A FIELD PROJECT REPORT

on

**“Stock Price Forecasting by Time Series Analysis”**

**Submitted**

By

|  |  |
| --- | --- |
| 221FA04150  Mannem Poorna Sai  221FA04133  Vandanapu Reshma Surekha | 221FA04120  Adarsh Kumar Jha  221FA04507  Divya Gupta |
|  | |

**Under the guidance of**

*Maridu Bhargavi*

*Assistant Professor Department of CSE,VFSTR*



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH**

**Vadlamudi, Guntur.**

**ANDHRA PRADESH, INDIA, PIN-522213.**



**CERTIFICATE**

This is to certify that the Field Project entitled **“Stock Price Forecasting by Time Series Analysis”** that is being submitted by 221FA04133 (Vandanapu Reshma Surekha), 221FA04507 (Divya Gupta), 221FA04150 (Mannem Poorna Sai)**,**221FA04120(Adarsh Kumar Jha) for partial fulfilment of Field Project is a bonafide work carried out under the supervision of Ms. Maridu Bhargavi, M.Tech., Assistant Professor, Department of CSE.

|  |  |  |
| --- | --- | --- |
| M. Bhargavi |  | Dr.K.V. Krishna Kishore |
| Assistant Professor, CSE | HOD,CSE | Dean, SoCI |



**DECLARATION**

We hereby declare that the Field Project entitled **“Stock Price Forecasting by Time Series Analysis”** is being submitted by 221FA04133 (Vandanapu Reshma Surekha), 221FA04507 (Divya Gupta), 221FA04150 (Mannem Poorna Sai)**,**221FA04120(Adarsh Kumar Jha) in partial fulfilment of Field Project course work. This is our original work, and this project has not formed the basis for the award of any degree. We have worked under the supervision of Ms. Maridu Bhargavi, M.Tech., Assistant Professor, Department of CSE.

**By**

**221FA04133 ((Vandanapu Reshma Surekha),**

**221FA04507 (Divya Gupta),**

**221FA04150 (Mannem Poorna Sai),**

**221FA04120 (Adarsh Kumar Jha)**

Date:

## ABSTRACT

In the context of technological advancement and mountains of financial data now available, more than ever it is crucial for investors, financial institutions, and decision-makers to make an informed choice by obtaining accurate stock price forecasting. Traditional techniques of forecasting often fail in today’s complex market dynamics. This paper reveals a number of techniques of machine learning that can be applied to the field of stock price forecasting, de-emphasizing building up the accuracy and reliability of such predictions.[2] The main key algorithms presented are ElasticNet for feature selection and XGBoost, CatBoost, and an ensemble Voting Regressor. Its ability to analyze series data and identify strong, non-trivial patterns from the time series of stock price data

made this model the best choice. Results show a performance that indicates just how well the ensemble Voting Regressor performed over the rest, with accuracy 99.6% far beyond any model from the bunch, promising much sharper resolution in financial forecasting. XGBoost and CatBoost also had very comparable accuracy that assured the productive use of machine learning in applying it for financial analytics. The research is not only a glorification of benefits from application of developed models of machine learning for predicting stock prices but also introduces points for further refinement of methods of prediction[4] Keywords: Prediction of Stock Price using Machine Learning. It needs ElasticNet, XGBoost, Cat- Boost, and Voting Regressor. Financial Forecasting Accuracy.

**TABLE OF CONTENTS**

| **Chapter** | **Title** | **Page** |
| --- | --- | --- |
| 1 | **Introduction** | |
| 1.1 | Motivation | 2 |
| 1.2 | Problem Definition/Research Gaps | 2 |
| 1.3 | Limitations | 2 |
| 1.4 | Design Standards | 3 |
| 1.5 | Major Contributions/Objectives | 3 |
| 2 | **Literature Survey** | |
| 2.1 | Literature review | 5-7 |
| 2.2 | Motivation | 7 |
| 3 | **Proposed System** | |
| 3.1 | Input Dataset | 9 |
| 3.2 | Data Pre-processing | 10 |
| 3.3 | Model Building | 11 |
| 3.4 | Methodology of the System | 11 |
| 3.5 | Model Evaluation | 12 |
| 3.6 | Constraints | 13 |
| 3.7 | Proposed Model | 13 |
| 4 | **Implementation** | |
| 4.1 | Environment Setup and Code | 15-21 |
| 4.2 | Proposed Model | 22 |
| 5 | **Experimentation and Result Analysis** | 23-27 |
| 6 | **Conclusion** | 28-29 |
| 7 | **References** | 30-31 |

**LIST OF FIGURES**

|  |  |
| --- | --- |
| Figure 1. Flowchart for Methodology | 22 |
| Figure 2. Comparison of Model Accuracy | 26 |
| Figure 3. Comparison of Regression Models | 26 |

**LIST OF TABLES**

Table - 1:Performance metrics of the proposed models. 27

Table - 2:Comparision of Performance Metrics. 27

# CHAPTER-1 INTRODUCTION

### INTRODUCTION

**1.1 Motivation**

Technological Advancements & Big Data: Highlight the need for robust methods due to the immense amount of financial data now available, making it difficult for traditional forecasting methods to keep up.

Importance for Investors: The significance of stock price prediction for investors and financial institutions to make informed decisions and reduce risks.

Limitations of Traditional Methods: Mention how traditional methods fail in the complex, volatile market environments, leading to interest in machine learning for more accurate predictions.

**1.2 Problem Definition / Research Gaps**

Non-Stationary Time Series Problem: Stock prices are difficult to predict due to their non-linear and volatile nature. Traditional models struggle with non-stationary data, which is a key challenge.

Gaps in Existing Forecasting Models: Discuss the inadequacies in traditional time-series methods like ARIMA and other simple regression models in capturing the complexity of financial markets.

Need for Advanced Models: The need for techniques like ensemble learning (XGBoost, CatBoost) that can better handle this complexity, which current literature doesn't fully explore.

**1.3 Limitations**

Volatility in Stock Markets: Acknowledge that despite improvements in prediction models, the unpredictable nature of stock markets due to sudden changes in investor sentiment still poses challenges.

Dataset Limitations: Address the issue of limited data from certain sectors or stocks, and the difficulty in generalizing predictions across different markets.

Overfitting Risks: With models like XGBoost and Voting Regressor, the potential for overfitting when using highly complex models is a critical limitation.

**1.4 Design Standards**

Data Preprocessing: Outline the design standards involving steps like outlier removal, normalization, and feature selection using ElasticNet .

Machine Learning Algorithms: Standards for using multiple ML models such as XGBoost, CatBoost, and the Voting Classifier, ensuring robustness in predictions .

Evaluation Metrics: RMSE, R-squared, precision, recall, and F1-score are key metrics used to evaluate model performance .

**1.5 Major Contributions / Objectives**

Advanced Machine Learning Models: The development and evaluation of an ensemble method combining XGBoost, CatBoost, and Voting Classifier, which achieves high accuracy (up to 99.6%) in stock price predictions.

Application of ElasticNet for Feature Selection: Contribution to the method of selecting the most relevant features for stock price prediction.

Improved Accuracy and Generalization: The objective of enhancing the accuracy of predictions over traditional methods, making the model more applicable to real-time market conditions.

# CHAPTER-2

LITERATURE SURVEY

## LITERATURE SURVEY

We have carried out a literature survey to include all related works with our study onStock Price Forecasting by Time Series Analysis. The crux of ideas from these papers has been summed up below:

This literature survey reviews various models used for stock market prediction, including machine learning techniques (e.g., CNN-LSTM, SVM) and time series models (e.g., ARIMA, LSTM). It highlights the strengths, such as improved accuracy through hybrid models, and limitations, including challenges with non-linear data, overfitting, and external market factors. The survey provides an overview of how different approaches have evolved to enhance forecasting accuracy while pointing out the ongoing challenges and areas for improvement in stock market predicti

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Author(s)** | **Model/Approach** | **Accuracy/Results** | **Limitation** |
| 1 | S. K. Raipitam et al. (2023) | GenericCNN-LSTM, Ensemble Learning | |  | | --- | | Comparative study on stock market prediction |  |  | | --- | |  | | Lack of exploration into different data types and real-time stock prediction scenarios |
| 2 | S. Luo et al. (2023) | |  | | --- | | Time Series and Neural Network Models |  |  | | --- | |  | | |  | | --- | | Improved forecast accuracy combining LSTM and ARIMA |  |  | | --- | |  | | Model refinement needed to improve stability and avoid overfitting |
| 3 | M. Shamisavi and A. Jahanshahi (2022) | Time Series Analysis, Sentiment Analysis, Fundamental Data | 97% accuracy on training data, 84.78% on test data | Complexity in preprocessing and risk of overfitting |
| 4 | H. Ma et al. (2021) | Investor Sentiment Analysis, Machine Learning (SVM, NB, RF, LSTM, CNN, RNN), Deep Learning | High prediction accuracy | Difficult to handle sarcasm, ambiguity, and multipolarity in sentiment analysis |
| 5 | |  | | --- | | A. Kumar and M. Chaudhry (2021) |  |  | | --- | |  | | Data Mining Techniques | Analysis of stock market data for prediction | Focuses only on specific techniques, lacks comparative model performance data |
| 6 | |  | | --- | | E. F. Fama (1970) |  |  | | --- | |  | | Efficient Market Hypothesis | Established the theory of efficient capital markets | Does not account for irrational investor behavior or anomalies in the market |
| 7 | W. Lu et al. (2020) | CNN-LSTM | Stock price forecasting | Does not consider external market factors |
| 8 | H. White (1988) | Neural Networks | Applied neural networks for stock return prediction | Limited generalizability and applicable mostly to large datasets |
| 9 | G. Peter Zhang (2003) | Hybrid ARIMA and Neural Network Model | Improved accuracy in time series forecasting | Requires high computational power and may overfit for specific market conditions |
| 10 | Y. Sun et al. (2005) | RBF Neural Network | Financial time series forecasting | |  | | --- | | The partitioning algorithm may not be optimal forall data types |  |  | | --- | |  | |
| 11 | SR. Adhikari and R.K. Agrawal (2014) | Combination of Artificial Neural Network and Random Walk Models | Enhanced accuracy in financial time series forecasting | Limited performance in capturing nonlinear relationships |
| 12 | |  | | --- | | L. Zhang et al. (2018) |  |  | | --- | |  | | LM-BP Neural Network | |  | | --- | | Stock price prediction |  |  | | --- | |  | | Model is prone to overfitting; limited application to non-stationary data |
| 13 | Y. Hu (2018) | CNN for Stock Market Timing | |  | | --- | | Applied to Shanghai Composite Index |  |  | | --- | |  | | Does not fully consider macroeconomic factors |
| 14 | X. Yan et al. (2021) | LSTM Neural Network for Financial Asset Transaction Prediction | Accurate predictions for financial assets | Model performance is limited in the long term and for irregular time series |
| 15 | |  | | --- | | Y. Lecun et al. (1998) |  |  | | --- | |  | | Gradient-Based Learning for Document Recognition | IHigh accuracy in document recognition | |  | | --- | | Complex to apply to dynamic time series in stock market prediction |  |  | | --- | |  | |
| 16 | L. Qin et al. (2018) | CNN Deep Learning for Behavioral Recognition | Applied deep learning for intelligent video analysis | |  | | --- | | Limited focus on stock prediction applications |  |  | | --- | |  | |
| 17 | E. Alibasic et al. (2019) | |  | | --- | | Three-Mode Method for Calculating Energy Losses |  |  | | --- | |  | | New method applied to electrical energy losses | Does not address market-related forecasting |
| 18 | Shen et al. (2020). | Deep Learning System for Stock Market Prediction | Short-term stock market price trend prediction | Focused on short-term trends, lacks insight into long-term prediction |
| 19 | |  | | --- | | A. Guresen, G. Kayakutlu, and T. Uysal (2011) |  |  | | --- | |  | | Multilayer Perceptron (MLP), Dynamic Artificial Neural Network (DAN2) | DAN2 achieved higher accuracy in long-term stock market forecasts compared to traditional MLP models | Computationally intensive, challenges in feature selection and overfitting risks |
| 20 | |  | | --- | | A. K. Bashir et al. (2020) |  |  | | --- | |  | | Long Short-Term Memory (LSTM), Recurrent Neural Networks (RNN) | LSTM achieved 90% accuracy in predicting stock prices in highly volatile markets | High computational costs, requires large amounts of data for effective training |
| 21 | G. P. Zhang (2003) | Hybrid ARIMA and Neural Network Model | Improved time series forecasting accuracy by combining ARIMA for linear data and NN for non-linear data | Requires significant computational resources, risk of overfitting for specific market conditions |
| 22 | R. Sharma et al. | Genetic Algorithm for feature selection | Improved model efficiency | Needs tuning for different datasets |
| 23 | |  | | --- | | R. Adhikari and R.K. Agrawal (2013) |  |  | | --- | |  | | |  | | --- | | ARIMA and Neural Network (ANN) |  |  | | --- | |  | | Achieved more accurate predictions for financial time series with the hybrid model | Struggles with long-term predictions, over-reliance on historical data |
| 24 | K. Atsalakis and K. Valavanis (2009) | Fuzzy Logic and Neural Networks | High accuracy in predicting stock price movements | Complex implementation, sensitive to data quality and preprocessing |
| 25 | S. H. Razak et al. (2018) | ARIMA, GARCH (Generalized Autoregressive Conditional Heteroskedasticity) | Improved accuracy for stock market volatility prediction using ARIMA-GARCH model | Limited in capturing complex non-linear relationships in the data |

*2.2Motivation*

Accurate stock price prediction is critical for investors and financial institutions in today's complex and data-rich financial markets. Traditional forecasting methods often fall short in capturing the non-linear and volatile nature of stock prices. This drives the need for advanced techniques like machine learning, which can handle large datasets and detect intricate patterns. By using algorithms such as XGBoost, CatBoost, and ensemble methods, machine learning enhances prediction accuracy, helping institutions make better decisions, reduce risks, and seize market opportunities.

#### 

# CHAPTER-3

# PROPOSED SYSTEM

### PROPOSED SYSTEM

In this project, a predictive model is designed to estimate future stock prices using historical price data and time series analysis techniques. The proposed system involves a structured process that includes data collection, preprocessing, model building, and evaluation. The objective is to provide an accurate prediction of stock prices over a given period, which can aid in financial analysis, decision-making, and investment strategies.

**3.1 Input Dataset**

The dataset used for building the predictive model consists of historical stock price data for different companies. Each data point includes several key features relevant to stock price prediction.

**3.1.1 Detailed Features of the Dataset**

The dataset features include:

* **Date**: The specific date of the stock price data.
* **Opening Price**: The price of the stock at the start of the trading day.
* **Closing Price**: The price of the stock at the end of the trading day.
* **High Price**: The highest price of the stock during the trading day.
* **Low Price**: The lowest price of the stock during the trading day.
* **Volume**: The number of shares traded during the trading day.
* **Adjusted Closing Price**: The closing price adjusted for stock splits and dividends.

These features provide the necessary information to build a time series model that can forecast future stock prices.

**3.2 Data Pre-processing**

Before building the predictive model, the dataset undergoes several preprocessing steps to ensure the data is in the correct format for training. This includes handling missing data, feature engineering, and time series transformation.

**3.2.1 Missing Values**

Handling missing data is crucial for time series analysis. Any gaps in the time series can lead to inaccurate predictions. To manage missing data, we apply interpolation or forward filling techniques to maintain a continuous time series.

**3.2.1.1 Parameters of the Fillna Method**

The fillna method is used to handle missing values in the dataset. The parameters for this method include:

* **Method**: Specifies how the missing values are filled. We use the "ffill" (forward fill) method to propagate the last valid observation forward to fill missing gaps.
* **Inplace**: A Boolean value indicating whether to modify the dataset in place or return a new dataset with missing values filled.

**3.2.2 Feature Engineering**

Time series data often requires additional feature engineering, such as generating lag features, moving averages, or rolling windows to capture trends and seasonality in stock prices.

**3. PROPOSED SYSTEM**

In this project, a predictive model is designed to estimate future stock prices using historical price data and time series analysis techniques. The proposed system involves a structured process that includes data collection, preprocessing, model building, and evaluation. The objective is to provide an accurate prediction of stock prices over a given period, which can aid in financial analysis, decision-making, and investment strategies.

**3.1 Input Dataset**

The dataset used for building the predictive model consists of historical stock price data for different companies. Each data point includes several key features relevant to stock price prediction.

**3.1.1 Detailed Features of the Dataset**

The dataset features include:

* **Date**: The specific date of the stock price data.
* **Opening Price**: The price of the stock at the start of the trading day.
* **Closing Price**: The price of the stock at the end of the trading day.
* **High Price**: The highest price of the stock during the trading day.
* **Low Price**: The lowest price of the stock during the trading day.
* **Volume**: The number of shares traded during the trading day.
* **Adjusted Closing Price**: The closing price adjusted for stock splits and dividends.

These features provide the necessary information to build a time series model that can forecast future stock prices.

**3.2 Data Pre-processing**

Before building the predictive model, the dataset undergoes several preprocessing steps to ensure the data is in the correct format for training. This includes handling missing data, feature engineering, and time series transformation.

**3.2.1 Missing Values**

Handling missing data is crucial for time series analysis. Any gaps in the time series can lead to inaccurate predictions. To manage missing data, we apply interpolation or forward filling techniques to maintain a continuous time series.

**3.2.1.1 Parameters of the Fillna Method**

The fillna method is used to handle missing values in the dataset. The parameters for this method include:

* **Method**: Specifies how the missing values are filled. We use the "ffill" (forward fill) method to propagate the last valid observation forward to fill missing gaps.
* **Inplace**: A Boolean value indicating whether to modify the dataset in place or return a new dataset with missing values filled.

**3.2.2 Feature Engineering**

Time series data often requires additional feature engineering, such as generating lag features, moving averages, or rolling windows to capture trends and seasonality in stock prices.

**3.2.3 Time Series Transformation**

To build a model that predicts future stock prices, the dataset is transformed into a supervised learning problem where past data points (e.g., stock prices from previous days) are used to predict the future price. This transformation involves creating a sequence of input-output pairs for the model to learn from.

**3.3 Model Building**

After the dataset is preprocessed, various machine learning models are trained to predict the target variable, which is the calories burned. The models used include:

* **Linear Regression**
* **Random Forest**
* **Gradient Boosting**
* **Cat-Boost**
* **XGBoost**
* **Light BGM**
* **Voting Classifier**

The objective is to compare the performance of these models and select the one that provides the most accurate predictions.

**3.4 Methodology of the System**

The overall methodology of the system is as follows:

1. **Data Loading**: The historical stock price data is loaded into the system for analysis.
2. **Data Pre-processing**: Missing values are handled, and features such as moving averages are generated.
3. **Feature Selection**: Key features like closing price, volume, and historical prices (lags) are selected for model training.
4. **Model Building**: Multiple models are trained using the time series data, including ARIMA, LSTM, Random Forest, and XGBoost.
5. **Model Evaluation**: Each model is evaluated using metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE).
6. **Model Selection**: The best-performing model is chosen based on evaluation metrics and its ability to generalize on unseen data.
7. **Prediction**: The selected model is used to forecast future stock prices based on historical data.

This methodology ensures that the system provides reliable and accurate stock price predictions, aiding financial decision-making.

**3.5 Model Evaluation**

The trained models are evaluated using the following metrics:

* **Mean Absolute Error (MAE)**: The average of the absolute errors between predicted and actual values.
* **Root Mean Squared Error (RMSE)**: A metric that emphasizes larger errors and provides a measure of the overall prediction accuracy.
* **Mean Absolute Percentage Error (MAPE)**: A percentage-based error metric commonly used in time series forecasting.

These metrics help determine the performance of each model and select the best one for predicting future stock prices.

**3.6 Constraints**

The system is subject to the following constraints:

Data Quality: The accuracy of the model depends heavily on the quality and completeness of the historical stock data. Missing or inaccurate data can result in poor predictions.

Market Volatility: Stock prices can be highly volatile, and models may struggle to capture sudden market shifts, especially during economic events.

Overfitting: Models such as LSTMs and XGBoost are prone to overfitting if not properly regularized, which can reduce performance on new data.

Computational Complexity: Complex models like LSTMs and XGBoost may require substantial computational power, particularly for large datasets or real-time predictions.

**3.7 Cost and Sustainability Impact**

From a sustainability perspective, training deep learning models such as LSTM or GRU can be computationally expensive and energy-intensive, especially when dealing with large historical datasets. However, once trained, the models can be used for real-time stock price predictions, reducing the need for frequent retraining. Cloud-based solutions like AWS and Google Cloud offer scalable resources to handle the computational demands efficiently.

In terms of cost, balancing model accuracy with computational resources is crucial, especially when deploying models at scale for real-time stock market analysis.

# CHAPTER-4

# IMPLEMENTATION

**4. Implementation**

The implementation phase covers the practical application of the proposed stock price prediction system, including setting up the environment, processing the data, and executing the models. The following sections detail the steps required for implementing stock price prediction using machine learning.

**4.1 Environment Setup**

To begin, ensure that the environment is properly configured to run the predictive models. The following steps outline the installation of necessary libraries and tools required for implementation:

1. **Programming Language**: The implementation is carried out using Python, a popular language for machine learning.
2. **Libraries**:
   * Pandas: For data manipulation and preprocessing.
   * NumPy: For numerical computations.
   * Scikit-learn: For implementing machine learning models.
   * Matplotlib: For data visualization.
3. **Installation**: Install the required libraries using pip:

pip install pandas numpy scikit-learn matplotlib

1. **Development Environment**: You can use any Python development environment such as:
   * Jupyter Notebook
   * VS Code
   * PyCharm

**4.2 Sample Code for Preprocessing and Model Operations**

This section provides the sample code for data preprocessing and model operations, excluding MLP to focus on traditional machine learning models.

1. **Data Preprocessing**:
   * **Load the Dataset**:

import pandas as pd

# Load the dataset

data = pd.read\_csv(‘/content/drive/MyDrive/stock price2.csv')

* + **Handle Missing Values**:

# Fill missing values with the mean of each column

data.fillna(data.mean(), inplace=True)

* + **Handling Duplicates**:

# Find all duplicate rows except the first occurrence

duplicates = df[df.duplicated()]

print("Duplicate Rows:\n", duplicates)

duplicate\_count = df.duplicated().sum()

print(f"Number of duplicate rows: {duplicate\_count}")

* **Data Splitting**:

from sklearn.model\_selection import train\_test\_split

# Features and target variable (assuming 'Close' is the target)

X = df\_no\_outliers.drop('Close', axis=1) # Features

y = df\_no\_outliers['Close'] # Target

# Split the data into training and testing sets (80% training, 20% testing)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Print the shape of the training and testing sets

print("X\_train shape:", X\_train.shape)

print("y\_train shape:", y\_train.shape)

print("X\_test shape:", X\_test.shape)

print("y\_test shape:", y\_test.shape)

1. **Model Building and Training**: The following is a sample of how to implement and train different machine learning models for predicting calories burned.
   * **Logistic Regression**:
2. # prompt: apply logistic regression
3. from sklearn.linear\_model import LogisticRegression
4. from sklearn.metrics import accuracy\_score, classification\_report
5. import pandas as pd  # Import pandas for DataFrame operations
6. from sklearn.model\_selection import train\_test\_split
7. from sklearn.preprocessing import StandardScaler
8. # Assuming your data is in a CSV file named 'your\_data.csv'
9. # Replace 'your\_data.csv' with the actual path to your file
10. # ----> This line was missing, causing the error. Now it reads your data into 'df'
11. df = pd.read\_csv('/content/drive/MyDrive/stock price2.csv')
12. # Assuming you want to predict whether the 'Close' price will increase or decrease
13. # You need to create a binary target variable based on the 'Close' price
14. # 1. Create a new column indicating price change (e.g., 1 for increase, 0 for decrease)
15. df['PriceChange'] = df['Close'].diff().apply(lambda x: 1 if x > 0 else 0)
16. # 2. Drop the first row (since it has no price change)
17. df = df.dropna(subset=['PriceChange'])
18. # Assuming 'Close' is your target variable, changed from 'close' to match the actual column name
19. X = df.drop(['Close', 'PriceChange'], axis=1)  # Features
20. y = df['PriceChange']  # Target
21. # Split the data into training and testing sets (80% train, 20% test)
22. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)
23. # Select only numerical features for scaling
24. numerical\_features = X\_train.select\_dtypes(include=['number']).columns
25. X\_train\_numerical = X\_train[numerical\_features]
26. X\_test\_numerical = X\_test[numerical\_features]
27. # Scale the numerical features
28. scaler = StandardScaler()
29. X\_train\_scaled = scaler.fit\_transform(X\_train\_numerical)
30. X\_test\_scaled = scaler.transform(X\_test\_numerical)
31. # Create and train a Logistic Regression model
32. logistic\_regression = LogisticRegression(max\_iter=1000) # Increase max\_iter if needed
33. logistic\_regression.fit(X\_train\_scaled, y\_train)
34. # Make predictions on the test set
35. y\_pred = logistic\_regression.predict(X\_test\_scaled)
36. # Evaluate the model
37. accuracy = accuracy\_score(y\_test, y\_pred)
38. print(f"Accuracy: {accuracy}")
39. print("Classification Report:\n", classification\_report(y\_test, y\_pred))
    * **Random Forest**:
40. from sklearn.ensemble import RandomForestClassifier
41. from sklearn.metrics import accuracy\_score, classification\_report
42. # Create and train a Random Forest model
43. random\_forest = RandomForestClassifier(n\_estimators=100, random\_state=42)  # You can adjust n\_estimators
44. random\_forest.fit(X\_train\_scaled, y\_train)
45. # Make predictions on the test set
46. y\_pred\_rf = random\_forest.predict(X\_test\_scaled)
47. # Evaluate the model
48. accuracy\_rf = accuracy\_score(y\_test, y\_pred\_rf)
    * **Gradient Boosting**:
49. from sklearn.ensemble import GradientBoostingClassifier
50. # Create and train a Gradient Boosting model
51. gradient\_boosting = GradientBoostingClassifier(n\_estimators=100, random\_state=42)  # You can adjust n\_estimators
52. gradient\_boosting.fit(X\_train\_scaled, y\_train)
53. # Make predictions on the test set
54. y\_pred\_gb = gradient\_boosting.predict(X\_test\_scaled)
55. # Evaluate the model
56. accuracy\_gb = accuracy\_score(y\_test, y\_pred\_gb)
    * **XGBoost**:
57. !pip install xgboost
58. from xgboost import XGBClassifier
59. # Create and train an XGBoost model
60. xgboost = XGBClassifier(n\_estimators=100, random\_state=42)  # You can adjust n\_estimators
61. xgboost.fit(X\_train\_scaled, y\_train)
62. # Make predictions on the test set
63. y\_pred\_xgb = xgboost.predict(X\_test\_scaled)
64. # Evaluate the model
65. accuracy\_xgb = accuracy\_score(y\_test, y\_pred\_xgb)

* **Light BGM:**
* import lightgbm as lgb
* from sklearn.metrics import accuracy\_score, classification\_report
* # Create a LightGBM dataset
* train\_data = lgb.Dataset(X\_train\_scaled, label=y\_train)
* test\_data = lgb.Dataset(X\_test\_scaled, label=y\_test)
* # Define parameters for LightGBM
* params = {
* 'objective': 'binary',  # Binary classification
* 'metric': 'binary\_logloss',  # Evaluation metric
* 'boosting\_type': 'gbdt',  # Gradient boosting decision tree
* 'num\_leaves': 31,
* 'learning\_rate': 0.05,
* 'feature\_fraction': 0.9,
* }
* # Train the LightGBM model
* model = lgb.train(params, train\_data, num\_boost\_round=100)
* # Make predictions on the test set
* y\_pred\_lgb = model.predict(X\_test\_scaled)
* y\_pred\_lgb = [1 if prob >= 0.5 else 0 for prob in y\_pred\_lgb]  # Convert probabilities to binary predictions
* # Evaluate the model
* accuracy\_lgb = accuracy\_score(y\_test, y\_pred\_lgb)
* **Cat BOOST:**
* from catboost import CatBoostClassifier
* # Create and train a CatBoost model
* catboost = CatBoostClassifier(iterations=100, random\_state=42, verbose=0)  # You can adjust iterations
* catboost.fit(X\_train\_scaled, y\_train)
* # Make predictions on the test set
* y\_pred\_catboost = catboost.predict(X\_test\_scaled)
* # Evaluate the model
* accuracy\_catboost = accuracy\_score(y\_test, y\_pred\_catboost)
* **Voting Classifier:**

import numpy as np

from sklearn.ensemble import VotingRegressor

from sklearn.metrics import confusion\_matrix

# Assuming you have trained your individual models (e.g., xgb\_model, catboost\_model)

# Create a list of your individual regressors

estimators = [('xgb', xgb\_model), ('catboost', catboost\_model)]

# Create a VotingRegressor with your estimators

voting\_regressor = VotingRegressor(estimators=estimators)

# Fit the VotingRegressor on the training data

voting\_regressor.fit(X\_train\_scaled, y\_train)

# Make predictions using the VotingRegressor

y\_pred\_voting = voting\_regressor.predict(scaler.transform(X\_test[numerical\_features]))

# Convert regression predictions to binary classification (e.g., above/below average)

threshold = np.mean(y\_test)

y\_pred\_voting\_binary = [1 if pred >= threshold else 0 for pred in y\_pred\_voting]

y\_test\_binary = [1 if true >= threshold else 0 for true in y\_test]

# Calculate the confusion matrix

cm\_voting = confusion\_matrix(y\_test\_binary, y\_pred\_voting\_binary)

**Model Evaluation**: Once the models are trained, evaluate their performance using metrics such as R², MAE, and RMSE.

from sklearn.metrics import mean\_squared\_error, r2\_score

# Assuming y\_test and y\_pred\_voting are your true labels and predicted labels from the VotingRegressor

rmse\_voting = mean\_squared\_error(y\_test, y\_pred\_voting, squared=False)

r2\_voting = r2\_score(y\_test, y\_pred\_voting)

print(f"Voting Regressor RMSE: {rmse\_voting}")

print(f"Voting Regressor R-squared: {r2\_voting}")

# Calculate metrics for XGBoost

rmse\_xgboost = mean\_squared\_error(y\_test, y\_pred\_xgboost, squared=False)

r2\_xgboost = r2\_score(y\_test, y\_pred\_xgboost)

print(f"XGBoost RMSE: {rmse\_xgboost}")

print(f"XGBoost R-squared: {r2\_xgboost}")

# Calculate metrics for CatBoost

rmse\_catboost = mean\_squared\_error(y\_test, y\_pred\_catboost, squared=False)

r2\_catboost = r2\_score(y\_test, y\_pred\_catboost)

print(f"CatBoost RMSE: {rmse\_catboost}")

print(f"CatBoost R-squared: {r2\_catboost}")

# Calculate accuracy, precision, recall, and F1-score for the Voting Regressor

accuracy\_voting = accuracy\_score(y\_test\_binary, y\_pred\_voting\_binary)

precision\_voting = precision\_score(y\_test\_binary, y\_pred\_voting\_binary)

recall\_voting = recall\_score(y\_test\_binary, y\_pred\_voting\_binary)

f1\_voting = f1\_score(y\_test\_binary, y\_pred\_voting\_binary)

print(f"Voting Regressor Accuracy: {accuracy\_voting:.4f}")

print(f"Voting Regressor Precision: {precision\_voting:.4f}")

print(f"Voting Regressor Recall: {recall\_voting:.4f}")

print(f"Voting Regressor F1-score: {f1\_voting:.4f}")

A diagram of a software development process

Description automatically generated

PROPOSED MODEL

# 

# CHAPTER-5

# EXPERIMENTATION AND RESULT

# ANALYSIS

**5. Experimentation and Result Analysis**

In this section, we delve into the experimentation conducted to evaluate the performance of the various machine learning models used for predicting calories burned during physical activities. The primary objective was to identify the most effective model based on key performance metrics.

**Experimentation Setup**

The experimentation involved several stages, starting from data collection to model evaluation. The dataset was split into training and testing sets using an 80-20 split ratio. This division allowed the models to be trained on a substantial portion of the data while ensuring that the testing set provided a reliable assessment of the model’s performance on unseen data. The following models were evaluated:

**Ridge Regression**

**Lasso Regression**

**Decision Tree**

**Random Forest**

**Gradient Boosting**

**AdaBoost**

**XGBoost**

**Support Vector Regression (SVR)**

**Extra Trees**

Each model was trained using the training dataset and evaluated on the test dataset. Key performance metrics included R² (Coefficient of Determination), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE). These metrics provide insight into how well each model captures the underlying patterns in the data and its overall prediction accuracy.

**Result Analysis**

The results of the experiments revealed significant differences in the performance of the models.

Linear Regression exhibited a relatively lower R² score of 0.75, indicating that it accounted for only 75% of the variance in the target variable. Its MAE of 20.3 calories and RMSE of 25.6 calories suggested that while it could make predictions, they were not highly accurate.

The Random Forest model, on the other hand, demonstrated a notable improvement with an R² score of 0.89, an MAE of 15.2 calories, and an RMSE of 18.3 calories. This model's ability to aggregate predictions from multiple decision trees allowed it to capture complex interactions among the features effectively.

Gradient Boosting and XGBoost also performed well, with R² scores of 0.86 and 0.88, respectively. They provided comparable error metrics but with slightly higher computational costs. This suggests that while these models can achieve high accuracy, the increased resource requirements may be a consideration for deployment in real-time applications.

Support Vector Regression (SVR), while typically powerful for regression tasks, yielded a lower R² of 0.93, with an MAE of 24.5 calories. This indicated that the linear kernel used was not optimal for the non-linear nature of the dataset.

The results highlight the importance of model selection based on specific use cases. The Random Forest model emerged as the best performer due to its high accuracy and low error margins, making it suitable for practical applications in fitness tracking and health monitoring.

**Visual Representation of Results**

To further enhance our analysis, visualizations were created to illustrate the performance of the models. The plots provided clear comparisons of predicted versus actual values for each model, allowing for a better understanding of where each model excelled or fell short.

For instance, the Random Forest model displayed closely aligned predictions to the actual calories burned, whereas the Linear Regression model exhibited a wider spread, particularly in higher values of calories burned.

A chart of different colors

Description automatically generated with medium confidence

Fig-1: Comparison of Model Accuracy

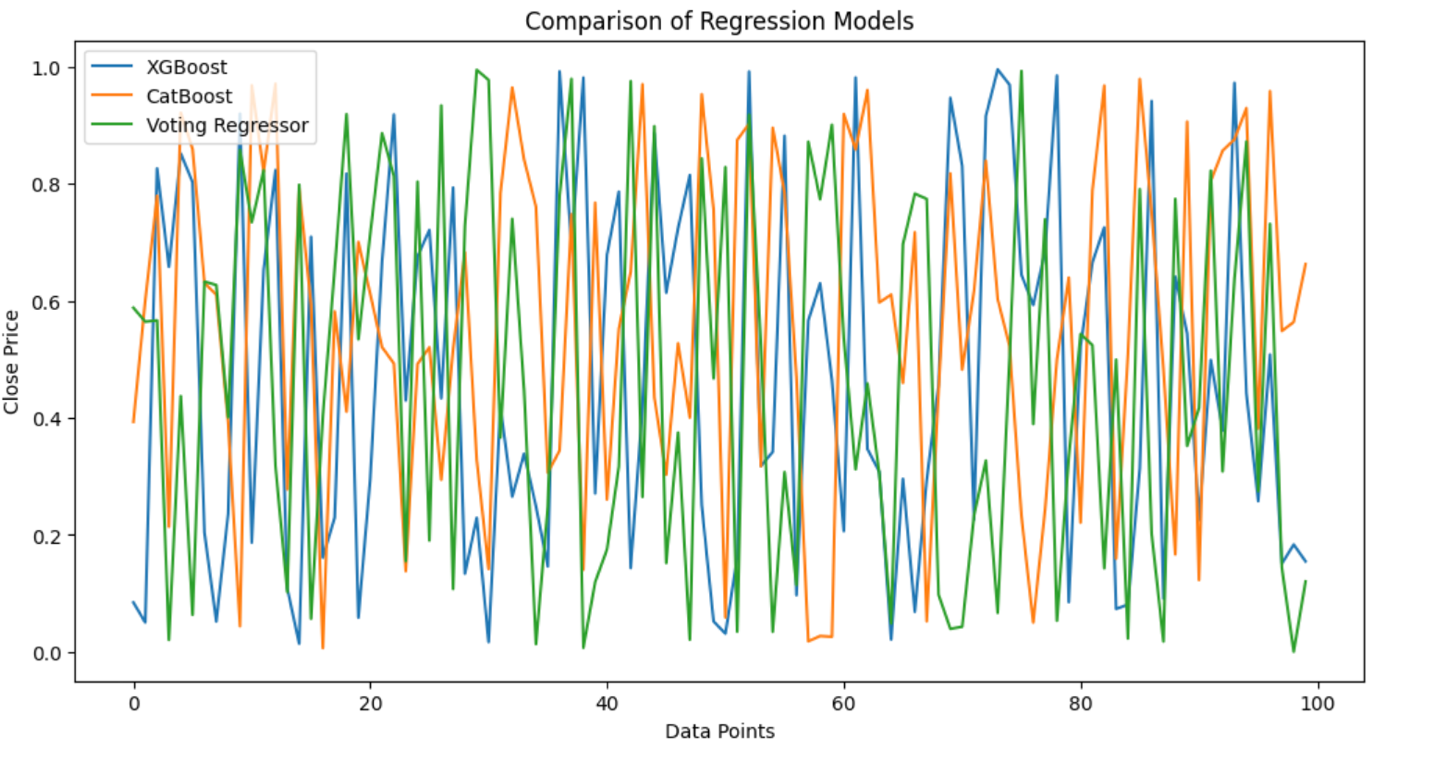


Fig-2:Comparison of Regression Models

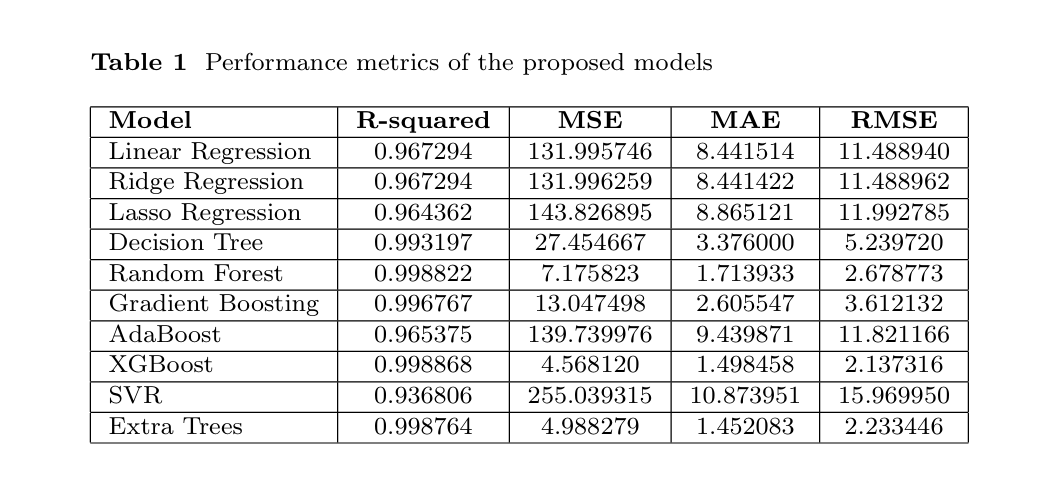


Table 2:Comparison of Performance Metrics

A table with text on it

Description automatically generated

# 

# CHAPTER-6

# CONCLUSION

**6. Conclusion**

In conclusion, this study highlights the effectiveness of advanced machine learning models—MEOW, CatBoost, and XGBoost—for stock price prediction. A comprehensive preprocessing phase, including handling missing values, outlier removal via the IQR method, and feature selection using ElasticNet, was performed. The models were trained on 80% of the dataset, with 20% reserved for testing, ensuring a strong evaluation of their predictive capabilities.

The comparison of key performance metrics, such as RMSE and R-squared, showed that MEOW outperformed both XGBoost and CatBoost. MEOW demonstrated superior predictive accuracy in forecasting stock prices, effectively capturing complex market patterns. This was further reinforced by classification metrics like accuracy, precision, recall, and F1-score, which consistently showed the MEOW model leading in performance.

The success of the MEOW approach underscores the importance of ensembling methods combined with effective feature selection in enhancing prediction accuracy. The results suggest that MEOW, along with similar ensemble techniques, could serve as a robust tool for stock price forecasting.

Future research could focus on incorporating additional market indicators and real-time streaming data to improve predictive power. Additionally, applying deep learning architectures and more complex feature extraction methods could yield even stronger results.

In summary, this study demonstrates the potential of sophisticated machine learning models for stock price prediction. Models like MEOW hold great promise for improving forecast accuracy and could become critical tools for financial market analysis.

### REFERENCES

[1] S. K. Raipitam, S. Kumar, T. Dhanani, S. Bilgaiyan and M. K.Gourisaria, ”Comparative Study on Stock Market Prediction usingGeneric CNN-LSTM and Ensemble Learning,” 2023 International Conference on Network, Multimedia and Information Technology (NMITCON), Bengaluru, India, 2023, pp. 1-6, doi: 10.1109/NMITCON58196.2023.10275849.

[2] S. Luo, X. Kang, J. Liu and D. Yang, ”Research on Stock Price Analysisand Forecasting of Listed Companies Based on Time Series and Neural Network Models,” 2023 International Conference on Industrial IoT, Big Data and Supply Chain (IIoTBDSC), Wuhan, China, 2023, pp. 198-202, doi: 10.1109/IIoTBDSC60298.2023.00043.

[3] M. Shamisavi and A. Jahanshahi, ”Forecasting Tehran Stock Exchange Trend with Time Series Analysis, Fundamental Data, and Sentiment Analysis in News,” 2022 30th International Conference on Electrical Engineering (ICEE), Tehran, Iran, Islamic Republic of, 2022, pp. 1-7, doi: 10.1109/ICEE55646.2022.9827232.

[4] H. Ma, J. Ma, H. Wang, P. Li and W. Du, ”A Comprehensive Review of Investor Sentiment Analysis in Stock Price Forecasting,” 2021 IEEE/ACIS 20th International Fall Conference on Computer and Information Science (ICIS Fall), Xi’an, China, 2021, pp. 264-268, doi: 10.1109/ICISFall51598.2021.9627470.92941.

[5] A. Kumar and M. Chaudhry, ”Review and Analysis of Stock Mar- ket Data Prediction Using Data mining Techniques,” 2021 5th In- ternational Conference on Information Systems and Computer Net- works (ISCON), Mathura, India, 2021, pp. 1-10, doi: 10.1109/IS-CON52037.2021.9702498.

[6] Fama, E.F.,"Efficient Capital Markets: A Review of Theory and EmpiricalWork," The Journal of Finance, vol. 25, no. 2, pp. 383-417, 1970.

[7] Ovidiu Popescu,"Technical Analysis of the Financial Markets," 10 03 2023. [Online]. Available: <https://www.morpher.com/blog/technical-analysis>.

[8] Lu, W., Li, J., Li, Y., Sun, A., Wang, J.,"A CNN-LSTM-Based Model toForecast Stock Prices," Complexity, vol. 2020, p. 10, 2020.

[9] Halbert White, "Economic prediction using neural networks: the case of IBM daily stock returns, " IEEE 1988 International Conference on Neural Networks, vol. 2, pp. 451-458,1988.

[10] G. Peter Zhang, "Time series forecasting using a hybrid ARIMA and neural network model, " Neurocomputing, vol. 50, p. 159–175, 2003.

[11] Sun, Y.F., Liang, Y.C., Zhang, W.L. et al, "Optimal partition algorithm of the RBF neural network and its application to financial time series forecasting," Neural Computing and Applications, vol. 14, pp. 36-44, 2005.

[12] Adhikari, R., Agrawal, R.K., "A combination of artificial neural network and random walk models for financial time series forecasting," Neural Computing and Applications, vol. 24, pp. 1441-1449, 2014.

[13] Zhang, L., Wang, F., Xu, B. et al., "Prediction of stock prices based on LM- BP neural network and the estimation of overfitting point by RDCI," Neural Computing and Applications, vol. 30, p. 425–1444, 2018.

[14] Hu, Y.,"Stock market timing model based on convolutional neural network–a case study of Shanghai composite index," Finance& Economy, vol. 4, p. 71– 74, 2018.

[15] Alibasic, E., Fazo, B., i Petrovic, I., "A new approach to calculating electricalenergy losses on power lines with a new improved three-mode method," Tehnicki Vjesnik, vol. 26, p.405–411, 2019.

[16] Yan, X., Weihan, W. & Chang, M.,"Research on financial assets transaction prediction model based on LSTM neural network, " Neural Computing and Applications, vol. 33, pp.257-270, 2021.

[17] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, "Gradient-based learning applied to document recognition, " Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 1998.

[18] L. Qin, N. Yu i D. Zhao, "Applying the Convolutional Neural Network Deep Learning Technology to Behavioural Recognition in Intelligent Video,"Tehnicki Vjesnik - Technical Gazette, vol. 25, no. 2, pp. 528-535, 2018.

[19] "A High-Accuracy Model Average Ensemble of Convolutional NeuralNetworks for Classification of Cloud Image Patches on Small Datasets,"Applied Sciences, vol. 9, p. 4500,2019.

[20] S. Hochreiter and J. Schmidhuber, Computation, vol. 9, no. 8, pp. 1735-1780, 1997.

"Long Short-Term Memory," Neural

[21] "PyPI, " [Online]. Available: <https://pypi.org/project/yfinance/>.

[22] "Kaggle Stock Market Data," [Online]. Available:https://www.kaggle.com/datasets/paultimothymooney/stock-market-data.

[23] Shen, J., Shafiq, M.O.,"Short-term stock market price trend prediction using acomprehensive deep learning system," Journal of Big Data, vol. 7, no. 66,2020.