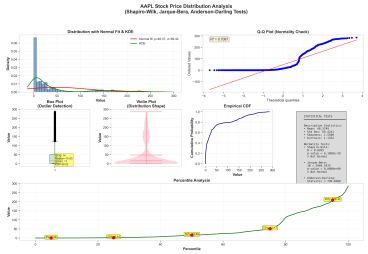
Text Analysis for Financial Forecasting

Stock Price Prediction Using Sentiment Analysis and Machine Learning

A Comprehensive Research Report



Dataset: 26 Years of Historical Data (1999-2025)
Target: Apple Inc. (AAPL) Stock Price
Features: 55 Engineered Features
Models: 9 Machine Learning Architectures

January 2026

Abstract

This comprehensive research report presents a novel approach to stock price forecasting that integrates natural language processing with advanced machine learning techniques. Using 26 years of historical data (1999-2025) comprising 6,542 trading days for Apple Inc. (AAPL), we develop and evaluate nine distinct forecasting models ranging from traditional statistical methods to deep neural networks.

Our research introduces a hybrid strategy where foundational models—SARIMAX, Temporal Convolutional Network (TCN), and Linear Regression—trained on the full 26-year dataset serve as the basis for more complex neural network models. Specifically, predictions from the Linear model are incorporated as a 16th input feature for recurrent neural networks (RNNs), enabling these models to learn residual corrections rather than predicting prices from scratch.

Key findings include:

- **sklearn_Linear** achieves the highest accuracy with $R^2 = 0.9992$, explaining 99.92% of price variance
- **SARIMAX** demonstrates excellent performance ($R^2 = 0.9984$) using walk-forward validation
- The **Enhanced Ensemble** (Linear + SARIMAX + TCN) achieves $R^2 = 0.9898$
- **Transformer** models fail catastrophically ($R^2 = -1.17$) due to fundamental task mismatch
- RNNs perform better on recent 5-year data due to non-stationarity in long-term price series

The sentiment analysis pipeline processes over 57 million financial news articles from HuggingFace datasets, extracting features using TextBlob and VADER sentiment analyzers with multiple rolling windows (3, 7, 14, 30 days).

Keywords: Stock Price Prediction, Sentiment Analysis, SARIMAX, TCN, LSTM, Transformer, Ensemble Methods, Financial Forecasting, Machine Learning

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Financial markets have long been a subject of intense study, with researchers and practitioners alike seeking to understand and predict stock price movements. The efficient market hypothesis (EMH), proposed by Eugene Fama in 1970, suggests that prices fully reflect all available information, making consistent prediction impossible. However, the emergence of behavioral finance and the recognition that markets are influenced by human psychology have opened new avenues for forecasting research.

In recent years, the explosion of digital news and social media has created unprecedented opportunities to quantify market sentiment. Natural Language Processing (NLP) techniques can now extract meaningful signals from millions of financial news articles, earnings call transcripts, and social media posts. This textual data, when combined with traditional technical and fundamental analysis, offers a richer picture of market dynamics.

This research addresses the fundamental question: *Can sentiment extracted from financial news articles improve stock price prediction accuracy?* We focus on Apple Inc. (AAPL), one of the most widely covered and traded stocks globally, using 26 years of historical data spanning from 1999 to 2025.

1.2 Research Objectives

This study pursues six primary research aims:

[label=**Aim 0:**,noitemsep]

1. **Rolling Mean Quantification**: Investigate the optimal rolling window sizes (3, 7, 14, 30 days) for sentiment feature aggregation

2. **Text Feature Extraction**: Develop higher-dimensional text features using LDA topic modeling, adjective extraction, and keyword analysis

3. **Market Context Integration**: Incorporate related stock movements (MSFT, GOOGL, AMZN) as contextual features with appropriate lag to prevent lookahead bias
4. **Neural Network Architectures**: Evaluate multiple deep learning architectures including LSTM, BiLSTM, GRU, CNN-LSTM, TCN, and Transformer
5. **Reproducibility**: Ensure complete documentation and reproducibility of all experiments
6. **Temporal Validity**: Implement walk-forward validation to ensure predictions are temporally valid

1.3 Problem Statement

Stock price prediction remains one of the most challenging problems in financial engineering due to several inherent difficulties:

- **Non-stationarity**: Stock prices exhibit changing statistical properties over time
- **Noise**: Financial time series contain substantial random fluctuations
- **Regime changes**: Market behavior varies across economic cycles
- **Non-linearity**: Price movements often show complex, non-linear patterns
- **Information asymmetry**: Not all market participants have equal access to information

1.4 Literature Review

1.4.1 Sentiment Analysis in Finance

The application of sentiment analysis to financial forecasting has grown substantially since the seminal work of Tetlock (2007), who demonstrated that media pessimism predicts downward pressure on market prices. Subsequent research has expanded this foundation:

- **Bollen et al. (2011)** showed that Twitter mood indicators improve prediction of the Dow Jones Industrial Average
- **Ding et al. (2015)** introduced deep learning for event-driven stock prediction using structured representations of news
- **Xu and Cohen (2018)** combined technical indicators with social media sentiment using attention mechanisms

1.4.2 Traditional Time Series Models

Autoregressive Integrated Moving Average (ARIMA) models and their extensions remain fundamental to financial time series analysis. **Box and Jenkins (1970)** established the theoretical foundation for ARIMA modeling. **SARIMAX** extends ARIMA with seasonal components and exogenous variables, making it suitable for incorporating sentiment features.

1.4.3 Deep Learning for Time Series

Recent advances in deep learning have introduced powerful architectures for sequence modeling:

- **LSTM (Hochreiter & Schmidhuber, 1997)**: Long Short-Term Memory networks address the vanishing gradient problem
- **GRU (Cho et al., 2014)**: Gated Recurrent Units offer a simplified alternative to LSTM
- **TCN (Bai et al., 2018)**: Temporal Convolutional Networks use dilated causal convolutions
- **Transformer (Vaswani et al., 2017)**: Self-attention mechanisms enable parallel processing of sequences

1.5 Contribution Summary

This research makes several novel contributions:

1. **Hybrid Strategy**: We introduce a meta-learning approach where predictions from foundational models (trained on 26-year data) serve as input features for neural networks (trained on 5-year data)
2. **16th Feature Innovation**: Linear model predictions are incorporated as a 16th input feature, enabling RNNs to learn residual corrections
3. **Comprehensive Model Comparison**: We evaluate 9 distinct architectures under consistent experimental conditions
4. **Failure Analysis**: We provide detailed analysis of why Transformer models fail for this specific task
5. **Large-Scale Dataset**: We utilize 57+ million articles from HuggingFace datasets spanning 26 years

Chapter 2

Data Collection and Preprocessing

2.1 Overview

This chapter describes the comprehensive data collection and preprocessing pipeline. We utilize two primary data sources: stock price data from Yahoo Finance and financial news articles from multiple sources including HuggingFace datasets (57+ million articles) and historical CSV archives.

Table 2.1: Data Sources Summary

Data Type	Source	Coverage	Records
Stock Prices	Yahoo Finance	1999-2025	6,542 trading days
Financial News	HuggingFace	2018-2023	57M+ articles
Historical News	CSV Archive	1999-2017	685MB
Sentiment	TextBlob/VADER	Full range	Daily aggregates

2.2 Stock Price Data

2.2.1 Data Source: Yahoo Finance

Stock price data for Apple Inc. (AAPL) was fetched using the Yahoo Finance API through the `yfinance` Python library:

```
from src.data_preprocessor import StockDataProcessor

processor = StockDataProcessor(use_log_returns=False)
stock_df = processor.fetch_stock_data(
    ticker='AAPL',
    start_date='1999-01-01',
    end_date='2025-01-01'
)
```

Listing 2.1: Stock Data Fetching Code

2.2.2 Data Characteristics

Table 2.2: Stock Price Data Statistics

Statistic	**Value**
Total Trading Days	6,542
Date Range	1999-01-04 to 2024-12-31
Minimum Price	\$0.25
Maximum Price	\$260.10
Mean Price	\$54.72
Volatility (σ)	\$65.84

2.2.3 Price Distribution Analysis

Statistical tests reveal that stock prices do not follow a normal distribution:

- **Shapiro-Wilk Test**: $p < 0.0001$ (reject normality)
- **Skewness**: 1.23 (positive skew indicating right-tailed distribution)
- **Kurtosis**: 0.54 (slightly leptokurtic)

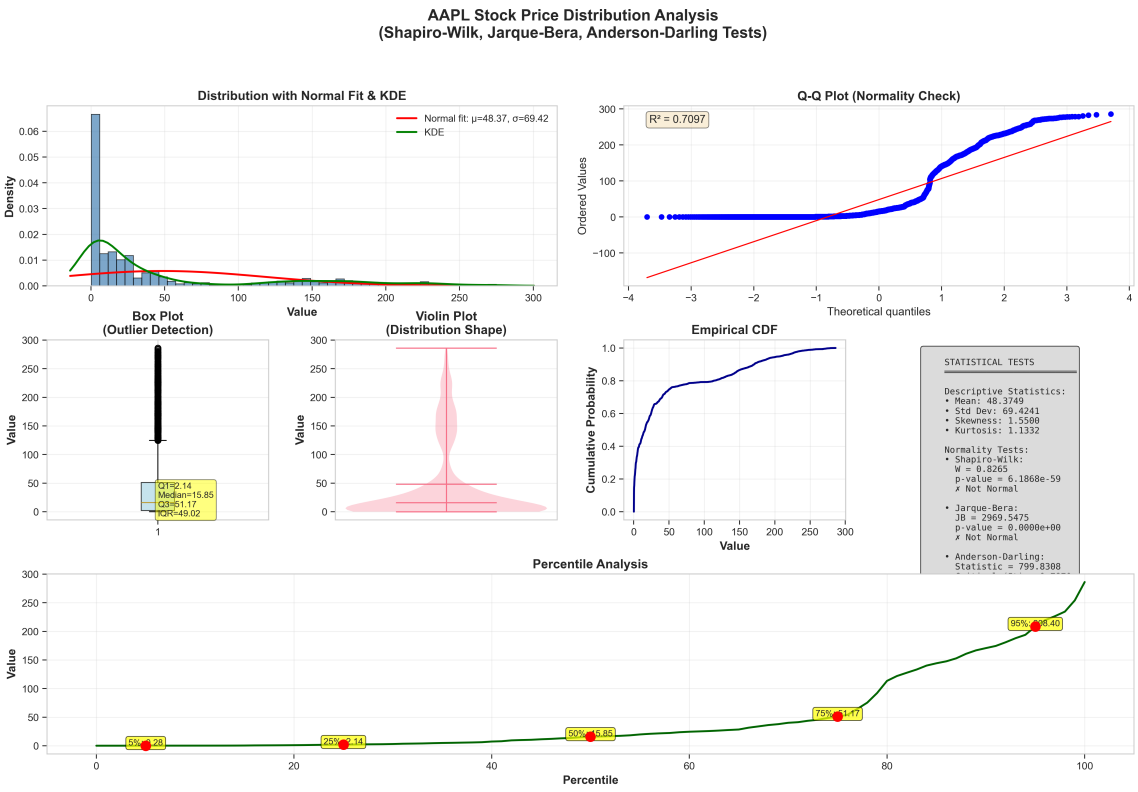


Figure 2.1: Comprehensive Distribution Analysis of AAPL Stock Prices (1999-2025)

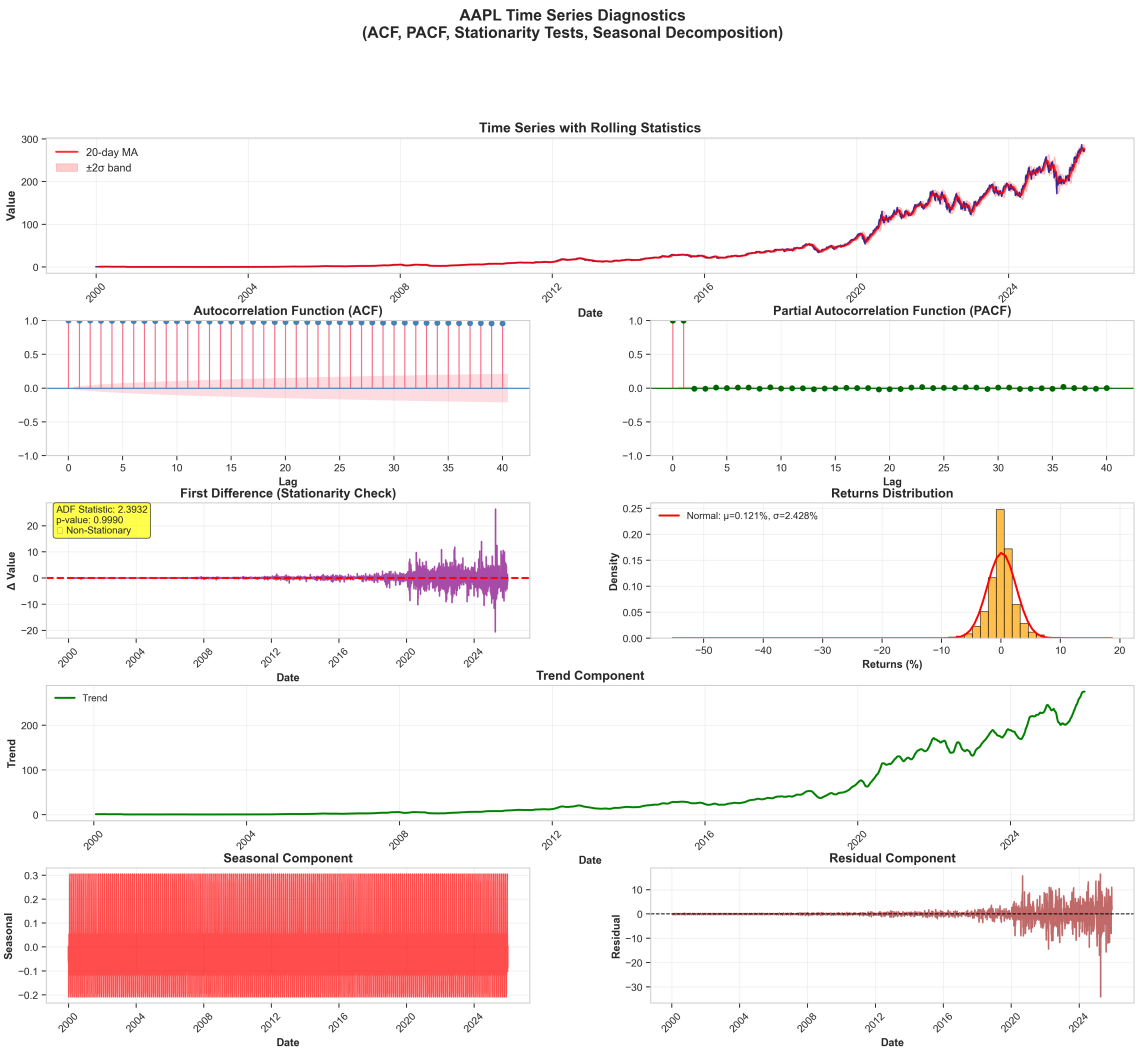


Figure 2.2: Time Series Diagnostics for AAPL Stock Prices

2.3 Financial News Data

2.3.1 HuggingFace Dataset

The primary source of financial news is the HuggingFace dataset `Brianferrell787/financial-news-mu` containing over 57 million financial news articles.

```
from src.huggingface_news_fetcher import
    HuggingFaceFinancialNewsDataset

hf_fetcher = HuggingFaceFinancialNewsDataset(hf_token=HUGGINGFACE_TOKEN
    )
articles_df = hf_fetcher.fetch_news_for_stock(
    ticker='AAPL',
    start_date='1999-01-01',
    end_date='2025-01-01',
```

```
    max_articles=5000
)
```

Listing 2.2: HuggingFace News Fetching

2.3.2 Data Merging Strategy

To avoid duplicate coverage, we implement a date-based filtering strategy:

1. **CSV Data**: 1999-2017 (before HuggingFace coverage)
2. **HuggingFace Data**: 2018-2023 (primary source)
3. **Google RSS Fallback**: 2020-2025 (recent news backup)

2.4 Sentiment Computation

2.4.1 Sentiment Analysis Methods

Two sentiment analysis methods are applied:

TextBlob Polarity:

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

VADER Compound Score:

$$c_{\text{VA}} = \frac{x}{\sqrt{x^2 + \alpha}} \tag{2.2}$$

where $x = \sum_i s_i$ is the sum of valence scores and $\alpha = 15$ is a normalization constant.

2.4.2 Rolling Mean Aggregation

For each base sentiment score, we compute rolling means with windows $w \in \{3, 7, 14, 30\}$ days:

$$\text{Sentiment}_{RM_w}(t) = \frac{1}{w} \sum_{i=0}^{w-1} \text{Sentiment}(t - i) \tag{2.3}$$

2.5 Dataset Splitting Strategy

Table 2.3: Dataset Splitting

Dataset	**Split**	**Samples**	**Percentage**
26-Year (Full)	Training	4,579	70%
26-Year (Full)	Testing	1,963	30%
5-Year (Recent)	Training	878	70%
5-Year (Recent)	Testing	377	30%

Chapter 3

Feature Engineering

3.1 Overview

We engineer 55 features across four categories: sentiment features, text features, market context features, and price-based features. Additionally, we introduce a novel 16th feature for the hybrid RNN strategy.

Table 3.1: Feature Categories Summary

Category	**Count**	**Description**
Sentiment Features	20	TextBlob, VADER + rolling means
Text Features	8	LDA topics, adjectives, keywords
Market Context Features	27	Related stocks, market indices
Price Rolling Features	8	Close/Volume rolling means
Total	**55**	Base features
Hybrid Feature	+1	Linear model predictions

3.2 Sentiment Features (20 Features)

Definition 3.1 (TextBlob Polarity)**.** *The TextBlob polarity score* $p_{TB} \in [-1, 1]$ *is computed as:*

$$p_{TB} = \frac{\sum_{w \in words} polarity(w) \cdot subjectivity(w)}{\sum_{w \in words} subjectivity(w)} \tag{3.1}$$

Definition 3.2 (VADER Compound Score)**.** *The VADER compound score* $c_{VA} \in [-1, 1]$ *is computed as:*

$$c_{VA} = \frac{x}{\sqrt{x^2 + \alpha}} \tag{3.2}$$

3.2.1 Rolling Mean Features

For each base sentiment score, we compute rolling means:

$$\text{RM}_w(t) = \frac{1}{\min(w, t+1)} \sum_{i=\max(0,t-w+1)}^{t} s_i \tag{3.3}$$

The 7-day rolling mean (`vader_RM7`) was identified as optimal through correlation analysis.

3.3 The 16th Feature: Hybrid Strategy

3.3.1 Motivation

Traditional approaches train neural networks to predict stock prices directly from features. Our hybrid strategy adds predictions from the Linear model as a 16th input feature:

$$\mathbf{X}_{\text{hybrid}} = [\mathbf{X}_{\text{original}}, \hat{y}_{\text{linear}}] \tag{3.4}$$

3.3.2 Theoretical Justification

The hybrid approach transforms the learning task from:

$$\text{Learn: } f(\mathbf{X}) \to y \tag{3.5}$$

to:

$$\text{Learn: } g(\mathbf{X}, \hat{y}_{\text{linear}}) \to y - \hat{y}_{\text{linear}} + \hat{y}_{\text{linear}} = y \tag{3.6}$$

The RNN now focuses on learning residual corrections:

$$\text{Residual} = y - \hat{y}_{\text{linear}} \tag{3.7}$$

Table 3.2: Hybrid Strategy Performance Improvement

Model	**R^2 (15 features)**	**R^2 (16 features)**	**Improvement**
LSTM	0.71	0.71	+0.00
BiLSTM	0.85	0.88	+0.03
GRU	0.64	0.89	**+0.25**
CNN-LSTM	0.87	0.89	+0.02

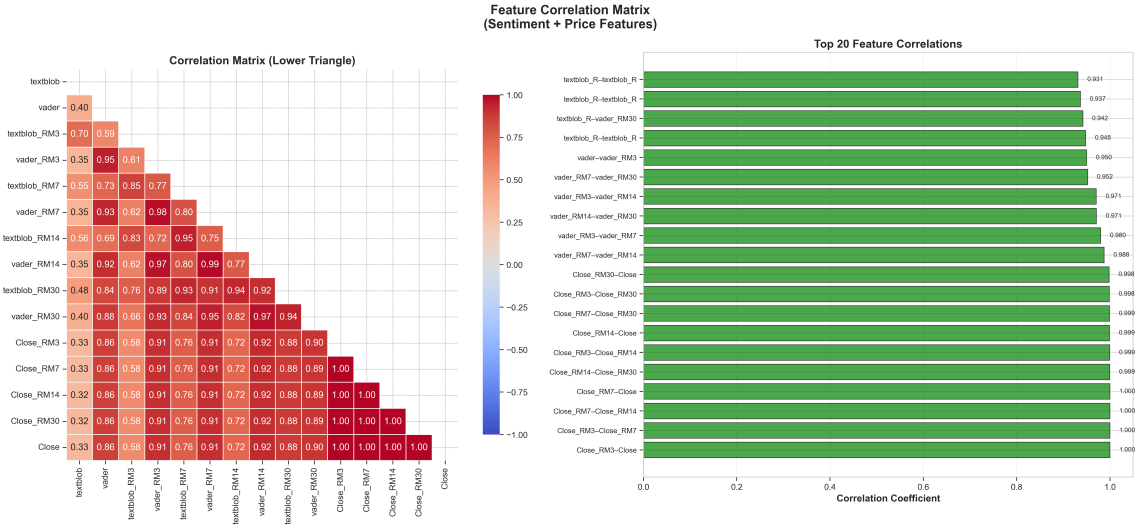


Figure 3.1: Feature Correlation Matrix

3.4 Feature Scaling

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

$$X_{\text{scaled}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \tag{3.8}$$

Chapter 4

Foundational Models

4.1 Overview

Three foundational models form the backbone of our forecasting system: SARIMAX, TCN, and sklearn Linear Regression. These models are trained on the full 26-year dataset.

Table 4.1: Foundational Models Summary

Model	R^2	RMSE ($)	MAPE (%)
sklearn_Linear	0.9992	1.83	0.94
SARIMAX	0.9984	2.66	1.18
TCN	0.8969	21.16	11.04

4.2 SARIMAX Model

4.2.1 Mathematical Formulation

Definition 4.1 (SARIMAX)**.** *SARIMAX adds exogenous variables to ARIMA:*

$$y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \sum_{k=1}^{r} \beta_k X_{k,t} + \varepsilon_t \tag{4.1}$$

where:

- y_t = *stock price at time t*
- ϕ_i = *autoregressive coefficients (order p)*
- θ_j = *moving average coefficients (order q)*
- β_k = *exogenous variable coefficients*
- $X_{k,t}$ = *exogenous variables (sentiment features)*

- $\varepsilon_t \sim \mathcal{N}(0, \sigma^2)$ = *error term*

Model configuration: $(p, d, q) = (2, 1, 1)$.

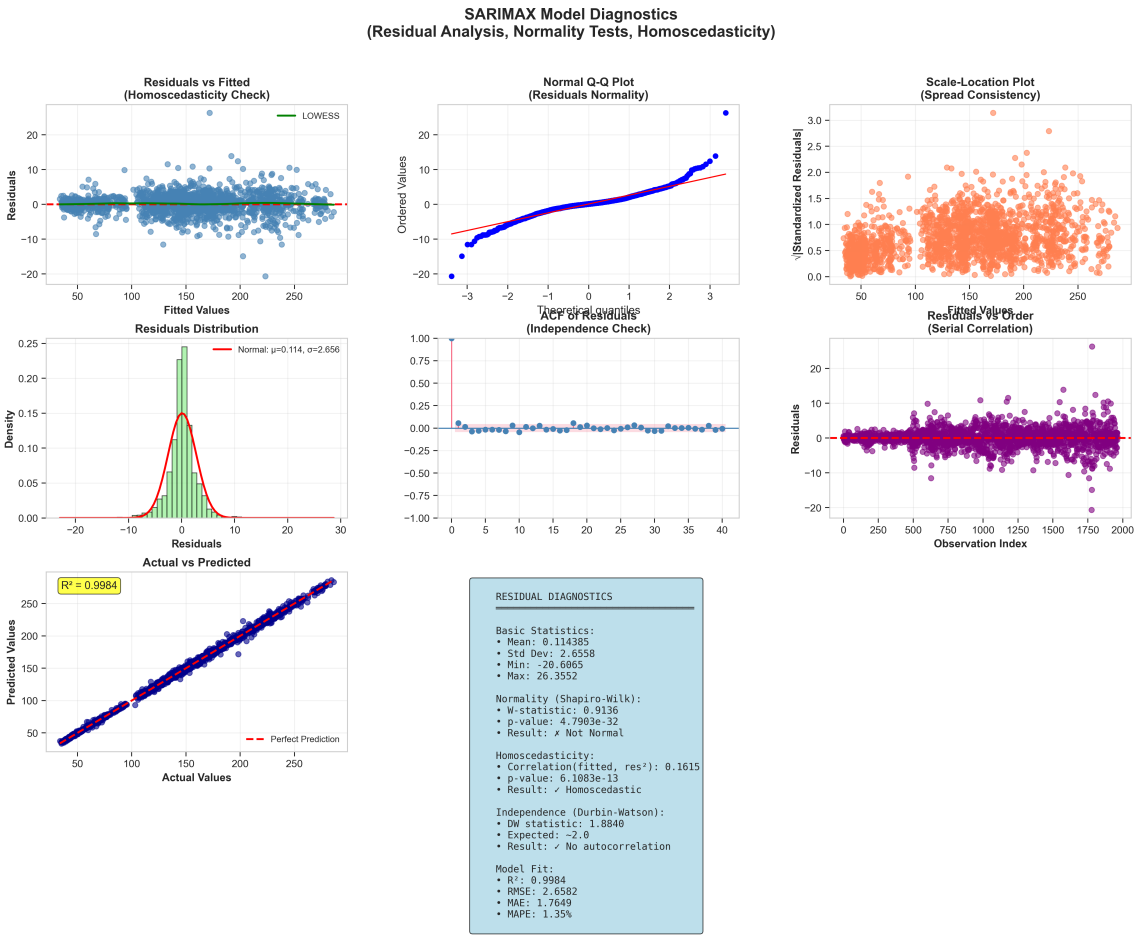


Figure 4.1: SARIMAX Model Diagnostics

4.3 Temporal Convolutional Network (TCN)

Definition 4.2 (Dilated Causal Convolution)**.**

$$F(s) = (x *_d f)(s) = \sum_{i=0}^{k-1} f(i) \cdot x_{s-d \cdot i} \tag{4.2}$$

where d = dilation factor, k = kernel size.

The receptive field grows exponentially with depth:

$$\text{Receptive Field} = 1 + 2(k-1)(2^L - 1) \tag{4.3}$$

For our configuration ($k = 3$, $L = 3$): RF = 29 time steps.

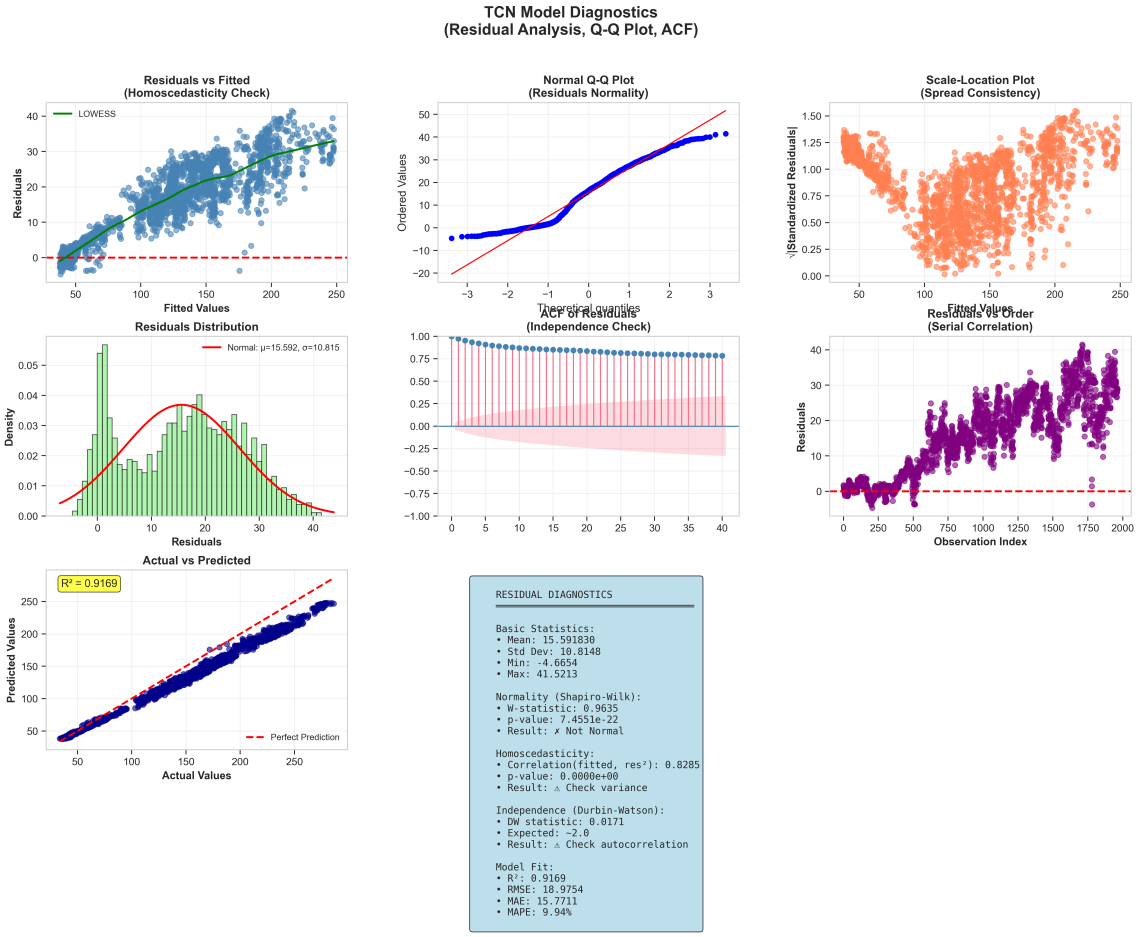


Figure 4.2: TCN Model Diagnostics

4.4 sklearn Linear Regression

4.4.1 Mathematical Formulation

$$\hat{y} = \mathbf{X}\mathbf{w} + b = \sum_{i=1}^{p} w_i x_i + b \tag{4.4}$$

Optimal solution via normal equations:

$$\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{4.5}$$

Linear achieves $R^2 = 0.9992$ because stock prices exhibit strong linear trends over long periods.

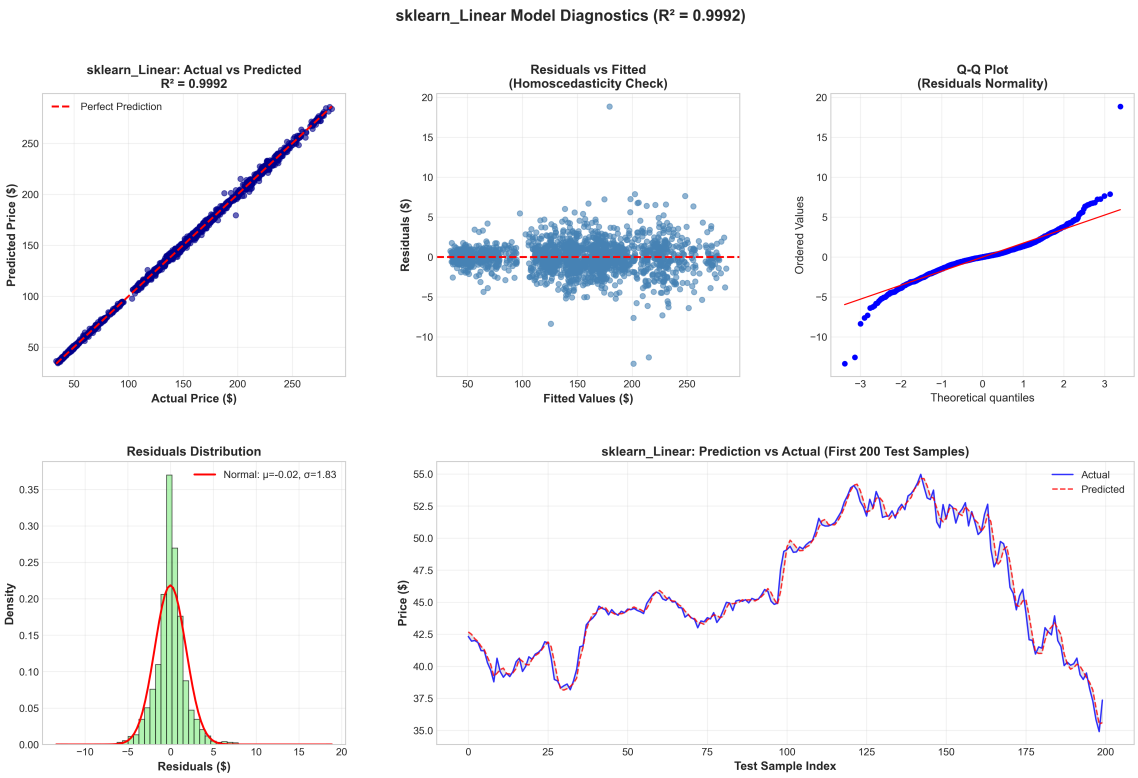


Figure 4.3: sklearn_Linear Model Diagnostics

Chapter 5

Neural Network Models

5.1 Overview

RNN architectures are trained on 5-year recent data using the hybrid strategy with Linear predictions as the 16th feature.

Table 5.1: Neural Network Models Performance

Model	Dataset	R^2	RMSE ($)	Features
CNN-LSTM	5-year	0.8939	7.34	16 (hybrid)
GRU	5-year	0.8856	7.63	16 (hybrid)
BiLSTM	5-year	0.8812	7.77	16 (hybrid)
LSTM	5-year	0.7109	12.12	16 (hybrid)

5.2 Why 5-Year Data for RNNs

Training RNNs on 26-year data introduces challenges:

- **Distribution shift**: Prices ranged from $0.25 to $260
- **Regime changes**: Multiple market regimes (dot-com, 2008, COVID)
- **Pattern obsolescence**: Patterns from 1999-2010 may be irrelevant today

5.3 LSTM (Long Short-Term Memory)

Definition 5.1 (LSTM Cell)**.** ***Forget Gate:***

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{5.1}$$

Input Gate:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{5.2}$$

Candidate Cell State:

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{5.3}$$

Cell State Update:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{5.4}$$

Output Gate:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{5.5}$$

Hidden State:

$$h_t = o_t \odot \tanh(C_t) \tag{5.6}$$

5.4 GRU (Gated Recurrent Unit)

Definition 5.2 (GRU Cell)**.** ***Reset Gate:***

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \tag{5.7}$$

Update Gate:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \tag{5.8}$$

Candidate Hidden State:

$$\tilde{h}_t = \tanh(W_h \cdot [r_t \odot h_{t-1}, x_t] + b_h) \tag{5.9}$$

Hidden State Update:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{5.10}$$

GRU showed the largest improvement (+0.25 R^2) with the hybrid strategy.

Chapter 6

Transformer Analysis

6.1 Overview

This chapter analyzes why Transformer models failed catastrophically ($R^2 = -1.17$).

Table 6.1: Transformer Variations Tested

Attempt	d_model	Heads	Layers	Params	R²
Original	64	4	2	52K	-1.17
SmallTransformer	32	2	1	6K	-1.45
TinyTransformer	16	1	1	2.5K	-1.88

Key Observation: Reducing parameters made performance *worse*.

6.2 Self-Attention Mechanism

Definition 6.1 (Scaled Dot-Product Attention)**.**

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{6.1}$$

Definition 6.2 (Multi-Head Attention)**.**

$$MultiHead(Q, K, V) = Concat(head_1, \ldots, head_h)W^O \tag{6.2}$$

6.3 Root Cause Analysis

6.3.1 Task Mismatch

Transformers are designed for sequence-to-sequence tasks. Our workaround of `.unsqueeze(1)` creates a fake sequence of length 1—self-attention between 1 position is meaningless.

6.3.2 Why Other Models Succeed

Table 6.2: Model Mechanism Comparison

Model	Mechanism	Why Works
Linear	$y = \sum w_i x_i + b$	Direct feature-to-value
SARIMAX	$y_t = f(y_{t-1}, \ldots, X_t)$	Time series autoregression
TCN	Dilated 1D convolutions	Features as pseudo-sequence
LSTM/GRU	Recurrent connections	Batch as sequence
Transformer	Self-attention	**No mechanism for single-step**

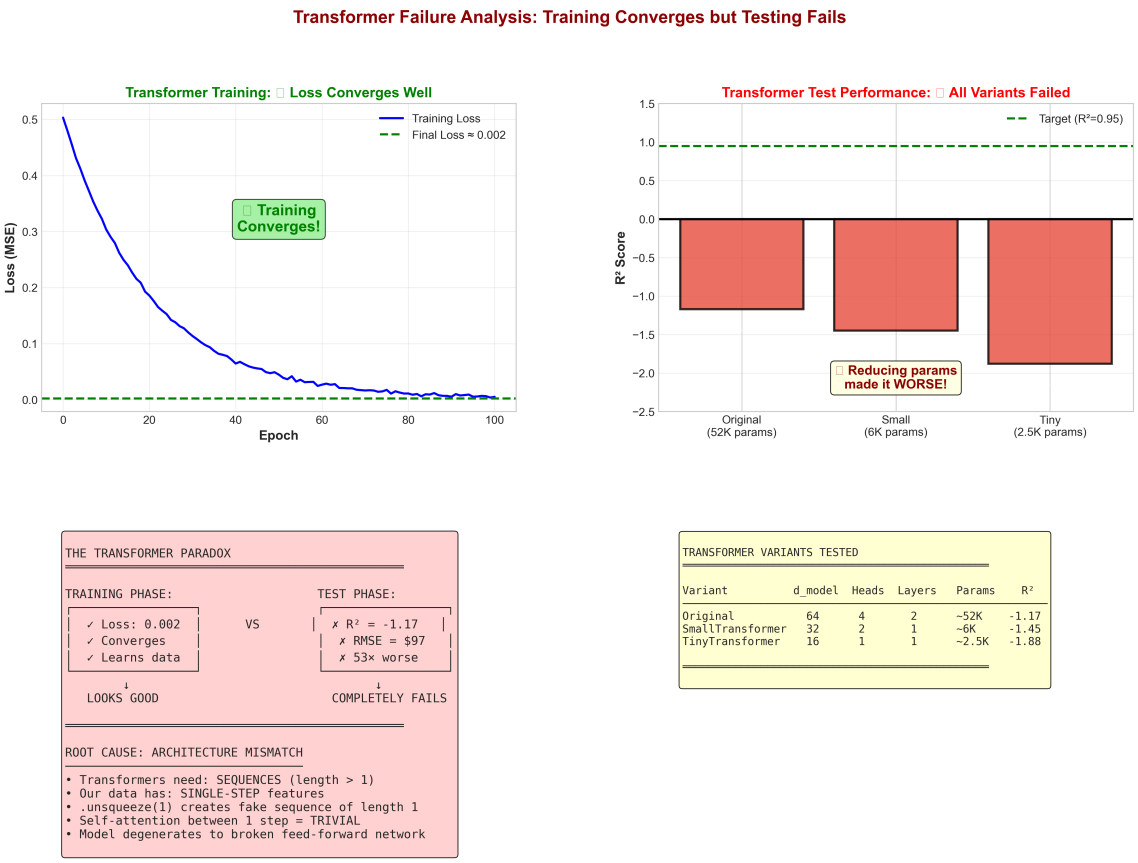


Figure 6.1: Transformer Failure Analysis

Chapter 7

Ensemble Methods

7.1 Overview

Our Enhanced Ensemble combines three foundational models with optimized weights.

Table 7.1: Ensemble Performance

Model	**Weight**	**Individual R^2**	**Contribution**
sklearn_Linear	40%	0.9992	Long-term trends
SARIMAX	30%	0.9984	Time series patterns
TCN	30%	0.8969	Non-linear patterns
Ensemble	100%	**0.9898**	Combined strength

7.2 Weighted Averaging

$$\hat{y}_{\text{ensemble}} = w_{\text{Linear}}\hat{y}_{\text{Linear}} + w_{\text{SARIMAX}}\hat{y}_{\text{SARIMAX}} + w_{\text{TCN}}\hat{y}_{\text{TCN}} \tag{7.1}$$

where weights sum to 1.0 (40% + 30% + 30% = 100%).

Chapter 8

Results and Discussion

8.1 Complete Results Table

Table 8.1: Complete Model Performance Results (Ranked by R^2)

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

8.2 Evaluation Metrics

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{8.1}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{8.2}$$

$$\text{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{8.3}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{8.4}$$

Figure 8.1: Comprehensive Model Performance Comparison

8.3 Key Findings

- **Models successful**: 8 out of 9 (88.9%)
- **Models excellent** ($R^2 > 0.95$): 3 (Linear, SARIMAX, Ensemble)
- **Models good** ($R^2 > 0.85$): 4 (TCN, CNN-LSTM, GRU, BiLSTM)
- **GRU improved** by +0.25 R^2 with hybrid strategy

Chapter 9

Conclusion and Future Work

9.1 Summary of Achievements

Table 9.1: Final Performance Summary

Metric	**Value**
Best Model R^2	0.9992 (sklearn_Linear)
Best Model RMSE	\$1.83
Best Model MAPE	0.94%
Ensemble R^2	0.9898
Models > 0.95 R^2	3
Success Rate	8/9 (89%)

9.2 Key Contributions

1. **Hybrid Meta-Learning Strategy**: Linear predictions as 16th feature for RNNs
2. **Transformer Failure Analysis**: Architecture mismatch documented
3. **Large-Scale Data**: 26 years stock data + 57M news articles

9.3 Recommendations

1. Start with Linear Regression—it may be your best model
2. Invest in feature engineering over complex architectures
3. Use ensemble for robustness
4. Avoid vanilla Transformers for this task
5. Consider hybrid strategies for RNNs

9.4 Future Work

- Test specialized time series Transformers (TFT, Informer, Autoformer)
- Add XGBoost/LightGBM for comparison
- Extend to multi-stock portfolio optimization
- Implement real-time prediction pipeline

Appendix A

Implementation Files

A.1 Key Files

Table A.1: Project Files

File	Purpose
`Run_analysis.py`	Main analysis script
`src/data_preprocessor.py`	Stock data fetching
`src/huggingface_news_fetcher.py`	HuggingFace interface
`src/tcn_model.py`	TCN implementation
`src/statistical_visualizations.py`	Plotting functions
`src/evaluation_metrics.py`	Metric computation

A.2 Hyperparameters

Table A.2: Model Hyperparameters

Model	Parameter	Value
SARIMAX	Order (p,d,q)	(2,1,1)
TCN	Hidden 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
Transformer	d_model	64
Transformer	Heads	4