

Feature Selection Methods in Twin Support Vector Machines

by

Ruchita Dodiya
202111028

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of

MASTER OF TECHNOLOGY
in
INFORMATION AND COMMUNICATION TECHNOLOGY
to

DHIRUBHAI AMBANI INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGY

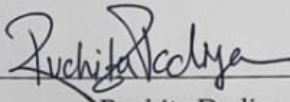


May, 2023

Declaration

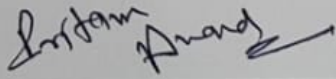
I hereby declare that

- i) the thesis comprises of my original work towards the degree of Master of Technology in Information and Communication Technology at Dhirubhai Ambani Institute of Information and Communication Technology and has not been submitted elsewhere for a degree,
- ii) due acknowledgment has been made in the text to all the reference material used.


Ruchita Dodiya

Certificate

This is to certify that the thesis work entitled **Feature Selection Methods in Twin Support Vector Machines** has been carried out by **Ruchita Dodiya** for the degree of Master of Technology in Information and Communication Technology at *Dhirubhai Ambani Institute of Information and Communication Technology* under my supervision.



Prof. Dr. Pritam Anand
Thesis Supervisor

Acknowledgments

First and foremost, I would like to thank my family for their unwavering support, understanding, and encouragement. Their love and belief in me have been a constant source of motivation and inspiration. My mother, Mrs. Sunitaben , father Mr. Pareshbhai and brother Mr. Bhavdeep who always supporting me and believing that I can achieve what I desire.

I would like to extend my heartfelt thanks to my supervisor, Prof. Dr. Pritam Anand, for their valuable guidance, support, and mentorship throughout this research project. Their expertise and insights have been instrumental in shaping the direction of my work, and I am truly grateful for their constant encouragement and feedback.

I am grateful to my panel members Prof. Manish Khare, Prof. Bhaskar Chaudhury, for their valuable time and insightful feedback during my stage presentations. I want to thank all the professors at DA-IICT who have contributed to my evolution. Their teaching provided good knowledge, which was helpful during the thesis work, and will be there with me as I grow.

I am grateful to my friends Maulik, Krutika, Juned, Divya, Nisarg, Hemani, Dhairya, Pavan, Malvi who helped me directly or indirectly emotionally, mentally, or physically in this journey.

Contents

List of Tables	vi
List of Figures	vii
1 Introduction	1
1.1 Support Vector Machine (SVM)	2
1.1.1 Introduction of SVM	2
1.1.2 Limitations of SVM	3
1.2 Twin Support Vector Machine (TWSVM)	4
1.2.1 Introduction of TWSVM	4
1.2.2 Advantage of TWSVM	4
1.3 Problem Statement	5
1.3.1 Features Selection	6
1.3.2 Parameter Tuning	7
1.4 Research Objective	7
1.5 Significance of the Study	8
2 Related Work	9
2.0.1 Filter Methods	9
2.0.2 Wrapper Methods	10
2.0.3 Embedded Methods	10
2.0.4 Hybrid Methods	11
2.1 Previous Study of features Selection using BGSA	11
2.2 Previous Study of features Selection using TLBO	12
3 Methods	14
3.1 Support Vector Machine(SVM)	14
3.2 Twin Support Vector Machine(TWSVM)	16
3.3 features Selection Methods	19
3.3.1 Binary Gravitational Search Algorithm (BGSA)	20

3.3.2	Teaching Learning Based Optimization (TLBO)	26
3.4	Parameter Tuning Methods	31
3.4.1	Grid Search	31
3.4.2	Simulated Annealing (SA)	33
4	Results	35
4.1	Description of Datasets	35
4.2	Parameter Setting	39
4.3	Results of Experiment	39
5	Conclusions	62
	References	64

Abstract

For the development of a machine learning model both parameter tuning and feature selection are necessary . The model's hyper parameters need to be tuned to achieve the best values because they have a significant impact on how well the model works and the objective of feature selection is to identify the most important subset of features that contribute to reliable predictions and model understanding.

The primary goal of this study is to examine the effectiveness of feature selection techniques when used Twin Support Vector Machines (TWSVM) and traditional Support Vector Machines (SVM). We want to determine that the feature selection technique results is the best performance increase for TWSVM and SVM by conducting extensive experiments on multiple datasets. The results of this study will give important information about how feature selection will improve the classification accuracy and effectiveness.

The methodology used in this study involves applying different kinds of parameter tuning and feature selection techniques for Support Vector Machines (SVM) and Twin Support Vector Machines (TWSVM) using linear and RBF kernels. We used a hybrid approach to parameter tuning and feature selection. Optimized the hyper parameters using the Grid Search and Simulated Annealing (SA) methods. Then, with SA-based parameter tuning, we combined the Binary Gravitational Search Algorithm (BGSA) and Teaching-Learning-Based Optimization (TLBO) for feature selection.

We use these techniques to enhance the performance of SVM and TWSVM models by tuning their parameters and selecting useful features. Our results show that feature selection methods are more effective at selecting relevant features while using less computation time in TWSVM compare to SVM.

List of Tables

4.1	About Datasets	38
4.2	Parameters of Methods	39
4.3	Results of Grid Search with Linear Kernel	40
4.4	Results of Grid Search with RBF Kernel	41
4.5	Results of SA with Linear Kernel	42
4.6	Results of SA with RBF Kernel	43
4.7	Selected Features of some datasets	44
4.8	Results of SA-BGSA with Linear Kernel	45
4.9	Results of SA-BGSA with RBF Kernel	46
4.10	Results of SA-TLBO with Linear Kernel	47
4.11	Results of SA-TLBO with RBF Kernel	48

List of Figures

3.1	Flow Chart of Population Based FS Methods	20
3.2	Flow Chart of BGSA Method	21
3.3	Flow Chart of TLBO Method	28
4.1	Iteration vs No. of Features	44
4.2	Comparison of Grid Search ,SA and SA-BGSA of SVM with Linear Kernel	49
4.3	Comparison of Grid Search ,SA and SA-BGSA of TWSVM with Linear Kernel	50
4.4	Comparison of Grid Search ,SA and SA-BGSA of SVM with RBF Kernel	51
4.5	Comparison of Grid Search ,SA and SA-BGSA of TWSVM with RBF Kernel	52
4.6	Comparison of Grid Search ,SA and SA-TLBO of SVM with Linear Kernel	53
4.7	Comparison of Grid Search ,SA and SA-TLBO of TWSVM with Linear Kernel	54
4.8	Comparison of Grid Search ,SA and SA-TLBO of SVM with RBF Kernel	55
4.9	Comparison of Grid Search ,SA and SA-TLBO of TWSVM with RBF Kernel	56
4.10	Comparison of BGSA with Linear Kernel	58
4.11	Comparison of BGSA with RBF Kernel	59
4.12	Comparison of TLBO with Linear Kernel	60
4.13	Comparison of BGSA with RBF Kernel	61

CHAPTER 1

Introduction

Machine learning plays a vital role in real life, particularly in fields that involve data analysis, pattern recognition, and prediction. Firstly, machine learning facilitates advanced data analysis by efficiently handling large and complex datasets. It involves the development of algorithms and models that enable computers to learn and make predictions or decisions without explicit programming. The ability of machines to automatically learn from data has revolutionized various industries, including healthcare, finance, marketing, and more.

Supervised learning and unsupervised learning are the two main categories for machine learning algorithms. In supervised learning, the input features and associated output labels or target values are both included in the training data. Based on the given labels, the model learns to map the input features to the desired output. Common applications of this kind of learning include classification and regression.

On the other side, unsupervised learning uses datasets without labelled outputs. Finding hidden patterns or structures in the data is the aim. Unsupervised learning is frequently used for clustering, dimension reduction, and anomaly detection. These methods can help in identifying groups or clusters of related data points and provide important information about the distribution of the underlying data.

There are numerous well-known machine learning algorithms that fall under these categories that have been successful in a variety of applications. For instance, decision trees classify or forecast using a hierarchical framework of rules. Support Vector Machines (SVM) identify the optimal hyperplane for classifying data points. Artificial neurons are arranged in interconnected layers in neural networks, which are modelled after the structure of the human brain and are capable

of learning complex patterns.

The algorithm that is used depends on the particular issue at hand as well as the type of data that is easily accessible. Every algorithm has its own advantages, constraints, and underlying assumptions. Some algorithms better at handling big datasets, while others better at high-dimensional data or handling classes with imbalances. Additionally, a number of variables like features selection, parameter tuning, and the quality of the training data affect an algorithm's effectiveness.

Among the various machine learning algorithms, Support Vector Machines (SVM) have gained significant popularity due to their effectiveness and versatility. SVM are particularly well-suited for complex problems with high-dimensional data. They can handle both classification and regression tasks. One of the key advantages of SVM is their ability to generalize well, meaning they can make accurate predictions on unseen data. This robustness against overfitting is crucial for achieving reliable and dependable results.

1.1 Support Vector Machine (SVM)

1.1.1 Introduction of SVM

The Support Vector Machine (SVM) algorithm, introduced by Vapnik in the 1990s, SVM [6] is a popular type of machine learning algorithm that is widely used for classification and regression tasks [2] [4] [1] [3]. SVM is particularly useful in cases where the data is non-linearly separable and a non-linear decision boundary is needed to classify the data accurately.

SVM work by finding the hyperplane that best separates the data into different classes. The hyperplane is chosen so as to maximize the distance between the closest data points of each class, also known as the margin. The data points closest to the hyperplane are known as support vectors and are used to define the decision boundary. This characteristic of SVM makes it a robust classifier, capable of handling complex decision boundaries and addressing the challenges of non-linearly separable data.

The main objective of SVM is to maximize the margin between the decision boundary and the nearest data points of each class. By maximizing this margin, SVM

aims to achieve better generalization performance, reducing the risk of overfitting and improving the ability to classify new, unseen samples accurately. The margin serves as a measure of the model's robustness and its ability to handle noise and variations in the data. Additionally, SVM can handle class imbalance issues by incorporating various techniques such as soft-margin regularization or adjusting class weights.

In addition to linear separation, SVM can handle nonlinear decision boundaries by employing kernel functions such as the polynomial, Gaussian radial basis function (RBF), sigmoid, or custom-defined kernels. These kernel functions allow SVM to capture complex patterns and relationships that might not be apparent in the original input space.

1.1.2 Limitations of SVM

SVM can run into problems when working with huge datasets, especially when there are more features than training samples. In such circumstances, SVM may underperform or have trouble determining the right decision boundary.

When one or more classes have significantly fewer samples than the others, SVM may be particularly sensitive to this imbalance. This imbalance may cause classification results to be biased in favour of the dominant class.

The kernel function and its associated parameters can have an impact on SVM. For the best performance, choosing the right kernel function and adjusting the parameters can be essential.

Particularly for large datasets and high-dimensional features spaces, SVM training can become computationally demanding in terms of memory and time requirements.

SVM look for the maximum-margin hyperplane that divides several classes. Finding a clear separation might be difficult, though, when working with noisy or overlapping data. SVM may struggle to function satisfactorily in some circumstances or may need more preparation or data modification.

To get around some of the drawbacks of conventional SVM, one option to investigate is twin support vector machines (TWSVM). TWSVM has a number of

characteristics that make it a potential option in some situations.

1.2 Twin Support Vector Machine (TWSVM)

1.2.1 Introduction of TWSVM

Twin Support Vector Machine (TWSVM) was proposed as an improvement over traditional SVM to address certain limitations. However, TWSVM also has a specific shortcoming related to the accuracy of points in the intersection of two planes.

Traditional SVM aims to produce a single hyperplane that maximises the margin between two classes. The categorization accuracy, however, could not be suitable for locations located in the area where the two planes overlap.

In order to get over this restriction, Jayadeva et al. [13] developed TWSVM in 2007. TWSVM tries to generate two nonparallel planes that are as far from the other class as possible and as close to one of the classes as possible. This helps in improving intersection region point accuracy.

TWSVM's formulation is relatively similar to that of regular SVM, however there is one key distinction. In TWSVM, two quadratic programming problems (QPPs) are solved, whereas in SVM, only one QPP is solved. The distribution of the data points ensures that the patterns from one class match the QPP constraints from the other class and vice versa. This data point distribution enables TWSVM to solve two smaller-sized QPPs rather than one huge QPP, leading to a faster calculation.

TWSVM is more computationally efficient than regular SVM since it solves two smaller QPPs rather than one bigger QPP.

1.2.2 Advantage of TWSVM

Proximal SVM based on Generalised Eigenvalues (GEPSVM) is the paradigm from which Twin Support Vector Machines (TWSVM) are formed. TWSVM's goal is to identify two nonparallel planes by resolving two associated SVM-type issues. TWSVM divides the task into two smaller quadratic programming issues

as opposed to the conventional SVM, which requires resolving a single, larger quadratic programming problem.

The computational complexity of the TWSVM training phase is greatly reduced by this method. In fact, TWSVM only requires a quarter of the computing power compared to standard SVM. TWSVM is a more effective and scalable alternative for SVM-based classification tasks because of its computational benefit, requiring only 1/4 the computer resources of regular SVM [6].

1.3 Problem Statement

Twin Support Vector Machines (TWSVM) is a well-known variation of the conventional SVM that has shown enhanced classification accuracy in a number of real-world applications [13] [6]. TWSVM's drawback, like with other conventional machine learning models, is that it requires a collection of informative and relevant features in order to produce precise predictions.

Unlike neural architectures, which can learn features representations directly from the data, TWSVM relies on a fixed set of supplied features. This means that the quality and relevance of the features used as input can significantly impact the performance of TWSVM.

One of the challenges in using TWSVM is the lack of an automated method for features extraction. features extraction involves transforming the original data into a new representation that highlights the most relevant information for the given task. While TWSVM itself does not provide a built-in mechanism for automated features extraction, researchers and practitioners often employ separate features selection or dimensionality reduction techniques to identify the most informative features prior to applying TWSVM.

By carefully selecting or extracting a strong set of features, TWSVM can overcome the limitations associated with fixed features sets and improve its classification accuracy.

In addition to that Twin Support Vector Machines (TWSVM) performance can be greatly improved through parameter adjustment, which is an essential component of machine learning. TWSVM, like other machine learning models, has

hyperparameters that need to be carefully selected to achieve the best possible performance.

1.3.1 Features Selection

In many situations in the real world, datasets often consist of a large number of features, not all of which may be informative or relevant for the given task. This abundance of features can introduce noise, increase computational complexity, and potentially lead to overfitting. Features selection methods provide a solution to these challenges by identifying the most relevant features that accurately capture the underlying patterns and relationships in the data while discarding redundant or noisy features.

Selection of features is the process of choosing a subset of features or variables from the original dataset. The aim is to reduce the dimensionality of the data by retaining only the most informative and discriminative features. By doing so, features selection helps improve model accuracy, enhance interpretability, reduce computational complexity, and mitigate the risk of overfitting.

The benefits of features selection are manifold. Firstly, it improves model performance by focusing on the most relevant features, thereby reducing the potential influence of irrelevant or noisy features. By discarding redundant features, features selection simplifies the model and enhances its interpretability, making it easier to understand and draw meaningful insights. Additionally, features selection can significantly reduce computational requirements, as models trained on a reduced set of features are more efficient to train, test, and deploy.

Features selection methods encompass various approaches, including filter methods, wrapper methods, and embedded methods. These techniques employ statistical measures, machine learning algorithms, or domain knowledge to evaluate and rank the importance of features. They consider various criteria such as features relevance, correlation with the target variable, and features redundancy.

By employing features selection techniques, researchers and practitioners can improve the efficiency, accuracy, and interpretability of machine learning models. These methods enable the extraction of the most informative features, allowing for more robust and reliable predictions while reducing the impact of irrelevant or noisy features.

1.3.2 Parameter Tuning

Parameter tuning involves selecting the values of hyperparameters, which are not learned from the data during training but are set by the user or model developer before the training process. These hyperparameters control various aspects of the model's behavior and performance. By tuning these hyperparameters, we aim to find the best combination of values that maximize the model's performance on a specific dataset or task.

Support Vector Machines (SVM) have several hyperparameters that need to be tuned for optimal model performance. Here are some of the key hyperparameters in SVM:

- **C Parameter (Cost):** The trade-off between increasing the margin and reducing the training error is managed by the C parameter, sometimes referred to as the regularisation parameter. While a bigger C value focused on correctly identifying training points but may result in a narrower margin, a smaller C value focuses on correctly classifying training points but may cause more training errors. It affects the degree of misclassification the model can tolerate.
- **Kernel Parameters:** If you are using a kernel-based SVM, such as the polynomial kernel or the radial basis function (RBF) kernel, there are specific parameters associated with each kernel. For example, in the polynomial kernel, you need to choose the degree of the polynomial, while in the RBF kernel, you need to determine the gamma parameter. These kernel parameters control the shape and flexibility of the decision boundary and influence the model's ability to capture non-linear relationships in the data.

1.4 Research Objective

The research objectives of features selection methods in Twin Support Vector Machines (TWSVM) can include:

- **Investigate the impact of features selection on TWSVM performance:** The objective is to examine how different features selection methods affect the classification performance of TWSVM. This involves comparing the accuracy of TWSVM with and without features selection.

- Explore the effectiveness of different features selection algorithms in TWSVM: Evaluation of the effectiveness of various features selection algorithms is the goal, such as BGSA and TLBO specifically in the context of TWSVM. This involves comparing the selected features subsets and the resulting classification performance.

1.5 Significance of the Study

The significance of studying features selection in Twin Support Vector Machines (TWSVM) can be summarized as follows:

- Improved Accuracy: features selection improves the classification accuracy of TWSVM models by selecting the most relevant features.
- Overfitting Reduction: TWSVM models with a large number of features can be prone to overfitting, where the model becomes too complex and fails to generalize well to unseen data. features selection helps in reducing overfitting by reducing the complexity of the model.
- Enhanced Generalization: By selecting informative features, features selection improves the generalization capability of TWSVM models, allowing them to perform well on unseen data.
- Interpretability: features selection provides insights into the important features, enhancing the interpretability of TWSVM models and facilitating better understanding of the classification process.

CHAPTER 2

Related Work

An overview of the literature and research papers on features selection in Twin Support Vector Machines (TWSVM) is provided in this section. The significance of features selection is discussed, along with improvements in features selection methods for Support Vector Machines (SVM) and how these methods have been used to TWSVM.

features selection has garnered significant attention in recent years, with numerous methods and approaches proposed to overcome the challenges and limitations of traditional techniques. The choice of an appropriate features selection method depends on the particular application and dataset characteristics, necessitating careful consideration of various factors, including the number of features and the correlation among them.

Different features selection methods employ various strategies to evaluate the relevance of features and select the most informative subset. Three different features selection techniques are available.

2.0.1 Filter Methods

Filter methods evaluate the intrinsic characteristics of features, such as their statistical properties or information content, without considering the specific learning algorithm. They are computationally efficient and independent of the classifier. Commonly used filter methods include statistical tests, correlation-based features selection, and information theory-based measures.

Statistical tests such as t-tests or analysis of variance (ANOVA) have been widely used in features selection. They assess the statistical significance of the relationship between each features and the target variable. For instance, in their work [12]

employed statistical tests to rank features based on their p-values.

Correlation-based features selection methods measure the relationship between features and the target variable. They evaluate the statistical dependency or correlation between each features and the target. One popular correlation-based measure is the Pearson correlation coefficient, as discussed in the work by Hall [11].

Information theory-based measures, such as mutual information or entropy, quantify the amount of information shared between a features and the target. Peng et al.[17] proposed a features selection method based on mutual information that considers the criteria of maximum dependency, maximum relevance, and minimum redundancy.

2.0.2 Wrapper Methods

Wrapper approaches train and test the learning algorithm on various features combinations in order to evaluate features subsets. The effectiveness of the learning algorithm is immediately incorporated into the features selection procedure. Although being computationally expensive, they frequently provide more precise features subsets for particular classifiers.

The concept of "wrappers" for features subset selection was introduced by Kohavi and John [14]. They explored the use of different search algorithms and evaluation criteria within the wrapper framework. Their work highlighted the importance of considering the interaction between the learning algorithm and the features subset during selection.

2.0.3 Embedded Methods

Embedded approaches take advantage of the unique characteristics of the learning algorithm through including features selection within the algorithm. These techniques choose features as the model is being trained, ensuring that the features chosen match the goals of the learning process.

Embedded methods have been particularly popular in the context of Support Vector Machines (SVM). Weston et al.[27] proposed a features selection method for SVM, where features selection is incorporated into the SVM training process. By leveraging the inherent properties of SVM, they achieved efficient and unified

features selection and classification

2.0.4 Hybrid Methods

In order to take use of their complimentary qualities, hybrid methods combine various features selection strategies. These techniques use filter and wrapper methods, among other combinations, in an effort to produce better features subsets.

One example of a hybrid features selection method is the work by Yu et al.[29]. They introduced a hybrid approach that combines a correlation-based filter with a wrapper technique. Their method utilizes a fast correlation-based filter solution to select a subset of features based on relevance and redundancy measures.

We emphasise the importance of features selection in machine learning and concentrate on how Support Vector Machines (SVM) are particularly affected by it. We talk about improvements made to features selection methods designed for SVM and look at how these methods might be used to Twin Support Vector Machines (TWSVM).

Enhancing the performance and interpretability of SVM models requires careful features selection. Several studies have concentrated on creating features selection techniques specifically designed for SVM. In order to increase classification accuracy, decrease overfitting, and improve the model's ability to generalise, these techniques aim to identify the most informative subset of features from the initial features space [10].

We used the Binary Gravitational Search Algorithm (BGSA) and Teaching Learning-based Optimisation (TLBO) to select features for Twin Support Vector Machines (TWSVM), which is the subject of this research. To demonstrate BGSA and TLBO's usefulness in features selection tasks, we give an outline of the related research that has been done on these two algorithms.

2.1 Previous Study of features Selection using BGSA

Yang and Honavar [28] introduced Genetic Algorithms (GA) as a features selection method, while Firpi and Goodman [7] utilized Particle Swarm Optimization,

and Diao and Sheng [5] employed Harmony Search for the same purpose. Zhang and Sun [30] applied Tabu Search in the context of features selection. It is worth noting that while various methodologies exist for tackling this problem, there is currently no consensus on which approach is superior to others. The suitability of different techniques may vary depending on the specific problem at hand and the domain of application.

In recent years, Rashedi et al. [20] introduced an innovative optimisation method called as Gravitational Search Algorithm (GSA). GSA has shown promise in finding solutions for both unimodal and multimodal functions. The algorithm is inspired by the law of attraction between masses, as described by Newtonian gravity. This law states that the force between particles in the universe is directly proportional to their masses and inversely proportional to the square of their distance. Rashedi et al. [21] also extended GSA to handle problems in multidimensional binary spaces and presented Binary GSA (BGSA), a Version of the original that has been modified slightly algorithm.

The introduction of GSA and its binary variant, BGSA, has opened up new avenues for optimization problems, showcasing their potential in various domains. These algorithms leverage the concept of gravitational attraction to guide the search process and discover optimal or near-optimal solutions. The successful application of GSA and BGSA in solving complex optimization problems has further fueled research interest in these algorithms.

2.2 Previous Study of features Selection using TLBO

Teaching Learning-based Optimization (TLBO) has emerged as a valuable approach for solving discrete and binary features selection (FS) problems. This method is known for its simplicity in implementation and the minimal number of control parameters it requires [25]. TLBO is based on the teaching and learning phenomena observed in human beings, where teachers influence the quality of results achieved by learners. More comprehensive information about TLBO can be found in previous studies [24].

Research in the literature has demonstrated that TLBO has outperformed existing metaheuristic methods when applied to both unconstrained and constrained

benchmark problems [18]. For instance, Shahbeig et al.[23] introduced a combination of teaching learning-based optimization and diffused Particle Swarm Optimization (PSO) to identify the minimal subset of genes associated with breast cancer with maximum accuracy. Various modifications of TLBO have also been explored in the available literature, which have exhibited improved performance compared to the original TLBO in terms of convergence rate.

TLBO's suitability for discrete and binary features selection problems, along with its successful application in different domains, highlights its potential as an effective optimization technique. Its advantageous characteristics, such as easy implementation and superior performance compared to other metaheuristic methods, make it a promising choice for features selection tasks. Researchers continue to investigate and enhance TLBO to further improve its convergence rate and overall performance.

CHAPTER 3

Methods

This section outlines the approach we took in our study to address the issue of parameter tuning and features selection in the context of machine learning. For features selection and parameter tuning, we used Support Vector Machines (SVM) and Twin Support Vector Machines (TWSVM) as classification algorithms. Additionally, we used the grid search and simulated annealing (SA) methods for parameter tuning and the binary gravitational search algorithm (BGSA) and teaching learning-based optimisation (TLBO) methods for features selection.

3.1 Support Vector Machine(SVM)

Let the patterns to be classified be denoted by a set of m row vectors $A_i (i = 1, 2, \dots, m)$ in the n -dimensional real space R^n , where $A_i = (A_{i1}, A_{i2}, \dots, A_{in})^T$. Also, let $y_i \in \{-1, 1\}$ denote the class to which the i^{th} pattern belongs. We first consider the scenario in which the patterns belonging to the two classes can be separated strictly linearly. Then, we need to determine $w \in R^n$ and $b \in R$ such that

$$A_i w \geq 1 - b \text{ for } y_i = 1 \text{ and } A_i w \leq -1 - b \text{ for } y_i = -1, \quad (3.1)$$

The plane described by

$$w^T x + b = 0 \quad (3.2)$$

lies midway between the bounding planes given by

$$w^T x + b = 1 \text{ and } w^T x + b = -1 \quad (3.3)$$

and separates the two classes from each other with margin of $1/\|w\|_2$ on each side. In other words, the margin of separation between the two classes is given by $2/\|w\|_2$. Here $\|w\|_2$ denotes the L_2 norm of a vector w . Data samples which lie on the planes given by (3.3) are termed as support vectors. The maximum margin

classifier, which is the standard SVM, is obtained by maximizing this margin and is equivalent to the following problem

$$(SVM1) \min_{w,b} \frac{1}{2} w^T w \quad (3.4)$$

subject to $A_i w \geq 1 - b$ for $y_i = 1$ and $A_i w \leq -1 - b$ for $y_i = -1$

When the two classes are not strictly linearly separable, there will be an error in satisfying the inequalities (3.1) for some patterns and we can modify (3.1) to

$$\begin{aligned} A_i w &\geq 1 - b \text{ for } y_i = 1 \text{ and } A_i w \leq -1 - b \\ &\text{for } y_i = -1, q_i \geq 0, i = 1, 2, \dots, m, \end{aligned} \quad (3.5)$$

where q_i denotes the error variable associated with the i^{th} data sample. In this case, the classifier is termed as a “soft margin” one, and it approximately classifies points into two classes with some error. The classification of a given test sample x is obtained by determining the sign of $w^T x + b$. The soft margin depends on the value of the non negative error variables q_i . In this case, one needs to choose a trade-off between the margin and the error and the standard SVM formulation for classification of the data points with a linear kernel is given by

$$(SVM2) \min_{w,b,q} c e^T q + \frac{1}{2} w^T w \quad (3.6)$$

subject to

$$\begin{aligned} A_i w + q_i &\geq 1 - b \text{ for } y_i = 1, \\ A_i w - q_i &\leq -1 - b \text{ for } y_i = -1, \\ q_i &\geq 0, \quad i=1,2,\dots,m. \end{aligned}$$

Here, c denotes a scalar whose value determines the trade-off, a larger value of c emphasizes the classification error, while a smaller one places more importance on the classification margin.

The application of kernels in the context of Support Vector Machines (SVMs) is an essential method to solve nonlinear classification issues. Kernels allow an implicit transformation rather than directly transferring the input data to a higher-dimensional features space. SVMs can successfully capture complicated patterns and perform nonlinear separation in the original input space by using kernels. This method is advantageous since it saves the extra computational work required for explicit features mapping[9].

3.2 Twin Support Vector Machine(TWSVM)

In our study, we employed a novel method called Twin Support Vector Machines (TWSVM) for SVM classification. TWSVM differs from traditional SVMs in its approach to generating nonparallel planes that cluster the data points of each respective class. Although TWSVM utilizes a distinct formulation, each of the two quadratic programming problems in the TWSVM pair follows the general structure of a typical SVM. However, it is important to note that the constraints of neither problem contain all of the patterns simultaneously. This unique formulation and the use of nonparallel planes distinguish TWSVM as an innovative approach for SVM classification.

The TWSVM classifier is obtained by solving the following pair of quadratic programming problems

$$(TWSVM1) \quad \min_{w^{(1)}, b^{(1)}, q} \frac{1}{2} (Aw^{(1)} + e_1 b^{(1)})^T (Aw^{(1)} + e_1 b^{(1)}) + c_1 e_2^T q \quad (3.7)$$

$$\text{subject to } (Bw^{(1)} + e_2 b^{(1)}) + q \geq e_2, q \geq 0 \text{ and,}$$

$$(TWSVM2) \quad \min_{w^{(2)}, b^{(2)}, q} \frac{1}{2} (Bw^{(2)} + e_2 b^{(2)})^T (Bw^{(2)} + e_2 b^{(2)}) + c_2 e_1^T q \quad (3.8)$$

$$\text{subject to } (Aw^{(2)} + e_1 b^{(2)}) + q \geq e_1, q \geq 0$$

where $c_1, c_2 > 0$ are parameters and e_1 and e_2 are vectors of ones of appropriate dimensions.

The Twin Support Vector Machines (TWSVM) algorithm aims to classify points by determining two hyperplanes, one for each class, and assigning points based on their proximity to these hyperplanes. The objective function of TWSVM consists of two terms.

The first term in the objective function corresponds to the sum of the squared distances from the hyperplane to the points of one class. Minimizing this term encourages the hyperplane to remain close to the points belonging to that specific class, such as class 1.

The constraints in TWSVM require that the hyperplane maintains a minimum distance of 1 from points of the other class, for example, class -1. To measure the error

when the hyperplane is closer than this minimum distance, a set of error variables is introduced. These error variables quantify the misclassification of points from class -1.

The second term in the objective function aims to minimize the sum of these error variables, thereby reducing the overall misclassification due to points belonging to class -1.

By minimizing the objective function and optimizing the placement of the hyperplanes, TWSVM seeks to effectively classify points based on their proximity to the respective hyperplanes and minimize misclassification between the classes.

In TWSVM, the data points of one class are observed to cluster around the plane defined by the equation $x^T w^{(1)} + b^{(1)} = 0$, while the data points of the other class cluster around the plane defined by $x^T w^{(2)} + b^{(2)} = 0$ in TWSVM2. Comparing the performance of TWSVM to a standard SVM, it has been observed that TWSVM is approximately four times faster on average.

The improved efficiency of TWSVM can be attributed to the fact that it solves two problems, specifically equations (3.7) and (3.8), each of which is roughly of size $m/2$. In contrast, the standard SVM typically has a complexity of no more than m^3 . As a result, the runtime ratio between TWSVM and the standard SVM is approximately reduced, leading to faster computation time.

Thus, the ratio of run-times is approximately

$$(m^3) / (2 * (\frac{m}{2})^3) = 4$$

The Lagrangian corresponding to the problem TWSVM1 (3.1) is given by

$$L(w^1, b^1, q, \alpha, \beta) = \frac{1}{2}(Aw^{(1)} + e_1 b^{(1)})^T (Aw^{(1)} + e_1 b^{(1)}) + c_1 e_2^T q - \alpha^T (-(Bw^{(1)} + e_2 b^{(1)}) + q - e_2) - \beta^T q, \quad (3.9)$$

Lagrange multipliers. The Karush-Kuhn-Tucker (K.K.T) necessary and sufficient optimality conditions [15] for (TWSVM1) are given by

$$A^T (Aw^{(1)} + e_1 b^{(1)}) + B^T \alpha = 0, \quad (3.10)$$

$$e_1^T(Aw^{(1)} + e_1b^{(1)}) + e_2^T\alpha = 0, \quad (3.11)$$

$$C_1e_2 - \alpha - \beta = 0, \quad (3.12)$$

$$-(Bw^{(1)} + e_2b^{(1)}) + q \geq e_2, \quad q \geq 0, \quad (3.13)$$

$$\alpha^T(-(Bw^{(1)} + e_2b^{(1)}) + q - e_2) = 0, \quad \beta^Tq = 0, \quad (3.14)$$

$$\alpha \geq 0, \quad \beta \geq 0 \quad (3.15)$$

Since $\beta \geq 0$, from (3.12) we have

$$0 \leq \alpha \leq c_1 \quad (3.16)$$

Next, combining (3.10) and (3.11) leads to

$$[A^T e_1^T] [A \ e_1] [w^{(1)}, b^{(1)}]^T + [b^T e_2^T]\alpha = 0, \quad (3.17)$$

We Define

$$H = [A \ e_1], \quad G = [B \ e_2] \quad (3.18)$$

and the augmented vector $u = [w^{(1)}, b^{(1)}]^T$. With these notations, (3.17) may be rewritten as

$$H^T H u + G^T \alpha = 0, \quad i.e., \quad u = -(H^T H)^{-1} G^T \alpha. \quad (3.19)$$

$H^T H$ is always positive semi-definite, but in some circumstances it might not be well conditioned. Similar to the regularisation term used in Ridge Regression methods as those in [22], we introduce a regularization term $\epsilon I, \epsilon > 0$, to take care of problems due to possible ill-conditioning of $H^T H$. Here, I is an identity matrix of appropriate dimensions. Therefore, (3.19) gets modified to

$$u = -(H^T H + I)^{-1} G^T \alpha. \quad (3.20)$$

However, in the following, we shall continue to use (3.19) with the understanding that, if need be, (3.20) is to be used for the determination of u .

Using (3.9) and the above K.K.T conditions, we obtain the Wolfe dual [15] of TWSVM1 as follows:

$$(DTWSVM1) \max_{\alpha} \quad e_2^T \alpha + \frac{1}{2} \alpha^T G (H^T H)^{-1} G^T \alpha \quad (3.21)$$

$$subject \ to \ 0 \leq \alpha \leq c_1$$

Similarly, we consider TWSVM2 and obtain its dual as

$$(DTWSVM1) \max_{\alpha} e_1^T \gamma + \frac{1}{2} \gamma^T P (Q^T Q)^{-1} P^T \gamma \quad (3.22)$$

subject to $0 \leq \gamma \leq c_2$

Here, $P = [A \ e_1]$, $Q = [B \ e_2]$ and the augmented vector $v = [w^{(2)}, b^{(2)}]^T$, which is given by

$$v = (Q^T Q)^{-1} P^T \gamma. \quad (3.23)$$

In the above discussion, the matrices $H^T H$ and $Q^T Q$ are matrices of size $(n+1) * (n+1)$, where, in general, n is much smaller in comparison to the number of patterns of classes 1 and -1.

Once vectors u and v are known from (3.20) and (3.23), the separating planes

$$x^T w^{(1)} + b^{(1)} = 0 \quad \text{and} \quad x^T w^{(2)} + b^{(2)} = 0 \quad (3.24)$$

are obtained. A new data sample $x \in R^n$ is assigned to class r ($r=1,2$), depending on which of the two planes given by (3.24) it lies closest to, i.e.,

$$x^T w^{(r)} + b^{(r)} = \min_{l=1,2} |x^T w^{(l)} + b^{(l)}| \quad (3.25)$$

where $|\cdot|$ is the perpendicular distance of point x from the plane $x^T w^{(l)} + b^{(l)} = 0$, $l = 1, 2$.

From the Karush-Kuhn-Tucker conditions (3.10), (3.11), (3.12), (3.13), (3.14), (3.15), and (3.16), we observe that patterns of class -1 for which $0 < \alpha_i < c_i$ $i = (1, 2, \dots, m_2)$ lie on the hyperplane given by $x^T w^{(1)} + b^{(1)} = 0$. Using inspiration from traditional SVM, we can describe these patterns of class -1 as support vectors of class 1 with regard to class -1 because they are crucial in identifying the necessary plane. For the issue TWSVM2, a same finding is true.

3.3 features Selection Methods

In this section, we explore and analyze different population-based features selection methods such as Binary Gravitational Search Algorithm(BGSA) and Teaching Learning Based Optimization (TLBO) methods .

Population-based features selection techniques provide a different strategy for addressing these problems. These techniques make use of a population of features subsets that go through iterative search and optimisation. They are inspired by the concepts of population dynamics and evolutionary computation. The quality of the features subsets is assessed using performance measures or fitness functions, and they are treated as individuals within the population.

We used the Binary Gravitational Search Algorithm (BGSA) and Teaching Learning-

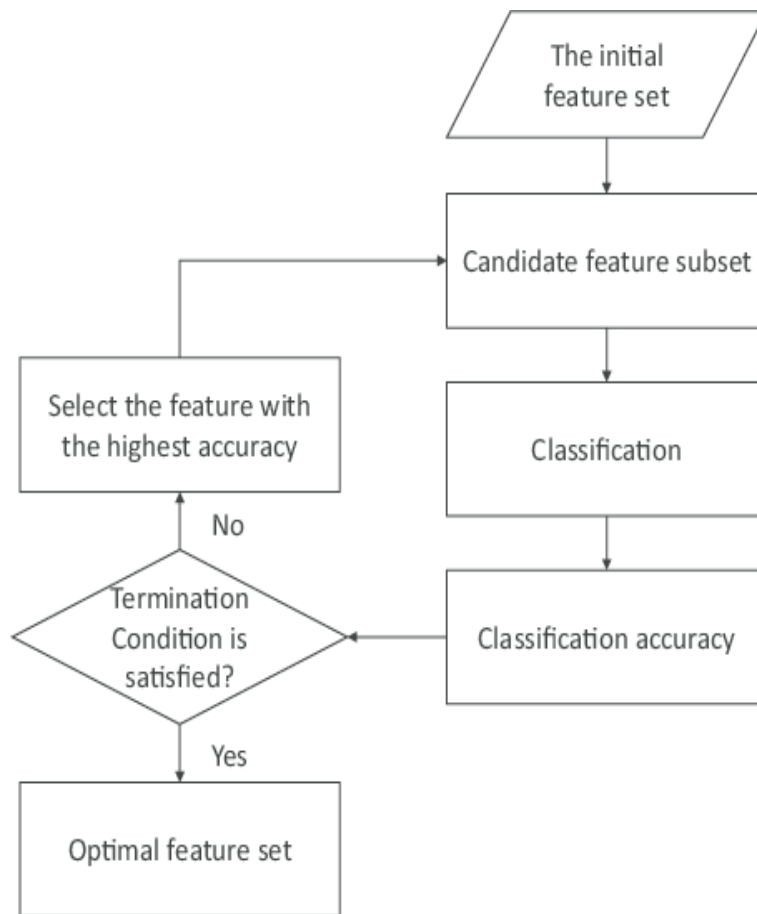


Figure 3.1: Flow Chart of Population Based FS Methods

based Optimisation (TLBO), two population-based optimisation techniques, in our features selection procedure. These techniques run by iteratively improving a population of features subsets according to evolutionary principles.

3.3.1 Binary Gravitational Search Algorithm (BGSA)

The Binary Gravitational Search Algorithm (BGSA) [8] is a metaheuristic optimization algorithm that draws inspiration from Newton's law of gravitation. It operates on a binary representation of the search space and employs concepts

such as gravitational force, movement, velocity, position update, mutation, and selection to explore and find the optimal solution. The BGSA algorithm has shown effectiveness in solving features selection problems, where the goal is to identify the most relevant subset of features from a larger set. By leveraging the principles of gravitation, the BGSA algorithm offers a promising approach to address features selection challenges and optimize the search for the best features subset.

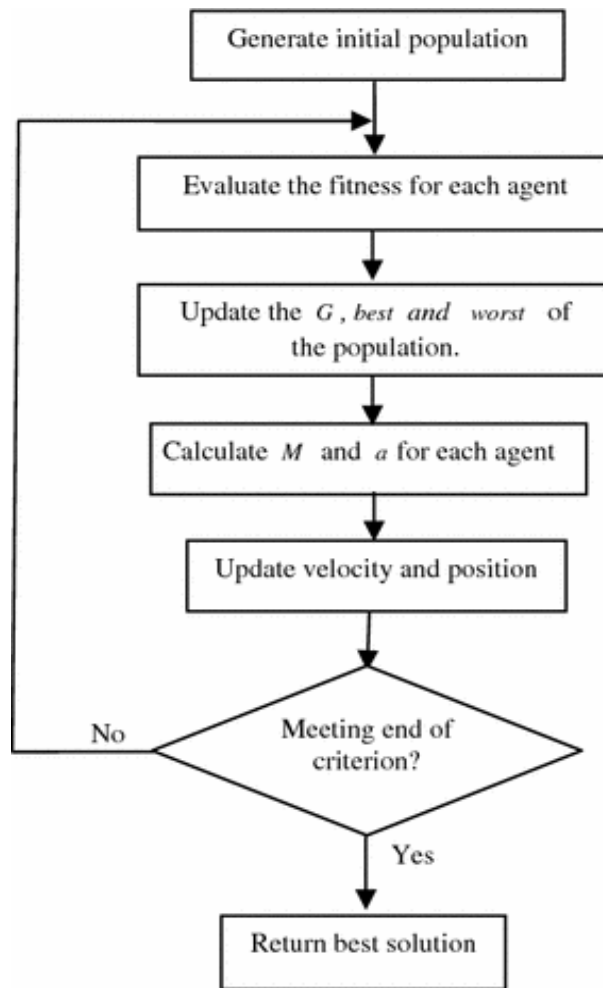


Figure 3.2: Flow Chart of BGSA Method

The particles or agents are encoded using binary strings of length m in the Binary Gravitational Search Algorithm (BGSA) [20]. Each component of the binary string indicates the inclusion or exclusion of the relevant features in the features subset. A value of "1" denotes the presence of the features, whereas a value of "0" denotes its absence. m , which is equal to the length of the binary strings, represents the total number of features in the population. The BGSA algorithm aims to identify an ideal features subset that maximises the objective function or fitness measure

by manipulating the binary strings and using gravitational principles.

Any wrapper-based algorithm's fitness function The fitness value of the particle (the features selected by the particle) is the capacity to predict the classes, or the classification accuracy of the features set provided to the classifier. FS is typically a learning model or a classifier. We used TWSVM in this report to find the fitness values. Masses are how BGSA refers to its potential solutions. As a result, each mass has a fitness value, and BGSA aims to maximise these values. Using Eq. (3.26), the mass of the i th particle at time t ($mass_i(t)$) is determined based on the fitness values.

$$mass_i(t) = \frac{fitness_i(t) - minimum(t)}{maximum(t) - minimum(t)} \quad (3.26)$$

The population's maximum fitness value ($max(t)$) and minimum fitness value ($min(t)$) are used to normalize the masses. After one unit of time (t), the masses are recalculated. By subtracting the value of $mass_i(t)$ from the sum of all masses, as indicated in Eq. (3.27) where n is the total number of particles, one can determine the value of $mass_i(t)$. In BGSA, the masses stand in for a particle's goodness. In order to improve (increase) the masses allotted to each candidate as t increase .

$$Mass_i(t) = \frac{mass_i(t)}{\sum_{j=1}^n mass_j(t)} \quad (3.27)$$

Other masses are forced by one mass, where the force of the j^{th} particle on the i^{th} particle $F_{ij}(t)$ is an m -dimensional vector. For each features, there is a component in the force. The force applied to the k^{th} features is stored in the vector's k^{th} position. As a result, $F_{ij}^k(t)$ represents the force that the j^{th} particle applies to the k^{th} features of the i^{th} particle and is determined using Eq. (3.28). The distance between two particles is expressed as the hamming distance ($dist(x_i, x_j)$), where x_i and x_j signify the current positions of the i^{th} and j^{th} particles in the search space, respectively.

$$F_{ij}^k(t) = G(t) * \frac{Mass_i(t) * Mass_j(t)}{dist(x_i, x_j)} * (x_j^k(t) - x_i^k(t)) \quad (3.28)$$

Where the gravitational constant $G(t)$, is determined using Eq. (3.29).

$$G(t) = \frac{\Omega * t}{E^{totaltime}} \quad (3.29)$$

Where t is the current time and total time is the total amount of time the algorithm can run, Ω is assumed to have a value of 20.

Using Eq. (3.30), the net force on the k^{th} features of the i^{th} particle, denoted as $F_i^k(t)$, is determined. In this case, $random_j$ is a random number between 0 and 1, inclusive.

$$F_i^k(t) = \sum_{j=1, j}^m random_j * F_{ij}^k(t) \quad (3.30)$$

According to the laws of physics, the force applied on the i^{th} particle has an impact on its k^{th} features, which is its velocity. Eq. (3.31) is used to compute the new velocity.

$$v_i^k(t+1) = random_j * v_i^k(t) + \frac{F_i^k(t)}{Mass_i(t)} \quad (3.31)$$

In the context of features selection, the velocity of a features in a particle denotes the likelihood of changing the state of that features, i.e., whether it is included in the features subset (expressed by '1') or excluded (represented by '0'). A features's higher velocity suggests that it has to be altered in order to make the particle perform better because it is different from the states of the better particles at that moment.

Equation (3.32) is used to determine the probability of changing a features's state and incorporates the velocity values. Based on the velocity values assigned to each features, its probability is determined. The state of the associated characteristic is reversed, or from "1" to "0," if a randomly generated number is smaller than the computed probability. It is important to note that the sigmoid function is frequently utilised in this stage. The tanh function, which has advantages and

ignores those drawbacks, is employed in place of the sigmoid function to address the problems mentioned in the sigmoid function in [26].

Because of BGSA's quick convergence rate, the algorithm may occasionally become stuck in local optima. In addition, a particle with a high fitness value attracts other particles towards it by applying a powerful pull on them. As a result, the features sets of all particles generally converge. The ability of BGSA to explore and find various, potentially better solutions is limited by this convergence.

The key to solving this problem is choosing an initial population that has an even distribution over the search space and sufficiently diverse. The algorithm can examine a wider range of features combinations and increase its chances of finding superior answers by starting with an expanded collection of candidate solutions. The diversity of the initial population encourages research and slows convergence to poor solutions.

$$probability = \tanh(v_i^k(t + 1)) \quad (3.32)$$

The BGSA algorithm uses a clustering procedure to prevent candidate solutions being convergent too early. The initial population is produced as a collection of p random particles. Equation (3.33) is used to calculate the n number of clusters. The clustering procedure starts by creating n cluster centers at random, each of which represents a potential solution. Then, Equation (3.34) is used to determine how similar each particle is to the cluster centers. There are two terms which make into the similarity metric. The inverse of the Hamming distance between a particle and a cluster center make up the first term. This phrase describes the ease with which a particle can be included or excluded from a cluster center. The inverse of the accuracy gap between a cluster center and a particle is the second term. It displays how well the particle and cluster center perform categorization tasks similarly. To compute the total similarity of a particle to all the cluster centers, these terms are merged with the proper weightings. Particles can be assigned using this method to the cluster center that shares the most similarities with them. The approach encourages diversity and exploration, preventing premature convergence, by clustering the particles and taking into account both their features similarity and classification performance.

$$n = \lfloor \alpha * p \rfloor ; \quad (3.33)$$

$$S_i = \beta * (1/H_d) + \eta * (1/D_a) \quad (3.34)$$

In this equation, α is a value between 0 and 1, p is the number of particles used to create the initial population, H_d stands for the hamming distance, D_a represents the accuracy gap (between the particle and cluster center), and S_i reflects how similar a cluster centre is to an agent. The two terms β and η in the expression for S_i , where $\beta = 1$, η denote the weighting of the two terms. A classifier is utilised to determine the classification accuracy of each particle and cluster centre in order to determine D_a . The classification accuracy differences between each particle and each cluster centre are then used to calculate D_a .

Now, a set of particles have been assigned to each cluster. Generating a features set from each cluster is the next stage. It makes sense to infer that increased accuracy results from better features. Let's say there are q particles in the cluster. The features of the particles of that cluster are analysed with the d^{th} cluster in mind in order to choose the most advantageous ones. It is suggested that Eq. (3.29) be used to calculate the goodness factor (h_i^d , which measures how good the i^{th} characteristic of the d^{th} cluster is). The goodness values of the best features must be higher than the mean of all goodness measures in a cluster.

$$h_i^d = \sum_{j=1}^q k_{ji}^d * Acc_j^d \quad (3.35)$$

where the position of the j^{th} particle in the d^{th} cluster is given by k_{ji}^d , and the accuracy of the j^{th} particle in the d^{th} cluster is given by Acc_j^d .

The following steps can be used to explain the BGSA algorithm:

1. Initialization: Start by randomly initializing a population of candidate solutions. Each candidate solution is represented by a binary string, where each binary digit represents the inclusion or exclusion of a specific features.
2. Fitness Evaluation: Evaluate the fitness of each candidate solution using a problem-specific fitness function. The fitness function measures how well a candidate solution solves the optimization problem at hand.
3. Gravitational Force Calculation: Calculate the gravitational force acting on each candidate solution based on its fitness. The fitness is treated as the mass of the solution, and the distance between the solution and the optimal solution (target) is used as the distance between two masses in Newton's gravitational force equation.

4. Movement Calculation: Determine the movement of each candidate solution based on the gravitational force acting on it. The movement is computed using a modified version of Newton's second law, where the acceleration is replaced by the gravitational force and the mass is replaced by the fitness.
5. Velocity Calculation: Calculate the velocity of each candidate solution based on its movement. The velocity is determined using a modified form of Euler's equation, where the acceleration is replaced by the movement.
6. Position Update: Update the position of each candidate solution based on its velocity. The position update is computed using a modified version of Euler's equation, where the velocity is treated as the differential of the position.
7. Mutation: Introduce diversity into the population by applying a mutation operator to some of the candidate solutions. The mutation operator randomly flips certain binary digits in a candidate solution, potentially altering its features selection.
8. Selection: Select the candidate solutions with the highest fitness values use for the next generation of solutions. These selected solutions will undergo further iterations of the algorithm.

After completing a specified number of iterations or reaching a termination condition, the algorithm produces the best-fit solution matrix, which represents the optimal features subset. Based on this best-fit solution, relevant features are selected, and classification or other problem specific tasks can be performed using the selected features.

3.3.2 Teaching Learning Based Optimization (TLBO)

In order to solve numerical optimisation problems, Rao et al. [19] presented a novel intelligence method called Teaching Learning-based Optimisation (TLBO). The population is treated as a group of students (nPop) in this method, and the best possible solutions to optimisation issues are taken into account as a teacher from the class as a whole. Through student collaboration and knowledge sharing, the TLBO algorithm tries to identify the greatest learner. To create superior results in terms of grades or marks, the TLBO algorithm's behaviour depends on the highly intelligent learners. They gain knowledge via the neighbour learning process. In Eq. (3.36), the entire population is represented as a vector.

$$X_{i,k} = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,k} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ x_{i,1} & x_{i,2} & \dots & x_{i,k} \end{bmatrix} \quad (3.36)$$

In this formula, i stands for the number of populations, k for the problem's dimension, and $x_{i,k}$ for the learner's position in the k^{th} dimension. The initialization of the learner X in the search space is random. The following Eq. (3.37) uses randomness to create $X_{i,k}$'s evolution.

$$X_{i,k} = L_k + r_1 * (U_k - L_k) \quad (3.37)$$

Where $i = 1, 2, 3, \dots, nPop$, $k = 1, 2, 3, \dots$. The uniform random numbers between 0 and 1 are denoted by r_1 . The lower bound value is denoted by L_k and the upper bound value by U_k . Two phases make up the TLBO procedure. The student can learn from the teacher during the teacher phase, and during the learner phase, they can learn through the interactions between the groups of students.

The following parts provide an explanation of the TLBO method's basic methodology:

Teacher phase :

A good teacher tries to bring his students up to his level of understanding during this stage. In actuality, however, this is not feasible, and a teacher may only, to some extent, depending on the ability of the students, increase the average level of knowledge in the class. In the case when $k = 1, 2, \dots, D$, let $M_{i,k} = (1/nPop)(\sum X_{i,k})$ be the mean value of the particular topic. Eq (3.38) gives a description of the updating equation for the process.

$$\begin{aligned} X_{i,k}^{new} &= X_{i,k}^{old} + r_2 * (X_{teacher,k} - T_f * M_{i,k}) \\ T_f &= round[1 + rand(0, 1)] \end{aligned} \quad (3.38)$$

Here, $X_{teacher,k}$ is the adopted population's best learner at the current algorithm iteration, r_2 is a random number between [0,1], and the value of T_f is taken to be considered as either 1 or 2.

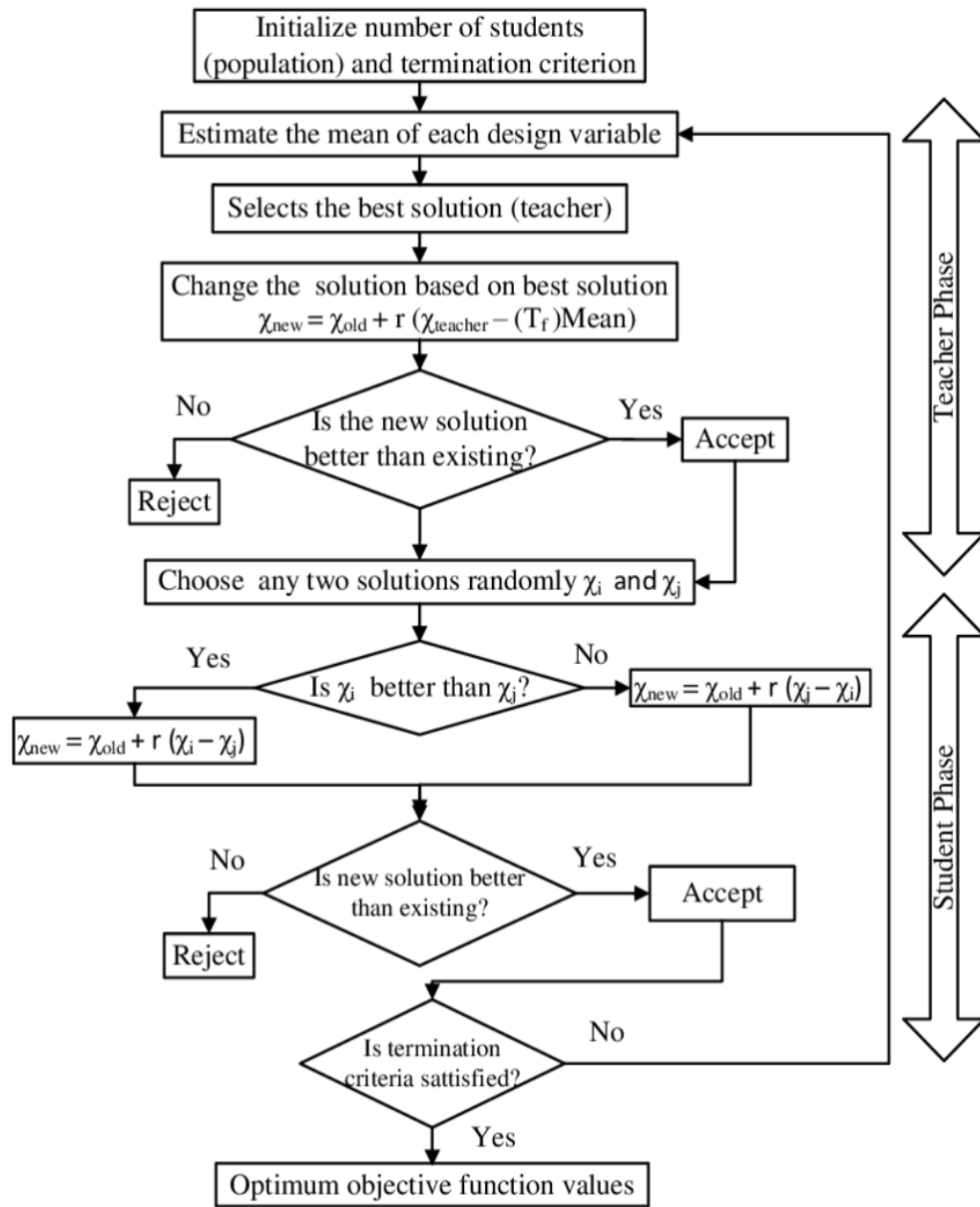


Figure 3.3: Flow Chart of TLBO Method

Learner phase :

Learner phase is the second phase of TLBO. During this phase, learners communicate with one another and with the teacher to expand their knowledge. Each student wants to randomly interact with other students to improve their communication skills. To put it another way, the objective is to choose the i^{th} learner as X_p and a different random learner as $X_q (p \neq q)$ through the mutual interaction with learners. Eqs (3.39) and (3.40) can be used to describe the updating equation for the i^{th} learners X_p and X_q throughout the learning phase.

Pseudo Code :

```
for i = 1 to nPop
  // pick two students, Xp and Xq.
  if (f(Xp) < f(Xq)) then
    newX_i = oldX_i + r * (X_p - X_q) // Equation (3.39)
  else
    newX_i = oldX_i + r * (X_q - X_p) // Equation (3.40)
  end if
end for
```

Equations used in pseudo code :

$$newX_i = oldX_i + r * (X_p - X_q) \quad (3.39)$$

$$newX_i = oldX_i + r * (X_q - X_p) \quad (3.40)$$

The best answers provided by the students X_p and X_q are $f(X_p)$ and $f(X_q)$, respectively, where r stands for a random number between [0,1]. A good learner communicates from a learner group that is $nPop$, taking the size of the learner population into consideration. These two occurrences continue to move the TLBO algorithm in the direction of good search regions with high-quality returns.

Binary TLBO

Because position vectors with a true continuous domain are present in this method, people travel around the search space constantly. In order for the search agents to move around in binary search space, a better velocity updating mechanism is needed. Due to this, we have provided a binary variant of TLBO known as BTLBO. A transfer function is also necessary, according to [16], to change the learner's position in relation to its velocity. Discrete values can be converted into

binary values using transfer functions. The processes for choosing the features that the BTLBO algorithm supports. A transfer function must offer a significant chance of changing the position to significantly raise the velocity.

Teacher Phase: During the teacher phase, the student's knowledge increases in connection with the teacher, who works to improve the average student's ability. Each learner's speed is estimated during this phase using Eq. (3.41)

$$V_i^{k+1} = r * (X_{teacher,k} - T_f * M_{i,k}) \quad (3.41)$$

Where $M_{i,k}$ indicates the mean value of all learners, T_f is taken into consideration as either 1 or 2, $X_{teacher,k}$ is the best learner of the approved population at the current iteration of the algorithm, and r specifies a random number in the range [0,1].

Learner Phase: During this stage, each learner develops their knowledge based on their interactions with other learners. Each learner's move will be calculated during this phase using Eq. (3.42):

$$V_i^{k+1} = r * (X_p - X_q) \quad (3.42)$$

Where X_p and X_q are two learners chosen at random by mutual interaction, and r is a random number between [0,1].

The approach uses a floating-point vector for velocity but a binary vector for position. When one of the learner positions can be modified, velocity is used to look for possibilities for a change from zero to one or one to zero.

The sigmoidal function, which is used to normalize the velocity between [0, 1] is the most widely used activation function in the literature that is currently available. Its drawbacks, however, include a requirement for larger movement based on the preceding location and a lack of differentiation between important values in V_i, d in the positive and negative directions. New transfer functions to the component of velocity were introduced to address this issue. Eq. (3.43) shows the V-shaped transfer function.

$$T(V_i^k) = \exp(|(V_i^{k+1} - a)/(1 + b)|) - 1 / \exp(|(V_i^{k+1} - a)/(1 + b)|) \quad (3.43)$$

where a and b are preset constant values that remain consistent during the whole

search procedure. After determining the probabilities, the student revises the positions according to the instructions given in Eq. (3.44).

$$X_i^k = \begin{cases} 1, & \text{if rand} < T(V_i^k) \\ 0, & \text{otherwise} \end{cases} \quad (3.44)$$

Like this the utilization of TLBO for features selection facilitates the identification of informative and discriminative features, thereby improving the accuracy and robustness of machine learning models.

3.4 Parameter Tuning Methods

Machine learning focuses significantly on parameter tuning to improve model performance. Grid Search and Simulated Annealing (SA) are two approaches for parameter tuning that are frequently used. These techniques offer systematic approaches to determine the ideal parameter values for a particular model.

Grid search involves evaluating every possible combination of parameters to identify the best setting for a certain model. It is reliable but computationally expensive for large search spaces since it carefully explores the whole parameter space.

However, SA is a metaheuristic optimisation technique that was motivated by the metallurgy annealing procedure. In order to find the best answer, it follows the slow cooling of a material by accepting "worse" ideas early on and decreasing exploration as it goes along. In non-convex and multi-modal search environments, SA works well because it starts a balance between exploitation and exploration. Even with a small number of evaluations, SA may find effective solutions by carefully examining the parameter space.

The choice between the two approaches depends on the complexity of the problem and the available computational power, and both play significant roles in optimising model performance by determining the best parameter values.

3.4.1 Grid Search

In machine learning, grid search is a typical technique for hyperparameter tuning. It involves carefully looking over a set of values for each hyperparameter and examining how well the model performs for each set of hyperparameters. Finding

the ideal set of hyperparameters that produces the optimum performance is the objective.

Grid search can be stated as follows:

1. Define the Hyperparameter Grid: Determine the range of values or specific discrete values to consider for each hyperparameter. This grid is created based on prior knowledge, domain expertise, or experimentation.
2. Generate Hyperparameter Combinations: Create all possible combinations of hyperparameters from the defined grid. Each combination represents a unique set of hyperparameters to evaluate.
3. Model Training and Evaluation: Train a model for each hyperparameter combination using the training data and evaluate its performance.
4. Select the Best Performing Combination: Compare the performance of the models trained with different hyperparameter combinations and select the combination that achieves the best performance.
5. Evaluate on Test Data: Once the best hyperparameter combination is chosen, evaluate the final model's performance on a separate test set to estimate its generalization ability.

However, some limitations and drawbacks of grid search:

1. Grid searches can be computationally expensive, especially when there are several hyperparameters involved and a wide range of possible values for each one. As the number of combinations grows exponentially with the number of hyperparameters and their values, grid search may become time-consuming or even infeasible for complex models or large datasets.
2. Limited to Discrete Values: Grid search is limited to exploring a predefined grid of values for each hyperparameter. This means that it may miss out on optimal values that fall between the specified grid points. If the optimal value lies outside the predefined grid, grid search may fail to find it.

To address these limitations, alternative methods such as Simulated Annealing (SA) algorithms can be employed. These approaches offer more efficient and flexible ways to explore the hyperparameter space, considering continuous values, exploring interactions, and adapting the search based on previous evaluations.

3.4.2 Simulated Annealing (SA)

Simulated Annealing (SA) is a metaheuristic optimization algorithm that can also be applied to parameter tuning in machine learning models. SA offers an alternative approach to grid search by providing a more flexible and efficient way to explore the parameter space.

The SA algorithm for parameter tuning generally follows these steps:

1. **Initialization:** Start with an initial set of parameters for the model. This can be done randomly or based on some predefined values.
2. **Define an Objective Function:** Establish an objective function that quantifies the performance of the model using the current set of parameters. This function can be based on metrics such as accuracy, precision, recall, or any other relevant measure for the specific problem.
3. **Define Neighboring Solutions:** Determine a way to generate neighboring solutions by making small perturbations to the current set of parameters. These perturbations can involve changing the values of one or more parameters by a small amount.
4. **Acceptance Criterion:** Establish a criterion for accepting or rejecting a neighboring solution. This criterion is typically based on the performance improvement or degradation compared to the current solution. Accepting worse solutions with a certain probability allows for exploration and avoids getting trapped in local optima.
5. **Temperature Schedule:** Define a temperature schedule that controls the exploration-exploitation trade-off during the search process. Initially, the temperature is set high, allowing for more exploration and acceptance of suboptimal solutions. As the search progresses, the temperature decreases, reducing the probability of accepting worse solutions and focusing on convergence.
6. **Iteration:** Perform iterations of the SA algorithm, where at each iteration, a neighboring solution is generated and its acceptance or rejection is determined based on the acceptance criterion. The algorithm continues until a stopping condition is met, such as reaching a maximum number of iterations or achieving a satisfactory performance level.

By using SA for parameter tuning, the algorithm explores the parameter space more efficiently compared to exhaustive grid search. It allows for a more flexible search and has the potential to find better solutions in complex and high-dimensional spaces. However, it is important to note that SA is not guaranteed to find the global optimal solution, but it can provide good approximations in a reasonable amount of time.

CHAPTER 4

Results

In this section, we present the experimental results obtained from the evaluation of Grid Search and Simulated Annealing (SA) parameter tuning methods and Binary Gravitational Search Algorithm (BGSA) and Teacher Learner Based Optimization (TLBO) features selection methods for Support Vector Machines (SVM) and Twin Support Vector Machines (TWSVM) using both linear and Radial Basis Function (RBF) kernels. The objective of this analysis was to assess the impact of different parameter tuning techniques and features selection on the classification performance of SVM and TWSVM models.

To conduct our experiments, we utilized a benchmark datasets. We randomly divided the datasets into training and testing sets, using a [80-20] % split, to ensure a robust evaluation.

In the subsequent sections, we present the dataset details, outline the specific parameter settings for each method, and provide a comprehensive analysis of the experimental results obtained.

4.1 Description of Datasets

Experiments are carried out on well-known UCI datasets to evaluate the effectiveness of the approach suggested. Perform experiments on 23 datasets such as Monk 1, Monk 2, Monk 3, Spect, Herbrman, Statlog, Ionosphere, Pima-Indian, Echo, Germans, Australian, Bupa, Daibetes, Fertility, Sonar, Ecoil, prlx, Medelon, Leukemia, Wine 1 vs 3, Wine 2 vs 3, Zoo and Vots.

Here's a brief description of each dataset:

1. Monk 1, Monk 2, and Monk 3: These datasets consist of three different versions of the Monk problem, which are synthetic datasets used for testing

machine learning algorithms. Each version has different characteristics and levels of difficulty.

2. Spect: This dataset contains spectral data from different types of tissues. It is often used for binary classification tasks.
3. Herbrman: The Herbrman dataset consists of data from herbicide resistance experiments. It is used for classification tasks.
4. Statlog: The Statlog datasets include various datasets such as Statlog (Australian Credit Approval), Statlog (German Credit), and Statlog (Heart). These datasets cover credit approval, credit scoring, and heart disease prediction tasks.
5. Ionosphere: The Ionosphere dataset contains radar data for detecting ionospheric abnormalities. It is used for binary classification tasks.
6. Pima-Indian: The Pima-Indian dataset consists of medical data related to diabetes diagnosis in Pima Indian women. It is commonly used for binary classification tasks.
7. Echo: The Echo dataset contains medical data related to diagnosing heart disease. It is used for binary classification tasks.
8. Germans: The Germans dataset is used for credit risk assessment, where the goal is to predict whether a credit applicant is a good or bad credit risk.
9. Australian: The Australian dataset is used for credit card approval prediction. The objective is to predict whether a credit card application will be approved or not
10. Bupa: The Bupa dataset includes liver disorder data. It is often used for classification tasks related to liver disease prediction.
11. Diabetes: The Diabetes dataset contains data related to diabetes diagnosis. It is used for binary classification tasks.
12. Fertility: The Fertility dataset includes data related to assessing fertility in women. It is used for classification tasks.
13. Sonar: The Sonar dataset consists of sonar signals reflected from different objects in the water. It is used for binary classification tasks.

14. Ecoil: The Ecoil dataset contains data related to protein localization sites. It is used for multi-class classification tasks.
15. PRLX: The PRLX dataset includes data related to predicting the likelihood of preterm labor. It is used for binary classification tasks.
16. Medelon: The Medelon dataset contains synthetic data for binary classification tasks.
17. Leukemia: The Leukemia dataset consists of gene expression data for predicting the subtype of leukemia. It is used for multi-class classification tasks.
18. Wine 1 vs 3 and Wine 2 vs 3: These datasets contain data related to classifying different types of wines. They are used for binary classification tasks.
19. Zoo: The Zoo dataset includes animal attributes and is used for multi-class classification tasks.
20. Vots: The Vots dataset contains data related to voting behavior. It is used for classification tasks.

Table (4.1) provides information about the datasets used in the study. It includes details such as the dataset number, name, number of samples, number of features, and number of classes.

S.No.	Datasets	samples	features	classes
1.	Monk 1	556	6	2
2.	Monk 2	601	6	2
3.	Monk 3	554	6	2
4.	Spect	267	23	2
5.	Herbrman	306	3	2
6.	Statlog	270	13	2
7.	Ionosphere	351	34	2
8.	Pima-Indian	768	8	2
9.	Germans	1000	20	2
10.	Australian	690	14	2
11.	Bupa	345	6	2
12.	Daibetes	768	8	2
13.	Fertility	100	9	2
14.	Sonar	208	60	2
15.	Ecoil	336	7	2
16.	prlx	182	12	2
17.	Medelon	83	5000	2
18.	Leukemia	72	7129	2
19.	Wine 1 vs 3	178	13	3
20.	Wine 2 vs 3	178	13	4
21.	Zoo	101	17	7
22.	Vots	18	14	2
23.	Echo	435	16	2

Table 4.1: About Datasets

4.2 Parameter Setting

Each experiment is conducted on supercomputer using MATLAB with Intel(R) Xeon(R) Gold 6145, 2.0 GHz, 64 GB. In the same computing environment, fair comparison methods such as TLBO, SA and BGSA algorithms run with the appropriate parameter settings, as shown in the table.

S.No.	Parameters	BGSA	TLBO	SA
1.	Population Size	50	50	20
2.	Number of iteration	20	20	10
3.	Performance	Accuracy	Accuracy	Accuracy

Table 4.2: Parameters of Methods

The effectiveness of the methods used is evaluated with the help of a simple classification algorithm called TWSVM. The parameters for TWSVM with a radial basis function of kernel are C and Gamma. Range of C and Gamma(P) are $[2^{-7}, \dots, 2^6, 2^7]$ and $[2^{-7}, \dots, 2^6, 2^7]$ which the TWSVM model uses. Population size and number of iteration for every methods are shows in table (4.2) .

4.3 Results of Experiment

Here, present the experimental results obtained from the evaluation of the parameter tuning methods and features selection methods. Also, we discuss the performance of each method for SVM and TWSVM models using both linear and RBF kernels. Include the comparison of improvement in accuracy of features selection in TWSVM over the features selection in SVM using graph. Also show the reduction in the number of features in each iteration of the features selection process.

Table (4.3) presents the results of the Grid Search method with a linear kernel for the SVM and TWSVM algorithms. The table includes Accuracy, C parameter, and Time (s) for both SVM and TWSVM.

Dataset	SVM			TWSVM		
	Accu.	C	Time(s)	Accu.	C	Time(s)
Monk 1	65.28	0.25	0.1118	65.97	0.5	0.0069
Monk 2	49.46	0.25	0.1678	53.01	1	0.0109
Monk 3	81.48	0.25	0.1759	84.72	0.25	0.0078
Spect	78.07	0.25	0.0308	70.05	0.125	0.0095
Haberman	74.19	0.0078	0.1710	77.42	0.25	0.0276
Statlog	85.19	0.25	0.1320	88.89	2	0.0172
Ionosphere	100	0.0078	0.2780	100	0.0625	0.0388
Pima-Indian	79.87	0.0156	0.7887	77.27	0.125	0.0684
German	92.59	0.0625	0.0271	96.30	8	0.0128
Australian	79	0.0078	1.2969	79.5	0.5	0.1382
Bupa	86.23	0.0078	0.7014	89.13	0.125	0.0710
Diabetes	73.91	32	0.1986	73.91	1	0.0348
Fertility	79.87	0.0156	0.7930	77.27	0.125	0.0834
Sonar	90	0.0078	0.0125	90	0.5	0.0140
Ecoli	73.81	64	0.0863	69.05	0.125	0.0213
Plrx	80.30	128	0.1811	77.27	8	0.0439
Madelon	56.76	0.0078	0.0548	62.16	0.0078	0.0194
Leukemia	59.83	0.0625	27.2217	58.83	0.5	0.7296
Wine 1 vs 2	93.33	0.0078	0.1734	93.33	0.0078	10.2920
Wine 2 vs 3	100	0.0078	0.0146	100	0.0078	0.0130
Zoo	95.83	0.0156	0.0183	100	0.0078	0.0143
Votes	75	0.0078	0.0013	75	0.0078	0.0135
Echo	97.70	0.25	0.2805	96.55	0.0078	0.0382

Table 4.3: Results of Grid Search with Linear Kernel

As we show in above table (4.3) for every datasets TWSVM takes less time compare to SVM and datasets Monk1, Monk2, Monk3, Haberman, Stalog, German, Bupa, Medelon and Zoo have high TWSVM accuracy compare to SVM Accuracy in this case.

Table (4.4) presents the results of the Grid Search method with the RBF kernel for the SVM and TWSVM algorithms. The table includes Accuracy, C parameter, kernel parameter (P), and Time (s) for both SVM and TWSVM.

Dataset	SVM				TWSVM			
	Accu.	C	P	Time(s)	Accu.	C	P	Time(s)
Monk 1	89.35	128	16	0.1717	84.26	0.5	0.0625	0.0221
Monk 2	81.94	0.25	0.0078	0.2635	81.94	0.0078	1	0.0238
Monk 3	85.42	128	32	0.1663	79.63	1	0.0078	0.0253
Spect	91.98	0.0078	0.0078	0.0585	92.51	4	0.0313	0.0180
Haberman	75.81	0.125	0.5	0.1975	77.42	0.25	0.0313	0.0492
Statlog	87.04	32	8	0.1857	88.89	4	0.0313	0.0302
Ionosphere	100	0.0078	0.0078	0.2674	100	0.0078	0.0078	0.0508
Pima-Indian	81.82	1	0.5	1.2630	82.47	0.125	1	0.1562
German	96.30	0.25	2	0.0368	96.30	16	0.25	0.0211
Australian	79.50	64	8	2.1486	78.50	0.125	0.0156	0.2256
Bupa	89.86	0.5	16	1.0964	87.68	0.0078	0.0078	0.1070
Diabetes	81.16	2	2	0.2764	81.16	1	0.125	0.0482
Fertility	81.82	1	0.5	1.3430	82.47	0.125	1	0.1569
Sonar	90	0.0078	0.0078	0.0205	90	0.0078	0.0078	0.0221
Ecoli	59.52	64	8	0.1082	59.52	4	0.0156	0.0291
Plrx	81.82	2	0.125	0.2311	93.94	0.0078	32	0.0430
Madelon	70.27	64	1	0.0777	59.46	0.0078	0.0313	0.0283
Leukemia	61.17	64	8	36.0673	60.33	1	0.0156	1.3174
Wine 1 vs 2	86.67	32	64	0.2699	26.67	0.0078	0.0078	0.0238
Wine 2 vs 3	100	0.5	0.5	0.0235	100	0.0078	0.0078	0.0234
Zoo	100	16	2	0.0292	95.83	0.0078	0.0078	0.0252
Votes	75	0.0078	0.0078	0.0008	100	0.0078	0.25	0.0136
Echo	98.85	128	8	0.3952	98.85	16	0.0078	0.0610

Table 4.4: Results of Grid Search with RBF Kernel

As we show in above table (4.4) for every datasets TWSVM takes less time compare to SVM and datasets Spect , Haberman, Stalog, Pima-indian, Fertility, Plrx and Votes have high TWSVM accuracy compare to SVM Accuracy in this case.

Table (4.5) presents the results of the Simulated Annealing (SA) method with a linear kernel for the SVM and TWSVM algorithms. The table includes Accuracy, C parameter, and Time (s) for both SVM and TWSVM.

Dataset	SVM			TWSVM		
	Accu.	C	Time(s)	Accu.	C	Time(s)
Monk 1	66.20	243.44	0.1941	64.58	0.01	0.0095
Monk 2	67.13	415.53	0.6245	40.74	0.48	0.0145
Monk 3	80.56	136.44	0.0895	83.80	0.01	0.0113
Spect	78.07	0.68	0.0453	70.59	0.12	0.0095
Haberman	74.19	404.16	0.1490	74.19	418.02	0.0379
Statlog	87.04	0.17	0.1101	88.89	2.45	0.0168
Ionosphere	100	369.54	0.2161	100	14.47	0.0340
Pima-Indian	77.92	445.12	0.7827	77.27	0.51	0.0783
German	92.59	206.77	0.0263	96.30	0.61	0.0118
Australian	78.00	0.15	1.3499	69.50	177.88	0.1763
Bupa	86.23	121.29	0.7955	87.68	0.28	0.0693
Diabetes	73.91	27.50	0.1825	66.67	45.56	0.0249
Fertility	77.92	0.10	0.7427	77.92	0.34	0.0756
Sonar	90.00	211.86	0.0126	90.00	4.88	0.0114
Ecoli	69.05	40.66	0.1225	69.05	309.54	0.0174
Plrx	80.30	149.38	0.1677	80.30	12.89	0.0205
Madelon	56.76	300.73	0.0528	62.16	0.01	0.0199
Leukemia	58.50	1.51	27.0121	54.67	498.17	0.8211
Wine 1 vs 2	93.33	358.52	0.1541	93.33	10.96	9.9088
Wine 2 vs 3	100	475.87	0.0173	100	475.87	0.0094
Zoo	95.83	63.64	0.0227	100	0.01	0.0104
Votes	75.00	71.89	0.0015	75.00	71.89	0.0090
Echo	97.70	206.03	0.2831	96.55	0.01	0.0289

Table 4.5: Results of SA with Linear Kernel

As we show in above table (4.5) for every datasets TWSVM takes less time compare to SVM and datasets Monk3, Statlog, German, Bupa, Madelon and Zoo have high TWSVM accuracy compare to SVM Accuracy in this case.

Table (4.6) presents the results of the Simulated Annealing (SA) method with the RBF kernel for the SVM and TWSVM algorithms. The table includes Accuracy, C parameter, kernel parameter (P), and Time (s) for both SVM and TWSVM.

Dataset	SVM				TWSVM			
	Accu.	C	P	Time(s)	Accu.	C	P	Time(s)
Monk 1	93.06	430.86	23.00	0.178	83.33	156.30	0.114	0.0156
Monk 2	81.94	512	0.01	0.261	81.94	242.70	36.02	0.0212
Monk 3	85.42	512	48.08	0.186	79.86	512	0.01	0.0208
Spect	91.98	305.61	479.10	0.052	91.98	280.65	0.01	0.0146
Haberman	75.81	512	3.36	0.234	74.19	0.01	0.01	0.0369
Statlog	87.04	285.65	37.65	0.187	88.89	314.06	0.023	0.0332
Ionosphere	100	387.88	68.01	0.429	100	236.59	78.19	0.0561
Pima-Indian	79.87	259.12	89.89	1.317	79.22	0.01	4.189	0.1422
German	92.59	342.15	25.99	0.052	88.89	94.31	4.476	0.0307
Australian	79.00	379.74	31.71	2.344	69.50	169.13	20.20	0.4068
Bupa	89.86	301.92	460.61	1.119	86.96	0.01	0.01	0.0968
Diabetes	78.26	368.82	5.79	0.299	71.01	512	51.29	0.0409
Fertility	79.87	402.39	108.03	1.333	79.87	0.01	0.01	0.1162
Sonar	90.00	94.83	78.98	0.024	90.00	436.95	391.06	0.0139
Ecoli	66.67	460.99	10.02	0.116	45.24	362.97	0.01	0.0254
Plrx	74.24	242.50	2.36	0.275	90.91	421.92	85.02	0.0339
Madelon	56.76	239.65	167.83	0.111	56.76	455.15	42.13	0.0306
Leukemia	61.17	329.48	511.85	35.55	59.67	359.47	0.0102	1.3531
Wine 1 vs 2	93.33	246.73	176.98	0.256	26.67	42.34	430.79	0.0209
Wine 2 vs 3	100	353.74	475.87	0.025	100	512	0.01	0.0149
Zoo	95.83	10.19	63.64	0.032	95.83	0.01	0.01	0.0164
Votes	75.00	101.44	71.89	0.002	100	101.44	71.89	0.0092
Echo	97.70	375.76	20.41	0.491	96.55	88.23	0.01	0.0569

Table 4.6: Results of SA with RBF Kernel

As we show in above table (4.6) for every datasets TWSVM takes less time compare to SVM and datasets Statlog and Vots have high TWSVM accuracy compare to SVM Accuracy in this case.

Table (4.7) presents selected features of certain datasets. The table includes the dataset names (e.g., Monks1, Monks2, Heberman, Fertility), the number of features in each dataset, and the binary representation of the selected features. The features are represented as 0 or 1, where 1 indicates that the feature is selected for classification, and 0 indicates that the feature is not selected.

Dataset	No. of Features	Features								
		1	2	3	4	5	6	7	8	9
Monks1	6	1	0	1	1	0	0	-	-	-
Monks2	6	1	0	0	1	0	0	-	-	-
Monks3	6	1	0	0	1	0	1	-	-	-
Heberman	3	1	0	1	-	-	-	-	-	-
Fertility	9	0	0	1	0	1	1	1	0	1

Table 4.7: Selected Features of some datasets

The graph (fig 4.1) presents the relationship between the number of iterations and the number of selected features. The number of selected features gradually decrease over the iterations. Here we show example of two dataset Madelon and Leukemia.

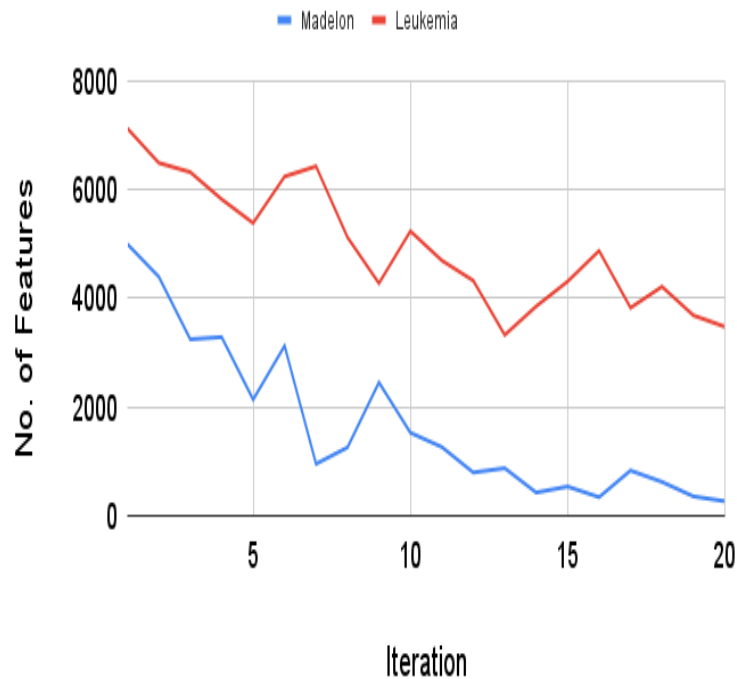


Figure 4.1: Iteration vs No. of Features

Table (4.8) presents the results of the SA-BGSA methods with Linear kernel for the SVM and TWSVM algorithms. The table includes the dataset names (e.g., Monk 1, Spect, Leukemia), the number of features in each dataset, and the corresponding accuracy , time and selected features results for both SVM and TWSVM.

Dataset		SVM			TWSVM		
	features	Accu.	Time(s)	No. of Selected features	Accu.	Time(s)	No. of Selected features
Monk 1	6	72.22	20.35	3	70.83	4.6	4
Monk 2	6	67.13	57.29	2	67.13	4.73	1
Monk 3	6	85.65	16.56	3	88.89	4.75	3
Spect	23	80.75	17.5	11	75.94	3.92	11
Haberman	3	74.19	23.56	2	74.19	21.45	2
Statlog	13	88.89	45.34	8	88.89	46.79	9
Ionosphere	34	100	98.34	26	100	89.75	27
Pima-Indian	8	80.52	138.22	3	83.12	17.67	4
German	20	96.30	36.33	15	96.30	36.33	15
Australian	14	79.00	13.77	8	76.50	16.03	9
Bupa	6	86.23	45.62	4	89.86	34.56	3
Diabetes	8	71.01	56.23	5	75.36	45.67	5
Fertility	9	81.17	2.48	2	83.12	4.58	4
Sonar	60	90.00	13.14	29	90.00	5.03	38
Ecoli	7	76.19	23.58	2	80.95	5.41	3
Plrx	12	96.97	7.39	4	100	4.02	6
Madelon	5000	56.76	1375.57	258	56.76	33.48	239
Leukemia	7129	61.50	18.78	3561	60.83	861.73	3511
Wine 1 vs 2	13	100	5.39	7	93.33	15.34	8
Wine 2 vs 3	13	100	920.16	10	100	131.32	8
Zoo	17	100	0.65	7	100	3.77	4
Votes	14	100	64.58	9	100	10.94	10
Echo	16	97.70	4.32	5	96.55	5.41	3

Table 4.8: Results of SA-BGSA with Linear Kernel

As we show in above table (4.8) for every datasets except Leukemia TWSVM takes less time compare to SVM and datasets Monk3, Pima-indian, Bupa, Diabetes, Fertility, Ecoli and Plrx have high TWSVM accuracy compare to SVM Accuracy in this case.

Table (4.9) presents the results of the SA-BGSA methods with RBF kernel for the SVM and TWSVM algorithms. The table includes the dataset names (e.g., Monk 1, Spect, Leukemia), the number of features in each dataset, and the corresponding accuracy, time and selected features results for both SVM and TWSVM.

Dataset		SVM			TWSVM		
	features	Accu.	Time(s)	No. of Selected features	Accu.	Time(s)	No. of Selected features
Monk 1	6	95.14	38.39	3	85.42	9.86	4
Monk 2	6	81.94	56.71	3	81.94	11.61	3
Monk 3	6	88.89	39.91	4	90.28	9.45	4
Spect	23	73.52	17.5	11	91.98	8.92	12
Haberman	3	75.81	43.05	2	87.56	21.45	2
Statlog	13	78.56	45.34	8	88.89	12.11	8
Ionosphere	34	100	98.34	26	100	89.75	27
Pima-Indian	8	83.12	325.85	7	83.12	17.67	4
German	20	82	495.98	13	82	111.34	15
Australian	14	90.58	207.89	10	91.3	41.94	6
Bupa	6	81.16	43.7	5	76.81	16.55	5
Diabetes	8	72.563	56.23	5	78.56	45.67	5
Fertility	9	90	4.4	5	95	6.36	4
Sonar	60	80.95	21.75	37	69.05	9.11	42
Ecoli	7	96.97	44.32	4	100	13.49	1
Plrx	12	70.27	17.39	4	64.86	7.64	7
Madelon	5000	63.67	2172.94	229	62.67	83.01	245
Leukemia	7129	100	29.07	3573	100	237.35	3510
Wine 1 vs 2	13	100	5.39	7	100	4.67	8
Wine 2 vs 3	13	100	920.16	10	100	235.45	8
Zoo	17	75	0.66	5	100	3.77	4
Votes	14	98.85	88.61	9	100	3.98	6
Echo	16	96.3	8.1	4	100	22.8	10

Table 4.9: Results of SA-BGSA with RBF Kernel

As we show in above table (4.9) for every datasets except Fertility, Zoo, Echo TWSVM takes less time compare to SVM and datasets Monk3, Sepct, Harberman, Statlog, Australian, Diabetes, Fertility, Ecoli, Zoo, Votes and Echo have high TWSVM accuracy compare to SVM Accuracy in this case.

Table (4.10) presents the results of the SA-TLBO methods with Linear kernel for the SVM and TWSVM algorithms. The table includes the dataset names (e.g., Monk 1, Spect, Leukemia), the number of features in each dataset, and the corresponding accuracy, time and selected features results for both SVM and TWSVM.

Dataset		SVM			TWSVM		
	features	Accu.	Time(s)	No. of Selected features	Accu.	Time(s)	No. of Selected features
Monks1	6	72.22	22.69	3	70.92	5.82	4
Monks2	6	67.49	60.71	3	67.94	5.7	3
Monks3	6	84.6	19.98	4	90.28	4.52	4
Spect	23	75.24	43.34	14	76.99	9.28	12
Haberman	3	74.81	55.98	2	74.98	24.15	2
Statlog	13	83.68	50.653	6	88.27	11.51	8
Ionosphere	34	100	101.49	25	100	90.52	24
Pima-Indian	8	84.69	402.79	7	84.69	19.74	4
German	20	96.84	35.93	14	96.97	132.48	16
Australian	14	79	15.876	10	79	49.14	6
Bupa	6	85.03	47.7	5	90.84	15.85	5
Diabetes	8	72.53	63.59	5	76.587	47.98	5
Fertility	9	85.786	4.6	5	82.9786	9.6	4
Sonar	60	82.6	19.49	42	90	11.98	42
Ecoli	7	75.97	25.88	4	81.9807	14.95	1
Plrx	12	95.47	9.71	4	100	6.47	7
Madelon	5000	62.37	1572.598	364	63.982	89.91	267
Leukemia	7129	64.74	23.57	3298	65.87	250.57	3479
Wine 1 vs 2	13	100	7.93	7	94.86	5.75	8
Wine 2 vs 3	13	100	987.667	10	100	289.55	8
Zoo	17	100	1.809	5	100	3.89	4
Votes	14	99.875	87.785	9	100	8.78	6
Echo	16	96.6	8.34	4	97.6	7.96	10

Table 4.10: Results of SA-TLBO with Linear Kernel

As we show in above table (4.10) for every datasets except German, Australian, Fertility, Leukemia, Zoo TWSVM takes less time compare to SVM and datasets Monk3, Sepct, Harberman, Statlog, German, Diabetes, Fertility, Sonar, Ecoli, Pirx, Madelon, Leukemia, Votes and Echo have high TWSVM accuracy compare to SVM Accuracy in this case.

Table (4.11) presents the results of the SA-TLBO methods with RBF kernel for the SVM and TWSVM algorithms. The table includes the dataset names (e.g., Monk 1, Spect, Leukemia), the number of features in each dataset, and the corresponding accuracy, time and selected features results for both SVM and TWSVM.

Dataset		SVM			TWSVM		
	features	Accu.	Time(s)	No. of Selected features	Accu.	Time(s)	No. of Selected features
Monks1	6	92.64	60.38	3	94.52	27.69	4
Monks2	6	82.74	70.71	3	86.42	16.87	3
Monks3	6	87.89	91.39	4	90.28	10.52	4
Spect	23	76.82	34.8	13	91.49	9.28	12
Haberman	3	74.81	55.05	2	87.98	24.15	2
Statlog	13	81.56	78.25	7	88.27	11.51	8
Ionosphere	34	100	108.84	24	100	90.52	24
Pima-Indian	8	84.12	398.57	7	81.76	19.74	4
German	20	80.48	535.93	14	83.897	132.48	16
Australian	14	89.46	289.07	10	90.79	49.14	6
Bupa	6	80.35	47.7	5	79.84	15.85	5
Diabetes	8	72.563	65.39	5	77.67	47.98	5
Fertility	9	90	6.4	5	95	9.6	4
Sonar	60	80.26	29.59	39	75.95	11.98	42
Ecoli	7	94.97	49.28	4	99.87	14.95	1
Plrx	12	71.47	19.37	4	69.69	6.47	7
Madelon	5000	70.23	2100.72	298	69.72	89.91	267
Leukemia	7129	100	32.75	3256	100	250.57	3479
Wine 1 vs 2	13	100	9.53	7	100	5.75	8
Wine 2 vs 3	13	100	960.36	10	100	289.55	8
Zoo	17	75	5.86	5	89.76	3.89	4
Votes	14	96.85	92.15	9	100	8.78	6
Echo	16	96.96	10.34	4	100	7.96	10

Table 4.11: Results of SA-TLBO with RBF Kernel

As we show in above table (4.11) for every datasets except Fertility and Leukemia TWSVM takes less time compare to SVM and datasets Monk 1, Monk 2, Monk 3, Spect, Herbrman, Statlog, Echo, Germans, Australian, Daibetes, Fertility, Ecoil, Zoo and Vots have high TWSVM accuracy compare to SVM Accuracy in this case.

This graph (fig 4.2) represents the accuracy comparison between the Grid Search, Simulated Annealing (SA), and SA-BGSA algorithms for SVM with a linear kernel. The x-axis represents the different datasets or experiments, while the y-axis represents the accuracy values.

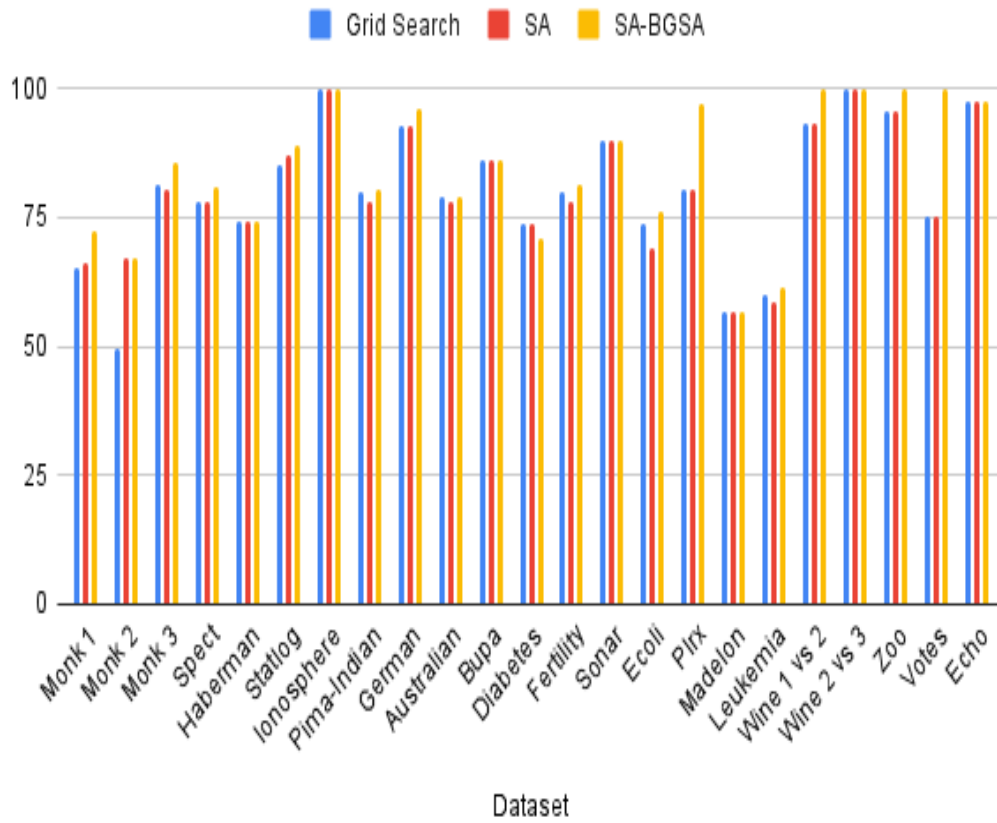


Figure 4.2: Comparison of Grid Search ,SA and SA-BGSA of SVM with Linear Kernel

This graph (fig 4.3) represents the accuracy comparison between the Grid Search, Simulated Annealing (SA), and SA-BGSA algorithms for TWSVM with a linear kernel. The x-axis represents the different datasets or experiments, while the y-axis represents the accuracy values.

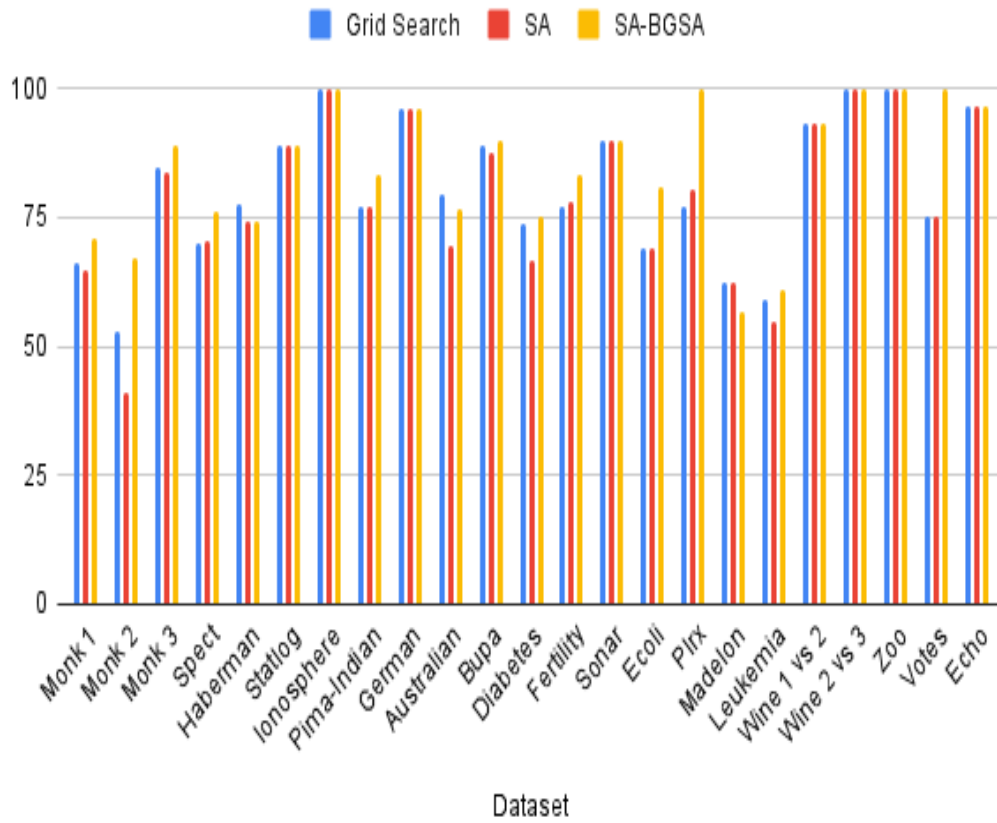


Figure 4.3: Comparison of Grid Search ,SA and SA-BGSA of TWSVM with Linear Kernel

This graph (fig 4.4) represents the accuracy comparison between the Grid Search, Simulated Annealing (SA), and SA-BGSA algorithms for SVM with a RBF kernel. The x-axis represents the different datasets or experiments, while the y-axis represents the accuracy values.

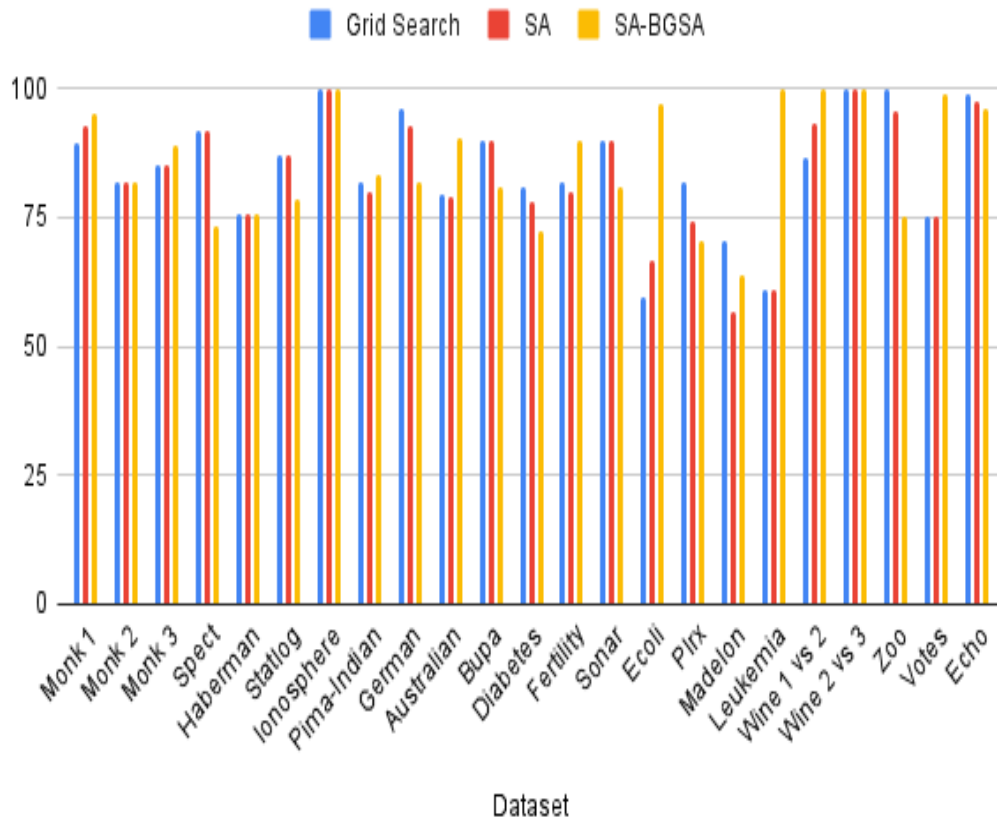


Figure 4.4: Comparison of Grid Search ,SA and SA-BGSA of SVM with RBF Kernel

This graph (fig 4.5) represents the accuracy comparison between the Grid Search, Simulated Annealing (SA), and SA-BGSA algorithms for TWSVM with a RBF kernel. The x-axis represents the different datasets or experiments, while the y-axis represents the accuracy values.

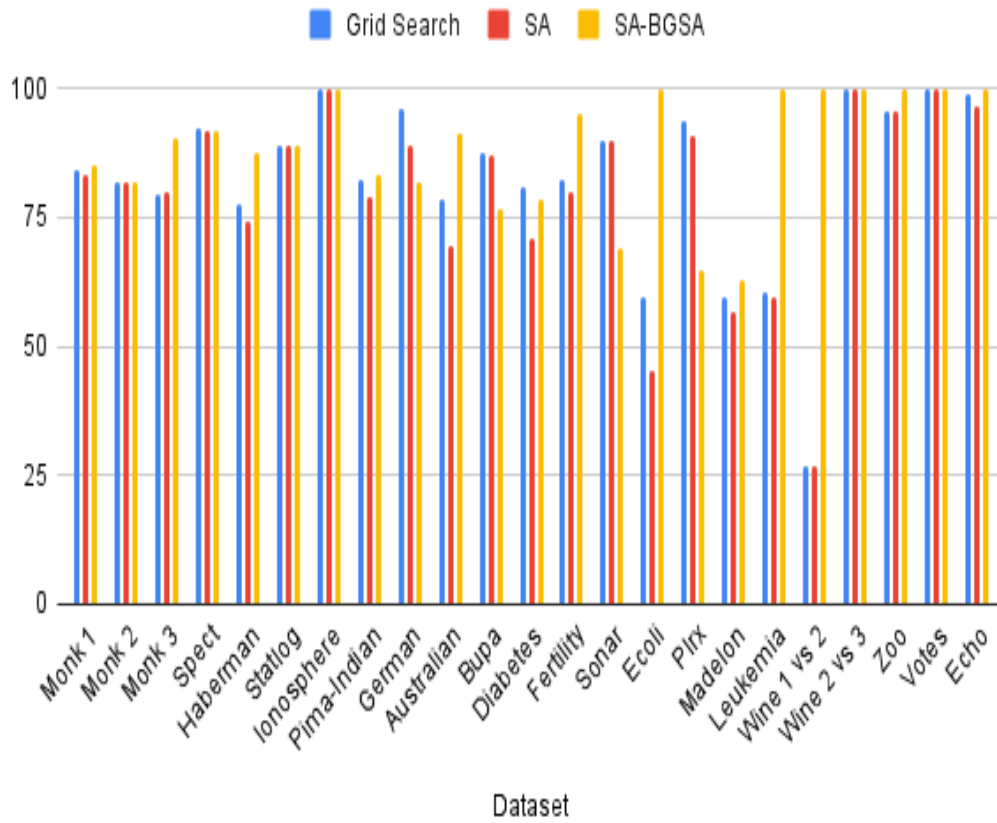


Figure 4.5: Comparison of Grid Search ,SA and SA-BGSA of TWSVM with RBF Kernel

This graph (fig 4.6) represents the accuracy comparison between the Grid Search, Simulated Annealing (SA), and SA-TLBO algorithms for SVM with a Linear kernel. The x-axis represents the different datasets or experiments, while the y-axis represents the accuracy values.

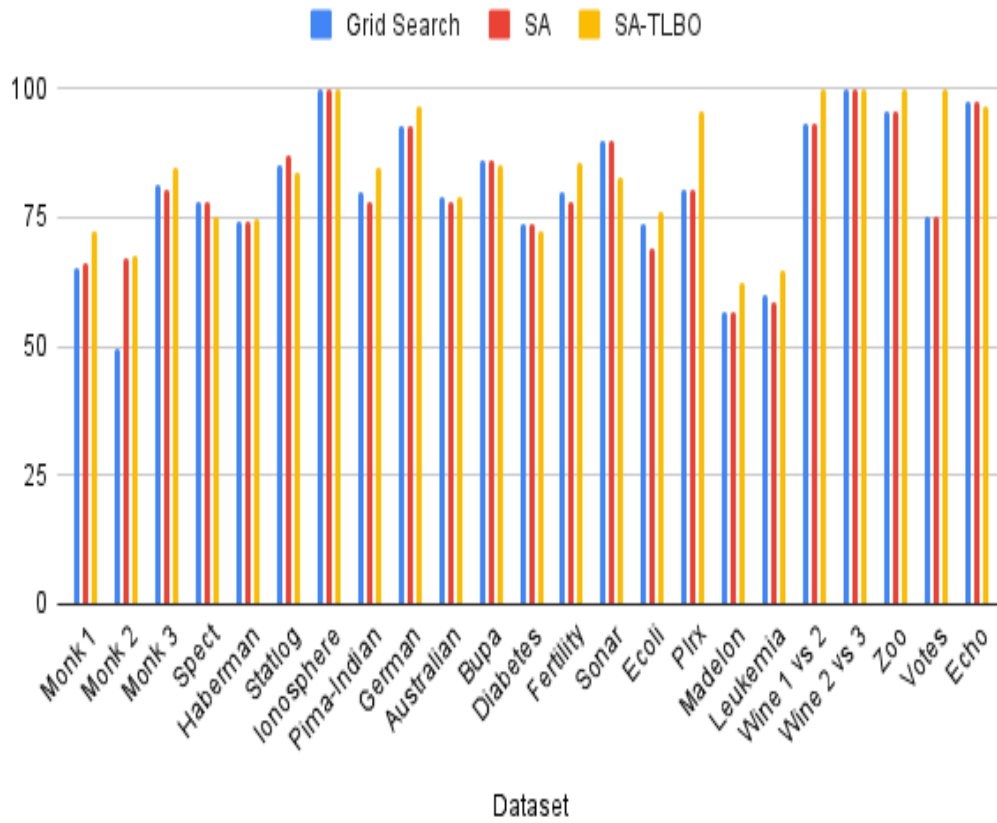


Figure 4.6: Comparison of Grid Search ,SA and SA-TLBO of SVM with Linear Kernel

This graph (fig 4.7) represents the accuracy comparison between the Grid Search, Simulated Annealing (SA), and SA-TLBO algorithms for TWSVM with a linear kernel. The x-axis represents the different datasets or experiments, while the y-axis represents the accuracy values.

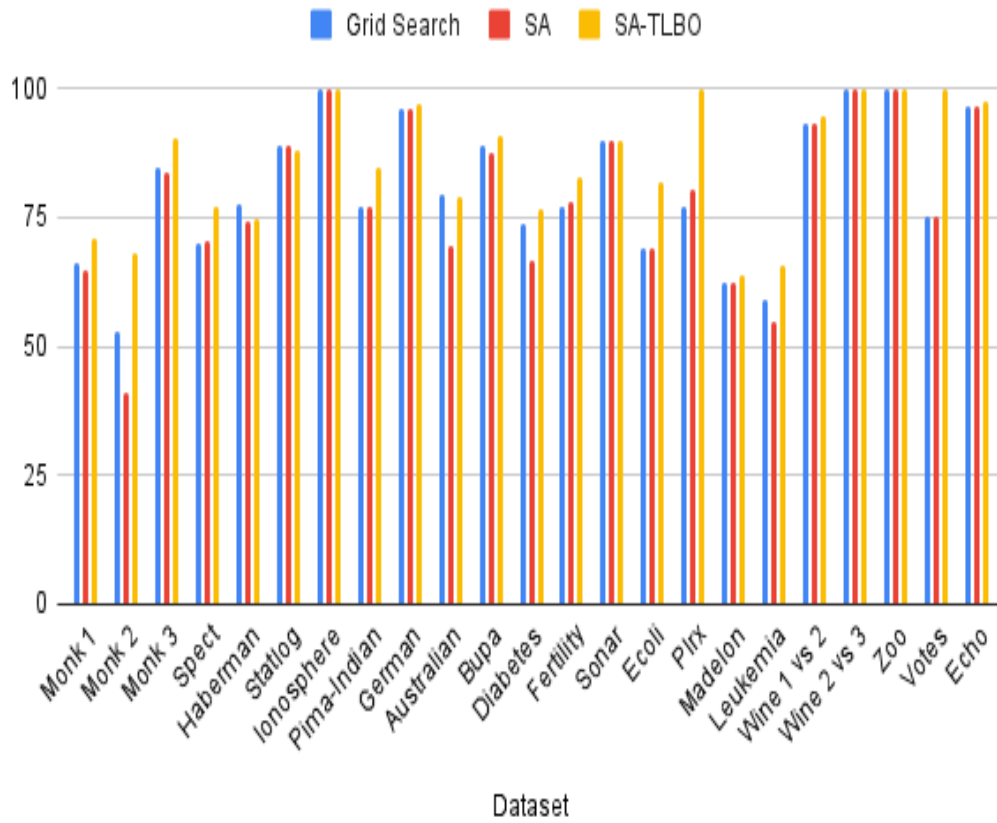


Figure 4.7: Comparison of Grid Search ,SA and SA-TLBO of TWSVM with Linear Kernel

This graph (fig 4.8) represents the accuracy comparison between the Grid Search, Simulated Annealing (SA), and SA-TLBO algorithms for SVM with a RBF kernel. The x-axis represents the different datasets or experiments, while the y-axis represents the accuracy values.

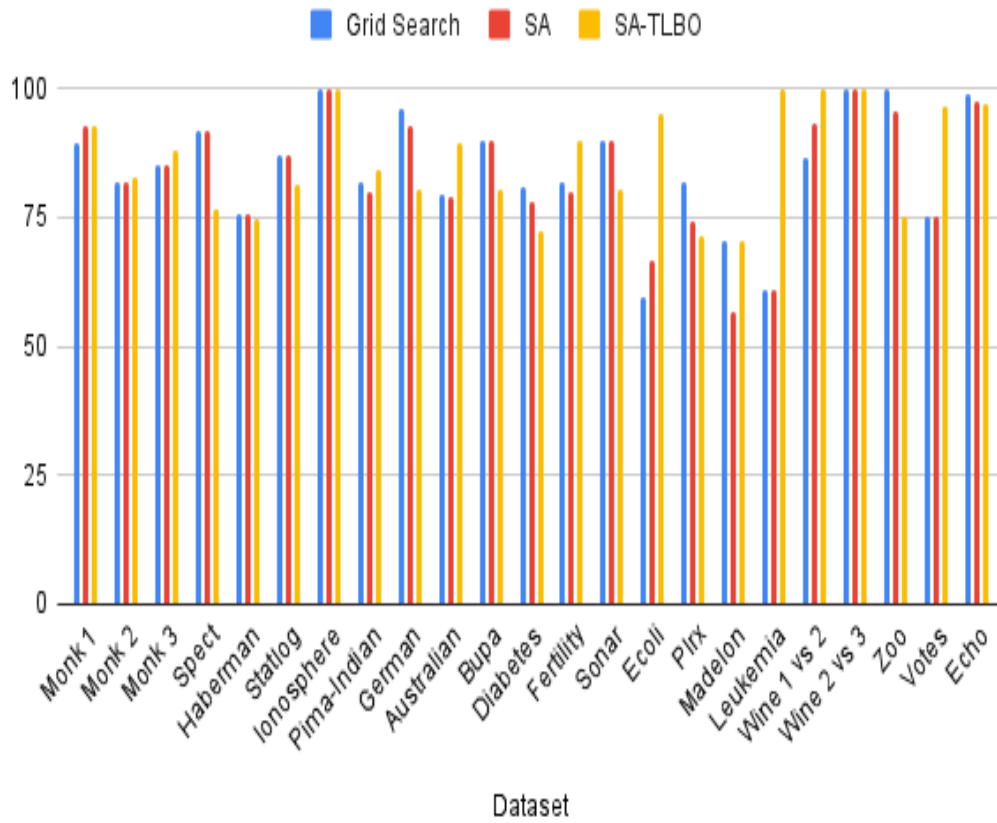


Figure 4.8: Comparison of Grid Search ,SA and SA-TLBO of SVM with RBF Kernel

This graph (fig 4.9) represents the accuracy comparison between the Grid Search, Simulated Annealing (SA), and SA-TLBO algorithms for TWSVM with a RBF kernel. The x-axis represents the different datasets or experiments, while the y-axis represents the accuracy values.

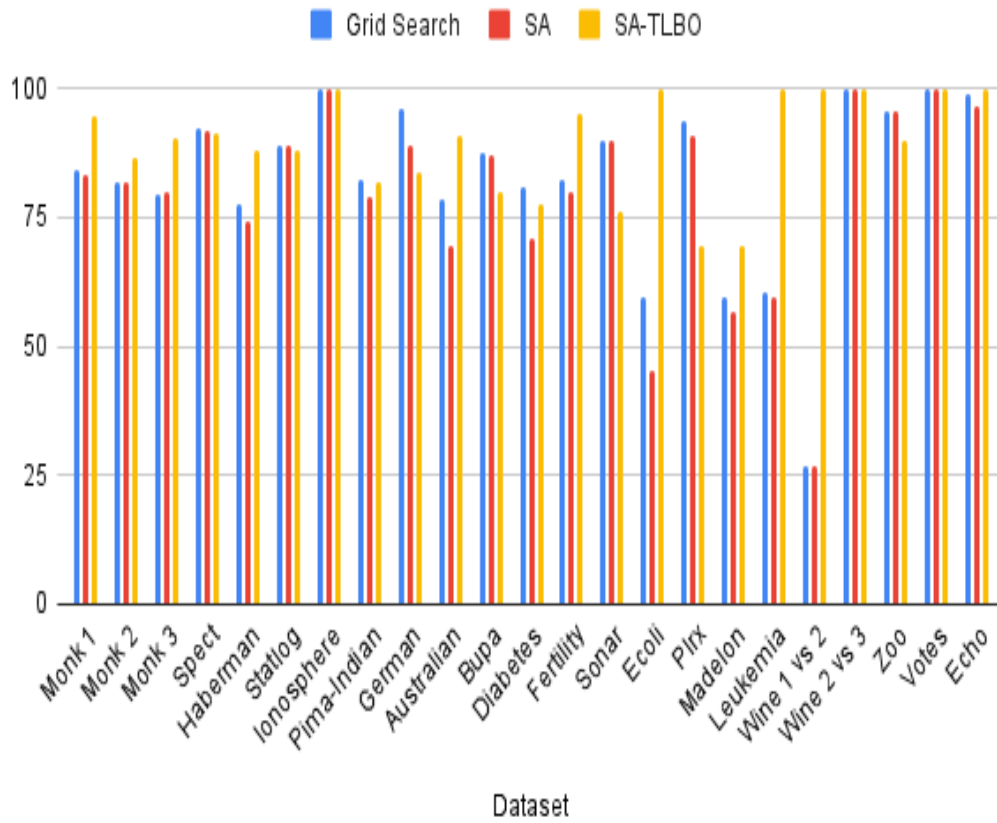


Figure 4.9: Comparison of Grid Search ,SA and SA-TLBO of TWSVM with RBF Kernel

The comprehensive analysis of figure [4.2 - 4.9] clearly demonstrates that the Simulated Annealing (SA) method outperforms Grid Search in all scenarios for both SVM and TWSVM models with Linear and RBF kernels. Across various datasets, we observed higher accuracy values for SVM and TWSVM models when SA was employed for parameter tuning. These results indicate the superior effectiveness of SA in identifying optimal hyperparameters for enhanced model performance.

Additionally, each figure shows the results of feature selection for SVM and TWSVM models with linear and RBF kernels using the SA-based hybrid approaches, SA-BGSA and SA-TLBO. However, feature selection using SA-BGSA and SA-TLBO improved accuracy in comparison with grid search and SA across all datasets.

The combination of SA for parameter adjustment and feature selection techniques (SA-BGSA and SA-TLBO) produced consistent performance increases in every case for SVM and TWSVM models with both Linear and RBF kernels. These results show the robustness and scalability of hybrid approaches.

This graph (fig 4.10) presents a comparison of accuracy results of SVM and TWSVM using the BGSA feature selection method with a linear kernel.

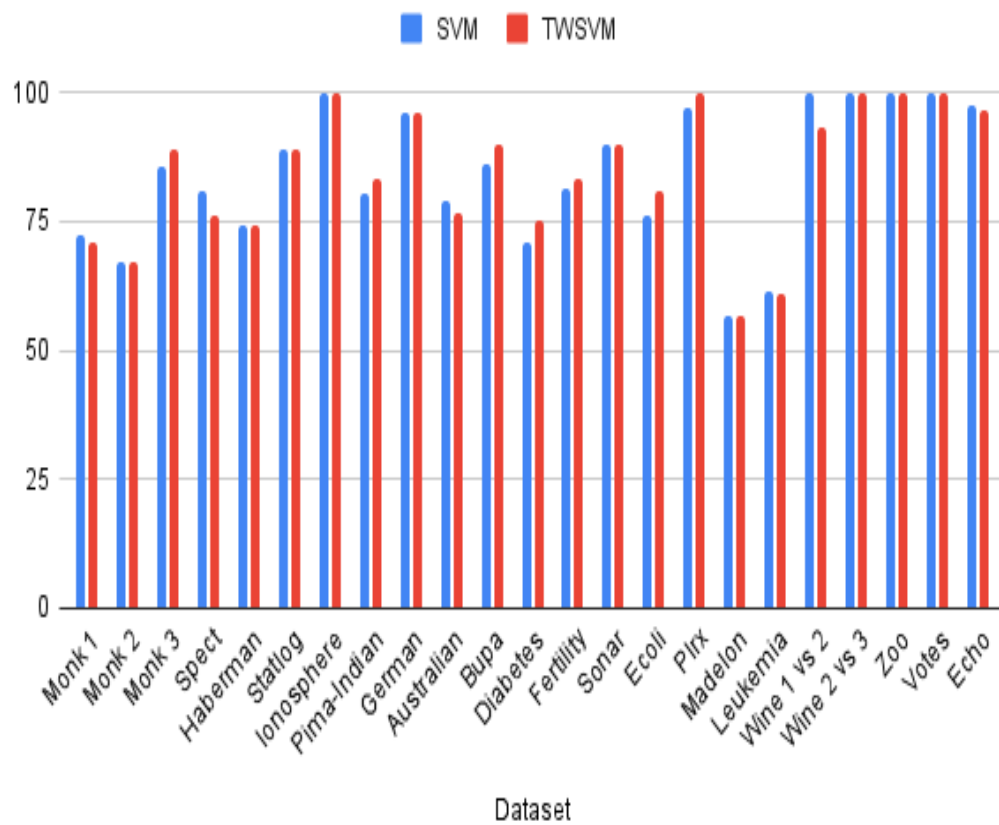


Figure 4.10: Comparison of BGSA with Linear Kernel

This graph (fig 4.11) presents a comparison of accuracy results of SVM and TWSVM using the BGSA feature selection method with a RBF kernel.

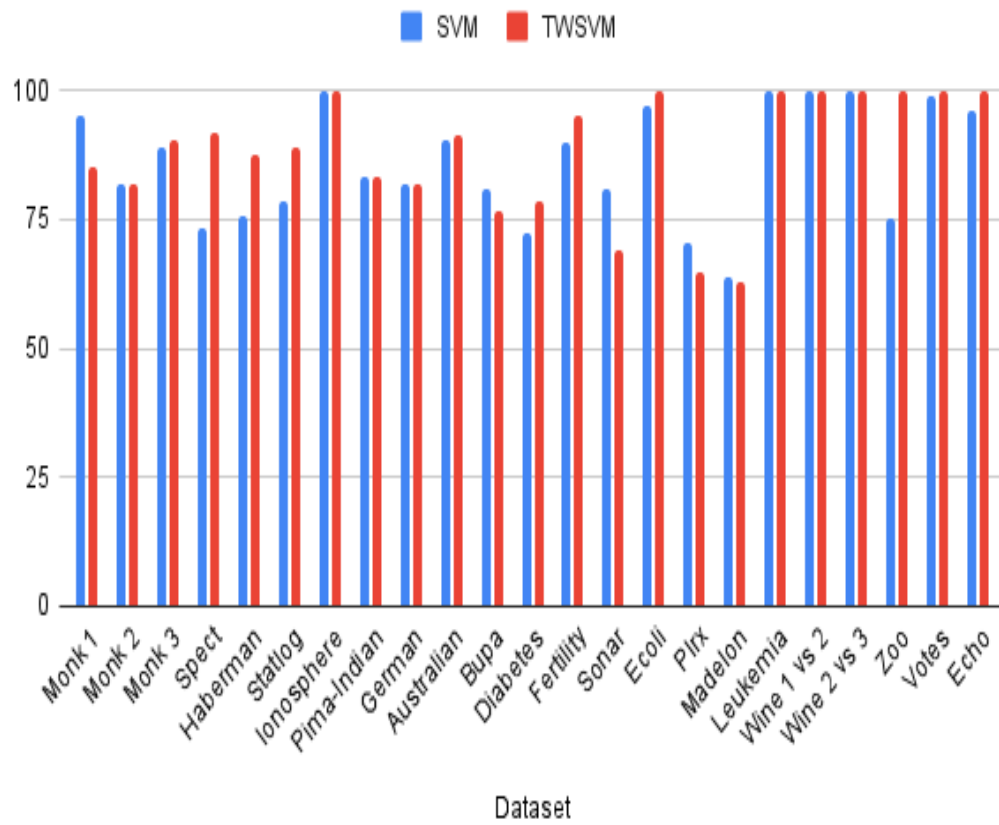


Figure 4.11: Comparison of BGSA with RBF Kernel

This graph (fig 4.12) presents a comparison of accuracy results of SVM and TWSVM using the TLBO feature selection method with a linear kernel.

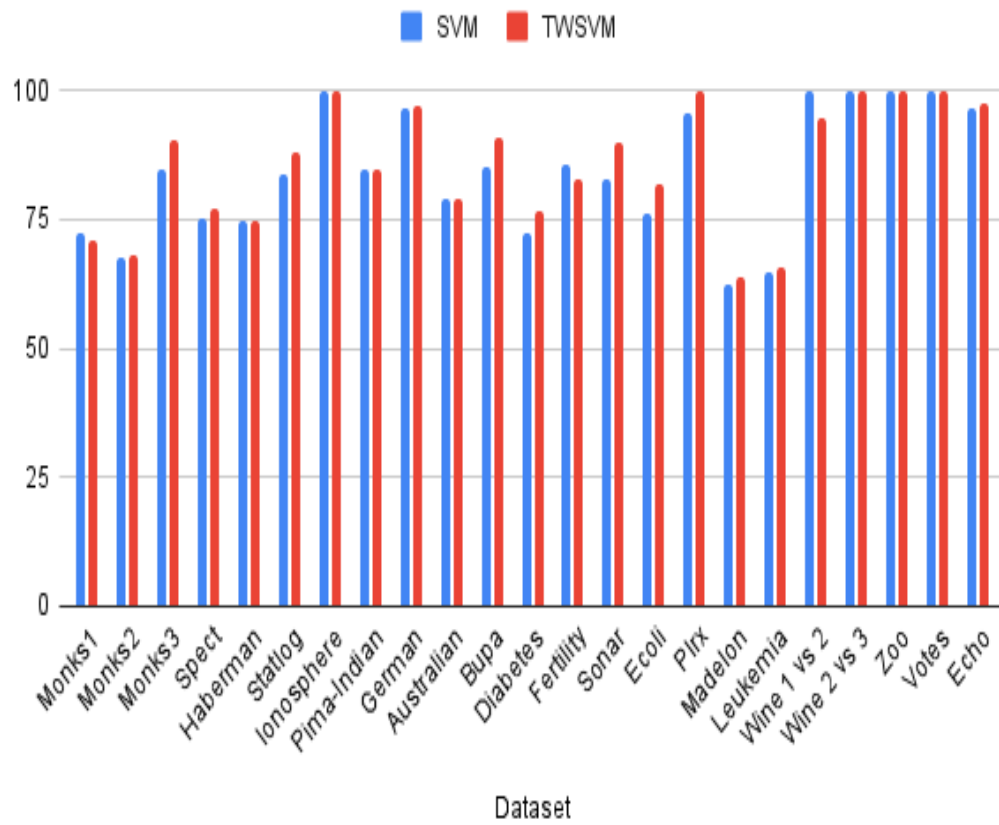


Figure 4.12: Comparison of TLBO with Linear Kernel

This graph (fig 4.13) presents a comparison of accuracy results of SVM and TWSVM using the TLBO feature selection method with a RBF kernel.

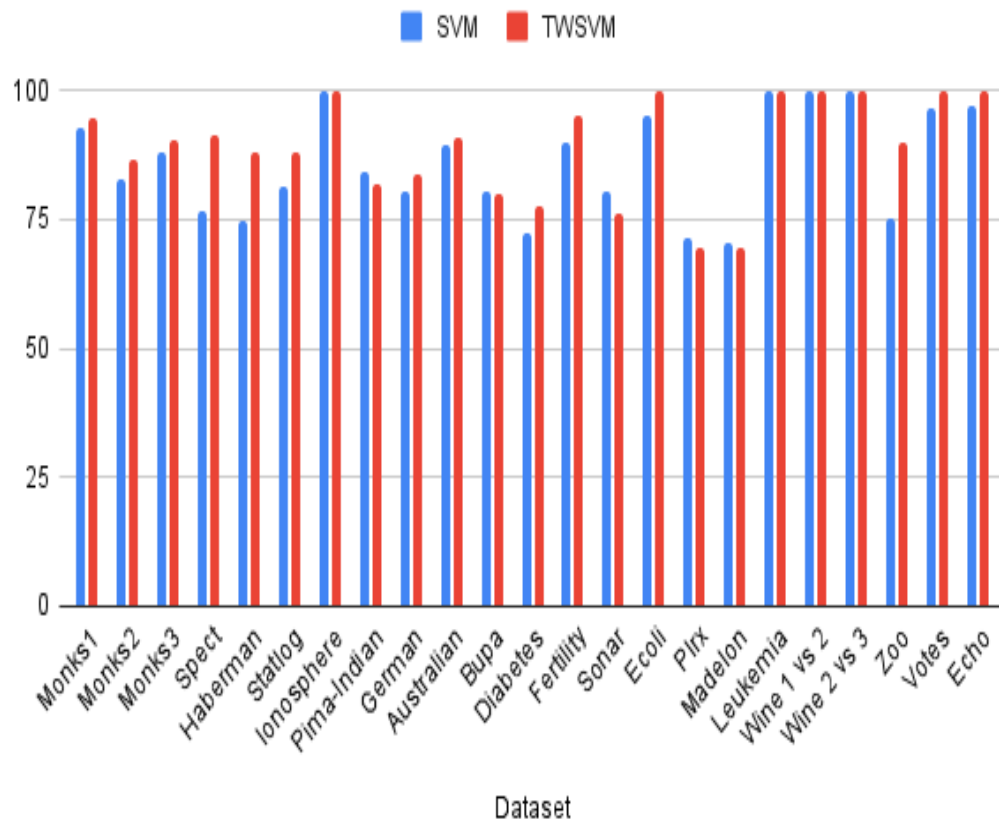


Figure 4.13: Comparison of BGSA with RBF Kernel

The comprehensive analysis of the figures [4.10 - 4.13] show that the comparison of feature selection accuracy of SVM with TWSVM for both feature selections methods SA-BGSA and SA-TLBO using linear and RBF kernel. In most of the datasets TWSVM performs well in both the methods as shows in above graphs.

CHAPTER 5

Conclusions

In conclusion, our study highlighted the importance of feature selection and parameter tuning in the development of machine learning models. The performance and accuracy of the models have been improved by parameter tuning through optimization of hyperparameters and the most important subset of features was simultaneously identified using feature selection, which improved the model's interpret-ability and allowed for more accurate predictions.

This study's primary goal was to determine how well feature selection methods for SVM and TWSVM models worked. To evaluate the effectiveness of these models using various feature selection methods, we have been done numerous experiments and run on a variety of datasets. The results showed that feature selection significantly improved TWSVM's classification accuracy and efficiency as compared to SVM.

For parameter tuning, we used Grid Search and Simulated Annealing (SA), and we discovered that SA performed better than Grid Search, which led to greater model accuracy and generalization. For the feature selection methods we used BGSA and TLBO for both SVM and TWSVM with linear and RBF kernels. Finally we used hybrid method which is parameter tune and feature selection. Combining feature selection and SA-based parameter tuning methods was successful in improving our model's performance

The performance of TWSVM and SVM models is significantly impacted by feature selection techniques, as shown by our study's information. There have been noticeable improvements in the precision and efficiency of TWSVM across the majority of datasets after applying feature selection using both TLBO and BGSA. These selected features not only made the predictions better but also made the model easier to understand.

Furthermore, the advantage of feature selection became more evident when comparing the computation times of TWSVM and SVM. For big datasets, TWSVM constantly outperformed SVM in terms of training time and making it a more time-effective solution. TWSVM is a desirable solution for real-world applications because, when combined with feature selection, it produced better results with less computational cost.

In conclusion, our research shows that feature selection techniques like TLBO and BGSA significantly improve the performance of TWSVM models. These methods takes less time for model generation process while simultaneously improving model accuracy. We can build robust machine learning models that perform well in various real-world scenarios and fields by employing feature selection on TWSVM.

References

- [1] P. Bradley and O. Mangasarian. Massive data discrimination via linear support vector machines. *Optimization methods and software*, 13(1):1–10, 2000.
- [2] C. J. Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.
- [3] V. Cherkassky and F. M. Mulier. *Learning from data: concepts, theory, and methods*. John Wiley & Sons, 2007.
- [4] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.
- [5] R. Diao and Q. Shen. Two new approaches to feature selection with harmony search. In *Proc. of the 19th International Conference on Fuzzy Systems*, pages 3161–3167, 2010.
- [6] S. Ding, J. Yu, B. Qi, and H. Huang. An overview on twin support vector machines. *Artificial Intelligence Review*, 42(2):245–252, 2014.
- [7] H. A. Firpi and E. Goodman. Swarmed feature selection. In *33rd Applied Imagery Pattern Recognition Workshop (AIPR'04)*, pages 112–118. IEEE, October 2004.
- [8] R. Guha, M. Ghosh, A. Chakrabarti, R. Sarkar, and S. Mirjalili. Introducing clustering based population in binary gravitational search algorithm for feature selection. *Applied Soft Computing*, 93:106341, 2020.
- [9] S. R. Gunn et al. Support vector machines for classification and regression. *ISIS technical report*, 14(1):5–16, 1998.
- [10] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.
- [11] M. A. Hall. *Correlation-based feature selection for machine learning*. PhD thesis, The University of Waikato, 1999.

- [12] G. H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Machine learning proceedings 1994*, pages 121–129. Elsevier, 1994.
- [13] R. Khemchandani, S. Chandra, et al. Twin support vector machines for pattern classification. *IEEE Transactions on pattern analysis and machine intelligence*, 29(5):905–910, 2007.
- [14] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1-2):273–324, 1997.
- [15] O. L. Mangasarian. *Nonlinear programming*. SIAM, 1994.
- [16] S. Mirjalili and A. Lewis. S-shaped versus v-shaped transfer functions for binary particle swarm optimization. *Swarm and Evolutionary Computation*, 9:1–14, 2013.
- [17] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, 27(8):1226–1238, 2005.
- [18] R. Rao and V. Patel. An elitist teaching-learning-based optimization algorithm for solving complex constrained optimization problems. *international journal of industrial engineering computations*, 3(4):535–560, 2012.
- [19] R. V. Rao, V. J. Savsani, and D. Vakharia. Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. *Computer-aided design*, 43(3):303–315, 2011.
- [20] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi. Gsa: A gravitational search algorithm. *Information Sciences*, 179(13):2232–2248, 2009.
- [21] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi. Bgsa: Binary gravitational search algorithm. *Natural Computing*, 9:727–745, 2010.
- [22] C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. 1998.
- [23] S. Shahbeig, M. S. Helfroush, and A. Rahideh. A fuzzy multi-objective hybrid tlbo–pso approach to select the associated genes with breast cancer. *Signal processing*, 131:58–65, 2017.

- [24] E. Vega-Alvarado, E. A. Portilla-Flores, M. B. Calva-Yáñez, G. Sepúlveda-Cervantes, J. A. Aponte-Rodríguez, E. Santiago-Valentín, and J. M. A. Rueda-Meléndez. Hybrid metaheuristic for designing an end effector as a constrained optimization problem. *IEEE Access*, 5:6002–6014, 2017.
- [25] Y. Wang, I. V. Tetko, M. A. Hall, E. Frank, A. Facius, K. F. Mayer, and H. W. Mewes. Gene selection from microarray data for cancer classification—a machine learning approach. *Computational biology and chemistry*, 29(1):37–46, 2005.
- [26] J. Wei, R. Zhang, Z. Yu, R. Hu, J. Tang, C. Gui, and Y. Yuan. A bpsosvm algorithm based on memory renewal and enhanced mutation mechanisms for feature selection. *Applied Soft Computing*, 58:176–192, 2017.
- [27] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for svms. *Advances in neural information processing systems*, 13, 2000.
- [28] J. Yang and V. Honavar. Feature subset selection using a genetic algorithm. *IEEE Intelligent Systems and their Applications*, 13(2):44–49, 1998.
- [29] L. Yu and H. Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 856–863, 2003.
- [30] H. Zhang and G. Sun. Feature selection using tabu search method. *Pattern recognition*, 35(3):701–711, 2002.