<STOCK TREND PREDICITION>

Submitted for

Statistical Machine Learning CSET211

Submitted by:

(<E23CSEU1182>) GITIKSHA MALIK

(<E23CSEU1186>) DAKSHI VASHISHTH

(<E23CSEU1200>) MANISH MAHESHWARI

Submitted to

DR. NITIN ARVIND SHELKE

July-Dec 2024

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

INDEX

1. Abstract

This project focuses on predicting stock trends using stock market data and machine learning techniques. A user can input a stock ticker to retrieve data from 2010 to 2019 in a tabular format, accompanied by visualizations such as closing price vs. time charts, moving averages (100 days), and predictions vs. actual values. The application is built using Streamlit, providing an intuitive interface for data visualization and trend forecasting. The project aims to offer insights into stock behavior, enabling better investment decisions.

2. Introduction

Stock market prediction has always been a challenging and intriguing domain due to its complex and dynamic nature. Leveraging AI and machine learning, this project explores how stock data can be used to forecast future trends. This Streamlit-based application allows users to visualize historical data, calculate moving averages, and predict stock prices with minimal effort. The project demonstrates how data-driven methods can enhance decision-making in financial markets while emphasizing the importance of accessible, user-friendly tools.

The core objective is to simplify the exploration of stock price patterns while introducing predictive capabilities that highlight future trends. Leveraging Python's rich ecosystem of libraries, this project demonstrates how advanced computational techniques can be democratized for broader audiences.

3. Related Work (If Any)

Several studies and projects have attempted to predict stock market trends using AI and machine learning. Traditional approaches, such as time-series analysis and technical indicators, have evolved with the inclusion of models like Linear Regression, ARIMA, LSTM, and GRU. For instance:

- Research on using Recurrent Neural Networks (RNN) for time-series predictions has shown promising results.
- Time-Series Models: ARIMA and Exponential Smoothing.
- Neural Networks: Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), which excel at learning temporal dependencies.
- Tools like Yahoo Finance and libraries such as yfinance and pandas have enabled seamless data retrieval and preprocessing.
- Similar projects have been developed using libraries like TensorFlow and PyTorch, providing baseline methodologies for comparison.

4. Methodology

The project workflow involves the following steps:

1. **Data Collection:**
    - Stock data (2010–2019) is retrieved using pandas_datareader.
    - Data includes Open, High, Low, Close, Volume, and Adjusted Close values.
2. **Preprocessing:**
    - Data cleaning and handling of missing values.
    - Calculation of technical indicators like moving averages (100 days).
3. **Visualization:**
    - Using matplotlib.pyplot, the following plots are created:
        - Closing Price vs. Time Chart.
        - Closing Price with 100-day Moving Average Chart.
        - Closing Price with 100-day and 200-day Moving Average Chart.
        - Predictions vs Actual
4. **Prediction Model:**
    - Use of machine learning techniques (e.g., Linear Regression or LSTM) for forecasting future stock prices.
    - Train-test split to evaluate model performance.
5. **Streamlit Integration:**
    - Building an interactive interface to input stock tickers, view data tables, visualizations, and predictions.

5. Hardware/Software Required

Software:

- Python 3.x

Libraries:

- numpy: For numerical computations.
- pandas: For data manipulation and analysis.
- matplotlib.pyplot: For creating visualizations.
- pandas_datareader: For retrieving financial data.
- Streamlit: For web application development**.**

6. Experimental Results

Data Representation:

The dataset (2010–2019) is displayed in tabular format, showing descriptive statistics such as mean, standard deviation, and range for each attribute.

Visualizations:

- Closing Price vs. Time: A simple line chart created using matplotlib.pyplot to show trends over time.
- Closing Price with Moving Average (100 days): Created using matplotlib.pyplot to highlight smoothed trends, reducing short-term fluctuations.
- Predictions vs. Actual Values: Evaluates model performance using the prediction model outputs.

Evaluation Metrics:

- Mean Squared Error (MSE) for regression-based models.
- Accuracy or R^2 score for predictive performance.

7. Conclusion

The project demonstrates the feasibility of using historical stock data for trend analysis and prediction. The integration of machine learning models with interactive

visualizations through Streamlit enables efficient and user-friendly exploration of stock trends. While the current model provides a basic understanding of stock movements, further improvements in model sophistication can lead to more accurate predictions.

8. Future Scope

- Enhanced Models:
  - Implementation of advanced models like Long Short-Term Memory (LSTM) networks for better time-series forecasting.
  - Use of ensemble techniques for improved accuracy.
- Real-Time Data:
  - Integration of live stock market data for real-time predictions and analysis.
- Feature Expansion:
  - Incorporation of external factors like economic indicators, news sentiment analysis, and global market trends.
- Scalability:
  - Extending the application to support multiple markets and indices globally.
- Deployment:
  - Hosting the application on platforms like AWS or Heroku for public accessibility.

9. GitHub Link of Your Complete Project

https://github.com/PandaXD7/Stocks-Predi.git

CODE SNIPPET :

Closing price v/s Time chart with 100MA -

```
st.subheader('Closings Price vs Time chart')
fig = plt.figure(figsize=(12, 6))
plt.plot(df['Close'])
st.pyplot(fig)

st.subheader('Closing Price vs Time chart with 100MA')
ma100 = df['Close'].rolling(100).mean()  # Add parentheses to call the mean() function
fig = plt.figure(figsize=(12, 6))
plt.plot(ma100, label='100MA')  # Optional: add label for clarity
plt.plot(df['Close'], label='Closing Price')
plt.legend()
st.pyplot(fig)
```

Closing price v/s Time chart with 100MA & 200MA -

```
st.subheader('Closing Price vs Time chart with 100MA & 200MA')
ma100 = df['Close'].rolling(100).mean()  # Line 38: Make sure parentheses are used with .mean()
ma200 = df['Close'].rolling(200).mean()  # Line 39

fig = plt.figure(figsize=(12, 6))
plt.plot(ma100, 'r', label='100MA')  # Label added for clarity
plt.plot(ma200, 'g', label='200MA')  # Label added for clarity
plt.plot(df['Close'], 'b', label='Closing Price')  # Label added for clarity
plt.legend()
st.pyplot(fig)
```