- ML
  - y present / not
    - SL
      - y datatype
        - Class.
        - Reg.
    - USL
      - Clustering
        - Hier
        - Non Hier.
        - Density
      - Asso.
      - Recc.

SL & 2

LDA
- SL feature / dimension reduction

PCA → USL feature / dimension reduction

* ✓ pattern (x / feature)
* ✓ predict (y)
* ✓ Metric (y)

* ✓ pattern (x / feature)
* x predict (No y)
* x metric (No y)

known ← (label) input — unknown (earlier) stage

Clustering
- what!
  - grouping
    - based on similarity & dissimilarity
      - based on x & 2 principles
        - within variance min.
        - b/w variance max.
          - calculated using distance method
            - eu
            - sq. eu.
            - manh
            - min
            - cheb
  - impacted by
    - distance method → so → outliers (remove) else impact mean & distance & variance
    - USL based on independent var (x)
      - so → feature selection (no forming any cluster pattern)
      - so → avoid multi column (?)
  - select the best config (K).

↳ How? / Algo?

↳ K means clustering

↙ no. of centroid ↘ specify mean of centroid

1. Choose k (num no)
2. Choose distance method (eu.)
3. align datapoints to centroid
4. align centroid to centre
5. align datapoints to centroid
6. do 4, 5 until convergence

Job of centroid is to align to the centre of the datapoints in the group

datapoint shift blw groups

↓

centre changes / means (blw groups)

↓

centroid changes.

↓

stop? / convergence

↳ no datapoint shift blw groups

Clustering framework of many.

Define Business problem (FD, DC, IE, etc)

↓

Feature selection

↓

Choose distance metric

↓

Choose cluster method

↓

Build cluster

↓

Profile/name clusters by datascientist (not algo.)

↓

Effectiveness

Silhouette score

no. of datapoint

→ Silh coeff value

↳ algorithm

↳ complete
↳ average
↳ centroid

Optimal K value

↳ techniques
- → WCSS / Inertia
- → Silh score
- → Silh. viz.

WCSS

→ what? within cluster sum of squares

↓ Variation / distance b/w datapoint & centroid

$WCSS_{k=N} = WCSS_{cluster1} + \dots + WCSS_{clusterN}$

$\sum_{i=1}^{\text{no. of feature}} (x_i - \bar{x}_i)^2$

→ ~~How to use it~~ why it?

↑k ⟶ ↓WCSS

as datapoints in clusters align more closer to the centroid

↳ so reduce variation / distance

↳ How to use it?

↳ elbow plot

↳ what: x-axis: k y-axis: WCSS
elbow shape (ideally)

↳ How:
drastic decrease in WCSS & then flatten out

⇓

Limitation: subjectivity wrt drastic decrease & flatten out

↳ so, elbow plot is mainly used to eliminate the unnecessary k value i.e. to short list.

Silh. score

- based on 2 principles
  - within variance/distance b/w datapt & centroid is min. (a)
  - b/w variance/distance b/w datapt & centroid is max (b).

- what?

$$\text{Silh. score} = \frac{\sum_{i=1}^{\text{no. of datapt}} \text{Silh. coeff}}{\text{No. of datapoints}}$$

- what is Silh. coeff?

$$\text{Silh. coeff. (i-th datapoint)} = \frac{(b-a)}{max(b,a)}$$

b → b/w cluster distance b/w one pt & all pts mean.

a → within cluster distance b/w the point & other pts mean.

- what Silh. coeff. signifies?
  - -1: datapoint wrongly clustered
  - 0: datapoint is in border case
  - 1: datapoint is correctly classified

$$\text{Silh. coeff of datapoint} = \frac{(b-a)}{max(b,a)} = \frac{(\infty-0)}{max(\infty,0)} = \frac{\infty}{\infty} = 1$$

$$\text{Silh. coeff of datapoint} = \frac{(b-a)}{max(b,a)} = \frac{(0-\infty)}{max(0,\infty)} = \frac{-\infty}{\infty} = -1$$

$$\text{Silh. coeff of datapoint} = \frac{(b-a)}{max(b,a)} = \frac{0}{b/a} = 0 \quad \& \; b = a.$$

- What Silh. score signifies?
  - ≃ 1 good clustering as majority of Silh. coeff is ≃ 1.
  - ≃ 0 bad clustered
  - ≃ -1 poorly clustered

So, Silh. score higher/i.e. most nearer to 1 → that k value is prefered as clustered good.

Silh. viz

- 3 criteria (for choosing optimal k value)
  1. no. outliers in any cluster (i.e. no -ve Silh. coeff in any cluster)
  2. Silh. coeff. of most of datapoints is greater than the Silh. score
  3. Highest Silh. score

- How to Silh. viz?
  1) For each cluster
     - Silh. coeff are arranged in descending order
     - & then plotted & joined
  2) Finally a vertical line to mark Silh. score.

  x-axis: Silh. coeff value.
  y-axis: cluster name.

Optimal k value
1. pair plot (only 2 features)
2. WCSS / Inertia / elbow plot
3. Silh. score
4. Silh. viz

Profiling clusters

- Non technical; business based; domain knowledge needed.
- More related to the descriptive Analysis of the clusters.
- F_1 F_2 - - - F_N → C_0 ... → C_K
  - then used in targeting the product ⇒ value added to the business.

Hierarchical clustering

→ what is hierarchical & (base) (based for Hs. clustering)

↑ hierarchical level → ↓ no. of persons

⇓

Ily. in the hierarchical clustering

→ Types of Hierarchical clustering

- Bottom up Approach (Agglomerative Clustering) [preferred as computation].
  - ↑ iteration → ↓ no. of datapoints / cluster
- top down (decisive clustering)
  - ↑ iterations → ↑ no. of cluster / datapoints

↳ How? / Algo.

1. each datapoint as cluster
2. pair wise distance b/w clusters
3. merge the two nearest clusters.
4. step 2:3 - until become one larger cluster.

↳ as clusters (not datapoint) use linkage method → decide to find distance b/w clusters/datapoints for distance calculation:
[Euc euclidean — which for b/w datapoints]

1. Single linkage → min dist
2. Complete linkage → max dist
3. Centroid linkage / base → b/w centroid dist
4. Avg linkage → Avg of dist.

linkage method distance - Ward's linkage

↳ Out of all possible pair combination of the cluster, the cluster formation with min. variance is merged.

↳ called min-variance method.

↳ variance: distance b/w all points in C_1 & C_2 to centroid of C_1 & C_2.

↳ i.e. centroid using all datapoints from C_1 & C_2

↳ why? (preference) / when? to use

- ↳ compact & sphere structure
- ↳ not effect by outlier → so, no chain effect
- ↳ Objective is min. variance.

linkage

↳ gives 3 things

1. Index of cluster(s) / datapoint(s) to be merged
2. Cophenetic distance / i.e. distance @ which they are merged.
3. no. of datapoints in that formed merged cluster

↓ gives

Dendrogram (high dissimilarity (high y), low dissimilarity (low y))

↳ is visual representation of how cluster(s) & datapoint(s) merged.

↳ x-axis: Index of cluster / datapoint

↳ y-axis: cophenetic distance / dissimilarity

↳ optimal no. of clusters is the range in y-axis that for which large range exist → cut @ any distance (y value) of that range (distance/y axis).

(Significant distance)

y is dissimilarity & it is stable for long distance → Natural group by avoid heterogeneity

Cluster formation method

↳ one of disadvantage of agglomerative clustering / hierarchical clustering

↓

: it stop when one big cluster is formed.

↳ 2 types

- ↓ distance based: Cophenetic distance i.e. y axis in dendrogram → cut @ that distance.
- ↓ max-cluster

↓↓ By this two methods we form the desired no. of clusters formation.

How to decide optimal no. of clusters? (in hierarchical clustering)

↓

Cophenetic correlation → -1 to 1.

↳ is pearson correlation b/w 'distance matrix & cophenetic matrix.

↳ Highest value is better cluster formation. (Signifies)

↳ why?

↳ because euclidean distance b/w datapoints (closeness in level of datapoints) given by distance matrix need to be similar to cophenetic distance (i.e. linkage merge distance) b/w datapoints (@ level of clusters)

↓↓

more similar means clusters are well formed else not clustered the datapoints into right cluster.

↳ key components

- ↳ 1. distance matrix (b/w datapoints - euclidean distance)
- ↳ 2. cophenetic matrix (b/w cluster(s) & datapoint(s) - cophenetic distance / linkage distance)

↓↓ ⓟ the no. of clusters.

PCA

when to drop a column

1. multicollinearity
2. NO correlation (x - y)
3. Redundancy
4. p ~~>>~~ n → insignificant
5. Regularization (lasso)
6. Domain knowledge - drop column
7. Std ≈ 0.
8. Image → processing speed as ↑ dimensions.

But info loss due to drop'g column.

PCA → reduce those impacts w/o droping column.

PCA → based on
1 Information
& 2 variation/variance

✓ Variation → ✓ Info
✗ Variation → ✗ Info
(key rule.

Curse of dimensionality

↳ Guideline
n → no. of feature
atleast n^3 → no. of rows / datapoints
to avoid curse of dimensionality

↳ for same no. of datapoints
→ ↑ no. of feature
↓ leads to
↑ sparcity
↓ leads to
overfit of model
(as less no. of datapoints but more no. of feature
(is memorizes ; not find pattern to generalize)

↳ overcome by
PCA : (w/o droping columns

What is PCA?

1. Principal component analysis
   ↳ is usl dimension reduction technique w/o droping column.
2. Principal component(s) are new axis formed.
3. PC is linear combination of all features.

$PC_1 = a_1x_1 + \cdots + a_nx_n$ (from eigen vector; from standardized data)

$PC_2 = b_1x_1 + \cdots + b_nx_n$

$PC_n = \cdots$

a_s, b_s are called loading / component / coefficient / eigen vector component

4. No. of PCs = No. of features / column.
5. PCs are orthogonal to each other
6. PCs are vectors
   ↳ so contains
   - Magnitude
     ↓ given by
     eigen value
     - ↳ also called explained variance
     - ↳ Captures the variances / variation
       ↳ i.e. information → i.e. Range (max − min)
   - direction
     ↓ given by
     eigen vector
     ↳ gives the direction of that PCs w.r.t original axes.
7. PC_1 always captures most maximum variance. (for datapoints value in other feature column)

   Order of variance / information / variation capturing is

   $PC_1 > PC_2 > \cdots$

8. Based on information, same ↔ variance/variance.

Why PCA?

1. dimension reduction w/o droping columns.
2. Reduce Noise.
3. Reduce multicolinearity

   ↓
   How?
   - as it is linear combination of features
   - droping PCs by selecting no. of PCs ≠ droping columns.

When to use

1. Only for Numerical columns / variables.
2. only if have Multicolinearity

How?

working in graphical

1. plot the data
2. Scale data → origin shift to mean
3. Rotate the axis till capture max. variance (PC_1)
4. Draw other PCs orthogonal to the PC_1.
5. drop the PCs that capturing least variance / info. / variation using the needful methods.

Algorithmic Steps

1. Standardize the data ($\mu = 0$, Std = 1)
2. Find covariance matrix.
3. Find eigen vector & eigen value. $AV = \lambda V$ (covariance)
4. Sort the eigen values in desc.
5. Select the eigen vectors based eigen value (i.e. PCs capturing maximum Amt of variance / variation / information)
6. Get the PCs by dot product of eigen vectors & standardized data.

A → Covariance matrix (is square matrix) = $\frac{S^T \times S}{(n-1)}$

V → eigen vector (vector)

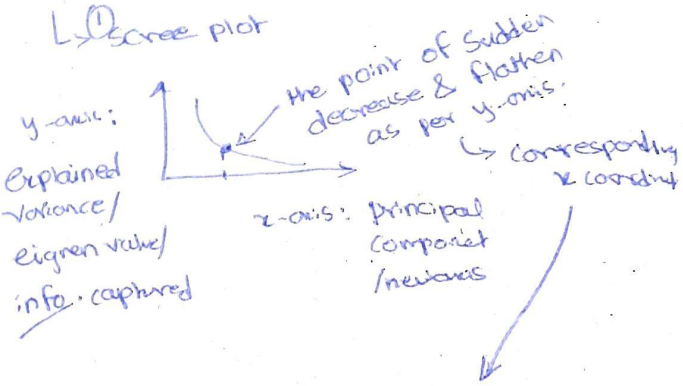
λ → eigen value. (Scalar)

How? → selecting ~~no of~~ optimal no. of PCs

↓

Three methods

↳ ① Scree plot



Limitation: Subjective

↳ ② Kaiser method

↳ PCs with eigenvalue (λ) >1 are selected.

↳ based on empirical experiment that if $\lambda > 1$ then we get atleast 60% explained variance/variation on total variance/variation

↳ limitation: if most of λs < 1 then it is not working as optimal.

↳ ③ percentage rule (most widely used)

↳ select PCs when arranged in descending order based on λ such that cummulative sum ≥ 80%.

Preference:

Percentage rule > Kaiser rule > Scree plot

Result/achivement

* dimension reduction w/o droping colums/features

↓ we drop PCs based on variance captured (λ) wrt total variance ($\leq \lambda$)

↓ we not drop features.

* So, new dataset we work with is colums as PCs, not original features/axis.

* For test data (if only linked to Supervised learning; as no testing in US L),

↳ we just transform based on fit on train data.

⇓

so that no data leakage blw train & test.

Two ways

↓

linear algebra way (one asked in GL exams) [not used in real world]

Sklearn way [preferred to use in real world]

⇓

PCA (n_component)

↓ 2 ways

discrete value → is no. of PCs.

Continuous value [0,1] → is % of explained variances captured.