Stock Price Predictor

Team Members: Keith Wood, Miguel Torres, John Garofalo, Juan Valencial, Joseph Ortega

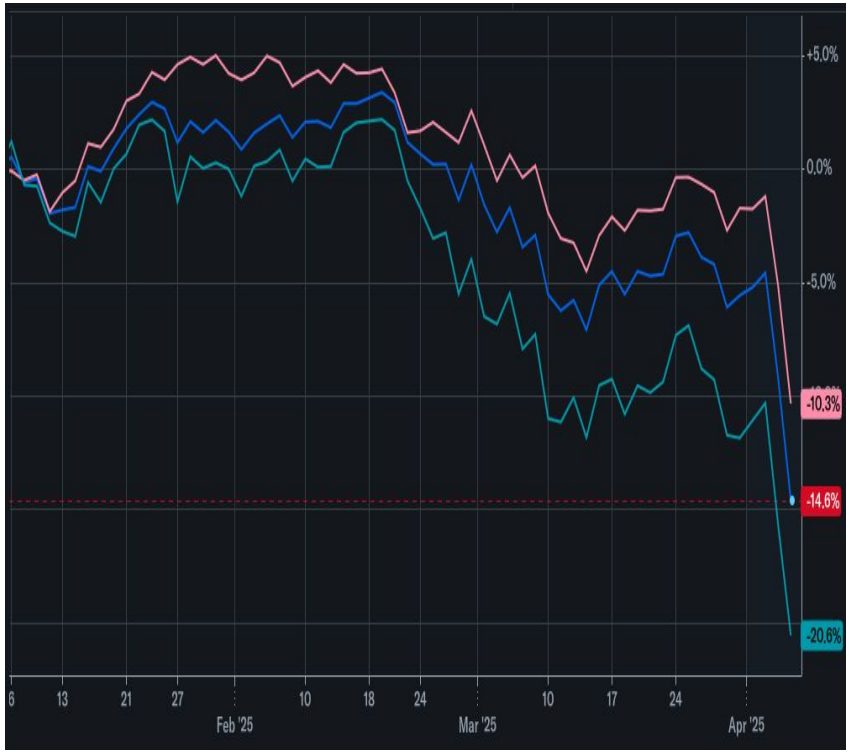
CAP 4630 | Florida Atlantic University

Professor Ahmed Imteaj

Date: 11/08/25

Why Stock Prediction Matters

- Stock markets move billions daily; even a small predictive edge helps investors manage risk.
- Investors prefer data driven decisions over gut reactions.
- AI lets us test predictive power on real world time series.
- Sets the stage for our modeling and evaluation deep dives.



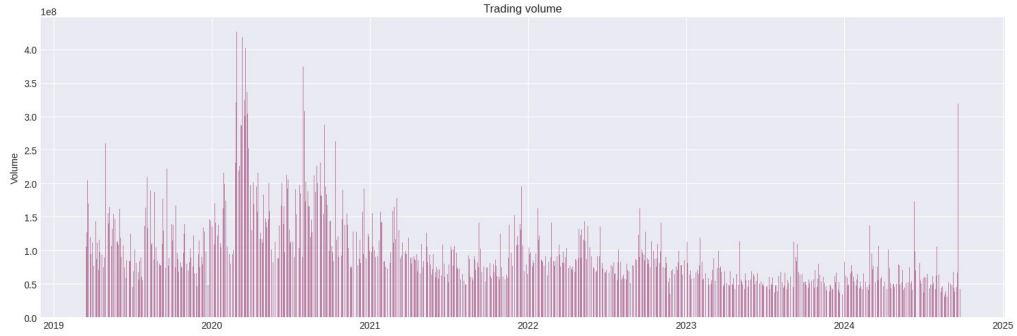
Project Objectives

- Predict the next day’s close for any ticker (default: AAPL)
- Compare Linear Regression, Random Forest, and LSTM
- Track both numerical accuracy (RMSE/MAE/R^2/MAPE) and direction (up/down)
- Deliver visualizations + reports ready for presentation

Dataset Source & Scope

We pull historical OHLCV data from Yahoo Finance using yfinance, which provides a reliable, time indexed feed for daily trading data. When the API is blocked or rate-limited, the pipeline automatically falls back to a cached CSV stored in data/{TICKER}_history.csv, ensuring the workflow still runs end to end

- Default ticker: AAPL (configurable)
- Date range: 2019-01-01 -> 2024-10-01
- Rows: ~1,400 trading days (final processed: 1,396 samples)
- Fields: Date, Open, High, Low, Close, Adj Close, Volume
- Format: Daily time series indexed by trading date



What's Inside the Raw Data?

- Daily OHLC captures trend and range; volume shows liquidity
- Adjusted Close accounts for dividends and splits
- Missing days handled automatically (weekends, market holidays)
- Data is ordered for time-series modeling (no shuffling)

Engineered Indicators & Why These Features Help

Engineered Indicators

- Trend/Momentum: MA-7/21/50, EMA-12/26, MACD, Signal Line, RSI
- Volatility: Bollinger Bands (mid/upper/lower), 21-day std dev of returns
- Context: Lagged closes (Close_Lag_{1,2,3,5,7}) for "memory" in classical ML
- Target: Next-day close (Target = Close.shift(-1))
- Total: 28 engineered features created

Why These Features Help

- Trend & momentum indicators mimic technical analysis signals
- Volatility bands surface risk zones and pressure points
- Lag features give Linear Regression/Random Forest short-term context
- Combined feature set feeds both classical and deep models effectively

With trend, momentum, and volatility features ready, we now feed them into Linear Regression & Random Forest models. Next up: how we train and validate those models.

Modeling Strategy: Data Preparation & Feature Engineering

Preprocessing Steps:

- Fetched AAPL data from Yahoo Finance
- Created **28 engineered features**, including:
  - **Moving Averages (MA 7, 21, 50)**
  - **Exponential Averages (EMA 12, 26)**
  - **MACD + Signal Line**
    RSI (14-day)
  - **Bollinger Bands**
  - **Volatility (21-day std)**
  - **Lag features** (1, 2, 3, 5, 7 days)
- Drop missing values
- Scaled inputs using **MinMaxScaler**
  - Normalizes all features to 0-1 range for equal contribution
- Time-ordered train/test split (no shuffling)
  - First 80% = training (~1,117 samples)
  - Last 20% = testing (~279 samples)

Model Training Flow (LR, RF, LSTM)

Models Used:

- **Linear Regression:** baseline model, handles linear relationships well
- **Random Forest:** nonlinear model, tree-based, often stronger but not here
- **LSTM:** optional deep learning model
    - Turned off (RUN_LSTM = False) to keep the workflow efficient
    - Requires 3D sequential input and longer training time
    - Our engineered features (MA, RSI, MACD, lags) already capture time patterns
    - Simpler models performed very well → LSTM not needed

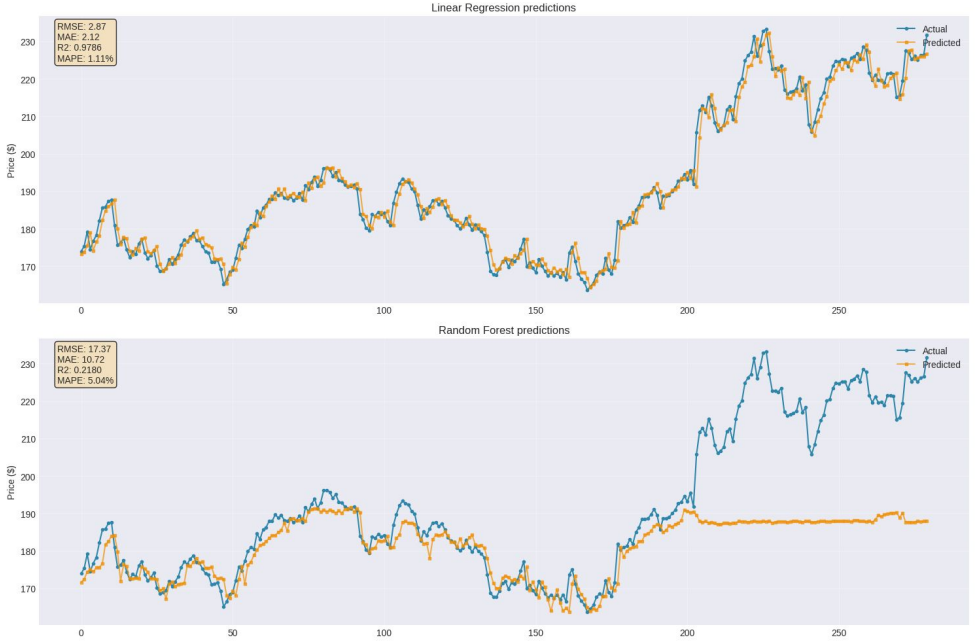
Training Flow:

1. Preprocess + scale data
2. Train **Linear Regression**
3. Train **Random Forest**
4. Optional: train LSTM on sequences
5. Generate predictions on test data
6. Compute metrics: RMSE, MAE, R^2, MAPE
7. Compare model performance
8. Save visualizations + report

Visualization and Insights (Juan)

Actual vs Predicted Prices: LR vs Random Forest

- Linear Regression closely follows the actual price line
- Random Forest deviates significantly, especially in upward trends
- Visual difference shows Linear Regression captures the market direction better
- Random Forest's predictions flatten out due to time-series limitations



Metrics: What and Why

RMSE — Root Mean Squared Error

- Measures **how big the errors are**, giving more weight to larger mistakes.
- Useful when large prediction errors are especially costly in finance.
- In stock prediction, a low RMSE means the model tracks the price closely without big spikes of inaccuracy.

MAE — Mean Absolute Error

- Measures the **average size of all errors** in dollars.
- Easier to interpret: “On average, the model is off by about $X.”
- Good for explaining model performance in real-world, dollar terms.

R^2 — Coefficient of Determination

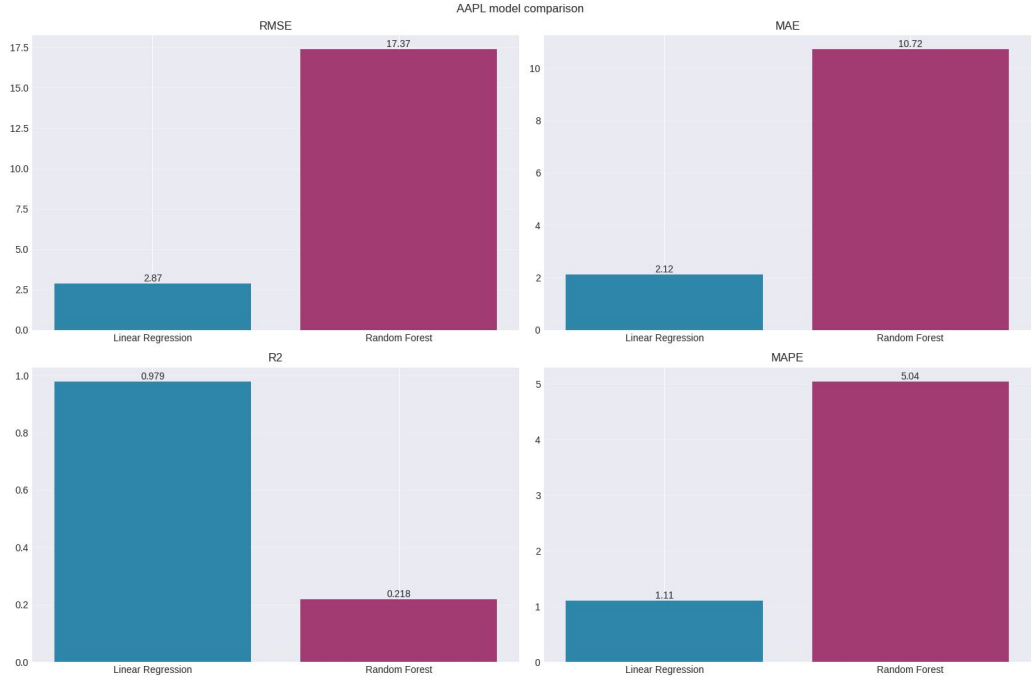
- Tells us **how much of the price movement the model explains**.
- R^2 = 0.98 means the model explains 98% of price changes.
- Useful for understanding whether the model captures overall market patterns.

MAPE — Mean Absolute Percentage Error

- Shows the **average error as a percentage** of the actual stock price.
- Helps evaluate performance independent of the stock’s price range.
- A MAPE of ~1% means predictions are extremely close in relative terms.

Model Performance Metrics

- Linear Regression scores significantly better on all evaluation metrics
- Random Forest has 5–6× higher errors (RMSE & MAE)
- R^2: Linear Regression explains 97.86% of price movements
- MAPE: Linear Regression's ~1% error vs Random Forest's ~5%



Why Did Linear Regression Outperform Random Forest?

- The stock prediction problem behaves closer to a linear relationship between technical indicators and future price.
- Linear Regression fits this structure naturally, which is why it generalizes well and produces stable predictions.
- Financial indicators like Moving Averages, RSI, MACD, and volatility measures often interact linearly with upcoming short-term price moves.
- Random Forest tends to model nonlinear jumps and discontinuous splits, which can break time-dependent relationships.
- The strong R^2 score (98%) shows the features contain enough linearly explainable signal for LR to exploit effectively.

Key Insights & Conclusion

- Random Forest treats each row as an **independent observation**, ignoring sequential order — a major weakness for time series.
- Tree splits cannot naturally capture **smooth trends**, momentum, or lag dependencies unless explicitly engineered.
- RF likely **overfit to noise**, learning small local patterns instead of global price movement.
- The model predictions “flatten out,” indicating underfitting on upward and downward trends.
- With weak temporal structure and technical indicators that behave linearly, RF never finds meaningful nonlinear patterns.
- **Key takeaway:** In time-series forecasting, complex models (RF, NN) are not guaranteed to outperform simpler ones unless temporal features or specialized architectures (e.g., LSTM, ARIMA, XGBoost with lag features) are introduced.

Takeaways & Lessons Learned

- Linear Regression consistently outperformed Random Forest across all metrics (RMSE, MAE, R^2, MAPE).
- The engineered technical indicators (MA, RSI, MACD, Bollinger Bands, lags) contained mostly linear signal, which boosted LR performance.
- Random Forest struggled because it treats each row as independent, missing the sequential nature of stock time-series.
- Simpler models can outperform more complex ones when the feature relationships are linear.
- Visualizations highlighted how LR tracked trends more smoothly, while RF predictions flattened during volatility.
- Overall insight: Model choice must match data structure, not complexity for its own sake.

Future Enhancements & Recommended Improvements

- Integrate true time-series models (LSTM, GRU, ARIMA, Prophet) to capture sequential patterns.
- Add more lag features (Close_Lag 14, 21, 30) to give classical models longer memory.
- Incorporate macro-level features (VIX, interest rates, sector indexes, volume anomalies).
- Use hyperparameter tuning (GridSearch, RandomSearch) for Random Forest to test whether better configurations reduce overfitting.
- Expand evaluation to include directional accuracy (correct up/down prediction rate).
- Test additional models like XGBoost/LightGBM, which handle tabular time data better than standard RF.
- Consider training on multiple tickers to improve generalization and reduce ticker-specific bias.

Thanks!