**CSF 407 – ARTIFICIAL INTELLIGENCE**

**Improving Rules of Classifiers**

**USING**

**Genetic Algorithms with Heuristic Searches, Random Search, Grid Search, Fuzzy Search, SVM based Rule Extraction Techniques, Particle Swarm Optimization, Ant Colony Optimization and Ensemble Learning**

| **S NO.** | **TEAM MEMBERS ID** | **NAMES** |
| --- | --- | --- |
| **1** | 2020A7PS0282H **(Leader)** | Anish Kumar Kallepalli |
| **2** | 2020A7PS1080H | Ashwin Naveen Pugalia |
| **3** | 2020A7PS0153H | Milind Jain |
| **4** | 2020A7PS0273H | Sriram Srivatsan |

**GROUP ID - 11**

# **1. INTRODUCTION**

Artificial Intelligence techniques have revolutionized the field of classification, enabling us to automate the process of categorizing data with a high degree of accuracy. With these techniques, we've been able to automate the process of putting data into groups while maintaining a high level of accuracy. AI-based classifiers are becoming more popular in a wide range of fields, such as natural language processing, image recognition, and finding fraudulent activity. For many different applications, like finding information, mining data, and making decisions, it is important to be able to classify data in an accurate and quick way.

Even so, AI-based classifiers are limited by the way they were made, no matter how well they work. One of the biggest problems that must be solved is how hard it is to come up with classification rules that are both accurate and useful. Creating rules that can handle the subtleties and complexities of real-world data is hard and takes a lot of time. Most of the time, the quality of the rules used limits how accurate the classifiers that come out of this process can be.

Researchers have turned to methods based on artificial intelligence to improve the accuracy and speed of classifiers and get around these problems. This group includes methods like machine learning, deep learning, genetic algorithms, and fuzzy logic. By automating the process of making rules and making them better, artificial intelligence techniques could make classifiers much more accurate.

Machine learning is one among the most common ways that artificial intelligence is used to enhance classifiers. In the field of machine learning, algorithms are "trained" by letting them look at a set of data. This lets the algorithms find patterns and relationships in the data. After being trained, the algorithm can use the patterns it has learned to figure out how to classify new data. Unsupervised machine learning algorithms use unlabeled data while supervised machine learning algorithms require labeled data.

Deep learning is an essential part of machine learning that has become a lot more popular in the last few years.Deep learning algorithms employ multiple-layer neural networks, which lets them learn intricate patterns from the data. Deep learning has worked really well in areas like speech and image recognition.

Genetic algorithms are another type of artificial intelligence that can be used to improve classifiers. These algorithms make a group of rules, judge how well they work, and then use genetic operators like crossover and mutation to make new rules. The process of making and judging new rules is done over and over again until a set of rules that works well is found. Another search algorithm that has been used to improve rule-based classifiers is particle swarm optimization. This algorithm looks for the best set of rules by simulating how particles move through a space with many dimensions.

AI techniques, like search algorithms, have shown a lot of promise for making rule-based classifiers better, but using them is not easy. One big problem is that it's hard to find a good balance between accuracy and being easy to understand. Search algorithms can make rules that are very complicated and hard to understand, making it hard for users to understand how the classifier decides what to do. This inability to be understood can make it hard to trust the classifier's decisions and make it less useful in some areas.

Unfairness and bias are also problems that need to be solved. AI-powered classifiers often get their rules from patterns that keep showing up in the data they've already collected. If the historical data has biases or mistakes, the classifiers that are made may also have these problems. This could lead to results that are unfair or discriminatory. In the past few years, this problem has gotten more and more attention, and researchers have been trying to find ways to make AI-based classifiers less biased.

Even with these problems, using AI to improve classifiers could completely change the way we solve classification problems. Because these methods automate rule creation and refinement, they can improve the performance of classifiers, save time, and cut down on the number of mistakes that happen as a result of rule creation and refinement. As the study of artificial intelligence continues to expand, it's probable that we'll see the creation of new, cutting-edge ways to improve classifiers by using AI-based methods.

This literature review gives an overview of the current state of research in this field. It talks about the different AI techniques that have been used, how well they work, and the problems that still need to be solved. In addition, the survey brings up a number of research questions that haven't been answered yet, as well as possible new research directions. In general, the application of AI methods to the enhancement of classifiers has the potential to revolutionize many different industries.

# **2. LITERATURE SURVEY**

The paper "Genetic Algorithm and Tabu Search for Feature Selection." by Sabra El Ferchichi, Kaouther Laabidi, and Salah Zidi propose an approach for selecting features using SVM with two heuristic optimizations. The aim of the paper is to efficiently search for the most defining features that will improve the computational performance of the previous algorithms. The misclassification rate is a critical criterion used for its evaluation. First, the Tabu algorithm is used as a supervisor to steer the search for the best feature set, after which a genetic algorithm is applied. *Tabu search* is based on exploring the entire solution space starting from an initial one. In every iteration, we choose the best solution near already generated current solutions. The tabu list is used to memorize the last visited solution so that it does not get stuck in some local optima instead of reaching the global optimum. Natural selection is the basis for genetic algorithms. They use genetics to work around the search space. The search starts from a randomly chosen sample of solutions. The SVM-driven Tabu search that the authors have suggested uses SVM as an inductive approach. An evaluation criterion is the performance of each combination of characteristics in classification. The search is supervised by an engine which uses the above mentioned heuristics.

In the paper titled "Classification Using Ant Colony Optimization," David Martens and his colleagues present a unique method for dealing with classification difficulties (ACO). The continuous data are discretized during the pre-processing stage of the proposed technique, which also helps minimise the total amount of characteristics. Following the extraction of the rules through the application of ACO, the data are subsequently classified accurately utilising those rules. The ACO search algorithm is a metaheuristic one that takes its cues from the way that actual ant colonies operate. The method is a simulation of the way in which ant colonies employ a probabilistic decision-making process to locate the food source that is the most direct route to their colony. ACO is a tool that may be used to determine the best set of rules for classifying a dataset's various elements. The authors put their suggested method to the test by comparing it to both decision trees and support vector machines by using various different benchmark datasets. According to the findings of the experiments, an ACO-based classification can provide a manageable number of rules while yet retaining classification accuracy that is comparable to or even greater to that of the other techniques.

Jasper Snoek, Hugo Larochelle and Ryan P. Adams in their paper “Practical Bayesian Optimization of Machine Learning Algorithms” have proposed a cutting-edge technique called Bayesian optimization. Bayesian optimization employs a probabilistic model to demonstrate how the hyperparameters affect the model's performance. To select the next hyperparameters to evaluate, the approach uses an acquisition function that balances exploration (sampling unexplored regions) and exploitation (sampling regions with high expected improvement). One of the acquisition functions used in the paper is the Expected Improvement, which is a heuristic that evaluates the expected improvement in the objective function if a certain hyperparameter configuration is sampled. There are diverse variations of Bayesian optimization, such as tree-structured Parzen estimators and Gaussian process-based methods, which can be considered distinct search strategies that explore the hyperparameter space and utilize different models to predict the performance of the model. This is similar to how search algorithms use other heuristics and search strategies to explore the solution space. One of the advantages of Bayesian optimization is its ability to manage noisy and expensive-to-evaluate functions, which is often the case when optimizing hyperparameters for machine learning models. Bayesian optimization has limitations, such as choosing appropriate priors for the model and the computational cost of training the probabilistic model. We can see the similarity between search algorithms and Bayesian optimization since even search algorithms may face challenges such as scalability and the curse of dimensionality. Bayesian optimization can be intricately linked to the concept of search algorithms in AI and how exploring different variations can lead to better results.

The captivating work of Leo Breiman in his paper "Bagging Predictors" is a compelling piece that unveils a machine learning technique known as bagging that directly correlates with the concept of search algorithms in AI. Bagging is a fascinating process in which different models are trained on subsets of the data, followed by predictions and are combined to make a more accurate overall model. This process is synonymous with exploring the space of potential models, which mirrors how search algorithms probe various solutions to a problem. Bagging has several benefits, including its ability to diminish the variance of models, which can be significantly helpful when dealing with noisy data or complex models that are prone to overfitting. This reduction in variance is like how search algorithms endeavor to balance exploration and exploitation to identify the best solution while avoiding getting stuck in local optima. The "out-of-bag error estimate" heuristic is used by the authors to estimate how well the bagged classifier performs on fresh data. The paper suggests another method for adding diversity to the base classifier ensemble called "random subspace method" as well. When training each instance of the base classifier using this method, a subset of the features is chosen at random. This approach resembles how search algorithms can utilize other heuristics and search strategies to probe the solution space. However, bagging has limitations, such as sensitivity to outliers and heightened computational complexity. This paper illuminates how exploring different variations can lead to better results, which is highly applicable to other fields that rely on AI-based solutions.

The paper "Rule extraction from support vector machines" by Haydemar Núñez introduces a mind-boggling approach for extracting lucid rules from Support Vector Machines using a search algorithm to identify the most relevant features and support vectors. SVMs are a powerful machine learning technique that can model intricate nonlinear relationships between features and labels. However, the models generated by SVMs are often perplexing to interpret as they rely on complex mathematical functions. Conversely, rule-based models are effortless to interpret and can provide penetrating insights into the decision-making process of a model. The proposed approach includes utilizing a genetic algorithm to rummage for the most relevant features and support vectors in an SVM model. The genetic algorithm employs a fitness function that assesses how well the rules were extracted based on their accuracy and complexity. The extracted rules are embodied as a set of decision trees, where each node corresponds to a decision based on a feature value. The proposed approach achieved comparable accuracy on several benchmark datasets while proffering more interpretable and concise rules. The paper also proposes a method for selecting the hyperparameters of the SVM to optimize the accuracy of the extracted rules. This method uses a combination of grid search and cross-validation to find the optimal values of the SVM's parameters. A genetic algorithm accentuates the significance of search algorithms in AI, which scrutinize large solution spaces and discover the optimal solution. This approach endows a promising direction for rendering SVM models more intelligible and comprehensible.

Nahla H. Barakat and Andrew P. in the paper “Rule extraction from Support Vector Machines: A Sequential Covering Approach” have devised a unique approach to extract rules from SVMs using an iterative approach called SQRex-SVM. The construction of rules is based on the ordered search's most relevant feature as determined by interclass separation. The algorithm proceeds in a separate-and-conquer search approach followed by a rule pruning set. The rule is tested using various criteria like TP and FP. The ROC curve is an essential graphical representation of a classification model. It assesses the overall performance of the classification algorithm used. The paper also compares the performance of general SVM rule extraction and SQRex-SVM on various popular datasets. SQRex-SVM is far superior to general SVM in rule extraction as it extracts a more generalized and smaller set of comprehensible rules for classification. The most common method to evaluate or judge the quality of a rule set is the number of rules it has along with the amount of conditions in every rule. Still, the performance of the two classifiers cannot be reliably compared using accuracy as the criterion. Accuracy does not account for biased class priors or misclassification costs. Sequential covering algorithms generally try to learn rules explaining a specific portion of the training dataset, that is, class-wise. It uses a one vs. rest scheme where all training examples belonging to the particular class are considered positives, and the rest are considered negatives. Each rule tries to maximize the number of good examples while avoiding as many bad examples as possible. Iteration wise algorithm continues to learn new rules by pruning the training dataset of examples already covered by previous rules. We can set the threshold for halting, or it will keep learning new rules until they cover all positive cases.

SQMRex-SVM makes use of the information discovered by SVMs in two ways :

1. It considers only TP or TN points, thus minimizing loss due to misclassified data points or noisy data.
2. Only limited rules are searched, and these rules are refined to both TP and TN.

Comparing extracted rule sets by SQMRex-SVM to the SVM across a different test set, extracted rule sets outperform the SVM in many criteria like accuracy, TP and FP rates. Furthermore, the retrieved rules have excellent fidelity to the original SVM.

One major issue classification algorithms face while extracting rules are giving equal importance to every sample in the dataset. The paper “A Fuzzy K-Nearest Neighbor Algorithm” by James M. Keller, Michael R. Gray, and James A. Givens, JR. introduces the theory of fuzzy sets and incorporates the fuzzy search techniques into the KNN algorithm. The fuzzy algorithm outperforms the generic KNN in terms of mistake rate and holds up well against more conventional, complex pattern recognition techniques. The issue with the base KNN classifier is that each sample is given an equal weight. This creates issues in certain situations when sample sets overlap. Equal weight is given to typical vectors and the actual representatives of the cluster. Another issue is that when one of the vectors is classified, the membership strength cannot be determined. Hence the fuzzy set theory is introduced to combat all these issues. The fuzzy KNN classifier provides a unique kind of insight from the data as we can measure how "typical" a particular object is with respect to each class. In general, the membership assignments or rules generated for the training samples using fuzzy KNN have better characteristics than the base KNN in the sense that a data point that has been misclassified initially will not belong to any class. However, a correctly classified data point will have the membership with respect to its correct class almost equal to unity.

“PSOLDA: A particle swarm optimization approach for enhancing classification accuracy rate of linear discriminant analysis” by Shih-Chieh Chen and Shih-Wei Lin clearly showed an approach to improve the accuracy of LDA. LDA is a popular classification technique. We can create a classification model based on crucial weight details provided by it. Classification can become really harder based on real-world data sets. By using an example, this study demonstrates how feature selection affects the LDA. As problems contain more features, the conventional LDA approaches for identifying the useful feature subset are difficult or cannot ensure optimal outcomes. PSOLDA is a Particle Swarm Optimization based approach that helps us to increase the accuracy of LDA and helps us identify the important features of LDA. Particle swarm optimization is a meta-heuristic tool in artificial intelligence. Many open datasets are used to gauge PSOLDA's performance by calculating the classification accuracy rate. As a result, numerous heuristic techniques are employed, including feature selection based on Principal Component Analysis, backward feature selection, and forward feature selection. We observed that the accuracy rate of classification was higher in the case of PSOLDA when compared with other techniques. In other words, the PSOLDA method can be used to eliminate features from LDA that are not important or necessary, improving the classification results in general.

(LDA - linear discriminant analysis)

Basheer M. Al-Maqaleh and Hamid Reza Shahbazkia in their paper “Genetic Algorithm for Discovering Classification Rules in Data Mining” have shown the significance of identifying categorization rules in data mining in an exceptional way. Yet, it can be challenging to identify a classification rule because of the numerous potential rules and the high dimensionality of the data. For locating classifying criteria, the authors suggest using a genetic algorithm. A particular kind of optimization algorithm called a genetic algorithm takes its cues from the natural selection process. An initial population of potential solutions forms the basis of the algorithm. The classification rule, in this instance, is represented as a bitstring. A suitability function that gauges how effectively the rules classify the data is used to assess each prospective solution's acceptability. The best candidate is then chosen, and the algorithm repeats the process to create a new set of candidate solutions. Until the exit requirements are satisfied, this process is continued. The authors go over the specifics of how genetic algorithms are implemented, such as population initialization, selection, crossover and mutation operators, and termination criteria. The performance of the existing techniques was compared to the suggested genetic algorithm for class rule discovery, and it is assessed against a number of benchmark data sets. On some datasets, the authors discovered that genetic algorithms found superior classification rules and outperformed other approaches. A sensitive analysis was carried out to check the effects of various parameter values on the performance of the algorithm. The proposed genetic algorithm is a viable method for locating classification rules in data mining, according to the paper. Large data sets can be handled, and sophisticated rules can be recognized.

“On hyperparameter optimization of machine learning algorithms: Theory and practice” by LiYang and Abdallah Shami has shown that an important aspect of improving classifiers is hyperparameter tuning. Exhaustive space search and surrogate models are the two primary divisions of hyper-parameter optimization techniques. A search space needs to be established in advance for each of these methods. Exhaustive searches include Grid Search and Random Search. It typically necessitates a thorough understanding of hyperparameter optimization techniques and machine learning algorithms. Even if there are many automatic optimization techniques, depending on the issue, each has specific benefits and drawbacks. This article presents the analysis of hyper-parameter tuning for well-known machine learning models. In order to employ ML models for real-world problems, their hyperparameters should be changed. In this overview paper, we have explored in detail the most current results in the field of hyper-parameter optimization and how to theoretically and practically apply them to different machine learning models. When using optimization techniques on ML models, the major factor in selecting an HPO (hyper-parameter optimization) methodology is the hyperparameter types in the model. If the input dataset is well represented by randomly selected subsets, then it is recommended to optimize the ML model using BOHB (Bayesian Optimization HyperBand). When the hyperparameter configuration space is large then we should us If not, PSO (particle swarm optimization), and when it is small, BO (Bayesian optimization) models are recommended.

Fabian Pedregosa in his paper “Hyperparameter optimization with approximate gradient” has elegantly talked about how to optimize hyperparameter with approximate gradient. The two main subcategories of hyper-parameter optimization strategies are exhaustive space search and surrogate models. For each of these techniques, a search space needs to be constructed beforehand. Grid Search and Random Search are two types of exhaustive search. The majority of machine learning models have at least one hyperparameter to account for model complexity. The accurate choice of hyperparameters must be made, even when it is computationally challenging. We present a gradient-based method for the optimization of continuous hyperparameters in this study. This has the advantage of allowing hyperparameter updates before the parameters of the model converge. We also propose appropriate criteria for the global convergence of this method based on the regulatory requirements of the associated functions and the summability of errors. We validate the empirical performance of our strategy by estimating the regularization constants for kernel Ridge regression and 2-regularized logistic regression. Empirical benchmarks demonstrate how competitive our strategy is against modern methods. We observe that despite HOAG having a lower cost per iteration, Iterdiff and HOAG (Hyperparameter optimization with approximate gradient) perform identically. This is understandable given that, after each step taken by HOAG, the gradient is computed using the previous inner optimization problem solution as a warm start. This is not the case with Iterdiff because the computation of the gradient critically depends on the inner optimization algorithm having enough iterations.

Rafel Palliser-Sans(2021) has presented a way to optimize the RULES algorithm (Pham and Aksoy , 1995) by implementing a more effective way to find and remove irrelevant rules and simultaneously check for stopping conditions more often. RULES algorithm tries to find IF-THEN rules from the training dataset and classifies the testing dataset into different classes using these conditions. RULES algorithm aims for 100% precision in training datasets unless there are inconsistencies in data. So to find rules in the dataset RULES algorithm first checks if there is a single selector with 100% precision. If not, it fits combinations of one or more selectors by increasing them one at a time. RRULES algorithm builds upon RULES algorithm and removes irrelevant rules, which helps improve our algorithm's generalization and prediction accuracy(as RULES algorithm creates too many specific rules on the training dataset and it leads to overfitting of the model and hence an increase in the testing error of model which is not desirable). To do so, it checks whether the current combination of selectors we are currently trying to find rules on is already present in non-classified instances. RRULES algorithm also improves the stopping condition of RULES by checking the stopping condition after the creation of each rule. RRULES results show that it outperforms RULES by reducing a coverage rate by a factor of 7 and running 2-3 times faster than it.

KNN and apriori algorithms are useful for classification and association set rules, respectively. But they are not so efficient in large databases and have a high runtime. In traditional KNN, we classify new data points by finding K closest points to it in the training dataset and using their class labels to classify new points. Ritika Agarwal(2012) has proposed a modified version of the apriori algorithm to improve the efficiency and accuracy of KNN. Firstly our data is converted into numeric values from 0 to 1 and sorted for each attribute. Then a modified apriori algorithm is applied to our dataset. The modified apriori helps determine a few attributes that mainly define the class. In the modified apriori algorithm, we don’t scan the entire dataset to count the support of every attribute; instead, we store the count of minimum support and compare it with the help of every attribute. Modified apriori saves more time and space for finding associations than traditional apriori. Now, KNN is applied to the itemsets generated by our modified Apriori Algorithm rather than our original dataset. This is useful as finding distance between points in high dimensional space is expensive, so by reducing dimensions the algorithm becomes efficient. Experimental benchmarks on some datasets also show that this algorithm performs much better than traditional KNN, gives better accuracy, and reduces computational complexity. So this modification for KNN can be used on high-dimensional datasets efficiently.

Decision trees are widely used for classification and are easy to understand and interpret, making them very suitable for cases where model transparency is essential. The decision tree uses information gain to split the training dataset at each tree node and takes the split with a maximum information gain at each step. Information gain is an indicator of the decline of uncertainty. But traditional decision tree induction algorithms are unsuitable for large datasets as they tend to produce decision trees with many irrelevant or redundant rules, which can lead to overfitting on the training dataset. So to avoid this the paper “Extracting Useful Rules Through Improved Decision Tree Induction Using Information Entropy”, the author proposes a way to improve decision trees using information entropy to help with tree building. The algorithm proposed by the paper consists of three steps: attribute-oriented induction, relevance analysis and multi-level mining. The paper also introduces a HeightBalancePriority algorithm that merges nodes and balances the height of the decision tree based on priority checks at every node. This algorithm improves the efficiency and quality of the decision tree and avoids over-branching problems caused by unnecessary attributes. The paper uses modified DMQL queries to evaluate the classification rules generated by C4.5 and compares them with the proposed algorithm and found that the algorithm proposed by the paper can make a smaller decision tree on large datasets, which is able to generalize better and have good accuracy.

In a simple multinomial naive bayesian classifier for text classification, we assume each document is a vector of words, ignore the word order, and treat each word independently. To label a new document, we calculate the probability of each class given the observed words in the document and assign the document to the class with the highest probability. This algorithm is considered "naive" because it assumes that words are independent of each other, which is generally not valid in real life. Yirong Shen and Jing Jiang(2003) have proposed a new method to improve the naive Bayes classifier. To do so, they add some discriminative model components to the generative model of naive Bayes so that independent property assumption in naive Bayes can be counteracted. So firstly, we assume that the input document can be divided into components(based on properties) and give weight to each part of the document using some exponents. We do so so that the parts of the document that have a higher dependency on the label of the document can be better represented than other parts. Now to choose the weights for our algorithm, we pick them such that they maximize the log likelihood of training sets data labels. They also compared their algorithm with normal naive bayes on 20 newsgroup data sets of USENET news posting and found that their model has much higher accuracy than that of naive Bayes.

**3. CONCLUSION**

First, we learned about a cutting-edge method called Bayesian optimization. It uses a probabilistic model to show how hyperparameters affect the performance of the model. This approach is similar to the way search algorithms combine exploration and exploitation to find the best answer. Tree-structured Parzen estimators and Gaussian process-based techniques are examples of Bayesian optimization variations that look at the hyperparameter space and make use of several models to forecast the performance of the model.

We then learned about Bagging Predictors, which describes a machine learning method called bagging. It directly ties to the idea of search algorithms in artificial intelligence. To create a more accurate overall model, various models are integrated after being trained on smaller amounts of data, a process known as bagging. It offers a number of advantages, including decreasing model variance, but it also has drawbacks like sensitivity to outliers and computational complexity. This study demonstrates how experimenting with diverse options might produce superior outcomes, which is relevant to other industries that use AI-based technology.

We then learned about a unique method for dealing with classification difficulties called ACO. The ACO search algorithm is a metaheuristic one that takes its cues from the way that actual ant colonies operate. The method is a simulation of the way in which ant colonies employ a probabilistic decision-making process to locate the food source that is the most direct route to their colony. ACO-based classification can provide a manageable number of rules while yet retaining classification accuracy that is comparable to or even greater to that of the other techniques.

We then learned about a method for extracting lucid rules from Support Vector Machines. The method mines an SVM model for the most pertinent features and support vectors using a genetic algorithm. The extracted rules are represented as a collection of decision trees, where each node represents a choice made in response to a value of a feature. This method provides a promising route for making SVM models more transparent and understandable.

The paper “Rule extraction from Support Vector Machines: A Sequential Covering Approach” shows how sequential covering approaches and separate-and-conquer searches yield more relevant rules for interclass separation and how SQRex-SVM outperforms the standard SVM methods for rule extraction. The approach used for finding the rules in SQRex-SVM uses SVM information by considering only True Positive and True Negative points and searching for limited rules refined for both True Positive and True Negative. The paper concludes that the retrieved rules have excellent fidelity to the original SVM and outperform the SVM in accuracy, TP, and FP rates.

Continuing to learn the challenges of getting relevant rules for classification, we encountered A fuzzy KNN Algorithm that uses the Fuzzy sets theory and fuzzy search techniques with KNN to outperform the generic KNN in terms of mistake rate and holds up well against conventional, complex pattern recognition techniques. We saw how Fuzzy KNN combats the biggest issue with the base KNN classifier is that it gives equal importance to each sample. Another issue is that the membership strength cannot be determined when one of the vectors is classified. The fuzzy KNN classifier helps to distinguish the "typicalness" of a particular object with respect to some class.

Getting better rules for classifiers involves complex algorithms for rule searching, and hyperparameter tuning is also an important aspect. “Random Search for Hyper-Parameter Optimization” shows how random trials on a grid give as good models as complex, extensive searches with a fractional computational cost. When searching on a larger solution space with similar computational constraints, random search is more effective in finding better models as compared to extensive grid searches.

Also, we learned about the Support Vector Machines (SVM)-based feature selection method and the Tabu Search and Genetic Algorithms metaheuristic optimization techniques. The goal is to find the most distinctive features that reduce dimensionality and processing time while enhancing SVM performance. Natural selection is the foundation of genetic algorithms, whereas Tabu Search is a regionally specific metaheuristic search. The SVM-driven Tabu search that the authors propose uses SVM as an inductive method under the control of a Tabu algorithm-based engine.

We studied the newly modified RULES Algorithm, which is more effective in finding relevant rules and simultaneously checks for stopping conditions, making it way efficient both performance-wise and computationally. The RULES algorithm strives for 100% precision in training datasets but often creates too many specific rules, leading to overfitting and increased testing errors. The RRULES algorithm builds upon RULES by removing irrelevant rules and improving the stopping condition, resulting in better generalization and prediction accuracy.

Then we look over the limitations of traditional KNN and apriori algorithms in large databases and how the proposed algorithm by Ritika Agarwal uses a different style of apriori to improve the efficiency and accuracy of KNN. The modified algorithm converts data into numeric values. It applies a modified apriori algorithm to determine attributes that mainly define the class, saving time and space for finding associations and reducing the dataset's dimensions, making the KNN algorithm more efficient. The results depict that the newly changed algorithm outperforms the traditional KNN both in terms of accuracy and overall computational cost, which makes it suitable for high-dimensional datasets.

As we have already seen, decision trees are one of the most widely used classification techniques because of their high interpretability. However, decision trees tend to make orthogonal boundaries, resulting in high overfitting issues and redundant rules issues. To counter this drawback of decision trees, the paper "Extracting Useful Rules Through Improved Decision Tree Induction Using Information Entropy" proposes an algorithm that uses information entropy to improve decision trees. Furthermore, the paper introduces a HeightBalancePriority algorithm that merges nodes and balances the decision tree's height, improving its efficiency and quality while avoiding unnecessary attributes.

Naive Bayes is considered a primary classifier that has several limitations. We looked over the example of how naive Bayes improves performance in text classification. The multinomial naive Naive Bayes is a fairly simple algorithm for text classification. It assumes that each document is a vector of words and treats each word independently. The algorithm calculates the probability of each class given with respect to observed words and predicts the class of the document using these probabilities. This algorithm is considered naive because it assumes that words are not dependent on each other, which is generally false in real life. Shen and Jiang proposed a way to improve this algorithm by adding discriminative model components to the generative model of naive Bayes. This improves the algorithm's performance by countering the independent property assumption in naive Bayes.

We then learned about employing a genetic algorithm to find classifying criteria in data mining. An initial population of viable solutions serves as the foundation for this algorithm, which draws inspiration from the natural selection process. The acceptability of each potential solution is evaluated using a suitability function, and the algorithm iteratively repeats the procedure until the exit conditions are met. The algorithm's performance is evaluated against several benchmark data sets and compared to other methods for discovering class rules. Due to its ability to handle enormous data sets and identify complex rules, the suggested genetic algorithm is a practical way to find classification rules in data mining.

We then learned about PSOLDA which is a particle swarm optimization technique used to increase the precision of linear discriminant analysis (LDA).We have seen that the PSOLDA technique outperforms other assessed approaches in terms of classification accuracy rates. It may also be used to remove characteristics from LDA that are not crucial or required, hence enhancing classification outcomes overall.

We then learned about the examination of hyper-parameter tuning for well-known machine learning models and how to use them with machine learning algorithms. The types of hyperparameters in the model are a key consideration when choosing an HPO (hyper-parameter optimization) methodology. Particle swarm optimization (PSO) is frequently the best option for huge configuration spaces, whereas BOHB (Bayesian Optimization HyperBand) is advised for optimizing ML models. For utilization in the real world and scientific endeavors, HPO tools and frameworks are offered.

We then learned about a gradient-based method for continuous hyperparameter optimization. This method has the benefit of permitting hyperparameter updates before the model's parameters converge. It competes favorably with contemporary techniques, outperforming both Iterdiff and HOAG (Hyperparameter optimization with approximate gradient).

We will be implementing a comparative analysis of all the research papers we studied. We will also implement some of the standard classification techniques given in some of the papers that we read. We will try to improve the rules of classifiers using Genetic Algorithms with Heuristic Searches. We will also give a comparative analysis of random search, grid search, and fuzzy search algorithms. We will implement Particle Swarm Optimization and Ant Colony Optimization which are really good optimization techniques and compare how well they perform to search for the best set of rules that can classify the data accurately.

**4. TIMELINE**

***Please Note S-Start E-End***

# 

# 

# **REFERENCES / BIBLIOGRAPHY**

Martens, D., De Backer, M., Haesen, R., Vanthienen, J., Snoeck, M., & Baesens, B. (2007). Classification with ant colony optimization. IEEE Transactions on Evolutionary Computation, 11(5), 651-665.

Breiman, L. (1996). Bagging predictors. Machine learning, 24, 123-140.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. Advances in neural information processing systems, 25.

Núñez, H., Angulo, C., & Català, A. (2002, April). Rule extraction from support vector machines. In Esann (pp. 107-112).

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of machine learning research*, *13*(2).

Keller, J. M., Gray, M. R., & Givens, J. A. (1985). A fuzzy k-nearest neighbor algorithm. *IEEE transactions on systems, man, and cybernetics*, (4), 580-585.

Barakat, N. H., & Bradley, A. P. (2007). Rule extraction from support vector machines: A sequential covering approach. *IEEE Transactions on Knowledge and Data Engineering*, *19*(6), 729-741.

El Ferchichi, S., Laabidi, K., & Zidi, S. (2009). Genetic algorithm and tabu search for feature selection. *Studies in Informatics and control*, *18*(2), 181-187.

Al-Maqaleh, B. M., & Shahbazkia, H. (2012). A genetic algorithm for discovering classification rules in data mining. International Journal of Computer Applications, 41(18), 40-44.

Yang, L., & Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. Neurocomputing, 415, 295-316.

Pedregosa, F. (2016, June). Hyperparameter optimization with approximate gradient. In International conference on machine learning (pp. 737-746). PMLR.

Lin, S. W., & Chen, S. C. (2009). PSOLDA: A particle swarm optimization approach for enhancing classification accuracy rate of linear discriminant analysis. *Applied Soft Computing*, *9*(3), 1008-1015.

Palliser-Sans, R. (2021). RRULES: An improvement of the RULES rule-based classifier. arXiv preprint arXiv:2106.07296.

Shen, Y., & Jiang, J. (2003). Improving the performance of Naive Bayes for text classification. CS224N Spring.

Ali, M. M., Qaseem, M. S., Rajamani, L., & Govardhan, A. (2013). Extracting useful rules through improved decision tree induction using information entropy. arXiv preprint arXiv:1302.2436.

Agarwal, R., Kochar, B., & Srivastava, D. (2012). A novel and efficient KNN using modified apriori algorithm. International Journal of Scientific and Research Publications, 2(5), 2250-3153.