Business Anaytics

M. Tech QROR – 2nd yr (2024)

Classification

(Predictive Analytics)

Dr. Prasun Das

SQC & OR Unit

Indian Statistical Institute

e-mail: prasun@isical.ac.in

Classification

Objective of a classification model is to predict a target variable that is *mostly* binary (e.g., a loan decision) or categorical (e.g., a customer type) when a set of input variables are given (e.g., credit score, income level, etc.).

Example:

Decision to play Golf (Yes / No) on the basis of temperature, humidity and outlook conditions (predictors)

Methods:

1. *Decision Trees*
2. *Rule Induction*
3. *Naïve Bayesian*
4. *K-nearest neighbour*
5. *ANN*
6. *Support Vector Machines (SVM)*
7. *Ensemble Learners*

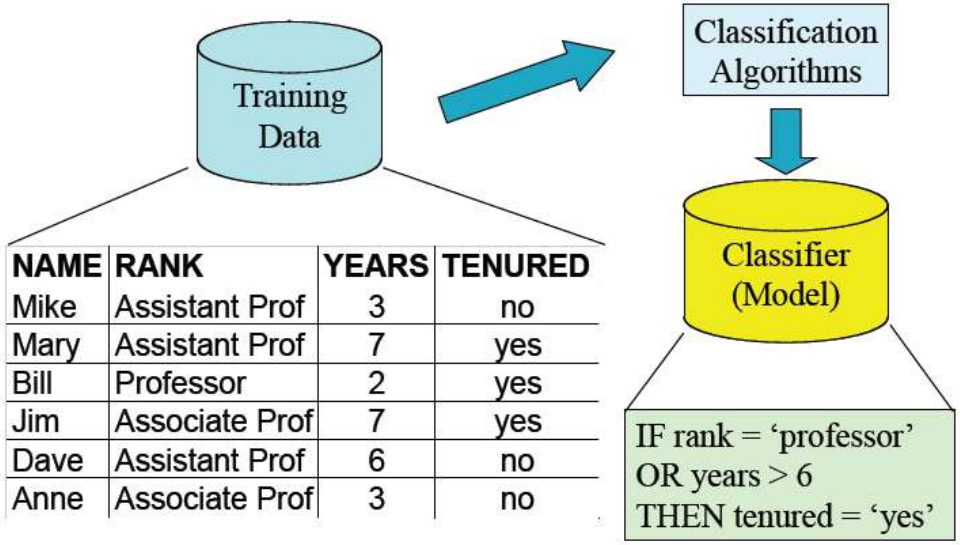
Performance Metric:

1. *Confusion Matrix*
2. *ROC Curves*
3. *Lift Charts*

Classification: 2-step process

Model construction: describing a set of predetermined classes

- Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
- The set of tuples used for model construction: training set
- The model is represented as classification rules, decision trees, or mathematical formulae

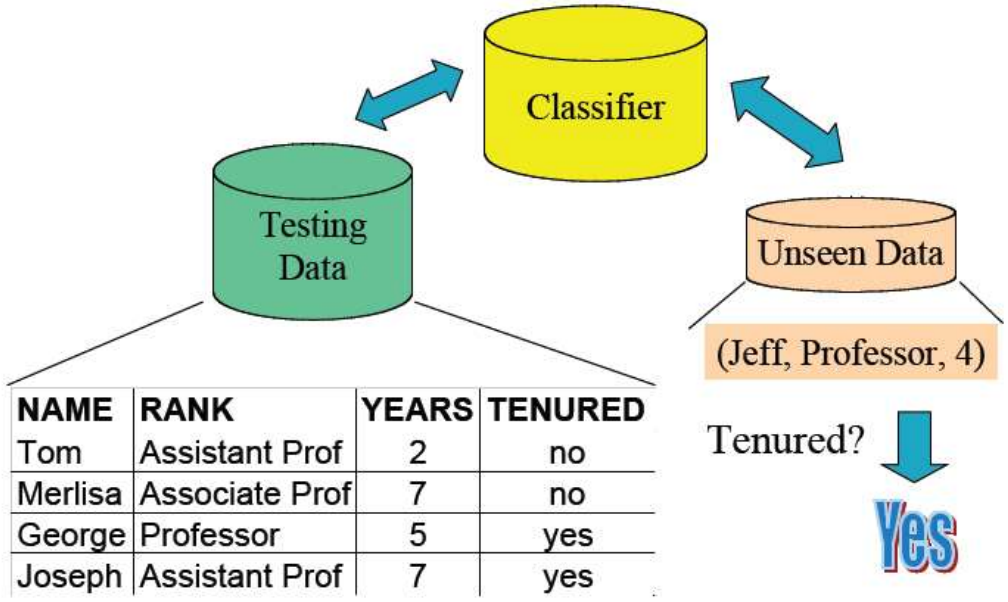


NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

Classification: 2-step process

Model usage: for classifying future or unknown objects

- Estimate accuracy of the model
  - The known label of test sample is compared with the classified result from the model
  - Accuracy rate is the percentage of test set samples that are correctly classified by the model



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Classification – Decision Tree

- **Decision Tree: Structure**
  - A tree structure, constructed in a top-down recursive divide-and-conquer manner
  - Internal node denotes a test on an attribute (categorical)
  - Branch represents an outcome of the test; Leaf nodes represent class labels
- **Decision tree : Two phase operation**
  - Tree construction
    - At start, all the training examples are at the root node
    - Partition examples, recursively, based on selected attributes.
    - Attributes are selected on the basis of a heuristic or statistical measure (e.g., Information Gain, Entropy, Gini index)
  - Tree pruning (post, basically)
    - Identify and remove branches that reflect noise or outliers
- **Decision tree: Use**
  - Test the attribute values of the unknown sample against the decision tree.

Decision Tree – Information Measures

Measure of Impurity (Randomness):

It meets certain criteria, based on computing a proportion of the data that belong to a class, as follows:

1. The measure of impurity of a data set must be at a **maximum** when all possible classes are equally represented.
2. The measure of impurity of a data set must be **zero** when only one class is represented [***ex***. choosing a yellow ball from a box of just yellow balls is displayed to have 0 entropy which implies 0 impurity or total purity]

Measures, such as, ***entropy*** *or* ***Gini index*** *easily meet these criteria and are used* to build decision trees. Different criteria will build different trees through different biases, for ex., ***information gain*** favors tree splits that contain many cases, while ***information gain ratio*** attempts to balance this.

Decision Tree – Information Measures

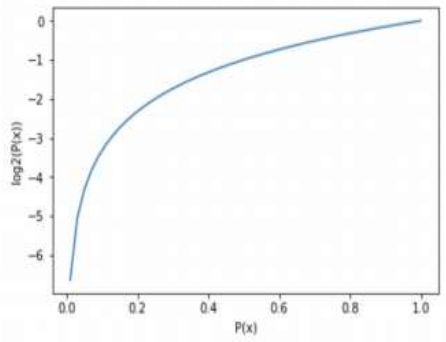
Entropy (H): [Claude Shannon, 1948**]**

Entropy is defined as **log(1/*P*)** or **-log *P*** where, ***P*** is the probability of an event occurring. If the probability for all events is not identical, we need a weighted expression and thus entropy, H, is adjusted as follows:

$$H = -\sum p_k \log_2 (p_k)$$

where k = 1, 2, 3, …, m represent the m classes of the target variable. The p_k represents the proportion of samples that belongs to class k. The **maximum** value for entropy depends on the number of classes $\{(m, H_{max}) = (2,1); (4,2); (8,3), (16,4)\}$.

Shannon's entropy model uses the logarithm function $\log_2(p(x)$ to measure the entropy because as the probability p(x) of randomly drawing an event increases, the result approaches closer to binary logarithm 1, as shown in the graph below.



Decision Tree – Information Measures

Example: Two variables, *age* and *weight*, that predict if a person is likely to sign up for a gym membership or not.

If this dataset had 100 samples with 50% of each, then the **Entropy(dataset)** is given by $H = -[(0.5 \log_2 (0.5)) + (0.5 \log_2 (0.5))] = -\log_2 (0.5) = 1$.

On the other hand, if we can partition the data into two sets of 50 samples each that contain all members and all nonmembers, the entropy of either of these two partitioned sets [**Entropy(feature)**] is given by $H = -1 \log_2 (1) = 0$.

Any other proportion of samples within a data set will yield entropy values between 0 and 1 for m=2.

Information Gain (IG):

$$\text{IG (feature)} = \text{Entropy}(\textit{dataset}) - \text{Entropy}(\textit{feature})$$

The feature with the **largest entropy information gain** should be the root node to build the decision tree.

Decision Tree – ID3 Algo

Original Decision Tree algorithm, the *Iterative Dichotomizer 3, or ID3* (Quinlan, 1986).

Two questions to answer, at each step of the tree building process:

Q1. where to split the data, and

Q2. when to stop splitting

The Classic Golf Data Set

Outlook	Temperature	Humidity	Windy	Play
sunny	85	85	FALSE	no
sunny	80	90	TRUE	no
overcast	83	78	FALSE	yes
rain	70	96	FALSE	yes
rain	68	80	FALSE	yes
rain	65	70	TRUE	no
overcast	64	65	TRUE	yes
sunny	72	95	FALSE	no
sunny	69	70	FALSE	yes
rain	75	80	FALSE	yes
sunny	75	70	TRUE	yes
overcast	72	90	TRUE	yes
overcast	81	75	FALSE	yes
rain	71	80	TRUE	no

Decision Tree – ID3 Algo

Splitting the Data w.r.t OUTLOOK:

$H_{outlook:overcast} = -(0/4)\log_2(0/4) - (4/4)\log_2(4/4) = 0.0$

$H_{outlook:sunny} = -(2/5)\log_2(2/5) - (3/5)\log_2(3/5) = 0.971$

$H_{outlook:rain} = -(3/5)\log_2(3/5) - (2/5)\log_2(2/5) = 0.971$

Information as a whole w.r.t OUTLOOK:

The total "information" is calculated as the weighted sum of the component entropies. There are four instances of **Outlook = overcast**, thus the proportion for overcast is given by $P_{outlook:overcast} = 4/14$. The other proportions (for **Outlook = sunny** & **rain**) are 5/14 each:

$$I_{outlook} = P_{outlook:overcast} * H_{outlook:overcast} + P_{outlook:sunny} * H_{outlook:sunny} + P_{outlook:rain} * H_{outlook:rain}$$

$$I_{outlook} = (4/14) * 0 + (5/14) * 0.971 + (5/14) * 0.971 = 0.693$$

Decision Tree – ID3 Algo

Information as a whole with no partition:

In that case, the total information would have been simply the weighted average of the respective entropies for the two classes whose overall proportions were 5/14 (**Play = no**) and 9/14 (**Play = yes**):

$I_{outlook,no\ partition} = -(5/14)\log_2(5/14) - (9/14)\log_2(9/14) = 0.940$

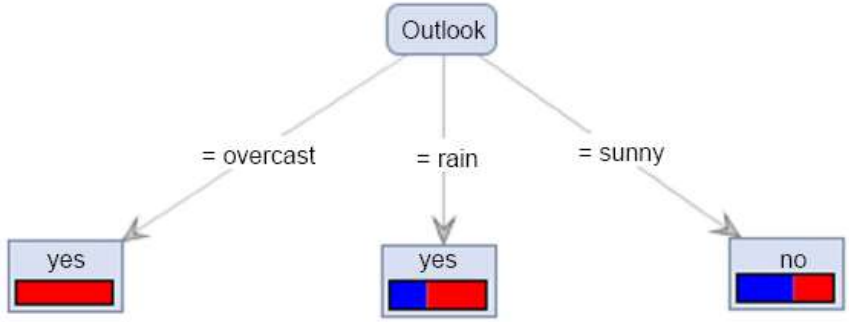
Information Gain:

It is nothing but reduction in entropy by creating splits/partitions.
In the case of **Outlook**, this is simply given by

$I_{outlook,no\ partition} - I_{outlook} = 0.940 - 0.693 = 0.247$

Decision Tree – ID3 Algo

Attribute	Information Gain
Temperature	0.009
Humidity	0.102
Wind	0.048
Outlook	**0.247**



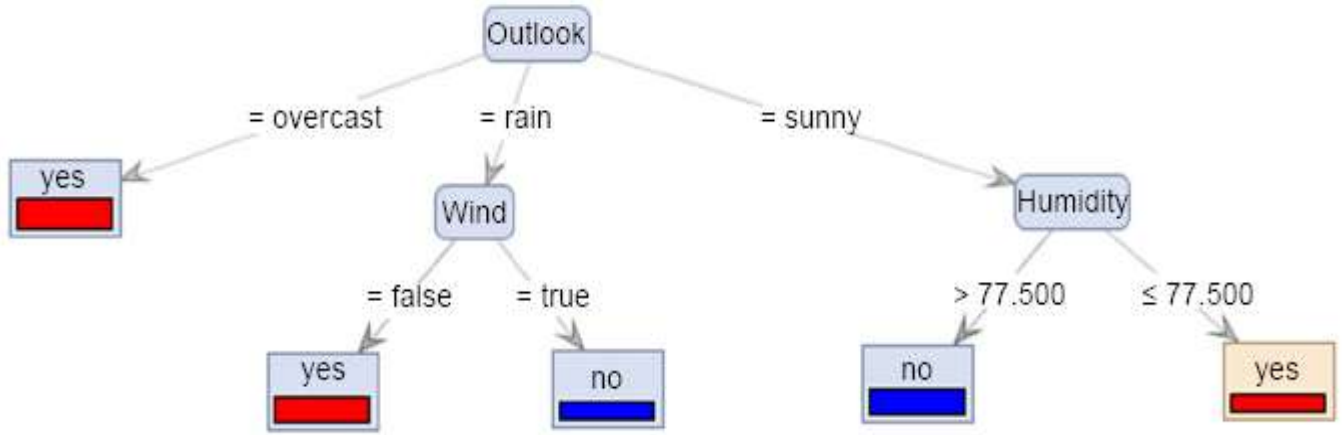
If we partition the data set into three sets along the three values of **Outlook**, we will experience the **largest information gain**. This gives the first node of the decision tree as shown above.

The terminal node for the **Outlook = overcast** branch consists of four samples, all of which belong to the class **Play = yes**. The other two branches contain a mix of classes.

Decision Tree – ID3 Algo

So, not all the final partitions are 100% homogenous. Hence, the same process could be applied for each of these subsets till we get “purer” results. So we revert back to the first question once again—**where to split the data?**

Simply use the other attributes that yielded the highest gains. Following the logic, the next split is along the **Outlook = sunny** branch w.r.t **Humidity** (the second highest information gain) and split along the **Outlook = rain** branch w.r.t **Wind** (the third highest gain). The fully grown tree is thus shown below.



Decision Tree – Summary

5-steps process for the application of the decision tree algorithm :

1) Using Shannon entropy, sort the data set into homogenous (by class) and non-homogenous variables. *Homogenous variables have* low information entropy and non-homogenous variables have high information entropy. **(calculation of $I_{outlook,no\ partition}$)**.
2) Weight the influence of each independent variable on the target or dependent variable using the entropy weighted averages (sometimes called joint entropy). **(calculation of $I_{outlook}$)**
3) Compute the information gain, which is essentially the reduction in the entropy of the target variable due to its relationship with each independent variable. **(calculation of $I_{outlook,no\ partition} - I_{outlook}$)**
4) The independent variable with the highest information gain will become the root" or the first node on which the data set is divided. **(calculation of the *information gain*)**.
5) Repeat this process for each variable for which the Shannon entropy is non-zero. If the entropy of a variable is zero, then that variable becomes a "leaf" node.

Decision Tree – Another Information Measure

Gini Index (*G*):

It is calculated by subtracting the sum of squared probabilities of each class from one. It favors larger partitions and is easy to implement, whereas information gain favors smaller partitions with distinct values.

$$Gini\ Index = 1 - \sum (P(x=k))^2$$

If a data set T is split into two subsets T_1 and T_2 with sizes N_1 and N_2 respectively, the gini index of the split data contains examples from n classes, the gini index $\mathbf{gini_{split}(T)}$ is defined as

$$gini_{split}(T) = \frac{N_1}{N} gini(T_1) + \frac{N_2}{N} gini(T_2)$$

The attribute provides the smallest $gini_{split}(T)$ is chosen to split the node.

Note: The value of ***G*** ranges between 0 and a maximum value of 1, but otherwise has properties identical to ***H***, and either of these formulations can be used to create partitions in the data (Cover, 1991).

Decision Tree – Splitting/Pruning

Where to stop *splitting* ?

- All samples for a given node belong to the same class (100% homogeneous) – *very unlikely in real world*.
- No attribute remains for further partitioning - majority voting is employed for classifying the leaf nodes.
- No attribute satisfies a minimum information gain threshold.
- A maximal depth is reached: as the tree grows larger, not only does interpretation get harder, but we run into a situation called **"overfitting."**
- There are less than a certain number of examples in the current subtree: again a mechanism to prevent ***overfitting***.

Decision Tree – *Overfitting*

Overfitting occurs when a model tries to memorize the training data with too many branches instead of generalizing the relationship between inputs and output variables. Overfitting often has the effect of performing very well on the training data set, but performing poorly on any new data previously unseen by the model.

To prevent overfitting, we may need to restrict tree growth or reduce it, using a process called *pruning. All of the stopping* techniques mentioned earlier constitute what is known as ***pre-pruning*** *the decision* tree, because the pruning occurs before or during the growth of the tree.

There is also a method to remove branches from a "fully grown" tree and get a sequence of progressively pruned trees that do not effectively change the classification error rates. This is called ***post-pruning***. Post-pruning may sometimes be a better option because it will not miss any small but potentially significant relationships while allowing the tree to reach its maximum depth. However, it requires additional computations, which may be wasted when the tree needs to be trimmed back.

k-NN: Principle of Decision Rule

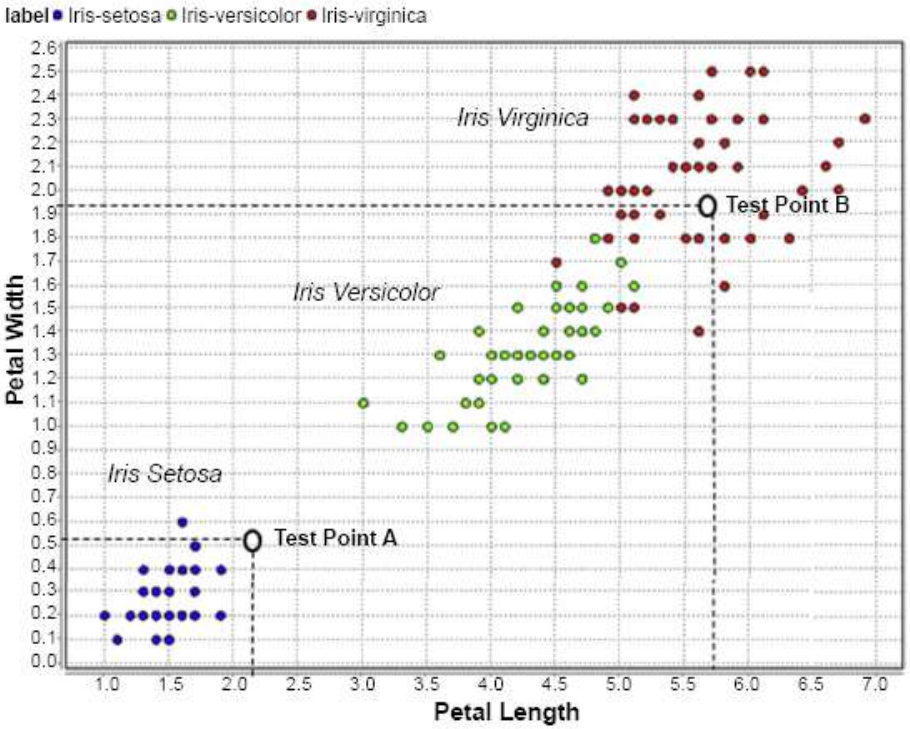
1. A nonparametric method, no generalization or attempt to find the distribution of the data set is made (Altman, 1992).
2. The entire training data set is "memorized" and, when unlabeled example records need to be classified, the input attributes of the new unlabeled records are compared against the entire training set to find a closest match.
3. The class label of the closest training record is the predicted class label for the unseen test record.

k-NN: Understanding thru visuals

Unseen test data point with (petal length, petal width) values

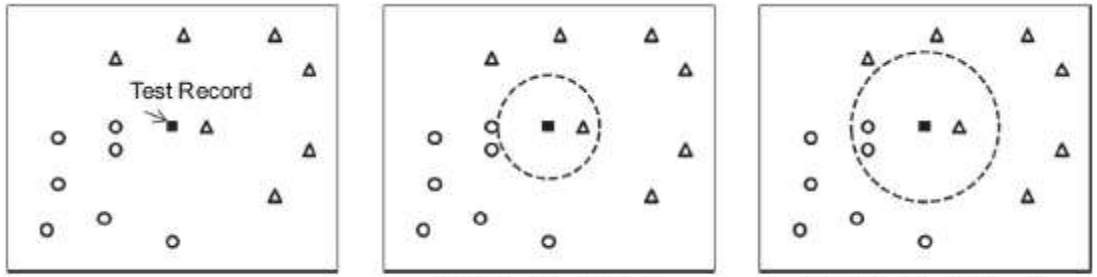
Data Point *A*(2.1,0.5): predicted species would be *Iris setosa since it is* in the neighborhood of other data points that belong to species *Iris setosa.*

Data Point *B*(5.7,1.9): it is in the neighborhood of *Iris versicolor, hence the test* data point can be classified as *Iris versicolor.*



Challenge: Consider a data point such as (5.0, 1.8), then the neighborhood has more than one species in the vicinity. classification can be tricky – Need efficient algorithm like *k*-NN.

k-NN: Understanding thru visuals



(a) Data set with a record of unknown class. (b) Decision boundary with k = 1 around unknown class record. (c) Decision boundary with k = 3 around unknown test record.

The **unlabeled test data** is the **dark square** in the center of the plot.
With *k* = 1, the predicted target class value of this test record is ***triangle*** *because the* ***closest*** *training* record is a **triangle**. But, what if the closest training record is an **outlier** with the incorrect class in the training set? Then, all the unlabeled test records near the outlier will get wrongly classified.
To prevent this misclassification,
Increase the value of *k* to, say, 3. When *k* = 3, *based on the majority* class of the nearest three training records, the predicted class of the test record is ***circle****. Since the class of the target record is evaluated* by voting, *k* is usually assigned an **odd number** for a **two-class** problem (Peterson, 2009).

k-NN: Working Rule

Suppose we are given n points $x_1, x_2, \ldots, x_n$. Let there be m classes. Let n_i of these sample points belong to class i; $i=1,2,\ldots,m$.

$$\sum_{i=1}^{m} n_i = n$$

Let x be the point to be classified. Let k be a positive integer. Find k nearest neighbors of x among $x_1,..,x_n$.

Let k_i of these nearest neighbors belong to i^{th} class; $i=1,2,\ldots,m$;

Put x in class i, if $k_i > k_j \quad \forall_j \neq i$

k-NN: Measures

The key task in the *k*-NN algorithm is determination of the nearest training record from the unlabeled test record using a **measure of proximity**. Once the nearest training record(s) are determined, the subsequent **voting** of the nearest training records is straightforward.

Measure of Proximity

The effectiveness of the *k*-NN algorithm hinges on the determination of how similar or dissimilar a test record is when compared with the memorized training record. A measure of proximity between two records is the measure of proximity of its attributes. To quantify similarity between two records, there is a range of techniques available such as calculating distance, correlation, Jaccard similarity, and cosine similarity (Tan, Michael, & Kumar, 2005).

k-NN: Distance Classifier

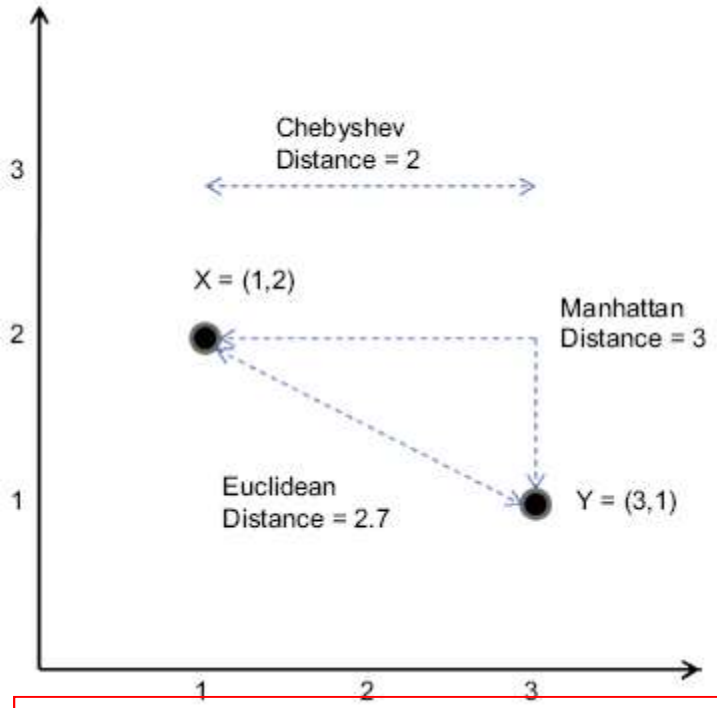
1. Euclidean Distance (p=2)
2. Manhattan Distance (p=1)
3. Chebyshev Distance ($p = \infty$)

4. **Minkowski Distance (p-norm distance)**

$$d = \left(\sum_{i=1}^{n} | x_i - y_i |^p \right)^{\frac{1}{p}}$$

The choice of distance measure depends on the data (Grabusts, 2011).

Euclidean measure: most commonly used for numeric data.
Manhattan (or, Hamming) distance: used for binary attributes.



The predicted target class y':

y' = maximum class (y_1, y_2, ⋯ , y_k), where y_i is the class outcome of i^{th} neighbor n_i.

k-NN: Distance Classifier

When $k > 1$, it can be argued that the closest neighbors should have more influence in determining the predicted target class than the far away neighbors (Hill & Lewicki, 2007). This can be accomplished by assigned weights for all the neighbors, with the weights increasing as the neighbors get closer to the data point. The weights are included in the final multi-voting step, where the predicted class is calculated.

Weights (w_i) should satisfy <u>two conditions</u>: they should be proportional to the distance of the data point from the neighbor and the sum of all weights should be equal to one.

$$w_i = \frac{e^{-d(x,n_i)}}{\sum_{i=1}^{k} e^{-d(x,n_i)}}$$

where w_i is the weight of i-th neighbor n_i, k the is total number of neighbors, and x is the test data point. The weight is used in predicting target class y':

y′ = maximum class (w_1*y_1, w_2*y_2, ⋯ ,w_k*y_k), where y_i is the class outcome of neighbor n_i.

k-NN: Distance Classifier

NOTE:

a) The distance measure works well for **numeric** attributes.

b) However, if the attribute is categorical (**nominal**), the distance between two points is either 0 or 1. If the attribute values are the same, the distance is 0 and if the attribute values are different, the distance is 1. In Golf Data Set example, distance (overcast, sunny) = 1 and distance (sunny, sunny) = 0.

c) If the attribute is **ordinal** with more than two values, then the ordinal values can be converted to the integer data type with values 0, 1, 2, …, n – 1 and the converted attribute can be treated as a numeric attribute for distance calculation. Obviously, converting ordinal into numeric retains more information than using it as a categorical data type, where the distance value is either 0 or 1.

k-NN: Correlation Similarity

Correlation between two data points X and Y is the measure of the **linear relationship** between the attributes X and Y.

$$\text{Correlation}\ (X, Y) = \frac{s_{xy}}{s_x * s_y}$$

Since correlation is a measure of "linear" relationship, a **zero value** doesn't mean there is no relationship. It just means that there is no linear relationship, but there may be a quadratic or any other higher degree relationship between the data points.

k-NN: Simple Matching Coefficient

The simple matching coefficient (SMC) is used when data sets have **binary** attributes.

Ex.: let ***X*** = (1,1,0,0,1,1,0) and ***Y*** = (1,0,0,1,1,0,0).

We can measure the similarity between these two data points based on simultaneous occurrence of 0 or 1 with respect to total occurrences. The simple matching coefficient for X and Y can be calculated as follows:

$$\text{Simple matching coefficient (SMC)} = \frac{matching\ occurences}{total\ occurences}$$

$$= \frac{m_{00} + m_{11}}{m_{10} + m_{01} + m_{11} + m_{00}}$$

where

m_{11} = #occurrences where X = 1 and Y = 1, i.e., m_{11} = 2

m_{10} = #occurrences where X = 1 and Y = 0 , i.e., m_{10} = 2

m_{01} = #occurrences where X = 0 and Y = 1 , i.e., m_{01} = 1

m_{00} = #occurrences where X = 0 and Y = 0 , i.e., m_{00} = 2

SMC = 4/7 (Check !!!)

k-NN: Jaccard Similarity

The Jaccard similarity measure is similar to the simple matching similarity but the nonoccurrence frequency (i.e., m_{00}) is ignored from the calculation.
If ***X*** and ***Y*** represent two text documents, each word can be an attribute in a data set called *document matrix* or *document vector*. In this application, the number of attributes would be very large in number. When comparing two documents ***X*** and ***Y***, most of the attribute values will be zero. This means that two documents do not contain the same rare words.
It is interesting to see the comparison of the *occurrence of the same word* and, since *nonoccurrence of* the same word doesn't convey any information, so we can ignore it.

Ex.: let ***X*** = (1,1,0,0,1,1,0) and ***Y*** = (1,0,0,1,1,0,0).

$$\text{Jaccard coefficient} = \frac{common\ occurences}{total\ occurences} = \frac{m_{11}}{m_{10} + m_{01} + m_{11}} = \frac{2}{5}$$

k-NN: Cosine Similarity

For the *document vector*s, as seen just now, attributes represent either the presence or absence of a word, which takes a binary form of either 1 or 0.

It is possible to construct a more informational vector (usually very long) with the number of occurrences in a document, instead of *occurrences* and *non-occurrences* (*binary*) denoted by 1 and 0 respectively.

Ex. Assume, ***X (doc_1)* =** (1,1,1,1,0,1,1,2): (*data, digital, economy, is, new, of, oil, the*) and, corresponding ***Y (doc_2)*** = (1,0,0,1,1,0,1,0).
Then, find out the cosine similarity measure for two these data points by using

$$\text{Cosine similarity}\,(|x \cdot y|) = \frac{x \cdot y}{\|x\|\,\|y\|}$$

NOTE: The choice of **distance or similarity measure** can also be **parameterized**, where multiple models are created with each different measure. The model with a distance measure that best fits the data with the smallest generalization error can be the appropriate distance measure for the data.
Question: What is the range of Cosine Similarity and interpretation of extreme values in terms of similarity of two documents ?