# Assignment 1 - Machine Learning

## Topic: Bayesian Machine Learning

Members:
1) Shantanu Ambekar (2021A7PS2540P)
2) Dhruv Shrimali (2021A7PS0008P)
3) Rudra Jewalikar (2021A7PS0450P)

# Naive Bayes Email Classifier

## 1. Introduction

Naive Bayes classifier is implemented for distinguishing between spam and ham (non-spam) emails. The algorithm uses a probabilistic approach to calculate the likelihood of an email being spam or ham based on the occurrence of words in the training dataset. The classifier assumes that the presence of each word in the email is independent of the presence of other words, which is a key assumption of the Naive Bayes algorithm.

## 2. Implementation Details

### 2.1 Data Preprocessing:

- The code reads email data from a CSV file where each line consists of an email and its corresponding label (spam or ham).
- The tokenize function is responsible for breaking down the email text into individual words, which are often referred to as tokens. Tokenization is a crucial step in natural language processing (NLP) tasks as it converts raw text into manageable units for analysis. In this implementation, 'istringstream' is used to split the email text based on spaces, creating a vector of individual words.\

Data source: https://www.kaggle.com/code/ayhampar/spam-ham-dataset
(Note: Data has been preprocessed to remove punctuations and numbers for simplicity.)

### 2.2 Training:

- During the training phase, the code reads emails and their corresponding labels from the provided dataset. For each email, it tokenizes the text and increments word counts based on the email's label (spam or ham). Separate word count maps (spamWordCount and hamWordCount) are maintained for spam and ham emails, allowing the classifier to learn the frequency of words in each category. It maintains separate counters as well for total counts of spam and ham emails (totalSpamEmails, totalHamEmails) and words (totalSpamWords, totalHamWords).

### 2.3 Classification:

- The classify function calculates the probabilities of an email being spam or ham given its words. It uses the Naive Bayes formula, incorporating Laplace (add-one) smoothing to handle words not present in the training set.
- The classifier then compares spamProbability and hamProbability to determine the class with higher probability, classifying the email accordingly.

## 3. Usage and Results

The code demonstrates the usage of the NaiveBayesClassifier class by training the classifier on a dataset (spam_ham_dataset.csv) and classifying three test emails.

- Test Emails:
    a. "Buy cheap viagra now!"
    b. "Hi, how are you?"
    c. "Hi how Dhruv you!"
- Classification Results:
    a. Email 1: Classified as spam.
    b. Email 2: Classified as ham.
    c. Email 3: Classified as ham.

## 4. Conclusion

The Naive Bayes Classifier provides a simple yet effective method for email classification. By leveraging the probabilities of word occurrences in spam and ham emails, the classifier can make reasonably accurate predictions. However, it is important to note that the Naive Bayes algorithm assumes independence between words, which might not always hold in real-world scenarios. Despite its simplicity, Naive Bayes classifiers often perform surprisingly well in practice, especially for text classification tasks like spam detection.

---

# Bayesian Belief Network

## 1. Algorithm Description

### 1.1 Bayesian Belief Network Construction

The implemented algorithm constructs a Bayesian Network based on given data. The following steps were performed for network construction:

- Data Reading: The algorithm begins by reading the car data from a CSV file. Each row of the dataset contains information about a car's attributes such as price rating (B), maintenance rating (M), number of doors (D), number of persons (P), luggage-boot rating (LB), safety rating (S), and overall class rating (OC). The data is processed to calculate the frequencies of different attribute values.
- Node Initialization: Nodes representing different attributes (B, M, D, P, LB, S, OC) are initialized. Each node has a name and a list of possible values.
- Conditional Probability Table (CPT) Construction: For each node, a Conditional Probability Table is constructed based on the frequencies of attribute values in the

dataset. The CPT represents the probabilities of the node taking on each possible value given the values of its parent nodes (if any).

- Edge Establishment: Edges are established between parent nodes (B, M, D, P, LB, S) and the child node (OC) to represent dependencies among variables. For example, an edge from B to OC indicates that the overall class rating (OC) depends on the price rating (B). In this case, OC was a child node, while the remaining nodes (B, M, D, P, LB, S) were parent nodes.

### 1.2 Inference Algorithm: Variable Elimination

The algorithm performs probabilistic inference using the variable elimination method. When querying the network with specific observed evidence (e.g., B=1, D=2), the algorithm:

- Observing Evidence: The algorithm takes observed evidence, such as B=1 and D=2, into account. These observed values restrict the possible states of the corresponding nodes in the network.
- Variable Elimination: The algorithm eliminates variables based on the observed evidence. For instance, if D=2 is observed, all rows in the CPT of D where D is not 2 are removed. This process narrows down the possible states of each variable in the network.
- Calculation of Probabilities: After eliminating variables based on the observed evidence, the algorithm calculates the probabilities for the query variable OC. For OC, a special case is handled where the probabilities are calculated based on the remaining rows in the CPT of OC. The probabilities represent the likelihood of each possible overall class rating given the observed evidence.
- Result Interpretation: The algorithm provides the probabilities for different values of OC, allowing users to understand the likelihood of different overall class ratings based on the given evidence (B=1, D=2 in this case).

## 2. Data Used

The Bayesian Network was constructed using a dataset extracted from a CSV file named bbn_car_data.csv. The dataset contained features related to car specifications.
The dataset was sourced from UC Irvine's machine learning repository. The Car Evaluation Data Set is a popular dataset for building a Bayesian belief network to evaluate car safety based on various attributes. The dataset was trimmed to include cells with only numeric values. Each row of the trimmed dataset represents a unique car entry with the following attributes:

- B: Buying price rating (values: 1, 2, 3, 4)
- M: Maintenance rating (values: 1, 2, 3, 4)
- D: Number of doors (values: 1, 2, 3)
- P: Number of persons (values: 2, 4)
- LB: Luggage-boot rating (values: 1, 2, 3)
- S: Safety rating (values: 1, 2, 3)
- OC: Overall class rating (values: 1, 2, 3, 4)

## 3. Results Obtained

For the given observed evidence (B=1, D=2), the algorithm performed inference and provided the following probabilities for the query variable OC (Overall Class Rating):

- Probability of OC=1: 0.694444
- Probability of OC=2: 0.180556
- Probability of OC=3: 0.083333
- Probability of OC=4: 0.041667

These probabilities represent the likelihood of different overall class ratings given the observed evidence in the Bayesian Network.
The Bayesian Network can be used to make inferences about the overall class rating of a car given information about its other attributes. For example, if we know that a car has a buying price rating of 1 and a number of doors of 2, the Bayesian Network would predict that there is a 69.44% chance that the car has an overall class rating of 1.

---

# Bayesian Linear Regression

## 1. Introduction

### 1.1 Problem Statement

The problem is to find the best-fitting linear model for a given dataset of height-weight pairs. The challenge lies in estimating the parameters (slope and intercept) of the linear equation that describes the relationship between height and weight.
The objective is to perform Bayesian Linear Regression using the Metropolis-Hastings algorithm. Bayesian Linear Regression is a probabilistic approach to modeling the relationship between variables. By incorporating prior knowledge and iteratively updating estimates based on observed data, Bayesian methods provide a principled way to handle uncertainty in regression tasks.

### 1.2 Metropolis-Hastings Algorithm

The Metropolis-Hastings algorithm is a Markov Chain Monte Carlo (MCMC) method used for obtaining a sequence of random samples from a probability distribution. In this context, the algorithm iteratively explores the parameter space, generating samples according to the posterior distribution given the observed data. The algorithm accepts or rejects proposed parameter values based on the likelihood of the data given the parameters and a prior distribution.

### 1.3 Bayesian Linear Regression

Traditional linear regression provides point estimates for the model parameters. In Bayesian Linear Regression, we treat the parameters as random variables with probability distributions (priors). The goal is to estimate the posterior distribution of the parameters given the observed data. This approach allows us to quantify uncertainty and make probabilistic predictions. The posterior distribution is proportional to the likelihood of the data given the parameters and the prior distribution of the parameters.
In practice, evaluating the posterior distribution for the model parameters is intractable for continuous variables, so we use sampling methods to draw samples from the posterior in order to approximate the posterior. The technique of drawing random samples from a distribution to approximate the distribution is done by Metropolis-Hastings Algorithm.

$$P(\beta|y, X) = \frac{P(y|\beta, X) * P(\beta|X)}{P(y|X)}$$

*This formula represents that the posterior probability of the model parameters is conditional upon the training inputs and outputs:*
*Here Beta represents the parameters, y the output and x the input*.

## 2. Methodology

### 2.1 Data Loading

The code reads data from a CSV file ("SOCR-HeightWeight.csv"). Each line in the file represents a data point containing a person's height and weight. These data points serve as the basis for estimating the linear regression parameters.
Only 1 feature has been used in the dataset for ease of implementation.
Source of dataset: Heights and Weights Dataset

(Note: Data has been slightly preprocessed and converted from an inch-pound system to a metric system.)

### 2.2 Likelihood Function

The likelihood function computes the probability of observing the given data points given specific parameter values (slope and intercept). It quantifies how well the linear model fits the observed data. In this code, the negative squared error is used in the likelihood calculation, reflecting the difference between predicted and actual values.

### 2.3 Metropolis-Hastings Iterations

The Metropolis-Hastings algorithm iterates a fixed number of times. In each iteration:

- Proposal Step: It proposes new values for the slope and intercept with a mean of the current parameter values. This introduces randomness and allows exploration of the parameter space.
- Likelihood Evaluation: The likelihood of the data given the current and proposed parameters is computed. It measures how well the data fits the model for both sets of parameters.
- Acceptance Step: The algorithm decides whether to accept the proposed parameters based on the likelihood ratio. If the proposed parameters yield a higher likelihood, they are accepted; otherwise, there's a chance of acceptance based on the likelihood ratio. This step ensures that the algorithm converges toward the regions of high likelihood, i.e., the most probable parameter values given the data.

## 3. Results

The algorithm outputs the estimated slope and intercept after each iteration. At the end of the iterations, it provides the weight prediction for a given test height (165 units) using the estimated parameters.

## 4. Conclusion

The Bayesian Linear Regression using the Metropolis-Hastings algorithm offers a probabilistic approach to estimating the parameters of a linear model. By exploring the posterior distribution iteratively, the algorithm provides valuable insights into the uncertainty associated with the parameter estimates. The final estimated parameters can be used for making predictions, as demonstrated by the weight prediction for a specific height in this implementation.

---

# Expectation Maximization Clustering Algorithm

## 1. Introduction

The Gaussian Mixture Model (GMM) is a probabilistic model used for clustering tasks. It assumes that the data points are generated from a mixture of several Gaussian distributions with unknown parameters. The GMM algorithm aims to estimate these parameters and assign each data point to one of the clusters represented by the Gaussian distributions.

### 1.1 Implementation Overview

The implemented GMM algorithm processes a dataset consisting of points with two features: height and weight. It leverages the Expectation-Maximization (EM) algorithm, a statistical technique, to estimate the parameters of multiple Gaussian distributions. These parameters include cluster means, covariances, and weights. The ultimate objective is to cluster the data points into distinct groups based on these learned parameters.

## 2. Implementation Details

### 2.1 Data Processing

The input data, read from a CSV file, comprises points in a two-dimensional space, with height and weight as the features. Each data point is represented by a structure, making it easier to manipulate and process individual data elements.
Data source: https://cdn.analyticsvidhya.com/wp-content/uploads/2019/10/Clustering_gmm.csv

### 2.2 Initialization

To initiate the clustering process, the algorithm randomly initializes clusters. This implementation sets the number of clusters (K) to 3. Each cluster is defined by its weight (representing the probability of a data point belonging to that cluster), mean (representing the centre of the cluster), and covariance (indicating the spread of the data points around the mean).
Data points are chosen randomly for initialising mean, equal weight is assigned to each cluster and covariance is taken as 1. Proper initialization of these parameters is crucial for the convergence and accuracy of the algorithm.

### 2.3 Expectation-Maximization (EM) Algorithm

The EM algorithm consists of two main steps: the E-step (Expectation step) and the M-step (Maximization step).

- E-step: In this step, the algorithm calculates the likelihood of each data point belonging to each cluster, using the Gaussian probability density function. These likelihoods are normalized to obtain responsibilities, which represent the probability of a data point belonging to a specific cluster. This involves leveraging Bayes' theorem. The E-step assigns a level of confidence to each data point's association with each cluster.

- Gaussian probability density function for 2-dimentional data:

$$f(x \mid \mu, \Sigma) = \frac{1}{\sqrt{2\pi|\Sigma|}} \exp\left[ -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^t \Sigma^{-1} (\boldsymbol{x}-\boldsymbol{\mu}) \right]$$

-

x is the input vector, μ is the 2D mean vector, and Σ is the 2×2 covariance matrix

- M-step: In the M-step, the algorithm updates the cluster parameters based on the calculated responsibilities. The mean and covariance of each cluster are recalculated, incorporating the weighted contributions of data points in the cluster. The cluster weights are also updated, reflecting the proportion of data points assigned to each cluster.

### 2.4 Convergence

To determine if the algorithm has converged, the change in log-likelihood between consecutive iterations is monitored. The log-likelihood quantifies the likelihood of the observed data given the estimated parameters. If the change in log-likelihood falls below a predefined threshold ($\epsilon$), the algorithm is considered to have converged. Convergence ensures that the algorithm has reached a stable state, and further iterations are unlikely to significantly improve the clustering result. If convergence is not achieved, the algorithm halts after a set number of iterations.

## 3. Results

The algorithm assigns each data point to one of the clusters based on the calculated responsibilities. The output includes the cluster assignments and the details of each cluster, including cluster mean, covariance, weight and size.

## 4. Conclusion

The GMM clustering algorithm successfully clusters the given dataset into three distinct clusters based on the height and weight features. The algorithm can identify underlying patterns in multidimensional data and group similar data points together.

---