Mini Project Report

StockX: Deep Learning-Based Time Series Analysis for Stock Market Trend Prediction

DIVISION OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF ENGINEERING

COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Meenakshi Pramod (20222067)

Sebin Thomas Pallivathukkal (20222093)

APRIL 2025

Mini Project Report

on

StockX: Deep Learning-Based Time Series Analysis for Stock Market Trend Prediction

Submitted by

Meenakshi Pramod (20222067)

Sebin Thomas Pallivathukkal (20222093)

In partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering.

DIVISION OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF ENGINEERING

COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY

APRIL 2025

DIVISION OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF ENGINEERING
COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY

CERTIFICATE

Certified that this is a Mini Project Report titled

StockX: Deep Learning-Based Time Series Analysis for Stock Market Trend Prediction

Submitted by

Meenakshi Pramod (20222067)
Sebin Thomas Pallivathukkal (20222093)

of VI Semester, Computer Science and Engineering in the year 2025 in partial fulfillment of the requirements for the award of Degree of Bachelor of Technology in Computer Science and Engineering of Cochin University of Science and Technology.

Dr. Pramod Pavithran	Dr. Pramod Pavithran	Dr. M Sudheep Elayidom
Head of Division	Project Coordinator	Project Guide

Acknowledgement

We extend our heartfelt gratitude to the Almighty, the ultimate source of knowledge and wisdom.

We would like to express our sincere thanks to our esteemed guide, Dr. M Sudheep Elayidom, for his continuous support, guidance, and motivation throughout the development of this project. His valuable suggestions and feedback have helped us shape this project in the best possible manner.

We extend our heartfelt appreciation to Dr. Pramod Pavithran, Head of the Division of Computer Science and Engineering and Project Coordinator, for his invaluable support, guidance, and encouragement throughout all stages of our project. We are also grateful to him for providing us with the opportunity to undertake this project and for the facilities made available to us.

Finally, we thank all our teachers, friends, and family members who helped us directly or indirectly during the course of this project.

Meenakshi Pramod (20222067)

Sebin Thomas Pallivathukkal (20222093)

Declaration

We, Meenakshi Pramod and Sebin Thomas, hereby declare that the mini project entitled "StockX: Deep Learning-Based Time Series Analysis for Stock Market Trend Prediction" is the record of original work carried out by us during the academic year 2025, and has not been submitted elsewhere for any other academic award or degree.

Abstract

The financial industry plays a pivotal role in driving global economic growth, with the stock market gaining increasing popularity due to its rising economic significance. Understanding stock price fluctuations and market volatility requires insights into complex variables, including corporate earnings, government regulations, shareholder dynamics, and market sentiment. According to the efficient market hypothesis, a market is considered efficient when it is transparent, competitive, and reflects all relevant factors in stock prices. However, the intricate interplay of variables that affect stock prices makes prediction a challenging task.

Traditional stock market forecasting models have struggled to adapt to the dynamic and volatile nature of financial markets. Machine Learning (ML) and Deep Learning (DL) approaches have emerged as powerful tools for addressing these challenges, offering the ability to analyze large volumes of data, uncover patterns, and provide actionable insights. Deep Learning, a subset of ML, is particularly well-suited for financial applications due to its ability to automatically learn features across multiple layers, generalize effectively, and adapt to new data.

This project aims to predict stock price trends using advanced deep learning architectures such as LSTM, BiLSTM, GRU, and CNN-BiLSTM. Implemented as an interactive web application using the Streamlit framework, it provides users with intuitive controls to select stock tickers, choose models, and visualize predictions along with evaluation metrics and confidence scores.

By leveraging the strengths of various Deep Learning techniques, this project seeks to address the challenges of ambiguity and volatility in stock market prediction, enhance accuracy, and provide an innovative framework for AI-driven financial analytics. The proposed approach has the potential to significantly improve trading strategies, portfolio management, and decision-making for investors and analysts.

Contents

List of Figures

Chapter 1

Introduction

The financial industry is vital to global economic growth, with stock markets playing a key role due to their economic impact. Predicting stock prices is challenging due to factors like corporate earnings, regulations, investor sentiment, and macroeconomic events. Traditional forecasting models often struggle with market volatility, but Machine Learning (ML) and Deep Learning (DL) offer improved solutions by identifying complex patterns in historical data.

DL models like LSTM, BiLSTM, GRU, and CNN-LSTM excel in time series forecasting by capturing temporal dependencies. Enhanced with technical indicators (e.g., Moving Averages, RSI), these models improve prediction accuracy. This project develops a Stock Price Prediction Dashboard using Streamlit, enabling users to select stocks, configure models, and visualize results with performance metrics (MAE, RMSE, R-squared).

By integrating technical analysis and deep learning, this system assists traders, analysts, and researchers in making data-driven decisions, advancing AI-powered financial forecasting.

Chapter 2

Literature Review

Forecasting stock market trends has long been a complex and extensively studied area in finance. Over the years, many approaches—ranging from statistical models to modern AI techniques—have been employed to improve prediction accuracy. With the recent advancement of Large Language Models (LLMs) and time series-focused deep learning models, new frameworks are being proposed that go beyond numerical data, incorporating textual and behavioral analysis as well.

2.1 Overview

Traditional statistical methods like ARIMA, GARCH, and moving average models, while widely used, often struggle with non-linearity and noise in financial data. Machine Learning (ML) methods such as Random Forests, SVM, and Gradient Boosted Trees marked a significant leap forward but still require careful feature engineering.

In contrast, Deep Learning (DL) models—especially LSTM, GRU, BiLSTM, and CNN-LSTM—can learn temporal dependencies and hidden patterns from raw time series data. These models offer improved forecasting capabilities by capturing non-linearities and adapting to dynamic market trends.

2.2 Review and Analysis

Literature Review

- **Predicting Financial Enterprise Stocks (Haotian Zheng et al., 2024):** This study explores the application of machine learning in financial time series analysis. It highlights the effectiveness of LSTM and CNN-BiLSTM hybrid models in capturing nonlinear patterns in stock markets, achieving significant predictive accuracy and robustness through empirical validation.[1]

- **A Time Series Analysis-Based Framework (Singh and Malhotra, 2024):** This paper presents a comprehensive stock price prediction framework that employs artificial intelligence techniques. The study focuses on long-term trend analysis and integrates modern ML models with economic data for improved forecasting.[4]

- **Global Stock Market Prediction (Mala K et al., 2024):** This research proposes a hybrid deep learning architecture using stock chart images as input to CNN and LSTM models. It enhances prediction accuracy using Deep Q-Learning and demonstrates superior performance across diverse global markets.[3]

- **Deep Learning for Time Series Prediction in IIoT (IEEE TNNLS, 2024):** This paper discusses the application of deep learning models such as Transformer and LSTM in time-series forecasting. The insights are extended to finance, where accurate modeling of time-series stock data is crucial.[2]

- **Financial Time Series Prediction (Agampreet Saini et al., 2024):** This work investigates the performance of deep learning and machine learning models on Apple's stock prediction. The study compares various architectures and highlights the significance of hybrid approaches.[5]

These works illustrate the growing importance of combining deep learning models and language models to tackle the challenges of stock price prediction.

Chapter 3

System Analysis

3.1 Proposed System

The proposed system aims to predict stock price trends using advanced deep learning architectures such as LSTM, BiLSTM, GRU, and CNN-BiLSTM. Implemented as an interactive web application using the Streamlit framework, it provides users with intuitive controls to select stock tickers, choose models, and visualize predictions along with evaluation metrics and confidence scores.

The system offers:

- Real-time data fetching using the Yahoo Finance API.
- Integration of technical indicators such as Moving Averages, RSI, MACD, Bollinger Bands, and Volume.
- Configurable deep learning models for trend prediction.
- Visualizations of actual vs predicted stock prices with prediction intervals.
- 5-day future forecasting with tabular and graphical comparison.

3.2 System Features

a. User Interaction

Users input a stock ticker and date range, select desired models and configuration parameters such as sequence length and epochs.

b. Technical Analysis

The system computes multiple technical indicators on the fetched data, which are fed into the models to improve forecasting accuracy.

c. Model Configuration

Users can choose from four different architectures — LSTM, BiLSTM, GRU, and CNN-BiLSTM — to observe and compare their performance.

d. Metrics Dashboard

The platform displays detailed evaluation metrics: MAE, RMSE, R^2 score, MAPE, and a custom confidence score.

e. Visualization

Predicted vs actual trends are plotted with Plotly, showing prediction intervals. Future 5-day forecasts are also visualized.

f. Model Comparison

If multiple models are selected, users can view a comparison table and a 5-day prediction comparison across models.

3.3 Advantages of the System

- **Interactive Dashboard:** Simple and dynamic UI for non-technical users to operate.
- **Multiple Deep Learning Models:** Users can evaluate performance across various architectures.
- **Confidence Score:** An innovative metric to indicate reliability of predictions.
- **Real-Time Financial Data:** Live data fetching using Yahoo Finance API ensures up-to-date predictions.
- **Modular Architecture:** Easy to extend or replace components, like adding new models or indicators.

Chapter 4

System Study

4.1 Software Requirements Specification (SRS)

The Software Requirements Specification (SRS) outlines the functionality, performance, and constraints of the proposed Stock Price Prediction Dashboard. It provides a comprehensive understanding of the system's intended behavior, ensuring alignment with user expectations and project objectives.

4.1.1 Purpose

The main objective of this project is to develop an intelligent and interactive web-based application that predicts stock price trends using deep learning architectures. By integrating real-time stock data with time series forecasting models, the system assists investors, researchers, and traders in making informed decisions. The dashboard allows users to select models, customize parameters, and visualize performance through metrics and future predictions.

4.1.2 Product Scope

This system focuses on building a web application that:

- Fetches real-time stock data using Yahoo Finance.

- Calculates the technical indicators.
- Trains and evaluates multiple deep learning models (LSTM, BiLSTM, GRU, CNN-BiLSTM).
- Displays model performance using evaluation metrics and prediction intervals.
- Provides a 5-day future price forecast.
- Offers comparative analysis between selected models.

4.1.3 System Description

The application is designed using Python and Streamlit for the frontend. Deep learning models are built using TensorFlow/Keras and trained on normalized time series data. The user inputs configurations like the stock ticker, date range, model types, sequence length, and number of epochs. The system processes the data, trains the models, evaluates their performance, and displays both graphical and tabular results.

4.1.4 System Architecture

The system follows a modular architecture, with each component responsible for specific tasks:

- **Data Acquisition Module:** Fetches historical stock data from Yahoo Finance.
- **Preprocessing Module:** Cleans and scales the data, adds technical indicators, and generates sequences.
- **Modeling Module:** Builds and trains the selected deep learning models.
- **Evaluation Module:** Computes performance metrics and a custom confidence score.
- **Forecasting Module:** Generates future price predictions for the next 5 days.
- **Visualization Module:** Uses Plotly for interactive plots and Streamlit widgets for user interaction.

4.1.5 Functional Requirements

- **Data Input:** User provides stock ticker, start and end dates.
- **Model Selection:** User can select one or more models (LSTM, BiLSTM, GRU, CNN-BiLSTM).
- **Configuration:** Sequence length, train/test split, and number of epochs can be adjusted.
- **Indicator Selection:** Users can enable/disable technical indicators like MA, RSI, MACD, etc.
- **Data Visualization:** Display historical stock data with technical indicators.
- **Model Training:** Selected models are trained on the input data.
- **Metrics Display:** Show MAE, RMSE, R^2, MAPE, and confidence score.
- **Forecasting:** Display 5-day forecasted stock prices for selected models.
- **Model Comparison:** Tabular comparison of models across all metrics and forecasts.

4.1.6 Non-Functional Requirements

- **Performance:** Models should be trained within a reasonable time (1 min) for standard datasets.
- **Reliability:** Predictions and metrics should remain consistent across runs with the same input.
- **Scalability:** Framework supports adding more models or indicators in the future.
- **Security:** Only publicly available financial data is used, so no personal data security is required.
- **Usability:** Streamlit ensures an intuitive, interactive UI even for non-programmers.
- **Portability:** Can be deployed on any system with Python and internet access.

4.1.7 User Classes and Characteristics

- **General Users:** Anyone interested in forecasting stock prices. No programming knowledge needed.
- **Traders and Analysts:** Users who wish to visualize technical indicators and evaluate models for portfolio insights.
- **Researchers and Students:** Those experimenting with time series models and financial prediction.

4.2 Hardware and Software Requirements

4.2.1 Hardware Requirements

- A system with at least 8 GB RAM and multi-core CPU
- Internet connection for accessing Yahoo Finance API

4.2.2 Software Requirements

- Python 3.10+
- Required libraries: `streamlit`, `yfinance`, `pandas`, `numpy`, `scikit-learn`, `tensorflow`, `plotly`, etc.

4.3 Platforms and Tools

4.3.1 Platform

- **Streamlit:** Streamlit is an open-source Python library used to build interactive dashboards and data applications with ease. It simplifies the front-end development process and allows for rapid prototyping.

- **Jupyter Notebook:** Used during initial model development and experimentation. It supports real-time code execution and visualization, making it suitable for machine learning workflows.
- **Local System (Windows/Linux):** The project was developed and tested on a personal computer running either Windows or Linux operating system with necessary Python dependencies installed.

4.3.2 Tools

- **Python:** The core programming language used throughout the project due to its extensive libraries for data science, finance, and machine learning.
- **yFinance:** A Python package used to fetch real-time and historical stock data from Yahoo Finance.
- **TensorFlow and Keras:** These libraries were used to design, train, and evaluate deep learning models such as LSTM, GRU, and CNN-BiLSTM.
- **Scikit-learn:** Utilized for preprocessing and model evaluation. It includes features like MinMaxScaler and functions to calculate MAE, RMSE, and R^2.
- **Pandas:** Used for data manipulation and analysis, especially handling time-series stock data.
- **NumPy:** Facilitates efficient numerical computations, particularly for matrix operations.
- **Matplotlib and Plotly:** Employed for data visualization. Matplotlib was used for static plots, while Plotly supported interactive financial charts.
- **VS Code:** Visual Studio Code served as the integrated development environment (IDE) for writing and organizing the source code.

Chapter 5

System Design

5.1 Introduction

System design focuses on structuring the internal components and data flow of the application to meet the specified requirements. It bridges the gap between system analysis and actual development, ensuring clarity, efficiency, and maintainability. In this project, the system is divided into several modules for data fetching, preprocessing, model training, evaluation, and visualization. These modules are designed to work independently and integrate smoothly through clearly defined interfaces.

5.2 Data Flow Diagrams (DFD)

5.2.1 DFD Level 0 - Context Diagram

The Level 0 DFD provides a bird's eye view of the system. It shows how a user interacts with the application to obtain stock predictions.

- **User:** Inputs stock ticker, dates, and configuration.
- **System:** Returns prediction results and performance metrics.

Figure 5.1: DFD Level 0 – Overview of Stock Prediction System

5.2.2 DFD Level 1 – Major Components

This level explains core processes:

- **Fetch Stock Data**
- **Add Technical Indicators**
- **Preprocess Data**
- **Train Deep Learning Models**
- **Evaluate and Forecast**
- **Display Results to User**

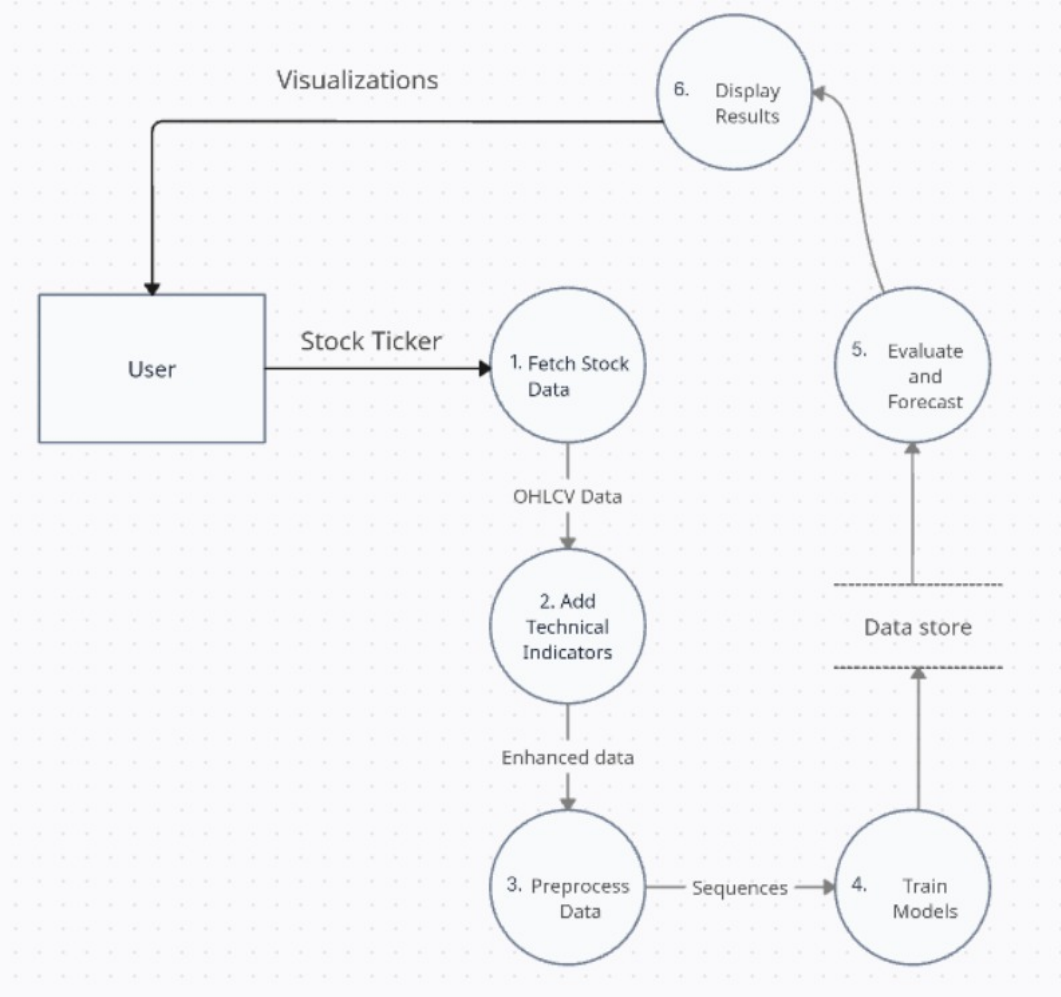

Figure 5.2: DFD Level 1 – Core System Processes

5.3 Use Case Diagram

5.3.1 Main Use Case – Forecasting System

- Users enter the stock ticker and configuration parameters.
- System processes data and trains the models.
- Predictions and metrics are displayed to the user.

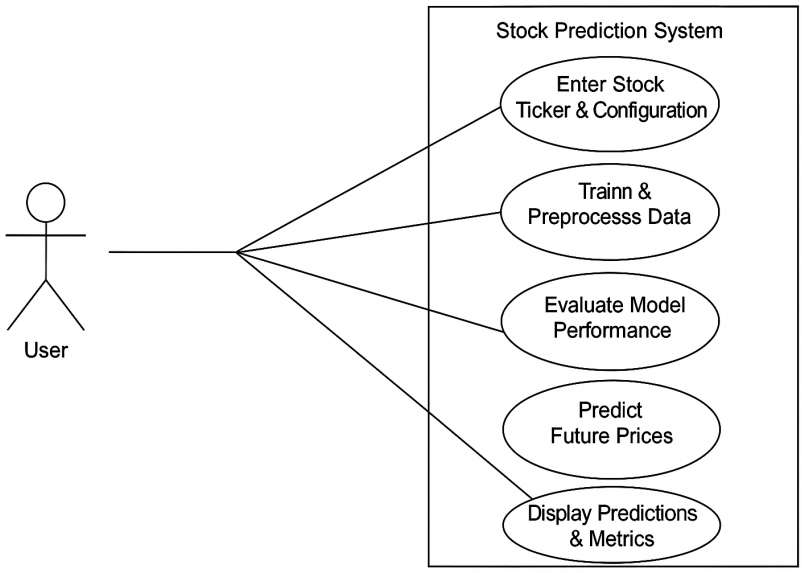

Figure 5.3: Use Case Diagram – Core Features

5.4 Deep Learning Model Architectures

LSTM

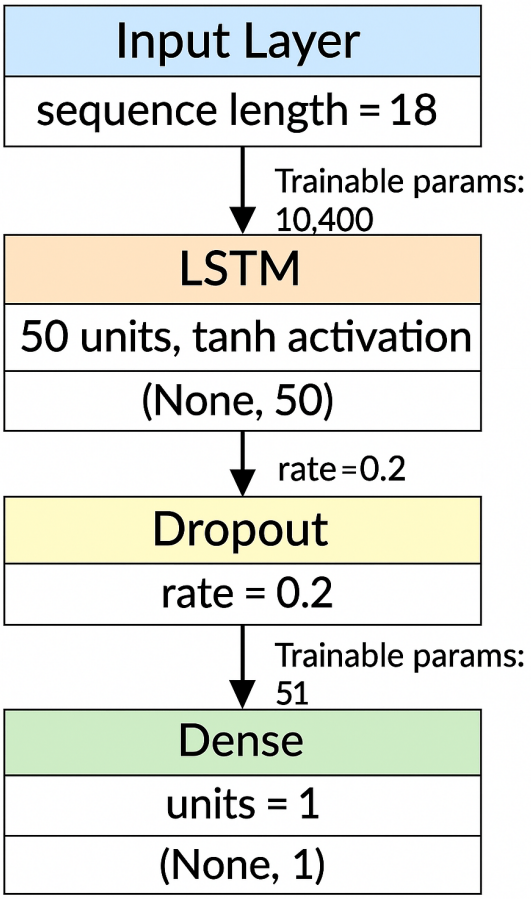

Figure 5.4: LSTM Architecture

- 50 LSTM units
- Dropout layer (0.2)
- Dense output layer (1 neuron)

BiLSTM

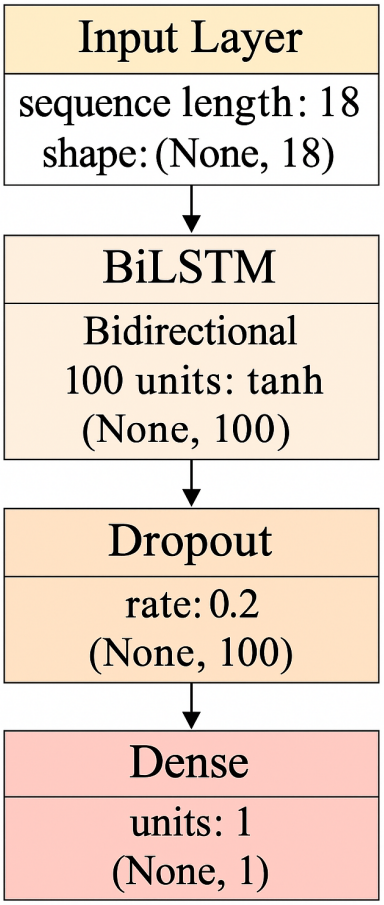

Figure 5.5: BiLSTM Architecture

- 50 LSTM with bidirectional wrappers
- Dropout layer (0.2)
- Dense output layer (1 neuron)

GRU

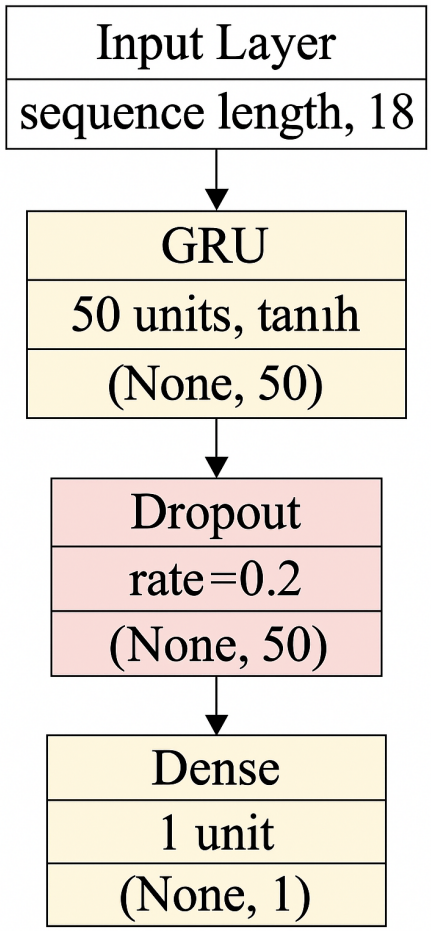

Figure 5.6: GRU Architecture

- 50 GRU units
- Dropout layer
- Dense layer for regression output

CNN-BiLSTM

- 1D Convolutional Layer
- Bidirectional LSTM
- Dropout and Dense Layer

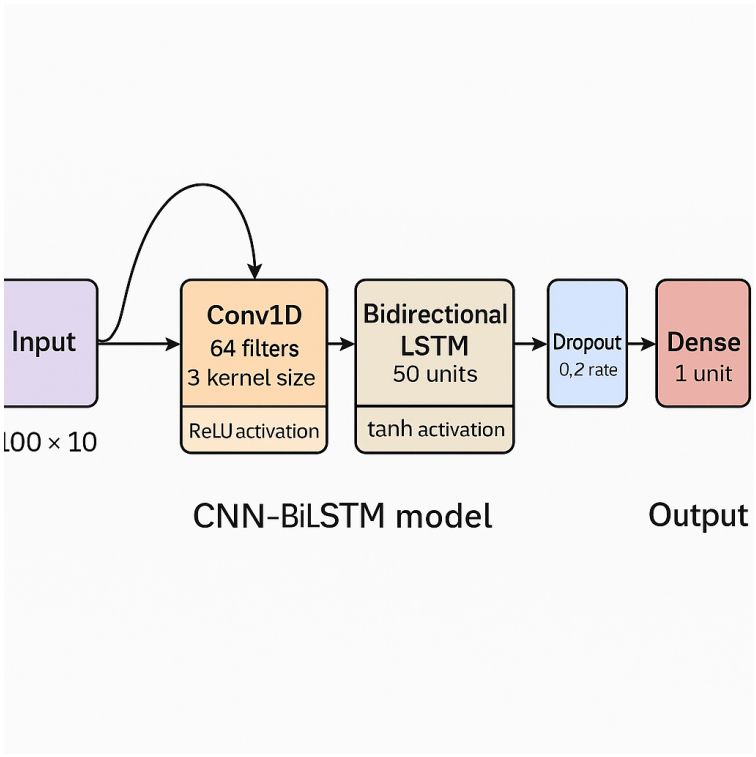

Figure 5.7: CNN-BiLSTM Architecture

5.5 Modular Design

This section presents a detailed breakdown of the system into modular components. Each module is responsible for a specific functionality and works in cohesion with other modules to deliver the final prediction and visualization output.

5.5.1 Data Acquisition Module

This module fetches historical stock price data from Yahoo Finance using the `yfinance` API. It retrieves time-series data including Open, High, Low, Close, and Volume values based on user input (stock ticker and date range).

Key Functions:

- Downloading raw OHLCV data.
- Structuring the data in pandas DataFrame for further processing.

5.5.2 Technical Indicator Module

This module adds technical indicators to the stock price data to enhance the feature set for training.

Indicators Used:

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

$$\text{SMA}_n = \frac{1}{n} \sum_{i=0}^{n-1} P_{t-i}$$

  Where P_{t-i} is the closing price on day $t - i$, and n is the window size.

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

$$RSI = 100 - \frac{100}{1 + \frac{\text{Avg Gain}}{\text{Avg Loss}}}$$

  Where Avg Gain and Avg Loss are average gains and losses over the last 14 days respectively.

- **Bollinger Bands:**

$$\text{Upper Band} = MA_{20} + 2\sigma, \quad \text{Lower Band} = MA_{20} - 2\sigma$$

  Where MA_{20} is the 20-day moving average and σ is the standard deviation of closing prices over the same period.

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

$$\text{MACD} = EMA_{12} - EMA_{26}, \quad \text{Signal Line} = EMA_9(\text{MACD})$$

  Where EMA is the Exponential Moving Average over the specified number of days.

- **Volume Indicators:** 10-day moving average of trading volume.

5.5.3 Preprocessing Module

This module prepares the dataset for modeling by performing the following steps:

- Handling missing values and dropping NaNs from rolling indicators.
- Scaling features between 0 and 1 using MinMaxScaler.
- Creating sequences of input features and target outputs for supervised learning.

Sequence Definition:

$$X_t = [x_{t-n}, x_{t-n+1}, \ldots, x_{t-1}], \quad y_t = x_t$$

Where X_t is the input sequence of length n, and y_t is the predicted closing price at time t.

5.5.4 Model Module

This module constructs and trains deep learning models using the Keras API. The models are designed for time-series forecasting:

- **LSTM:** Long Short-Term Memory, a type of RNN capable of learning long-term dependencies.
- **BiLSTM:** Bidirectional LSTM for learning from both past and future context.
- **GRU:** Gated Recurrent Unit, a simpler alternative to LSTM with comparable performance.
- **CNN-BiLSTM:** Uses a 1D Convolutional layer followed by BiLSTM for feature extraction and sequence learning.

All models use a dense output layer with one neuron to predict the closing price. Training is done using the Adam optimizer with MAE (Mean Absolute Error) as the loss function.

5.5.5 Evaluation Module

After training, each model is evaluated using the following metrics:

- **Mean Absolute Error (MAE):**

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

  Where y_i is the actual price and $\hat{y}_i$ is the predicted price.

- **Root Mean Squared Error (RMSE):**

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

  RMSE penalizes larger errors more than MAE.

- **R-squared (R^2):**

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

  Where $\bar{y}$ is the mean of actual values.

- **Mean Absolute Percentage Error (MAPE):**

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

  Gives a percentage-based error measure.

- **Confidence Score:**

$$\text{Confidence} = 100 \cdot \left(0.4 \cdot \left(1 - \frac{\text{MAE}}{\bar{y}}\right) + 0.6 \cdot \text{Coverage}\right)$$

  Where $\bar{y}$ is the mean of the true values and Coverage is the percentage of actual values within the prediction interval. This custom metric combines both accuracy and uncertainty.

5.5.6 Visualization Module

This module provides an intuitive and interactive interface using Plotly and Streamlit to:

- Plot actual vs. predicted stock prices.
- Display 95% prediction intervals using confidence bands.
- Show a 5-day forecast with percentage change.
- Compare performance of multiple models side-by-side.

5.6 Input and Output Design

Inputs

- Stock Ticker (e.g., AAPL)
- Date Range (Start and End Date)
- Model Selection (Checkbox)
- Sequence Length, Split Ratio, Epochs (Sliders)
- Technical Indicators (Checkboxes)

Outputs

- Raw Stock Data Table
- Technical Indicator Visuals
- Model Performance Metrics (MAE, RMSE, R^2, etc.)
- Plotly Graphs with Prediction Intervals
- Forecasted Prices Table
- Comparison Table Across Models

Chapter 6

System Implementation

The StockX system is a stock price prediction platform that enables users to forecast future stock values based on historical market data. The application integrates technical analysis with deep learning to enhance prediction accuracy. It fetches stock data in real-time from Yahoo Finance and allows users to choose various technical indicators such as Moving Averages (MA), Relative Strength Index (RSI), Bollinger Bands, MACD, and Volume-based metrics. These indicators help in capturing short and long-term market trends and are added as new features to the input dataset.

The core of the system is powered by deep learning models including LSTM, BiLSTM, GRU, and CNN-BiLSTM, each of which is tailored to capture temporal dependencies in time-series data. Among these, the CNN-BiLSTM model combines the local feature extraction capabilities of convolutional layers with the long-term sequence learning ability of bidirectional LSTMs. The models are trained on preprocessed sequences of normalized stock prices, with adjustable hyperparameters like sequence length, number of epochs, and train/test split ratio. The output is a 5-day forecast along with key evaluation metrics.

Model performance is measured using standard regression metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), R^2 score, and Mean Absolute Percentage Error (MAPE). Additionally, a customized Confidence Score is computed using a combination of prediction accuracy and prediction interval coverage, offering a clearer indication of model reliability to the user.

The entire system is built using Python and Streamlit. While TensorFlow and Keras

handle the model training and predictions, Streamlit provides an intuitive dashboard that allows users to interact with the system in real-time. Users can compare predictions across multiple models, visualize prediction intervals, and explore short-term trends.

The following section will explain the implementation details in code.

6.1 Sample Code

```python
import streamlit as st
import yfinance as yf
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout, Bidirectional, GRU, Conv1D
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
import matplotlib.pyplot as plt
import plotly.graph_objects as go
from datetime import datetime, timedelta

# Set page config
st.set_page_config(
    page_title="Stock Price Prediction",
    page_icon="",
    layout="wide"
)

# Title and description
st.title("Stock Price Prediction Dashboard")
st.markdown("""
This dashboard allows you to predict stock prices using various deep learning models.
Select a stock ticker, choose your preferred models, and adjust parameters to get started.
""")


```

Figure 6.1: Dashboard Setup Code

Streamlit dashboard setup with user inputs for stock ticker, date range, model selection, and technical indicators.

```python
# Sidebar for user inputs
st.sidebar.header("Model Configuration")

# Stock ticker input
ticker = st.sidebar.text_input("Enter Stock Ticker (e.g., AAPL)", "AAPL")

# Date range selection
col1, col2 = st.sidebar.columns(2)
with col1:
    start_date = st.date_input("Start Date", datetime.now() - timedelta(days=365))
with col2:
    end_date = st.date_input("End Date", datetime.now())

# Model selection
st.sidebar.subheader("Select Models")
```


File - E:\JupyterProject\mini_project.py

```python
models = {
    "LSTM": st.sidebar.checkbox("LSTM", value=True),
    "BiLSTM": st.sidebar.checkbox("BiLSTM", value=True),
    "GRU": st.sidebar.checkbox("GRU", value=True),
    "CNN-BiLSTM": st.sidebar.checkbox("CNN-BiLSTM", value=True)
}

# Model parameters
st.sidebar.subheader("Model Parameters")
sequence_length = st.sidebar.slider("Sequence Length", 30, 120, 60)
split_ratio = st.sidebar.slider("Train/Test Split Ratio", 0.6, 0.9, 0.8, 0.05)
epochs = st.sidebar.slider("Number of Epochs", 10, 100, 20)

# Fixed future prediction period (5 days)
future_days = 5

# Technical indicators selection
st.sidebar.subheader("Technical Indicators")
indicators = {
    "Moving Averages": st.sidebar.checkbox("Moving Averages", value=True),
    "RSI": st.sidebar.checkbox("RSI", value=True),
    "Bollinger Bands": st.sidebar.checkbox("Bollinger Bands", value=True),
    "MACD": st.sidebar.checkbox("MACD", value=True),
    "Volume Indicators": st.sidebar.checkbox("Volume Indicators", value=True)
}
```

Figure 6.2: Sidebar Configuration Code

Code for sidebar configuration allowing users to select models, adjust hyperparameters, and enable technical indicators.

```python
def fetch_data(ticker, start_date, end_date):
    """Fetch stock data from Yahoo Finance"""
    data = yf.download(ticker, start=start_date, end=end_date)
    return data


def add_technical_indicators(df):
    """Add technical indicators to the dataframe"""
    # Ensure multi-level columns are flattened
    if isinstance(df.columns, pd.MultiIndex):
        df.columns = df.columns.droplevel(1)

    if indicators["Moving Averages"]:
        df['MA_7'] = df['Close'].rolling(window=7).mean()
        df['MA_21'] = df['Close'].rolling(window=21).mean()
        df['MA_50'] = df['Close'].rolling(window=50).mean()
        df['MA_200'] = df['Close'].rolling(window=200).mean()

    if indicators["RSI"]:
        delta = df['Close'].diff()
        gain = (delta.where(delta > 0, 0)).rolling(window=14).mean()
        loss = (-delta.where(delta < 0, 0)).rolling(window=14).mean()
        rs = gain / loss
        df['RSI'] = 100 - (100 / (1 + rs))

    if indicators["Bollinger Bands"]:
        df['Bollinger_Mid'] = df['Close'].rolling(window=20).mean()
        df['Bollinger_Upper'] = df['Bollinger_Mid'] + 2 * df['Close'].rolling(window=20
).std()
        df['Bollinger_Lower'] = df['Bollinger_Mid'] - 2 * df['Close'].rolling(window=20
).std()

    if indicators["MACD"]:
        df['MACD'] = df['Close'].ewm(span=12, adjust=False).mean() - df['Close'].ewm(
span=26, adjust=False).mean()
        df['Signal_Line'] = df['MACD'].ewm(span=9, adjust=False).mean()

    if indicators["Volume Indicators"]:
        df['Volume_MA'] = df['Volume'].rolling(window=10).mean()

```


File - E:\JupyterProject\mini_project.py

```python
    df['Daily_Return'] = df['Close'].pct_change()
    df['Volatility'] = df['Daily_Return'].rolling(window=20).std()
    df['High_Low_Spread'] = (df['High'] - df['Low']) / df['Close'] * 100
    df['Open_Close_Spread'] = (df['Close'] - df['Open']) / df['Open'] * 100

    df.dropna(inplace=True)
    return df

```

Figure 6.3: Data Fetching and Indicators

Data fetching from Yahoo Finance and technical indicator calculation (e.g., Moving Averages, RSI) based on user selections.

```python
def preprocess_data(df, sequence_length, split_ratio):
    """Preprocess data for model training"""
    split_index = int(len(df) * split_ratio)
    train_df = df.iloc[:split_index]
    test_df = df.iloc[split_index:]

    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_train = scaler.fit_transform(train_df)
    scaled_test = scaler.transform(test_df)

    def create_sequences(data):
        X, y = [], []
        for i in range(sequence_length, len(data)):
            X.append(data[i - sequence_length:i])
            y.append(data[i, df.columns.get_loc('Close')])
        return np.array(X), np.array(y)

    X_train, y_train = create_sequences(scaled_train)
    X_test, y_test = create_sequences(scaled_test)

    return X_train, X_test, y_train, y_test, scaler, train_df, test_df

```

Figure 6.4: Data Preprocessing

Normalization, sequence generation, and train-test split for model training.

```python
def build_lstm_model(input_shape):
    """Build LSTM model"""
    model = Sequential([
        LSTM(50, activation='tanh', input_shape=input_shape),
        Dropout(0.2),
        Dense(1)
    ])
    model.compile(optimizer='adam', loss='mae')
    return model


def build_bilstm_model(input_shape):
    """Build BiLSTM model"""
    model = Sequential([
        Bidirectional(LSTM(50, activation='tanh'), input_shape=input_shape),
        Dropout(0.2),
        Dense(1)
    ])
    model.compile(optimizer='adam', loss='mae')
    return model


def build_gru_model(input_shape):
    """Build GRU model"""
    model = Sequential([
        GRU(50, activation='tanh', input_shape=input_shape),
        Dropout(0.2),
        Dense(1)
    ])
    model.compile(optimizer='adam', loss='mae')
    return model


def build_cnn_bilstm_model(input_shape):
    """Build CNN-BiLSTM model"""
```

Page 3 of 8

File - E:\JupyterProject\mini_project.py

```python
    model = Sequential([
        Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=input_shape),
        Bidirectional(LSTM(50, activation='tanh')),
        Dropout(0.2),
        Dense(1)
    ])
    model.compile(optimizer='adam', loss='mae')
    return model
```

Figure 6.5: Model Architectures

Implementation of LSTM, BiLSTM, GRU, and CNN-BiLSTM architectures for time-series prediction.

```python
def evaluate_model(model, X_test, y_test, scaler, df):
    """Evaluate model performance with improved confidence scoring"""
    y_pred_scaled = model.predict(X_test).flatten()

    dummy_array = np.zeros((len(y_pred_scaled), X_test.shape[2]))
    dummy_array[:, df.columns.get_loc('Close')] = y_pred_scaled
    y_pred_actual = scaler.inverse_transform(dummy_array)[:, df.columns.get_loc('Close')]

    dummy_array[:, df.columns.get_loc('Close')] = y_test
    y_test_actual = scaler.inverse_transform(dummy_array)[:, df.columns.get_loc('Close')]

    mae = mean_absolute_error(y_test_actual, y_pred_actual)
    rmse = np.sqrt(mean_squared_error(y_test_actual, y_pred_actual))
    r2 = r2_score(y_test_actual, y_pred_actual)
    mape = np.mean(np.abs((y_test_actual - y_pred_actual) / y_test_actual)) * 100

    # Calculate improved confidence score
    confidence_score = calculate_confidence_score(y_test_actual, y_pred_actual)

    return mae, rmse, r2, mape, confidence_score, y_test_actual, y_pred_actual


def predict_future(model, last_sequence, scaler, df, days=5):
    """Predict future stock prices"""
    future_predictions = []
    current_sequence = last_sequence.copy()

    for _ in range(days):
        # Reshape for prediction
        current_sequence_reshaped = current_sequence.reshape(1, current_sequence.shape[0], current_sequence.shape[1])
        # Predict next value
        next_pred_scaled = model.predict(current_sequence_reshaped).flatten()[0]

        # Create dummy row to inverse transform
        dummy_array = np.zeros((1, current_sequence.shape[1]))
        dummy_array[0, df.columns.get_loc('Close')] = next_pred_scaled
        next_pred = scaler.inverse_transform(dummy_array)[0, df.columns.get_loc('Close')]

        future_predictions.append(next_pred)

        # Create new row for the prediction
        new_row = current_sequence[-1].copy()
        # Update the close price with our prediction
        new_row[df.columns.get_loc('Close')] = next_pred_scaled

        # Remove first row and add new prediction at the end
        current_sequence = np.vstack((current_sequence[1:], new_row))

    return future_predictions

```

Figure 6.6: Evaluation and Prediction

Model evaluation (MAE, RMSE, R^2) and 5-day future price prediction using autoregressive inference.

```
    """Plot actual vs predicted values with prediction intervals"""
    # Calculate prediction intervals
    residuals = y_test_actual - y_pred_actual
    std_residuals = np.std(residuals)
    upper_bound = y_pred_actual + 1.96 * std_residuals
    lower_bound = y_pred_actual - 1.96 * std_residuals

    fig = go.Figure()

    # Add prediction interval
    fig.add_trace(go.Scatter(
        x=df.index[-len(y_pred_actual):],
        y=upper_bound,
        fill=None,
        mode='lines',
        line_color='rgba(255,165,0,0.3)',
        name='Upper Bound',
        showlegend=False
    ))

    fig.add_trace(go.Scatter(
        x=df.index[-len(y_pred_actual):],
        y=lower_bound,
        fill='tonexty',
        mode='lines',
        line_color='rgba(255,165,0,0.3)',
        name='95% Prediction Interval'
    ))

    # Add actual values
    fig.add_trace(go.Scatter(
        x=df.index[-len(y_test_actual):],
        y=y_test_actual,
        name='Actual',
        line=dict(color='blue')
    ))

    # Add predicted values
    fig.add_trace(go.Scatter(
        x=df.index[-len(y_pred_actual):],
        y=y_pred_actual,
        name='Predicted',
        line=dict(color='red')
    ))

    # Add future predictions if available
    if len(future_predictions) > 0:
        fig.add_trace(go.Scatter(
            x=future_dates,
            y=future_predictions,
            name='5-Day Forecast',
            line=dict(color='green', dash='dash')
        ))

    fig.update_layout(
        title=f'{model_name} Predictions with Confidence Intervals',
        xaxis_title='Date',
        yaxis_title='Price',
        template='plotly_dark',
        xaxis=dict(range=[df.index[-30], future_dates[-1]] if len(future_predictions) >
0
        else [df.index[-30], df.index[-1]])
    )

    return fig
```

Figure 6.7: Results Visualization

Interactive Plotly visualization of actual vs. predicted prices with confidence intervals.

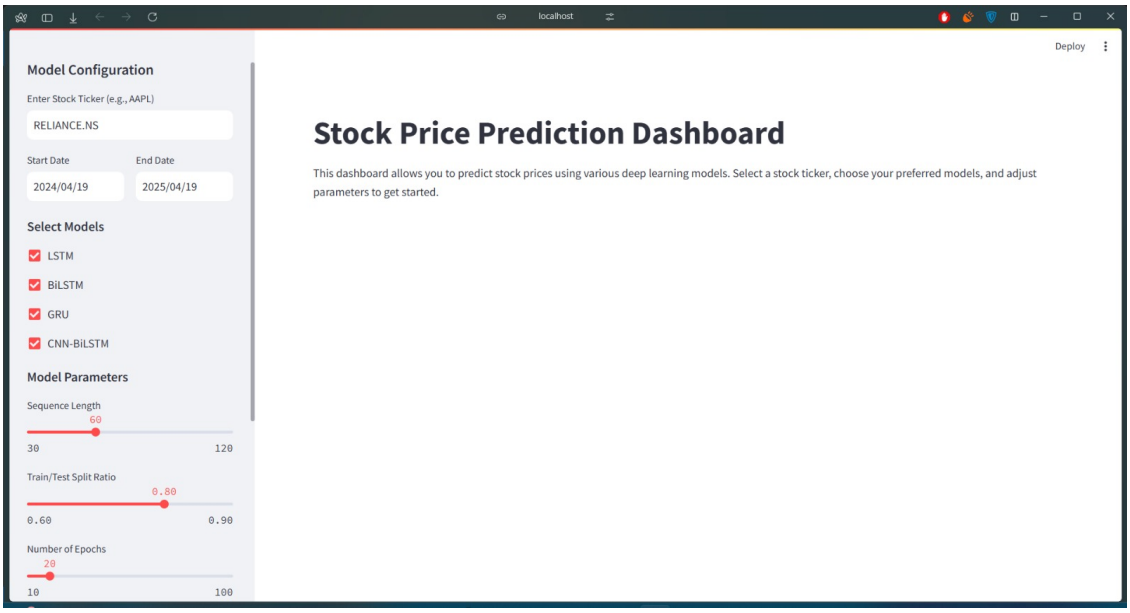

Figure 6.8: Dashboard

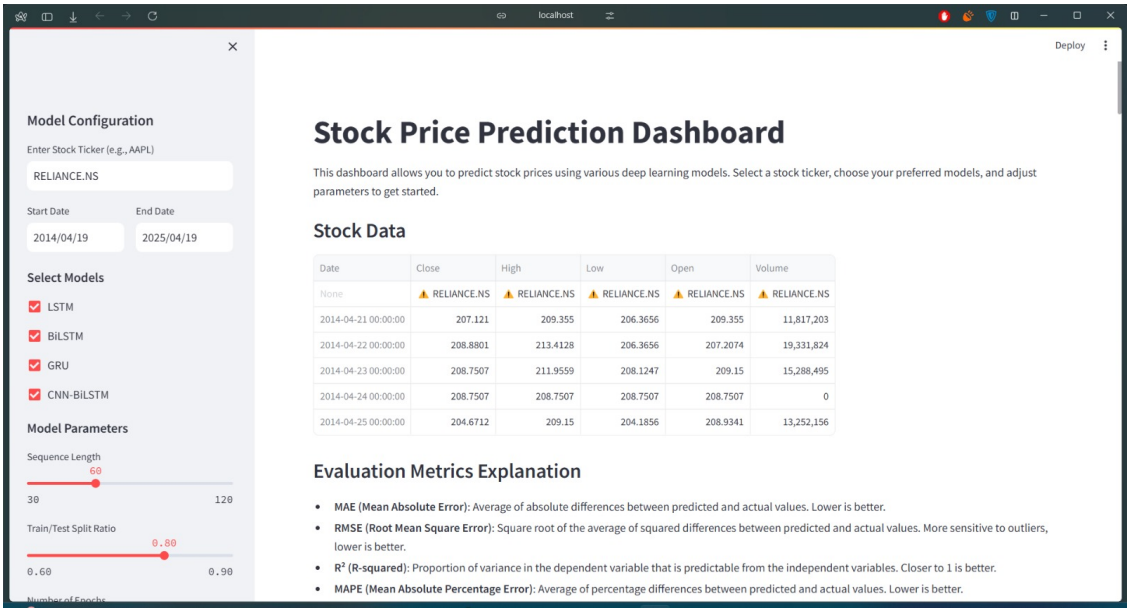

Figure 6.9: Stock Data

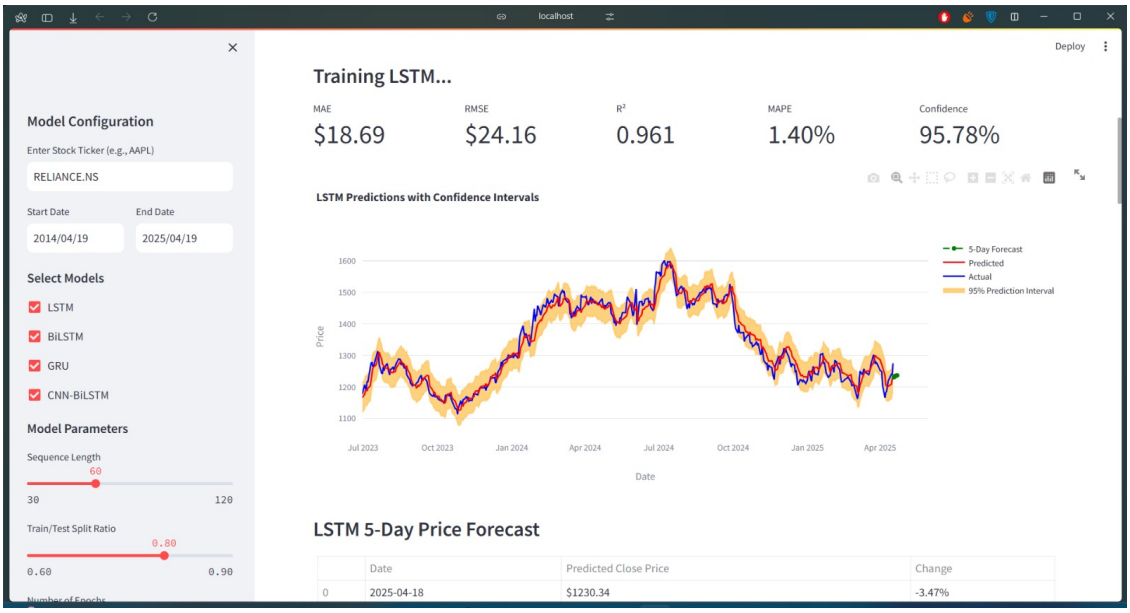

Figure 6.10: Plot Actual vs Predicted values

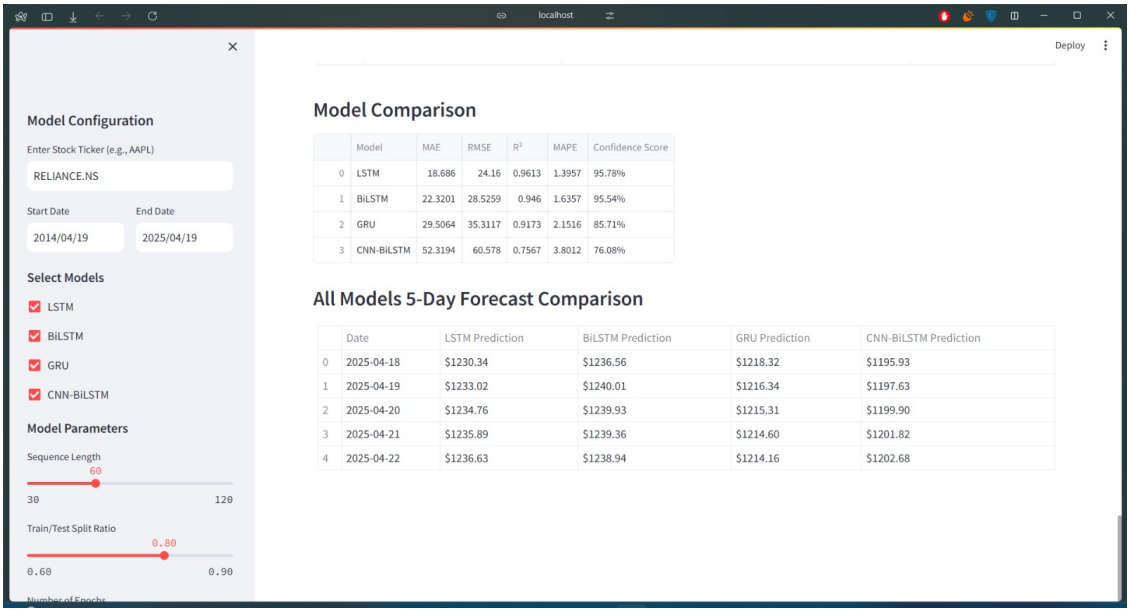

Figure 6.11: Model Comparison and Forecasting

Chapter 7

Conclusion

The project **StockX** aimed to develop a robust and user-friendly stock price prediction dashboard using advanced deep learning techniques and real-time market data. By integrating models like LSTM, BiLSTM, GRU, and CNN-BiLSTM, the system effectively captured both short-term fluctuations and long-term trends in stock price behavior.

The project successfully demonstrated the potential of combining deep learning architectures with technical analysis for financial forecasting. The results validate that hybrid models, particularly CNN-BiLSTM, can offer improved predictive accuracy for short-term stock price forecasting. Moreover, the modular and scalable design of StockX ensures that it can be extended to support other financial instruments, longer forecast horizons, and integration with external news or sentiment data in future versions.

Overall, this work serves as a practical implementation of machine learning in finance and provides a solid foundation for further research and development in stock market prediction systems.

Chapter 8

Future Enhancements

While the StockX system provides an effective and flexible framework for stock price prediction, several avenues remain for future development and enhancement:

- **Integration of News and Sentiment Analysis:** Incorporating real-time financial news and social media sentiment (using Natural Language Processing techniques) could significantly improve prediction accuracy by capturing market-moving events not reflected in historical price data alone.
- **Model Optimization and Hyperparameter Tuning:** Future work could explore automated hyperparameter optimization techniques such as Grid Search or Bayesian Optimization to improve the performance of the deep learning models further.
- **Support for Multi-stock Portfolio Prediction:** Extending the system to handle and forecast multiple stocks simultaneously would enable users to analyze entire portfolios and optimize investment decisions accordingly.
- **Mobile-Friendly Interface and Alerts:** A mobile-responsive version of the dashboard with features like push notifications for significant stock movements or prediction updates could make the platform more interactive and user-friendly.
- **Inclusion of Advanced Architectures:** Implementation of newer architectures such as Transformer-based models (e.g., Temporal Fusion Transformers, Informer) could offer state-of-the-art performance in long-sequence time series forecasting.

Bibliography

[1] Haotian Zheng,Jiang Wu,Runze Song,Lingfeng Guo,Zeqiu Xu, *Predicting Financial Enterprise Stocks and Economic Data Trends Using Machine Learning Time Series Analysis*, 2024. Available at: `https://doi:10.20944/preprints202407.0895.v1`

[2] *Deep Learning for Time-Series Prediction in IIoT: Progress, Challenges, and Prospects*, IEEE Transactions on Neural Networks and Learning Systems, Volume 35, Issue 11, November 2024. Available at: `https://doi.org/10.1109/TNNLS.2023.3291371`

[3] Mala K , Harish G N , Pradeep M , Harshith T C, *Global Stock Market Prediction Using Stock Chart Images with Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) Networks Enhanced by Deep Q-Learning*, 2024.

[4] Harmanjeet Singh, Manisha Malhotra, *A Time Series Analysis-Based Stock Price Prediction Framework Using Artificial Intelligence*, 2024. Available at: `http://dx.doi.org/10.1007/978-3-031-48781-1_22`

[5] Agampreet Saini,Nisha Singh, Rahul Kumar Singh, Manoj Kumar Sachan,*Financial Time Series Prediction on Apple Stocks using Machine and Deep Learning Models.*