**Stock Trend Prediction Using LSTM and SVM Hybrid Model with Technical Indicators**

**Name:** Yongzhen “Michael” Zhang

**Date:** 4/21/2025

**Introduction**

The stock market is a vital component of the global economy, and accurately forecasting stock price trends can provide significant advantages for investors, traders, and financial analysts. However, due to the market’s inherent volatility and the influence of diverse factors—such as economic indicators, geopolitical developments, investor behavior, and unexpected global events—traditional statistical models and human judgment often fall short. These variables introduce substantial noise and uncertainty, making precise predictions extremely challenging. This project addresses this complexity by developing a hybrid machine learning model that combines Long Short-Term Memory (LSTM) neural networks with Support Vector Machines (SVMs) to identify meaningful patterns in historical and real-time market data, enabling more reliable stock trend prediction.

**Methods & Approach**

The project was implemented in Python using libraries including yfinance, NumPy, Pandas, Matplotlib, TensorFlow, and scikit-learn.

1. Data Acquisition & Preprocessing

* Collected historical daily stock data for Apple Inc. (AAPL) from 2020–2024 using the yfinance API.
* Cleaned the dataset by dropping missing and duplicate values.
* Calculated two common technical indicators—Relative Strength Index (RSI) and the 10-day Moving Average (MA\_10)—which were appended to the feature set for trend classification.
* One-hot encoded day-of-week (Weekday\_0 to Weekday\_4)
* All numerical features were normalized using MinMaxScaler for faster and more stable LSTM training.

2. Sequence Preparation

* A window size of 60 days was chosen to build time-series sequences.
* Each input sample contains 60 timesteps of 12 features (including price and indicators)
* The output for each sequence is the next day’s normalized closing price (for LSTM regression).
* Binary trend labels (1 = up, 0 = down) were generated from the LSTM output for SVM classification.

3. LSTM Model Architecture

A three-layer Bidirectional LSTM was used:

First layer: 64 units, return\_sequences=True

Second layer: 32 units, return\_sequences=True

Dropout (rate = 0.3) was applied between layers to reduce overfitting.

Third layer: 16 units, return\_sequences = False

Dropout (rate = 0.2)

Output layers:

Dense (32, activation=tanh)

Dense (1, activation=Linear) for continuous price prediction

Loss function: Mean Squared Error (MSE)

Optimizer: Adam

4. SVM Trend Classifier

LSTM predictions were processed into trend labels (up/down).

An SVM classifier with RBF kernel and a balanced class weight was trained to predict directional movement based on predicted prices.

The SVM model was trained on 80% of the data and evaluated on the remaining 20%.

**Results & Analysis**

LSTM Regression Performance:

LSTM Mean Squared Error: 0.003751764539629221

LSTM Root Mean Squared Error: 0.06125164928089056

SVM Classification Performance:

SVM Accuracy: 76%

A blue squares with numbers and a chart

AI-generated content may be incorrect.

The final LSTM model achieved an MSE of 0.00375 and an RMSE of 0.0613, indicating strong predictive accuracy.

The SVM classifier, using LSTM-predicted values as input, achieved an accuracy of 76%, with an F1-score of 0.73 for Down trends and 0.78 for Up trends. This balanced performance shows that the LSTM successfully encodes trend-relevant signals, which the SVM leverages to make robust directional predictions.

**Visualizations**

Actual vs. Predicted Stock Prices

A graph of a graph showing the price of a stock market

AI-generated content may be incorrect.

Visually, the predicted price closely follows the actual price trend, although some underestimation during sharp upward movements remains.

**Conclusion & Reflection**

The choice of a hybrid LSTM-SVM architecture was based on the strengths of each component model. LSTM networks are designed specifically to capture long-term dependencies in sequential data, making them highly suitable for time-series forecasting problems like stock price prediction. However, LSTM outputs are continuous values that may not directly indicate clear trends. Therefore, an SVM classifier was added to interpret the predicted price signals and convert them into directional movement classifications (up/down).

Hyperparameters such as window size (60), number of LSTM units, batch size (64), and epochs (50) were chosen based on simplicity, standard practice, and computational efficiency. These values offered a good trade-off between model complexity, training time, and performance. A window size of 60 provides two months of market context for each prediction, which is long enough to capture meaningful trends without adding too much noise or complexity to the model.

Hyperparameter Selection & Validation Strategy

In this project, a formal hyperparameter tuning process (such as grid search or random search) was not implemented due to the high computational cost of training deep learning models like LSTMs and the sequential dependency in time-series data, which complicates traditional cross-validation methods.

A single train-test split (80/20) was used rather than k-fold cross-validation. Although more advanced techniques like Bayesian optimization or walk-forward validation could further improve model robustness, the current strategy provided a practical balance between getting solid results and keeping things practical.

Conclusion

This project successfully built and tested a hybrid model using LSTM and SVM to predict stock market trends. The LSTM was able to learn patterns in stock prices and technical indicators over time, while the SVM helped turn those predictions into clear up or down-trend decisions. Together, they worked well for both price prediction and trend classification. And due to the time constraints, a real-time prediction pipeline was not implemented.

While there’s still room to improve the accuracy of the SVM, the results show that the LSTM captures useful signals, even with the noisy nature of market data. In the future, performance could be improved by adding more technical indicators like MACD or Bollinger Bands, trying out other classifiers like XGBoost or Random Forest, and testing the model with real-time or walk-forward validation. Overall, this project helped me better understand how deep learning and machine learning can work together for time-series forecasting and showed how hybrid models can be used in real-world financial applications.

**Reference:**

1. Hochreiter, S., & Schmidhuber, J. (1997). *Long Short-Term Memory.* Neural Computation, 9(8), 1735–1780.

2. Cortes, C., & Vapnik, V. (1995). *Support-vector networks*. Machine Learning, 20(3), 273–297.

3. Brownlee, J. (2017). *Introduction to Time Series Forecasting with Python*. Machine Learning Mastery.

4. Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). *Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques*. Expert Systems with Applications, 42(1), 259-268.