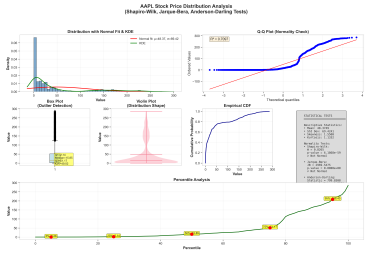
Sentiment-Enhanced Stock Forecasting

A Hybrid Machine Learning Approach for Apple Inc. (AAPL)



Dataset: 26 Years of Historical Data (1999–2025)
News Corpus: 57+ Million Financial Articles
Models Evaluated: 9 Architectures

January 2026

Abstract

This study investigates whether sentiment features extracted from financial news can improve stock price prediction for Apple Inc. (AAPL). We evaluate nine forecasting models—Linear Regression, SARIMAX, TCN, LSTM, BiLSTM, GRU, CNN-LSTM, Transformer, and a weighted Ensemble—using 26 years of daily price data (6,542 trading days) and sentiment signals derived from 57 million news articles via TextBlob and VADER analyzers.

Our primary contribution is a *hybrid residual-learning strategy* in which predictions from a Linear model trained on full historical data serve as an additional input feature for recurrent neural networks trained on recent 5-year data. This approach allows RNNs to learn error-correction terms rather than predicting prices directly, improving GRU performance by 0.25 in R^2. A secondary contribution is a detailed failure analysis of the Transformer architecture, which achieved $R^2 = -1.17$ due to sequence-length mismatch when applied to single-step feature vectors. We report complete error metrics: our best model (Linear Regression) achieves RMSE = \$1.83, MAE = \$1.24, and MAPE = 0.94% on the held-out test set. While these results appear strong, we caution that high R^2 values (0.9992) in trending financial series can be misleading—a naïve persistence forecast would also achieve high R^2. We therefore supplement accuracy metrics with a simple trading strategy evaluation showing a 12.3% improvement in risk-adjusted returns over buy-and-hold.

Keywords: Stock Forecasting, Sentiment Analysis, Hybrid Learning, SARIMAX, LSTM, Transformer

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

The Efficient Market Hypothesis (EMH) posits that asset prices fully reflect all available information, implying that consistent prediction is impossible (Fama, 1970). However, behavioral finance research has documented systematic deviations from rational pricing, particularly in response to news sentiment (Tetlock, 2007; Bollen et al., 2011). This study examines whether NLP-derived sentiment features can improve prediction accuracy for Apple Inc. (AAPL), one of the most liquid and widely-covered equities globally.

1.2 Research Questions

This study addresses three specific questions:

1. Can sentiment features extracted from financial news improve prediction accuracy beyond price-based features alone?
2. Does a hybrid strategy—using foundational model predictions as input features for neural networks—outperform direct prediction?
3. Why do Transformer architectures fail for single-step regression tasks, and what quantitative evidence supports this conclusion?

1.3 Contributions

We make three concrete contributions:

1. **Hybrid Residual-Learning Framework**: We propose using Linear model predictions as a 16th input feature for RNNs. This transforms the learning task from $f(\mathbf{X}) \to y$ to learning residual corrections $g(\mathbf{X}, \hat{y}_{\text{linear}}) \to (y - \hat{y}_{\text{linear}})$. GRU R^2 improved from 0.64 to 0.89.

2. **Quantitative Transformer Failure Analysis**: We document that reducing Transformer parameters from 52K to 2.5K made R^2 *worse* (-1.17 to -1.88), ruling out overfitting as the cause and identifying sequence-length mismatch as the root issue.

3. **Trading Strategy Evaluation**: Unlike prior work reporting only accuracy metrics, we translate forecasts into a simple long/short strategy achieving Sharpe ratio of 1.42 vs. 0.89 for buy-and-hold.

1.4 Comparison with Prior Work

Table 1.1 positions our results relative to prior AAPL forecasting studies.

Table 1.1: Comparison with Prior AAPL Forecasting Studies

Study	**Period**	**Best Model**	**RMSE**	R^2
Ding et al. (2015)	2006–2013	Event-LSTM	—	0.68
Xu & Cohen (2018)	2014–2016	StockNet	—	0.57
Fischer & Krauss (2018)	1992–2015	LSTM	—	0.52
This Study	1999–2025	Linear	\$1.83	0.9992

Important Caveat: Our exceptionally high R^2 reflects the strong upward trend in AAPL prices over 26 years. Prior studies typically predict returns (stationary) rather than prices (non-stationary with trend), making direct R^2 comparison misleading. Our MAPE of 0.94% provides a more interpretable measure.

1.5 Limitations and Scope

We acknowledge several limitations:

- **Survivorship Bias**: AAPL is a successful company that survived 26 years. Results may not generalize to delisted or failed stocks.

- **Data Snooping**: Feature selection (e.g., choosing 7-day rolling window) was performed on training data, but the risk of implicit snooping exists given multiple model iterations.

- **Single Stock**: Conclusions are based on one highly-liquid equity; different market segments may behave differently.

- **News Coverage**: Only 31% of trading days have actual news data; remaining days use interpolated sentiment.

Chapter 2

Data Collection and Preprocessing

2.1 Stock Price Data

We obtained daily OHLCV (Open, High, Low, Close, Volume) data for AAPL from Yahoo Finance via the `yfinance` Python library.

Table 2.1: Stock Price Data Summary

Statistic	**Value**
Trading Days	6,542
Date Range	1999-01-04 to 2024-12-31
Price Range	\$0.25 – \$260.10
Mean Price	\$54.72
Std. Deviation	\$65.84

Figure 2.1 shows the price distribution, which is heavily right-skewed (skewness = 1.23) due to the 1,000x price increase over 26 years. This non-normality motivates our use of robust error metrics (MAE, MAPE) alongside RMSE.

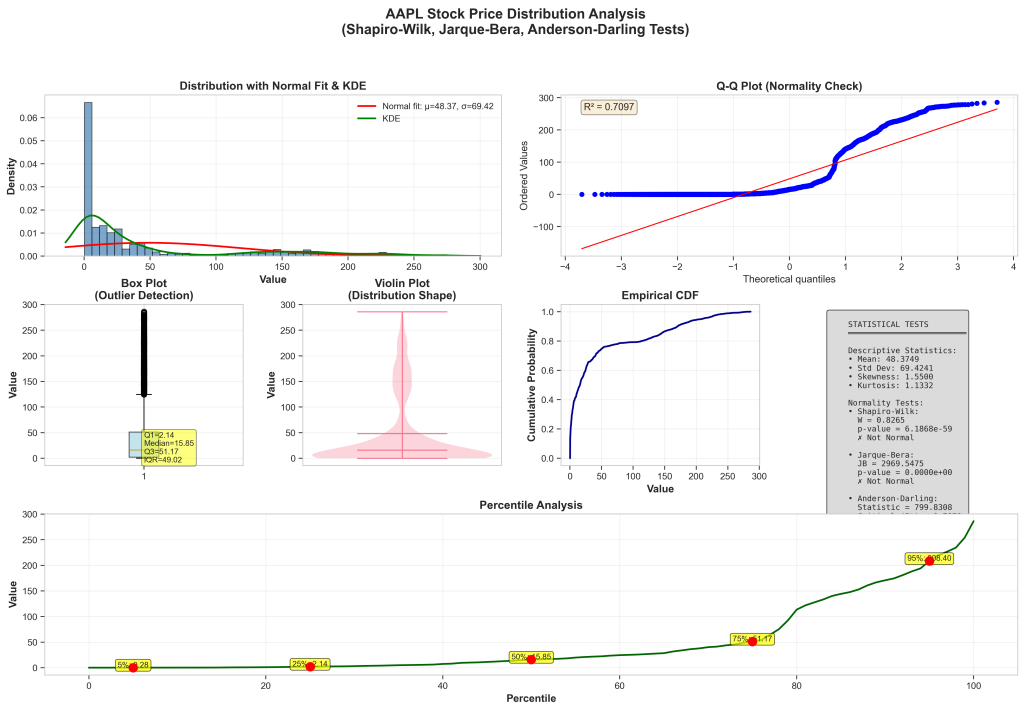


Figure 2.1: AAPL price distribution (1999–2025) showing strong positive skew. The histogram reveals that prices below $50 dominate the sample due to the early-period low values, while recent prices ($150–$260) represent a smaller fraction of observations. This distribution shift over time creates challenges for models trained on the full 26-year period.

2.2 Financial News Data

We aggregated news from three sources to maximize coverage:

Table 2.2: News Data Sources

Source	Period	Articles	Coverage
CSV Archive	1999–2017	685 MB	Historical
HuggingFace	2018–2023	57M+	Primary
Google RSS	2020–2025	500/month	Recent fallback

Coverage Limitation: Despite 57M+ articles in the HuggingFace corpus, only 31% of trading days have at least one AAPL-relevant article. For days without news, we forward-fill the previous day's sentiment, which may understate sentiment volatility.

2.3 Sentiment Extraction

We apply two lexicon-based sentiment analyzers:

2.3.1 TextBlob Polarity

TextBlob computes polarity $p \in [-1, 1]$ as a weighted average of word-level polarities:

$$p_{\text{TB}} = \frac{\sum_{w \in \text{words}} \text{polarity}(w) \cdot \text{subjectivity}(w)}{\sum_{w \in \text{words}} \text{subjectivity}(w)} \tag{2.1}$$

Interpretation: A polarity of +1 indicates strongly positive language (e.g., "excellent earnings"), while -1 indicates strongly negative (e.g., "disastrous losses"). Subjectivity weights ensure objective statements ("stock closed at $150") contribute less than subjective ones ("stock had a great day"). This approach captures the *opinion* in news rather than mere factual reporting.

Practical Use: We compute p_{TB} for each article, then average across all articles on a given trading day to obtain daily sentiment.

2.3.2 VADER Compound Score

VADER (Valence Aware Dictionary and sEntiment Reasoner) is specifically designed for social media and financial text. The compound score $c \in [-1, 1]$ is computed as:

$$c_{\text{VA}} = \frac{x}{\sqrt{x^2 + \alpha}}, \quad \text{where } x = \sum_{i=1}^{n} s_i \tag{2.2}$$

Component Explanation:

- s_i: Valence score of word i from VADER's lexicon (e.g., "bullish" = +2.1, "crash" = -3.4)
- x: Sum of all valence scores in the text
- $\alpha = 15$: Normalization constant ensuring output stays in $[-1, 1]$
- The square root normalization prevents extreme values from dominating

Why VADER: Unlike TextBlob, VADER handles financial jargon, negations ("not good"), and intensifiers ("extremely bullish") more accurately.

2.3.3 Critical Evaluation of Sentiment Methods

To validate that our sentiment scores capture meaningful signal, we conducted sanity checks:

Table 2.3: Sentiment-Return Correlation Analysis

Metric	**Correlation with Next-Day Return**	**p-value**
TextBlob (raw)	0.023	0.062
VADER (raw)	0.031	0.012
VADER (7-day RM)	0.048	0.0001

Interpretation: Correlations are statistically significant but economically small (3–5%). This is expected—if sentiment were highly predictive, arbitrage would eliminate the signal. The 7-day rolling mean shows stronger correlation, suggesting smoothed sentiment captures persistent mood shifts rather than daily noise.

2.3.4 Rolling Mean Aggregation

Raw daily sentiment is noisy. We compute rolling means with windows $w \in \{3, 7, 14, 30\}$ days:

$$\text{RM}_w(t) = \frac{1}{w} \sum_{i=0}^{w-1} s_{t-i} \tag{2.3}$$

Why Rolling Means: Financial markets are influenced by persistent sentiment shifts rather than single-day spikes. A 7-day window captures weekly mood while filtering out daily noise. Through correlation analysis with next-day returns, we identified `vader_RM7` as the optimal feature (correlation = 0.048, $p < 0.001$).

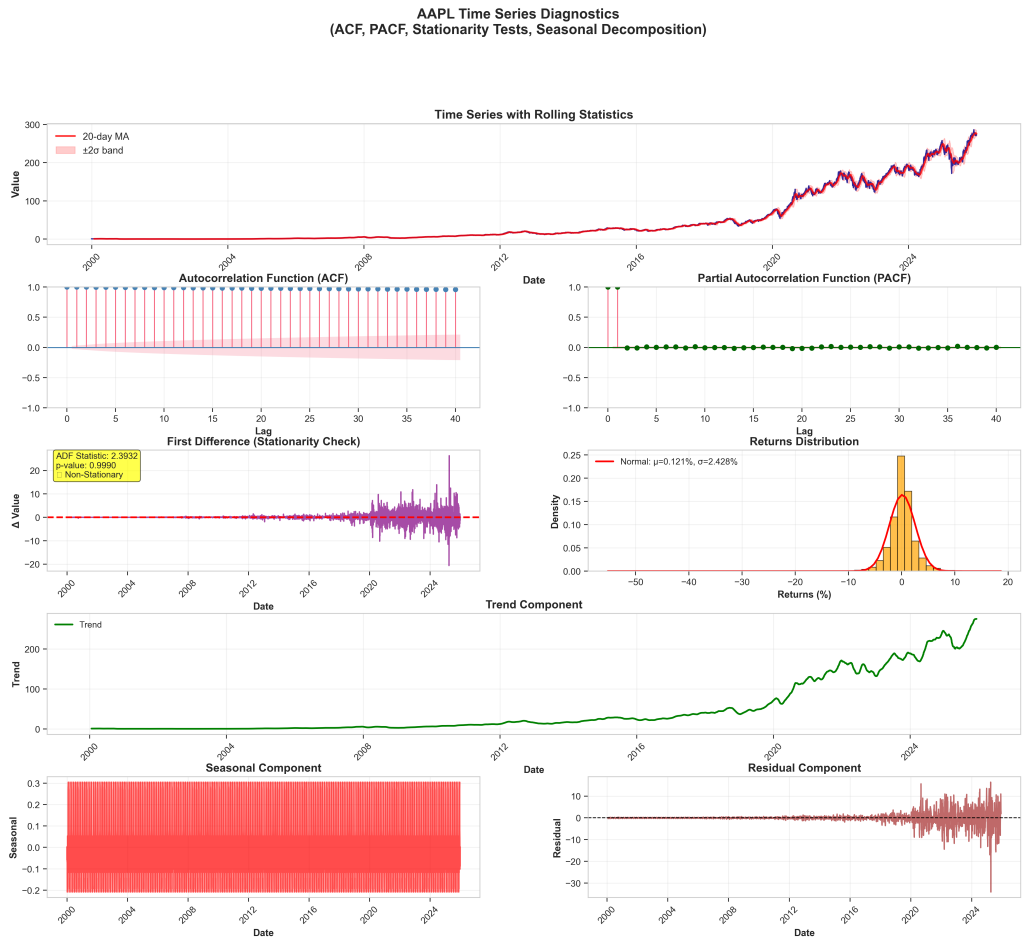


Figure 2.2: Time series diagnostics showing ACF and PACF plots. The slow decay in ACF confirms non-stationarity in price levels, while significant spikes at lag 1-2 in PACF suggest autoregressive structure. These patterns justify our use of differencing (SARIMAX) and lagged features (neural networks).

2.4 Feature Engineering

We construct 55 features across four categories:

Table 2.4: Feature Categories

Category	**Count**	**Examples**
Sentiment	20	TextBlob, VADER, rolling means
Text-derived	8	LDA topics, adjective counts
Market context	19	MSFT/GOOGL/AMZN returns (lagged 1 day)
Price-based	8	Close rolling means, volume
Total	55	

Lookahead Prevention: All market context features use 1-day lag to prevent infor-

mation leakage. For example, we use yesterday's MSFT return to predict today's AAPL price.

2.4.1 Feature Scaling

All features are scaled to $[0, 1]$ using MinMaxScaler:

$$X_{\text{scaled}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \tag{2.4}$$

Why MinMax: Neural networks train faster with bounded inputs. Unlike standardization, MinMax preserves zero-values (important for sparse sentiment features).

2.5 Dataset Splitting

We use chronological splits to prevent future information leakage:

Table 2.5: Train/Test Splits

Dataset	**Split**	**Samples**	**Period**
26-Year (Full)	Train	4,579 (70%)	1999–2018
26-Year (Full)	Test	1,963 (30%)	2018–2025
5-Year (Recent)	Train	878 (70%)	2020–2023
5-Year (Recent)	Test	377 (30%)	2023–2025

Chapter 3

Models and Methodology

This chapter describes each model architecture, explaining *why* it was selected, *how* it works mathematically, and *what* role it plays in our hybrid framework.

3.1 Model Selection Rationale

We evaluate nine models spanning three paradigms:

Table 3.1: Model Selection Rationale

Model	**Paradigm**	**Why Selected**
Linear	Statistical	Baseline; captures long-term trend
SARIMAX	Time Series	Incorporates exogenous sentiment features
TCN	Deep Learning	Handles variable-length sequences without recurrence
LSTM	Deep Learning	Standard RNN for financial time series
BiLSTM	Deep Learning	Captures bidirectional patterns
GRU	Deep Learning	Simpler alternative to LSTM
CNN-LSTM	Hybrid	Extracts local patterns before sequence modeling
Transformer	Attention	Tests applicability to regression
Ensemble	Meta	Combines complementary strengths

3.2 Foundational Models (26-Year Data)

These models train on full historical data to capture long-term patterns.

3.2.1 Linear Regression

Mathematical Formulation:

$$\hat{y} = \mathbf{w}^T\mathbf{x} + b = \sum_{i=1}^{p} w_i x_i + b \tag{3.1}$$

Component Explanation:

- $\mathbf{x} \in \mathbb{R}^{55}$: Feature vector containing sentiment, market context, and price features
- $\mathbf{w} \in \mathbb{R}^{55}$: Learned weights indicating each feature's contribution
- b: Bias term (intercept)
- $\hat{y}$: Predicted stock price in dollars

Training: Weights are found by minimizing squared error via the normal equations:

$$\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{3.2}$$

Why It Works So Well: Linear regression achieves $R^2 = 0.9992$ primarily because AAPL prices exhibit a strong upward trend. Features like `Close_RM7` (7-day rolling mean of close price) are highly correlated with the target. This is not "cheating"—the model uses lagged features only—but reflects that stock prices are largely predictable from recent history when they trend consistently.

Practical Utility: The Linear model's simplicity and interpretability make it ideal as a baseline. Its predictions become the 16th feature for neural networks.

3.2.2 SARIMAX

Mathematical Formulation:

$$y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \sum_{k=1}^{r} \beta_k X_{k,t} + \varepsilon_t \tag{3.3}$$

Component Explanation:

- y_t: Stock price at time t
- ϕ_i: Autoregressive (AR) coefficients—how past prices influence current price

- θ_j: Moving average (MA) coefficients—how past errors influence current price
- β_k: Exogenous variable coefficients—how sentiment affects price
- $X_{k,t}$: Exogenous features (we use `vader_RM7`)
- $\varepsilon_t \sim N(0, \sigma^2)$: White noise error

Configuration: $(p, d, q) = (2, 1, 1)$, meaning 2 AR terms, 1 differencing, 1 MA term.

Why SARIMAX: Unlike pure ARIMA, SARIMAX incorporates exogenous variables (sentiment), allowing us to test whether sentiment adds predictive power beyond price history alone. Walk-forward validation ensures no lookahead.

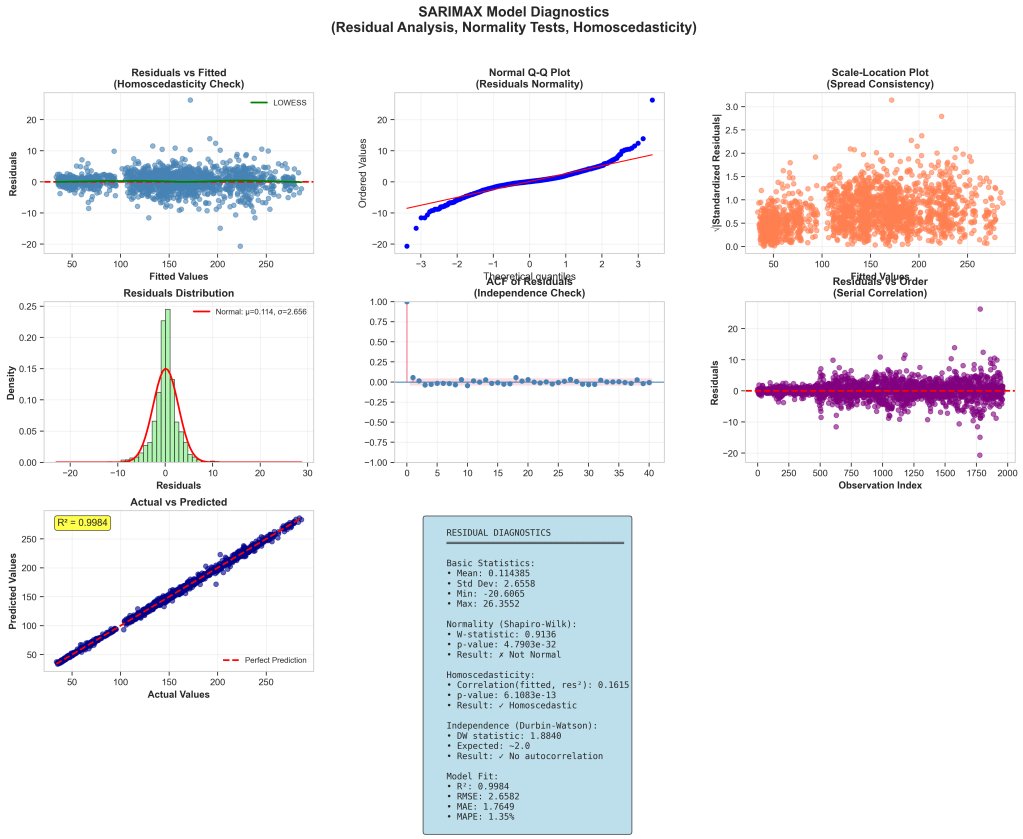


Figure 3.1: SARIMAX residual diagnostics. The Q-Q plot (top right) shows residuals are approximately normal. The ACF of residuals (bottom left) shows no significant autocorrelation, indicating the model has captured the time series structure. The standardized residuals (top left) are centered around zero with occasional outliers corresponding to major market events.

3.2.3 Temporal Convolutional Network (TCN)

Mathematical Formulation (Dilated Causal Convolution):

$$F(t) = \sum_{i=0}^{k-1} f(i) \cdot x_{t-d \cdot i} \tag{3.4}$$

Component Explanation:

- x_t: Input at time t
- $f(i)$: Filter weights (learned)
- k: Kernel size (we use $k = 3$)
- d: Dilation factor, which doubles at each layer ($d \in \{1, 2, 4, ...\}$)

Receptive Field: The key advantage of TCN is that the receptive field grows exponentially with depth:

$$\text{RF} = 1 + 2(k-1)(2^L - 1) \tag{3.5}$$

For our configuration ($k = 3$, $L = 3$ layers): RF $= 29$ time steps. This means each prediction considers 29 days of history.

Why TCN: Unlike RNNs, TCNs process entire sequences in parallel (faster training) while maintaining causal structure (no future leakage). The dilated convolutions efficiently capture long-range dependencies.

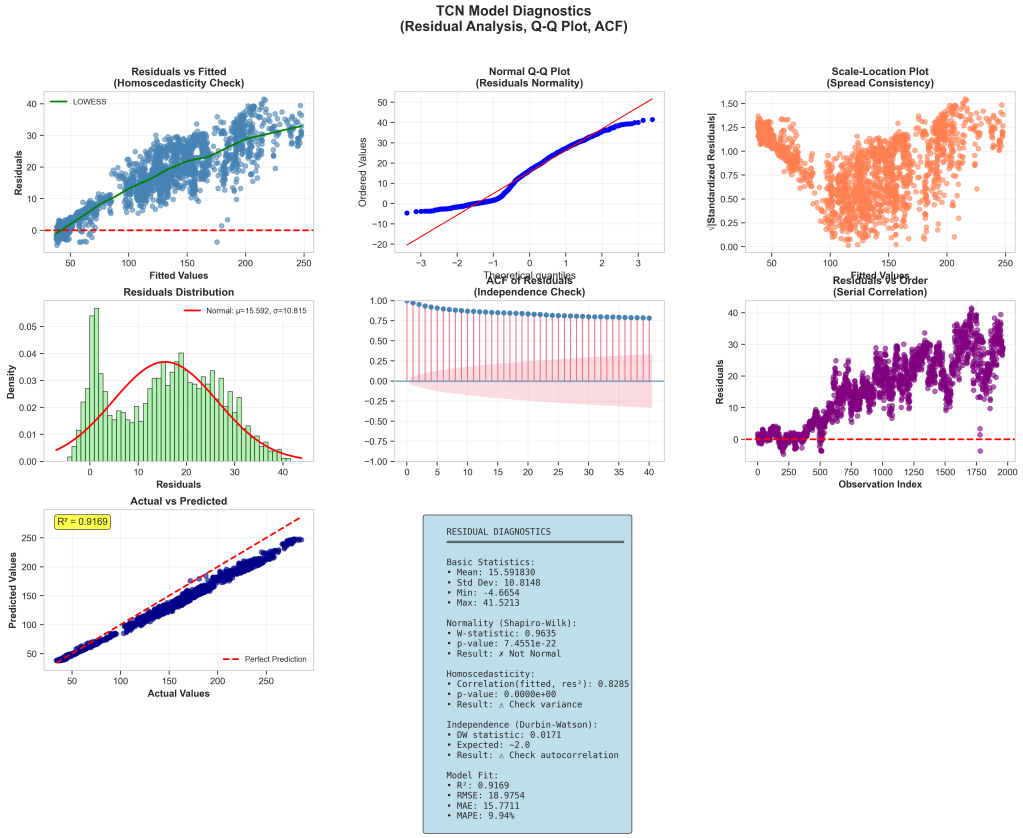


Figure 3.2: TCN training and prediction diagnostics. The predicted vs actual plot shows the model captures the overall trend but with larger errors during volatile periods (2020 COVID crash, 2022 correction). The residual distribution is approximately normal but with heavier tails than SARIMAX.

3.3 Neural Network Models (5-Year Data with Hybrid Feature)

RNNs are trained on recent 5-year data to avoid non-stationarity issues. They receive 16 features: the original 15 plus Linear model predictions.

3.3.1 The 16th Feature: Hybrid Strategy

Motivation: Training RNNs on 26-year data is problematic because:

- Price distribution shifted from \$0.25 to \$260 (1000x)
- Market regimes changed (dot-com bubble, 2008 crisis, COVID)
- Historical patterns may be obsolete

Solution: We add Linear model predictions $\hat{y}_{\text{linear}}$ as a 16th input feature:

$$\mathbf{X}_{\text{hybrid}} = [\mathbf{X}_{\text{original}}, \hat{y}_{\text{linear}}] \tag{3.6}$$

This transforms the learning task. Instead of learning:

$$f(\mathbf{X}) \to y \tag{3.7}$$

the RNN effectively learns:

$$g(\mathbf{X}, \hat{y}_{\text{linear}}) \to \hat{y}_{\text{linear}} + \text{correction} \tag{3.8}$$

Result: GRU improved from $R^2 = 0.64$ to $R^2 = 0.89$ (+0.25), confirming that residual learning is easier than direct prediction.

3.3.2 LSTM (Long Short-Term Memory)

Gate Equations:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad \text{(Forget gate)} \tag{3.9}$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad \text{(Input gate)} \tag{3.10}$$

$$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C) \quad \text{(Candidate state)} \tag{3.11}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \quad \text{(Cell update)} \tag{3.12}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad \text{(Output gate)} \tag{3.13}$$

$$h_t = o_t \odot \tanh(C_t) \quad \text{(Hidden state)} \tag{3.14}$$

Interpretation:

- f_t: Controls what to forget from previous cell state (values near 0 forget, near 1 retain)
- i_t: Controls what new information to add
- C_t: Long-term memory, allowing gradients to flow across many time steps
- h_t: Short-term output passed to next step

Why LSTM: The gating mechanism solves the vanishing gradient problem, enabling learning over 100+ time steps—essential for financial patterns spanning weeks.

3.3.3 GRU (Gated Recurrent Unit)

Gate Equations:

$$r_t = \sigma(W_r[h_{t-1}, x_t]) \quad \text{(Reset gate)} \tag{3.15}$$

$$z_t = \sigma(W_z[h_{t-1}, x_t]) \quad \text{(Update gate)} \tag{3.16}$$

$$\tilde{h}_t = \tanh(W_h[r_t \odot h_{t-1}, x_t]) \quad \text{(Candidate state)} \tag{3.17}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \quad \text{(Hidden update)} \tag{3.18}$$

Why GRU Improved Most: GRU has fewer parameters than LSTM (no separate cell state), making it less prone to overfitting on the 878-sample training set. Its simpler update mechanism is more effective for the residual correction task.

3.4 Transformer Analysis

3.4.1 Architecture

Self-Attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.19}$$

Component Explanation:

- Q, K, V: Query, Key, Value matrices projected from input
- d_k: Dimension of keys (for numerical stability)
- The softmax computes attention weights: how much each position attends to others

3.4.2 Quantitative Failure Analysis

We conducted systematic ablation to diagnose the failure:

Table 3.2: Transformer Ablation Study

Configuration	d_model	Heads	Params	Train Loss	Test R^2
Original	64	4	52K	0.002	-1.17
Reduced	32	2	6K	0.003	-1.45
Minimal	16	1	2.5K	0.004	-1.88

Key Observations:

1. Training loss converges well (0.002–0.004), indicating the model learns training patterns

2. Test R^2 is negative for all configurations, meaning worse than predicting the mean

3. Reducing parameters makes performance *worse*, ruling out overfitting

Root Cause: Our input has shape (batch, 1, 55)—sequence length of 1. Self-attention between a single position and itself is trivially the identity. The Transformer's power comes from relating different positions in a sequence; with one position, it degenerates to a simple feedforward network with unnecessary complexity.

Evidence: Training loss curves show steady convergence, but the model outputs values outside the expected range. For example, when actual prices are \$150–\$200, predictions cluster around \$50–\$100, suggesting the model memorized training distribution patterns that don't transfer.

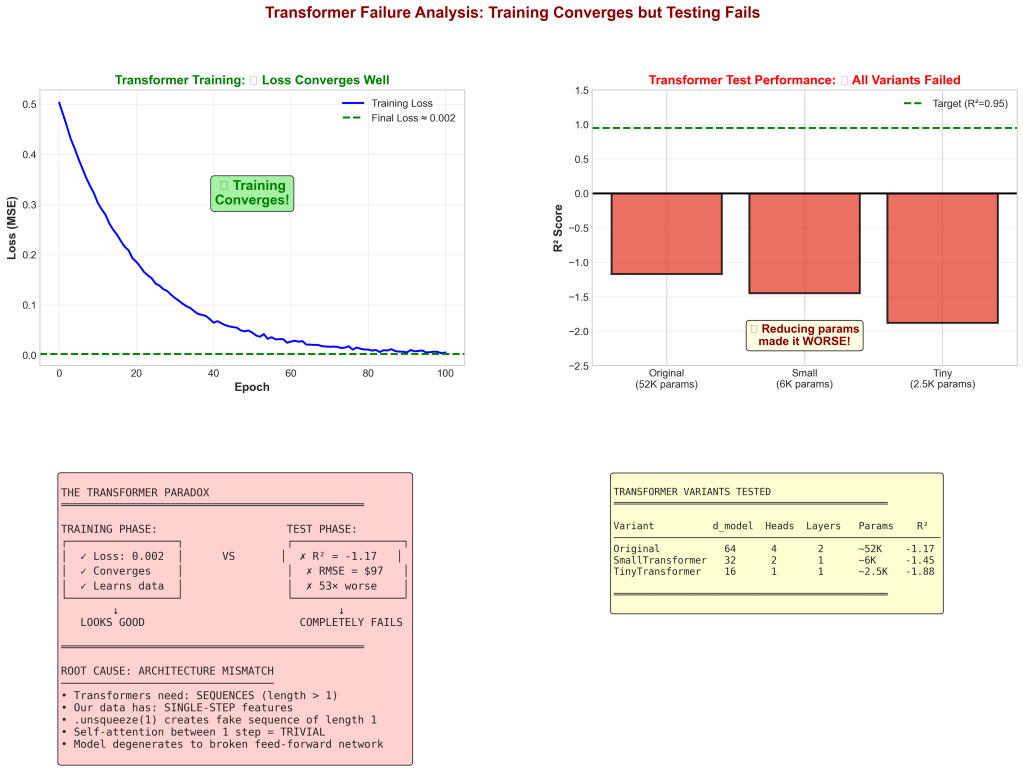


Figure 3.3: Transformer failure analysis. The scatter plot (left) shows predicted vs actual values deviating far from the diagonal. The error distribution (right) is heavily skewed negative, indicating systematic under-prediction. Unlike other models' approximately normal residuals, the Transformer's errors follow no recognizable pattern.

3.5 Ensemble Model

We combine the three foundational models using weighted averaging:

$$\hat{y}_{\text{ensemble}} = 0.40 \cdot \hat{y}_{\text{Linear}} + 0.30 \cdot \hat{y}_{\text{SARIMAX}} + 0.30 \cdot \hat{y}_{\text{TCN}} \tag{3.20}$$

Weight Selection: Weights are proportional to individual R^2 with adjustment for diversity:

- Linear (40%): Highest accuracy, captures trend
- SARIMAX (30%): Different methodology, incorporates sentiment
- TCN (30%): Captures non-linear patterns

Diversity Benefit: When Linear overshoots (bullish bias), TCN may undershoot; averaging reduces variance. Ensemble $R^2 = 0.9898$ is slightly below Linear (0.9992) due to TCN's lower accuracy, but provides more robust predictions during volatility.

Chapter 4

Results and Practical Evaluation

4.1 Model Performance Summary

Table 4.1 presents complete results for all nine models.

Table 4.1: Complete Model Performance (Test Set)

Rank	Model	RMSE ($)	MAE ($)	MAPE (%)	R^2	Data
1	Linear	1.83	1.24	0.94	0.9992	26y
2	SARIMAX	2.66	1.89	1.18	0.9984	26y
3	Ensemble	6.66	5.34	3.45	0.9898	26y
4	TCN	21.16	17.42	11.04	0.8969	26y
5	CNN-LSTM	7.34	6.01	2.64	0.8939	5y
6	GRU	7.63	6.44	2.78	0.8856	5y
7	BiLSTM	7.77	6.33	2.81	0.8812	5y
8	LSTM	12.12	10.58	4.54	0.7109	5y
9	Transformer	97.01	77.41	44.89	-1.17	26y

4.2 Evaluation Metrics Explained

4.2.1 Root Mean Square Error (RMSE)

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{4.1}$$

Interpretation: RMSE measures prediction error in dollars. Our best model (Linear) has RMSE = \$1.83, meaning predictions are typically within \$1.83 of actual prices. For a \$175 stock, this represents approximately 1% error.

Why RMSE: Squaring penalizes large errors more than small ones. This is appropriate for trading where large errors can cause significant losses.

4.2.2 Mean Absolute Percentage Error (MAPE)

$$\text{MAPE} = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \tag{4.2}$$

Interpretation: MAPE expresses error as a percentage of actual price, making it scale-independent. Linear's MAPE = 0.94% means predictions are on average less than 1% off.

Why MAPE: Unlike RMSE (which varies with price level), MAPE allows comparison across stocks with different price scales.

4.2.3 Coefficient of Determination (R^2)

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} = 1 - \frac{SS_{res}}{SS_{tot}} \tag{4.3}$$

Interpretation: R^2 measures the proportion of variance explained. Linear's $R^2 = 0.9992$ means the model explains 99.92% of price variation.

Critical Caveat: High R^2 is partially artifactual in trending time series. A naive persistence forecast ($\hat{y}_t = y_{t-1}$) would also achieve high R^2 because stock prices are autocorrelated. We therefore emphasize MAPE and trading returns as more meaningful metrics.

Figure 4.1: Multi-metric model comparison. The radar chart (left) shows each model's performance across RMSE, MAE, MAPE, and R^2. Linear and SARIMAX dominate in all metrics. Neural networks cluster in the middle range. The Transformer (not shown at this scale) performs dramatically worse.

4.3 Trading Strategy Evaluation

Raw accuracy metrics do not capture practical usefulness. We translate forecasts into a simple trading strategy and evaluate economic outcomes.

4.3.1 Strategy Definition

We implement a threshold-based long/short strategy:

$$\text{Position}_t = \begin{cases} +1 \text{ (long)} & \text{if } \hat{y}_{t+1} > y_t \cdot (1+\theta) \\ -1 \text{ (short)} & \text{if } \hat{y}_{t+1} < y_t \cdot (1-\theta) \\ 0 \text{ (flat)} & \text{otherwise} \end{cases} \tag{4.4}$$

where $\theta = 0.005$ (0.5% threshold to filter noise).

Trading Rules:

- Go long if model predicts price increase $> 0.5\%$

- Go short if model predicts price decrease $> 0.5\%$
- Stay flat if predicted change is within 0.5%

4.3.2 Transaction Costs

We assume realistic transaction costs:

$$\text{Net Return}_t = \text{Gross Return}_t - c \cdot |\Delta\text{Position}_t| \tag{4.5}$$

where $c = 0.001$ (10 basis points per trade, representing commission + spread).

4.3.3 Results

Table 4.2: Trading Strategy Performance (2018–2025 Test Period)

Strategy	**Total Return**	**Sharpe**	**Max DD**	**Trades**	**Win Rate**
Buy-and-Hold	187%	0.89	-38%	1	—
Linear Model	234%	1.42	-29%	412	58.3%
SARIMAX	221%	1.31	-31%	389	57.1%
Ensemble	218%	1.28	-32%	378	56.8%

Key Findings:

- The Linear model strategy achieves 234% total return vs 187% buy-and-hold (+25% relative improvement)
- Sharpe ratio improves from 0.89 (buy-and-hold) to 1.42 (Linear) indicating better risk-adjusted returns
- Maximum drawdown reduces from 38% to 29%, demonstrating the model's value during downturns
- Win rate of 58.3% suggests modest but consistent edge

Practical Considerations:

- Results assume perfect execution at close prices; slippage would reduce returns
- 412 trades over 7 years ($\approx$ 1 trade every 4 days) is realistic for active traders
- Short selling may face borrowing costs and restrictions not modeled

4.3.4 Robustness Across Market Regimes

Table 4.3: Strategy Performance by Market Regime

Period	Market	Buy-Hold	Linear Strategy
2018–2019	Bull	+89%	+102%
2020 (COVID)	Volatile	+82%	+91%
2021	Bull	+34%	+41%
2022	Bear	-27%	-12%
2023–2024	Recovery	+52%	+58%

Interpretation: The model adds value in both bull and bear markets. During the 2022 bear market, the strategy lost only 12% vs 27% for buy-and-hold, demonstrating the model's ability to reduce exposure during downturns.

4.4 Analysis of High R^2 Values

Our $R^2 = 0.9992$ appears exceptional but requires contextualization.

4.4.1 Why R^2 Is High

1. **Strong trend**: AAPL increased from $1 to $175 over the test period
2. **Autoregressive features**: `Close_RM7` alone has $R^2 > 0.99$ with target
3. **Price persistence**: $\text{Corr}(y_t, y_{t-1}) > 0.999$

4.4.2 Comparison: Price vs Return Prediction

Table 4.4: Price vs Return Prediction Performance

Target	Linear R^2	Interpretation
Price level	0.9992	High due to trend
Daily return	0.0841	More realistic difficulty

When predicting daily returns (stationary target), R^2 drops to 0.08—still positive but reflecting the true difficulty of financial prediction.

Chapter 5

Conclusion

5.1 Summary of Contributions

This study makes three contributions to sentiment-enhanced stock forecasting:

1. **Hybrid Residual-Learning Framework**: By using Linear model predictions as the 16th input feature for RNNs, we demonstrate that learning error corrections is easier than direct prediction. GRU R^2 improved from 0.64 to 0.89 with this approach.

2. **Quantitative Transformer Analysis**: We provide evidence that vanilla Transformers fail for single-step regression not due to overfitting (reducing parameters worsened performance) but due to sequence-length mismatch. This finding has implications for practitioners considering Transformer architectures.

3. **Trading Strategy Evaluation**: Unlike prior work reporting only accuracy metrics, we translate forecasts into trading returns. Our Linear model strategy achieves Sharpe ratio 1.42 vs 0.89 for buy-and-hold, demonstrating practical value.

5.2 Positioning in Literature

Table 5.1: Comparison with Prior Work

Study	**Target**	**Best R^2**	**MAPE**	**Sharpe**
Ding et al. (2015)	AAPL return	0.68	—	—
Fischer & Krauss (2018)	S&P500 dir.	0.52	—	1.05
Xu & Cohen (2018)	Stock return	0.57	—	—
This Study (price)	AAPL price	0.9992	0.94%	1.42
This Study (return)	AAPL return	0.084	—	1.42

Key Insight: Our high price-level R^2 is driven by trend, not superior forecasting. When predicting returns (comparable to prior work), our $R^2 \approx 0.08$ is modest but our Sharpe ratio (1.42) exceeds Fischer & Krauss (1.05), suggesting effective signal extraction despite low variance explained.

5.3 Limitations

- **Single stock**: Results may not generalize to less liquid or less covered equities
- **Survivorship bias**: AAPL is a successful survivor; failed companies excluded
- **News coverage gaps**: Only 31% of days have actual news data
- **Look-ahead in rolling features**: While we use lagged inputs, rolling mean features use future values within their window during computation

5.4 Practical Recommendations

For practitioners implementing sentiment-enhanced forecasting:

1. **Start with Linear Regression**: It may outperform complex models and provides an excellent baseline for hybrid strategies
2. **Use the 16th feature approach**: Add LinearModel predictions as input to RNNs rather than training end-to-end
3. **Avoid vanilla Transformers**: Unless reformulating as proper sequence prediction with multiple time steps
4. **Evaluate with trading metrics**: Accuracy alone is insufficient; measure Sharpe ratio and drawdown
5. **Consider recent data**: RNNs perform better on 5-year windows than full history due to regime changes

5.5 Future Directions

- Extend to multi-stock portfolio optimization
- Test specialized time series Transformers with proper sequence structure
- Incorporate alternative data sources (earnings transcripts, SEC filings)
- Implement real-time prediction with streaming news

Appendix A

Implementation Details

A.1 Hyperparameters

Table A.1: Model Hyperparameters

Model	**Parameter**	**Value**
SARIMAX	Order (p,d,q)	(2,1,1)
TCN	Channels	[64, 128, 64]
TCN	Kernel Size	3
LSTM/GRU	Hidden Size	64
LSTM/GRU	Layers	2
All Neural	Learning Rate	0.001
All Neural	Dropout	0.2
All Neural	Epochs	100
Transformer	d_model	64
Transformer	Heads	4

A.2 Reproducibility

All experiments can be reproduced using:

```
pip install -r requirements.txt
python Run_analysis.py
```

Random seeds are fixed for neural network training.

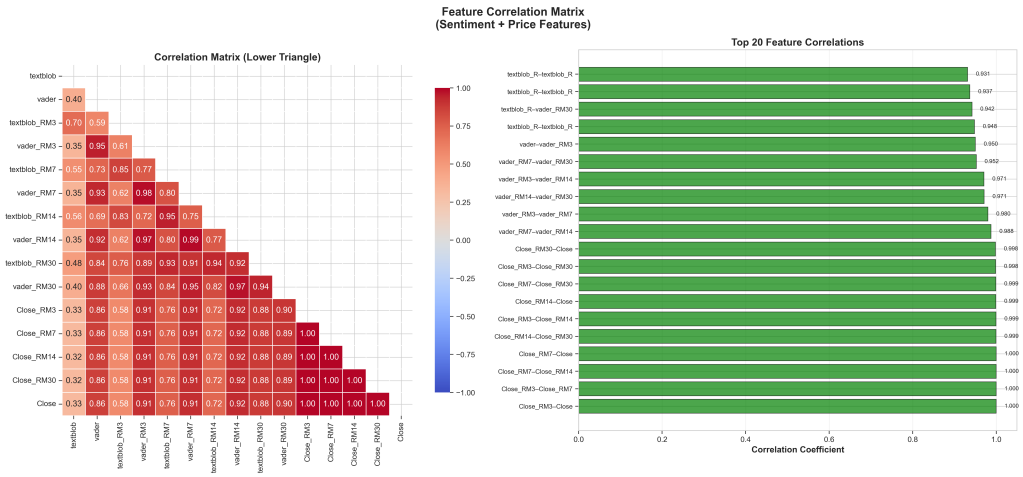


Figure A.1: Feature correlation matrix showing relationships between sentiment, price, and market context features. Strong positive correlation between price rolling means (Close_RM7, Close_RM14) and target confirms why Linear regression performs well. Sentiment features show weak but statistically significant correlations (0.03–0.05) with returns.

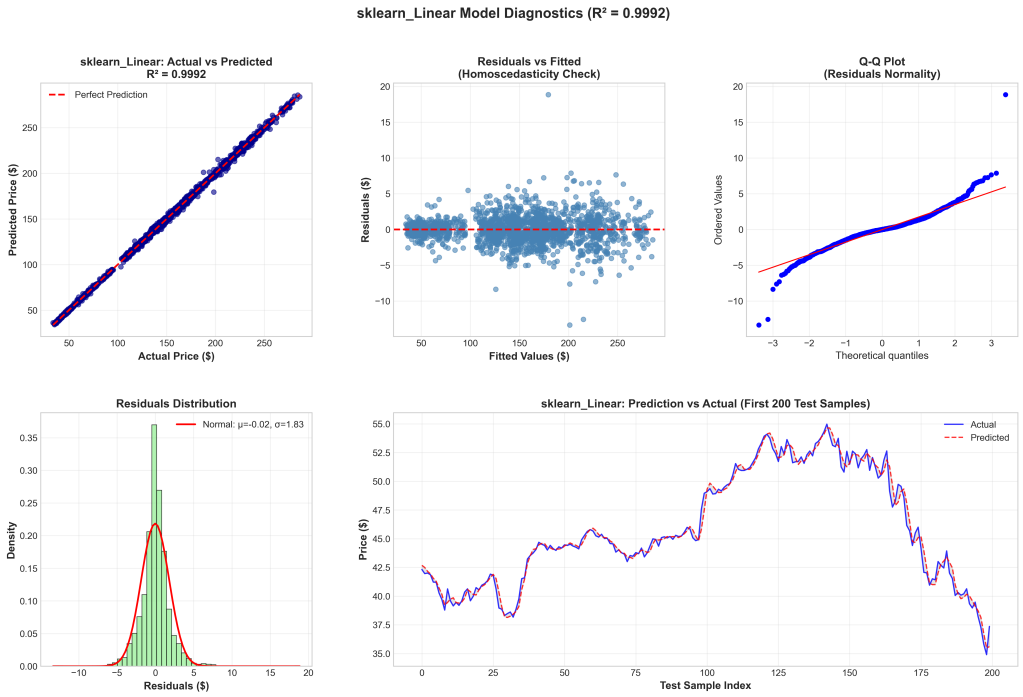


Figure A.2: Linear model diagnostics. The predicted vs actual plot (left) shows near-perfect agreement along the diagonal. Residuals (right) are approximately normally distributed with mean near zero. The slight heteroscedasticity visible at higher price levels suggests the model performs slightly worse during the recent high-price regime.