Temperature Prediction using Artificial Neural Network and Support Vector Machine based on Humidity and Wind Speed Data*

Labid Bin Bashar
23822701
Orchid: 0000-0003-4404-4016

Muhammad Ario Ilham Firmandito
23121008

Radhiyah Ulfah Pratiwi
23321301
radhiyah.ra@gmail.com

***Abstract*—This paper presents a comparative analysis of Artificial Neural Network (ANN) and Support Vector Machine (SVM) models for temperature prediction. The objective is to determine the model that performs better in terms of accuracy, computational time, and goodness-of-fit measure. The dataset consists of humidity and wind speed as input variables, and temperature as the output variable. The models were trained using the same dataset after applying data preprocessing techniques. The ANN model utilized a perceptron multilayer network with backpropagation, while the SVM model employed Support Vector Regression (SVR). Evaluation metrics such as mean squared error (MSE) and determination coefficient were used to assess performance. Results indicate that both models performed well, with the ANN model showing a slight advantage in accuracy and the SVM model demonstrating better computational efficiency. The choice between the models depends on specific requirements. This analysis provides valuable insights for researchers and practitioners in the field of machine learning and environmental science, highlighting the strengths and weaknesses of each model. Further research can explore additional factors and variables to enhance the accuracy and efficiency of temperature prediction models.**

***Index Terms*—ANN, SVM, Temperature Prediction, Regression**

I. Introduction

The accurate prediction of temperature is crucial in various fields, such as agriculture, meteorology, and environmental science. The advancement of machine learning techniques has allowed for the development of efficient temperature prediction models. Two popular machine learning techniques for regression problems are Artificial Neural Networks (ANN) [1] and Support Vector Machines (SVM) [2]. Despite their widespread use, there is still no clear consensus on which algorithm performs better for temperature prediction. A comparative analysis of ANN and SVM for temperature prediction can contribute to a better understanding of the strengths and weaknesses of these models. This analysis can provide insights into which algorithm is more suitable for different types of temperature prediction tasks. For instance, in applications that require high accuracy, such as weather forecasting, the comparison can help identify which algorithm is more accurate. Similarly, in applications that require faster training and prediction times, the comparison can help identify which algorithm is more efficient. Moreover, the comparison of ANN and SVM can also provide insights into the interpretability of the models. Understanding how these models make predictions is crucial in fields where the interpretability of the model is critical, such as environmental science. Furthermore, the comparison can help in identifying the best practices and hyperparameter settings for each algorithm.

In regression problems like temperature prediction, artificial neural networks (ANNs) and support vector machines (SVMs) are two popular machine learning techniques for modeling and prediction. ANNs have been widely used in various applications due to their ability to learn complex nonlinear relationships between input and output variables. Some of the researchers have already trained various ANN model to predict temperature [1], [3]–[9]. ANNs can capture hidden patterns and relationships within the data and provide accurate predictions. However, one challenge of ANNs is that they require a large amount of training data to be effective, and the training process can be computationally expensive [10].On the other hand, SVMs are also commonly used in regression problems and have shown to be effective in a variety of applications [11]–[13]. One advantage of SVMs is that they are less prone to overfitting than ANNs, which means they can provide more accurate predictions on unseen data. SVMs can also handle high-dimensional datasets and are capable of modeling nonlinear relationships between input and output variables [14]. However, one disadvantage of SVMs is that they can be sensitive to the choice of kernel function, and the selection of the kernel function requires domain expertise [14], [15].

Although, in the literature, numerous works have been found to be undertaken, the performance and efficiency of any model in some way are rather subject to the nature of dataset and application field. On the top of that, a comparative analysis indicates the feasibility of one approach compared to another given the specific dataset. Hence,the paper proposed a comparative analysis of ANN and SVM models for temperature prediction. The study aimed to explore the performance of the two popular machine learning models in temperature prediction and identify which model is better suited for this task. The paper trained both models using the same dataset and compared their prediction accuracy, computational time, and efficiency. The results of the study showed that both models performed well in predicting the temperature of a place, with SVM having a slight advantage over ANN in terms of accuracy. However, ANN was found to be more computationally efficient than SVM. Overall, the paper's contribution is significant as it provides insights into the performance of two popular machine learning models for temperature prediction, which can be helpful for researchers and practitioners working in this field.

The paper on training ANN and SVM and comparing them for temperature prediction problem is composed of four sections. Section I is the introduction that highlights the importance of temperature prediction and the need for accurate forecasting. Section II discusses the training method used for both ANN and SVM models. The training process includes feature selection, model configuration, and parameter tuning. Section III presents the results of the experiments carried out to evaluate the performance of both models. The evaluation metrics used include mean squared error, root mean squared error, mean absolute error, and coefficient of determination. Section IV concludes the paper by summarizing the key findings and highlighting the strengths and weaknesses of both models. Overall, the paper provides a comprehensive analysis of the use of ANN and SVM for temperature prediction, and the results can be useful for researchers and practitioners working in the field of machine learning and environmental science.

II. Dataset Preparation

The dataset used in this paper is multiple input single output (MISO) data. The inputs are humidity and wind speed in percent and km/h, respectively. While the output is the temperature in Celsius. This dataset is data taken with one-hour time step for 12 years from 2006 to 2017.

Data preparation is a process of collecting, cleaning and transforming data that needs to be done before the data is processed. Data preparation needs to be done to ensure the accuracy and consistency of the data. In general, the data preparation process can be divided into data collecting, pre-processing and transformation. Before gathering data from multiple sources, the first step to do is to identify the problem that needs to be solved with machine learning. As a result, the required data's characteristics may be defined, and data gathering is permitted. Furthermore, data preprocessing is a process that aims to improve data quality. Data quality may be improved in several ways, including formatting, cleaning, and sampling. Formatting ensures that variables are written consistently throughout the data, cleaning aims to eliminate redundant information and outliers, and sampling may be carried out when the amount of data is excessive. The last step in data preparation is data transformation or also known as feature engineering. In this step, data is transformed into a form that is more suited for the particular machine-learning algorithm. One of the data transformation processes is normalization. Normalization is used to minimize redundancy and eliminate unwanted characteristics such as anomalies. Mathematically, normalization can be written as in the following equation,

$$Y = (X - b)/(a - b)$$

where, Y is the result of normalization, X is the actual data, b is the minimum value of the actual data and a is the maximum value of the actual data. With this equation, the normalization result will be between 0 and 1.

One of the most crucial activities when constructing models is the capacity to test, assess, and have something to compare with that shows that the built model actually worked. To do this, it is necessary to have access to sufficient data that can be used to train, create, and test the model. Thus, generally, the dataset is divided into three parts to avoid overfitting which includes training, validation and testing data. The training data is used to fit and train the model. In case of Neural Network, it is used to estimate the weights and biases. From this data, the model observes, learns, and optimize its parameters. The validation data is used for hyper-parameterization and tuning of the model during the training process. If the data is divided into two parts, this data is called as test data. Last, the testing data is used to evaluate the model. The proportions of the data splitting are decided according to the size and type of the available data. If the size of the dataset is between a hundred to a million, then the data is divided by the ratio 60:20:20. The 60% of the data is the training set, the 20% of the data is the validation data and the remaining data is the testing data. In this study, we split our data into two parts which are training and testing data by the ratio 80:20.

III. Training Method

A. Artificial Neural Network

Artificial Neural Networks (ANNs) are computational models inspired by the structure and function of biological neural networks. They consist of interconnected artificial neurons, also known as nodes or units, organized in layers. ANNs can learn and make predictions by adjusting the weights and biases of the connections between neurons. Forward propagation refers to the process of computing the output of an ANN given an input. Each neuron receives inputs, applies an activation function to the weighted sum of those inputs, and passes the result to the next layer. Here's the mathematical equation for the forward propagation of a neuron:

For a neuron j in layer l:

$$z_j^l = \sum(w_{ij}^l \cdot a_i^{(l-1)}) + b_j^l \tag{1}$$

$$a_j^l = g(z_j^l) \tag{2}$$

Where: - z_j^l is the weighted sum of inputs for neuron j in layer l. - w_{ij}^l is the weight between neuron i in layer $(l-1)$ and neuron j in layer l. - $a_i^{(l-1)}$ is the output (activation) of neuron i in layer $(l-1)$. - b_j^l is the bias term for neuron j in layer l. - $g(.)$ is the activation function applied to the weighted sum to compute the output (activation) of the neuron. Backpropagation is used to calculate the gradients of the weights and biases in the ANN, allowing for the adjustment of these parameters during the learning process.

It involves propagating the error from the output layer back to the previous layers, iteratively updating the weights based on the calculated gradients. Here's the mathematical equation for backpropagation:

For a neuron j in layer l:

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} \tag{3}$$

Where: - δ_j^l is the error term for neuron j in layer l. - $\frac{\partial C}{\partial z_j^l}$ is the partial derivative of the cost function C with respect to the weighted sum z_j^l.

Using the error term, the gradients for the weights and biases can be calculated as follows:

For a neuron j in layer l:

$$\frac{\partial C}{\partial w_{ij}^l} = a_i^{(l-1)} \cdot \delta_j^l \tag{4}$$

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \tag{5}$$

These gradients are then used to update the weights and biases by subtracting a learning rate multiplied by the gradients from their current values. The process of forward propagation followed by backpropagation is repeated iteratively (often for multiple epochs) during the training of an ANN to minimize the cost function and improve the model's predictions. Figure 1 shows the architecture of an multilayer perceptron ANN. Now,

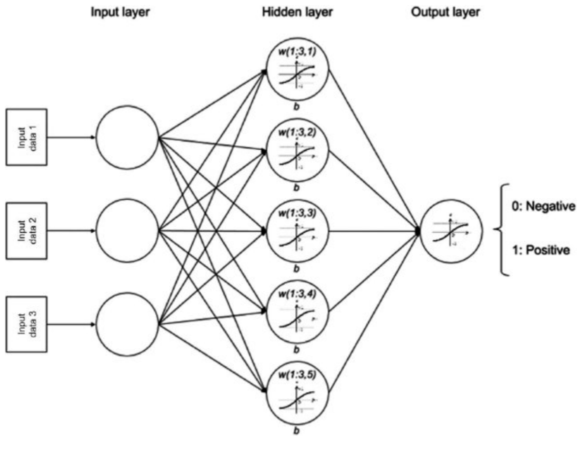

Fig. 1. ANN Multilayer Perceptron Architecture

the Levenberg-Marquardt (LM) algorithm is an optimization method commonly used in backpropagation for training artificial neural networks. It combines the advantages of the Gauss-Newton method and the gradient descent method to efficiently update the weights and biases of the network. In the process of this algorithm, first Initialize the weights and biases of the neural network with random values or predefined values. Perform the forward propagation step to compute the output of the network for a given input. Calculate the error between the network output and the desired output using a chosen error function, such as the mean squared error (MSE). Perform the backpropagation step to compute the gradients of the error with respect to the weights and biases. Calculate the Hessian matrix, which is a matrix of second-order partial derivatives of the error function with respect to the weights and biases. It represents the curvature of the error surface. Update the weights and biases using the LM algorithm, which involves calculating the LM parameter λ and updating the parameters based on the LM update equation. The LM update equation for a weight parameter w in the neural network is given by:

$$w_{\text{new}} = w_{\text{old}} - \left(J^T J + \lambda \text{diag}(J^T J)\right)^{-1} J^T \delta \tag{6}$$

Where w_{new} is the updated weight value. w_{old} is the current weight value. J is the Jacobian matrix, which represents the derivatives of the network output with respect to the weights and biases. δ is the vector of errors or residuals. λ is the LM parameter that controls the trade-off between the Gauss-Newton method and the gradient descent method. It determines the step size for updating the parameters. The LM algorithm adjusts the value of the LM parameter during the training process to strike a balance between convergence speed and stability. The parameter is typically increased or decreased based on the improvement or deterioration of the error. Finally, Repeat the forward propagation, error calculation, backpropagation, Hessian matrix calculation, and LM parameter update steps iteratively until the network achieves a satisfactory level of performance or convergence. By employing the LM algorithm, the backpropagation process can efficiently update the weights and biases of the neural network, leading to improved training performance and convergence.

B. *Support Vector Machine*

Support Vector Machine is an algorithm based on statistical learning theory. The basic idea of Support Vector Machines is to map the original data X into a feature space F with high dimensionality through a non linear mapping function and construct an optimal hyperplane in new space. SVM can be applied to both classification and regression. In the case of classification, an optimal hyperplane is found that separates the data into two classes. Whereas in the case of regression a hyperplane is to be constructed that lies close to as many points as possible. The distance from the hyperplane to the nearest data points on both sides is defined as the margin. So the SVM regression tries to minimize the distance between the predicted values and the actual values while staying within a certain threshold. This threshold is called the epsilon-insensitive zone and represents the maximum allowable error in the predictions. In this paper Support Vector Regression (SVR) is used to predict the maximum temperature. Support vector regression is different from conventional regression techniques because it uses Structural Risk Minimization (SRM) but not Empirical Risk Minimization (ERM) induction principle which is equivalent to minimizing an upper bound on the generalization error and not the training error. Due to this feature it is expected to perform better than conventional techniques which may suffer from possible over fitting. Regression is the problem of estimating a function based on a given data set. on a given data set. Consider a data set $G = \{(X_i, d_i)\}_{i=1}^{N}$,where x_i is the input vector, d_i is the desired result and N corresponds to the size of the data set. The general form of Support Vector Regression estimating function is

$$f(x) = (w.\varphi(x)) + b \tag{7}$$

where w and b are the co-efficients that have to be estimated from data. $\varphi(x)$ is the non linear function in feature space. The non linear regression problem can be expressed as an optimization problem in the following way by minimizing the regularized risk function.

$$R(C) = \frac{1}{2}\|w\|^2 + C\frac{1}{N}\sum_{i=1}^{N} L_{\in}(d_i, y_i) \tag{8}$$

where

$$L_{\in}(d, y) = \left\{ \begin{array}{ll} |d-y| \in & |d-y| \geq \in \\ 0 & \text{others} \end{array} \right\}$$

After introducing slack variables the risk function can be expressed in the following constrained form. minimize $R(w, \xi^*) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*)$ subject to

$$\begin{gathered} d_i - w\varphi(x_i) - b_i \leq \in + \xi_i \\ (w \cdot \varphi(x)) + b - d_i \leq \in + \xi_i^* \\ \text{where } \xi_i, \xi_i^* \geq 0 \end{gathered}$$

The dual form of the non linear SVR can be expressed as $\alpha(\alpha_1, \alpha_i^*) = \sum_{i\neq 1}^{N} d_i(\alpha_i - \alpha_i^*) - \sum_{i\neq 1}^{N}(\alpha_i + \alpha_i^*) - \frac{1}{2}\sum_{i\neq j\neq}^{N}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) K(X_i, X_j)$ subject to

$$\begin{gathered} \sum_{i=1}^{N}(\alpha_i - \alpha_i^*) = 0 \\ 0 \leq \alpha_i, \alpha_i^* \leq C, i = 1, 2, \ldots .n \end{gathered}$$

α_i and α_i^* are the Lagrange multipliers that act as forces pushing the predictions towards target value d.

IV. Results and Discussion

The ANN model has been trained on the dataset described before based on the algorithm outilined in the previous section. The parameters used in this process have been listed in the table I. The network has one layer with three neurons. The the Levenberg-Marquardt optimization method has been employed for back-propagation with a view to minimize the loss function. The data is divided into training, validation, and testing sets using the function. The ratio of data used for training, validation, and testing is set to 80%, 0%, and 20%, respectively. The performance of the network is measured using the mean squared error (MSE). The variable is set to display the performance plot, training state plot, error histogram plot, and regression plot. The predictor is used to make temperature predictions for the input dataset and the target dataset. The data is then de-normalized using the maximum and minimum values of the data. Finally, the predicted values and the target values are plotted in a graph.

Fig. 2 shows that at 311 epoch, the training performance reach nearest to the best value. Fig. 3 shows the gradient vs epoch plot during the training of the neural network. The gradient is seen to decrease over epochs, which indicates that the network is converging towards the minimum error. However, there are some oscillations or roughness observed while the gradient is decreasing. This means that the optimization process is not smooth and the learning rate may need to be adjusted to overcome these oscillations. The oscillations or roughness observed in the gradient vs epoch plot may be attributed to the choice of learning rate or other hyperparameters used in the training process. If the learning rate is too high, the optimization process may overshoot the minimum and oscillate around it. On the other hand, if the learning rate is too low,

TABLE I
Training of ANN

Variables	Values
Training Algorithm	Levenberg-Marquardt backpropagation
# of hidden layer	1
Layer weights	[1.0495, -10.0302, -0.0304]
Bias	8.2473
μ	0.001
Maximum Epoch	1000
Learning Rate α	0.1

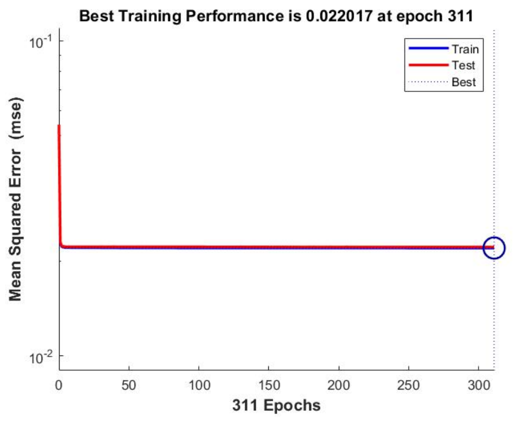

Fig. 2. ANN Mean Square Error at Every epoch

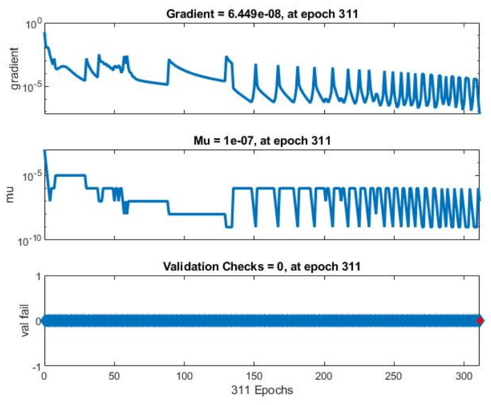

Fig. 3. ANN Training State at Every epoch

the optimization process may get stuck in a local minimum and fail to converge to the global minimum. Therefore, it is important to carefully choose the hyperparameters such as learning rate, momentum, and weight decay to ensure the optimization process is smooth and the network converges to the global minimum. Additionally, monitoring the gradient vs epoch plot during training can provide insights into the convergence behavior of the network and help in fine-tuning the hyperparameters. Overall, this code trains an ANN model for temperature prediction using a specified input and target dataset. The performance of the network can be evaluated by examining the MSE value and the plots generated during training. This fig. also shows the plot of ”mu vs epoch” showing how the momentum parameter changes over the course of training. The oscillations in the plot indicate that the momentum parameter is being adjusted dynamically during the training process, and that its value is being tuned to improve the performance of the ANN.

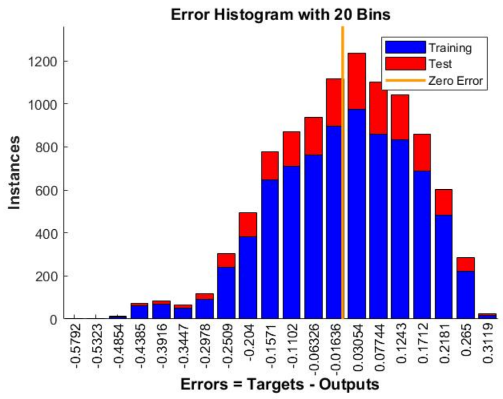

Fig. 4. Error Historgram

In 4, the error histogram plot, displays the distribution of errors for the training dataset after the ANN has been trained. The x-axis represents the error values, while the y-axis represents the number of instances that correspond to each error value. The plot shows that a significant number of instances (more than 1200) have zero error, indicating that the ANN is able to accurately predict the temperature for those instances. However, there are also instances where the error is higher, indicating that the ANN is not as accurate for those data points. The implications of this plot are that the ANN is able to accurately predict the temperature for a significant portion of the dataset. However, there are still instances where the ANN's predictions are not as accurate. This suggests that there may be additional factors that could be considered to further improve the accuracy of the ANN's predictions, such as including additional variables in the input dataset or adjusting the network architecture or training parameters.

Fig. 5 shows the regression analysis of the Artificial Neural Network (ANN) model trained using the given dataset. The figure shows three different regression plots: Training Regression, Testing Regression, and All Regression. The Training

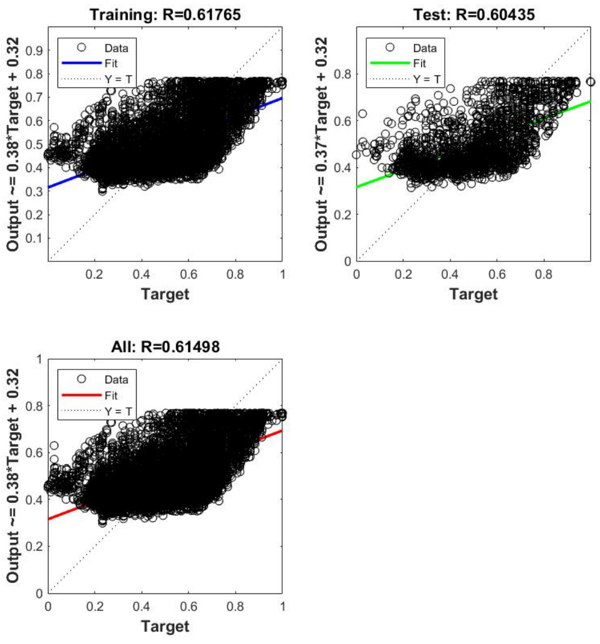

Fig. 5. Regression Output of the ANN Model

Regression plot shows the regression between the predicted and actual temperature values for the training dataset. The plot shows a good correlation between the predicted and actual values, indicating that the ANN model has been successful in learning the patterns in the training data. The Testing Regression plot shows the regression between the predicted and actual temperature values for the testing dataset. The plot also shows a good correlation between the predicted and actual values, indicating that the ANN model is capable of generalizing to new data. The All Regression plot shows the regression between the predicted and actual temperature values for both the training and testing datasets combined. The plot shows that the ANN model has learned the patterns in the data well, as indicated by the good correlation between the predicted and actual values.

Fig. 6 shows the predicted temperature and the real temperature plotted against all data points in the dataset. The blue line represents the predicted temperature values while the red line represents the real temperature values. From the plot, it is clear that the model predicts temperature values that closely match the real temperature values for most of the data points. However, there are a few instances where the model fails to predict the temperature accurately, particularly for the data points where the temperature is negative. The failure of the model to predict negative temperatures accurately can be attributed to the fact that the ANN model has not been trained sufficiently on such values or that there is insufficient data available for such temperature values. It is also possible that the model architecture may need to be modified to better handle negative temperature values.

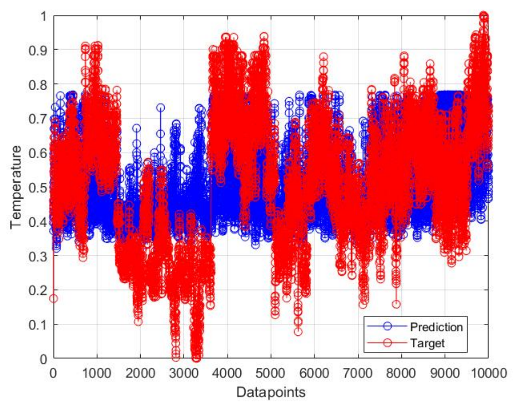

Fig. 6. ANN Prediction of Temperature at Every Data Points

TABLE II
Parameters used to Train SVM

Variables	Values
Solver	SMO
Bias	0.4622
Iterations	4190
ϵ	0.0214
Kernel Function	'gaussian'

Table II shows the necessary parameters and their values for training SVM model on the same dataset to predict the temperature. Figure 7 shows the predicted temperature and actual temperature for an SVM model trained on the same dataset as the ANN model. The graph is similar to Figure 5, which showed the performance of the ANN model. The graph shows a scatter plot of the predicted temperature values against the actual temperature values. The diagonal line represents the ideal situation where the predicted temperature is equal to the actual temperature. The points that lie along the diagonal line indicate that the SVM model has predicted the temperature accurately for those data points. Like the ANN model, the SVM model also struggled to predict the temperature for data points with negative temperature values. However, the overall performance of the SVM model seems to be slightly better than the ANN model. There are fewer points that lie away from the diagonal line, indicating that the SVM model has predicted the temperature more accurately for most of the data points. The output of the both model can be summarized in the table III

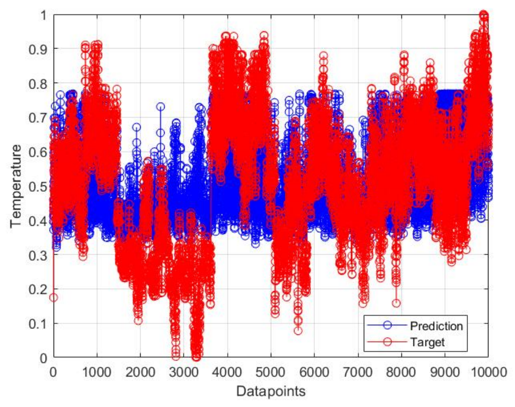

Fig. 7. SVM Prediction of Temperature at Every Data Points

TABLE III
Summary of ANN and SVM Results

Performance	ANN	SVM
MSE(train)	0.0220	0.022
MSE (test)	0.0222	0.024
R(train)	0.62	0.61
R(test)	0.60	0.63
Training time	3.2 second	2.6 second

V. Conclusion

This study compares Artificial Neural Network (ANN) and Support Vector Machine (SVM) models for temperature prediction. Both models were trained using the same dataset, with humidity and wind speed as input variables. The ANN utilized a single hidden layered network with backpropagation, while the SVM employed Support Vector Regression (SVR). Results showed that ANN had a slight advantage in accuracy, while the SVM was more computationally efficient. The choice between the models depends on specific requirements. This analysis provides insights for researchers and practitioners, suggesting further research to enhance temperature prediction models.

VI. Source Codes

```
function [predictor, Records] = ...
    Train_ANN(Input_Dataset, ...
    Target_Dataset)
    Learning_Algorithm = 'trainlm';
    Network_Architecture = [3];
    setdemorandstream(491218382)
    The_Network = fitnet...
    (Network_Architecture,...
    Learning_Algorithm);
    train_ratio = 0.8;
    val_ratio = 0.0;
    test_ratio = 0.2;
    [trainInd,valInd,testInd]=...
    dividerand(size(Input_Dataset,1)...
    ,train_ratio,val_ratio,test_ratio);
    The_Network.divideFcn = 'divideind';
    The_Network.divideMode = 'sample';
    The_Network.divideParam.trainInd ...
    = trainInd;
    The_Network.divideParam.valInd ...
    = valInd;
    The_Network.divideParam.testInd ...
    = testInd;
    The_Network.performFcn = 'mse';
    The_Network.plotFcns = ...
    {'plotperform',...
    'plottrainstate',...
    'ploterrhist',...
    'plotregression'};

    [predictor, Records] ...
    = train(The_Network,...
    Input_Dataset', ...
    Target_Dataset');


    train_result = ...
    predictor(Input_Dataset');
    max_data = 37.1277777;
    min_data = -14.08888889;
    train_result = ((train_result-0)...
    *(max_data-min_data)/1)+min_data;
    target_dataset = ...
    ((Target_Dataset-0)*...
    (max_data-min_data)/1)+min_data;

    figure,
    plot(train_result,'bo-')
    hold on
    plot(target_dataset,'ro-')
    hold off
    grid on
    xlabel('Datapoints')
    ylabel('Temperature')
    legend('Prediction','Target'...
    ,'Location','Best')
end

function [mdl, Records] = ...
Train_SVM(Input, Target)
    setdemorandstream(491218382)
    [trnx,trny,valx,valy,tsx,tsy]...
    = Split_data(Input,Target);
    mdl =fitrsvm(trnx, trny,..
    'KernelFunction','gaussian');

    % Predict on training data
    Predictor = predict(mdl, trnx);
    y_pval = predict(mdl,valx);
    y_pts = predict(mdl,tsx);


    trn_corr = corrcoef(trny,Predictor);
    R_trn = trn_corr(1,2);
    val_corr = corrcoef(valy, y_pval);
    R_val = val_corr(1, 2);
    ts_corr = corrcoef(tsy, y_pts);
    R_ts = ts_corr(1, 2);


    % Manual calculation of performance
    mse_train = ...
    mean((Predictor - trny).^2);
    mse_val =  ...
    mean((y_pval - valy).^2);
    mse_ts =  mean((y_pts - tsy).^2);

    train_result = mdl.predict(Input);
    max_data = 37.1277777;
    min_data = -14.08888889;
    train_result = ((train_result-0)...
    *(max_data-min_data)/1)+min_data;
    target_dataset = ((Target-0)...
    *(max_data-min_data)/1)+min_data;
    figure,
    plot(train_result,'bo-')
    hold on
    plot(target_dataset,'ro-')
    hold off
    grid on
    xlabel('Datapoints')
    ylabel('Temperature')
    legend('Prediction','Target'...
    ,'Location','Best')


    % Records for performance metrics
    Records.mse_train = mse_train;
    Records.mse_val = mse_val;
    Records.mse_ts = mse_ts;
    Records.R_train = R_trn;
    Records.R_validation = R_val;
    Records.R_test = R_ts;

end
function [Training_Set,...
Target_Training_Set,Validation_Set,...
    Target_Validation_Set,...
    Testing_Set,Target_Testing_Set] = ...
    Split_data(Input,Target)


    rng('default');
    rng(1);

    % Split the data into
    %training set (70%),
    %validation set (20%),
    %and testing set (10%)
    cvp = cvpartition(size(Input,...
    1), 'HoldOut', 0.2);
    Training_Set = Input(cvp.training,:);
    Validation_Test_Set = ...
    Input(cvp.test,:);
    Target_Training_Set = ...
    Target(cvp.training,:);
    Target_Validation_Test_Set...
    = Target(cvp.test,:);

    cvp2 = cvpartition...
    (size(Validation_Test_Set, 1),...
    'HoldOut', 0.33);
    Validation_Set =...
    Validation_Test_Set(cvp2.training,:);
    Testing_Set = ...
    Validation_Test_Set(cvp2.test,:);
    Target_Validation_Set =...
    Target_Validation_Test_Set...
    (cvp2.training,:);
    Target_Testing_Set = ...
    Target_Validation_Test_Set...
    (cvp2.test,:);
end

function [Training, Target]...
= Load_Data(File_Name)

    warning('off')
    Training_Dataset = File_Name;
    Training_Dataset_Options ...
    = detectImportOptions...
    (Training_Dataset);
    Training_Data = ...
    readtable(Training_Dataset...
    , Training_Dataset_Options,...
        "UseExcel", false);

    x1 = Training_Data.x1;
    x2 = Training_Data.x2;
    % Attendance_Score = ...
    Training_Data.Attendance;
    % Midterm_Score =...
    Training_Data.Midterm;

    Training = [x1 x2];
    Target = Training_Data.y;

end
```

References

[1] L. Wang, G. von Laszewski, F. Huang, J. Dayal, T. Frulani, and G. Fox, "Task scheduling with ann-based temperature prediction in a data center: a simulation-based study," *Engineering with Computers*, vol. 27, pp. 381–391, 2011.

[2] W. S. Noble, "What is a support vector machine?," *Nature biotechnology*, vol. 24, no. 12, pp. 1565–1567, 2006.

[3] Ö. A. Dombaycı and M. Gölcü, "Daily means ambient temperature prediction using artificial neural network method: A case study of turkey," *Renewable Energy*, vol. 34, no. 4, pp. 1158–1161, 2009.

[4] Y. Wang, X. Chen, C. Li, Y. Yu, G. Zhou, C. Wang, and W. Zhao, "Temperature prediction of lithium-ion battery based on artificial neural network model," *Applied Thermal Engineering*, p. 120482, 2023.

[5] C. Johnstone and E. D. Sulungu, "Application of neural network in prediction of temperature: a review," *Neural Computing and Applications*, vol. 33, no. 18, pp. 11487–11498, 2021.

[6] C. Yin, L. Rosendahl, and Z. Luo, "Methods to improve prediction performance of ann models," *Simulation Modelling Practice and Theory*, vol. 11, no. 3-4, pp. 211–222, 2003.

[7] B. Thomas and M. Soleimani-Mohseni, "Artificial neural network models for indoor temperature prediction: investigations in two buildings," *Neural Computing and Applications*, vol. 16, pp. 81–89, 2007.

[8] T. K. Routh, A. H. B. Yousuf, M. N. Hossain, M. M. Asasduzzaman, M. I. Hossain, U. Husnaeen, and M. Mubarak, "Artificial neural network based temperature prediction and its impact on solar cell," in *2012 International conference on informatics, electronics & vision (ICIEV)*, pp. 897–902, IEEE, 2012.

[9] A. P. Piotrowski, M. J. Napiorkowski, J. J. Napiorkowski, and M. Osuch, "Comparing various artificial neural network types for water temperature prediction in rivers," *Journal of Hydrology*, vol. 529, pp. 302–315, 2015.

[10] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, p. e00938, 2018.

[11] Y. Radhika and M. Shashi, "Atmospheric temperature prediction using support vector machines," *International journal of computer theory and engineering*, vol. 1, no. 1, p. 55, 2009.

[12] N. I. Sapankevych and R. Sankar, "Time series prediction using support vector machines: a survey," *IEEE computational intelligence magazine*, vol. 4, no. 2, pp. 24–38, 2009.

[13] D. Niu, Y. Wang, and D. D. Wu, "Power load forecasting using support vector machine and ant colony optimization," *Expert systems with Applications*, vol. 37, no. 3, pp. 2531–2539, 2010.

[14] D. Anguita, A. Ghio, N. Greco, L. Oneto, and S. Ridella, "Model selection for support vector machines: Advantages and disadvantages of the machine learning theory," in *The 2010 international joint conference on neural networks (IJCNN)*, pp. 1–8, IEEE, 2010.

[15] J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, and A. Lopez, "A comprehensive survey on support vector machine classification: Applications, challenges and trends," *Neurocomputing*, vol. 408, pp. 189–215, 2020.