# MODEL QUESTIONS AND ANSWERS

# PATTERN RECOGNITION (PEC-IT602D)

**1. What is Pattern Matching? Write three main categories of Pattern Matching.**

Ans: Humans have evolved to be able to identify patterns and compare them to memories that we have stored. Pattern recognition, in its broadest sense, refers to the capacity to learn and recall patterns through repeated exposure. Pattern recognition in machine learning refers to matching the data from the database with the incoming data.
A fundamental idea in computer science, pattern matching is essential to the field of compiler design. It entails locating particular patterns or structures in data, which is essential for many jobs linked to compilers, such as lexical analysis, syntax parsing, and code optimization. Pattern matching techniques have advanced significantly in recent years, and this has had a major impact on compiler capabilities and efficiency. This blog investigates the significance of pattern matching in compilers, looks at new advancements in pattern matching techniques, and talks about how compiler designers use them.

<u>Three main categories of pattern matching:</u>

i. Supervised pattern matching: It is a technique in which a human trains a computer algorithm to identify patterns based on a predefined set of labeled data and then categorize new data.

ii. Unsupervised pattern matching: In this instance, the model learns without getting specific instructions. Based on their similarity, the algorithm looks for correlations between various data components (inputs). For better outcomes, supervised learning and unsupervised learning can be combined, or they can be utilized independently.

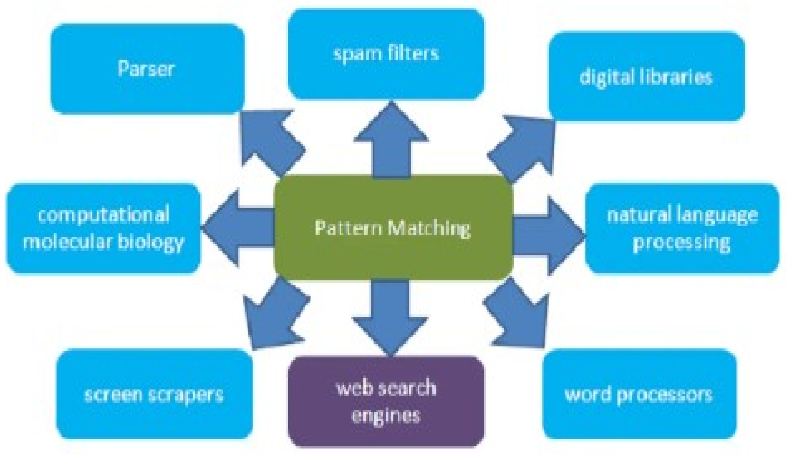
iii. Reinforcement learning: Through trial and error, the agent solves a problem via reinforcement learning. An environment is given to the AI, and it learns how to operate in that setting to get the best results. Read our article on the fundamentals of pattern recognition and ML for a thorough explanation of how pattern recognition functions.

**2. Discuss the significance of pattern matching in code refactoring and optimization.**
<u>Ans:</u> By discovering code patterns that can be improved, advanced pattern matching algorithms assist in automatic code refactoring, making the codebase more efficient and manageable.
As a result of recent developments in pattern matching methods, compilers are now much more capable of carrying out complicated and effective code transformations and optimization's. These applications make pattern matching a key component of contemporary compiler design and are essential for improving the performance, security, and maintainability of software systems. Refactoring can be performed after a product has been deployed, before adding updates and new features to existing code, or as a part of day-to-day programming. When the process is performed after deployment, it is normally done before developers move on to the next project. An organization may be able to refactor more code at this point in the software delivery lifecycle, where the developers have increased availability and more time to work on the source code changes needed. A better time to perform refactoring, though, is before adding updates or new features to existing code. When

performed at this point, refactoring makes it easier for developers to build onto the existing code because they are going back and simplifying the code, making it easier to read and understand.
When an organization has a strong grasp on the refactoring process, it can make it a regular process. Whenever a developer needs to add something to a code base, they can look at the existing code to see if it is structured in a way that would make the process of adding new code straightforward. If it is not, then the developer can refactor the existing code. Once the new code is added, the developer can refactor the same code again to make it clearer.
Refactoring can provide the following benefits:

- Makes the code easier to understand and read because the goal is to simplify code and reduce complexities.
- Improves maintainability and makes it easier to spot bugs or make further changes.
- Encourages a more in-depth understanding of code. Developers have to think further about how their code will mix with code already in the code base.
- Focus remains only on functionality. Not changing the code's original functionality ensures the original project does not lose scope.

3. **Explain the role of pattern matching in natural pattern recognition, such as identifying shapes or structures.**
Ans: Pattern matching is the technique that identifies specific patterns or sequences or a combination of characters within a larger piece of information.
The role of pattern matching include identification of phrases within the larger text, finding specific shapes within an image, distinguish sound patterns in the spoken language, search all instances of a particular gene within the larger set of DNA sequences, etc. Basically, pattern matching helps us in information retrieval from complex datasets, for implementing in fields such as data science, machine learning, & bioinformatics.
Some of the most notable and widely observed roles of pattern matching are highlighted below:



Text Mining and Information Extraction
In text mining, pattern matching is used to find relevant information from large datasets, such as finding specific names, dates, or phrases within a document corpus.
Computer Vision and Image Analysis
Pattern matching plays a critical role in computer vision and image analysis, where algorithms identify shapes, objects, or features within images based on predefined templates or learned patterns.
Bioinformatics and DNA Sequencing

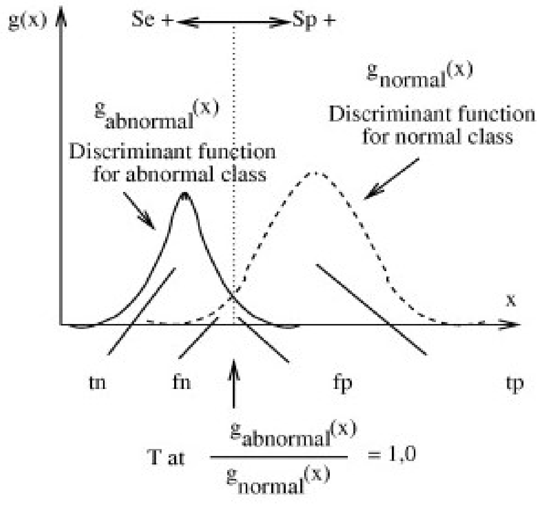
In bioinformatics, pattern matching is used to identify specific sequences or motifs in DNA, RNA, or protein data. It aids in the understanding of gene functions, disease mechanisms, and drug targeting.

4. **In what ways can pattern matching be advantageous in the field of bioinformatics?**
   Ans: In bioinformatics, pattern-matching algorithms are used to search for specific DNA or RNA sequences in a large database of genetic data. This helps researchers identify genes and their functions, which can lead to the development of new drugs and treatments for diseases.

DNA Processing

In Bioinformatics, identifying similar patterns has a major part in sequence alignment, sequence assembly, a search of patterns in a DNA sequence, sequence comparison, and many more. For detecting a similar pattern, EPM has to be modified to ignore a few mismatches.

5. **Design a Bayes classifier in terms of a set of discriminant functions.**
   Ans: A Bayesian classifier can be trained by determining the mean vector and the covariance matrices of the discriminant functions for the abnormal and normal classes from the training data. Instead of computing the maximum of the two discriminant functions $g_{abnormal}(x)$ and $g_{normal}(x)$, the decision was based on the ratio $g_{abnormal}(x)/g_{normal}(x)$. A decision threshold $T$ was set, such that if the ratio is larger than $T$ the unknown pattern vector is classified as abnormal, else as normal. By changing $T$, the sensitivity/specificity trade-off of the Bayes classifier can be altered. A larger $T$ will result in lower TP and FP rates, while a smaller $T$ will result in higher TP and FP rates. The procedure described is illustrated in figure below:


g(x)
Se +
Sp +
g_normal(x)
Discriminant function for normal class
g_abnormal(x)
Discriminant function for abnormal class
x
tn
fn
fp
tp
T at g_abnormal(x) / g_normal(x) = 1,0

6. **Write a short note on generalized linear discriminant function.**
   Ans: Linear discriminant functions can be solved in the context of dimensionality reduction. The problem of a two-class classification becomes finding the projection **w** that maximizes the separation between the projected classes. Let us assume that our data are 2*d* and we want to find a 1*d* projection

direction (embedded in the original 2*d* space) such that the separation between the projected classes is maximum. Geometrically the separation between classes is a measure that depends on the projected means and the projected scatter of the two classes; the further the classes are from each other, the larger the distance between their centroids, and the larger the separation. Similarly the smaller the scatter within each class, the further apart they become (and, therefore, the minimum the overlap between classes). We can use the ratio of class separation to total class scatter as a measure (objective) for class separation. This is called the Fisher criterion:

$$J_w = \frac{m_1 - {m_2}^2}{s_1^2 - s_2^2}$$

*where*

$$m_i \in R = \mathrm{w}^{\mathrm{T}} \frac{1}{N_i} \sum_{n \in \mathrm{Ci}} x_n$$

$$s_k^2 \in R = \sum_{n \in \mathrm{Ci}} y_n - {m_i}^2, \ \ldots \quad i = 1, 2$$

where $y_n \in R = \mathbf{w}^{\mathrm{T}} \mathbf{x}_n$ is the 1d projection of original data $\mathbf{x}$ and $s^2{}_k$ is the $k$th within-class variance (we use variance since data is now 1d). Equivalently, we can rewrite the criterion in terms of the within-class scatter, $\mathbf{S}_W$, and between-class scatter, $\mathbf{S}_B$:

$$Jw = \frac{w^{\mathrm{T}} S_B w}{w^{\mathrm{T}} S_W w}$$

where

$$\mathrm{S_B} = \mathrm{m}_2 - \mathrm{m}_1 \mathrm{m}_2 - {\mathrm{m}_1}^{\mathrm{T}}$$

$$\mathrm{S_W} = \sum_{\in C_1} \mathrm{x}_n - m_1 x_n - {m_1}^{\mathrm{T}} + \sum_{\in C_2} \mathrm{x}_n - m_2 x_n - {m_2}^{\mathrm{T}}$$

$$m_i \in R^d = \frac{1}{N} \sum_{n \in C_i} \mathrm{x}_n$$

It can be shown that maximizing the equation above with respect to **w** results in the following solution:

$$\omega \propto S_{\mathrm{W}}^{-1} \mathrm{m}_2 - \mathrm{m}_1$$

This is called the Fisher's linear discriminant. It can be shown that if the class distributions are MVNs with shared covariance, then the within-class scatter is nothing but a scaled MVN covariance matrix, Σ. If the classes were further isotropic, then the Fisher linear discriminant will result in the following solution:

$$\omega \propto \mathrm{m}_2 - \mathrm{m}_1$$

This is the same as the minimum distance classifier.

**7. Compare and contrast supervised and unsupervised learning.**

Ans:

| Supervised Learning | Unsupervised Learning |
|---|---|
| Supervised learning algorithms are trained using labeled data. | Unsupervised learning algorithms are trained using unlabeled data. |
| Supervised learning model takes direct feedback to check if it is predicting correct output or not. | Unsupervised learning model does not take any feedback. |
| Supervised learning model predicts the output. | Unsupervised learning model finds the hidden patterns in data |

| | |
|---|---|
| In supervised learning, input data is provided to the model along with the output. | In unsupervised learning, only input data is provided to the model. |
| The goal of supervised learning is to train the model so that it can predict the output when it is given new data. | The goal of unsupervised learning is to find the hidden patterns and useful insights from the unknown dataset. |
| Supervised learning needs supervision to train the model. | Unsupervised learning does not need any supervision to train the model. |
| Supervised learning can be categorized in **Classification** and **Regression** problems. | Unsupervised Learning can be classified in **Clustering** and **Associations** problems. |
| Supervised learning can be used for those cases where we know the input as well as corresponding outputs. | Unsupervised learning can be used for those cases where we have only input data and no corresponding output data. |
| Supervised learning model produces an accurate result. | Unsupervised learning model may give less accurate result as compared to supervised learning. |
| Supervised learning is not close to true Artificial intelligence as in this, we first train the model for each data, and then only it can predict the correct output. | Unsupervised learning is more close to the true Artificial Intelligence as it learns similarly as a child learns daily routine things by his experiences. |
| It includes various algorithms such as Linear Regression, Logistic Regression, Support Vector Machine, Multi-class Classification, Decision tree, Bayesian Logic, etc. | It includes various algorithms such as Clustering, KNN, and Apriori algorithm. |

**8. Describe the basic steps involved in the design of pattern recognition system.**

Ans: Pattern recognition involves three key steps: analyzing the input data, extracting patterns, and comparing it with the stored data. The process can be further divided into two phases:

1. **Explorative phase:** In this phase, computer algorithms tend to explore data patterns.
2. **Descriptive phase:** Here, algorithms group and attribute identified patterns to new data.

These phases can be further subdivided into the following modules:

i. **Data collection**

Data collection is the first step of pattern recognition. The accuracy of recognition is largely dependent on the quality of the datasets. As such, using open-source datasets is preferable and can save time instead of manual data collection processes. Thus, receiving data from the real world kick-starts the recognition process.

ii. **Pre-processing**

Once the data is received as input, the algorithms start the pre-processing step, where data is cleaned, and impurities are fixed to produce comprehensive datasets that yield good predictions. Pre-processing involves data segmentation. For instance, when you look at a group photograph posted by a friend on social media, you realize that you're familiar with some of the faces in the picture, which attracts your attention. This is what pre-processing means. Pre-processing is coupled with enhancement. For example, consider you are viewing the same photograph, but it is ten years older. Now, just to be sure that the familiar faces are real, you start comparing their eyes, skin tone, and other physical traits. This is where enhancement happens. It involves a smoothing and normalization process that tries to correct the image from strong variations. As a result, data becomes easy to interpret for the models.

Feature extraction

Next, features are extracted from the pre-processed input data. Here, the input data is converted into a feature vector, representing a reduced version of a set of features. This step solves the problem of the high dimensionality of the input dataset. This means that only relevant features are extracted rather than using the entire dataset. Once the features are extracted, you should select features with the highest potential of delivering accurate results. Upon shortlisting such features, they are sent for further classification.

iii. **Classification**

Extracted features are then compared to a similar pattern stored in the database. Here, learning can happen in a supervised and unsupervised manner. The supervised method has prior knowledge of each pattern category, while unsupervised method learning happens on the fly. As patterns are eventually matched to the stored data, the classification of input data happens.

iv. **Post-processing**

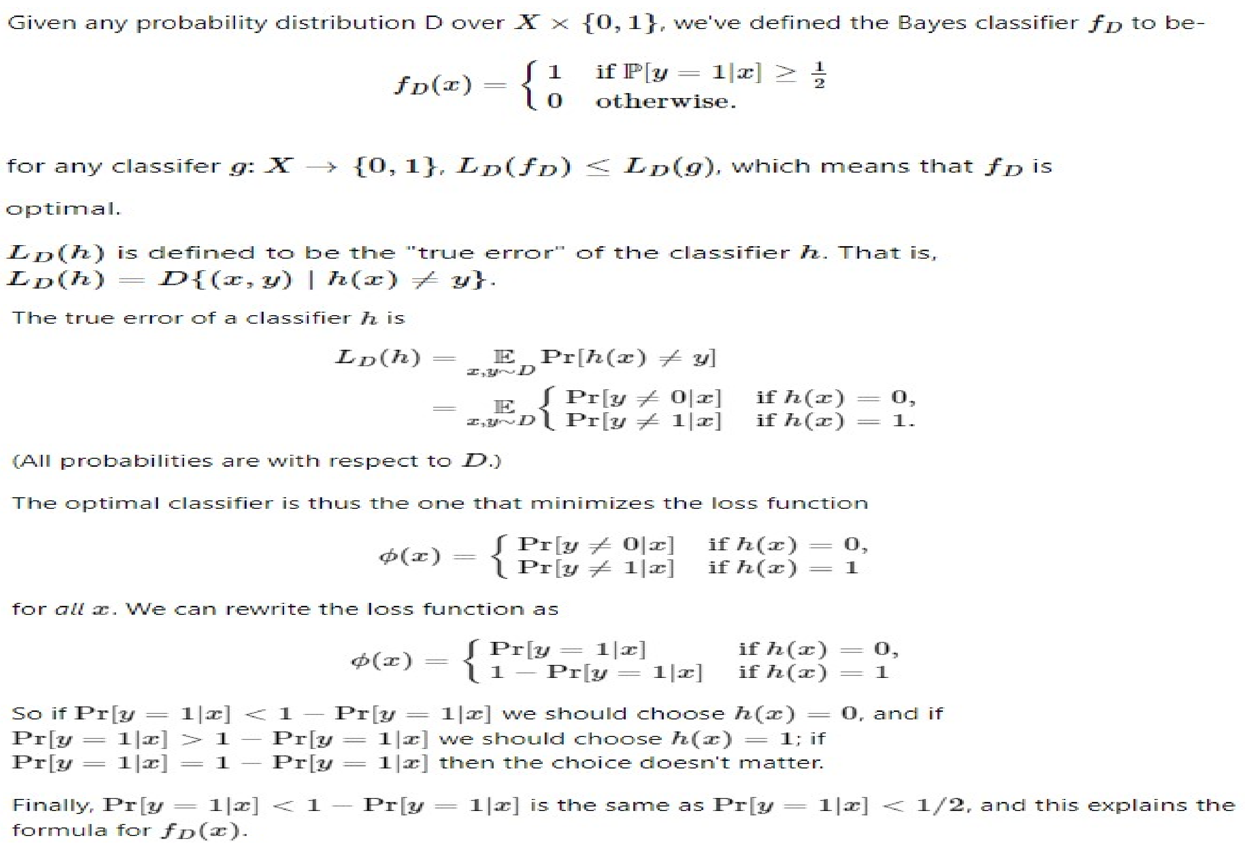
Classification is followed by a post-processing step, which makes decisions on the best ways to utilize the results to guide the system efficiently. Moreover, it involves analyzing each segment of the identified or classified data to derive further insights. These extracted insights are then implemented in practice for future pattern recognition tasks.

**9. What is Bayesian classifier? Prove that it is an optimal classifier.**

Ans: A **Bayesian classifier** is based on the idea that the role of a (natural) class is to predict the values of features for members of that class. Examples are grouped in classes because they have common values for the features. Such classes are often called **natural kinds**. In this section, the target feature corresponds to a discrete **class**, which is not necessarily binary.

A Bayesian classifier is a probabilistic model where the classification is a latent variable that is probabilistically related to the observed variables. Classification then become inference in the probabilistic model.

Consider domain X , label set Y= $\{0,1\}$ = $\{0,1\}$ and the zero-one loss.



Given any probability distribution D over $X \times \{0,1\}$, we've defined the Bayes classifier $f_D$ to be-

$$f_D(x) = \begin{cases} 1 & \text{if } \mathbb{P}[y=1|x] \geq \frac{1}{2} \\ 0 & \text{otherwise.} \end{cases}$$

for any classifer $g\colon X \rightarrow \{0,1\}$, $L_D(f_D) \leq L_D(g)$, which means that $f_D$ is optimal.

$L_D(h)$ is defined to be the "true error" of the classifier $h$. That is, $L_D(h) = D\{(x,y) \mid h(x) \neq y\}$.

The true error of a classifier $h$ is

$$\begin{aligned} L_D(h) &= \mathop{\mathbb{E}}_{x,y\sim D} \Pr[h(x) \neq y] \\ &= \mathop{\mathbb{E}}_{x,y\sim D} \begin{cases} \Pr[y \neq 0|x] & \text{if } h(x) = 0, \\ \Pr[y \neq 1|x] & \text{if } h(x) = 1. \end{cases} \end{aligned}$$

(All probabilities are with respect to $D$.)

The optimal classifier is thus the one that minimizes the loss function

$$\phi(x) = \begin{cases} \Pr[y \neq 0|x] & \text{if } h(x) = 0, \\ \Pr[y \neq 1|x] & \text{if } h(x) = 1 \end{cases}$$

for *all* $x$. We can rewrite the loss function as

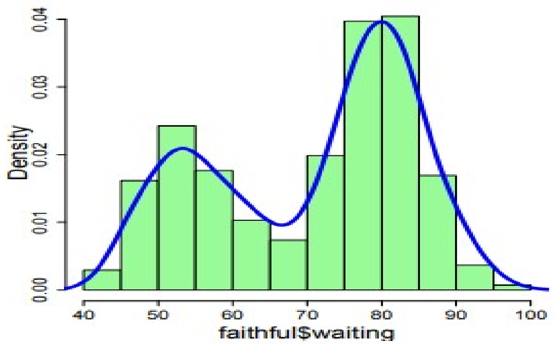
$$\phi(x) = \begin{cases} \Pr[y = 1|x] & \text{if } h(x) = 0, \\ 1 - \Pr[y = 1|x] & \text{if } h(x) = 1 \end{cases}$$

So if $\Pr[y=1|x] < 1 - \Pr[y=1|x]$ we should choose $h(x) = 0$, and if $\Pr[y=1|x] > 1 - \Pr[y=1|x]$ we should choose $h(x) = 1$; if $\Pr[y=1|x] = 1 - \Pr[y=1|x]$ then the choice doesn't matter.

Finally, $\Pr[y=1|x] < 1 - \Pr[y=1|x]$ is the same as $\Pr[y=1|x] < 1/2$, and this explains the formula for $f_D(x)$.

**10. What is density estimation ? What are the necessary conditions for its convergence ?**

Density estimation is the problem of reconstructing the probability density function using a set of given data points. Namely, we observe X1, · · · , Xn and we want to recover the underlying probability density function generating our dataset. A classical approach of density estimation is the histogram. The KDE(kernel density estimator) is one of the most famous method for density estimation. The follow picture shows the KDE and the histogram of the faithful dataset in R. The blue curve is the density curve estimated by the KDE.



**Necessary conditions for its convergence:**

Suppose that $X_1,\dots,X_n$ are samples drawn from a population with density function $f$ and $f_n(x)=f_n(x;X_1,\dots,X_n)$ is an estimate of $f(x)$, Denote by $m_{nr}=\int|f_n(x)-f(n)|^r\,dx$ and $M_{nr}=E(m_{nr})$ the Integrated $r$–th Order Error and Mean Integrated $r$–th Order Error of $f_n$ for some $r\geqslant 1$ (when $r=2$, they are the familiar and widely studied ISE and MISE), In this paper the same necessary and sufficient condition for $\lim\limits_{n\to\infty} Mnr = 0$ and $\lim\limits_{n\to\infty} m_{nr} = 0$ a.s. is obtained when $f_n(x)$ is the ordinary histogram estimator.

**11. What is Parzen Window? How to make decision using Parzen window method of density estimation?**

Ans: **Parzen Window** is a non-parametric density estimation technique. Density estimation in Pattern Recognition can be achieved by using the approach of the Parzen Windows. Parzen window density estimation technique is a kind of generalization of the histogram technique.

To make decision using Parzen window of density estimation:

-> Given samples $X_1{}^j, X_2{}^j, \dots, X_{d_j}{}^j$, for class $w_j$, i=1,2...,c

The total number of samples for all classes is $N$.

Choose $w_j$ such that $P(w_j|\overrightarrow{x_0}) \geq P(w_i|\overrightarrow{x_0})$, for all i=1,2,...,c

$$\Longleftrightarrow p(\overrightarrow{x_0}|w_j)P(w_j) \geq p(\overrightarrow{x_0}|w_i)P(w_i)\forall i$$

$$\Longleftrightarrow p(\overrightarrow{x_0}, w_j) \geq p(\overrightarrow{x_0}, w_i)\forall i$$

,where $p(\overrightarrow{x_0}, w_j) \simeq \frac{1}{N\cdot V}\sum_{l=1}^{d_j}\varphi(\frac{\overrightarrow{x_l{}^j}-\overrightarrow{x_0}}{h}) \geq \frac{1}{N\cdot V}\sum_{l=1}^{d_i}\varphi(\frac{\overrightarrow{x_l{}^i}-\overrightarrow{x_0}}{h})$

Therefore, $\sum_{l=1}^{d_j}\varphi(\frac{\overrightarrow{x_l^j}-\overrightarrow{x_0}}{h}) \geq \sum_{l=1}^{d_i}\varphi(\frac{\overrightarrow{x_l^i}-\overrightarrow{x_0}}{h})$

In the last equation, the first term represents $K_j$ around $x_0$, and the second term represents $K_i$ around $x_0$.

"Assign category of majority vote of neighbors", where total # of samples = $\sum_{i=1}^{d} d_i$.

Observation: Training error can be made=0 (by taking V small enough) but V too small=>spikiness =>high test error

**12. Compare Parzen Windows and k-Nearest Neighbor density estimation technique.**

Ans: Parzen Windows: Parzen Windows is a technique that estimates the PDF by means of placing a window (regularly a Gaussian kernel) round each information point and summing up the contributions from all of the windows. The width or bandwidth of the window determines the smoothness of the estimated density.

Mathematically, the Parzen Windows density estimator is described as follows:

$$\mathbf{f(x) = 1/n\ \sum^{n}_{i=1}\ 1/h^{2}\ K\,(x\text{-}x_{i}/h)}$$

where:

- f(x) is the estimated PDF at point x
- n is the number of data points.
- h is the width or bandwidth of the window.
- d is the dimensionality with the ith data point.
- xi represents the ith data point.
- K (.) is the kernel function, often a symmetric function such as the Gaussian kernel.

Application of Parzen Windows:

1. **Density Estimation:** Parzen Windows offer a flexible way to estimate the underlying chance density feature of a given dataset without making any assumptions about its distribution. It is especially useful while dealing with small or irregularly allotted datasets.
2. **Outlier Detection:** By estimating the density at every point, Parzen Windows can discover regions of low density, making it effective for outlier detection. Data factors that lie in regions with substantially decreased density can be taken into consideration as ability outliers.
3. **Pattern Recognition:** Parzen Windows can be used in sample recognition tasks, consisting of character reputation or photo segmentation. By estimating the densities of different lessons, it will become feasible to classify new instances based totally on their likelihoods.

**KNN:** The K-Nearest Neighbors (KNN) algorithm is a popular machine learning technique used for classification and regression tasks. It relies on the idea that similar data points tend to have similar labels or values. During the training phase, the KNN algorithm stores the entire training dataset as a reference.

**K-Nearest Neighbor density estimation technique:**

Suppose $X \in \mathbb{R}^q$ has multivariate density $f(x)$ and we are estimating $f(x)$ at $x$.
A multivariate uniform kernel is

$$w(\|u\|) = c_q^{-1} 1\left(\|u\| \leq 1\right)$$

where

$$c_q = \frac{\pi^{q/2}}{\Gamma\left(\frac{q+2}{2}\right)}$$

is the volume of unit ball in $\mathbb{R}^q$. If $q = 1$ then $c_1 = 2$.

Treating $R_x$ as a bandwidth and using this uniform kernel

$$\begin{aligned}\tilde{f}(x) &= \frac{1}{nR_x^q}\sum_{i=1}^{n} c_q^{-1}1\left(\|x - X_i\| \leq R_x\right) \\ &= \frac{1}{nR_x^q}\sum_{i=1}^{n} c_q^{-1}1\left(D_i \leq R_x\right)\end{aligned}$$

But as $R_x = D_{(k)}$ is the $k$'th order statistic for $D_i$, there are precisely $k$ observations where $\|x - X_i\| \leq R_x$. Thus the above equals

$$\tilde{f}(x) = \frac{k}{nR_x^q c_q}$$

To compute $\tilde{f}(x)$, all you need to know is $R_x$.

The estimator is inversely proportional to $R_x$. Intuitively, if $R_x$ is small this means that there are many observations near $x$, so $f(x)$ must be large, while if $R_x$ is large this means that there are not many observations near $x$, so $f(x)$ must be small.

A motivation for this estimator is that the effective number of observations to estimate $\tilde{f}(x)$ is $k$, which is constant regardless of $x$. This is in contrast to the conventional kernel estimator, where the effective number of observations varies with $x$.

While the traditional $k$-nn estimator used a uniform kernel, smooth kernels can also be used. A smooth $k$-nn estimator is

$$\tilde{f}(x) = \frac{1}{nR_x^q}\sum_{i=1}^{n} w\left(\frac{\|x - X_i\|}{R_x}\right)$$

where $w$ is a kernel weight function such that

$$\int_{\mathbb{R}^q} w\left(\|u\|\right)(du) = 1.$$

In this case the estimator does not simplify to a function of $R_x$ only

The analysis of $k$-nn estimates are complicated by the fact that $R_x$ is random.

The solution is to calculate the bias and variance of $\hat{f}(x)$ conditional on $R_x$, which is similar to treating $R_x$ as fixed. It turns out that the conditional bias and variance are identical to those of the standard kernel estimator:

$$\begin{aligned}E\left(\tilde{f}(x) \mid R_q\right) &\simeq f(x) + \frac{\kappa_2(w)\nabla^2 f(x)R_x^2}{2} \\ var\left(\tilde{f}(x) \mid R_q\right) &\simeq \frac{R(w)f(x)}{nR_x^q}.\end{aligned}$$

We can then approximate the unconditional bias and variance by taking expectations:

$$\begin{aligned}E\left(\tilde{f}(x)\right) &\simeq f(x) + \frac{\kappa_2(w)\nabla^2 f(x)}{2}E\left(R_x^2\right) \\ var\left(\tilde{f}(x)\right) &\simeq \frac{R(w)f(x)}{n}E\left(R_x^{-q}\right)\end{aligned}$$

We see that to evaluate these expressions we need the moments of $R_x = D_{(k)}$ the $k$'th order statistic for $D_i$. The distribution function for order statistics is well known. Asymptotic moments for the order statistics were found by Mack and Rosenblatt (Journal of Multivariate Analysis, 1979):

$$E\left(R_x^{\lambda}\right) \simeq \left(\frac{k/n}{c_q f(x)}\right)^{\lambda/q}$$

This depends on the ratio $k/n$ and the density $f(x)$ at $x$. Thus

$$\begin{aligned} E\left(R_x^2\right) &\simeq \left(\frac{k}{nc_q f(x)}\right)^{2/q} \\ E\left(R_x^{-q}\right) &\simeq \frac{c_q f(x) n}{k} \end{aligned}$$

Substituting,

$$\begin{aligned} Bias\left(\tilde{f}(x)\right) &\simeq \frac{\kappa_2(w)\nabla^2 f(x)}{2}\left(\frac{k}{nc_q f(x)}\right)^{2/q} \\ &= \frac{\kappa_2(w)\nabla^2 f(x)}{2\left(c_q f(x)\right)^{2/q}}\left(\frac{k}{n}\right)^{2/q} \end{aligned}$$

$$\begin{aligned} var\left(\tilde{f}(x)\right) &\simeq \frac{R(w) f(x)}{n}\frac{c_q f(x)}{k}n \\ &= \frac{R(w) c_q f(x)^2}{k} \end{aligned}$$

For $k$-nn estimation, the integer $k$ is similar to the bandwidth $h$ for kernel density estimation, except that we need $k \longrightarrow \infty$ and $k/n \longrightarrow 0$ as $n \longrightarrow \infty$.

The MSE is of order

$$MSE\left(\tilde{f}(x)\right) = O\left(\left(\frac{k}{n}\right)^{4/q} + \frac{1}{k}\right)$$

This is minimized by setting

$$k \sim n^{4/(4+q)}.$$

The optimal rate for the MSE is

$$MSE\left(\tilde{f}(x)\right) = O\left(n^{-4/(4+q)}\right)$$

which is the same as for kernel density estimation with a second-order kernel.

Kernel estimates $\hat{f}$ and $k$-nn estimates $\tilde{f}$ behave differently in the tails of $f(x)$ (where $f(x)$ is small). The contrast is

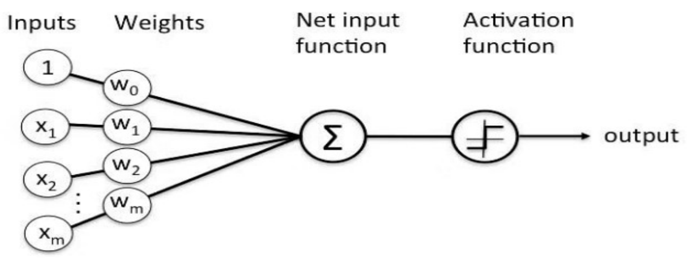
$$\begin{aligned} Bias\left(\hat{f}(x)\right) &\simeq \nabla^2 f(x) \\ Bias\left(\tilde{f}(x)\right) &\simeq \frac{\nabla^2 f(x)}{f(x)^{2/q}} \end{aligned}$$

$$\begin{aligned} var\left(\hat{f}(x)\right) &\simeq f(x) \\ var\left(\tilde{f}(x)\right) &\simeq f(x)^2 \end{aligned}$$

In the tails, where $f(x)$ is small, $\tilde{f}(x)$ will have larger bias but smaller variance than $\hat{f}(x)$. This is because the $k$-nn estimate uses more effective observations than the kernel estimator. It is difficult to rank one estimator versus the other based on this comparison. Another way of viewing this is that in the tails $\tilde{f}(x)$ will tend to be smoother than $\hat{f}(x)$.

**13.What is perceptron ? Discuss briefly the perceptron based learning algorithm.**

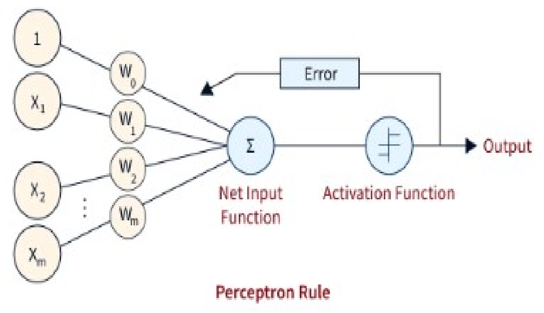
**Ans:** Perceptron is a single-layer neural network linear or a Machine Learning algorithm used for supervised learning of various binary classifiers. It works as an artificial neuron to perform computations by learning elements and processing them for detecting the business intelligence and capabilities of the input data.



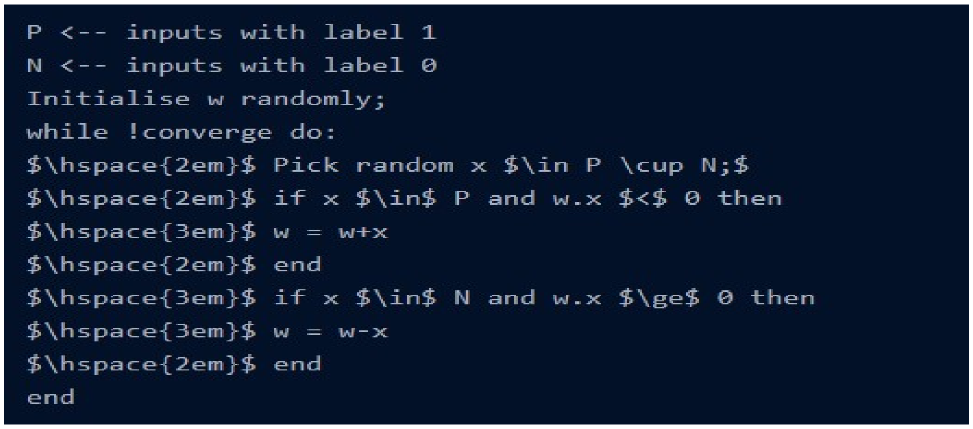
**There are four steps in perceptron based learning algorithm:**

1. First, multiply all input values with corresponding weight values and then add them to determine the weighted sum. Mathematically, we can calculate the weighted sum as follows: $\sum w_i * x_i = x_1 * w_1 + x_2 * w_2 + \ldots + w_n * x_n$. Add another essential term called bias 'b' to the weighted sum to improve the model performance. $\sum w_i * x_i + b$.
2. Next, an activation function is applied to this weighed sum, producing a binary or a continuous-valued output. $Y = f(\sum w_i * x_i + b)$
3. Next, the difference between this output and the actual target value is computed to get the error term, E, generally in terms of mean squared error. The steps up to this form the forward propagation part of the algorithm. $E = (Y - Y_{actual})^2$
4. We optimize this error (loss function) using an optimization algorithm. Generally, some form of gradient descent algorithm is used to find the optimal values of the hyper parameters like learning rate, weight, Bias, etc. This step forms the backward propagation part of the algorithm.

**An overview of this algorithm is illustrated in the following Figure:**



**The perceptron learning algorithm is as follows:**



```
P <-- inputs with label 1
N <-- inputs with label 0
Initialise w randomly;
while !converge do:
$\hspace{2em}$ Pick random x $\in P \cup N;$
$\hspace{2em}$ if x $\in$ P and w.x $<$ 0 then
$\hspace{3em}$ w = w+x
$\hspace{2em}$ end
$\hspace{3em}$ if x $\in$ N and w.x $\ge$ 0 then
$\hspace{3em}$ w = w-x
$\hspace{2em}$ end
end
```

**14.What is clustering ? Categorize the different clustering algorithms of the pattern recognition domain.**

Ans: Clustering is the task of dividing the unlabeled data or data points into different clusters such that similar data points fall in the same cluster than those which differ from the others. In simple words, the aim of the clustering process is to segregate groups with similar traits and assign them into clusters.

Types of clustering:

Clustering broadly divides into two subgroups:

- **Hard Clustering:** Each input data point either fully belongs to a cluster or not. For instance, in the example above, every customer is assigned to one group out of the ten.
- **Soft Clustering:** Rather than assigning each input data point to a distinct cluster, it assigns a probability or likelihood of the data point being in those clusters. For example, in the given scenario, each customer receives a probability of being in any of the ten retail store clusters.

Different types of clustering algorithm:
There are more than 100 clustering algorithms known. But few of the algorithms are used popularly. Those are:
<u>Connectivity Model:</u> These models are based on the notion that the data points closer in data space exhibit more similarity to each other than the data points lying farther away. These models can follow two approaches. In the first approach, they start by classifying all data points into separate clusters & then aggregating them as the distance decreases. In the second approach, all data points are classified as a single cluster and then partitioned as the distance increases. Also, the choice of distance function is subjective. These models are very easy to interpret but lack scalability for handling big datasets. Examples of these models are the hierarchical clustering algorithms and their variants.
<u>Centroid Model:</u> These clustering algorithms iterate, deriving similarity from the proximity of a data point to the centroid or cluster center. The k-Means clustering algorithm, a popular example, falls into this category. These models necessitate specifying the number of clusters beforehand, requiring prior knowledge of the dataset. They iteratively run to discover local optima.
<u>Distribution Model:</u> These clustering models are based on the notion of how probable it is that all data points in the cluster belong to the same distribution (For example: Normal, Gaussian). These models often suffer from overfitting. A popular example of these models is the Expectation-maximization algorithm which uses multivariate normal distributions.
<u>Density Model:</u> These models search the data space for areas of the varied density of data points in the data space. They isolate different dense regions and assign the data points within these regions to the same cluster. Popular examples of density models are DBSCAN and OPTICS. These models are particularly useful for identifying clusters of arbitrary shape and detecting outliers, as they can detect and separate points that are located in sparse regions of the data space, as well as points that belong to dense regions.

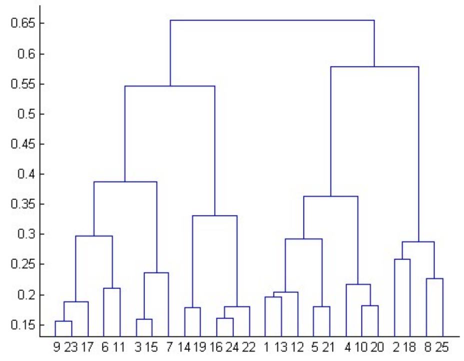
<u>K Means Clustering</u>: **K means is an iterative clustering algorithm that aims to find local maxima in each iteration. This algorithm works in these 5 steps:**

1. Specify the desired number of clusters K: Let us choose k=2 for these 5 data points in 2-D space.
2. Randomly assign each data point to a cluster: Let's assign three points in cluster 1, shown using red color, and two points in cluster 2, shown using grey color.
3. Compute cluster centroids: The centroid of data points in the red cluster is shown using the red cross, and those in the grey cluster using a grey cross.
4. Re-assign each point to the closest cluster centroid: Note that only the data point at the bottom is assigned to the red cluster, even though it's closer to the centroid of the grey cluster. Thus, we assign that data point to the grey cluster.
5. Re-compute cluster centroids: Now, re-computing the centroids for both clusters. Repeat steps 4 and 5 until no improvements are possible: Similarly, we'll repeat the 4th and 5th steps until we'll reach global optima, i.e., when there is no further switching of

data points between two clusters for two successive repeats. It will mark the termination of the algorithm if not explicitly mentioned.

Hierarchical Clustering: Hierarchical clustering, as the name suggests, is an algorithm that builds a hierarchy of clusters. This algorithm starts with all the data points assigned to a cluster of their own. Then two nearest clusters are merged into the same cluster. In the end, this algorithm terminates when there is only a single cluster left.

The results of hierarchical clustering can be shown using a dendrogram. The dendrogram can be interpreted as:



- Here is a live coding window where you can try out K Means Algorithm using the scikit-learn

  library.

**15. What is the difference between K Means and Hierarchical Clustering?**

- Ans: Hierarchical clustering can't handle big data well, but K Means can. This is because the time complexity of K Means is linear, i.e., O(n), while that of hierarchical is quadratic, i.e., O(n2).
- Since we start with a random choice of clusters, the results produced by running the algorithm multiple times might differ in K Means clustering. While in Hierarchical clustering, the results are reproducible.
- K Means is found to work well when the shape of the clusters is hyperspherical (like a circle in 2D or a sphere in 3D).
- K Means clustering requires prior knowledge of K, i.e., no. of clusters you want to divide your data into. But, you can stop at whatever number of clusters you find appropriate in hierarchical clustering by interpreting the dendrogram.

**16. Explain Fuzzy-C-means clustering algorithm. Write a short note about its criterion function.**

**Ans:** This algorithm works by assigning membership to each data point corresponding to each cluster center on the basis

of distance between the cluster center and the data point. More the data is near to the cluster center more is its

membership towards the particular cluster center. Clearly, summation of membership of each data point should be equal to one. After each iteration membership and cluster centers are updated according to the formula:

$$\mu_{ij} = 1 / \sum_{k=1}^{c} (d_{ij} / d_{ik})^{(2/m-1)}$$

$$v_j = (\sum_{i=1}^{n} (\mu_{ij})^m x_i) / (\sum_{i=1}^{n} (\mu_{ij})^m), \forall j = 1, 2, \ldots.c$$

where,

*'n'* is the number of data points.

*'vj'* represents the *j*th cluster center.

*'m'* is the fuzziness index m € [1, ∞].

*'c'* represents the number of cluster center.

*'µij'* represents the membership of *i*th data to *j*th cluster center.

*'dij'* represents the Euclidean distance between *i*th data and *j*th cluster center.

Main objective of fuzzy c-means algorithm is to minimize:

$$J(U,V) = \sum_{i=1}^{n} \sum_{j=1}^{c} (\mu_{ij})^m \left\| x_i - v_j \right\|^2$$

where,

*'$||x_i - v_j||$'* is the Euclidean distance between *i*th data and *j*th cluster center.

**Algorithmic steps for Fuzzy c-means clustering**

Let X = {$x_1$, $x_2$, $x_3$ ..., $x_n$} be the set of data points and V = {$v_1$, $v_2$, $v_3$ ..., $v_c$} be the set of centers.

1) Randomly select *'c'* cluster centers.

2) Calculate the fuzzy membership *'$\mu_{ij}$'* using:

$$\mu_{ij} = 1 / \sum_{k=1}^{c} (d_{ij} / d_{ik})^{(2/m-1)}$$

3) Compute the fuzzy centers *'$v_j$'* using:

$$v_j = (\sum_{i=1}^{n} (\mu_{ij})^m x_i) / (\sum_{i=1}^{n} (\mu_{ij})^m), \forall j = 1, 2, \ldots.c$$

**4)** Repeat step 2) and 3) until the minimum *'J'* value is achieved or $||U^{(k+1)} - U^{(k)}|| < \beta$.

where,

*'k'* is the iteration step.

*'β'* is the termination criterion between [0, 1].

*'$U = (\mu_{ij})_{n*c}$'* is the fuzzy membership matrix.

*'J'* is the objective function.

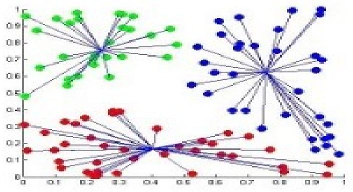


**Fig.: Result of Fuzzy c-means clustering**

Criterions are helpful to train a neural network. Given an input and a target, they compute a gradient according to a given loss function. AbsCriterion and MSECriterion are perfect for regression problems, while ClassNLLCriterion is the criterion of choice when dealing with classification. Criterions are serializable.

**17. What is feature generation ? Write a short note on principal component analysis.**

Ans: Feature generation is the process of constructing new features from existing ones. The goal of feature generation is to derive new combinations and representations of our data that might be useful to the machine learning model. By generating polynomial features, we can uncover potential new relationships between the features and the target and improve the model's performance.

**Principal Component Analysis(PCA)**

As the number of features or dimensions in a dataset increases, the amount of data required to obtain a statistically significant result increases exponentially. This can lead to issues such as overfitting, increased computation time, and reduced accuracy of machine learning models this is known as the curse of dimensionality problems that arise while working with high-dimensional data. As the number of dimensions increases, the number of possible combinations of features increases exponentially, which makes it computationally difficult to obtain a representative sample of the data and it becomes expensive to perform tasks such as clustering or classification because it becomes. Additionally, some machine learning algorithms can be sensitive to the number of dimensions, requiring more data to achieve the same level of accuracy as lower-dimensional data.
To address the curse of dimensionality, Feature engineering techniques are used which include feature selection and feature extraction. Dimensionality reduction is a type of feature extraction technique that aims to reduce the number of input features while retaining as much of the original information as possible.
The PCA algorithm is based on some mathematical concepts such as:

- Variance and Covariance
- Eigenvalues and Eigen factors

Some common terms used in PCA algorithm:

- **Dimensionality:** It is the number of features or variables present in the given dataset. More easily, it is the number of columns present in the dataset.
- **Correlation:** It signifies that how strongly two variables are related to each other. Such as if one changes, the other variable also gets changed. The correlation value ranges from -1 to +1. Here, -1 occurs if variables are inversely proportional to each other, and +1 indicates that variables are directly proportional to each other.
- **Orthogonal:** It defines that variables are not correlated to each other, and hence the correlation between the pair of variables is zero.
- **Eigenvectors:** If there is a square matrix M, and a non-zero vector v is given. Then v will be eigenvector if Av is the scalar multiple of v.
- **Covariance Matrix:** A matrix containing the covariance between the pair of variables is called the Covariance Matrix.

Principal Components in PCA:

As described above, the transformed new features or the output of PCA are the Principal Components. The number of these PCs are either equal to or less than the original features present in the dataset. Some properties of these principal components are given below:

- The principal component must be the linear combination of the original features.
- These components are orthogonal, i.e., the correlation between a pair of variables is zero.
- The importance of each component decreases when going to 1 to n, it means the 1 PC has the most importance, and n PC will have the least importance.

Steps for PCA algorithm

1. **Getting the dataset**
   Firstly, we need to take the input dataset and divide it into two subparts X and Y, where X is the training set, and Y is the validation set.
2. **Representing data into a structure**
   Now we will represent our dataset into a structure. Such as we will represent the two-dimensional matrix of independent variable X. Here each row corresponds to the data items, and the column corresponds to the Features. The number of columns is the dimensions of the dataset.
3. **Standardizing the data**
   In this step, we will standardize our dataset. Such as in a particular column, the features with high variance are more important compared to the features with lower variance.
   If the importance of features is independent of the variance of the feature, then we will divide each data item in a column with the standard deviation of the column. Here we will name the matrix as Z.
4. **Calculating the Covariance of Z**
   To calculate the covariance of Z, we will take the matrix Z, and will transpose it. After transpose, we will multiply it by Z. The output matrix will be the Covariance matrix of Z.
5. **Calculating the Eigen Values and Eigen Vectors**
   Now we need to calculate the eigenvalues and eigenvectors for the resultant covariance matrix Z. Eigenvectors or the covariance matrix are the directions of the axes with high information. And the coefficients of these eigenvectors are defined as the eigenvalues.
6. **Sorting the Eigen Vectors**
   In this step, we will take all the eigenvalues and will sort them in decreasing order, which means from largest to smallest. And simultaneously sort the eigenvectors accordingly in matrix P of eigenvalues. The resultant matrix will be named as P*.
7. **Calculating the new features Or Principal Components**
   Here we will calculate the new features. To do this, we will multiply the P* matrix to the Z. In the resultant matrix Z*, each observation is the linear combination of original features. Each column of the Z* matrix is independent of each other.
8. **Remove less or unimportant features from the new dataset.**
   The new feature set has occurred, so we will decide here what to keep and what to remove. It means, we will only keep the relevant or important features in the new dataset, and unimportant features will be removed out.

Applications of Principal Component Analysis

- PCA is mainly used as the dimensionality reduction technique in various AI applications such **as computer vision, image compression, etc.**
- It can also be used for finding hidden patterns if data has high dimensions. Some fields where PCA is used are Finance, data mining, Psychology, etc.

**18. Explain one suboptimal feature subset selection technique.**

Ans: **eature Selection** is the most critical pre-processing activity in any machine learning process. It intends to select a subset of attributes or features that makes the most meaningful contribution to a machine learning activity. In order to understand it, let us consider a small example i.e. **Predict the**

**weight of students based on the past information about similar students**, which is captured inside a 'Student Weight' data set. The data set has 04 features like **Roll Number, Age, Height & Weight.** Roll Number has no effect on the weight of the students, so we eliminate this feature. So now the new data set will be having only 03 features. This subset of the data set is expected to give better results than the full set.

| Age | Height | Weight |
|---|---|---|
| 12 | 1.1 | 23 |
| 11 | 1.05 | 21.6 |
| 13 | 1.2 | 24.7 |
| 11 | 1.07 | 21.3 |
| 14 | 1.24 | 25.2 |

**The above data set is a reduced dataset.** Before proceeding further, we should look at the fact why we have reduced the dimensionality of the above dataset OR what are the issues in **High Dimensional Data? High Dimensional** refers to the high number of variables or attributes or features present in certain data sets, more so in the domains like DNA analysis, geographic information system (GIS), etc. It may have sometimes hundreds or thousands of dimensions which is not good from the machine learning aspect because it may be a big challenge for any ML algorithm to handle that. On the other hand, a high quantity of computational and a high amount of time will be required. Also, a model built on an extremely high number of features may be very difficult to understand. **For these reasons, it is necessary to take a subset of the features instead of the full set.** So we can deduce that the objectives of feature selection are:

1. Having a faster and more cost-effective (less need for computational resources) learning model
2. Having a better understanding of the underlying model that generates the data.
3. Improving the efficacy of the learning model.

**<u>Main Factors Affecting Feature Selection</u>**

a. **Feature Relevance:** In the case of supervised learning, the input data set (which is the training data set), has a class label attached. A model is inducted based on the training data set — so that the inducted model can assign class labels to new, unlabeled data. Each of the predictor variables, ie expected to contribute information to decide the value of the class label. In case of a variable is not contributing any information, it is said to be irrelevant. In case the information contribution for prediction is very little, the variable is said to be weakly relevant. The remaining variables, which make a significant contribution to the prediction task are said to be strongly relevant variables.

b. **Feature Redundancy:** A feature may contribute to information that is similar to the information contributed by one or more features. For example, in the Student Data-set, both the features **Age & Height** contribute similar information. This is because, with an increase in age, weight is expected to increase. Similarly, with the increase in Height also weight is expected to increase. So, in context to that problem, Age and Height contribute similar information. In other words, irrespective of whether the feature **Height** is present or not, the learning model will give the same results. In this kind of situation where one feature is similar to another feature, **the feature is said to be potentially redundant** in the context of a machine learning problem.

**19. Compare parametric and non-parametric techniques.**

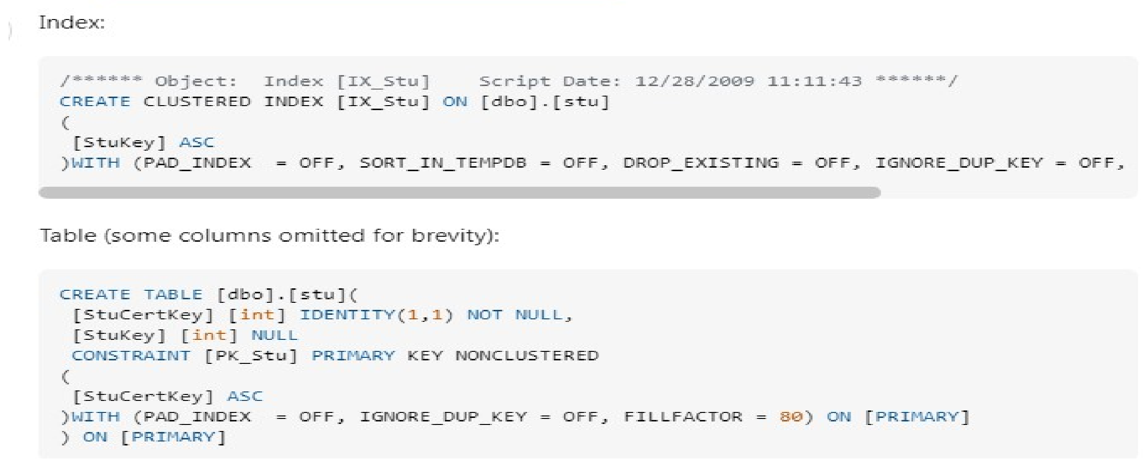
Ans: The key **difference between parametric and nonparametric technique** is that the parametric technique relies on statistical distributions in data whereas nonparametric do not depend on any distribution. Non-parametric does not make any assumptions and measures the central tendency with the median value. Some examples of non-parametric tests include Mann-Whitney, Kruskal-Wallis, etc.

Parametric is a statistical technique which assumes parameters and the distributions about the population are known. It uses a mean value to measure the central tendency. These techniques are common, and therefore the process of performing research is simple.

**20. Define performance index in cluster-seeking problem.**

**Ans:** A clustered index is a special type of index that determines the physical order of the rows in the table. There can be only one clustered index per table, and it usually contains the primary key column or columns. The clustered index is also called the row locator, because it defines how the data is stored and accessed on the disk.

To improve the performance on an *Index Seek:*



Index:

```
/****** Object:  Index [IX_Stu]    Script Date: 12/28/2009 11:11:43 ******/
CREATE CLUSTERED INDEX [IX_Stu] ON [dbo].[stu]
(
 [StuKey] ASC
)WITH (PAD_INDEX  = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, IGNORE_DUP_KEY = OFF,
```

Table (some columns omitted for brevity):

```
CREATE TABLE [dbo].[stu](
 [StuCertKey] [int] IDENTITY(1,1) NOT NULL,
 [StuKey] [int] NULL
 CONSTRAINT [PK_Stu] PRIMARY KEY NONCLUSTERED
(
 [StuCertKey] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF, FILLFACTOR = 80) ON [PRIMARY]
) ON [PRIMARY]
```

**21. What is the kernel trick in SVMs and how does it transform non-linearly separable data for accurate classification in machine learning?**

Ans: Support Vector Machines (SVM) is a popular and effective machine learning algorithm used in classification and regression tasks. It works by finding the best possible boundary that can separate two classes of data points with maximum margin, also known as the hyperplane. SVM utilizes kernel functions to map the input data points into a higher-dimensional space where the separation between the two classes becomes easier. This allows SVM to solve complex non-linear problems as well. SVM has been shown to be very effective in many real-world applications such as text classification, image classification, and bioinformatics. Its ability to handle high-dimensional feature spaces, excellent generalization performance, and robustness to noisy data make it one of the most preferred algorithms in the field of machine learning.

Kernel Trick : The "Kernel Trick" is a method used in Support Vector Machines (SVMs) to convert data (that is not linearly separable) into a higher-dimensional feature space where it may be linearly separated. This technique enables the SVM to identify a hyperplane that separates the data with the maximum margin, even when the data is not linearly separable in its original space. The kernel functions are used to compute the inner product between pairs of points in the transformed feature space without explicitly computing the transformation itself. This makes it computationally efficient to deal with high dimensional feature spaces.

The most widely used kernels in SVM are the linear kernel, polynomial kernel, and Gaussian (radial basis function) kernel. The choice of kernel relies on the nature of the data and the job at hand. The linear kernel is used when the data is roughly linearly separable, whereas the polynomial kernel is used when the data has a complicated curved border. The Gaussian kernel is employed when the data has no clear boundaries and contains complicated areas of overlap.

**22. Show that the maximum likelihood (ML) estimation of the mean for a Gaussian is unbiased but the ML estimate of variance is biased (i.e., slightly wrong). Show how to correct this variance estimate so that it is unbiased.**

**Solution**: Sample mean is unbiased:

$$\begin{aligned} E[\hat{\mu}] &= E[\frac{1}{n}\sum_{i=1}^{n} x_i] \\ &= \frac{1}{n}\sum_{i=1}^{n} E[x_i] \\ &= \frac{1}{n}\sum_{i=1}^{n} \mu \\ &= \frac{1}{n} n\mu \\ &= \mu \end{aligned}$$

This document will show two ways to prove that the sample variance is biased. Proof 1:

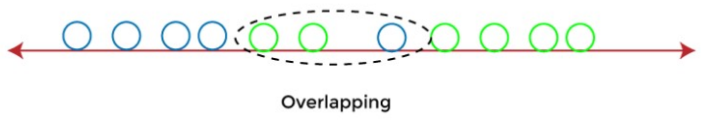
$$\begin{aligned} E[\sum_{i=1}^{n}(x_i - \hat{\mu})^2] &= E[\sum_{i=1}^{n}(x_i)^2] - nE[\hat{\mu}^2] \\ &= nE[(x_i)^2] - \frac{1}{n}E[(\sum_{i=1}^{n} x_i)^2] \\ &= n(var(x_i) + (E[x_i])^2) - \frac{1}{n}E[(\sum_{i=1}^{n} x_i)^2] \\ &= n\sigma^2 + \frac{1}{n}(nE[x_i])^2) - \frac{1}{n}E[(\sum_{i=1}^{n} x_i)^2] \\ &= n\sigma^2 - \frac{1}{n}[E[(\sum_{i=1}^{n} x_i)^2] - (E[\sum_{i=1}^{n} x_i])^2] \\ &= n\sigma^2 - \frac{1}{n}var(\sum_{i=1}^{n} x_i) \\ &= n\sigma^2 - \frac{1}{n}n\sigma^2 \\ &= (n-1)\sigma^2 \end{aligned}$$

Proof 2:

$$\begin{aligned} E[\sigma\hat{_{ML}}^2] &= E[\frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu})^2] \\ &= \frac{1}{n}\sum_{i=1}^{n} E[x_i^2 - 2x_i\hat{\mu} + \hat{\mu}^2] \\ &= \frac{1}{n}\sum_{i=1}^{n} E[x_i^2] - 2\hat{\mu}E[\sum_{i=1}^{n} x_i] + \sum_{i=1}^{n} E[\hat{\mu}^2] \\ &= \frac{1}{n}[n(\sigma^2 + \mu^2) - 2nE[\hat{\mu}^2] + nE[\hat{\mu}^2]] \\ &= \frac{1}{n}[n\sigma^2 + n\mu^2 - nE[\hat{\mu}^2]] \\ &= \frac{1}{n}[n\sigma^2 + n\mu^2 - n(frac\sigma^2 n + \mu^2)] \\ &= \frac{n-1}{n}\sigma^2 \\ & Therefore \\ \text{unbiased: } \hat{\sigma}^2 &= \frac{n}{n-1}\sigma\hat{_{ML}}^2 \\ &= \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \hat{\mu})^2 \end{aligned}$$

**23. What is Linear Discriminant Analysis ( LDA )?**

**Ans:** Linear Discriminant Analysis (LDA) is one of the commonly used dimensionality reduction techniques in machine learning to solve more than two-class classification problems. It is also known as Normal Discriminant Analysis (NDA) or Discriminant Function Analysis (DFA).
Although the logistic regression algorithm is limited to only two-class, linear Discriminant analysis is applicable for more than two classes of classification problems.
Linear Discriminant analysis is one of the most popular dimensionality reduction techniques used for supervised classification problems in machine learning. It is also considered a pre-processing step for modeling differences in ML and applications of pattern classification. Whenever there is a requirement to separate two or more classes having multiple features efficiently, the Linear Discriminant Analysis model is considered the most common technique to solve such classification problems. For e.g., if we have two classes with multiple features and need to separate them efficiently. When we classify them using a single feature, then it may show overlapping.



To overcome the overlapping issue in the classification process, we must increase the number of features regularly.

**24. Describe the isodata algorithm to categorize points in different clusters.**

**Ans:** The first automatic threshold selecting method was probably by isodata algorithm, which was originally proposed by Ridler and Calvard (1978). In the algorithm, a threshold is first guessed (in most cases it is selected by the average intensity value of the image) and then used to segment the histogram into two classes, i.e. *A* and *B*. The average intensity values, $m_A$ and $m_B$, for both classes are calculated, and the new threshold is then determined as the average of $m_A$ and $m_B$. The new threshold is updated iteratively by the new average intensity values until convergence is achieved.
Alternatively, the objective function method might be used. Here, the histogram is preliminarily normalized and regarded as probability distributions using equation below:

$$h(j) = H(j) \; / \; \sum_{i=I}^{L} (i)$$

The distribution is classified into two groups (i.e. objects and background) using a threshold, which is an intensity value iteratively selected from the minimum to the maximum of the intensity values. The optimal threshold is determined as the one that maximizes the objective function, and is based on the interaction of the two classes with regard to evaluating the success of the thresholds. Two kinds of objective functions are mostly used: variance-based and entropy-based. The optimal threshold *t* is selected to maximize the between-class variance, which can be calculated by

$$\sigma = \frac{[\mu(L)\omega(t) - \mu(t)]^2}{\omega(t)[1 - \omega(t)]}$$

where ω and μ are the zero-th- and first-order cumulatives of the probability distribution, respectively.
In the entropy-based objective function, the optimal threshold is selected as the intensity value at which the sum entropies of the two classes are maximized.