 



**YENEPOYA INSTITUTE OF ARTS, SCIENCE AND COMMERCE**

**MANAGEMENT**

**STOCK MARKET PREDICTION USING MACHINE LEARNING**

## PROJECT SYNOPSIS

STOCK MARKET PREDICTION USING MACHINE LEARNING

## BACHELOR OF COMPUTER APPLICATION

BCA BIG DATA WITH IBM

SUBMITTED BY GUIDED BY

Mohammed Baasil Muneer – 22BDACC171 Sumit K Shukla

 

## TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **S.NO** | **TITLE** | **PAGE NO** |
| **1.** | **INTRODUCTION** | **3** |
| **2.** | **LITERATURE SURVEY** | **3** |
| **3.** | **METHODOLOGY/ PLANNING OF WORK** | **5** |
| **4.** | **FACILITIES REQUIRED FOR PROPOSED WORK** | **6** |
| **5.** | **REFERENCES** | **8** |

 

# INTRODUCTION

Breast cancer remains one of the most prevalent and life-threatening diseases worldwide, with early detection being a critical factor in improving patient outcomes and survival rates. The "Breast Cancer Prediction System Using Machine Learning" project aims to address this challenge by leveraging advanced data analytics and machine learning techniques to predict breast cancer diagnoses with high accuracy. By transforming raw medical data into actionable insights, this system provides a user- friendly platform for healthcare professionals and stakeholders to assess cancer risk, identify diagnosis patterns, and make informed decisions. Built using Flask, SQLite, and a logistic regression model trained on the `breast-cancer-wisconsin-data.csv` dataset, the project combines robust data preprocessing, secure user authentication, and an interactive web interface to deliver reliable predictions. With features like prediction history tracking, admin monitoring, and a visually appealing design with transparent containers and a consistent background (`nn.webp`), this system offers a seamless experience while prioritizing usability and precision in breast cancer prediction.

# LITERATURE SURVEY

The development of the **Stock Market Prediction System using Deep Learning** builds upon a rich body of research and advancements in financial data analytics, machine learning applications in stock forecasting, and web-based predictive systems. Below is a survey of relevant literature that informs the methodologies, technologies, and approaches adopted in this project.

### Machine Learning in Stock Market Prediction

Studies in financial forecasting have shown that machine learning algorithms can significantly enhance the accuracy of stock market predictions. Researchers have explored models such as support vector machines (SVM), random forests, and deep learning architectures to analyze historical stock data and detect complex patterns. Among these, **Long Short-Term Memory (LSTM)** networks have emerged as particularly effective for time-series forecasting due to their ability to learn temporal dependencies. This aligns with the use of an LSTM model in this project, implemented through the load\_model() function in app.py. LSTM models are well-suited for financial data, which is sequential and often influenced by past trends. Public datasets retrieved via **yfinance** offer access to standardized stock features such as open, close, high, low, and volume, which are used to train and evaluate the predictive model. These datasets serve as a benchmark for assessing the performance of stock trend prediction systems.

 

### Data Preprocessing and Feature Engineering in Stock Market Datasets

Research in financial data science emphasizes the critical role of data preprocessing and feature engineering in enhancing the performance of prediction models. Studies such as those by Patel et al. (2015) stress the need to clean and normalize stock data to handle noise, missing values, and scale inconsistencies. In this project, similar preprocessing techniques were applied using **pandas** in Python (app.py, load\_model()), where raw historical stock data retrieved via **yfinance** was cleaned and structured. Key features such as **closing price**, **volume**, and **technical indicators** like **Moving Averages (MA)** and **Relative Strength Index (RSI)** were engineered to provide meaningful inputs to the model. Feature scaling using **MinMaxScaler** or **StandardScaler** from scikit-learn ensures that the LSTM model learns effectively without bias toward features of higher magnitude, thereby improving prediction accuracy.

### Web-Based Prediction Platforms in Financial Applications

Studies such as those by Nguyen et al. (2019) have examined the role of web-based platforms in delivering real-time financial insights, highlighting the effectiveness of frameworks like **Flask** for building lightweight, interactive forecasting tools. Their research emphasized the importance of usability, responsiveness, and real-time data integration in decision-support systems. This project utilizes **Flask** to build an interactive web interface that allows users to input stock tickers, generate predictions, and visualize results through dynamic charts. While user authentication is not included in this project, the system maintains a clean, intuitive layout and ensures fast access to live data and forecasts, enhancing usability for investors and market analysts.

### Time-Series Analysis in Stock Market Forecasting Applications

Research by Fischer and Krauss (2018) has demonstrated the importance of temporal modeling in stock prediction, especially when using deep learning techniques. Their work shows how **time-based features** and **sequence modeling** contribute to uncovering hidden trends and cyclical behaviors in financial markets. This project incorporates **time-series analysis** through an **LSTM model**, which processes sequential stock price data to predict future movements. The use of historical intervals (e.g., 30-day windows) and the plotting of trends over time using **Plotly** enables users to better interpret market behavior and anticipate changes. This temporal approach supports more informed investment decisions by capturing the evolving nature of stock prices.

 

# METHODOLOGY/ PLANNING OF WORK

The Stock Market Prediction System was developed systematically, focusing on data retrieval, preprocessing, model training, web integration, and interactive interface design. Below is a concise overview of the work plan, reflecting the project’s activities (e.g., app.py, index.html).

### Data Collection and Preprocessing

* + - **Objective**: Prepare the stock data for trend prediction.
    - **Steps**: Used stock data from yfinance, cleaned and structured it using pandas, engineered features like Moving Average (MA) and RSI, and scaled them using StandardScaler (app.py, load\_model()). Time-related values were also used to display trends dynamically in charts (app.py, predict())
    - **Tools**: Python, pandas, scikit-learn.

### Model Development and Training

* + - **Objective**: Build a stock price prediction model.
    - **Steps**: Trained an **LSTM deep learning model** on processed stock data using a sequence of past prices. Serialized the model and scaler as model.h5 and scaler.pkl for reuse (app.py, get\_model\_and\_scaler()). Validated predictions by comparing actual vs. predicted values.
    - **Tools**: Python, Keras, scikit-learn, pickle.

### Database Design and Integration

* + - **Objective**: Store user stock predictions and output data.
    - **Steps**: Used a static/ folder structure to save output charts and data per stock query. Although no SQLite database is used, the app organizes prediction results by timestamp, ticker, and performance for easy access (app.py).
    - **Tools**: Python, Flask, pandas.

### Web Application Development

* + - **Objective**: Develop a user-friendly stock prediction platform.
    - **Steps**: Created Flask routes (/, /predict) to handle stock input, trigger predictions, and render dynamic charts using Plotly (app.py). Integrated Bootstrap for responsive design. While user login is not implemented, the UI ensures easy navigation and instant feedback.
    - **Tools**: Flask, Plotly, Bootstrap.

 

### User Interface Design

* + - **Objective**: Create a user-friendly interface.
    - **Steps**: Designed templates (index.html, predict.html) with Jinja2, using a shared styles.css for consistent styling—transparent containers (rgba(255, 255, 255, 0.3)), a clean background image, and responsive design (styles.css). Added interactive charts with Plotly, gradient borders, and dynamic stock prediction displays.
    - **Tools**: HTML, CSS, Jinja2, Plotly.

### Testing and Deployment

* + - **Objective**: Ensure functionality and usability.
    - **Steps**: Tested Flask routes, UI responsiveness, and data flow. Validated prediction accuracy and chart rendering locally at <http://127.0.0.1:5000> (app.py). Fixed errors such as missing data handling and template syntax issues.
    - **Tools**: Flask, browser developer tools

# FACILITIES REQUIRED FOR PROPOSED WORK

The development, testing, and deployment of the Stock Market Prediction System require a combination of hardware, software, and data resources to ensure successful implementation. Below is a concise list of the facilities utilized, based on the project’s activities (e.g., app.py, index.html, styles.css) and setup instructions.

### Hardware Requirements

* + - **Computer System**: A laptop or desktop with at least 8 GB RAM and a multi-core processor (e.g., Intel i5 or equivalent) to handle data preprocessing, deep learning model training, and Flask server hosting. Used for development on C:\Users\YourName\Desktop\stock\_prediction\.
    - **Storage** Minimum 500 MB of free disk space to store project files, stock data, model files (model.h5, scaler.pkl), and generated graphs.
    - **Internet Connection**: Stable internet for downloading dependencies (e.g., Flask, Keras, yfinance) and retrieving real-time stock data (app.py, get\_stock\_data()).

### Software Requirements

* + - **Operating System**: Windows 10/11 (used in the project setup at C:\Users\YourName\), or any OS supporting Python (e.g., macOS, Linux).

 

* + - **Python Environment** Python 3.12 with a virtual environment (venv) for dependency management. Activated via .\venv\Scripts\activate.

### Development Tools:

* + - **VS Code**: for coding, debugging, and running the Flask app (python app.py from terminal).
    - **pip** for installing dependencies like flask, keras, tensorflow, pandas, numpy, scikit-learn, yfinance, and plotly.
    - **Web Browser**: Chrome, Firefox, or Edge for testing the web interface (e.g., http://localhost:5000) and verifying UI elements such as interactive charts and backgrounds.

### Data and Libraries

* + - **Dataset**: Stock data retrieved dynamically using yfinance for various ticker symbols, used to train and validate the LSTM model (app.py, get\_stock\_data()).

### Python Libraries:

* + - pandas and numpy for data preprocessin.
    - scikit-learn for feature scaling (StandardScaler).
    - keras and tensorflow for deep learning model development and prediction.
    - flask for web app development
    - plotly for interactive visualization of stock trends and predictions
    - **Static Assets**: Background images and CSS files stored in the static/ folder for UI consistency and styling (styles.css, templates).

### Development Environment Setup

* + - **Project Directory:** Organized structure at C:\Users\YourName\Desktop\stock\_prediction\ with subfolders: templates/ (for HTML files), static/ (for CSS, images, and generated stock charts), and root files (app.py, model files, scripts).
    - **Data and Model Storage**: Stock data fetched dynamically and charts saved in the static/ folder. Model files like model.h5 and scaler.pkl are stored in the project root for loading during prediction (app.py).
    - **Local Server**: Flask development server running locally on http://127.0.0.1:5000 to test and deploy the web app (app.py).

 

### Testing and Validation Tools

* + - **Browser Developer Tools**: Used for UI testing and debugging (e.g., inspecting interactive charts, verifying transparency, background images).
    - **Terminal/Logs**: VS Code terminal to monitor Flask server logs (http://127.0.0.1:5000) and troubleshoot errors during development.
    - **Manual Testing**: Conducted to verify core functionalities like stock data retrieval, prediction accuracy, chart rendering, and UI consistency across pages (index.html, predict.html, etc.).

# REFERENCES

### Academic Papers

Hochreiter, S., & Schmidhuber, J. (1997). "Long Short-Term Memory." Neural Computation, 9(8), 1735-1780.  
Relevance: Justifies the use of LSTM for sequential stock data modeling (app.py, model training.

Kim, H. Y., & Kim, S. Y. (2019). "Stock Market Prediction Using LSTM Networks." Expert Systems with Applications, 135, 112-121.  
Relevance: Supports LSTM effectiveness for stock trend prediction (app.py, load\_model()).

* Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). "Predicting Stock Market Index Using Fusion of Machine Learning Techniques." Expert Systems with Applications, 42(4), 2162-2172.  
  Relevance: Highlights technical indicators for improved prediction accuracy (MA, RSI).
* Sarker, I. H. (2021). "Machine Learning: Algorithms, Real-World Applications and Research Directions." SN Computer Science, 2, 160.  
  Relevance: Guides model selection and data preprocessing (app.py)).

Grinberg, M. (2018). "Flask Web Development." O'Reilly Media.  
Relevance: Provides framework for building the web app (app.py, routes)).

### Documentation and Resources

 Flask Documentation. <https://flask.palletsprojects.com/en/3.0.x/>  
Relevance: Flask framework guide (app.py).

 scikit-learn Documentation. <https://scikit-learn.org/stable/>  
Relevance: Model training and preprocessing (app.py).

 Plotly Documentation. https://plotly.com/python/  
Relevance: Interactive data visualization (templates/, static/).

 yfinance Documentation. <https://pypi.org/project/yfinance/>  
Relevance: Stock data retrieval (app.py)

css).