**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI**

# A Report

## On

# Bayesian Machine Learning


## Submitted To:

Prof. Navneet Goyal
Professor,
Department of Computer Science and Information Systems,
BITS Pilani

## Submitted By:

Group 3
Arjoo Kumari (2020B3A70770P)
Adarsh Goel (2020B3A70821P)

# 1. Naive Bayes Classifier

## Theoretical Background

The Naive Bayes Classifier is a probabilistic machine learning algorithm that is based on Bayes' theorem. It is particularly well-suited for classification tasks, where the goal is to assign a label or category to a given input based on its features. The algorithm is considered "naive" because it makes a strong assumption of feature independence, meaning that the presence or absence of a particular feature is assumed to be unrelated to the presence or absence of any other feature given the class label.

Bayes' theorem is expressed as:

$$P(A|B) = P(B|A) \cdot P(A) / P(B)$$

In the context of classification, we can express this as the probability of a class (C) given certain features (X):

$$P(C|X) = P(X|C) \cdot P(C) / P(X)$$

The "naive" assumption here is that the features are conditionally independent given the class label, allowing us to simplify the expression:

$$P(C|X) \propto P(C) \cdot \Pi^{n}_{i=1} P(X_i|C)$$

Here, P(C|X) is the posterior probability i.e. the probability of a class C given the observed features X, P(C) is the prior probability of class C which means that it is the probability of a particular class occurring without considering any features, $P(X_i|C)$ is the likelihood of feature $X_i$ given class C, and P(X) is the normalising constant. The ‘naive’ assumption of independence between features allows us to express the likelihood of observing the given set of features given a particular class as the product of the individual feature probabilities.

## Implementation of Algorithm in C++

To implement the Naive Bayes Classifier in C++ we create a structure ‘DataPoint’ that holds a vector of features and a label. Each data point in the training set consists of a set of features and the corresponding class label. For the test data the label of the datapoint is taken as an empty string.

We then created a ‘NaiveBayesClassifier’ class that contains private data members to store the likelihoods, class probabilities, unique classes, and the total number of examples. Two public methods are defined within the class to train and predict using the classifier algorithm. The ‘train’ method calculates the counts of each class and each feature value for a specific label in the

training data and then computes the feature likelihoods and class probabilities based on these counts. The predict method takes a new data point and calculates the posterior probability for each class using the trained model. It applies Laplace smoothing for unseen features to handle cases where a feature value has not been observed during training.

## Dataset

The dataset that has been used to check the implementation of Naive Bayes Classifier is the credit dataset used in the class discussion. This dataset was used to ensure that the values of probability we were getting, matched the theoretical probability values.

## Results

Following is the training data:

```
vector<DataPoint> trainingData =
{
    {{"<=30", "high", "No", "fair"}, "No"},
    {{"<=30", "high", "No", "excellent"}, "No"},
    {{"31..40", "high", "No", "fair"}, "Yes"},
    {{">40", "medium", "No", "fair"}, "Yes"},
    {{">40", "low", "Yes", "fair"}, "Yes"},
    {{">40", "low", "Yes", "excellent"}, "No"},
    {{"31..40", "low", "Yes", "excellent"}, "Yes"},
    {{"<=30", "medium", "No", "fair"}, "No"},
    {{"<=30", "low", "Yes", "fair"}, "Yes"},
    {{">40", "medium", "Yes", "fair"}, "Yes"},
    {{"<=30", "medium", "Yes", "excellent"}, "Yes"},
    {{"31..40", "medium", "No", "excellent"}, "Yes"},
    {{"31..40", "high", "Yes", "fair"}, "Yes"},
    {{">40", "medium", "No", "excellent"}, "No"}

};
```

Following is the query on the training data:

```
DataPoint testPoint = {{"<=30", "medium", "Yes", "fair"}, ""};
```

Following is the result from the query:

```
PS C:\Users\ADARSH GOEL\OneDrive\Desktop\ML Assignment> cd "c:\Users\ADARSH GOEL\OneDrive\Desktop\ML Assignment\" ; if ($?) { g++ NaiveBayes.cpp -o NaiveBayes } ; if ($?) { .\NaiveBayes }
Predicted class: Yes
Probability : 0.028219
```

## Advantages and Disadvantages of Naive Bayes Classifier

### Advantages:

1. **Simplicity and Ease of Implementation:** The algorithm is easy to understand and implement, making it suitable for quick prototyping and applications where a complex model may not be necessary.
2. **Efficiency in Training:** The algorithm is computationally efficient, especially during the training phase. The calculations involve counting occurrences of features and class labels, which is a relatively fast process which makes it well-suited for large datasets.
3. **Good Performance on High-Dimensional Data:** Naive Bayes tends to perform well, particularly in situations where the number of features is high. It is commonly used in text classification tasks, such as spam filtering or sentiment analysis, where the dataset is represented by a large number of words or terms.
4. **Handling of Missing Data:** The algorithm can gracefully handle missing data. Since it calculates probabilities independently for each feature, missing values for a particular feature do not disrupt the entire calculation.
5. **Interpretability:** The predictions made by Naive Bayes are easily interpretable. The model provides insights into the importance of different features in making a classification decision.

### Disadvantages:

1. **Assumption of Feature Independence:** The most significant limitation of Naive Bayes is its assumption of feature independence. In reality, features may be correlated, and this assumption may not hold true for all datasets. The model may not perform well when the features are highly dependent on each other.
2. **Sensitivity to Irrelevant Features:** It is sensitive to the inclusion of irrelevant features in the dataset. If irrelevant features are present, they can affect the model's performance by introducing noise. Feature selection or dimensionality reduction techniques may be required to mitigate this issue.
3. **Handling of Continuous Data:** The algorithm assumes that features are categorical or discrete. When dealing with continuous data, discretization methods are applied, which may result in information loss and affect the model's accuracy.
4. **Lack of Model Complexity:** While simplicity is an advantage, it can also be a limitation. Naive Bayes may not capture complex relationships within the data as well as more sophisticated models. In situations where the underlying patterns are intricate, other algorithms like decision trees or ensemble methods might be more suitable.
5. **Impact of Imbalanced Datasets:** Naive Bayes can be sensitive to imbalanced datasets, where one class significantly outnumbers the other. The model may be biased towards the majority class, leading to suboptimal performance on the minority class.

## 2. Bayesian Belief Networks

### Theoretical Background

Bayesian Belief Networks, also known as Bayesian Networks, are probabilistic graphical models that represent the probabilistic relationships among a set of random variables and their conditional dependencies via a directed acyclic graph. They are based on a combination of Bayes' theorem and graph theory. BBNs are widely used for reasoning under uncertainty and modelling complex systems. In this report we will explore the details of the Bayesian belief network algorithm with respect to the example of an alarm system that considers the events of an earthquake and burglary.

The graph structure of a BBN is a directed acyclic graph (DAG). Directed means that the dependency is only in one direction and acyclic means that there are no cycles or loops. The absence of cycles ensures that the joint probability distribution can be factorised into a product of conditional probabilities, simplifying calculations.

In this graph, the nodes represent random variables, and directed edges between nodes represent probabilistic dependencies. Each node is associated with a probability distribution given its parents in the graph. The absence of a direct edge between two nodes indicates conditional independence given the values of their common ancestors. If two nodes are not connected, they are conditionally independent.

Let’s consider the example. The dependencies of the graph can be represented as follows:

**`Burglary -----> Alarm <----- Earthquake`**

Alarm represents the status of the alarm (on or off), Earthquake represents the occurrence of an earthquake and Burglary represents the occurrence of a burglary. Looking at the dependency graph we can say that Burglary and Earthquake are independent given the state of Alarm.

As mentioned earlier, each node has a conditional probability table (CPT) that specifies the probability distribution of the variable given its parents. For the Alarm node, the CPT might look like:

| **Burglary** | **Earthquake** | **P(Alarm = ON)** |
|---|---|---|
| true | true | 0.95 |
| true | false | 0.93 |
| false | true | 0.29 |
| false | false | 0.001 |

BBNs allow for efficient probabilistic inference. Given observations for certain variables, the network can compute the probabilities of other variables. For example, if we observe the Alarm is ON, we can infer the probabilities of Earthquake and Burglary using the Bayes' theorem. The formula for probability of Earthquake given Alarm is ON would be:

**P(Earthquake | Alarm) = (P(Alarm | Earthquake) * P(Earthquake)) / P(Alarm = on)**

BBNs also support dynamic updating of probabilities. If new evidence becomes available, the probabilities of other variables can be updated accordingly using Bayes' theorem.

## Implementation of Algorithm in C++

For the implementation of the Bayesian Belief Network in C++, we created a struct 'Node' that represents a random variable in the Bayesian Network. It has a name, a list of parents, that is the nodes that influence it, and a map of probabilities. To introduce the dependencies between nodes we can add the parent node to the list of parents of the child node. Additionally the map of probabilities can be populated with the conditional probability table as discussed in the previous section. Using this structure any DAG representation of the Bayesian Belief Network can be constructed. However, the task of construction of the graph is left to the users, who can modify it according to their needs.

Furthermore, two functions variableElimination and generalQuery are defined to introduce the functionality inherent to the BBNs.

The function variableElimination computes the probability of a set of evidence variables based on the Bayesian Network using the variable elimination method. The function initialises a variable probability to 1.0. The function iterates through each node in the Bayesian Network, represented by the network map. If the current node has no parents (i.e., it is a root node), the function retrieves the evidence value for that node and multiplies it with the current probability.

If the current node has parents, the function constructs an event string by concatenating the values of evidence variables for each parent. The event string is used to look up the conditional probability in the node's probability map. The probability is then updated based on whether the evidence for the current node is "true" or "false." The final probability is the product of all the probabilities calculated for each node in the Bayesian Network.

The function generalQuery handles general queries in the Bayesian Network, where some variables are unknown ("-"). It iterates through all possible combinations of unknowns, updating the evidence and computing probabilities using the Bayes theorem as discussed above.

## Dataset

For this we have used the Alarm example discussed above. The values of the conditional probability table have been populated using random values.

Following is the dataset we have considered:

```
// Creating nodes for the Bayesian Network
Node A = {"Buglary"};
Node B = {"Earthquake"};
Node C = {"Alarm"};
Node D = {"David Calls"};
Node E = {"Sophia Calls"};

// Define relationships between nodes
C.parents.push_back(&A);
C.parents.push_back(&B);
D.parents.push_back(&C);
E.parents.push_back(&C);

// Define probabilities (conditional probability distributions) for each node
A.probabilities["false"] = 0.001;
A.probabilities["false"] = 0.999;

B.probabilities["true"]= 0.002;
B.probabilities["false"] = 0.998;

// Conditional Probability Distribution for C given A and B
C.probabilities["true,true"] = 0.95;
C.probabilities["true,false"] = 0.94;
C.probabilities["false,true"] = 0.29;
C.probabilities["false,false"] = 0.001;

// Conditional Probability Distribution for D given C
D.probabilities["true"] = 0.95;
D.probabilities["false"] = 0.05;

// Conditional Probability Distribution for E given C
E.probabilities["true"] = 0.8;
E.probabilities["false"] = 0.01;
```

## Results

Following is the query on the dataset:

```
// Perform inference - Querying variable D given evidence A=false, B=true
map<string, string> query = {{"Buglary", "false"}, {"Earthquake", "false"}, {"Alarm", "true"}, {"David Calls", "true"}, {"Sophia Calls", "-"}};
```

Following is the result on the query:

```
PS C:\Users\ADARSH GOEL\OneDrive\Desktop\ML Assignment> cd "c:\Users\ADARSH GOEL\OneDrive\Desktop\ML Assignment\" ; if ($?) { g++ BayesianBeliefNetworks.cpp -o BayesianBeliefNetworks } ; if ($?) { .\Ba
yesianBeliefNetworks }
Probability of the query: 0.000947152
```

The result is correct as:

0.000947152 = 0.999 * 0.998 * 0.001 * 0.95

## Advantages and Disadvantages of Bayesian Belief Networks

### Advantages:

1. **Probabilistic Representation:** BBNs offer a probabilistic representation of uncertainty, allowing for the modelling of real-world situations where outcomes are uncertain or not fully known.
2. **Modularity:** The modular structure of BBNs allows for easy updation and modification of the model as new information becomes available which makes them adaptable to changing circumstances.
3. **Interpretability:** The intuitive and graphical representation of probabilistic dependencies provided by BBNs makes them easy to understand and interpret.
4. **Efficient Inference:** BBNs enable efficient probabilistic inference, allowing users to calculate the probability of specific events or states based on observed evidence. This is crucial for decision-making under uncertainty.
5. **Handling Incomplete Information:** BBNs can handle incomplete or missing information by incorporating available evidence and updating probabilities accordingly. This is valuable in scenarios where not all variables are observed.
6. **Causal Relationships:** BBNs naturally capture causal relationships between variables, providing insights into cause-and-effect scenarios. This is beneficial for understanding the impact of interventions or changes.

### Disadvantages:

1. **Computational Complexity:** Inference in large and complex BBNs can be computationally demanding. As the number of nodes and edges increases, the complexity of calculations grows which negatively impacts the efficiency of the model.
2. **Learning Structure and Parameters:** Determining the appropriate structure and parameters of a BBN from data can be challenging. Learning the dependencies and conditional probabilities requires careful consideration and may involve computational challenges.
3. **Assumption of Conditional Independence:** BBNs assume conditional independence given the parents of a node. In some real-world scenarios, this assumption may not hold, leading to a potential mismatch between the model and the true underlying relationships.

4. **Selection of Prior Probabilities:** The choice of prior probabilities can influence the model's outcomes. Subjective decisions in selecting priors may introduce bias, and obtaining accurate prior information may be challenging in some cases.

5. **Sensitivity to Model Structure:** BBN results can be sensitive to the chosen structure of the network. Small changes in the model structure or prior probabilities can lead to different outcomes, requiring careful consideration during model development.

6. **Subjectivity in Elicitation:** The process of eliciting expert opinions to define prior probabilities can be subjective and may introduce biases. The model's reliability depends on the quality and accuracy of the information provided by experts.

# 3. Bayesian Linear Regression

## Theoretical Background

Linear regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables by fitting a linear equation to observed data.

The linear relationship between the independent variables X and the dependent variable y can be represented as follows:

$y = X \cdot w + \epsilon$ where w represents the weights (parameters) to be estimated, and $\epsilon$ is the error term.

Bayesian Linear Regression (BLR) extends this framework by incorporating Bayesian principles, introducing a probabilistic perspective, and providing a more flexible and robust approach to modelling uncertainty. Bayesian Linear Regression treats model parameters as random variables with associated probability distributions. This allows us to quantify uncertainty in our parameter estimates. We specify a prior distribution p(w) over the model parameters before observing any data. This encodes our beliefs or assumptions about the parameters. In our implementation of the algorithm we have used a multivariate Gaussian prior which can be represented as

$$\mathbf{p(w) \sim N(0 , \Sigma_0)}$$

where $\Sigma_0$ is a covariance matrix.

The likelihood function p(y|X, w) represents the probability of observing the data given the model parameters. For Gaussian-distributed errors, this likelihood is often expressed as:

$$\mathbf{p(y|X, w) \sim N(X \cdot w, \sigma^2 I)}$$

where $\sigma^2$ is the variance of the errors.

Using Bayes' theorem, the posterior distribution is calculated from the above information on priors and likelihood. It can be represented as:

$$p(w|X, y) \propto p(y|X, w) \cdot p(w)$$

The posterior combines the information from the prior and the likelihood, providing an updated probability distribution for the model parameters given the observed data. Thus, we start our regression method with an estimate which is the prior distribution value. As we begin to include additional data points, we continue updating the prior using the formula given above and the accuracy of our model keeps improving gradually.

This is the training step of the algorithm. Once the model has been trained the predictions for new and unseen data can be made by using the posterior predictive distribution $p(\hat{\mathbf{Y}}|X, y)$, which is given by:

$$p(\hat{Y}|X, y) = \int p(\hat{Y}|X, w) \cdot p(w|X, y) \cdot dw$$

## Implementation of Algorithm in C++

For the implementation of this algorithm BayesianLinearRegression class was defined whose variables MatrixXd X represents the input features for training, VectorXd y: represents the target values for training and alpha is the precision parameter for the prior. The constructor of this class initialises the value of these variables based on the input. It also initialises the mean vector and covariance matrix of the weight distribution to represent the Gaussian prior with alpha precision.

The method fit of this class is used to train the model by updating the weight distribution using the theoretical background discussed in the previous section. The method predict is used to predict the mean and variance of the target variable.

## Dataset

For the Bayesian Linear Regression we took 500 datapoints from California Housing Prices dataset which is available at https://www.kaggle.com/code/shtrausslearning/bayesian-regression-house-price-prediction/input?select=housing.csv. This dataset had 8 features namely:  Longitude, Latitude, Housing median Age, total rooms, total bedroom, population, households and median income. The dependent variable is median house price.

## Result

Following was the query on the model:

```
// Make predictions
VectorXd new_data_point(8);
new_data_point << -122.23,37.88,41,880,129,322,126,8.3252; // Replace with your new data point
```

Following is the result:

```
PS C:\Users\ADARSH GOEL\OneDrive\Desktop\ML Assignment> cd "c:\Users\ADARSH GOEL\OneDrive\Desktop\ML Assignment\" ; if ($?) { g++ BayesianLinearRegression.cpp -o Bay
esianLinearRegression } ; if ($?) { .\BayesianLinearRegression }
Predicted mean: 407002
Predicted variance: 0.142455
```

The mean value from prediction is 407002. However, the correct value from dataset is around 450000. So, Bayesian Linear Regression is performing good.

## Advantages and Disadvantages of Bayesian Linear Regression

**Advantages:**

1. **Incorporates Uncertainty:** Bayesian Linear Regression provides a full probability distribution over the model parameters. Instead of providing point estimates, it offers a range of likely parameter values, reflecting the uncertainty inherent in statistical modelling.

2. **Regularisation:** The Bayesian framework naturally includes regularisation through the choice of prior distributions for model parameters. This helps prevent overfitting by penalising overly complex models and automatically finds a balance between fitting the data and staying within the bounds set by the prior.

3. **Flexibility in Prior Specification:** Users can incorporate prior knowledge into the model by selecting prior distributions for the parameters. This flexibility allows the integration of domain expertise, making Bayesian Linear Regression suitable for scenarios where prior information is available.

4. **Sequential Learning:** Bayesian models can be updated sequentially as new data becomes available. This sequential learning capability is beneficial for adapting the model to changing environments or incorporating additional observations without retraining on the entire dataset.

5. **Outlier Robustness:** Bayesian methods tend to be less sensitive to outliers because they consider the entire distribution of parameter values. Outliers have less influence on the estimates, leading to more robust predictions in the presence of extreme data points.

6. **Avoids Overfitting:** The Bayesian approach naturally avoids overfitting by incorporating uncertainty. The model penalises overly complex parameter configurations, preventing it from fitting noise in the data.

**Disadvantages:**

1. **Computational Complexity:** Bayesian methods, including Bayesian Linear Regression, can be computationally expensive due to the repeated calculations for the value of posteriors.

2. **Choice of Priors:** The choice of prior distributions can impact the model's performance. Selecting appropriate priors requires a good understanding of the problem domain, and the results can be sensitive to these choices.

3. **Interpretability Challenges:** Interpreting results from Bayesian models can be more challenging than from frequentist models. Instead of a single point estimate, one must consider the entire distribution, making the interpretation more nuanced.

4. **Assumption of Gaussian Noise:** Like frequentist linear regression, Bayesian Linear Regression assumes that the noise in the data follows a Gaussian distribution. Deviations from this assumption may affect the model's performance.

5. **Limited for Non-linear Patterns:** While Bayesian Linear Regression can capture linear relationships between variables, it may struggle to model complex non-linear patterns.

# 4. Expectation Maximisation Clustering Algorithm

### Theoretical Background:

The Expectation Maximisation Algorithm is the soft assignment generalisation of the K-means clustering algorithm. The K-means clustering algorithm works on the assumption that each point can belong to only one class. In the EM algorithm, the addition of soft assignment means that each point can now belong to multiple classes and we are concerned about the probability of the point belonging to the different classes. For this we assume that the data distribution is obtained from a mixture of Gaussians.

**Gaussian Mixture Models (GMMs):** GMMs are probabilistic models that represent a mixture of multiple Gaussian distributions. In the context of clustering, each Gaussian component in the mixture represents a cluster, and data points are assumed to be generated by one of these components. The probability density function of a Gaussian mixture model is given by the weighted sum of the individual Gaussian distributions:

$$\mathbf{P(x) = \sum_{k=1}^{K} \pi_k \cdot N(x|\mu_k, \Sigma_k)}$$

Here, $\pi_k$ is the weight of the k-th Gaussian component, $\mu_k$ is its mean vector, and $\Sigma_k$ is its covariance matrix.

### Expectation-Maximization (EM) Algorithm:

The EM algorithm is an iterative optimization algorithm used to find maximum likelihood estimates (MLE) of parameters in the presence of latent (unobserved) variables. For GMMs, the latent variables are the cluster assignments of data points. Now, we will have a look at the steps of the algorithm in detail.

**Initialization:** The first step is to initialise the following parameters of the GMM randomly or using a heuristic method.

- $\boldsymbol{\mu_k}$**:** Mean vectors of the Gaussian components.
- $\boldsymbol{\Sigma_k}$**:** Covariance matrices of the Gaussian components.
- $\boldsymbol{\pi_k}$**:** Weights of the Gaussian components.

**Expectation Step (E-step):** The second step is the expectation in which we compute the posterior probabilities (responsibilities) of each data point belonging to each cluster using Bayes' rule:

$$\gamma(z_{nk}) = \pi_k \cdot N(x_n|\mu_k, \Sigma_k) \; / \; \sum_{j=1}^{K} \pi_j \cdot N(x_n|\mu_j, \Sigma_j)$$

Here, $\gamma(z_{nk})$ is the responsibility of cluster $k$ for data point $n$.

**Maximisation Step (M-step):** The third step is to update the parameters $\boldsymbol{\mu_k}$, $\boldsymbol{\Sigma_k}$, $\boldsymbol{\pi_k}$ to maximise the expected log-likelihood obtained in the Expectation step. The log-likelihood function is given by:

$$\ln(P(\{x_n\}|\Theta) = \sum_{n=1}^{N} \ln\left(\sum_{k=1}^{K} \pi_k \, N(x_n|\mu_k, \Sigma_k)\right)$$

where $\Theta$ represents the parameters of the GMM.

We then keep repeating the Expectation and Maximisation step until convergence. Convergence can be defined in two ways. Either when the change in log-likelihood is below a threshold or after a predetermined number of iterations. EM guarantees an increase in the log-likelihood with each iteration. The algorithm converges to a local maximum of the likelihood function.

After convergence, the cluster assignments are determined by selecting the cluster with the highest responsibility for each data point.

**Implementation of Algorithm in C++**

To implement the Expectation Maximisation Clustering algorithm in C++ we create a structure ‘DataPoint’ that holds a vector of features (represented by double values). Each data point in the training set consists of a set of features representing a data instance.

‘EMClustering’ Class is used to implement the EM clustering algorithm. The class has private data members for the number of clusters, number of features, number of iterations, data points, cluster means, covariances, and weights. The Constructor of this class takes the number of clusters (k), number of features per data point (features), and the number of iterations (iterations) as input and

reads data from a CSV file using the ‘parseCSVtoMatrix’ function and populates the dataPoints vector.

The initialise method of this class initialises the means, covariances, and weights of each cluster randomly. The means are sampled from a uniform distribution between 0 and 10, and the initial covariance is set to 1.0, which represents the assumption of spherical covariance. Equal weights are assigned to each cluster initially. The ‘gaussianPDF’ method computes the probability density function of a multivariate Gaussian distribution for a given data point, mean, and variance.

The run method of the class iteratively performs the E-step and M-step for a specified number of iterations. In the E-step it computes the responsibilities of each data point belonging to each cluster using the current parameter estimates and normalises the responsibilities to ensure they sum to 1 for each data point. After this, in the M-step it updates the parameters to maximise the expected log-likelihood obtained in the E-step.

## Dataset

The dataset used for this algorithm is the classical Iris dataset which has three clusters namely: Setosa, Virginica and Versicolor. This dataset has 4 features namely: sepal length, sepal width, petal length and petal width.

## Results

Following are the results from the EM clustering:

```
PS C:\Users\ADARSH GOEL\OneDrive\Desktop\ML Assignment> cd "c:\Users\ADARSH GOEL\OneDrive\Desktop\ML Assignment\" ; if ($?) { g++ EMClustering.cpp -o EMClustering }
; if ($?) { .\EMClustering }
Cluster means and covariances:
Cluster 0 - Mean: (6.80964, 3.07124, 5.72461, 2.10602) - Covariance: (0.284525, 0.0821644, 0.248572, 0.0601976)
Cluster 1 - Mean: (5.92776, 2.7504, 4.40637, 1.41354) - Covariance: (0.232006, 0.0873541, 0.276251, 0.0691561)
Cluster 2 - Mean: (5.006, 3.428, 1.462, 0.246) - Covariance: (0.121764, 0.140816, 0.029556, 0.010884)
```

Here, the cluster 0 refers to Virginica which has actual mean value of (6.588, 2.974, 5.552, 2.026). The cluster 1 refers to Versicolor which has actual mean value of (5.936, 2.77, 4.26, 1.326). The cluster 2 refers to which has actual mean value of (5.006, 3.428, 1.462, 0.246). The results are pretty close to actual values.

## Advantages and Disadvantages of Expectation-Maximization Clustering Algorithm

### Advantages:

1. **Soft Assignment:** EM provides soft assignments, meaning each data point is probabilistically associated with multiple clusters. This is beneficial in scenarios where data points may belong to more than one cluster i.e. there is an overlap in the clusters.

2. **Handling Complex Cluster Shapes:** EM can model clusters with different shapes and sizes, as it uses a mixture of Gaussians. This flexibility makes it suitable for datasets where clusters exhibit varying patterns.

3. **Modelling Covariance:**Unlike K-means, EM takes into account the covariance between features when modelling clusters. This allows it to capture correlations and orientations in the data, providing a more accurate representation.

4. **Handling Missing Data:** EM is robust in handling datasets with missing or incomplete information. The algorithm computes expectations over latent variables, making it suitable for scenarios where not all features are observed for every data point.

5. **Convergence Guarantee:** The EM algorithm guarantees an increase in the likelihood function with each iteration, and it converges to a local maximum. This property ensures the stability and reliability of the algorithm.

6. **Applicability to Skewed Distributions:** EM clustering can effectively handle clusters with skewed or elongated shapes, which might be challenging for algorithms like K-means that strongly assume spherical clusters.

7. **Probabilistic Framework:** EM provides a probabilistic framework, allowing for the incorporation of prior knowledge or constraints. It facilitates a more intuitive understanding of the uncertainty associated with cluster assignments.

**Disadvantages:**

1. **Sensitivity to Initial Conditions:** EM clustering is sensitive to the choice of initial parameters. Different initializations may lead to different final results, including convergence to different local optima.

2. **Computational Complexity:** The algorithm involves computing probabilities and updating parameters in an iterative manner, which can be computationally expensive, especially for large datasets. The complexity increases with the number of clusters and features.

3. **Assumption of Gaussian Distribution:** EM assumes that the underlying data distribution is a mixture of Gaussians. If the true distribution significantly deviates from this assumption, the model may provide suboptimal results.

4. **Number of Clusters Specification:** The number of clusters needs to be specified beforehand. Selecting an inappropriate number of clusters can lead to the detection of spurious patterns or the merging of distinct clusters.

5. **Not Suitable for Non-Gaussian Distributions:** EM assumes that each cluster follows a Gaussian distribution. If the data is not well-modelled by Gaussian distributions, alternative clustering algorithms may be more appropriate.

6. **Sensitive to Outliers:** EM clustering can be sensitive to outliers, as they can disproportionately influence parameter updates during the expectation-maximisation steps.

7. **Convergence Speed:** Convergence to a local maximum may be slow, especially in high-dimensional spaces, and the algorithm may require a large number of iterations to reach convergence.