '_abc_impl', '_fit', '_fit_full', '_fit_svd_solver', '_fit_trunca
ted', '_get_param_names', '_get_tags', 'components_', 'copy', 'explained_varia
nce_', 'explained_variance_ratio_', 'fit', 'fit_transform', 'get_covariance',
'get_params', 'get_precision', 'inverse_transform', 'iterated_power', 'mean_',
'n_components', 'n_components_', 'n_features_', 'n_samples_', 'noise_variance_
', 'random_state', 'score', 'score_samples', 'set_params', 'singular_values_',
'svd_solver', 'tol', 'transform', 'whiten']
```

```
dr.components_[:2] # [:2] can be removed, if you want to see more results of r
otation matrix.
```

```
## array([[-1.20772373e-05,  1.11800234e-03,  1.39935119e-03,
##          9.52570715e-04,  2.48690529e-04,  9.40628085e-04,
##         -4.15401765e-07,  9.14069295e-08,  3.98507719e-04,
##          1.40289157e-04,  5.07769261e-04,  6.87236820e-06,
##          8.13559199e-07,  8.52284533e-06,  2.85013735e-05,
##         -2.64088142e-06,  3.85927180e-03,  2.52352309e-03,
##          4.47249694e-03,  3.06992719e-02, -4.70489024e-07,
##         -3.72382194e-07,  1.89140825e-05,  2.36813742e-05,
##          1.07032223e-07, -3.77790202e-07, -2.60762862e-05,
##         -2.88382099e-06, -2.04797676e-06,  4.59901695e-06,
##          3.40979143e-07, -5.41351578e-07, -3.63271978e-06,
##          2.40700604e-06, -7.66636701e-06,  2.44217672e-05,
##          1.14568916e-05,  1.49472962e-05, -1.16156582e-05,
##          8.51909135e-05,  2.09939072e-07, -1.70034827e-06,
##         -1.96609334e-06, -1.79799998e-06,  9.67078769e-07,
##         -1.88217019e-06,  5.24739478e-06,  8.04863458e-06,
##          7.40593712e-01,  4.69191466e-01,  1.61895667e-01,
##          4.51862380e-01,  7.26861406e-04,  3.71841771e-04,
##          7.59918222e-04,  2.98077199e-05, -1.43658866e-04,
##         -1.37681271e-04],
##        [ 2.12581608e-05, -2.68374511e-04,  1.30045966e-03,
##         -1.12786669e-03, -3.64008269e-04, -1.92919185e-03,
##          3.04717255e-07,  1.00102791e-07,  3.37747427e-04,
##         -2.04776688e-04, -1.08017057e-03,  2.11861099e-06,
##         -9.51419445e-06, -1.36747734e-05, -1.12801572e-05,
##          4.48336024e-06,  1.82882051e-04, -4.00217100e-04,
##         -1.07262609e-03, -2.74478711e-03,  2.09941789e-08,
##          1.05117916e-08, -3.23766369e-06, -1.07306291e-05,
##          6.66252157e-07,  6.27804271e-07,  1.49711626e-05,
##         -1.62046940e-06, -6.35220368e-06, -7.85177205e-06,
##          7.38650679e-07,  9.91619257e-07, -5.63744748e-06,
##          2.25840662e-05,  1.66775434e-05,  9.16663635e-06,
##         -1.88640693e-05, -2.37173687e-05, -1.61950483e-05,
##         -2.79365856e-05, -1.50508544e-07,  3.06601353e-06,
##          1.33954125e-06, -1.33644490e-06,  3.10996567e-06,
##          4.63405918e-06,  9.05566152e-07,  6.86521445e-06,
##          6.57513374e-01, -3.71191529e-01, -1.50167780e-01,
##         -6.38218269e-01,  6.82321741e-04, -2.83726353e-04,
##         -1.55158886e-03, -1.82824730e-05, -2.05493644e-04,
##          2.14671008e-05]])
```

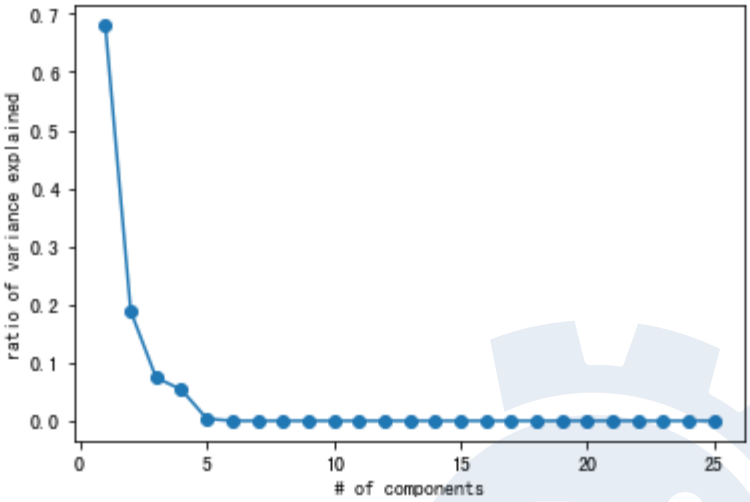
```
type(dr.components_) # numpy.ndarray
```

```
## <class 'numpy.ndarray'>
```

```
dr.components_.shape # (58, 58)
```

```
## (58, 58)
```

```
# scree plot
dr.explained_variance_ratio_
import matplotlib.pyplot as plt
plt.plot(range(1, 26), dr.explained_variance_ratio_[:25], '-o')
plt.xlabel('# of components')
plt.ylabel('ratio of variance explained')
```



Scree Plot of PCA

```
# list(range(1,59))
# range(1,59).tolist() # AttributeError: 'range' object has no attribute 'toli
st'

cell_dr = cell_pca[:,:5]
cell_dr
# pd.DataFrame(cell_dr).to_csv('cell_dr.csv')
```

```
## array([[  84798.34361854, -109206.43140231,   30823.01955784,
##           17203.74850499,   10043.47510989],
##        [ -21459.52630159,  -17741.72601452,   -1784.68558447,
##           -1034.14605652,    2397.90486914],
##        [ -53577.02497608,   10396.82979117,    9159.69476304,
##           -4910.95656292,     959.5703947 ],
##        ...,
##        [ -25729.8792278 ,  -24326.05603993,  -59999.62957106,
##           -6529.78382658,   -3576.79204107],
##        [ -27158.77402526,  -19476.08687044,  -49401.95049235,
##          -12577.75498609,    -961.91880638],
##        [   1145.04120403,  -41970.21778886,  -23059.18929957,
##           -6122.01345777,    1546.57100978]])
```

References:

- Hastie, T., Tibshirani, R., Friedman, J. (2001), The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer.
- Kuhn, M. and Johnson, K. (2013), Applied Predictive Modeling, Springer.

- Lantz, B. (2013), Machine Learning with R, Packt Publishing.
- Ledolter, J. (2013), Data Mining and Business Analytics with R, John Wiley & Sons.
- Provost, F. and Fawcett, T. (2013), Data Science for Business: What You Need to Know About Data Mining and Data-Analytic Thinking, O'Reilly.
- Raschka, S. (2015), Python Machine Learning: Unlock deeper insights into machine learning with this vital guide to cutting- edge predictive analytics, PACKT publishing.
- Tan, P.-N., Steinbach, M., and Kumar, V. (2006), Introduction to Data Mining, Pearson.
- Torgo, L. (2011), Data Mining with R: Learning with Case Studies, CRC Press.
- Varmuza, K. and Filzmoser, P., 2009. Introduction to Multivariate Statistical Analysis in Chemometrics, CRC.
- Williams, G. (2011), Data Mining with Rattle and R: The Art of Excavating Data for Knowledge Discovery, Springer.
- Zhao, Y. (2013), R and Data Mining: Examples and Case Studies, Academic Press.

- Thanks for your attention.
- Email me at cstsou @ mail.mcut.edu.tw or cstsou @ ntub.edu.tw if there is anything I can help.