Team 1 GitHub Documentation

1. Abstract

- The project focuses on building predictive modeling for stock price forecasting and investment recommendations using three powerful machine learning and statistical models: **LSTM (Long Short-Term Memory)**, **ARIMA (AutoRegressive Integrated Moving Average)**, and **XGBoost (Extreme Gradient Boosting)**. These models are implemented within a streamlined framework that leverages the `yfinance` **library** to fetch real-time stock market data, eliminating the need for a static dataset.
- The primary objective of this implementation is to enable accurate short-term and medium-term stock price predictions, visualize historical and forecasted trends, and provide actionable insights for investors. Each model has been optimized for its specific strength: LSTM for handling sequential dependencies in time-series data, ARIMA for trend and seasonality analysis, and XGBoost for feature-driven regression-based predictions.

2. Features Implemented

1. **Dynamic Data Retrieval with `yfinance`:**
   - **Description:** Fetches real-time stock data for any company based on user inputs (company name, start date, and end date).
   - **Significance:** Eliminates dependency on static datasets, ensuring predictions are based on the latest market data, improving accuracy and relevancy.
2. **LSTM Model for Sequential Analysis:**
   - **Description:** Implements a Long Short-Term Memory (LSTM) model to capture time-series patterns in stock price data.
   - **Significance:** Handles sequential dependencies effectively, providing accurate predictions for stock price trends over different time ranges.
3. **ARIMA Model for Statistical Forecasting:**
   - **Description:** Uses the AutoRegressive Integrated Moving Average (ARIMA) model to forecast stock prices by analyzing trends, lags, and seasonality.
   - **Significance:** Well-suited for data with clear patterns or periodicity, offering reliable medium-term predictions.
4. **XGBoost Model for Feature-Based Predictions:**
   - **Description:** Applies XGBoost regression to predict stock prices using feature transformations derived from stock data.
   - **Significance:** Enhances prediction accuracy by leveraging the strength of gradient boosting and its ability to handle large-scale, non-linear data.
5. **Visualization of Historical and Predicted Prices:**
   - **Description:** Provides visual representations of stock trends, overlaying predictions from all three models on historical data.
   - **Significance:** Helps users easily compare model predictions with historical trends for better interpretability.
6. **Customizable Prediction Inputs:**
   - **Description:** Users can input the company name, start and end dates for historical data, and choose prediction ranges (short-term, medium-term, etc.).

   - **Significance:** Enables tailored predictions, ensuring user-specific investment decision-making.
7. **Model Comparison and Recommendation Table:**
   - **Description:** Consolidates predictions from all three models into a summary table, highlighting the most accurate model for a given company.
   - **Significance:** Helps users identify the best-performing model for their stock of interest, improving decision confidence.
8. **Scalable and Modular Codebase:**
   - **Description:** Code is written in a modular format, making it easy to integrate into a larger application or extend with new models/features.
   - **Significance:** Facilitates seamless integration and future scalability of the project.

3. Code Highlights

1. **Dynamic Stock Data Retrieval Using `yfinance`:** Fetch historical stock data for the selected ticker symbol and date range.

```
# Fetch stock data
def get_stock_data(ticker, start_date, end_date):
    stock_data = yf.download(ticker, start=start_date, end=end_date)
    if stock_data.empty:
        raise ValueError("No data fetched for the given ticker and date range.")
    return stock_data
```

2. **LSTM Model for Time-Series Prediction:** Implements a multi-layer LSTM model for sequential stock price predictions.

```
# LSTM prediction
def lstm_predict(data, time_step=60, future_days=5):
    X, y, scaler = prepare_data(data, time_step)
    X = X.reshape(X.shape[0], X.shape[1], 1)

    model = Sequential()
    model.add(LSTM(units=50, return_sequences=True, input_shape=(X.shape[1], 1)))
    model.add(LSTM(units=50, return_sequences=False))
    model.add(Dense(units=1))
    model.compile(optimizer='adam', loss='mean_squared_error')
    model.fit(X, y, epochs=5, batch_size=32, verbose=0)

    future_predictions = []
    last_sequence = X[-1]

    for _ in range(future_days):
        pred = model.predict(last_sequence.reshape(1, time_step, 1))
        future_predictions.append(pred[0, 0])
        last_sequence = np.append(last_sequence[1:], pred[0, 0])

    return scaler.inverse_transform(np.array(future_predictions).reshape(-1, 1)).flatten()
```

3. **ARIMA Model Prediction:** Predict future stock prices using the ARIMA model based on past data.

```python
# ARIMA prediction
def arima_predict(data, future_days=7):
    model = ARIMA(data['Close'], order=(7,0,6))
    model_fit = model.fit()
    forecast = model_fit.forecast(steps=future_days)
    return forecast
```

4. **Evaluation Metrics (MAE and RMSE):** Compute metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) to evaluate model performance.

```python
# Function to compute MAE and RMSE
def compute_metrics(true_values, predicted_values):
    mae = mean_absolute_error(true_values, predicted_values)
    rmse = np.sqrt(mean_squared_error(true_values, predicted_values))
    return mae, rmse
```

4. Challenges and issues

- **Data Quality:** The yfinance library may occasionally return incomplete or inaccurate data, requiring preprocessing and validation before use.
- **Seasonality and Trends:** Stock data is influenced by external factors like market news or macroeconomic trends, which models like ARIMA and XGBoost may not fully capture.
- **Model Limitations:**
  - LSTM may overfit on smaller datasets, reducing generalizability.
  - ARIMA assumes stationarity, which may not hold for all stocks without proper transformation.
- **Scalability:** Processing large volumes of data for multiple stocks can strain memory and computational resources.
- **Edge Cases:** Unpredictable events (e.g., sudden market crashes) are challenging to forecast and may lead to inaccurate predictions.

5. Solutions and Fixes

- **Data Validation:** Implement preprocessing to handle missing values, outliers, and validate data integrity before model training.
- **Hybrid Approaches:** Combine statistical and machine learning methods to balance bias and variance.
- **Feature Engineering:** Incorporate additional financial indicators (e.g., moving averages, RSI) to improve model predictions.

- **Efficient Processing:** Use batch processing for large-scale data handling and optimize model training pipelines.
- **Stress Testing:** Simulate edge cases to evaluate model robustness and adjust algorithms accordingly.

6. Future Work and suggestions

- **Additional Indicators:** Integrate economic indicators and news sentiment data for better context in predictions.
- **Real-Time Predictions:** Extend functionality for near real-time forecasting to suit high-frequency traders.
- **User Feedback Loop:** Allow users to provide feedback on prediction accuracy, which can guide future model improvements.

7. Repository and Documentation Links

GitHub Repository:
AutoProphet GitHub Repository
Project DEMO VIDEO:
Project DEMO video
Additional Resources:

- yfinance Documentation
- LSTM Model Documentation
- XGBoost Documentation