**STOCK PRICE PREDICTOR APP - SENTIMENT ANALYSIS PROJECT REPORT**

**Problem Statement:** The stock market is highly volatile, and predicting stock prices is a complex challenge due to numerous influencing factors such as economic conditions, company performance, investor sentiment, and global events. The need for an accurate and reliable stock price prediction model has led to the integration of machine learning and deep learning techniques to enhance decision-making for investors. This project aims to develop a stock price predictor application using historical stock data and deep learning models to provide insights into future stock price trends.

**Aim of the Project:** The primary objective of this project is to build a stock price prediction application using a deep learning model. The application is designed to:

1. Fetch real-time stock price data from Yahoo Finance.
2. Process and analyze historical stock data.
3. Utilize a trained deep learning model to predict future stock prices.
4. Visualize stock trends using moving averages and comparison graphs.
5. Assist investors in making data-driven investment decisions.

**Techniques Used in the Project:**

1. **Data Collection**: The application fetches stock market data using the Yahoo Finance API, ensuring up-to-date information is available for analysis.
2. **Data Preprocessing**:
   * Handling missing values and ensuring data consistency.
   * Normalizing stock prices using MinMaxScaler to scale values between 0 and 1.
3. **Deep Learning Model**:
   * A pre-trained TensorFlow-based deep learning model is used for stock price predictions.
   * The model processes stock closing prices over a specified time window (100 previous days) to predict future values.
4. **Visualization**:
   * Moving Averages (100, 200, 250-day) are plotted to identify trends in stock price movements.
   * A comparison graph is used to visualize the original stock prices and predicted values.
5. **Error Handling and Optimization**:
   * Implemented error handling to address API rate limits and missing data scenarios.
   * Optimized model loading and preprocessing for efficient real-time predictions.

**Explanation of the Code:**

1. **Importing Libraries**:
   * The code imports essential Python libraries such as Streamlit, Pandas, NumPy, Matplotlib, TensorFlow, and Yahoo Finance to handle data fetching, processing, visualization, and deep learning-based predictions.
2. **User Input and Stock Data Fetching**:
   * The user inputs a stock symbol (e.g., 'RELIANCE.NS'), which is then used to fetch historical stock price data from Yahoo Finance using the yfinance.download() function.
   * A retry mechanism is implemented to handle API rate limits and ensure successful data retrieval.
3. **Preprocessing and Data Preparation**:
   * The application checks if the fetched stock data contains the 'Close' column, which is essential for predictions.
   * Data is split into training and testing sets (70% training, 30% testing).
   * MinMaxScaler is used to normalize the closing price values before feeding them into the model.
4. **Deep Learning Model Loading and Prediction**:
   * A pre-trained TensorFlow model is loaded from the local directory.
   * The last 100 days of stock prices are used as input to predict future stock prices.
5. **Visualization of Data and Predictions**:
   * Moving Averages (100, 200, and 250 days) are calculated and plotted.
   * A graph is generated to compare actual vs. predicted stock prices.
   * The results are displayed in an interactive Streamlit app.
6. **Error Handling and Debugging**:
   * The code includes multiple error-handling mechanisms, such as checking if the model file exists before loading, handling empty data responses, and implementing retry strategies in case of API rate limits.

**Use of Python in the Project:** Python plays a crucial role in this project, enabling data collection, preprocessing, model integration, and visualization. The specific use cases include:

1. **Data Fetching**: The yfinance library is used to retrieve historical stock data from Yahoo Finance.
2. **Data Processing**: Pandas and NumPy are used for data manipulation, handling missing values, and structuring the dataset for model input.
3. **Machine Learning & Deep Learning**: The TensorFlow and Keras libraries are employed to load the pre-trained model and make predictions on stock prices.
4. **Scaling Data**: MinMaxScaler from Scikit-learn is used to normalize the data before feeding it into the model.
5. **Graphical Visualization**: The Matplotlib library is used to create various plots such as moving averages and stock price comparisons.
6. **Web Application**: Streamlit is utilized to create an interactive web-based interface for users to enter stock symbols and view predictions dynamically.
7. **Error Handling & Performance Optimization**: Python’s exception handling is implemented to manage API rate limits and missing data scenarios, ensuring smooth execution.

**Conclusion:** The Stock Price Predictor App successfully implements deep learning techniques to predict future stock prices using historical data. The integration of real-time data fetching, machine learning-based price prediction, and visualization techniques enhances the application's usability for investors and analysts. While the model provides valuable insights, stock price prediction remains inherently uncertain due to market fluctuations and unforeseen global events. Future improvements can include incorporating sentiment analysis from financial news and social media data to enhance prediction accuracy further.