WEEK-8

⇒ K-Nearest Neighbors

↳ KNN is a non-parametric method for classification.
↳ lazy learning algorithm → all computation is deferred until classification.
↳ instance based learning algorithm where the function is approx locally.

↳ simplest
↳ no explicit training.

↳ used when there are non linear decision boundaries bw classes.
↳ when the amount of data large.

→ Input features
  ↳ both quantitative and qualitative
→ Outputs
  ↳ categorical
    ↳ which are classes of data.

KNN explains a categorical values using the majority voks of nearest neighbors.

→ Assumptions:

↳ since it is non-parametric, there are no assumptions about the underlying data distribution.
↳ Select the parameter K based on the data.
↳ Requires a distance metric to define proximity blw any two data points.
eg: Euclidian dist, Mahalanobis distance, Hamming distance.

→ Algorithm:

there are 4 steps:

① Compute the distance metric blw the test data point and all the labeled data points
② Order the labeled data points in the Ising order of the distance metric
③ Select the top K labeled data points and look at the class labels.
④ Find the class label that the majority of these K

Labeled data points have and assign it to the test data point.

→ things to consider

1. parameter selection
2. presence of noise
3. Feature selection and scaling
4. Curse of dimensionality.

→ Parameter Selection:

↳ the best choice of k depends on the data
↳ Larger values of k reduce the effect of noise on classification but makes the decision boundaries b/w classes less distinct.
↳ smaller values of k tend to be affected by the noise, with clear separation b/w classes

Feature Selection:

- It is important to remove irrelevant features
- When the number of features is too large, and suspected to be highly redundant, feature extraction is required.
- If the features are carefully chosen then it is expected that the classification will be better.

K-~~means~~ nearest neighbours implementation in R:

```
Setwd("Path of the directory with data files")
rm(list=ls()) # clear the environment
library(caret) # for confusion matrix
library(class) # for Knn.

ServiceTrain <- read.csv("ServiceTrainData.csv")
ServiceTest <- read.csv("ServiceTestData.csv");
```

Knn(train, test, cl, K=1)

↳ no. of neighbours to be considered

cl → factor of true classifications of training set

K-means clustering

A technique to partition N observations in K clusters (K≤N) in which each observation belongs to cluster with nearest mean.

↳ one of the simplest unsupervised algorithms
↳ works well for all distance metrics where mean is defined (ex Euclidian distance)

Given N observations $(x_1, x_2 \dots x_N)$, K means clustering will partition n observations into k (K≤N) sets $S = \{S_1, \dots S_K\}$ so as to minimize the within cluster sum of squares (WCSS)

$$\arg\min_{S} \sum_{i=1}^{K} \sum_{x \in S_i} \|x - \mu_i\|^2$$

Where μ_i is the mean of points in S_i

→ Determing the number of clusters:

Elbow Method: looks at percentage of variance explained as a function of number of clusters

↳ The point where marginal decrease plateaus is an indicator of the optimal number of clusters

→ Disadvantages:

↳ algorithm could converge to a local minima, therefore role of initial position is very important.
↳ if the clusters are not spherical, then K means can fail to identify the correct number of clusters

IMPLEMENTATION IN R:

tripDetails = read.csv("tripDetails.csv", row.name=1)

- Kmeans clustering in R can be applied on data using 'Kmeans()' function

object = Kmeans(x, centers, iter.max=10, nstart=1)

- x: numeric matrix of data, or an object that can be coerced to such a matrix
- centers: either the no. of centers, or a set of initial centers.
- iter.max: max no. of iterations allowed.
- nstart: if centers is a number, how many sets should be chosen.

tripCluster = Kmeans(tripDetails, 3)

↳ elbow method

↳ method to calculate optimal K.

```
k.max ← 10
wss ← rep(NA, K.max)
nClust ← list()
for (i in 1:K.max) {
    driveClasses ⇒ ← Kmeans(tripDetails, i)
    wss[i] ← driveClasses$ tot. withinss
    nclust[[i]] ← driveClasses$ size
}
plot (1:K.max, wss, type = "b", pch = 19,
      xlab = "Number of clusters K",
      ylab = "Total within-clusters sum of
      squares : Trips")
```