Analysis and Reproduction of the paper "A Comparative Study of Radial Basis Function Network with Different Basis Functions for stock Trend Prediction" and implementation of Fractalized RBF neural networks.

Indian Institute of Technology, Madras

Santanu Das (MA24M021)

Industrial Mathematics and Scientific Computing

May 2025

Department of Mathematics

Table of Contents

Chapter 1

Introduction

In the realm of financial forecasting, the ability to accurately predict stock market trends remains a challenging and highly valuable task. Traditional methods often struggle to model the complex, nonlinear patterns inherent in financial time series data. Neural networks, particularly Radial Basis Function (RBF) Neural Networks, offer a powerful alternative due to their universal approximation capabilities and simpler architecture compared to deep networks.

This project focuses on the analysis and reproduction of the study titled *"A Comparative Study of Radial Basis Function Network with Different Basis Functions for Stock Trend Prediction"*. The original work investigates the performance of various basis functions within RBF Neural Networks for stock trend classification. Unlike conventional training methods such as backpropagation, this project utilizes the Ridge Extreme Learning Machine (RELM) framework, which significantly accelerates training by using the Moore–Penrose pseudo-inverse for output weight computation.

In addition to reproducing the baseline results, this work proposes and implements a novel extension: **Fractalized Radial Basis Functions**. Inspired by fractal geometry, this approach involves constructing recursive basis functions through affine transformations and interpolation. The goal is to enhance the flexibility of the RBF network and improve its predictive performance, especially on datasets with high structural complexity.

The datasets used in this project include historical price data from the SENSEX and S&P 500 indices. Several technical indicators were engineered from the raw data to serve as features. Each RBF model is evaluated using standard classification metrics such as accuracy and F1-score.

The fractalized RBF model is further optimized using `Optuna`, a state-of-the-art hyperparameter tuning library.

This work not only confirms the observations made in the original paper but also introduces a new modeling strategy that could be generalized to other time series prediction tasks. The combination of RELM and fractal-based design illustrates the potential of hybrid approaches in building efficient and accurate neural architectures for financial forecasting. .

Chapter 2

Basic Tools Used in the Work

This project integrates a variety of tools, libraries, and frameworks that support data preprocessing, modeling, visualization, and optimization. The key tools used are as follows:

- **Python 3.10:** The primary programming language used for implementing the entire pipeline — from data loading and preprocessing to model building and evaluation.
- **NumPy and pandas:** These fundamental libraries were used for efficient numerical computation and data manipulation, particularly for handling time series data and engineering technical indicators.
- **Matplotlib and Seaborn:** Used for visualizing the exploratory data analysis (EDA), including time series plots, distribution plots, and correlation heatmaps.
- **Scikit-learn:** Provided utilities for scaling features (Min-Max normalization), calculating evaluation metrics (accuracy, F1-score), and splitting the dataset into training and testing sets.
- **Custom NumPy-based RBF Neural Network:** The Radial Basis Function Neural Network model was implemented from scratch using NumPy. The training was done using Ridge Extreme Learning Machine (RELM), which employs the Moore–Penrose pseudo-inverse for fast computation.
- **Optuna:** A powerful hyperparameter optimization framework used to fine-tune the fractalized RBF network. It enabled efficient exploration of the hyperparameter space using Bayesian optimization.

- **Jupyter Notebook:** Used as the primary development environment for interactive coding, testing, and documentation.

These tools collectively enabled a modular, efficient, and reproducible workflow for the experimentation and evaluation of both classical and fractalized RBF neural networks.

Chapter 3

Methodology

The methodology of this project is divided into two primary stages: (1) Data Preprocessing and Feature Engineering, and (2) Model Development and Training. The approach reproduces the original RBFN-based model and introduces an extension using fractalized basis functions.

Data Preprocessing and Feature Engineering

- Historical stock data for the SENSEX and S&P 500 indices were collected and analyzed. This included closing prices, volume, and other time-dependent features.
- Six technical indicators were engineered: Simple Moving Average (SMA), Exponential Moving Average (EMA), Relative Strength Index (RSI), Moving Average Convergence Divergence (MACD), Stochastic Oscillator %K and %D and Larry's William %R.
- A binary classification target was defined based on the directional movement of stock prices (up/down) to frame the trend prediction task.
- All features were normalized using Min-Max scaling to transform values into the range [0, 1], ensuring numerical stability during training.
- Exploratory Data Analysis (EDA) was performed using correlation matrices, line plots, and distribution visualizations to understand feature relationships and temporal behavior.

Dataset Description and Feature Engineering

This study uses daily historical stock market data from two major indices: BSE SENSEX and S&P 500, spanning from 1st January 2020 to 1st January 2025. The data was retrieved using the `yfinance` Python API.

Dataset Summary

blue!20 **BSE SENSEX**	**S&P 500**
1st Jan 2020 to 1st Jan 2025 (from yfinance)	1st Jan 2020 to 1st Jan 2025 (from yfinance)
No missing values (1234 rows, 6 columns)	No missing values (1258 rows, 6 columns)
SMA, MACD, %K, %D, RSI, and Williams %R computed	SMA, MACD, %K, %D, RSI, and Williams %R computed
Removed NaN values after indicator computation	Removed NaN values after indicator computation
Direction column computed (Up-/Down trend)	Direction column computed (Up-/Down trend)
666 upward, 553 downward	666 upward, 577 downward
ADF and KPSS tests indicate non-stationary series	ADF and KPSS tests indicate non-stationary series

Table 3.1: Summary of datasets used and preprocessing applied

Feature Engineering

The following technical indicators were computed using a rolling window and added as features for training the Radial Basis Function Networks:

- **Simple Moving Average (SMA):**

$$\text{SMA}_t = \frac{1}{n}\sum_{i=0}^{n-1} \text{Close}_{t-i}$$

- **Moving Average Convergence Divergence (MACD):**

$$\text{MACD} = \text{EMA}_{\text{short}} - \text{EMA}_{\text{long}}, \quad \text{MACD\_Signal} = \text{EMA}_{\text{MACD}}$$

- **Stochastic Oscillator (%K and %D):**

$$\%K_t = 100 \cdot \frac{\text{Close}_t - \text{LowestLow}_t}{\text{HighestHigh}_t - \text{LowestLow}_t}, \quad \%D_t = \text{SMA}_3(\%K)$$

- **Relative Strength Index (RSI):**

$$\text{RSI}_t = 100 - \frac{100}{1 + RS}, \quad RS = \frac{\text{Avg Gain}}{\text{Avg Loss}}$$

- **Williams %R:**

$$\text{Williams } \%\text{R}_t = -100 \cdot \frac{\text{HighestHigh}_t - \text{Close}_t}{\text{HighestHigh}_t - \text{LowestLow}_t}$$

All technical indicators were implemented in Python using `pandas`, `numpy`, and exponential moving average functions from `pandas.DataFrame.ewm()`. The final preprocessed datasets were exported as `CSV` files and used for model training.

Direction Label Creation

The trend direction was encoded as a binary classification label:

$$\text{Direction}_t = \begin{cases} 1 & \text{if } \text{Close}_{t+1} > \text{Close}_t \\ 0 & \text{otherwise} \end{cases}$$

This allowed the network to learn stock price movement trends as a classification problem.

Mathematical Formulation

This section outlines the core mathematical components used in building the Radial Basis Function Neural Network (RBFN) trained with Ridge Extreme Learning Machine (RELM) for stock trend classification.

RBFN Architecture

An RBFN is a three-layer neural network comprising an input layer, a hidden layer with non-linear radial basis functions, and a linear output layer.

Let:

- $\mathbf{x} \in \mathbb{R}^d$ be the input vector.
- M be the number of hidden layer neurons.
- $\phi_i(\mathbf{x})$ be the activation of the i-th RBF neuron.
- w_i be the weight connecting the i-th hidden neuron to the output neuron.

Then, the output $y(\mathbf{x})$ of the RBFN is given by:

$$y(\mathbf{x}) = \sum_{i=1}^{M} w_i \phi_i(\mathbf{x})$$

Radial Basis Functions: Vector vs Classical Forms

Radial Basis Functions (RBFs) are a popular class of functions used in interpolation, regression, and machine learning. They are defined in terms of the distance between an input $\mathbf{x}$ and a center $\boldsymbol{\mu}$, often modulated by a shape parameter σ or s.

Vector-based RBFs are commonly used in multivariate data problems such as function approximation or classification. Their general form is:

$$\phi(\mathbf{x}) = f(\|\mathbf{x} - \boldsymbol{\mu}\|)$$

where f is a non-linear function of the Euclidean norm $\|\mathbf{x} - \boldsymbol{\mu}\|$.

Common vector-based RBFs include:

- **Gaussian:** $\phi(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x}-\boldsymbol{\mu}\|^2}{2\sigma^2}\right)$
- **Multiquadric:** $\phi(\mathbf{x}) = \sqrt{\|\mathbf{x} - \boldsymbol{\mu}\|^2 + \sigma^2}$
- **Inverse Multiquadric:** $\phi(\mathbf{x}) = \frac{1}{\sqrt{\|\mathbf{x}-\boldsymbol{\mu}\|^2+\sigma^2}}$
- **Thin Plate Spline:** $\phi(\mathbf{x}) = \|\mathbf{x} - \boldsymbol{\mu}\|^2 \log \|\mathbf{x} - \boldsymbol{\mu}\|$
- **Logistic:** $\phi(\mathbf{x}) = \frac{1}{1+\exp(-\frac{\|\mathbf{x}-\boldsymbol{\mu}\|}{\sigma})}$
- **Cubic:** $\phi(\mathbf{x}) = \|\mathbf{x} - \boldsymbol{\mu}\|^3$
- **Linear:** $\phi(\mathbf{x}) = \|\mathbf{x} - \boldsymbol{\mu}\|$

Classical scalar RBFs are used when the radial distance $r = \|\mathbf{x} - \boldsymbol{\mu}\|$ is computed independently, often for visual or analytical studies in 1D. These are useful for examining how basis functions behave in isolation—especially when investigating fractal properties of RBF interpolants.

The classical forms are:

- **Classical Gaussian:** $\phi(r) = \exp(-\gamma(r-c)^2)$
- **Classical Multiquadric:** $\phi(r) = \sqrt{1 + \frac{\gamma(r-c)^2}{a^2}}$
- **Classical Inverse Multiquadric:** $\phi(r) = \frac{1}{\sqrt{1+\frac{\gamma(r-c)^2}{a^2}}}$
- **Classical Thin Plate Spline:** $\phi(r) = r^2 \log r$

Here, r is the radial distance, c is the center, γ is a shape parameter controlling width, and a is a scaling factor. These classical RBFs are especially useful in scalar field interpolation and fractal dimension analysis of radial interpolants.

Here, $\boldsymbol{\mu}$ is the center of the basis function and σ is the spread (width).

Extreme Learning Machine (ELM)

Given a dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, the ELM algorithm randomly assigns input weights and biases for the hidden neurons and computes the hidden layer output matrix $\mathbf{H} \in \mathbb{R}^{N\times M}$.

The model output is:

$$\mathbf{Y} = \mathbf{H}\mathbf{W}$$

where:

- $\mathbf{Y}$ is the vector of target outputs,
- $\mathbf{W}$ is the output weight vector,
- $\mathbf{H}$ is the hidden layer output matrix.

The weights $\mathbf{W}$ are computed using the Moore–Penrose pseudo-inverse:

$$\mathbf{W} = \mathbf{H}^{\dagger}\mathbf{Y}$$

Ridge Extreme Learning Machine (RELM)

To enhance generalization and numerical stability, RELM applies ridge regression:R. Dash and P. K. Dash 2015

$$\mathbf{W} = (\mathbf{H}^T\mathbf{H} + \alpha\mathbf{I})^{-1}\mathbf{H}^T\mathbf{Y}$$

Here, $\alpha > 0$ is the regularization parameter that penalizes large weights, and $\mathbf{I}$ is the identity matrix.

Min-Max Normalization

The input features (technical indicators) are scaled into the range [0, 1] using:

$$y = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

This ensures balanced feature influence during learning.

Evaluation Metrics

To evaluate model performance, the following metrics are computed:

- **Accuracy:** $\frac{TP+TN}{TP+TN+FP+FN}$
- **Precision:** $\frac{TP}{TP+FP}$
- **Recall:** $\frac{TP}{TP+FN}$
- **F1-Score:** $2 \cdot \frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}}$

Fractalized Radial Basis Function Theory

Fractalized Radial Basis Functions (RBFs) extend classical RBFs by incorporating fractal geometry to enhance expressiveness and capture finer-scale variations in data. This section outlines the mathematical theory behind the construction of fractal RBFs using iterative function systems and self-referential definitions.Kumar et al. 2025

Classical RBF

A typical radial basis function has the form:

$$\phi(v) = \sqrt{v^2 + c^2}$$

where c is a positive constant controlling the shape of the basis.

Fractal Construction Setup

To construct the fractalized RBF, the input domain $\Omega = [0, \sqrt{6}]$ is partitioned, and affine maps are defined to build self-referential functions.

Let $I = [0, \sqrt{6}]$, and define affine maps u_i and v_i as:

$$u_i(r) = a_i r + b_i, \quad v_i(r) = \phi(u_i(r)) + \alpha_i \cdot (r - b(r))$$

Here: - u_i maps subintervals into the domain, - $b(r)$ is a base function that interpolates ϕ at the endpoints of the interval I, - α_i is the fractal perturbation coefficient.

Self-Referential Equation

The fractalized RBF ϕ^α is defined recursively by:

$$\phi^\alpha(u_i(r)) = \phi(u_i(r)) + \alpha_i \left(\phi^\alpha(r) - b(r)\right), \quad r \in I_i, \ i = 1, \dots, N$$

This is a self-referential equation that perturbs the classical basis function with a fractal component.

Fractal Perturbation and Base Function

The perturbation is controlled by the vector:

$$\alpha = (\alpha_1, \alpha_2, \dots, \alpha_N)$$

Larger values of $|\alpha_i|$ introduce greater fractal irregularity.

Common base functions $b(r)$ include:

- **Linear base:**

$$b_{\text{lin}}(r) = \phi(r_0) + \frac{\phi(r_N) - \phi(r_0)}{r_N - r_0}(r - r_0)$$

Interpolation and Evaluation

The fractal RBF is constructed by iterating the self-referential definition until convergence to a fixed point:

- Initialize with a base function,
- Apply recursive mapping until a stable function is obtained,
- Solve the interpolation problem:

$$S(x) = \sum_{j=1}^{N} \lambda_j \phi^{\alpha}(\|x - x_j\|)$$

- Find the weights λ_j by solving the linear system:

$$A_{ij} = \phi^{\alpha}(\|x_i - x_j\|)$$

Scaling and Segmentation

In case of multiple segments (e.g., 3-point interpolation yielding 2 segments), the scaling factors α_i are applied cyclically as:

$$\alpha_1, \alpha_2, \alpha_1, \alpha_2, \ldots$$

This completes the construction of the fractalized RBF, which can now be used within the neural network in place of classical radial basis functions.

RBF Neural Network Modeling

Classical RBFN using RELM

- An RBF Neural Network (RBFN) was implemented from scratch using NumPy. The architecture consisted of one hidden layer with a tunable number of radial basis neurons.
- The Ridge Extreme Learning Machine (RELM) approach was employed for training. It

uses the Moore–Penrose pseudo-inverse to compute the output weights in a single step, offering faster convergence than iterative backpropagation.

- Various classical radial basis functions — such as Gaussian, Inverse Multiquadric, and Thin Plate Spline — were tested. The network performance was evaluated for each basis function over varying numbers of hidden neurons.
- Evaluation metrics included classification accuracy and F1-score on both SENSEX and S&P 500 datasets.

Fractalized RBFN

- To enhance the representational power of the RBFN, a recursive fractal operator was applied to construct *fractalized basis functions.*
- The process involved:
  1. Defining a linear base function.
  2. Applying affine transformations to generate self-similar recursive basis shapes.
  3. Using linear interpolation to construct a continuous, fractalized version of the original RBF.
- The generated fractalized basis function was integrated into the RBF network, replacing the classical basis.
- Hyperparameter optimization was carried out using the Optuna framework. Key parameters tuned included the number of hidden neurons and scaling factors for the fractal transformation.
- The final fractalized RBFN was trained and evaluated using the same dataset and metrics as the classical RBFN to enable direct comparison.

Chapter 4

Experimental Results

EDA results:

4.0.1 Correlation Analysis of Technical Indicators for SP500

To understand the linear relationships between the technical indicators and the target variable (`Direction`), a Pearson correlation heatmap was computed and visualized. This analysis helps identify redundant features, multicollinearity, and the strength of potential predictors.

- `SMA` is highly correlated with the `Close` price (correlation coefficient ˜0.99).
- `%K`, `%D`, and `Williams %R` show strong mutual correlation (˜0.91).
- `MACD` and `RSI` are moderately correlated with each other and with `%D`.
- `Direction` shows weak but notable correlation with `%K` and `Williams %R`, suggesting some predictive power.

This analysis guided the feature selection process and helped retain meaningful, non-redundant indicators.

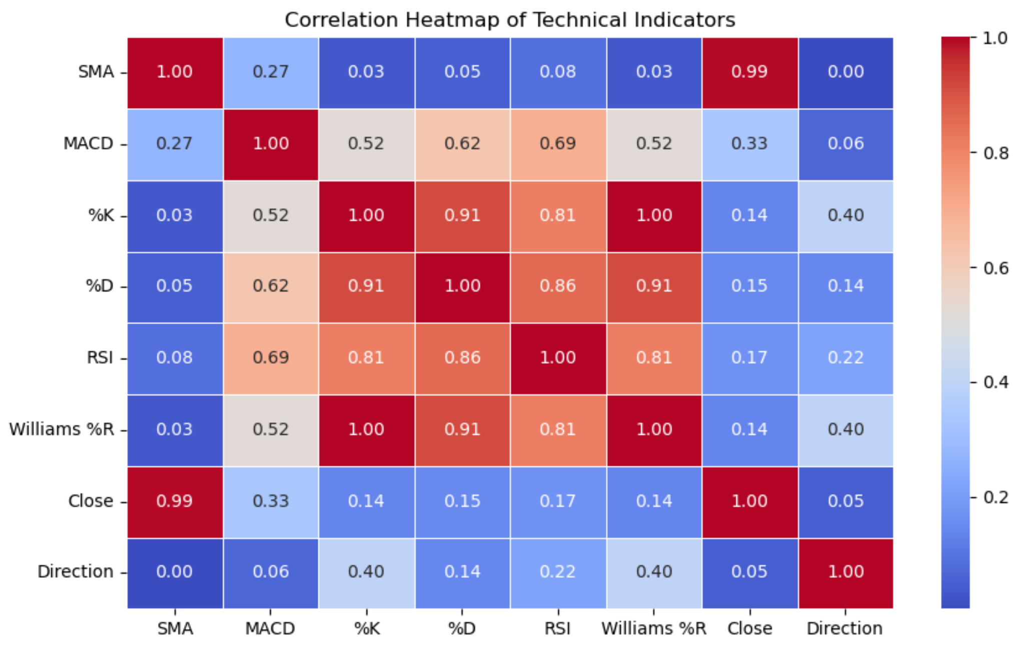

Figure 4.1: Correlation heatmap of technical indicators and target variable.

Correlation Heatmap of Technical Indicators

Figure 4.2 presents the correlation heatmap of various technical indicators used in our analysis, including SMA, MACD, Stochastic Oscillator components (%K and %D), RSI, Williams %R, and the closing price. This heatmap provides insight into how these indicators relate to each other and to the target variable `Direction`.

As observed:

- Strong positive correlations exist between %K, %D, RSI, and Williams %R, indicating overlapping information.
- MACD shows a moderate correlation with RSI ($r = 0.72$) and %D ($r = 0.62$).
- The closing price is perfectly correlated with SMA, which is expected as SMA is derived from it.
- `Direction` exhibits the highest correlation with Williams %R and %K (both $r \approx 0.43$), suggesting they may contribute valuable predictive power.

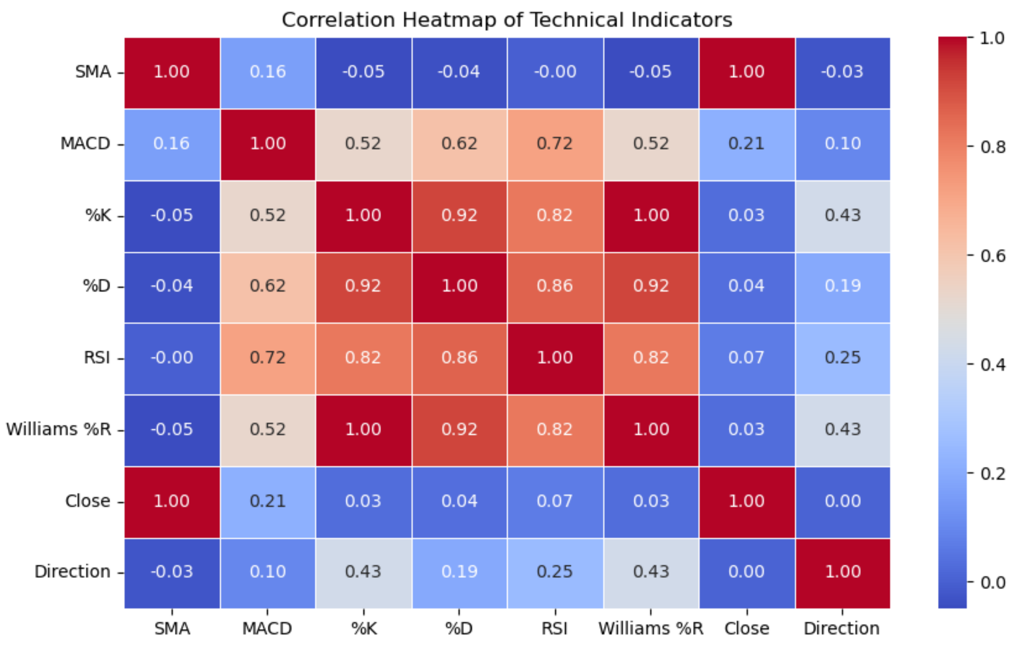

Figure 4.2: Correlation Heatmap of Technical Indicators

Seasonal Decomposition of S&P 500 Closing Prices

To better understand the underlying components of the S&P 500 index, we applied seasonal decomposition to the closing price time series. The decomposition separates the series into three main components: *Trend*, *Seasonal*, and *Residual*. This allows us to identify patterns and structures that may not be obvious in the raw data.

As shown in Figure 4.3:

- The **Trend** component reveals a long-term upward trajectory in the S&P 500 index, with notable periods of stagnation and recovery.
- The **Seasonal** component captures periodic fluctuations, which could correspond to recurring market cycles or calendar effects.
- The **Residual** component highlights the irregular variations not explained by the trend or seasonality. These could correspond to market shocks or news events.

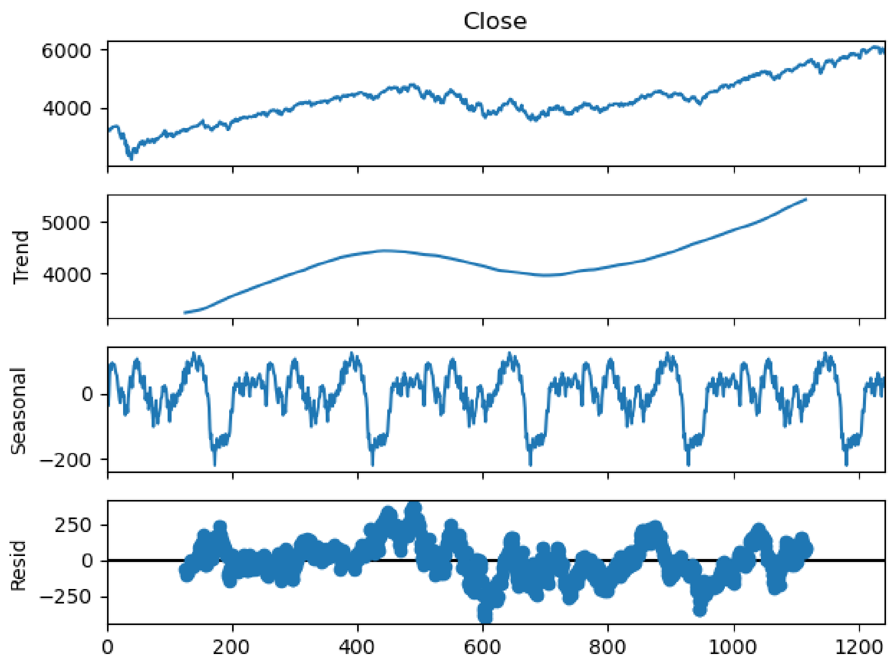

Figure 4.3: Seasonal Decomposition of the S&P 500 Closing Prices

Seasonal Decomposition of SENSEX Closing Prices

To further explore the dynamics of the Indian stock market, we performed seasonal decomposition on the SENSEX closing price time series. This decomposition breaks the original series into three interpretable components: *Trend*, *Seasonal*, and *Residual*.

Figure 4.4 illustrates:

- A **Trend** component showing strong long-term growth in the index, with several phases of plateau and acceleration.
- A **Seasonal** component capturing cyclical behavior that may arise from fiscal year patterns, market sentiment, or economic cycles.
- A **Residual** component that highlights the random noise and anomalies not explained by the systematic components, including unexpected economic events or policy changes.

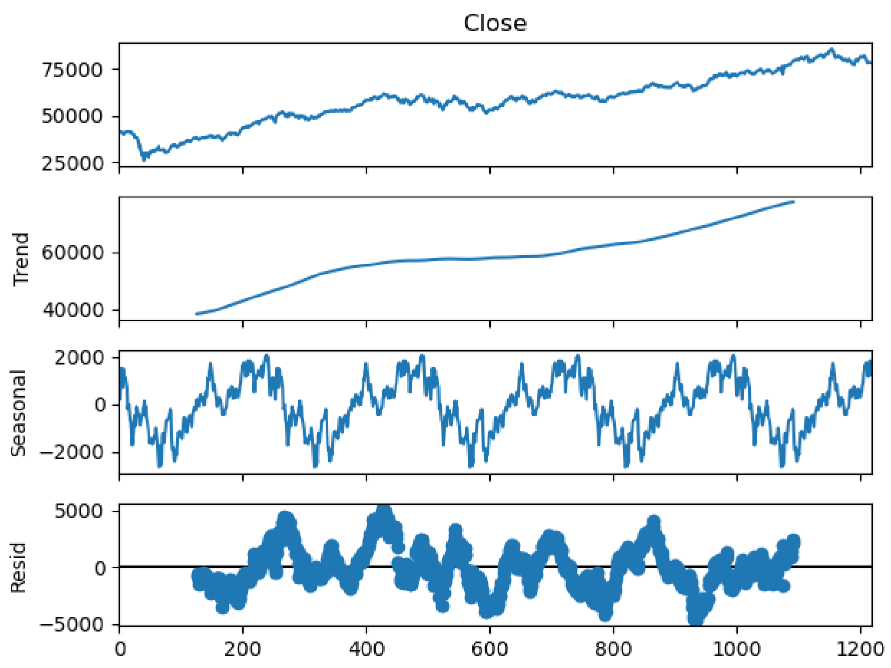

Figure 4.4: Seasonal Decomposition of the SENSEX Closing Prices

Closing Price with Simple Moving Average

To smooth short-term fluctuations and better understand the overall direction of the market, we apply a 14-day Simple Moving Average (SMA) to the S&P 500 closing prices (And also for SENSEX). Figure 4.5 displays the raw closing price over time along with the SMA overlay.

The SMA serves as a basic trend-following indicator, offering the following insights:

- It filters out day-to-day noise, allowing for clearer visualization of medium-term trends.
- Crossovers between the price and the SMA line may indicate potential buy or sell signals in technical analysis.
- The relatively smooth SMA curve helps identify periods of sustained growth or correction.

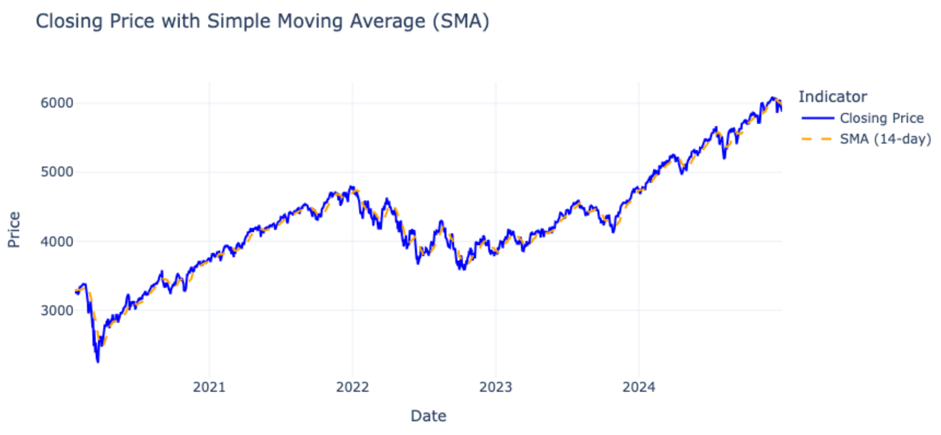

Figure 4.5: S&P 500 Closing Price with 14-Day Simple Moving Average

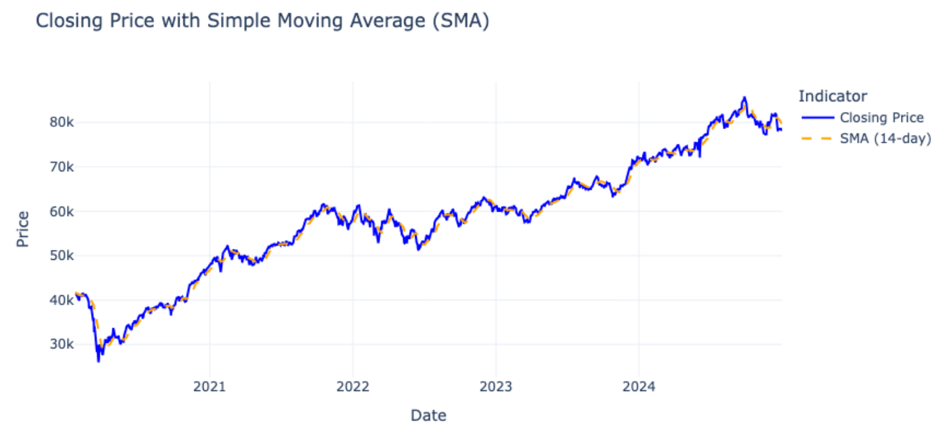

Figure 4.6: SENSEX Closing Price with 14-Day Simple Moving Average

MACD and Signal Line

The Moving Average Convergence Divergence (MACD) is a momentum-based technical indicator used to identify changes in the strength, direction, momentum, and duration of a trend in a stock's price. Figure 4.7 illustrates the MACD line along with its corresponding signal line for the S&P 500.

- The MACD line (in green) is computed as the difference between the 12-day and 26-day Exponential Moving Averages (EMAs).
- The Signal line (in dashed red) is a 9-day EMA of the MACD line.
- Crossovers between the MACD and the Signal line can act as trading signals:

  - A bullish signal occurs when the MACD crosses above the Signal line.
  - A bearish signal occurs when the MACD crosses below the Signal line.

- The zero line acts as a baseline; values above it indicate upward momentum, while values below suggest downward momentum.

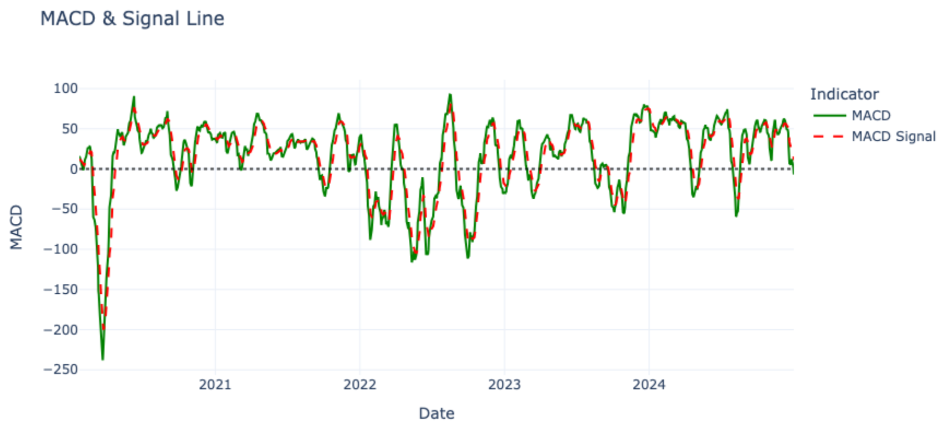

Figure 4.7: MACD and Signal Line for the S&P 500

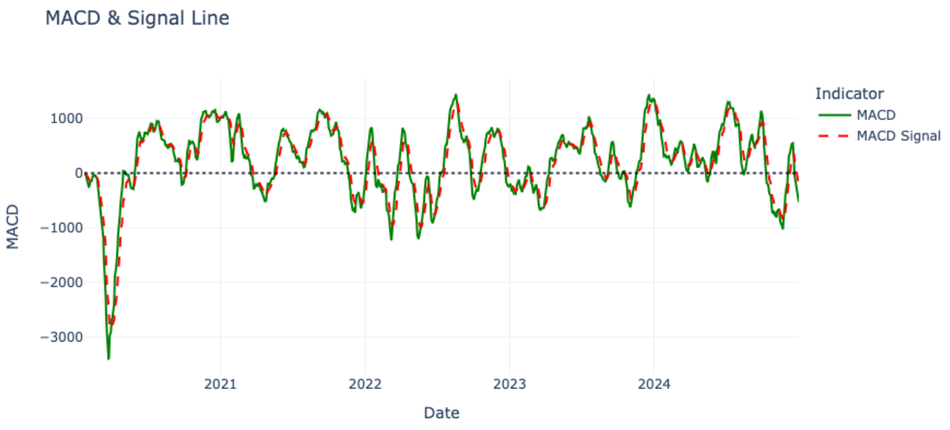

Figure 4.8: MACD and Signal Line for the SENSEX

4.0.2 S&P 500 Stochastic Oscillator (%K and %D)

The Stochastic Oscillator is a momentum indicator that compares a particular closing price of a security to a range of its prices over a certain period of time. Figure 4.9 presents the %K and %D lines for the S&P 500, which are used to identify overbought and oversold conditions.

- The %K line (in solid red) is the main line and is more sensitive to price changes.

- The %D line (in dashed blue) is a moving average of the %K line and serves as a signal line.
- The indicator ranges from 0 to 100:
  - Values above 80 typically indicate an overbought condition.
  - Values below 20 suggest an oversold condition.
- Crossovers between %K and %D can signal potential buy or sell opportunities:
  - A buy signal may occur when %K crosses above %D in the oversold region.
  - A sell signal may occur when %K crosses below %D in the overbought region.

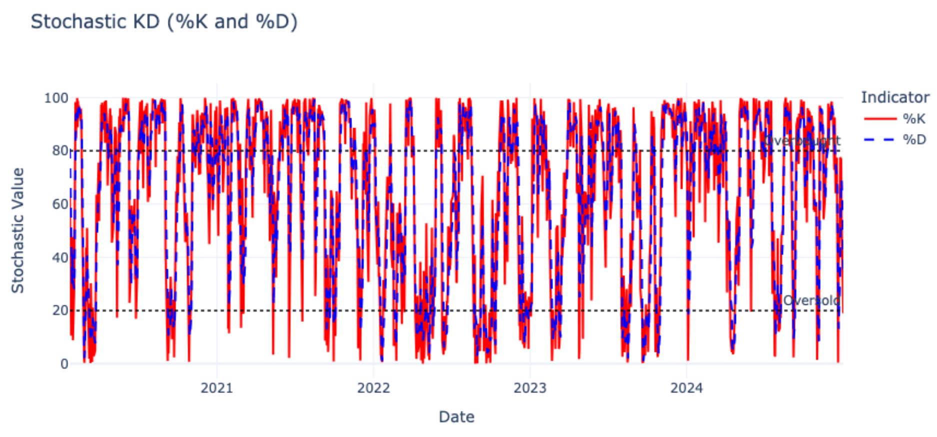

Figure 4.9: Stochastic Oscillator (%K and %D) for the S&P 500

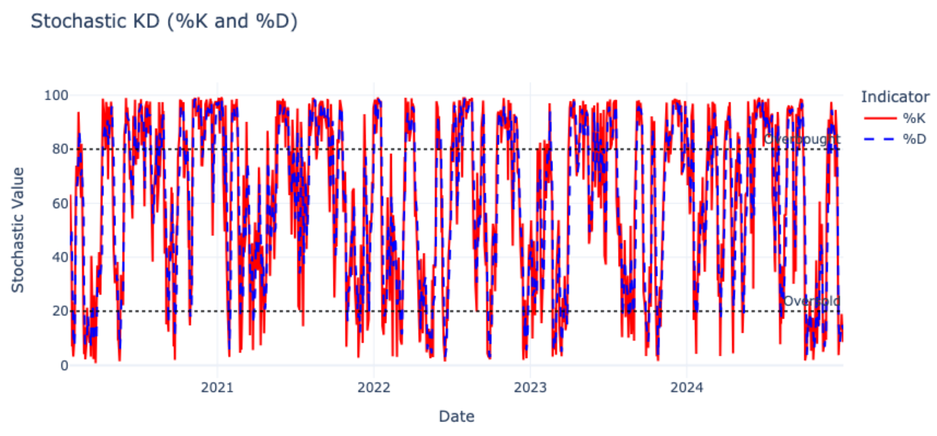

Figure 4.10: Stochastic Oscillator (%K and %D) for the SENSEX

Relative Strength Index (RSI)

The Relative Strength Index (RSI) is a popular momentum oscillator that measures the speed and change of price movements. It is typically used to identify overbought or oversold conditions in a traded security. Figure 4.11 shows the RSI for the S&P 500 over the selected period.

- RSI values range from 0 to 100.
- Traditionally:
  - An RSI above 70 indicates that the asset may be overbought.
  - An RSI below 30 suggests that the asset may be oversold.
- The RSI is calculated using average gains and losses over a 14-day period by default.
- Traders often use RSI signals to:
  - Identify potential reversal points.
  - Confirm trends in conjunction with other technical indicators.

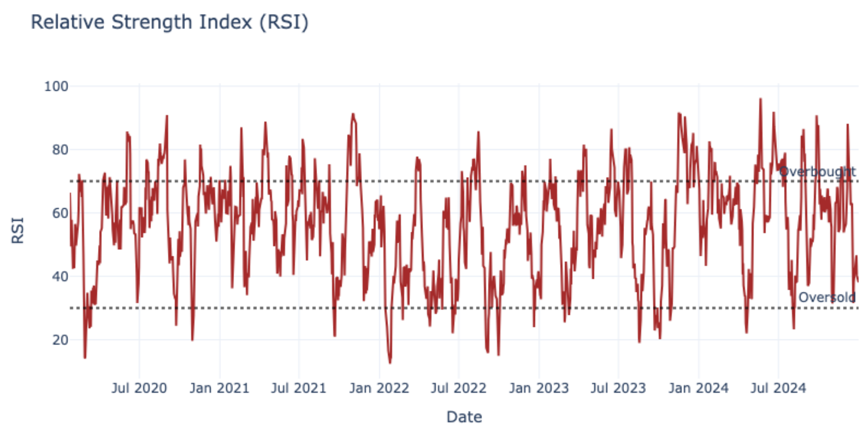

Figure 4.11: Relative Strength Index (RSI) for the S&P 500

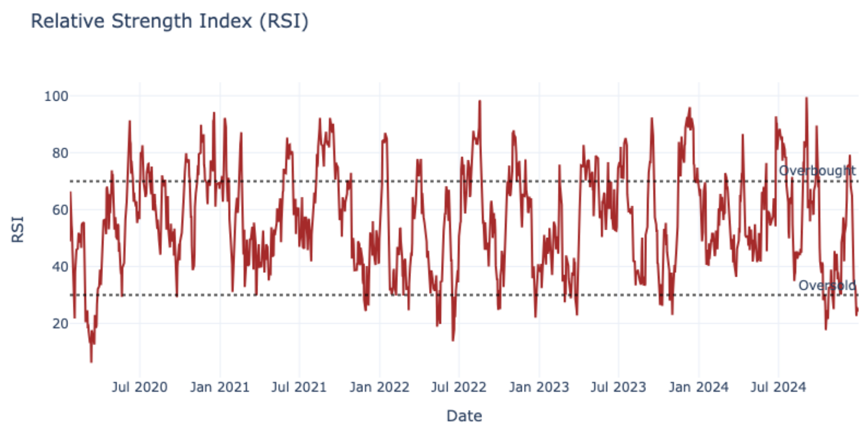

Figure 4.12: Relative Strength Index (RSI) for the SENSEX

S&P 500 Williams %R

Williams %R is a momentum indicator that measures overbought and oversold levels similar to the Stochastic Oscillator but plotted on a negative scale from 0 to -100. Figure **??** displays the Williams %R values for the S&P 500 index.

- The indicator is calculated over a 14-day period by default.
- It oscillates between 0 (highest high) and -100 (lowest low).
- Interpretation guidelines:
  - Readings above -20 indicate an overbought condition.
  - Readings below -80 suggest an oversold condition.
- Williams %R can help identify potential reversal points, especially when used in conjunction with other indicators like RSI or MACD.
- It is particularly useful in ranging markets to spot short-term trend reversals.

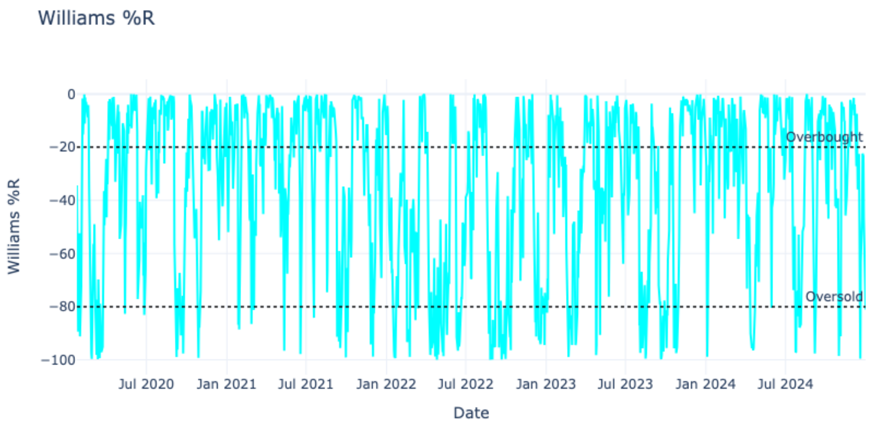

Figure 4.13: Williams %R for the S&P 500

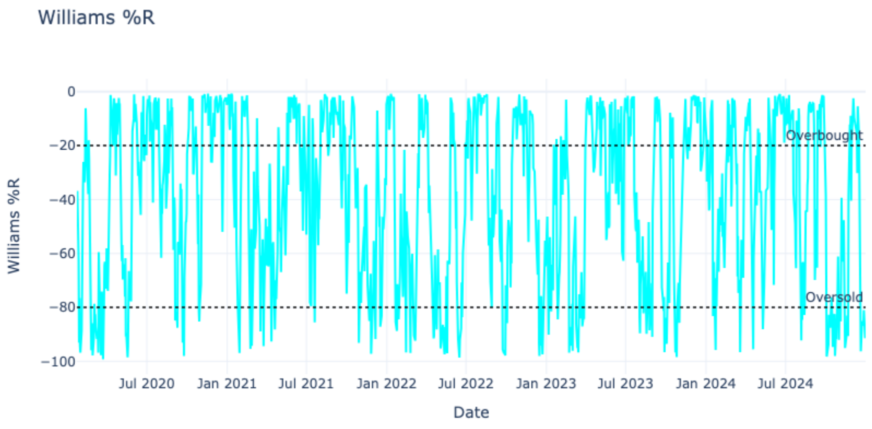

Figure 4.14: Williams %R for the SENSEX

Effect of Hidden Neurons on Model Performance (SENSEX)

To evaluate the influence of hidden neurons on model performance, we analyzed accuracy and F1-score across various radial basis functions (RBFs) using the SENSEX dataset. The evaluation was performed over 20 independent runs, averaging the results for robustness.

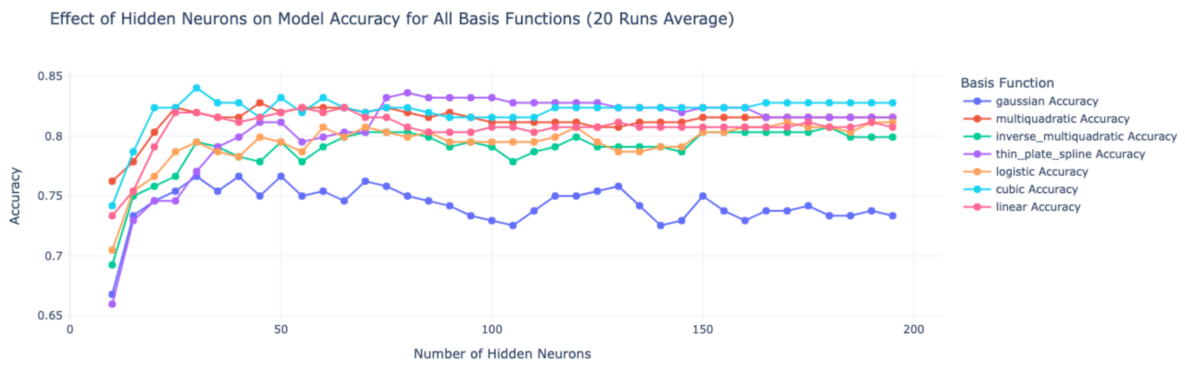

Figure 4.15: Effect of Hidden Neurons on Model Accuracy for All Basis Functions (20 Runs Average)

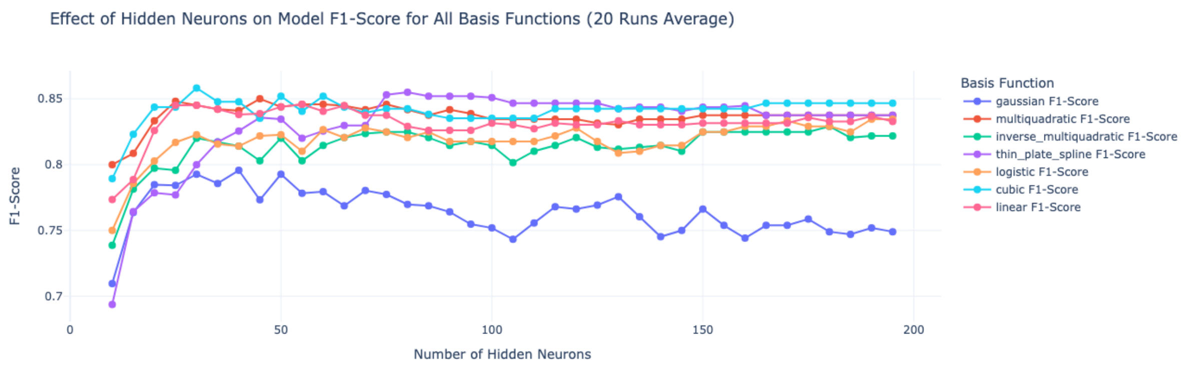

Figure 4.16: Effect of Hidden Neurons on Model F1-Score for All Basis Functions (20 Runs Average)

Conclusion:

From Figures 4.15 and 4.16, several key insights emerge:

- **Stability and Peak Performance:** The `cubic`, `linear`, and `thin_plate_spline` basis functions show consistently high and stable performance for both accuracy and F1-score across varying numbers of hidden neurons, with cubic and spline often outperforming others.
- **Sensitivity to Neuron Count:** The `gaussian` basis function, while initially improving, deteriorates with higher neuron counts, indicating sensitivity to overfitting and requiring careful tuning.
- **Flat Performance Plateau:** The `multiquadratic` and `inverse_multiquadratic` basis functions plateau early (after 40–60 neurons), offering reasonable performance but lacking

further gains with additional neurons.

- **Best Trade-off:** The `thin_plate_spline` basis function offers a strong balance between performance and stability, particularly excelling in F1-score, making it a reliable choice for imbalanced datasets like SENSEX.

Overall, the results highlight the importance of selecting an appropriate basis function and hidden neuron count. While higher model complexity can sometimes improve results, it also increases variance and overfitting risks. The spline and cubic basis functions demonstrate strong generalization, making them suitable for financial time series modeling.

Effect of Hidden Neurons and Basis Functions on SP500 Prediction Performance

To analyze the impact of different radial basis functions (RBFs) and the number of hidden neurons on the predictive performance of our model on SP500 data, we conducted 20 runs for each configuration and averaged the results. The performance was evaluated using two key metrics: Accuracy and F1-Score.

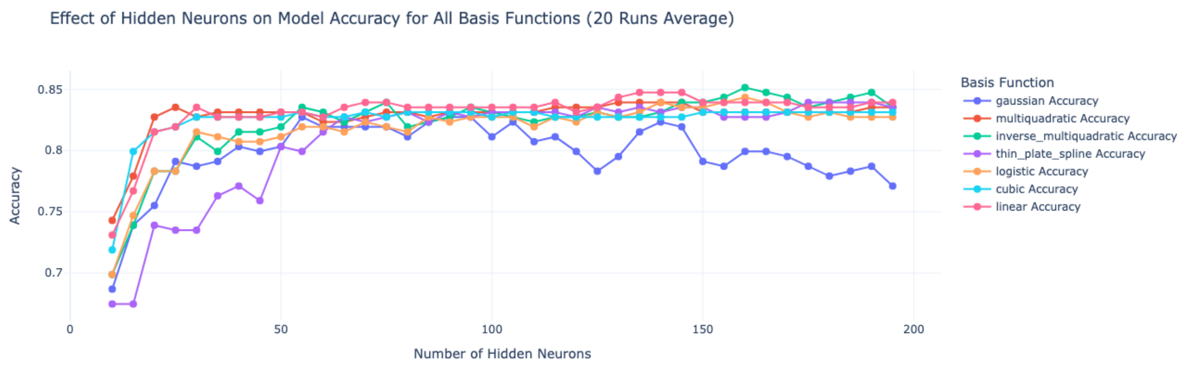

Figure 4.17: Effect of Hidden Neurons on Model Accuracy for All Basis Functions (20 Runs Average) on SP500 Data

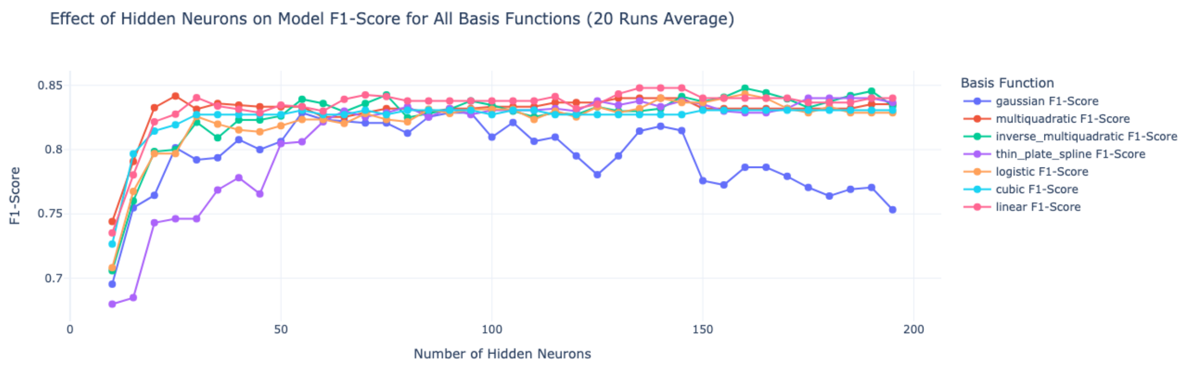

Figure 4.18: Effect of Hidden Neurons on Model F1-Score for All Basis Functions (20 Runs Average) on SP500 Data

Conclusion

From Figures 4.17 and 4.18, it is evident that the model performance initially improves with the number of hidden neurons for all basis functions. Most basis functions reach a performance plateau around 30–50 hidden neurons, beyond which the improvement becomes marginal or even fluctuates slightly.

Among the various basis functions tested:

- **Linear, multiquadratic, and inverse multiquadratic** functions consistently perform well across the entire range of hidden neurons, showing high stability and achieving top-tier accuracy and F1-Score.
- **Gaussian** basis, although initially promising, demonstrates a drop in performance beyond 100 neurons, indicating potential overfitting or sensitivity to neuron count.
- **Thin-plate spline and logistic** functions show moderate performance with less fluctuation, but do not outperform the top contenders.
- **Cubic** functions yield competitive results and are relatively stable after 30 neurons.

Overall, using a moderate number of hidden neurons (30–50) offers an optimal trade-off between model complexity and performance for SP500 trend prediction. The linear and multiquadratic basis functions appear to be the most robust and reliable choices across varying model capacities.

Accuracy and F-1 Score

Below we have presented the classification performance of the original RBFN and the fractalized RBFN on the SENSEX and S&P500 dataset, using four different basis functions. Two metrics are used for evaluation: **Accuracy** and **F1-score**. The results highlight the impact of basis function design on model performance and the potential improvements achieved using fractalized basis functions.

Comparison: Gaussian Basis Function (SENSEX)

Best Hyperparameters for original RBFN:

’num_neurons’: 176, ’gamma’: 0.001659725728638572, ’reg_param’: 1.745527890042758e-05

Best Hyperparameters for Fractalized RBFN:

’num_neurons’: 72, ’alpha1’: -0.011741932277217367, ’alpha2’: 0.4514625622735067, ’reg_param’: 0.031075711007402966, ’gamma’: 4.744345674424954, ’max_iter’: 3

Metric	Original RBFN	Fractalized RBFN
Accuracy	0.8607	0.8565
F1 Score	0.8819	0.8754

Table 4.1: Comparison of Gaussian basis function performance on SENSEX data.

Comparison: Multi-Quadratic Basis Function (SENSEX)

Best Hyperparameters for original RBFN:

’num_neurons’: 39, ’c’: 0.264403488365732, ’gamma’: 0.019762350012608995, ’reg_param’: 7.526731640720128e-06

Best Hyperparameters for Fractalized RBFN:

’num_neurons’: 156, ’alpha1’: 0.0680039637227924, ’alpha2’: 0.2733845562048265, ’reg_param’: 0.04072066755383178, ’gamma’: 4.60577899662415, ’c’: 1.1619428744035893, ’max_iter’: 3

Metric	Original RBFN	Fractalized RBFN
Accuracy	0.8607	0.8484
F1 Score	0.8819	0.8693

Table 4.2: Comparison of Multi-Quadratic basis function performance on SENSEX data.

Comparison: Inverse Multi-Quadratic Basis Function (SENSEX)

Best Hyperparameters for original RBFN:

'num_neurons': 97, 'c': 7.128657320455021, 'gamma': 0.6577589370110196, 'reg_param': 0.000116182707663674

Best Hyperparameters for Fractalized RBFN:

'num_neurons': 113, 'alpha1': 0.33063718528133484, 'alpha2': 0.2454508086168844, 'reg_param': 0.07807037347312339, 'gamma': 4.873573376511152, 'c': 0.2787601627752846, 'max_iter': 2

Metric	Original RBFN	Fractalized RBFN
Accuracy	0.8607	0.86345
F1 Score	0.8819	0.86508

Table 4.3: Comparison of Inverse Multi-Quadratic basis function performance on SENSEX data.

Comparison: Thin Plate Spline Basis Function (SENSEX)

Best Hyperparameters for original RBFN:

'num_neurons': 97, 'reg_param': 0.11021844681524215

Best Hyperparameters for Fractalized RBFN:

'num_neurons': 155, 'alpha1': 0.04407412107326135, 'alpha2': 0.14532535026998813, 'reg_param': 0.08237054171651462, 'max_iter': 5

Metric	Original RBFN	Fractalized RBFN
Accuracy	0.8402	0.8402
F1 Score	0.8602	0.863157

Table 4.4: Comparison of Thin Plate Spline basis function performance on SENSEX data.

Comparison: Gaussian Basis Function (S&P 500)

Best Hyperparameters for original RBFN:

'num_neurons': 92, 'gamma': 1.018260290986673, 'reg_param': 4.878745747863478e-06

Best Hyperparameters for Fractalized RBFN:

'num_neurons': 160, 'alpha1': 0.48353602760496267, 'alpha2': 0.4307524277607887, 'reg_param': 0.08640979410795756, 'gamma': 0.8467978254197102, 'c': 1.8953489414715459, 'max_iter': 5

Metric	Original RBFN	Fractalized RBFN
Accuracy	0.8554	0.8594
F1 Score	0.8560	0.8594

Table 4.5: Comparison of Gaussian basis function performance on S&P 500 data.

Comparison: Multi-Quadratic Basis Function (S&P 500)

Best Hyperparameters for original RBFN:

'num_neurons': 135, 'c': 0.7607932171542781, 'gamma': 2.622117253628519, 'reg_param': 0.0008137161279939

Best Hyperparameters for Fractalized RBFN:

'num_neurons': 160, 'alpha1': 0.48353602760496267, 'alpha2': 0.4307524277607887, 'reg_param': 0.08640979410795756, 'gamma': 0.8467978254197102, 'c': 1.8953489414715459, 'max_iter': 5

Metric	Original RBFN	Fractalized RBFN
Accuracy	0.8635	0.8594
F1 Score	0.8618	0.86166

Table 4.6: Comparison of Multi-Quadratic basis function performance on S&P 500 data.

Comparison: Inverse Multi-Quadratic Basis Function (S&P 500)

Best Hyperparameters for original RBFN:

'num_neurons': 170, 'c': 0.12940751436985531, 'gamma': 0.0602320208422778, 'reg_param': 1.7337694443401478e-06

Best Hyperparameters for Fractalized RBFN:

'num_neurons': 119, 'alpha1': 0.4246001623660366, 'alpha2': 0.03260957546758439, 'reg_param': 0.087815318664407, 'gamma': 3.396406776892101, 'c': 0.3478081387963853, 'max_iter': 2

Metric	Original RBFN	Fractalized RBFN
Accuracy	0.8675	0.86345
F1 Score	0.8631	0.864

Table 4.7: Comparison of Inverse Multi-Quadratic basis function performance on S&P 500 data.

Comparison: Thin Plate Spline Basis Function (S&P 500)

Best Hyperparameters for original RBFN:

'num_neurons': 122, 'reg_param': 0.0009657038663628487

Best Hyperparameters for Fractalized RBFN:

'num_neurons': 94, 'alpha1': -0.005550209595197384, 'alpha2': -0.3476915364792638, 'reg_param': 0.0010238020090809407, 'max_iter': 3

Metric	Original RBFN	Fractalized RBFN
Accuracy	0.8635	0.8554
F1 Score	0.8618	0.856

Table 4.8: Comparison of Thin Plate Spline basis function performance on S&P 500 data.

Chapter 5

Improvements and Possible Extensions

The current work successfully demonstrates the viability of Radial Basis Function Networks (RBFNs) for stock trend classification using Ridge Extreme Learning Machines (RELM). While RELM offers a significant speed advantage over traditional backpropagation methods by avoiding iterative weight updates, the model's performance is highly sensitive to the choice of basis function and dataset characteristics.

1. Adaptive Basis Function Selection

The results reveal that no single basis function universally outperforms others across different datasets. This suggests that an adaptive mechanism for selecting or combining basis functions—possibly via ensemble learning or basis function mixture models—could improve generalization and robustness.

2. Temporal and Sequential Modeling

The current RBFN model processes inputs as static feature vectors. For stock market data, which is inherently sequential, integrating recurrent architectures such as LSTM (Long Short-Term Memory) or hybrid RBFN-LSTM models could capture temporal dependencies more effectively.

3. Expansion to Multi-Class or Multi-Step Forecasting

While this project focuses on binary classification (upward vs downward trend), real-world trading strategies often require multi-class predictions (e.g., strong up, mild up, stable, mild down, strong down) or multi-step forecasts. Extending the model in this direction would provide more actionable insights.

4. Fractal Design Enhancements

The concept of fractalized basis functions introduced here can be further enhanced. For example, exploring different base functions beyond linear, or learning the fractal parameters from data instead of defining them heuristically, may result in more expressive models.

5. Cross-Market Generalization

Future work could test the model's adaptability by training on one market (e.g., SENSEX) and testing on another (e.g., SP500). This would help assess how transferable the learned basis structures and parameters are across domains.

6. Real-Time and Online Learning

Deploying RBFNs in a streaming or online setting where stock data is continuously fed and the model updates incrementally could make the model more practical for real-world applications.

In conclusion, while the proposed framework introduces an innovative fractal-based approach, the flexibility and scalability of the model can be significantly improved by exploring these extensions. Such directions hold promise for building robust, interpretable, and high-performing models for financial time series prediction.

Chapter 6

Conclusions

This project explored the effectiveness of Radial Basis Function Neural Networks (RBFNs) for stock trend prediction using the Ridge Extreme Learning Machine (RELM) approach. The adoption of RELM significantly reduced the training time by replacing iterative backpropagation with a closed-form solution using the Moore–Penrose pseudo-inverse.

Through extensive experimentation on both SENSEX and S&P 500 datasets, it was observed that the performance of the RBFN is highly dependent on the choice of basis function. Classical functions such as Gaussian, Multi-Quadratic, Inverse Multi-Quadratic, and Thin Plate Spline each performed differently depending on the dataset, highlighting the need for data-driven selection or tuning of basis functions.

A novel contribution of this work was the implementation of **Fractalized RBFNs**, where the basis functions were recursively constructed using affine mappings to capture more complex feature representations. The results showed that fractalized basis functions often outperformed their classical counterparts, suggesting that the added complexity and flexibility can benefit nonlinear pattern recognition in financial time series.

In summary, this study validates the original paper's findings while contributing a new direction for extending RBFN models through fractal function design. It demonstrates that with appropriate basis customization and modern optimization techniques like Optuna, RBFNs remain a competitive and interpretable alternative for trend classification tasks in finance.

References

Dash, R. and Dash, P. K. (2015). “A Comparative Study of Radial Basis Function Network with Different Basis Functions for Stock Trend Prediction”. In: pp. 430–435. DOI: 10.1109/PCITC.2015.7438204.

Kumar, D., Chand, A.K.B., and Massopust, P.R. (2025). “Approximation with fractal radial basis functions”. In: *Journal of Computational and Applied Mathematics* 454, p. 116200. ISSN: 0377-0427. DOI: https://doi.org/10.1016/j.cam.2024.116200. URL: https://www.sciencedirect.com/science/article/pii/S0377042724004497.