# A Hybrid Approach to Questionnaire Generation Using Las Vegas Algorithm and Naïve Bayes Algorithm

Kyle Dominic Galima[1], Kyle Marin[1], Richard Regala[1], Mark Christopher Blanco[1], Dan Michael Cortez[1]

[1]*krmarin2018@plm.edu.ph*
[1]*kdjgalima2018@plm.edu.ph*
[1]*rcregala@plm.edu.ph*
[1]*mcrblanco@plm.edu..ph*
[1]*dmacortez@plm.edu.ph*
*Computer Science Department, College of Engineering and Technology*
*Pamantasan ng Lungsod ng Maynila (University of the City of Manila)*
*Intramuros, Manila 1002, Philippines*

**Abstract**

Examination plays an important role in testing the student's knowledge. The conventional style of creating, preparing, and generating test questions manually is intriguing and cumbersome. In this study, we introduce an automated questionnaire generation system that uses a hybrid approach using Naïve Bayes (NB) and Las Vegas Algorithm (LVA). LVA is a randomized algorithm by Laslo Babai that always produces a correct result but has a running time that is based on a random value. While NB is one of the most well-known data mining algorithms for classification.

*Keywords*: Las Vegas Algorithm; Naïve Bayes Algorithm; Questionnaire Generation;

## 1. Introduction

Test examinations and quizzes are what we use today to evaluate and assess students' intellectual capabilities. Examinations play an important role in testing the students' knowledge (Venitha, 2021). It is a powerful educational tool that helps both the student and the teacher measure one's knowledge to reflect on whether the teaching methods or materials used by the teachers and professors are effective and if the students are learning the information that a teacher/professor expects them to learn.

Assessment is ubiquitous in any educational context since it serves several functions such as monitoring teaching and learning (Collins et al., 2018). Without it, we cannot determine whether the goals of education are being met and we cannot identify the needs and strengths of an individual. Designing tests is an important part of assessing students' understanding of course content and their level of competency in applying what they are learning (University of Washington, 2020). Creation, preparation, and generation of test questions can be

time-consuming. It also limits the amount of content that can be tested, and grading can be highly subjective and unreliable.

To eliminate the disadvantages of creating, preparing, and generating test questions, we propose an automated questionnaire generation system. In framing test questions, it requires several parameters like difficulty level, test subject, etc. This study aims to use Naïve Bayes Algorithm to classify each question and use Las Vegas Algorithm to efficiently and effectively select the appropriate test questions that will be used.

In the year 1979, Laslo Babai introduced the Las Vegas Algorithm. It is created as a dual to Monte-Carlo Algorithms, wherein it was established as a context for graph isomorphism problems. Babai named it as "Las Vegas Algorithm" alongside an example concerning coin flips where the algorithm relies on a series of independent coin flips with a small chance of failure (no result). Randomization is the process of generating something random in which it does not follow a deterministic pattern, but follows an evolution described by a probability distribution. In an algorithm, it is used as a source of randomness as a part of its logic (Brilliant.org) to reduce both the time and memory spent/used. It works by generating a random number within a specified range and then making decisions based on the value generated. In computing, Las Vegas Algorithm is a randomized algorithm that often produces correct results or, if not, informs the user of the failure. It guides their search with randomness in such a way that a correct answer is assured even if unfortunate choices are made (Nandi, 2011).

While Naïve Bayes Algorithm is a classification technique based on Bayes' theorem with an assumption of independence among predictors (Ray, 2017). The necessity to estimate multivariate probabilities from training data motivates this assumption. In practice, most attribute value combinations are either absent or in insufficient numbers in the training data. As a result, direct estimation of each multivariate probability will be unreliable.

## 2. Related Studies

There have been a lot of works that have tried to tackle the arduous task of shortening the run-time of the Las Vegas Algorithm (LVA). In fact, in the study conducted by Truchet et al., (2013), it was described as "a randomized algorithm whose run-time might vary from one execution to another, even with the same input." That is because the analysis of LVA does not depend on input distribution but on the random choices that the algorithm makes. This can be seen in one of its iterations, the Quick Sort, where a random pivot value is chosen for each running time, and the run-time will be dependent on the pivot's position on the list. "Since this is a comparison-based algorithm, the worst-case scenario will occur when performing the pairwise comparison, taking

$$O(n\text{^}2)O(n^2) \quad (1)$$

", which means that the run-time goes up linearly while the *n* goes up exponentially (Alman and Williams, 2020). Some people argue that the run-time of the algorithm is dependent on the CPU. "Instead of actually measuring run-time distributions in terms of CPU-time, it is often preferable to use representative operation counts as a more machine-independent measure of an algorithm's performance (Hoos and Stutzle**,** 2013)". This operation count method is done by selecting one or more arithmetic operations, running it together with the algorithm, comparing each operation, and identifying the most time-consuming operation, making that the maximum execution time. If the operation count method is integrated into the example used earlier which was

the Quick Sort algorithm, the count would be at every swap operation. This method proves that the run-time of the Las Vegas algorithm is not CPU-dependent and can be improved upon by using other methods.

The main problem that the Las Vegas algorithm has is its random run-time. "There are several ways to reduce the degree of randomness in probabilistic algorithms…, one approach is to derive a Las Vegas algorithm using Monte Carlo algorithms. Another way is to completely remove the randomization in an algorithm to make it deterministic (Tempo and Ishii, 2007)", this can also be called *derandomization*. Though this method can be expanded upon to further improve the algorithm, the questionnaire generation system where the researchers will implement the enhanced Las Vegas algorithm needs randomness for some of its features and in turn, this method cannot be used.

The algorithm has never been used or modified for a questionnaire generation system, but some studies have used it for searching-systems. In a study conducted by Alman et. al. (2020), the researchers presented "a Las Vegas algorithm for offline Approximate Nearest Neighbor search (ANN)," and the algorithm used managed to match the best running time of the Monte Carlo algorithm, which is an algorithm often compared with Las Vegas algorithm, but it usually has a faster run-time. What the researchers did, in the context of the example used earlier in Quick Sort, was the use of random partitions instead of one random sample (pivot). The researchers made one key modification each step so that the resulting probabilistic polynomials either gave the correct answer or a value indicating that an error has occurred. The run-time has been shortened but there was still a possibility of outputting the wrong answer.

Another way that this random run-time problem has been addressed is by improving on the Las Vegas Filter (LVF) which is another iteration of the Las Vegas algorithm that utilizes randomness to guide their search wherein a correct solution is guaranteed. The Las Vegas Filter finds the current best solution while the search for the best attributes (*M*) continues. It generates a random subset (*S*), from several features (*N*) in every round. If *N* of *S* (*C*) is less than the current best *C*

$$(C_{best}), \text{ i.e., } C < C_{best}, \tag{2}$$

the inconsistency of the data (*D*) with the features prescribed in *S* is checked against an inconsistency criterion. If its inconsistency rate is below a pre-specified one (*A*),

$$C_{best} \text{ and } S_{best} \tag{3}$$

are replaced by *C* and *S* respectively; the new current best *S* is printed. If

$$C = C_{best} \tag{4}$$

and the inconsistency criterion is satisfied, then an equally good current best is found and printed. When LVF loops MAXIMUM_TRIES times, it stops. The value of MAXIMUM_TRIES is taken at input run-time. Figure (1):

| Input | |
|---|---|
| D | Data Set |
| N | Number of Attributes |
| A | Allowable inconsistency rate |
| Output | |
| Sets of M features satisfying the inconsistency criterion | |

$C < C_{best}$, = $N$;
for $i$ = 1 to MAXIMUM_TRIES
    $S$ = randomSet(seed);
    $C$ = numOfFeatures($S$);
If ($C < C_{best}$)
  If (InconsistencyCheck ($S$, $D$) < $A$)
    $S_{best}$= $S$;
    $C_{best}$ = $C$;
  print_Current_Best($S$);
else if (($C = C_{best}$) and (InconsistencyCheck ($S$, $D$) <))
print_Current_Best($S$);
**end for**

Figure 1. Las Vegas Filter (LVF) Algorithm

Basically, it outputs the current best outcome it has gotten at the end of the indicated MAXIMUM_TRIES. The higher the value of the MAXIMUM_TRIES, the better the output would be, but the problem is "when datasets are huge, the running time of LVF is no doubt longer. In order to speed up the preprocessing, one way of improving the present one-go approach is to go for sampling (Nandi, 2011)." The idea the researchers presented was similar to the way Alman did it, by using random partitions rather than a random sample but they modified it to tackle the problem the LVF had. When they have a large dataset, their proposed Enhanced Las Vegas (ELV) algorithm will take a percentage of that dataset and make it into training data. ELV does the same iteration of steps as LVF to the partitioned data and does it for MAXIMUM_TRIES times. The process is repeated with the next partitioned data until the current best features are found and compared to the previous current best features. This entire process is repeated up to $k$ number of times (taken as input at run-time) and the final best feature ($S_1$) is the output. Figure (2):

| Input | |
|---|---|
| D | Data set Described by X, \|X\| = n |
| n | Number of Attributes |
| A | Allowable inconsistency rate |
| k | Number of Samples |
| p | Sample percentage |
| MAXIMUM_TRIES | Number of iterations of each sample |
| Output | |
| Sets of M features satisfying the inconsistency criterion (m<n) | |

**for** $j$ = 1 to $k$
  $S_0$ = draw a sample of size $p$% from $D$
  $C_{best}$ = n; Seed = X;
  **for** i = 1 to MAXIMUM_TRIES
    $X^1$ = randomSet(seed);

C = numOfFeatures($X^1$);
If (C < $C_{best}$) and (InconsistencyCheck ($X^1$,$S_0$) < $A$)
Best = $X^1$ ; $C_{best}$ = C;
print_Current_Best (*Best*);
end if
**end for**
if ($j$ = 1)
$S_1$ = *Best*;
Inconsistency = InconsistencyCheck (*Best*, $S_0$)
$C^1 = C_{best}$
else if(InconsistencyCheck(Best, $S_0$) < Inconsistency) and ($C_{best} \leq C_1$)
$S_1$ = *Best*
$C_1 = C_{best}$ ;
*Inconsistency* = InconsistencyCheck(*Best*, $S_0$)
**end for**
print_Current_Best ($S_1$)

Figure 2. Enhanced Las Vegas Filter (ELV) Algorithm

In another study conducted by Mahalanobis et al. (2018), they used the las vegas algorithm to solve the elliptic curve discrete algorithm problem by reducing it into a linear algebra problem. They chose the LVA instead of using an exhaustive search because in the exhaustive search it would randomly pick a set and then check the sum of the points while the LVA selects and checks any random points simultaneously. Using LVA provides efficient process and result which is generally it's one of the main advantages.

Naïve Bayes algorithm as stated by Vangara et al. (2020) is "a classification technique that can be used as the supervised machine learning algorithm that uses Bayes theorem which relies on conditional probability.". It assumes that the presence of one feature in a class is unrelated to the presence of any other feature. When applied to text classification, spam filtering, or sentiment analysis, it predicts the tag of a text and calculates the probability of it as used in a document or sentence.

Naïve Bayes algorithm can be applied to 4 applications: (1) Real-time prediction where it is a fast eager learning classifier, (2) Multi-class prediction where it predicts multiple classes of a target variable, (3) Text classification/spam filtering/sentiment analysis where it is widely used due to its high success rates compared to other algorithms, and (4) recommendation system where it can be used together with a collaborative filtering system to filter hidden information and predict or recommend it to a user if he/she would like a copy of the information.

## 3. Existing Las Vegas Algorithm

### *3.1. Overview*

The Las Vegas Algorithm was created in 1979 by Laslo Babai. It was developed as an alternative to Monte-Carlo Algorithms, in which it was used to solve graph isomorphism issues. Las Vegas Algorithm can be also classified as a randomized algorithm which is a procedure that utilizes a source of randomness as part of its logic (Brilliant.Org). It is usually used to lessen the time complexity and memory used in a standard algorithm. If the goal is to just eliminate errors in the output, better results are possible with randomized Las Vegas Algorithms (Alman et al., 2020). A basic example is a randomized Quicksort, where the pivot is randomly chosen and divides the elements into three parts. The elements that are less than the pivot, the

element that is equal to the pivot, and the elements that are greater than the pivot. The randomized quicksort always generates a solution which is the sorted array.

### *3.2. The Problem in Las Vegas Algorithm*

In LVA, the data is not classified/categorized. The algorithm has to pass through each data every iteration until the data needed is located or until it fails to locate it resulting in the termination of the program with prompting a message "No results...". Its time complexity also depends on the number of data and the random choices made. The process may take some time if the element needed isn't found therefore failing, or it successfully locates the element needed immediately. There exists a relatively general way of creating efficient Las Vegas versions of state-of-the-art high-dimensional search data structures (Ahle, 2017).

### *3.3. Pseudocode of Las Vegas Algorithm*

```
Pick an element x (from an array/database) randomly.
    Choose randomly between 1 and n.
        Generate a random number t
        Since the range of numbers in which we want a random number is
        [start, end]
        Hence, we do, t = t % (end-start+1)
        Then, t = start + t;
        Hence t is a random number between start and end.
Compare x with a randomly selected element.
    If x matches the randomly selected element, return the element.
    Else If x is greater than the mid element, then x can only lie in the right
    half subarray after the mid element. So, recur for the right half.
    Else (x is smaller) recur for the left half.
```

## 4. Enhanced Las Vegas Algorithm (Hybrid Las Vegas Algorithm and Naïve Bayes Algorithm)

### *4.1. Enhancement of the Algorithm*

To address the problem, Naïve Bayes Algorithm is merged with the Las Vegas Algorithm to create an accurate and efficient approach to classifying and grouping the data based on key matches. We then now select a group from the database based on the parameter needed.

Feature Selection as referred to by Guyon et al. (2008) is "a dimensionality reduction technique that tries to remove irrelevant and redundant features from original data.". Thus, its objective is to determine a subset of features that accurately identifies a problem while causing the least amount of performance degradation, resulting in simpler and more accurate schemes. It is also known as feature subset selection or attribute selection or variable subset selection and is one of the core concepts that is commonly used in machine learning, wherein you select a subset of the features available from the data to apply in a learning algorithm.

Feature selection can be broadly categorized into 2 methods, the (1) Wrapper Method which uses an evaluation function dependent on a learning algorithm (Kohavi & John, 1997). As part of the learning process, they are aimed at optimizing a predictor. And the (2) Filtering Method in which it uses other

selection techniques as separability measures or statistical dependences. They only consider the general characteristics of the dataset, as they are independent of any predictor (Guyon & Elisseeff, 2003).

In this research study, the method that will be used is the filtering method defined by Guyon & Elisseeff. Due to learning independence, filtering methods usually lead to better generalization. However, since they commonly choose greater feature subsets, they can sometimes require the use of a threshold. Filtering methods must be used when the number of features is high (especially in the case of big data), as they are much faster than the other approaches.

*4.2. Pseudocode of the Hybrid Algorithm*

```
Naive Bayes Algorithm
    Text Preprocessing
    Calculate the probability for each word in a text and filter the
    words which have a probability less than the threshold
    probability.
    Train the data set.
    Predict/Classify using conditional probabilities.
Pick a group x from the database based on the parameter needed.
    Choose randomly between 1 and n from the group chosen.
    Generate a random number t
    Since the range of numbers in which we want a random number is
    [start, end]
    Hence, we do, t = t % (end-start+1)
    Then, t = start + t;
    Hence t is a random number between start and end.
Compare x with a randomly selected element.
    If x matches the randomly selected element, return the element.
    Else If x is greater than the mid element, then x can only lie in the right
    half subarray after the mid element. So, recur for the right half.
    Else (x is smaller) recur for the left half.
```

**5. Methodology**

To determine whether the proposed approach will enhance the original Las Vegas Algorithm, the researchers used an experimental design. The original Las Vegas Algorithm without the Naïve Bayes Algorithm serves as the baseline and be compared to the proposed enhancement defined in section 4.2. Two data sets will be used for producing the results of the experiments for comparison between the original and enhanced algorithms. The two (2) data sets will be consisting of 5 subjects with 10 sets of questions each. Both data sets will not be classified and there will be parameters set. The algorithms need to create 1 set of questionnaires with 10 items consisting of the 5 subjects. If the quiz is created randomly, it can't accommodate specific constraints (Fakhrusy & Widyani, 2017). The enhanced algorithm provides an opportunity to generate a questionnaire that approximates the constraints.

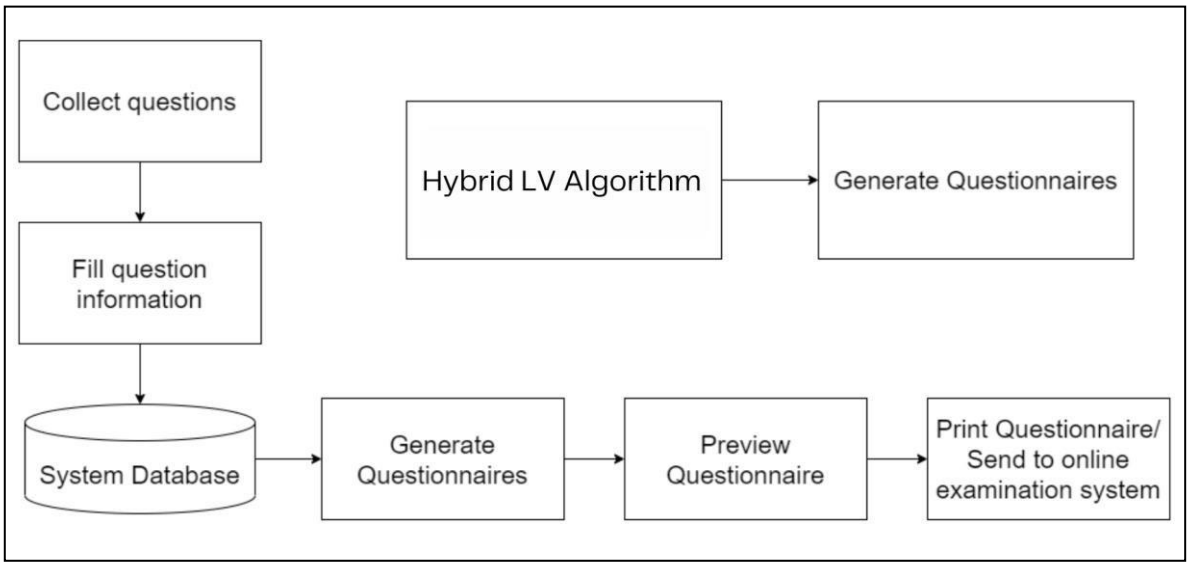


Figure 3. System Architecture

*5.1. Naïve Bayes Method*

In the figure above, it first collects the questions and fills in the information needed (question, answers, and other parameters like time created, level of difficulty). It is then stored in a database wherein the algorithm fetches the data and analyzes and executes it. The enhanced algorithm first preprocesses the data where it calculates the probability for each word in a question and filters the words which have less than threshold probability. We then train the data set so that it can predict/classify it using conditional probabilities.

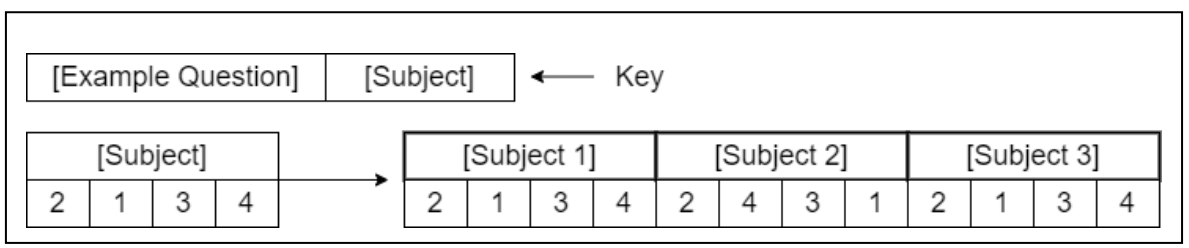


Figure 4. Hybrid Naïve Bayes Classifier

The figure above shows a simple implementation of how the Naïve Bayes algorithm is used to key each data and group them based on key matches. An example question is classified by giving it a key based on its intended subject. Questions are then grouped into their respective subjects where they can be picked based on

the parameters needed. If a certain subject is chosen, a random number between 1 and *n* from the group will be chosen. Afterward, a pivot is picked from the questions, and the Naïve Bayes algorithm is used to arrange the questions.

## 6. Results

The results of the questionnaire generated are presented here. To generate a questionnaire, parameters are set. There should be 2 test questions each from each subject to create a 10-item questionnaire. Each subject has 10 test questions, and the database has a total of 50 questions. The questions are not classified. Questions generated using the traditional Las Vegas algorithm did not give an equal number of questions for each subject, on the other hand, the hybrid algorithm presented 2 questions each from each subject, dividing the data equally.

Table 1. Generated Questionnaires

| Subjects | LVA | Hybrid Algorithm |
|---|---|---|
| Math | 1 | 2 |
| Science | 4 | 2 |
| English | 1 | 2 |
| PE | 2 | 2 |
| Arts | 2 | 2 |
| Total | 10 | 10 |

The Hybrid Algorithm showed accurate and consistent results apart from the original algorithm which only randomly chooses test questions to generate a questionnaire. The original algorithm successfully created a questionnaire, but it failed to achieve the required number of test questions per subject.

Table 2. Naïve Bayes Classifier

| Subjects | Naïve Bayes Classifier | Actual Total Number of Questions per Subject |
|---|---|---|
| Math | 10 | 10 |
| Science | 10 | 10 |
| English | 10 | 10 |
| PE | 10 | 10 |
| Arts | 10 | 10 |
| Total | 50 | 50 |

The Las Vegas Algorithm was shown to be faster than the proposed Hybrid Algorithm. This is due to the addition of a classification method that keys out data based on certain parameters and groups them into their assigned databases for a more efficient and concise search method.

Table 3. Time Complexity

| | LVA | Hybrid Algorithm |
|---|---|---|
| Runtime | 7 | 10 |

## 7. Conclusion

Based on the results collected, the enhanced algorithm performs more accurately in complying with the set parameters. While the original algorithm performs faster but it doesn't meet the required parameters. Although the enhanced algorithm is slightly slower than the original one, it can still be up to par, and it is significantly more precise. Accuracy and consistency are much more important than run time in creating a questionnaire. If a questionnaire is inaccurate and inconsistent, its integrity and value are compromised.

For future works, the researchers recommend considering additional features like calibration of the level of difficulty of a test question and using Natural Language Processing to create questions based on information from journals, websites, etc. for the questionnaire. A neural network can also be used to modify test questions and their other parameters to further enhance the system. The questionnaire generation system can also be paired with an online examination system so that it can be fully utilized and produce immediate results when calibrating the test questions.

## Acknowledgments

The researchers wish to express their gratitude to God for providing them with the knowledge, wisdom, patience, and will they need throughout their journey. To their family for their unending love and support. To Prof. Richard Regala, for being the study's adviser. For the solid support and guidance from the Pamantasan Ng Lungsod Ng Maynila - College of Engineering and Technology - Computer Science Department faculty and staff.

## References

Venitha, M. E. (2021). AUTOMATIC QUIZ GENERATOR USING DEEP LEARNING. International Journal of Advanced Engineering Science and Information Technology, 4(4).

Collins, D. P., Rasco, D., & Benassi, V. A. (2018). Test-enhanced learning: Does deeper processing on quizzes benefit exam performance?. Teaching of Psychology, 45(3), 235-238.

University of Washington (2020). Constructing Tests. Center for Teaching and Learning. https://teaching.washington.edu/topics/preparing-to-teach/constructing-tests/.

Randomized Algorithms. Brilliant.org. Retrieved 22:44, May 14, 2021, from https://brilliant.org/wiki/randomized-algorithms-overview/

Nandi, G. (2011, March). An enhanced approach to Las Vegas Filter (LVF) feature selection algorithm. In 2011 2nd National Conference on Emerging Trends and Applications in Computer Science (pp. 1-3). IEEE.

Ray, S. (2017). 6 Easy Steps to Learn Naive Bayes Algorithm with codes in Python and R. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/

Truchet, C., Richoux, F., & Codognet, P. (2013, October). Prediction of parallel speed-ups for las vegas algorithms. In 2013 42nd International Conference on Parallel Processing (pp. 160-169). IEEE.

Alman, J., Chan, T. M., & Williams, R. (2020). Faster deterministic and Las Vegas algorithms for offline approximate nearest neighbors in high dimensions. In Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (pp. 637-649). Society for Industrial and Applied Mathematics.

Hoos, H. H., & Stutzle, T. (2013). Evaluating las vegas algorithms-pitfalls and remedies. arXiv preprint arXiv:1301.7383.

Tempo, R., & Ishii, H. (2007). Monte carlo and las vegas randomized algorithms for systems and control*: An introduction. European journal of control, 13(2-3), 189-203.

Mahalanobis, A., Mallick, V. M., & Abdullah, A. (2018, December). A Las Vegas algorithm to solve the elliptic curve discrete logarithm problem. In *International Conference on Cryptology in India* (pp. 215-227). Springer, Cham.

Vangara, V., Vangara, S. P., & Thirupathur, K. (2020). Opinion Mining Classification using Naive Bayes Algorithm. International Journal of Innovative Technology and Exploring Engineering (IJITEE), 9(5), 495-498.

Ahle, T. D. (2017, October). Optimal las vegas locality sensitive data structures. In 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS) (pp. 938-949). IEEE.

Guyon, I., Gunn, S., Nikravesh, M., & Zadeh, L. A. (Eds.). (2008). Feature extraction: foundations and applications (Vol. 207). Springer.

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. Artificial intelligence, 97(1-2), 273-324.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. Journal of machine learning research, 3(Mar), 1157-1182.

Fakhrusy, M. R., & Widyani, Y. (2017, November). Moodle plugins for quiz generation using genetic algorithm. In 2017 International Conference on Data and Software Engineering (ICoDSE) (pp. 1-6). IEEE.