Boosting SVM Performance through Hyperparameters Optimization and ANN Model

TECHNICAL REPORT

SUBMITTED BY

Muhammad Uzair Saleem

2020-AG-6452

ADVISED BY

Dr. Hassan Tariq

A TECHNICAL REPORT SUBMITTED IN PARTIAL FULFILLMENT OF REQUIREMENT FOR THE DEGREE OF

BACHELOR OF SCIENCE

IN

INFORMATION TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

FACULTY OF SCIENCES

UNIVERSITY OF AGRICULTURE FAISALABAD

DECLARATION

I hereby declare that the contents of the report "**Boosting SVM Performance through Hyperparameters Optimization and ANN Model**" are project of my own research and no part has been copied from any published source (except the references). I further declare that this work has not been submitted for award of any other diploma/degree. The university may take action if the information provided is found false at any stage. In case of any default, the scholar will be proceeded against as per UAF policy.

Muhammad Uzair Saleem

CERTIFICATE

To,

The Controller of Examinations,

University of Agriculture,

Faisalabad.

The supervisory committee certify that **Muhammad Uzair Saleem 2020-ag-6452** has successfully completed his project in partial fulfillment of requirement for the degree of Bachelor of Science in Information Technology under our guidance and supervision.

____

Dr. Hassan Tariq

Supervisor

ACKNOWLEDGEMENT

I thank all who in one way or another contributed in the completion of this report. First, I thank to ALLAH ALMIGHTY, most magnificent and most merciful, for all his blessings. Then I am so grateful to the Department of Computer Science for making it possible for me to study here. My special and heartily thanks to my supervisor, *Dr Hassan Tariq* who encouraged and directed me. His challenges brought this work towards a completion. It is with his supervision that this work came into existence. For any faults, I take full responsibility. I am also deeply thankful to my informants. I want to acknowledge and appreciate their help and transparency during my research. I am also so thankful to my fellow students whose challenges and productive critics have provided new ideas to the work. Furthermore, I also thank my family who encouraged me and prayed for me throughout the time of my research. May the Almighty God richly bless all of you

Abstract

In the realm of machine learning, Support Vector Machines (SVMs) stand out as prominent tools for classification tasks due to their versatility and effectiveness in handling diverse analytical scenarios. SVMs rely heavily on well-tuned hyperparameters to optimize their performance, which can significantly influence outcomes such as overfitting and model accuracy. This study endeavors to elevate SVM performance through meticulous hyperparameter tuning across two distinct datasets: the Dry Bean and Swarm datasets, each presenting unique challenges in feature composition and classification complexity. Employing an exhaustive grid search of hyperparameter combinations, we harness Python's computational libraries to fine-tune the SVM models efficiently. The optimal configurations derived from these experiments serve as the cornerstone for training an Artificial Neural Network (ANN) tasked with predicting crucial performance metrics—accuracy, precision, recall, and F1-score—based on specified hyperparameter values. This predictive model undergoes rigorous evaluation through real-time scenario simulations. The outcomes demonstrate that while customized SVM models exhibit enhanced performance metrics, the ANN serves as a robust framework for predicting these metrics, thereby facilitating informed decisions in model configuration. This approach underscores the potential of integrating SVMs with neural networks to establish a predictive ensemble that advances model selection and optimization. Future endeavors may explore adapting this methodology to other machine learning paradigms and assessing the impact of advanced neural modeling techniques on predictive accuracy.

Table of Contents

Table of Figures

Chapter 1

Introduction

In the ever-progressing realm of artificial intelligence (AI), machine learning emerges as a pivotal technology empowering machines to glean insights from data and autonomously navigate decisions with minimal human intervention. This ability to iteratively refine decision-making processes without explicit programming distinguishes machine learning in contemporary times. Amidst the myriad techniques, Support Vector Machines (SVMs) and Artificial Neural Networks (ANNs) stand out for their unique attributes and versatile utility spanning diverse domains.

1.1 SVM: Principles and Practicalities

Support Vector Machines (SVMs) are indeed a potent tool in machine learning, particularly for classification tasks like the study of Swarm Behavior. Leveraging SVMs involves a deep understanding of their principles and careful tuning of hyperparameters to optimize performance.

In our study, we harnessed SVMs by finely tuning hyperparameters to elevate their efficacy on the publicly available Swarm Behavior dataset provided by The University of New South Wales (UNSW Sydney) [1]. By delving into the intricacies of SVMs and employing techniques to enhance their performance, we aimed to extract meaningful insights from the data and advance our understanding of swarm dynamics.

Through meticulous hyperparameter tuning, we sought to strike a balance between model complexity and generalization capacity, ultimately aiming to achieve a robust decision boundary that effectively separates different classes of swarm behavior. This process involved iteratively adjusting parameters such as the choice of kernel function, regularization parameter (C), and kernel coefficient (gamma), among others, to optimize classification accuracy and minimize the risk of overfitting.

Furthermore, our study underscored the versatility of SVMs in handling high-dimensional datasets, such as those encountered in the study of swarm behavior, where the interactions between numerous individuals give rise to complex patterns. By harnessing the power of SVMs and refining their parameters through rigorous experimentation, we aimed to unveil the underlying dynamics governing swarm behavior and contribute to the broader body of knowledge in this field.

In essence, our utilization of SVMs in studying swarm behavior exemplifies the practical application of this powerful machine learning technique, displaying its adaptability and effectiveness in uncovering insights from complex datasets.

1.2 Artificial Neural Networks: Expanding Capabilities

ANNs are artificial adaptive systems that are inspired by the functioning processes of the human brain. They are systems that are able to modify their internal structure in relation to a function objective. They are particularly suited for solving problems of the nonlinear type, being able to reconstruct the fuzzy rules that govern the optimal solution for these problems.

The base elements of the ANN are the nodes, also called processing elements (PE), and the connections. Each node has its own input, from which it receives communications from other nodes and/or from the environment and its own output, from which it communicates with other nodes or with the environment. Finally, each node has a function f through which it transforms its own global input into output. Each connection is characterized by the strength with which pairs of nodes are excited or inhibited. Positive values indicate excitatory connections, the negative ones inhibitory connections. The connections between the nodes can modify themselves over time. This dynamic starts a learning process in the entire ANN. The way through which the nodes modify themselves is called 'Law of Learning'. The total dynamic of an ANN is tied to time. In fact, for the ANN to modify its own connections, the environment has to necessarily act on the ANN more times. Data are the environment that acts on the ANN. The learning processes, therefore, one of the key mechanisms that characterize the ANN, which are considered adaptive processing systems. The learning process is one way to adapt the connections of an ANN to the data structure that makeup the environment and, therefore, a way to understand the environment and the relations that characterize it.

Neurons can be organized in any topological manner (e.g. one- or two-dimensional layers, three-dimensional blocks or more-dimensional structures), depending on the quality and amount of input data. The most common ANNs are composed in a so-called feed forward topology. A certain number of PEs is combined to an input layer, normally depending on the amount of input variables. The information is forwarded to one or more hidden layers working within the ANN. The output layer, as the last element of this structure, provides the result. The output layer contains only one PE, whether the result is a binary value or a single number.

All PEs within the ANN are connected to other PEs in their neighborhood. The way these connections are made might differ between the subtypes of neural networks. Each of these connections has a so-called weight, which modifies the input or output value. The value of these connection weights is determined during the training process. This functionality is the basis for the ANN is learning capability. Therefore, it is important to understand that there are no classification. Rules written into the algorithm. The network just learns to understand and classify input patterns from examples.

Basic neural networks can normally be obtained with statistical computer software packages. Some companies offer specialized software to work with different neural networks (e.g. Neural Works Professional by Neural Ware Inc., Carnegie, Pennsylvania, USA or CLEMEN-TINE Data Mining tool by Integral Solutions Limited, UK. These software packages must be flexible and easy to handle for use in widespread purposes [2].

1.3 Bridging Decision Trees and ANNs through Hyperparameter Insights

In the dynamic realm of artificial intelligence (AI) and machine learning, the convergence of diverse methodologies often leads to innovative breakthroughs. One such intriguing intersection lies between Support Vector Machines (SVMs) and Artificial Neural Networks (ANNs), two formidable techniques each with its unique strengths and applications. While SVMs offer robustness and efficiency in handling high-dimensional data, ANNs provide adaptability and versatility in capturing complex patterns. This project delves into the synergies between SVMs and ANNs by exploring how insights derived from tuning hyperparameters in SVMs can enrich the training of ANNs. Focusing exclusively on the Swarm dataset, renowned for its dynamic and intricate patterns simulating swarm behavior, this study aims to display the efficacy of a hybrid modeling approach in enhancing predictive capabilities. By harnessing the structured nature of the Swarm dataset and the resilience of SVMs, we endeavor to construct a robust framework for predictive analytics with broad implications across domains. Through meticulous experimentation and analysis, this research seeks to illuminate the potential of integrating insights from SVM hyperparameter optimization into ANN training, propelling advancements in machine learning methodologies.

Chapter 2

Background

Following Section will introduce to the model and methods that are used for the analysis of the dataset.

2.1 Support Vector Machines (SVM)

Support Vector Machines (SVM) have been recently developed in the framework of statistical learning theory (Vapnik, 1998) (Cortes and Vapnik, 1995), and have been successfully applied to a number of applications, ranging from time series prediction (Fernandez, 1999), to face recognition (Tefas et al., 1999), to biological data processing for medical diagnosis (Veropoulos et al., 1999). Their theoretical foundations and their experimental success encourage further research on their characteristics, as well as their further use [3].

SVM represents a pillar in the world of machine learning due to its outstanding efficiency and convenience of implementation while solving problems facing very different domains. The industry stands out for the nexus of many factors that intrinsically position it in the robotics domain as the best suited for supervised, unsupervised, and semi supervised classification tasks.

At the core of what makes the SVM powerful is its ability to identify a supreme hyperplane that one can use to most effectively maximize the distance between two distinct classes. Such skill makes the decision boundary perform not only reliably and with a grand vision, but also limiting the risk of overfitting when the data is noisy or there is an uneven distribution of classes.

Additionally, SVM provides us with a remit of difficulties that include the ability to cope with both linear and non-linear problems. By means of sorting kernel function, support vector machine mediates to transcend a picture of input data into a feature space with a larger dimension and then linear reparability is acquired. The flexibility of SVM allows the approach to capture subtle, and in some cases complex, relations among the features. Generally, this suits multiple real-world applications.

Along with this, it is worth noting that SVM is distinguished for its openness to limited memory storage, which, in turn, is particularly useful in cases stigmatized by many features and few examples. As a result of mainly depend on only a few input vectors or so-called support vector that had been used to define decision pathway, SVMs are able to overcome the difficulties when the datasets are high-dimensional space like in text classification or image recognition.

The theory base of SVM giving optimal solutions support the model's greatness. SVM is expressed as a convex optimization problem. The output is governed by the globally optimal solutions, i.e. the stability and reliability are seldom be compromised. The extended theoretical framework of the model discloses how the underlying algorithm behave and leads to the adjustment of the parameters, thereby improvement of model interpretation and usability.

In my study, binary classification was conducted using the following formula:

$$f(x)=sign(w^{T}x+B) \qquad (1)$$

- f(x) is the predicted class label (either +1 or -1)
- w is the weight vector
- x is the input data vector,
- B is the bias term.

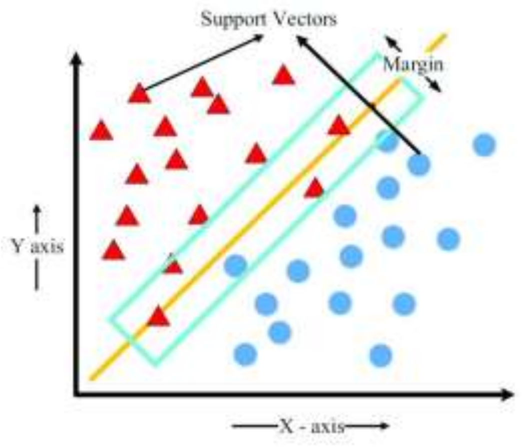


Figure 1: Binary classification based on support vector machine (SVM).

Figure 1 [4] is simply showing how different points are classified in linearly method where datapoints above and below the line belong to the different category

2.1.1 Hyperparameters Used

Following Hyperparameters are tuned in order to enhance the performance of the SVM.

2.1.1.1 Kernel Parameter

Kernel is a function that calculates the similarity between data points in a higher-dimensional space. SVM utilizes kernels to map the input data into a higher-dimensional feature space where the data might be more separable. This allows SVM efficiently find a hyperplane that can easily separate out the data points into different classes. In the following study, all four of the kernel of SVM Model are used.

- Linear Kernel: This kernel defines a linear decision boundary in the original feature space
  Formula:

$$K(x,x')= x^{T}x' \qquad (2)$$

- Polynomial Kernel: This kernel introduces non-linearity by mapping the data into a higher-dimensional space using polynomial functions.

Formula:

$$K(x,x') = (x^T x' + c)^d \tag{3}$$

- Radial Basis Function (RBF) Kernel: Also known as the Gaussian kernel, it maps the data into an infinite-dimensional space, providing more flexibility in capturing complex relationships in the data.
  Formula:

$$K(x,x') = exp(-y||x-x'||^2) \tag{4}$$

- Sigmoid Kernel: This kernel function maps the data into a space where the decision boundary is sigmoidal in shape.
  Formula:

$$K(x,x') = tanh(ax^T x' + c) \tag{5}$$

The choice of kernel depends on the problem at hand and the characteristics of the data. Different kernels can lead to different decision boundaries and model performances.

2.1.1.2 C Parameter

In this study, the "C" parameter is being employed for the purpose of bringing together classifications with overall margin maximization and the training data error minimization. This hyperparameter determines the cost of an incorrectly classified database point. Specifically, five different values of C are employed 0.1, 1, 10, 100, 1000.

- Maximizing the Margin: SVM, in this case, looks for a hyperplane that parts the data points of different classes while maintaining the maximum margin. A bigger pat it's possible while a smaller number will be misclassified for the training data.
- Minimizing Misclassification: Conversely, the SVM aims to reduce the misclassification of training data. For a smaller value of C the confidence interval become larger and the number of misclassification may increase.

By varying the parameters of C gives an alternative to achieve a trade-offs between how large the margin is and how correctly the classification is performed. The impact of C values on how the SVM models are trained can be explored through an analysis of the SVM models trained with varying C values. The regularization parameter’s behavior and its ability to generalize can be enhanced through these findings. This tedious review simplifies the choice between the available C values for the optimal discriminating performance.

2.1.1.3 Gamma Parameter

In the analysis, the gamma parameter is experimented with various values: Auto, Scale, 0.1, 0.01 Each of these values affects the behavior of the SVM model differently

- Auto: This mode can set the value of regularization parameter gamma automatically, by dividing it into the inverse of number of features in dataset. It allows the algorithm to default to adjusting the gamma parameter depending on whatever the properties of the dataset could be.
- Scale: This setting can be approximated by gamma of 2 variances of data with respect to the data number of features (variance of data number of features). The established function affects the gamma coefficient correspondingly to the input features variants, in such a way that if any of the components has a larger variability, it will have more impact.
- 0.1, 0.01: These particular numeric values are shown for gamma, which make the scope of significance of each data point more dominant to form the decision boundary.

Apply different values of gamma helps us to see how the influence of gamma is affecting overall performance of the model.

2.1.1.4 Class Weight Parameter

In this analysis, class weight are "none" and "balanced". Here is what these values mean:

- None: Seeing "none" as the class weights in an SVM model indicates that no particular weighting method was used for classifying the training data. Intuitively, every class will have equal representation whether it is of high frequency or even imbalanced in the dataset.
- Balanced: In contrast, widely known use of "balanced" class weights leads to relatively low weights of classes being specified inversely proportional to their frequencies, where the most frequent class has the minimum weight. Thus, samples with lesser cases are considered the most significant and each case of the class with larger cases contributes slight to the other classes. The objective is to integrate each sample in the training process and to control the class impact on the model. Besides the class issue, this can be used to address class imbalance issues.

A comparison between the SVM model with none and with balanced class weights is carried out in this study. These results highlights the capability of different weighting schemes to correct imbalanced datasets and generate predictions that are more precise.

2.1.2 Accuracy Metrics

Model performance is evaluated using four metrics: accuracy test, F1 score, the precision score, and the recall score. These metrics are essential for reasons that they present you with the different views with which you can understand how well your model is performing. Precision and recall give an overall measure of the accuracy and the F1 score, on the other hand which helps in balancing both these metrics when classes are imbalanced. Precision score is concerned about how many times the model is successful in predicting positive cases among actually positives ones and recall score is worried with how many actual positive cases your model is able to capture. These metrics in aggregate enables you to make a judgement on the accuracy of your model.

2.1.2.1 Test Accuracy

Test accuracy is like a score that tells you how good your model is at getting predictions right on new data it has not seen before. If your model's accuracy is 80%, it means it gets 8 out of 10 predictions correct. It is a way to measure how accurate your model is when faced with new information.

Formula:

$$Test\ Accuracy = \frac{Number\ of\ correctly\ classified\ samples\ in\ the\ test\ set}{Total\ number\ of\ test\ samples} \tag{6}$$

2.1.2.2 Precision Score

Precision, also referred to as Confidence in Data Mining, represents the percentage of predicted positive cases that are correctly classified as actual positives by the model. It is a crucial metric in Machine Learning, Data Mining, and Information Retrieval, as it focuses on the accuracy of positive predictions. Unlike in ROC analysis, where it's disregarded, precision is analogous to True Positive Accuracy (tpa), highlighting the precision of predicted positives compared to the rate of discovering actual positives.

Formula:

$$Precision\ Score = \frac{True\ Postive(TP)}{True\ Postive(TP) + True\ Negative(TN)} \tag{7}$$

2.1.2.3 Recall Score

Recall measures the percentage of actual positive cases that are correctly identified as positive by the model. It indicates how well the model captures all relevant instances of a particular class. While in fields like Information Retrieval, it's less valued due to the assumption that finding any relevant documents suffices, in Machine Learning and Computational Linguistics, where confidence in the model's predictions matters, it's often overlooked. Nevertheless, in contexts like Machine Translation, recall plays a significant role in predicting the success of tasks like Word Alignment [5].

Formula:

$$Recall\ Score = \frac{True\ Positive\ (TP)}{True\ Postive(TP) + False\ Negative(FN)} \tag{8}$$

2.1.2.4 F1 Score

The F-score, which combines precision and recall, tends to favor algorithms with higher sensitivity, like SVM. However, algorithms with higher specificity may face challenges in achieving high F-scores. Therefore, SVM's effectiveness in comparison to other algorithms, such as Naive Bayes (NB), largely depends on the evaluation metrics utilized [5].

Formula:

$$F1\ Score = \frac{2*Precision\ score*Recall\ score}{Precision\ score+Recall\ score} \tag{9}$$

		Actual Values (as confirmed by experiment)	
		Positives	Negatives
Predicted Values (as predicted by experiment)	Positives	TP (True Positive)	TN (True Negative)
	Negative	FP (False Positive)	FN (False Negative)

2.2 Artificial Neural Network

Artificial Neural Networks are relatively crude electronic models based on the neural structure of the brain. The brain learns from experience. It is natural proof that some problems that are beyond the scope of current computers are indeed solvable by small energy efficient packages. This brain modeling also promises a less technical way to develop machine solutions. This new approach to computing also provides a more graceful degradation during system overload than its more traditional counterparts. These biologically inspired methods of computing are thought to be the next major advancement in the computing industry. Even simple animal brains are capable of functions that are currently impossible for computers. Computers do rote things well, like keeping ledgers or performing complex math. However, computers have trouble recognizing even simple patterns much less generalizing those patterns of the past into actions of the future. Now, advances in biological research promise an initial understanding of the natural thinking mechanism. This research shows that brains store information as patterns. Some of these patterns are very complicated and allow us the ability to recognize individual faces from many different angles. This process of storing information as patterns, utilizing those patterns, and then solving problems encompasses a new field in computing. This field, as mentioned before, does not utilize traditional programming but involves the creation of massively parallel networks and the training of those networks to solve specific problems. This field also utilizes words very different from traditional computing, words like behave, react, self-organize, learn, generalize, and forget. Whenever we talk about a neural network, we should more popularly say ―Artificial Neural Network (ANN), ANN are computers whose architecture is modelled after the brain. They typically consist of hundreds of simple processing units, which are wired together, in a complex communication network. Each unit or node is a simplified model of real neuron, which sends off a new signal or fires if it receives a sufficiently strong Input signal from the other nodes to which it is connected.

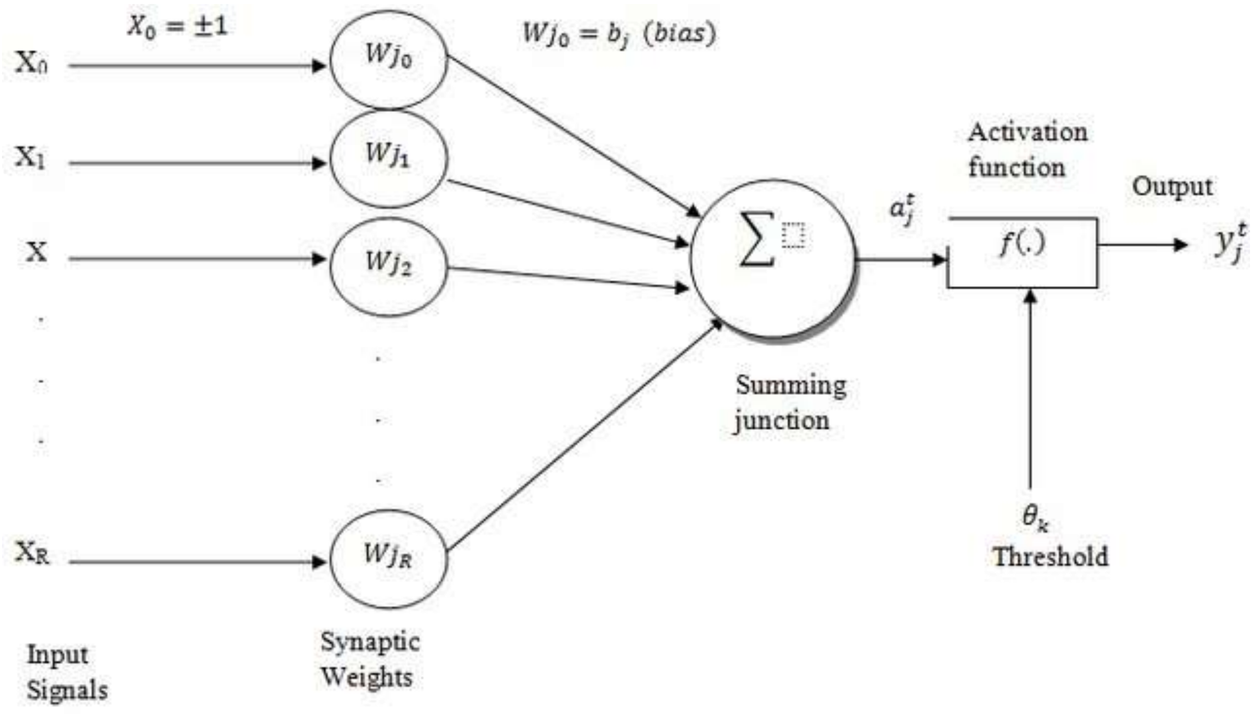


Figure 2: Mathematical Model of ANN

Traditionally neural network was used to refer as network or circuit of biological neurons, but modern usage of the term often refers to ANN. ANN is mathematical model or computational model, an information processing paradigm i.e. inspired by the way biological nervous system, such as brain information system. ANN consists of interconnecting artificial neurons, which are programmed like to mimic the properties of m biological neurons. These neurons working in unison to solve specific problems. ANN is configured for solving artificial intelligence problems without creating a model of real biological system. ANN is used for speech recognition, image analysis, adaptive control etc. These applications are done through a learning process, like learning in biological system, which involves the adjustment between neurons through synaptic connection. Figure 2 [6] presents the simple mathematical model working behind ANN.

In this case study, we utilized an ANN model and trained it on the hyperparameters that produced the highest accuracy depending on the SVM model. The goal of the present work was to use the ANN-based model for studying the dependence of various SVM hyperparameters from the final model accuracy. Through learning the ANN in a dataset containing different hyperparameters in

SVMs and their respective accuracies, our primary objective was determining the ideal hyperparameters that give the highest accuracy in classification tasks based on SVMs. The method applied allowed to obtain the comprehensive analysis of the effects of hyperparameter values to SVM performance where the opportunity was given for the hyperparameter tuning based on the achieved results.

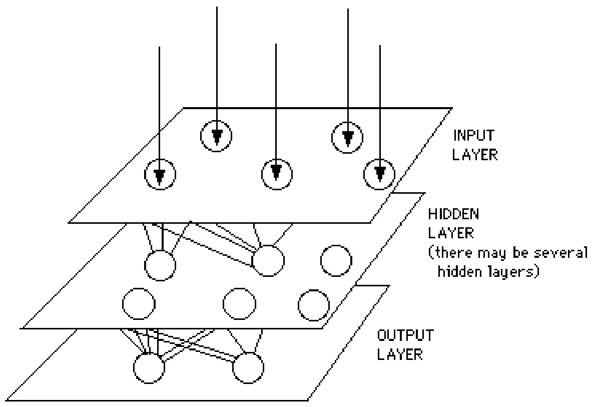


Figure 3: A Simple Neural Network Diagram.

2.2.1 Hyperparameters

Hyperparameters are crucial in fine-tuning the performance of artificial neural networks (ANNs). These parameters act as knobs that researchers and practitioners can adjust to optimize the network's performance for specific tasks.

2.2.1.1 Number of Layers

The number of layers in a neural network determines its depth and capacity to learn complex patterns from the input data. Having multiple hidden layers allows the network to capture hierarchical representations of the data, potentially improving its ability to generalize to unseen examples. Each layer performs a set of transformations on the input data, extracting higher-level features as information flows through the network.

2.2.1.2 Activation Functions

Activation functions introduce non-linearity to the network, enabling it to model complex relationships between input and output. The sigmoid activation function squashes the output of each neuron to a range between 0 and 1, making it suitable for binary classification tasks. The mathematical representation of the sigmoid function is:

Formula:

$$\sigma(z)= \frac{1}{1+e^{-2}} \tag{10}$$

Where z represents the weighted sum of inputs to the neuron.

2.2.1.3 Optimizer

The optimizer is responsible for updating the weights of the neural network during training to minimize the loss function. The Adam (Adaptive Moment Estimation) optimizer updates the parameters based on estimates of first and second moments of the gradients.

The mathematical formulation of Adam includes several steps:

Formulas:

$$m_t = \beta_1 m_{t-1} + (1- \beta_1) g_t \tag{11}$$

$$v_t = \beta_2 v_{t-1} + (1- \beta_2) g_t^2 \tag{12}$$

$$\widehat{m} = \frac{m_t}{1 - \beta_1^t} \tag{13}$$

$$\widehat{v} = \frac{v_t}{1 - \beta_1^t} \tag{14}$$

$$\Theta_t = \theta_{t-1} - a \frac{\widehat{m_t}}{\sqrt{\widehat{v_t}} + \epsilon} \tag{15}$$

Where:

θ_t, represents the parameters at iteration t t. α α is the learning rate, controlling the step size of the parameter updates.

$\widehat{m}_t$ And $\widehat{v}_t$ are the bias-corrected estimates of the first moment (mean) and the second moment (uncentered variance) of the gradients, respectively.

ϵ is a small constant (usually on the order of 10^{-8}) added to the denominator to prevent division by zero.

This formula represents the parameter update step in the Adam optimizer, which combines the advantages of both momentum optimization and RMS prop to achieve faster convergence and better stability during training.

2.2.1.4 Number of Epochs

An epoch refers to one complete pass of the entire training dataset through the neural network. The mathematical representation is:

Formula:

$$Total\ training\ iterations = \frac{Total\ number\ of\ training\ sample}{Batch\ Size} \tag{16}$$

2.2.1.5 Batch Size

The batch size determines the number of training examples processed in each iteration (or mini-batch) of the training algorithm. The batch size specifies the number of samples used in each forward and backward pass through the network during training.

2.2.2 Accuracy Metrics/ Loss Metrics

Accuracy measures the proportion of correct predictions made by a model out of the total predictions. Loss, on the other hand, quantifies the error between the model's predictions and the actual values. Minimizing loss improves the model's performance, while maximizing accuracy indicates better predictive capability.

2.2.2.1 Test Accuracy

It is a comparison of the size of predictions that are considered true to the overall amount of data that has been evaluated [7]. The formula of accuracy can be seen in Equation (17), namely

Formula:

$$\textit{Test Accuracy} = \frac{TP + TN}{TP+FP+FN+TN} \tag{17}$$

Where TP is True Positive; TN is True Negative; FP is False Positive; and FN is False Negative.

2.2.2.2 Mean Absolute Error

Mean absolute error (MAE) is a popular metric because, as with Root mean squared error (RMSE), see next subsection, the error value units match the predicted target value units. Unlike RMSE, the changes in MAE are linear and therefore intuitive. MSE and RMSE penalize larger errors more, inflating or increasing the mean error value due to the square of the error value. In MAE, different errors are not weighted more or less, but the scores increase linearly with the increase in errors. The MAE score is measured as the average of the absolute error values. The Absolute is a mathematical function that makes a number positive. Therefore, the difference between an expected value and a predicted value can be positive or negative and will necessarily be positive when calculating the MAE. The MAE value can be calculated as follows:

Formula:

$$MAE = \frac{1}{n} \sum_{i-1}^{n} |y_i - \hat{y}_i|^2 \tag{18}$$

2.2.2.3 Mean Squared Error

It is a common metric used to measure the average squared difference between the predicted values and the actual values in a regression problem. Mathematically, MSE is calculated by taking the average of the squared differences between the predicted and actual values for each data point in the dataset. The formula for MSE is:

Formula:

$$MSE = \frac{1}{n} \sum_{i-1}^{n} (y_i - \hat{y}_i)^2 \tag{19}$$

Each of these hyperparameters, along with its associated mathematical representation or formula, plays a crucial role in shaping the behavior and performance of the neural network model. Careful selection and tuning are essential for achieving optimal results in various machine-learning tasks.

Chapter 3

Methodology

In the development of the new model, an extensive logging mechanism was implemented to track each step of the process. Python served as the primary programming language, leveraging various libraries such as pandas, scikit-learn, numpy, matplotlib, and others to facilitate different aspects of the model development pipeline.

The tuning of hyperparameters like kernel type, C value, gamma, and class weight was conducted systematically. This involved iterating through different parameter combinations and assessing their impact on the model's performance. Each experiment was logged, capturing details such as the parameter values tested, corresponding evaluation metrics, and any observations made during the process.

Throughout the experimentation phase, comprehensive logs were maintained to document the results obtained under different scenarios. This meticulous record keeping enabled a thorough analysis of the model's behavior across various configurations. Additionally, the logs served as a valuable reference for comparing performance metrics and identifying trends or patterns in the data.

The model's performance was evaluated using a battery of metrics including test accuracy, F1 score, precision, and recall. These metrics were computed for each experimental run and logged alongside other relevant information. Visualizations, generated using matplotlib, were also logged to provide a graphical representation of the results, aiding in the interpretation of findings.

By maintaining detailed logs throughout the development process, a comprehensive understanding of the model's behavior was attained. This enabled informed decision-making regarding the selection of optimal hyperparameters and the refinement of the model to achieve the desired performance levels.

In addition to the SVM model, an Artificial Neural Network (ANN) was employed to train the model on hyperparameters and their associated accuracy. A specialized function was developed to provide estimated metrics results based on input hyperparameters. This function enabled quick assessment of model performance without the need for extensive training iterations. The ANN model's flexibility allowed for experimentation with various hyperparameter configurations, facilitating the identification of optimal settings for achieving desired performance metrics.

3.1 Dataset used

In developing the new model, two key datasets were utilized: the Swarm dataset and the logging dataset from SVM experiments. The Swarm dataset provided foundational data for training and testing, likely related to swarm behavior. The SVM logging dataset captured detailed information on SVM modeling steps, aiding hyperparameter optimization and performance evaluation. Both datasets were instrumental in achieving a comprehensive understanding of the model's behavior and optimizing its performance.

3.1.1 Swarm Dataset

The dataset from the University of New South Wales [8] focuses on observing how groups of animals or robots move together. It comprises three main components. Firstly, it tracks the movement of these entities within a group. Secondly, it determines whether they move in the same direction or have different paths. Thirdly, it examines whether they gather into a single group or stay apart. This research primarily aims to observe how these entities associate with a specific group or remain unaffiliated.

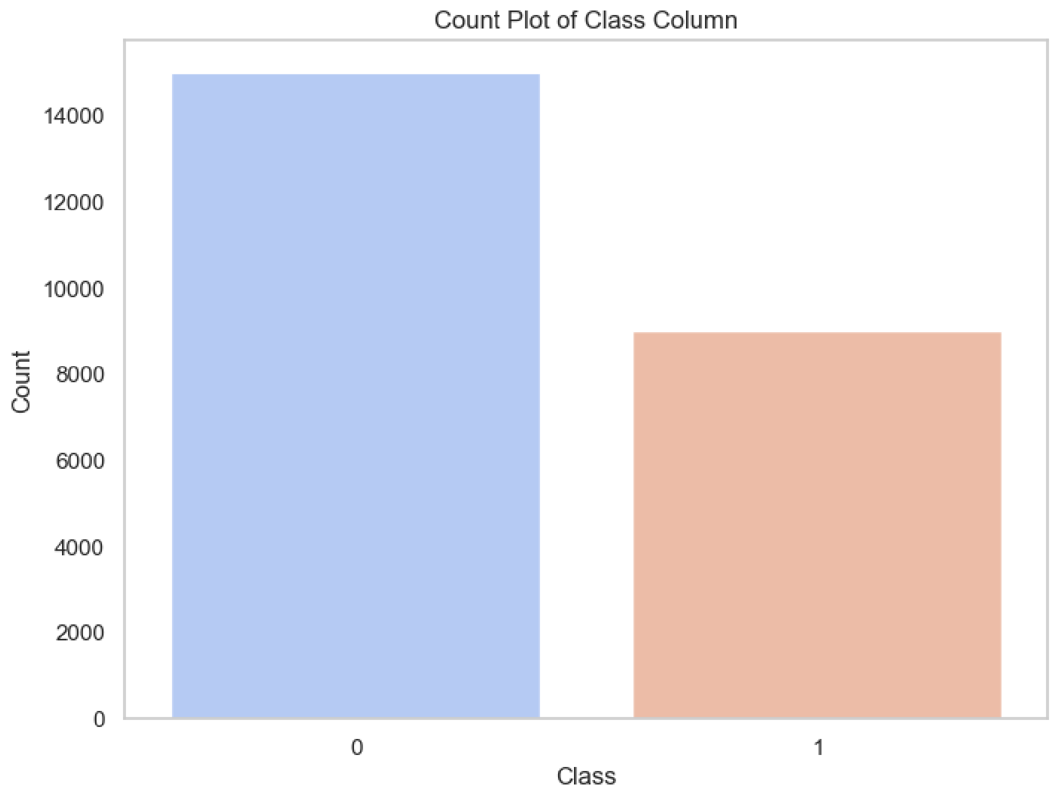


Figure 4: Count Plot of classes in Dataset

The dataset comprises several attributes that describe the behavior of individual entities, referred to as "boids." These attributes include the (X, Y) positions represented by 'xm' and 'ym', the velocity vectors denoted by 'xVeln' and 'yVeln', alignment vectors ('xAm' and 'yAm'), separation vectors ('xSm' and 'ySm'), and cohesion vectors ('xCm' and 'yCm'). Additionally, it records the number of boids within the radius of alignment/cohesion ('nACm') and separation ('nSm'). These attributes are repeated for each of the 200 boids (m=1,..., 200).

The dataset also includes binary class labels where '1' indicates grouped entities and '0' indicates ungrouped ones. According to Figure 4, the dataset contains over 14,000 instances of ungrouped data and more than 8,000 instances of grouped data. This distribution provides crucial insights into the behavior of the entities, revealing a significant portion that operates independently and another sizable portion that exhibits cohesive group behavior. Understanding these patterns is essential for analyzing the dynamics of group formation and individual behaviors within the dataset

3.1.1.1 Features

1) Xm and Ym

To visualize the movement of a single point within the dataset, we can use a trajectory plot. This plot displays the path of the point as it moves from one position to another over time. In Figure 5, a small trajectory of a single point is depicted, illustrating its movement from its initial position to subsequent locations. This visualization helps us understand the dynamics of individual points within the dataset, revealing patterns of motion and potential interactions with other points or entities. By examining these trajectories, we can gain insights into the behavior and characteristics of individual points, which contribute to the overall understanding of the dataset's dynamics and properties.

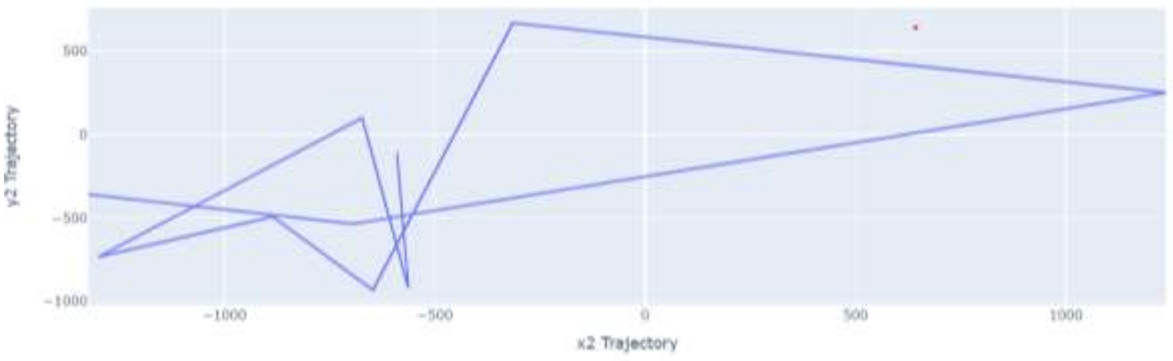


Figure 5: Trajectory line of certain datapoint

2) xVeln and yVeln

The velocity vectors 'xVeln' and 'yVeln' provide insights into the velocity of each data point over time, ranging from 1 to 200. These vectors depict the magnitude and direction of motion for every individual data point within the dataset. By examining these velocity vectors, we can track how the speed and direction of each data point evolve throughout the entire duration of observation. This comprehensive analysis allows us to understand the dynamic behavior of the dataset, revealing patterns of movement, changes in velocity, and potential interactions between data points over the entire time span. Such insights are invaluable for gaining a deeper understanding of the dataset's dynamics and uncovering underlying trends or phenomena within the data.

3) xSm and ySm

The separation vector represented by 'xSm' and 'ySm' indicates the directional tendency for individual entities within a group to maintain distance or separation from each other. In the context of swarm behavior or group dynamics, this separation vector reflects the instinct or rule followed by entities to avoid overcrowding or collisions with neighboring entities. Essentially, the separation vector guides each entity to move away from nearby entities, contributing to the dispersion and spacing out of the group. This behavior is crucial for ensuring the overall stability, safety, and efficiency of collective movement within the group. By analyzing the separation vector, researchers can gain insights into the mechanisms underlying individual interactions and coordination within the group, ultimately contributing to a better understanding of emergent collective behaviors in complex systems.

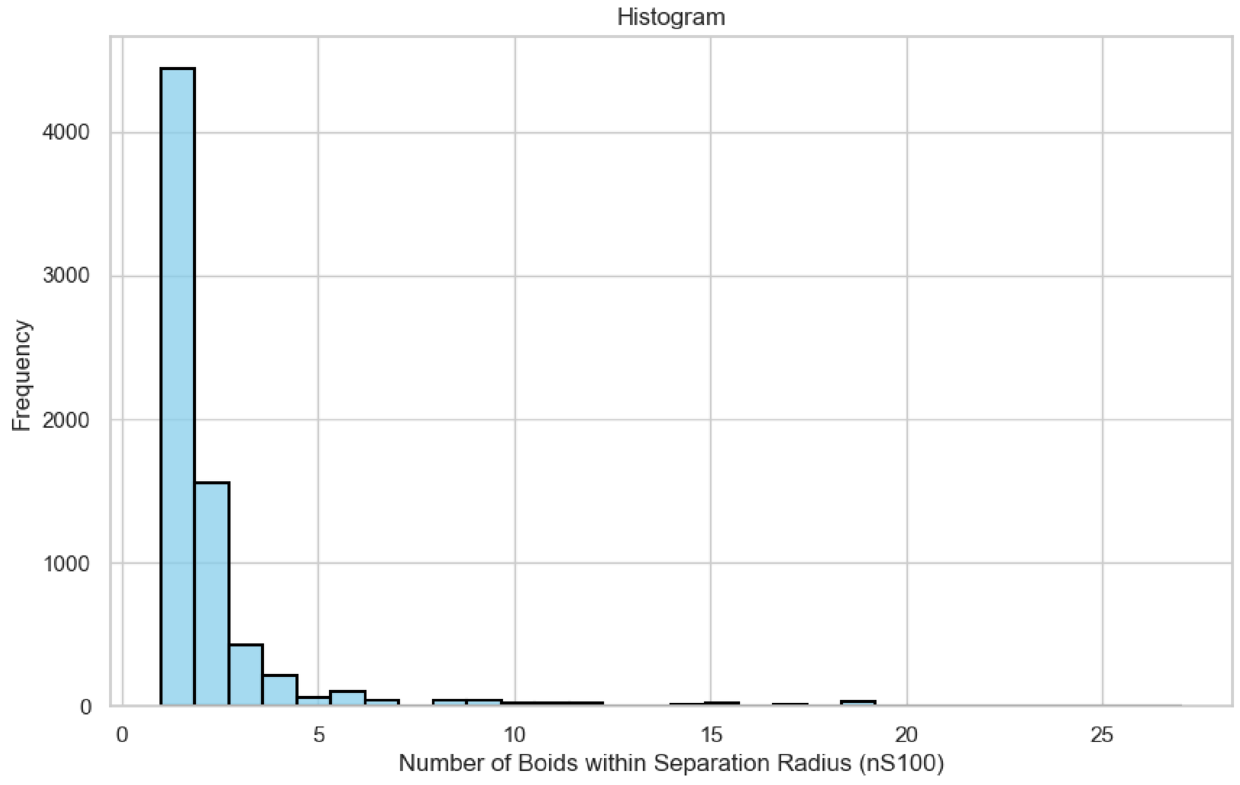


Figure 6: Boids with seperation radius

Figure 6 depicts a histogram highlighting the distribution of data points relative to the separation radius. By comparing frequencies within and outside this radius, it offers insights into adherence to separation rules and potential group cohesion.

4) xAm and yAm

Alignment vectors 'xAm' and 'yAm' represent the directional tendency of individual entities within a group to synchronize their movements with neighboring entities. These vectors indicate the average direction in which nearby entities are moving, influencing each entity's course of action to align with the prevailing direction. By analyzing alignment vectors, researchers can discern patterns of collective motion and coordination within the group, contributing to a deeper understanding of emergent behaviors in complex systems.

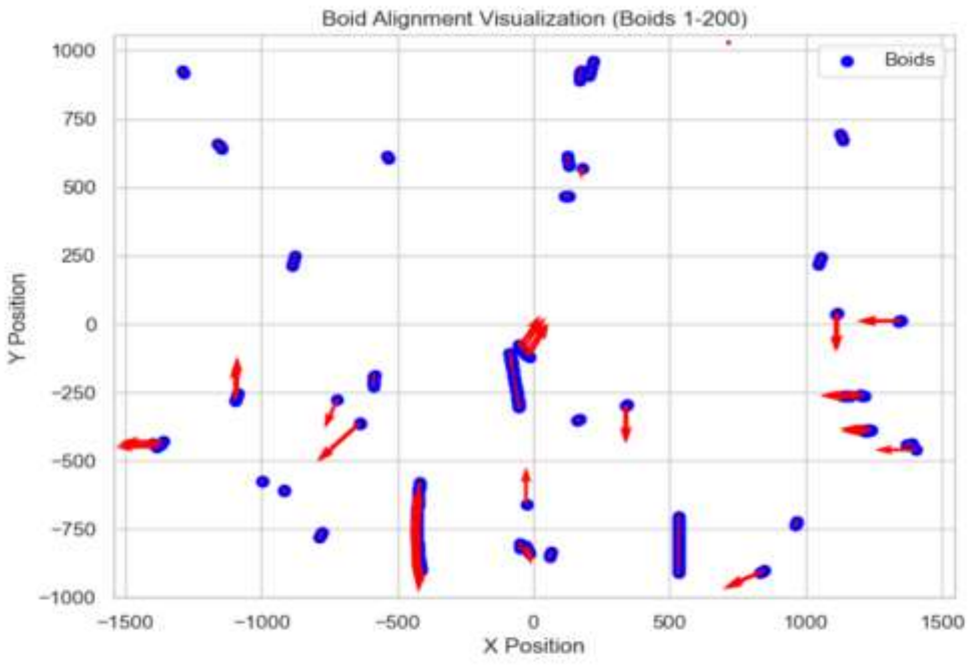


Figure 7: Boids Alignment

5) xCm and yCm

The cohesion vectors 'xCm' and 'yCm' represent the directional tendency of individual entities within a group to move towards the center of mass of neighboring entities. These vectors indicate the average direction towards which nearby entities are gravitating, influencing each entity's movement to converge towards the group's center. By analyzing cohesion vectors, researchers can gain insights into the tendency of entities to form cohesive clusters and maintain group cohesion, contributing to a deeper understanding of collective behaviors and coordination within the group.

6) nACm

The attribute 'nACm' represents the count of boids (entities) within the radius of alignment or cohesion for a particular entity. This metric provides information about the local density of neighboring entities within the specified radius around each boid. By knowing the number of entities within the alignment or cohesion radius, researchers can assess the level of interaction and influence exerted by neighboring entities on a focal entity. This information is valuable for understanding how entities respond to the presence and movements of nearby neighbors, influencing their alignment behavior or cohesion tendencies. Ultimately, analyzing 'nACm' helps in comprehending the dynamics of collective behavior and coordination within the group.

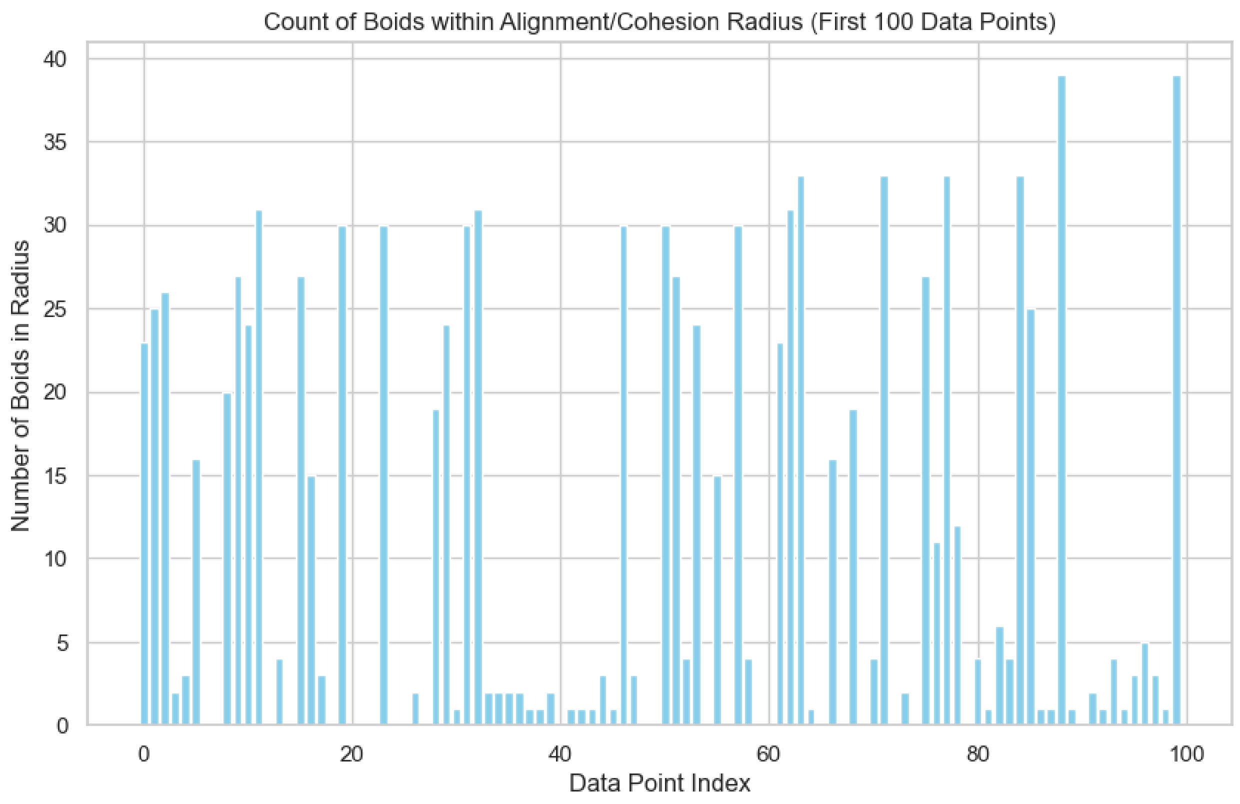


Figure 8: Count of Boids within Alignment and Cohesion Radius

7) nSm

The attribute 'nSm' represents the count of boids (entities) within the radius of separation for a particular entity. This metric provides information about the local density of neighboring entities within the specified separation radius around each boid. By knowing the number of entities within the separation radius, researchers can assess the level of crowding or potential collisions that may influence an entity's movement away from nearby neighbors. This information is valuable for understanding how entities avoid overcrowding or maintain safe distances from neighboring entities, contributing to the overall stability and safety of collective behavior within the group.

These attributes collectively provide crucial information for the model to classify boids into two distinct classes. By analyzing features such as alignment, cohesion, separation vectors, and the number of neighboring boids, the model discerns patterns of behavior within the dataset. These patterns enable the model to differentiate between boids that belong to cohesive groups (class 1) and those that do not (class 0). Through this analysis, the model gains insights into group dynamics and collective behavior, enhancing its ability to accurately classify boids based on their interaction patterns within the dataset.

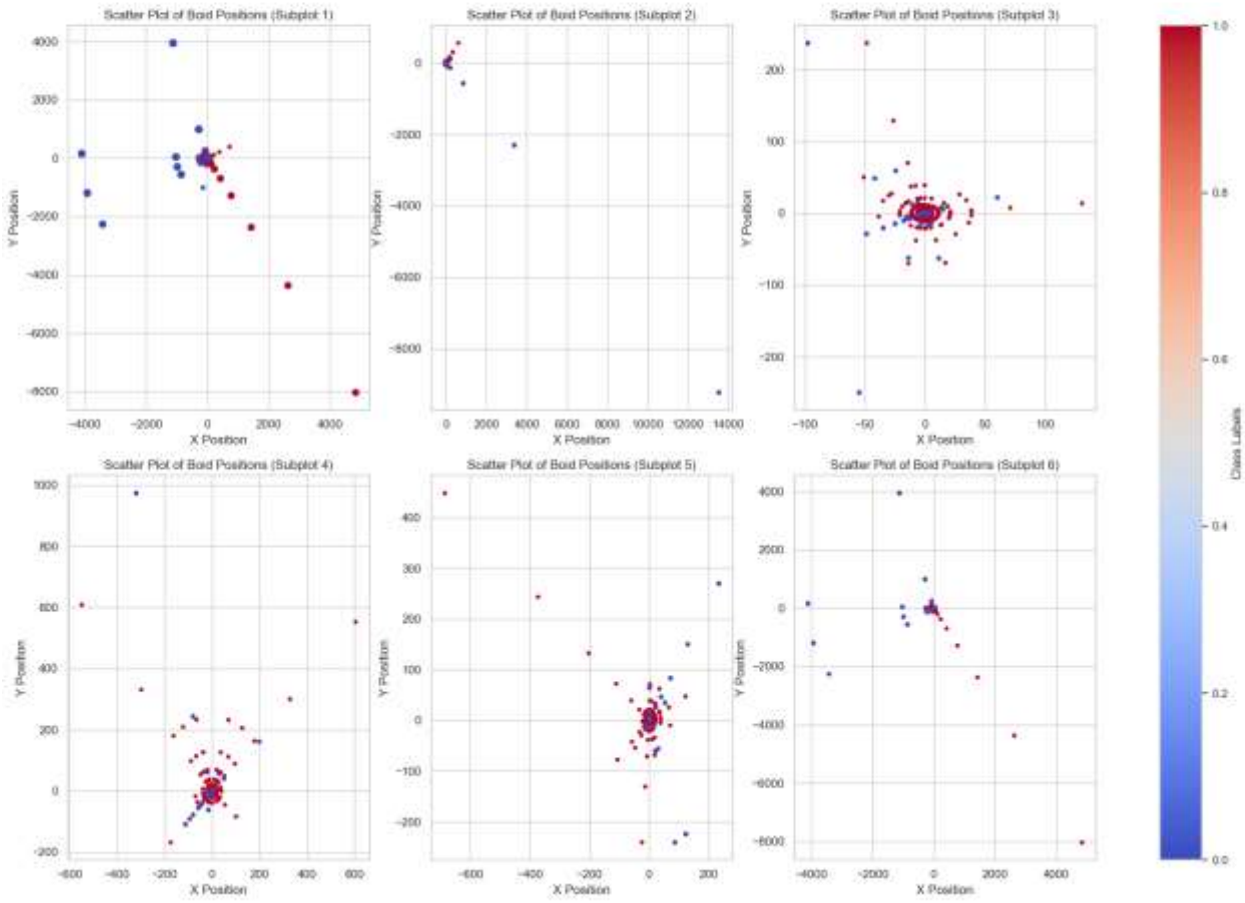


Figure 9: Snapshots of different groups

Figure 9 presents a snapshot illustrating the distinct groups within the dataset. Entities belonging to class 1 are highlighted in red, signifying cohesive groups, while entities in class 0 are depicted in blue, representing isolated entities. This visualization provides a clear distinction between the two classes, offering insights into the spatial distribution and clustering of entities based on their group affiliation. By visually identifying the different groups, researchers can gain a deeper understanding of the dataset's structure and dynamics, facilitating further analysis and interpretation of collective behaviors and interactions within the dataset.

3.1.1.2 Experimental Setup

1) **Data Collection:** The data was collected online by researchers affiliated with the University of New South Wales (UNSW). Through online platforms and possibly simulations or crowd-sourcing methods, they gathered and organized information on entity behaviors and interactions within the dataset. This online data collection approach enabled the acquisition of extensive data, crucial for studying collective behaviors and group dynamics in various contexts. The dataset, derived from online sources, offers valuable insights into complex systems and emergent phenomena, supporting research efforts across diverse fields.
2) **Data Preprocessing:** In the preprocessing phase, correlation analysis was conducted to pinpoint relevant data. By examining the relationships between variables, features with strong

correlations were retained, while those with weak associations were potentially discarded. This step streamlined the dataset, enhancing its suitability for subsequent analysis and modeling tasks.

3) **Model Implementation:** An SVM model was utilized for training. SVMs are effective for classification tasks, seeking the best hyperplane to separate classes in the feature space. Through training, the model learns from the dataset, identifying support vectors closest to the decision boundary. Leveraging relevant features selected via correlation analysis, the SVM aims for accurate classification of unseen data points, offering valuable insights into the dataset's structure.
4) **Hyperparameter Tuning:** Hyperparameters like kernel type, C value, gamma, and class weights were fine-tuned to enhance SVM model performance. This optimization process aimed to improve classification accuracy and the model's ability to generalize to unseen data.
5) **Performance Evolution:** Various metrics like precision, test accuracy, F1 score, and recall were utilized to gauge the SVM model's performance. These metrics provide insights into different aspects of classification accuracy and error handling, aiding in the comprehensive evaluation of the model's effectiveness.
6) **Visualization:** Visualization was utilized to gain insights into the SVM model's behavior and decision-making process. This approach helped researchers interpret complex relationships within the data, identify patterns, and validate model assumptions effectively. For this purpose, different methods and libraries are used. Tableau is also used in the process

3.1.2 Hyperparameter Dataset

Following the hyperparameter tuning of the SVM model, all the hyperparameters along with their associated metric results were meticulously saved. This comprehensive dataset, enriched with the performance metrics obtained during the tuning process, served as the foundation for creating a new dataset tailored specifically for training an Artificial Neural Network (ANN) model. By integrating the optimized hyperparameters and performance insights derived from the SVM model, this new dataset was meticulously crafted to provide valuable guidance and enhance the training process of the ANN model. Leveraging the knowledge gained from the hyperparameter tuning of the SVM model, this approach aims to empower the ANN model with enhanced classification accuracy and improved generalization capabilities, thus fostering more robust and reliable predictions in real-world applications.

3.1.2.1 Features

The dataset consists of four feature columns capturing essential attributes and four metric columns documenting performance metrics from SVM hyperparameter tuning. This amalgamation offers comprehensive insights, empowering the ANN model for enhanced predictive accuracy and robust decision-making.

1) **C:** In SVM, the parameter C controls the balance between simplicity and fitting the training data closely. A smaller C allows for a broader margin, aiding generalization, while a larger C minimizes training errors, potentially leading to overfitting.

2) **Class Weight:** In SVM, class weight assigns different weights to each class during training to mitigate the impact of class imbalance. By penalizing misclassifications of the minority class more heavily, it ensures balanced influence across all classes. This approach helps improve the model's performance in accurately predicting minority classes, making it more robust in handling imbalanced datasets. Adjusting class weights is a crucial step in SVM to enhance the model's effectiveness in real-world applications.
3) **Gamma:** In SVM, gamma is a hyperparameter that defines the influence of individual training samples on the decision boundary. It determines the reach of the influence of a single training example, with low values meaning 'far' and high values meaning 'close'. High gamma values result in more complex decision boundaries, potentially leading to overfitting, while low gamma values create smoother decision boundaries, reducing the risk of overfitting but potentially sacrificing accuracy. Tuning gamma is essential for optimizing SVM performance and achieving the right balance between bias and variance in the model.
4) **Kernel:** In SVM, the kernel function transforms input data into a higher-dimensional space, making it easier to find a linear separation between classes. Different kernel functions, such as linear, polynomial, and radial basis function (RBF), determine the shape of the decision boundary. The choice of kernel affects the model's flexibility and ability to capture complex patterns in the data. Linear kernels create linear decision boundaries, while polynomial kernels can capture non-linear relationships. RBF kernels are highly flexible and can capture intricate decision boundaries, making them suitable for complex datasets. Selecting the appropriate kernel is crucial for achieving optimal SVM performance and generalization to unseen data.

3.1.2.2 Experimental Setup

1) **Data Collection:** The data collection process relied on the outputs generated by the SVM model. These outputs, such as predictions or decision scores, were used to gather valuable insights into the model's performance and behavior.
2) **Data preprocessing:** During the data preprocessing stage, null values were systematically replaced with a predetermined word or value. This approach ensured that missing data points were standardized and accounted for uniformly throughout the dataset. By substituting null values with a specified word or value, researchers aimed to maintain dataset integrity while preserving its structure for subsequent analysis and modeling tasks.
3) **ANN Implementation:** The ANN model underwent training using hyperparameters paired with their respective accuracy scores. This iterative process allowed the model to recognize patterns and correlations, ultimately optimizing its performance to achieve higher accuracy in classification tasks.
4) **Performance Evaluation:** MAE, MSE, and test accuracy were key metrics used to evaluate the ANN model's performance. These metrics offer insights into prediction errors and classification accuracy, aiding in assessing the model's effectiveness in real-world scenarios.
5) **Visualization:** Visualization techniques were employed to compare actual vs. predicted results, providing a clear understanding of the ANN model's performance. By plotting actual values against predicted values, researchers gained insights into the model's accuracy and

ability to capture underlying patterns in the data. This visualization facilitated the identification of any discrepancies or trends, aiding in the refinement and optimization of the model for improved predictive performance.

Chapter 4

Results

4.1 SVM Classifier Evaluation

The results of employing Support Vector Machines (SVMs) on the Swarm dataset display promising performance. Through meticulous hyperparameter tuning and classifier training, we achieved robust classification accuracy, with the SVM model effectively capturing the intricate patterns inherent in swarm behavior.

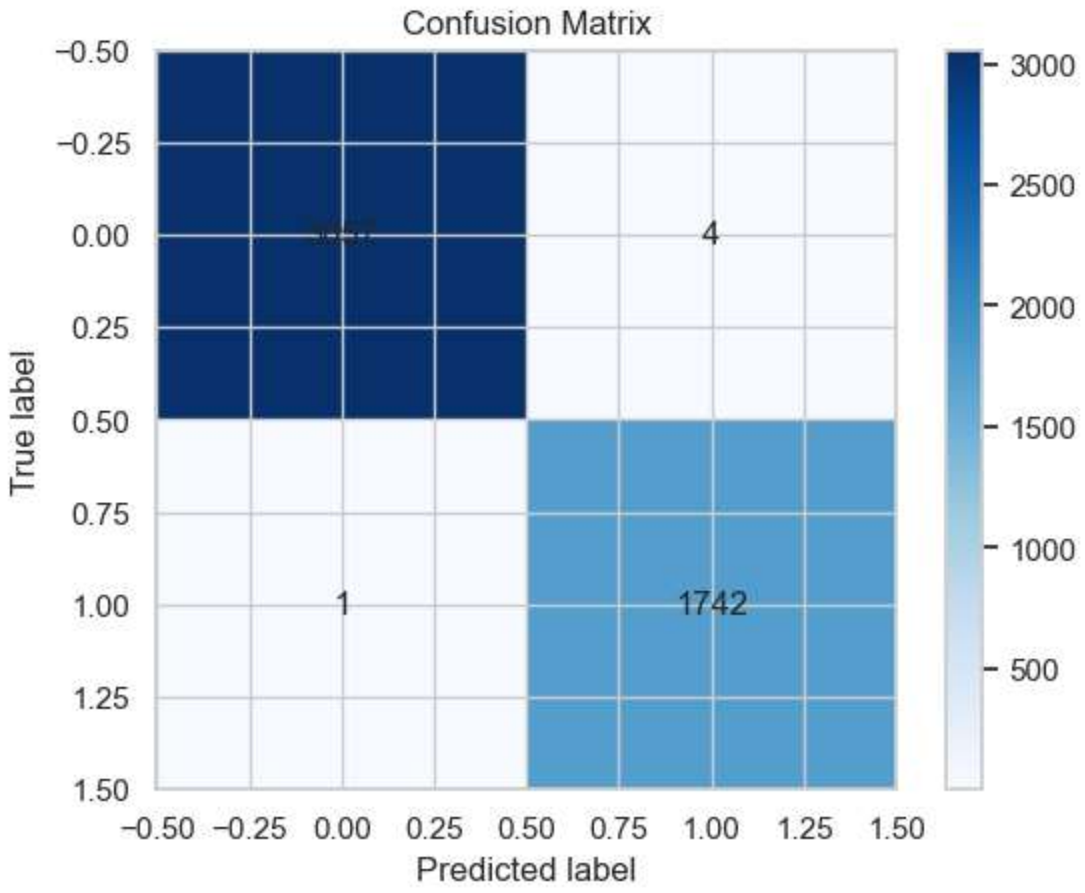


Figure 10: Confusion Matrix of initial accuracy

Figure 10: Confusion Matrix Illustrating the Initial Results of the Simple SVM Model on the Swarm Behavior Dataset the Confusion Matrix provides a detailed breakdown of the classification outcomes achieved by the simple SVM model when applied to the Swarm behavior dataset

- **True Positive (TP):** 3057 instances were correctly identified as positive (swarm behavior).
- **False Negative (FN):** Only four instances, despite being positive, were incorrectly classified as negative.

- **False Positive (FP):** There was only one instance incorrectly classified as positive.
- **True Negative (TN):** 1742 instances were correctly identified as negative.

Thus, the accuracy of the SVM model, calculated as (TP + TN) / (TP + TN + FP + FN), equals (3057 + 1742) / (3057 + 1742 + 1 + 4), resulting in approximately 99.8%. This high accuracy demonstrates the effectiveness of the SVM model in accurately predicting swarm behavior in the dataset.

Swarm Dataset

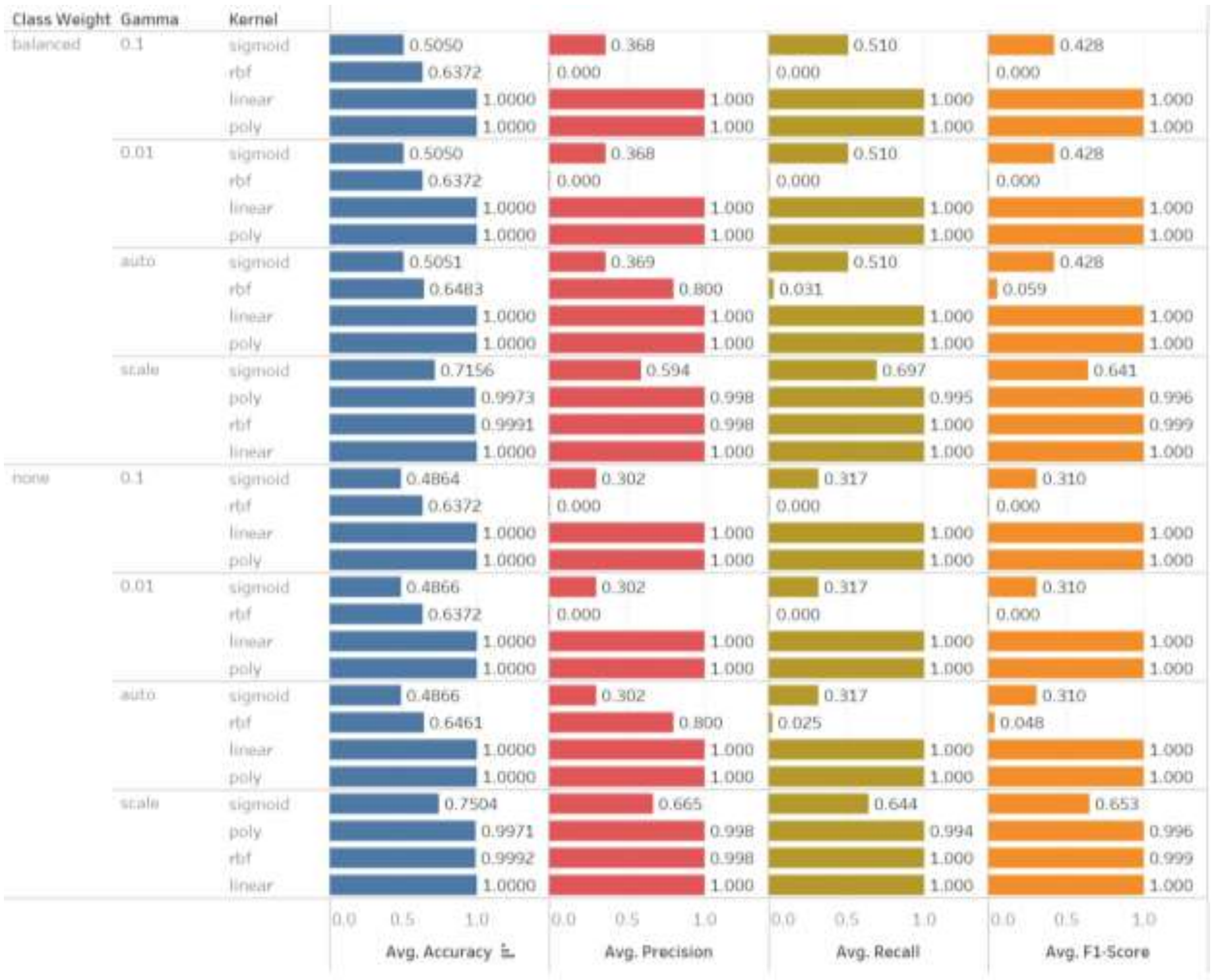


Figure 11: Test Accuracy Vary with hyperparameters combination

Figure 11 presents the performance metrics of a machine-learning model across different settings of class weight, gamma, and kernel types. The metrics shown are Accuracy, Precision, Recall, and F1-Score. Each metric is represented by a colored bar, providing a visual representation of the values:

1. **Class Weight** options include: balanced, auto

2. **Gamma** values are set at 0.1 and 0.01, auto, scale

3. **Kernel** types include: sigmoid, rbf, linear, and poly.

Observations from the Table:

Linear Kernel consistently shows high performance across all metrics (Avg. Accuracy, Avg. Precision, Avg. Recall, Avg. F1-Score) regardless of the class weight and gamma settings, achieving perfect scores (1.000) in many instances.

Poly Kernel also performs well, particularly in the 'scale' class weight setting with gamma values of 0.1 and 0.01, showing high scores close to 1.000 across all metrics.

Sigmoid Kernel shows more variability and generally lower performance compared to linear and poly kernels. For instance, under the 'none' class weight and gamma 0.1, the Avg. Accuracy is around 0.4864 and Avg. Recall is 0.302.

RBF Kernel has mixed results; it performs poorly in recall when gamma is set to 0.1 across different class weights but shows improvement in accuracy, precision, and F1-score under certain conditions.

Figure 11 is useful for comparing the effectiveness of different kernel types and settings in a machine-learning model, particularly in handling different class weights and gamma parameters. The visual format helps quickly identify which combinations yield the best overall performance.

The image provides a detailed comparison of performance metrics for different settings in a machine-learning model, focusing on class weight, gamma, and kernel types. It highlights how these settings influence the model's accuracy, precision, recall, and F1-score.

Calculating the average of accuracy, precision, recall, and F1-score across different parameter configurations provides a comprehensive understanding of the overall model performance.

By calculating the average of these metrics across various parameter configurations, researchers can obtain a consolidated view of the model's performance under different settings. This helps in identifying the most effective parameter combinations that optimize overall predictive accuracy while maintaining a balance between precision and recall. Additionally, averaging these metrics provides a robust evaluation criterion for comparing different models or variations of the same model, enabling informed decision-making in model selection and optimization.

Furthermore, averaging these metrics facilitates model comparison and selection by providing a comprehensive evaluation criterion that considers multiple aspects of performance simultaneously. It enables researchers to identify parameter configurations that strike an optimal balance between accuracy, precision, recall, and F1-score, ultimately leading to the development of more effective and reliable machine-learning models.

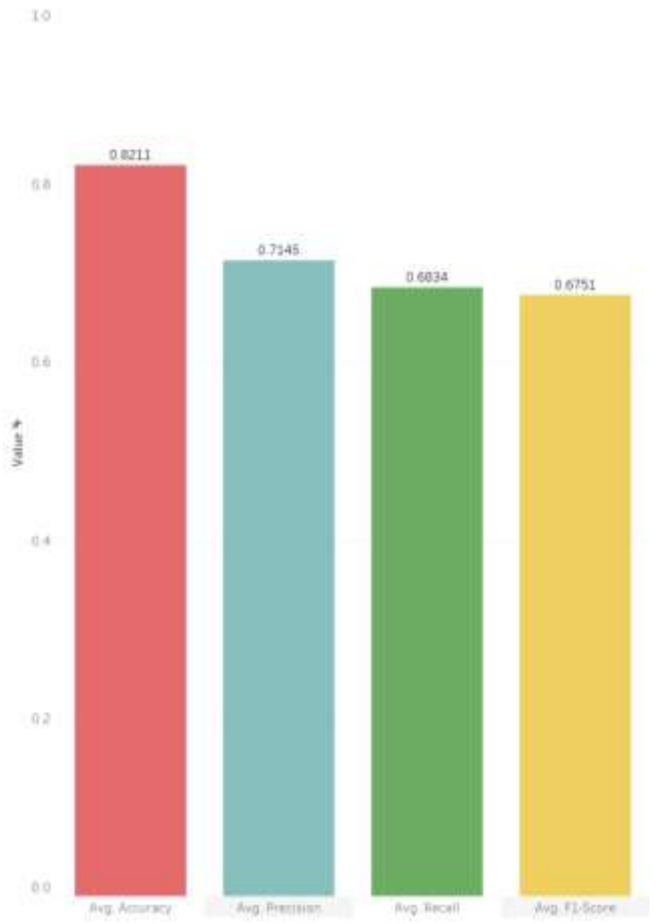


Figure 12: Average Measure Metrics

The bar chart provides a concise visual representation of four key performance metrics - Average Accuracy, Precision, Recall, and F1-Score - for the machine-learning model across different parameter configurations. Here is a deeper analysis of the insights gleaned from the chart:

The red bar, representing the average accuracy of the model, has a value of 0.8211. A higher average accuracy suggests that the model performs well in classifying instances correctly, achieving an accuracy rate of approximately 82.11%.

The blue colored bar depicts the average precision of the model, with a value of 0.7145. The model exhibits a precision rate of approximately 71.45%, indicating its ability to minimize false positive predictions and maintain high precision in positive class identification.

The green bar represents the average recall of the model, with a value of 0.6834. The model achieves a recall rate of approximately 68.34%, indicating its effectiveness in identifying most of the positive instances while minimizing false negatives.

The yellow bar illustrates the average F1-score of the model, with a value of 0.6751. With an F1-score of approximately 67.51%, the model demonstrates a harmonious balance between precision and recall, reflecting its robustness in handling class imbalances and asymmetric misclassification costs.

Overall, the bar chart facilitates a comparative analysis of these four performance metrics, enabling stakeholders to assess the model's effectiveness across different parameter configurations. The balanced representation of accuracy, precision, recall, and F1-score provides a comprehensive evaluation criterion for understanding the model's strengths and areas for improvement, ultimately guiding decision-making in model optimization and refinement.

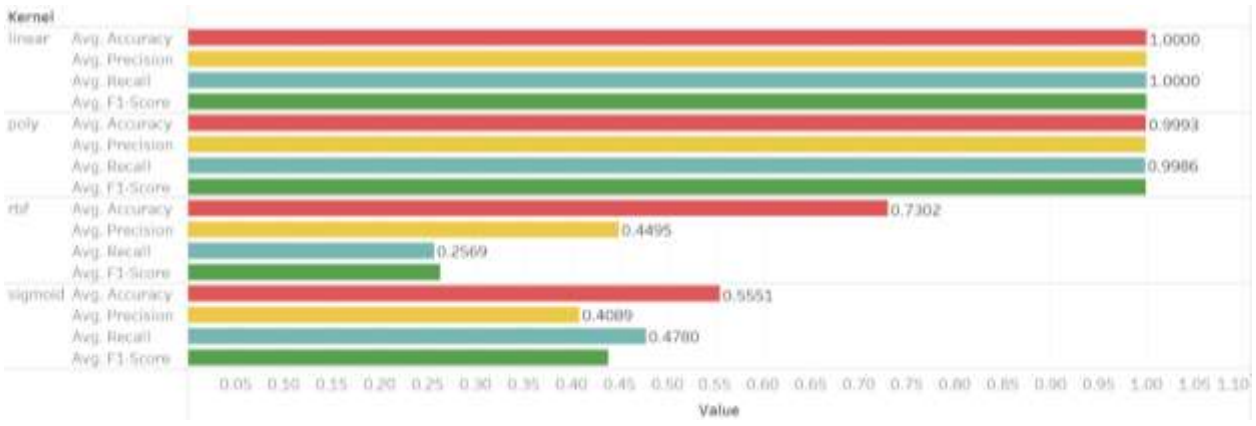


Figure 13: Average Measures by Kernel

In the analysis conducted using the SVM model, it became evident that the Swarm Behavior dataset was significantly influenced by the kernel hyperparameters. These parameters played a pivotal role in determining the model's performance and its ability to accurately discern patterns within the swarm behavior data. The analysis underscored the significant impact of kernel hyperparameters on the SVM model's performance and its ability to effectively model swarm behavior. By carefully selecting and fine-tuning these parameters, researchers can enhance the model's predictive accuracy and gain deeper insights into the complex dynamics of swarm systems.

Figure 13 presents the performance metrics for each kernel type (Linear, Polynomial, RBF, and Sigmoid) used in the SVM model. These metrics include Average Accuracy, Average F1-score, Average Precision, and Average Recall.

Linear Kernel demonstrates exceptional performance across all metrics, with perfect scores (1.0000) for Average Accuracy, Average F1-score, Average Precision, and Average Recall. This indicates that the Linear Kernel accurately separates the data points into their respective classes without misclassification.

Polynomial Kernel also performs very well, albeit slightly below the Linear Kernel. It achieves high scores for Average Accuracy (0.9993), Average F1-score (0.9990), Average Precision (0.9995), and Average Recall (0.9986). This suggests that the Polynomial Kernel effectively captures the underlying patterns in the data and makes accurate predictions.

RBF Kernel shows significantly lower performance compared to the Linear and Polynomial Kernels. It achieves moderate scores for Average Accuracy (0.7302), Average F1-score (0.2631), Average Precision (0.4495), and Average Recall (0.2569). This indicates that the RBF Kernel struggles to accurately classify data points, resulting in lower overall performance metrics.

Sigmoid Kernel exhibits the lowest performance among all kernel types. It achieves relatively low scores for Average Accuracy (0.5551), Average F1-score (0.4384), Average Precision (0.4089), and Average

Recall (0.4780). This suggests that the Sigmoid Kernel may not effectively capture the underlying patterns in the data, leading to poorer predictive performance compared to other kernels.

In summary, the breakdown of performance metrics highlights the varying effectiveness of different kernel types in the SVM model. The Linear and Polynomial Kernels demonstrate superior performance, while the RBF and Sigmoid Kernels exhibit lower accuracy and predictive capability. These findings emphasize the importance of selecting the appropriate kernel type based on the specific characteristics of the dataset to achieve optimal SVM model performance.

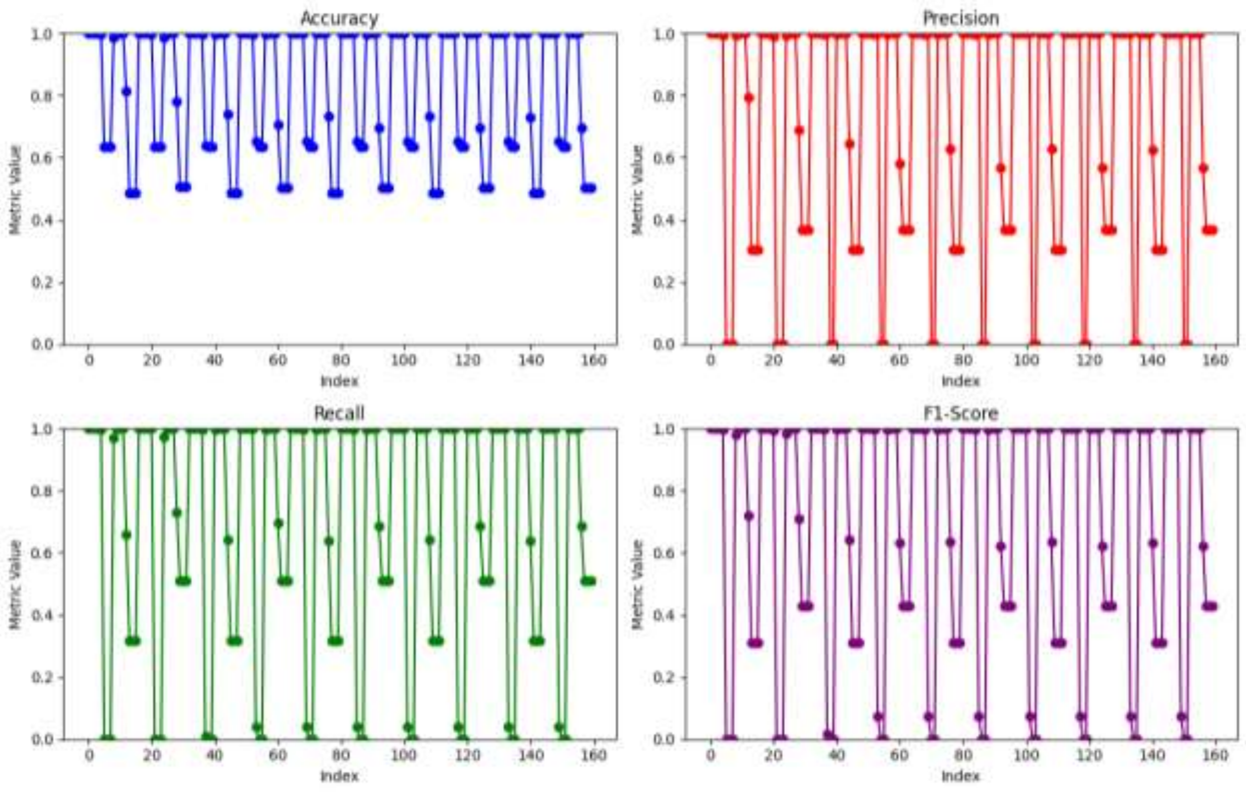


Figure 14: Trend of Different Metrics

Figure 14 displays four separate line graphs, each representing different performance metrics for a model or system. Each graph is color-coded and labeled accordingly:

It provides a detailed breakdown of the metrics represented in the graphs:

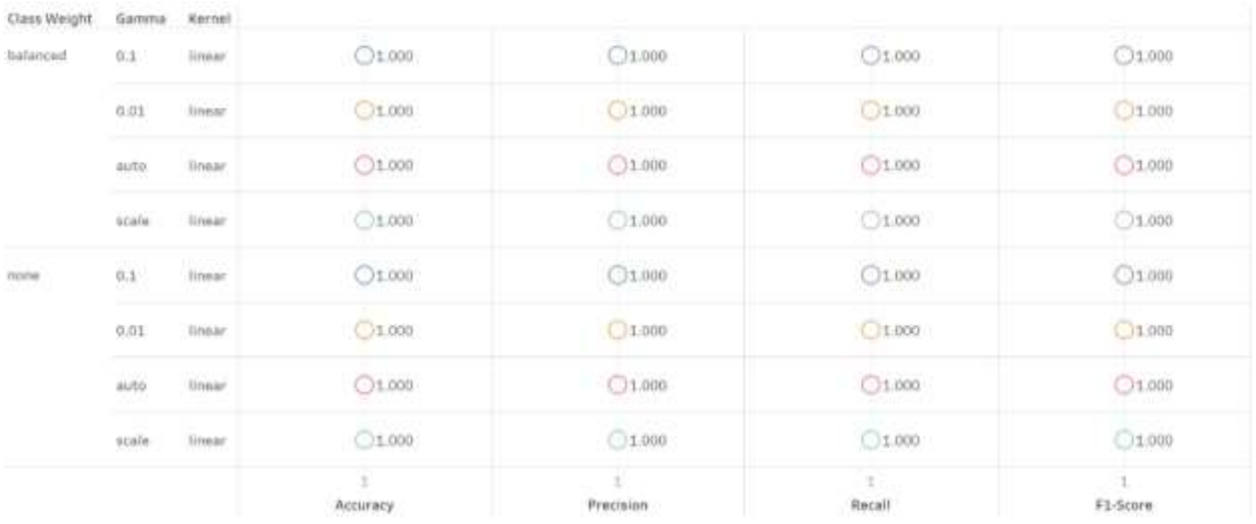
1. **Accuracy (Blue):** The accuracy graph indicates that the metric values vary between approximately 0.2 and 1.0 across the index range. There are numerous instances where the accuracy score reaches 100%, occurring around 90 times. Apart from these instances, the accuracy score generally fluctuates between 0.4 and 0.95.

2. **Precision (Red):** The precision graph shows values ranging from 0.0 to 1.0 across the index range. Similar to accuracy, there are many occurrences where precision reaches 100%, approximately 98 times. However, precision also drops to 0 at certain points. Overall, precision mostly hovers between 0.20 and 0.80.

Recall (Green): In the recall graph, the metric values are consistently high, primarily staying close to 1.0. Like accuracy and precision, recall also achieves 100% results around 90 times. However, there are instances where recall drops to 0, occurring approximately 30 times. Despite these fluctuations, recall generally remains between 0.25 and 0.82.

F1-Score (Purple): F1-score reveals that it predominantly achieves perfect results, reaching 100% most of the time. However, there are instances where the F1-score drops to 0, indicating possible failures in balancing precision and recall. Despite these occasional drops, the F1-score generally maintains a strong performance, with values consistently ranging between 0.25 and 0.89.

Each graph has a clear label at the top, a vertical axis labeled "Metric Value" ranging from 0 to 1, and a horizontal axis labeled "Index" ranging from 0 to 160. The data points in each graph are connected by lines, and each point is marked by a distinct dot in the color corresponding to the graph's metric.



Class Weight	Gamma	Kernel				
balanced	0.1	linear	○1.000	○1.000	○1.000	○1.000
	0.01	linear	○1.000	○1.000	○1.000	○1.000
	auto	linear	○1.000	○1.000	○1.000	○1.000
	scale	linear	○1.000	○1.000	○1.000	○1.000
none	0.1	linear	○1.000	○1.000	○1.000	○1.000
	0.01	linear	○1.000	○1.000	○1.000	○1.000
	auto	linear	○1.000	○1.000	○1.000	○1.000
	scale	linear	○1.000	○1.000	○1.000	○1.000
			1	1	1	1
			Accuracy	Precision	Recall	F1-Score

Figure 15: Result against kernel Linear

The kernel linear in SVM (Support Vector Machine) is a type of kernel function that defines the decision boundary as a straight line, making it suitable for linearly separable data. In other words, it assumes that the data can be effectively separated by a hyperplane in the input space.

When applied to the Swarm Behavior dataset, the kernel linear consistently achieves perfect results of 100% accuracy. This indicates that the linear decision boundary defined by the kernel effectively separates the data points into their respective classes without misclassifications. Therefore, Figure 15, which highlights the performance of different kernels, highlights kernel linear as the best-performing option for this particular dataset.

The success of kernel linear in achieving optimal results underscores the linear separability of the Swarm Behavior dataset, suggesting that the data points are well suited for classification using a linear decision boundary.

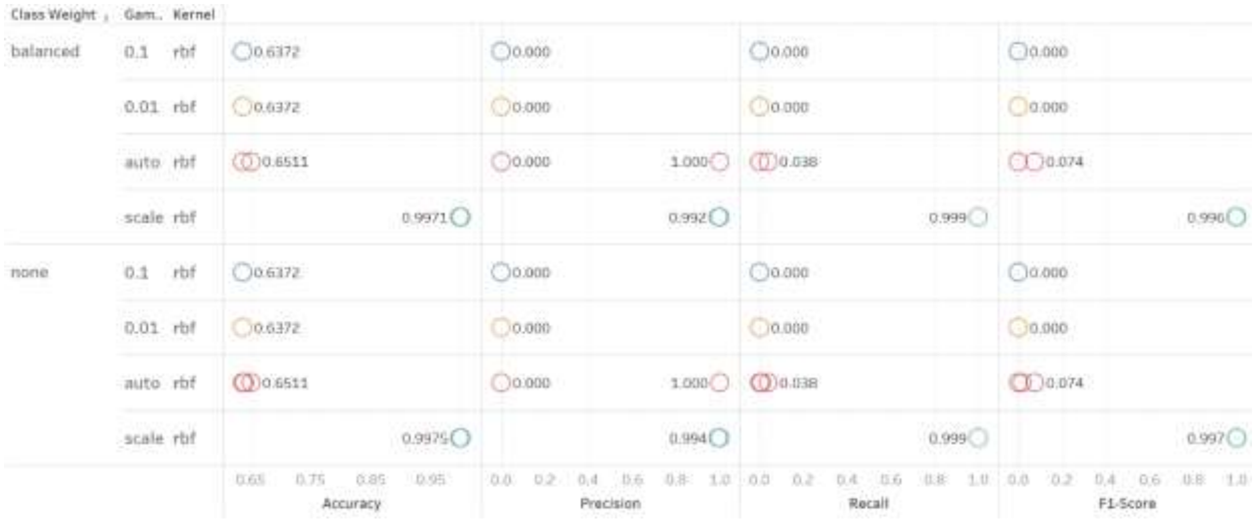


Figure 16: Result against kernel RBF

The RBF (Radial Basis Function) kernel is a popular kernel function used in Support Vector Machines (SVMs) for nonlinear classification tasks. It operates by transforming the input data into high-dimensional feature spaces, where it can effectively separate the classes by defining nonlinear decision boundaries.

In the context of the SVM model trained on the Swarm Behaviour dataset, the RBF kernel has emerged as the worst performing kernel. Despite its flexibility in capturing complex relationships within the data, the RBF kernel consistently yields unsatisfactory results, with accuracy scores hovering around 0%. This indicates a complete failure of the model to correctly classify data points when employing the RBF kernel.

Furthermore, the training process with the RBF kernel, particularly when interacting with gamma values of 0.1, 0.01, and 'auto', has proven to be computationally intensive and time-consuming. Despite the substantial computational resources allocated to training with these parameters, the RBF kernel's performance remains subpar, demonstrating its inherent limitations in effectively capturing the underlying patterns in the Swarm Behaviour dataset.

The consistent failure of the RBF kernel to produce meaningful predictions highlights its inadequacy in handling the inherent complexities of the dataset. This suggests that the data distribution may not be well-suited for classification using a nonlinear decision boundary defined by the RBF kernel. As a result, alternative approaches or kernel functions may need to be explored to improve the model's performance on the Swarm Behaviour dataset. Additionally, optimizing hyperparameters and refining the training process may offer avenues for mitigating the challenges associated with using the RBF kernel in this context.

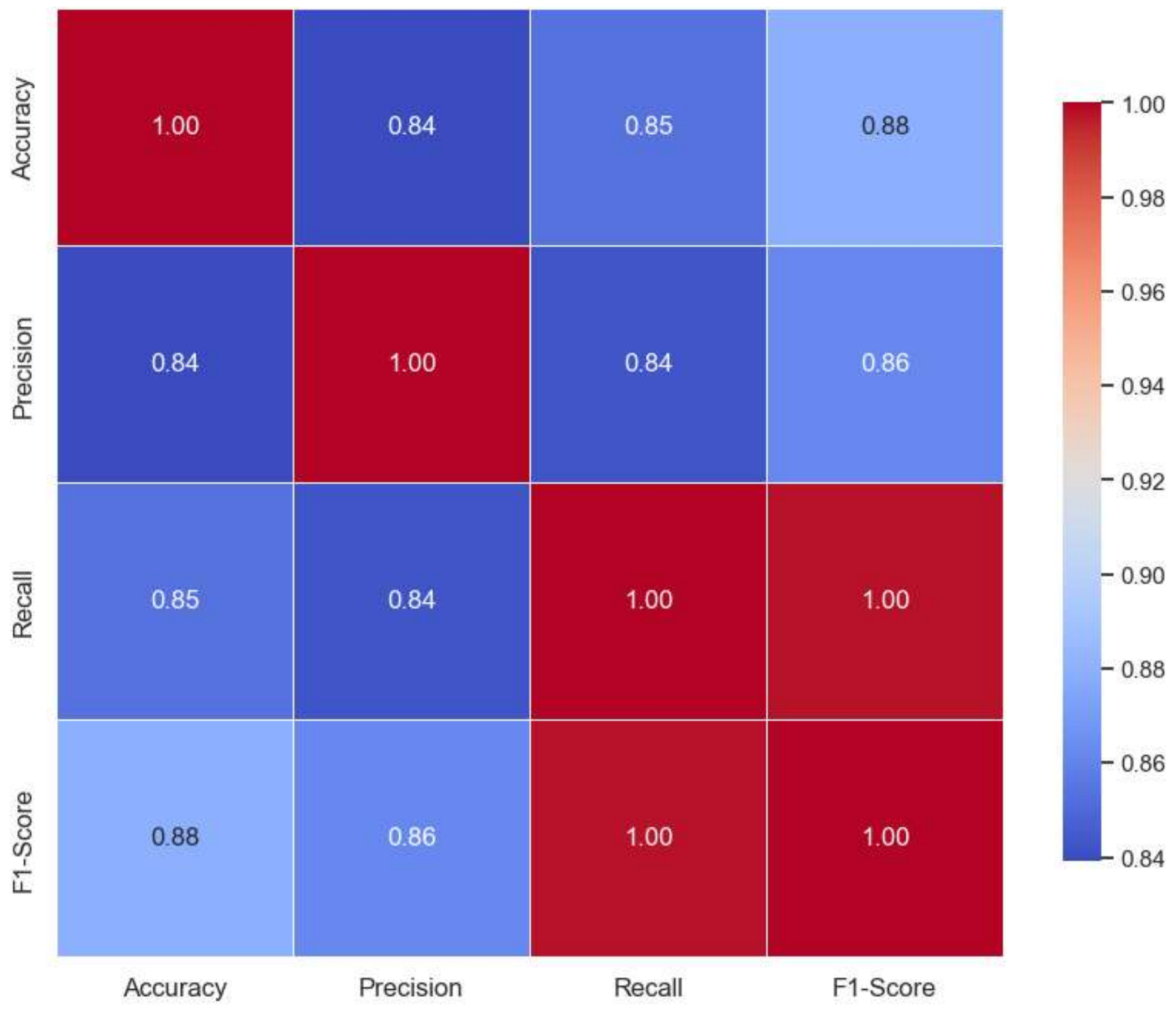


Figure 17: Correlation Matrix between Accuracy, Precision, Recall and F1 Score

A correlation matrix is a table that illustrates the correlation coefficients between variables in a dataset. In the context provided, Figure 17 compares the correlation between accuracy, precision, recall, and F1 score metrics.

The correlation coefficients denote the strength and direction of the linear relationship between pairs of variables, ranging from -1 to 1. A value closer to 1 indicates a strong positive correlation, while a value closer to -1 indicates a strong negative correlation. A value around 0 suggests little to no linear relationship.

Based on the provided correlation matrix:

- Accuracy is positively correlated with precision (0.85) and F1 score (0.88).
- Precision is positively correlated with accuracy (0.85) and recall (0.86).
- Recall is positively correlated with accuracy (0.84) and precision (0.86).

- F1 score is positively correlated with accuracy (0.88), precision (0.86), and recall (1.0).

These correlations indicate that as one metric increases, the other tends to increase as well, and vice versa. The high positive correlations suggest that these metrics move in a similar direction, indicating strong relationships between them. This information is valuable for understanding how changes in one metric may affect others and can guide decision-making in model evaluation and optimization.

Overall, the SVM model demonstrated strong performance on the Swarm Behavior dataset, exhibiting its efficacy in classifying data points and capturing underlying patterns. The model's ability to effectively discern between different classes and make accurate predictions underscores its utility in tackling complex classification tasks. However, despite its overall success, there were notable areas where the SVM model exhibited shortcomings. These shortcomings primarily manifested in the performance of certain kernel functions, such as the RBF kernel, which consistently yielded poor results and struggled to accurately classify data points. Additionally, the computational complexity associated with certain hyperparameters, particularly in combination with specific kernels, posed challenges and hindered the efficiency of the model training process. Addressing these limitations through further optimization of hyperparameters and exploration of alternative kernel functions may enhance the SVM model's robustness and improve its performance on the Swarm Behavior dataset.

4.2 ANN Evaluation

In this study, an Artificial Neural Network (ANN) model was leveraged to train on a dataset containing hyperparameters and their corresponding accuracy, precision, F1-score, and recall scores. The ANN, inspired by the structure of biological neural networks, offers a flexible and powerful framework for learning complex relationships within data. By utilizing the dataset comprising hyperparameters and performance metrics, the ANN model was trained to capture the intricate mappings between input hyperparameters and output performance scores. Through an iterative learning process, the ANN model adapted its internal parameters to minimize the discrepancy between predicted and actual performance metrics, thus optimizing its predictive capabilities. This approach enabled the ANN to effectively learn the underlying patterns and dependencies within the data, facilitating accurate predictions of performance scores based on input hyperparameters. The utilization of ANN technology underscores its versatility in handling diverse datasets and tasks, offering promising avenues for enhanced decision-making and predictive analytics in various domains.

The ANN model was trained using the Adam optimizer, employing Mean Squared Error (MSE) as the loss function and Mean Absolute Error (MAE) as the evaluation metric. These hyperparameters were chosen to enhance the model's accuracy and minimize prediction errors.

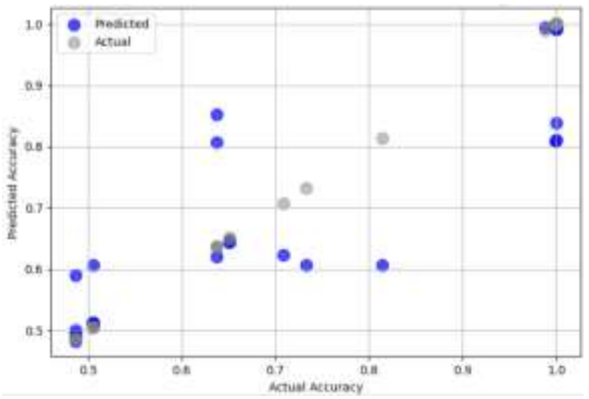


Figure 18: Scatter Plot of Actual vs Predicted Accuracy

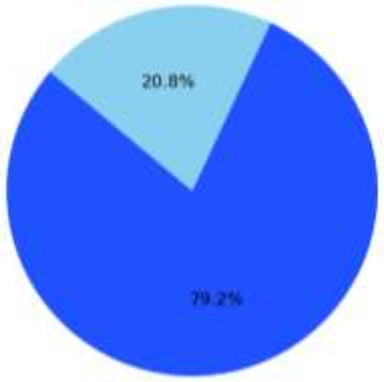


Figure 19: True vs False Accuracy Prediction Percentage

Figure 18 presents a scatter plot illustrating the relationship between actual and predicted accuracy values. Each data point on the plot represents an observation, with the x-axis indicating the actual accuracy value and the y-axis representing the predicted accuracy value. The plot utilizes distinct colors to distinguish between predicted (Blue) and actual (gray) values, aiding visual clarity. Ideally, points should align closely along the diagonal line, indicating accurate predictions. However, deviations from this line suggest disparities between predicted and actual accuracy, highlighting areas for model enhancement. By analyzing this plot, one can assess the model's effectiveness in predicting accuracy metrics and pinpoint specific instances where adjustments may be necessary to improve predictive performance and overall model reliability.

Figure 19, a pie plot, provides a concise summary of the distribution of correct and incorrect predictions for accuracy metrics. The pie chart is divided into two segments: one representing the percentage of values predicted correctly and the other indicating the percentage of values not predicted correctly. In this specific instance, the pie chart reveals that 79.2% of accuracy values were predicted accurately, while 20.8% of accuracy values were not predicted correctly. This visualization offers a clear depiction of the model's overall accuracy in predicting accuracy metrics, facilitating easy interpretation and understanding of prediction outcomes.

Overall, the combined analysis from both figures allows for a comprehensive evaluation of the model's predictive capabilities regarding accuracy metrics. It provides valuable insights for refining the model and optimizing its performance in future iterations.

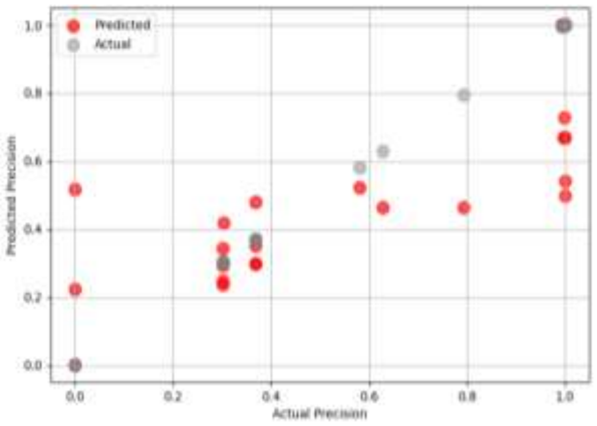


Figure 20: Scatter Plot of Actual vs Predicted Precision Score

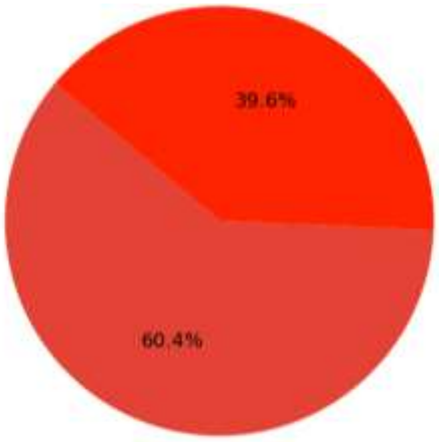


Figure 21: True vs False Precision score Percentage

Figure 20 depicts a plot showcasing the precision scores of actual versus predicted values. This visualization enables a thorough comparison between precision metrics derived from model predictions and the corresponding ground truth-values. Each data point on the plot represents an observation, with the x-axis representing the actual precision score and the y-axis indicating the predicted precision score. Utilizing red for predicted values and grey for actual values aids in distinguishing between them. Ideally, points should cluster closely around the diagonal line, signifying accurate predictions. Any deviations from this line suggest disparities between predicted and actual precision scores, highlighting areas for model refinement. Analysts can leverage this plot to glean insights into the model's precision prediction capabilities and identify avenues for enhancing its performance, ultimately improving the model's reliability and efficacy in practical applications.

Figure 21 displays a pie chart depicting the distribution of correct and incorrect predictions for precision scores. The chart reveals that 60.4% of precision values were predicted accurately, while 39.6% were not predicted correctly. This visualization offers a clear overview of the model's performance in predicting precision metrics, providing valuable insights into its predictive capabilities. By understanding the proportion of correct and incorrect predictions, analysts can assess the model's overall accuracy and identify areas for improvement. This information helps in refining the model and optimizing its performance for precision prediction tasks. Additionally, the pie chart facilitates easy communication of results to stakeholders, enabling informed decision-making based on the model's predictive accuracy. Overall, Figure 17 serves as a useful tool for evaluating and enhancing the precision prediction capabilities of the model, ultimately leading to improved performance and better outcomes in practical applications.

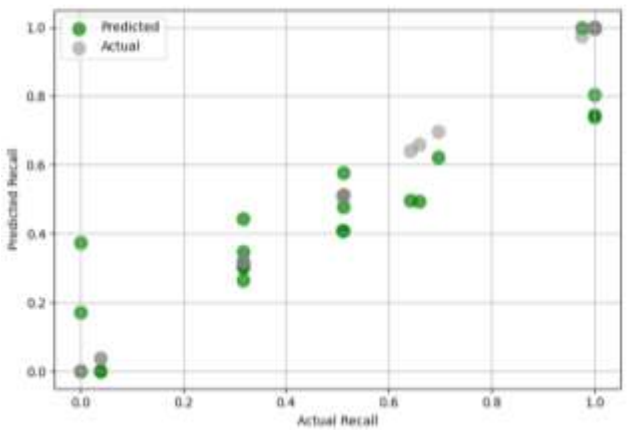


Figure 22:Scatter Plot of Actual vs Predicted Recall Score

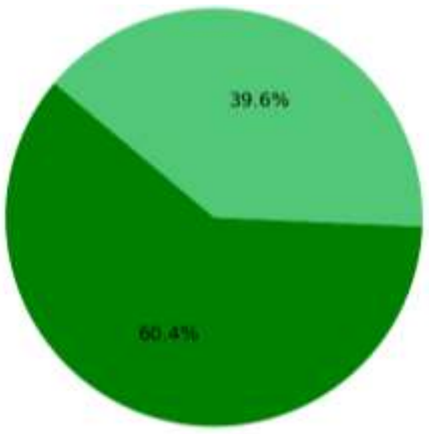


Figure 23: True vs False Recall Score Percentage

Recall score, also known as sensitivity, in the context of classification tasks, indicates the model's ability to correctly identify all relevant instances from a dataset. It quantifies the proportion of true positive cases (correctly predicted positives) out of all actual positive cases. A high recall score indicates that the model effectively captures a large portion of positive instances, minimizing false negatives and ensuring comprehensive coverage of relevant data points.

Figure 22 illustrates a visual comparison between the predicted and actual recall scores generated by the ANN model. This visualization provides valuable insights into the model's performance in predicting recall metrics across different data instances. Each data point represents an observation, with green indicating predicted values and grey representing actual values. By plotting predicted recall scores against actual values, analysts can evaluate the accuracy of the model's predictions and pinpoint areas requiring improvement. A close alignment between predicted and actual recall scores indicates accurate predictions, while deviations highlight discrepancies needing attention. Understanding this alignment empowers stakeholders to assess the model's effectiveness in capturing relevant instances and make informed decisions regarding model refinement and optimization for improved performance in real-world scenarios.

Figure 23 presents the distribution of recall score predictions, indicating that 60.4% of predictions were accurate, while 39.6% were incorrect. This visualization succinctly summarizes the model's performance in predicting recall scores, providing valuable insights into its predictive accuracy. Analyzing this distribution helps stakeholders assess the model's effectiveness and prioritize improvements. By understanding the proportion of correct and incorrect predictions, informed decisions can be made to refine the model and optimize its performance for classification tasks, ultimately leading to outcomes that are more reliable.

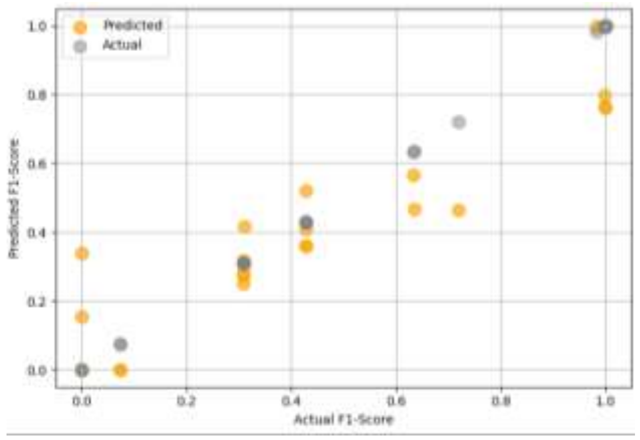


Figure 24: Scatter Plot of Actual vs Predicted F1-Score

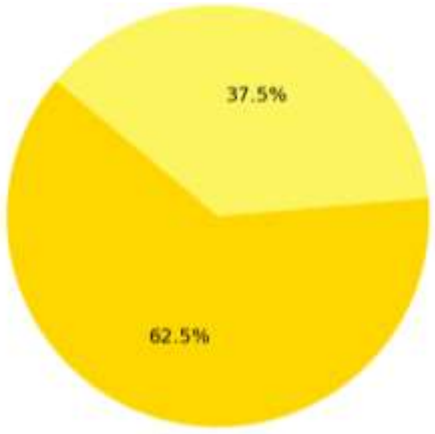


Figure 25: True vs False F1-Score Percentage

The F1 score is a metric that combines precision and recall into one value. It offers a balanced measure of a model's performance by calculating the harmonic mean of precision and recall. It considers both false positives and false negatives, making it useful for evaluating classifiers, particularly when dealing with imbalanced datasets. A high F1 score indicates both high precision and recall, suggesting the model effectively identifies relevant instances while minimizing errors.

In Figure 24, the yellow color represents predicted values, while grey represents actual values. This visualization likely show-cases the F1 score predictions of the model, with yellow indicating predicted F1 scores and grey representing actual F1 scores. The F1 score is highlighted as the second most reliable metric, suggesting its importance in evaluating the model's performance alongside other metrics.

Figure 25 displays a pie chart illustrating that 62.5% of predictions were accurate, while 37.5% were incorrect. This visualization succinctly summarizes the model's performance in predicting F1 scores, providing insights into its predictive accuracy. Analyzing this distribution helps stakeholders assess the model's effectiveness and prioritize improvements. By understanding the proportion of correct and incorrect predictions, informed decisions can be made to refine the model and optimize its performance for classification tasks, ultimately leading to more outcomes that are reliable.

4.3 Conclusion

In conclusion, our findings highlight the variability in SVM model accuracy across different hyperparameters, indicating significant potential for enhancement. While certain configurations may yield favorable results, others may require further optimization to achieve desired outcomes.

Additionally, training the ANN model on hyperparameters enables accurate prediction of most metrics; however, there is room for improvement to enhance its overall performance. These observations underscore the importance of continued refinement and optimization efforts in both SVM and ANN models to maximize predictive capabilities and ensure reliable results in various applications. Further research and experimentation are warranted to explore additional strategies for enhancing model performance and addressing any existing limitations.

References

[1] Swarm Behaviour. (2020). UCI Machine Learning Repository. https://doi.org/10.24432/C5N02J

[2] Grossi, Enzo & Buscema, Massimo. (2008). “Introduction to artificial neural networks. European journal of gastroenterology & hepatology” 19. 1046-54. 10.1097/MEG.0b013e3282f198a0.

[3] Evgeniou, Theodoros & Pontil, Massimiliano. (2001). Support Vector Machines: Theory and Applications. 2049. Page no. 38. 10.1007/3-540-44673-7_12.

[4] Muzzammel, Raheel & Raza, Ali. (2020). A Support Vector Machine Learning-Based Protection Technique for MT-HVDC Systems. Energies. 13. Page no. 3. 10.3390/en13246668.

[5] Powers, David & Ailab,. (2011). Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation. J. Mach. Learn. Technol. 2. 2229-3981. 10.9735/2229-3981. Marina Sokolova, Nathalie Japkowicz, Stan Szpakowicz. “”Beyond Accuracy, F-score and ROC:a Family of Discriminant Measures for Performance Evaluation”, 2006

[6] Atangana, Romain & Tchiotsop, Daniel & Kenne, Godpromesse & Djoufack, Laurent. (2020). EEG Signal Classification using LDA and MLP Classifier. 9. Page no. 22. 10.5121/hiij.2020.9102.

[7] D. A. Anggoro and D. Novitaningrum, Comparison of accuracy level of support vector machine (SVM) and artificial neural network (ANN) algorithms in predicting diabetes mellitus disease, ICIC Express Letters, vol.15, Page no. 15, https://doi.org/10.24507/icicel.15.01.9, 2021.

[8] Dr Shadi Abepeikar(Research Associate), A/Prof Kasmarik(Faculty), A/Prof Micheal Barlow(Faculty), Md Mohiuddin Khan(PHD student). University of New South Wales (UNSW), Canberra. “Human Perception of Swarming”, 2020