LEX LOCALIS-JOURNAL OF LOCAL SELF-GOVERNMENT
ISSN:1581-5374 E-ISSN:1855-363X
VOL. 23, NO. S5 (2025)

ENHANCED LSTM MODELS FOR SHORT-HORIZON FORECASTING: FROM FINANCIAL TIME SERIES TO PUBLIC GOVERNANCE

Aadhitiya Singh[1], Arnav Mhatre[1], Krish Sachdev[1], Deven Bhole[2], Archana Lakhe[2], Jaykrishna Joshi[2]

[1, 2] Department of Data Science, Mukesh Patel School of Technology Management and Engineering, SVKM's NMIMS, Mumbai-400056, India

Abstract— This research paper evaluates the performance of an improved Long Short-Term Memory (LSTM) model in forecasting stock prices for Google and Microsoft over varying timeframes (1-year, 2-year, and 3-year). The improved LSTM model consistently outperforms the original model for Google, demonstrating significant reductions in Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) across all timeframes. In contrast, the results for Microsoft are mixed; the improved model exhibits substantial performance gains for the 1-year data but underperforms relative to the original model for the 2- year and 3-year periods. This discrepancy suggests potential overfitting or limitations of the improved model in capturing long- term trends for Microsoft. The findings highlight the importance of model adaptability when applied to different stocks and time horizons, offering insights into the strengths and challenges of using advanced LSTM architectures in financial forecasting. This paper also further demonstrates how the improved LSTM pipeline can be adapted to public-sector short-horizon forecasting problems (e.g., municipal property-tax receipts, short-term water demand, daily waste generation, and clinic throughput). We describe domain adaptations — such as domain-specific lag windows, GIS/spatial features, exogenous policy and weather indicators, and uncertainty quantification (prediction intervals) — that are required for responsible deployment in local self-government contexts. These extensions show the technique's operational value for municipal budgeting, resource allocation, and short-term operational decision-making, and motivate pilot deployments with co-designed evaluation metrics oriented to administrative use.

Keywords—Long Short-Term Memory Model, RSME, Time Frame, Financial Forecasting

I. INTRODUCTION

Long Short-Term Memory (LSTM), a specialized version of Recurrent Neural Networks (RNNs) that is used for modelling sequential information as they can catch long- range dependencies. While traditional RNNs [1] are good at processing temporal sequences, they have shown erratic convergence because of gradient magnitude dilemma that restrict their prowess to learn patterns over long-time frames. LSTM overcomes these issues by incorporating a memory cell and a gating mechanism, which enables its capacity to assimilate and judiciously modify information sequences. This structural advantage makes LSTM especially effective in tasks like time-series forecasting, natural language processing, and speech recognition, acoustic modelling of speech, protein secondary structure prediction, speech synthesis and video data [2].

The architecture of LSTM consists of three primary gates: the input gate, the forget gate, and the output gate, which regulate the flow of information within the network. The forget gate enacts a paramount portrayal in determining which information should be discarded, thereby aiding in efficient memory management. In contrast, the input gate regulates the addition of new data into the cell state. Meanwhile, the output gate is accountable for the ultimate output in conformity with adjusted cell state. Collectively, these elements enable Long Short-Term Memory (LSTM) networks to address the limitations of conventional Recurrent Neural Networks (RNNs), significantly improving their capacity to learn complex temporal relationships.

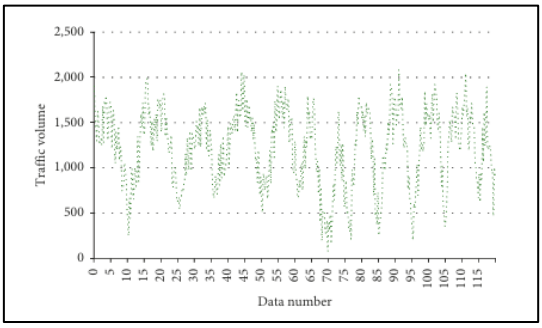

Fig.1 Predicted results of LSTM [1]

This paper will explore the applications of GIS in landscaping, highlighting its role in site analysis, planning, and implementation. We will examine how GIS facilitates the Interpretation of spatial relationships, identify suitable planting locations, and the potential growth and impact of vegetation over time. Furthermore, we will discuss the incorporation into GIS [5] with other advancements like remote sensing and 3D modeling to upgrade the precision and expedience of landscape design projects.

LSTM models offer advantages but encounter various obstacles, such as high computational requirements, extended training times, and increased sensitivity to hyperparameter selection. To elevate the resourcefulness and flexibility of Long Short-Term Memory (LSTM) networks, researchers have investigated a miscellany of optimization methodologies, including dropout regularization, adaptive learning rates, and modifications to the architecture. Furthermore, the emergence of Transformer-based models has created new challenges in the realm of sequential data modelling, leading to increased research efforts aimed at improving LSTM architectures for better performance.

LSTM models offer advantages but encounter various obstacles, such as high computational requirements, extended training times, and increased sensitivity to hyperparameter selection. To elevate the resourcefulness and flexibility of Long Short-Term Memory (LSTM) networks, researchers have investigated a miscellany of optimization methodologies, including dropout regularization, adaptive learning rates, and modifications to the architecture. Furthermore, the emergence of Transformer-based models has created new challenges in the realm of sequential data modelling, leading to increased research efforts aimed at improving LSTM architectures for better performance.

II. LITERATURE REVIEW

The paper focuses on optimizing traffic congestion management using intelligent transportation systems (ITS) in smart cities. It discusses various congestion control techniques, including adaptive traffic signal control, vehicle- to-infrastructure (V2I) communication, and data-driven approaches using artificial intelligence (AI) and machine learning (ML). The study emphasizes the role of real-time data analytics in enhancing traffic flow efficiency, reducing travel time, and minimizing fuel consumption. The authors explore different optimization algorithms, such as genetic algorithms and deep learning models, to improve traffic signal timing. The paper concludes that implementing a combination of AI-driven prediction models, IoT-enabled sensors, and decentralized traffic control can significantly reduce congestion levels and improve urban mobility.[1]

This paper presents an approach to stock market analysis using the Long Short-Term Memory (LSTM) model, a specialized type of recurrent neural network (RNN). The authors highlight the challenges of stock market prediction due to its non-linearity and volatility. The study explores how LSTM models can capture long-term dependencies in financial data and enhance prediction accuracy. Various corpus are used to build and

gauge the model, comparing it with traditional machine learning models. The findings suggest that LSTM outperforms conventional statistical models, providing more reliable predictions of stock price trends. The paper concludes that LSTM-based models are effective in financial time series forecasting, helping investors make informed decisions.[2]

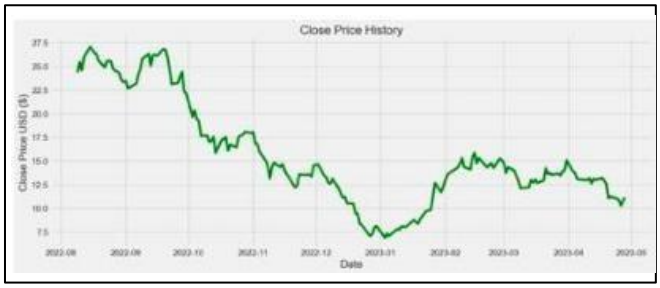

Fig.2 Close Price History [2]

This paper meticulously explores the sophisticated application of machine learning methodologies aimed at prognosticating outcomes for the future price movements of gold and Bitcoin, two significant financial assets. The authors analyze historical data and explore various influencing factors such as macroeconomic indicators, market sentiment, and global economic conditions. This research leveraged deep learning, including LSTM frameworks, to prognosticate price trends. The results indicate that ML-based models can effectively capture market patterns and offer reliable predictions. The paper highlights the benefits of the meticulous collation and distillation of multifarious inputs from disparate wellsprings of knowledge for improved forecasting accuracy. It concludes that AI-driven empirically-substantiated constructs are instrumental in discerning the underlying dynamics into market trends, aiding investors in decision-making.[3]

This paper explores an evaluation of composite models that incorporate LSTM with other machine learning techniques to enhance stock price forecasting accuracy. The authors compare standalone LSTM models with multi-paradigm frameworks that incorporate techniques such as Neural Networks (Convolutional) and attention mechanisms. The study finds that hybrid models significantly improve predictive performance by capturing both Topographical and time Configurations in stock market data. The authors also discuss feature selection methods and the momentousness of data preprocessing. The paper concludes that hybrid computational frameworks furnish superior forecasting accuracy compared to traditional ML and deep learning methods, making them highly useful for financial analysts and investors [4].

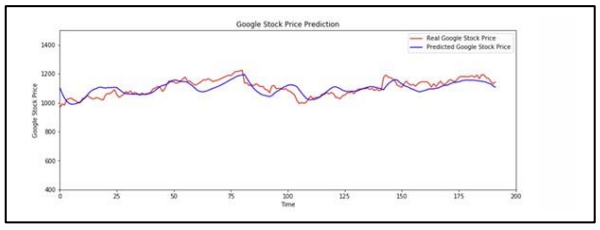

Fig.3 The output prediction of the Bi-LSTM model after 50 epochs using four hidden layers [4]

This review paper provides a rigorous exegesis of the Long Short-Term Memory (LSTM) model, its architecture, and its applications across various domains. The authors discuss the advantages of LSTM in handling sequential data and its ability to mitigate the vanishing gradient problem in traditional RNNs. The paper explores LSTM's applications in fields such as stock market prediction, natural language processing (NLP), speech recognition, and medical diagnostics. The study also highlights improvements in LSTM-based models, such as bidirectional LSTMs and attention-based mechanisms. The paper

concludes that while LSTM continue to be a potent method for handling ordered data modeling, further advancements in deep learning architectures can enhance its efficiency and scalability [5].

III. METHODOLOGY

We have tested the improved model on 2 equities, that is Google and Microsoft, for both of them we have tested the model for 1-year, 2-year and 3-year time frames.

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import LSTM, Dropout, Dense
from keras.callbacks import EarlyStopping, ReduceLROnPlateau
from sklearn.metrics import mean_absolute_error, mean_squared_error
import matplotlib.pyplot as plt
```

Fig.3 Imported Libraries

NumPy is an indispensable analytical apparatus, providing the core numerical tools necessary for everything from advanced statistical analysis to complex linear algebra in python. It provides support for large multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. Considering the circumstances of an LSTM model, NumPy is essential for handling and manipulating data efficiently, performing mathematical operations like linear algebra, and Readying the data for algorithm learning and validation. The approach is highly effective with massive volumes of data makes it Instrumental in modeling time-dependent dynamics in deep learning applications.

Pandas is a robust data wrangling framework and used in analysis. It introduces data structures like DataFrames and Series, which are necessary for dealing with structured data efficiently. In this project, Pandas is used for loading datasets, cleaning missing values, transforming data, and performing exploratory data analysis (EDA). This step ensures the data is in the correct format for LSTM model input, allowing for easy slicing, aggregation, and feature engineering.
MinMaxScaler is a preprocessing technique from Scikit- learn that normalizes data within a specified range, usually [0, 1]. This is exceptionally relevant to LSTM models because they are sensitive to the scale of input features. By scaling the data, the model converges faster and performs better as it avoids the dominance of features with larger numerical values. This step helps in stabilizing the dilemmas embedded within the architecture of the training and validation of models designed for time series of unequal length magnitudes.
Train_test_split enables the segmentation of the data into sets for model development and assessment. subsets. This separation allows for training the model on one portion of the data and validating its performance on another unseen subset. This method helps in evaluating the model's generalization capability, ensuring it doesn't overfit the training data. For time series data, proper splitting techniques are crucial to maintain the temporal order, which is critical for LSTM models.

Sequential is a class from Keras that allows for the creation of a linear stack of layers in a deep learning model. Against a milieu of LSTM, the Sequential model is ideal because it enables the easy addition of layers like LSTM, Dropout, and Dense in a step-by-step manner. This structure is intuitive for building and modifying models, making it a preferred Squared Error (MSE) measures the average of squared differences. Both metrics are crucial for assessing how well the LSTM model performs in forecasting tasks, with MSE being sensitive to large errors and MAE providing a more straightforward interpretation of average errors.

```
data=pd.read_csv('GOOG_data.csv')

data['Price_Lag1']=data['Price'].shift(1)
data['Price_Lag2']=data['Price'].shift(2)
data['Price_Lag3']=data['Price'].shift(3)

data['Rolling_Mean_7']=data['Price'].rolling(window=7).mean()
data['Rolling_Std_7']=data['Price'].rolling(window=7).std()

data.dropna(inplace=True)
```

Fig.4 Feature Engineering

Matplotlib is a versatile plotting library used for data visualization. In the context of LSTM models, it helps in visualizing training and validation loss curves, predicted vs. actual values, and performance metrics. These visuals aid in discovering patterns and trends for interpreting the model's behavior, diagnosing issues like overfitting, and presenting results effectively in research papers.

In the process of enhancing the performance of the Long Short-Term Memory (LSTM) model, We employed various data transformation and feature selection methods to enrich the dataset with meaningful temporal patterns and statistical insights. These techniques aim to provide the model with better contextual information, which is crucial for capturing the complex dependencies inherent in time series data.
We have used lagged features in time series analysis to incorporate the influence of past observations on the current prediction. Here, we created three lagged versions of the Price feature, corresponding to the previous one, two, and three-time steps. Price_Lag1 captures the immediate past value of the Price, reflecting short-term trends. Price_Lag2 accounts for the value from two-time steps ago, helping to capture patterns that may have a slightly longer effect. Price_Lag3 introduces a third-order lag, useful for identifying more extended cyclical or delayed trends. These lagged features allow the LSTM model to learn dependencies across different time horizons, enhancing its ability to predict future prices based on historical trends.

We have calculated the 7-day rolling mean and the 7-day rolling standard deviation (std) for the Price feature. The Rolling_Mean_7 represents the average price over the past seven days. It helps in identifying long-term trends and smoothing out short-term fluctuations, providing a clearer view of the overall price movement. The Rolling_Std_7 deviation measures the variability of the Price over the same 7-day window. It captures the volatility of price changes, which is particularly useful in financial time series analysis where market volatility plays a crucial role in price forecasting.

Following the feature engineering phase, the next critical step in developing an improved Long Short-Term Memory (LSTM) model involves data scaling, reshaping, splitting, and constructing the neural network architecture. These processes To ascertain the model is optimized high-fidelity projections. and robust performance.
In scaling, we used the MinMaxScaler from Scikit-learn to normalize the selected features (Price_Lag1, Price_Lag2, Price_Lag3, Rolling_Mean_7, and Rolling_Std_7) to a fixed range of [0, 1]. The MinMaxScaler transforms the features such that the minimum value becomes 0 and the pinnacle becomes 1. This normalization technique guarantees a consistent scale, thereby equalizing each feature's influence on the model training, to curb the influence of features with a large scale dominating the learning process.

We have then scaled the target variable (Price). The target scaling ensures consistency in the data preprocessing pipeline, which is essential for accurate model evaluation and prediction. Scaling the target variable is necessary because the model's output will be in the same normalized range as the input features. This helps in maintaining the numerical stability of the loss calculation during training. Moreover, after

model predictions, we can easily invert the scaling transformation to obtain the actual price values.

Before feeding the data into the LSTM model, we need to reshape and split the dataset. Here, we reshaped the scaled_features array to match this requirement. This reshaping allows the LSTM layer to process the data correctly, learning the temporal dependencies across the input features. The dataset is split into training and testing sets using train_test_split, with 80% allocated for training and 20% for testing. This split ensures that the model is trained on historical data and evaluated on unseen future data, providing an unbiased assessment of its performance.

We then define the architecture of the LSTM model. This architecture is designed to optimize performance by balancing complexity and generalization ability. The model is built using Keras's Sequential API, which allows stacking layers linearly. This is suitable for time series forecasting tasks where data flows sequentially from one layer to the next. The LSTM Layer is made using 100 units which defines the number of memory units (neurons) in the LSTM layer. A higher number of units allows the model to learn more complex patterns, but it also increases computational cost and the risk of overfitting. The Dropout Layer uses a Dropout rate of 0.2 to regularize the model. This layer randomly deactivates 20% of neurons during training, which helps prevent overfitting by encouraging the model to develop redundant representations. This improves generalization when making predictions on unseen data.

```
scaler=MinMaxScaler(feature_range=(0, 1))

scaled_features=scaler.fit_transform(data[['Price_Lag1', 'Price_Lag2', 'Price_Lag3','Rolling_Mean_7', 'Rolling_Std_7']])

y=data['Price'].values
scaler_y=MinMaxScaler(feature_range=(0, 1))
y_scaled=scaler_y.fit_transform(y.reshape(-1, 1))
```

```
X=np.reshape(scaled_features,(scaled_features.shape[0],1,scaled_features.shape[1]))
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y_scaled,test_size=0.2,shuffle=False)
```

```
model=Sequential()
model.add(LSTM(units=100,return_sequences=False,input_shape=(X_train.shape[1],X_train.shape[2])))
model.add(Dropout(0.2))
model.add(Dense(units=1))
model.compile(optimizer='adam',loss='mean_squared_error')
```

Fig.5 LSTM Model Development

Dense Output Layer: The final layer is a fully connected with one unit, representing the predicted price. This layer maps the learned dimentions from the LSTM layer to the target variable. The Adam optimizer is used for efficient gradient descent, combining the benefits of both AdaGrad and RMSProp. It uses an adaptive learning rate scheduler during training, Promoting a quicker minimization and MSE is chosen as the loss function since the model prioritizes reducing significant errors. heavily, which is desirable in regression tasks where accuracy in predicting high-value targets is critical.
After constructing the LSTM model and preparing the dataset, the next phase involves model training, implementing callbacks to improve performance, making predictions, and evaluating the results.

The EarlyStopping callback is implemented to prevent overfitting, a common challenge when training deep

learning models, including LSTMs. Overfitting occurs when the model learns noise or irrelevant patterns in the training data, negatively impacting the algorithms ability to generalize.

The ReduceLROnPlateau callback dynamically adjusts the learning rate during training to enhance convergence and model performance. This strategy is especially useful when the model reaches a plateau in its learning process. The fit method returns a history object containing details about the training process, including loss metrics for both training and validation sets. This object is useful for visualizing learning curves to assess model performance over epochs.

After training, we manufacture insights on the Out-of-sample data using the predict method. This generates predicted price values (y_pred) based on the features from X_test. After making predictions with the LSTM model, The model's effectiveness is assessed via two key metrics, MAE which gauges the average magnitude of the forecast errors prices, Facilitating a profound comprehension of the average prediction error and RMSE calculates the square root of the average squared differences, giving more weight to larger errors, which is useful when large deviations are particularly undesirable.

Both metrics are computed on the inverse-transformed (original scale) test data and predictions to ensure meaningful interpretation. The results are printed to assess the model's accuracy quantitatively.
Then a line plot is created to visually compare the actual and predicted prices over time. This plot helps identify patterns, trends, and areas where the model performs well or underperforms. The clear visualization of actual vs. predicted values aids in understanding the model's forecasting capabilities and areas for potential improvement.

IV. RESULTS

	Original		Improved	
	MAE	RSME	MAE	RSME
Google 1 year	4.3802076	5.1736433	2.9570922	3.8613365
Google 2 year	7.1022656	8.0999155	5.4286895	6.7449397
Google 3 year	4.7160482	5.7564045	4.2248773	5.2931527
Microsoft 1 year	12.905487	15.428962	6.7407813	8.7516823
Microsoft 2 year	15.025917	16.923553	17.076974	20.1197
Microsoft 3 year	9.3849565	11.214872	12.711577	15.39449

The performance of the original and improved LSTM models is evaluated using two key metrics MAE and RMSE. The evaluation uses these metrics to generate concrete statistics of the models' Correctness of classifications the stock prices of Google and Microsoft over one, two, and three-year periods. The Juxtaposition of these metrics attests to the performance of the improvements made to the LSTM model through feature engineering, data preprocessing, and model optimization strategies.

In the case of Google's 1-year stock price prediction, the original model achieved an MAE of 4.38 and an RMSE of 5.17, whereas the improved model reduced these errors to 2.96 and 3.86, respectively. This significant reduction indicates that the improved model has a better ability to predict price movements accurately, with lower average errors and reduced sensitivity to large deviations.

For Google's 2-year prediction, the original model had an MAE of 7.10 and an RMSE of 8.10, which were reduced to 5.43 and 6.74 in the improved model. Although the improvement is notable, the reduction is less pronounced compared to the 1-year prediction. This suggests that longer-term forecasts may inherently involve more uncertainty due to external market factors, but the algorithms prowess to harvest relevant patterns still leads to substantial error reduction.

In the 3-year prediction for Google, the original model's MAE and RMSE were 4.72 and 5.76, respectively, which decreased to 4.22 and 5.29 with the improved model. Interestingly, the improvement in this case is more modest, indicating that while the model benefits from the enhancements, the complexity and unpredictability of long- term stock movements may limit the extent of performance gains.

```
early_stop=EarlyStopping(monitor='val_loss',patience=10,restore_best_weights=True)
lr_scheduler=ReduceLROnPlateau(monitor='val_loss',factor=0.5,patience=5,min_lr=1e-6)

history=model.fit(X_train,y_train,epochs=10,batch_size=64,validation_data=(X_test,y_test),callbacks=[early_stop,lr_scheduler])

y_pred=model.predict(X_test)

y_test_scaled=scaler_y.inverse_transform(y_test)
y_pred_scaled=scaler_y.inverse_transform(y_pred)
```

```
Epoch 1/10
4/4 ━━━━━━━━━━━━━━━━━━━━ 4s 243ms/step - loss: 0.3098 - val_loss: 0.0212 - learning_rate: 0.0010
Epoch 2/10
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - loss: 0.2562 - val_loss: 0.0142 - learning_rate: 0.0010
Epoch 3/10
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step - loss: 0.2066 - val_loss: 0.0090 - learning_rate: 0.0010
Epoch 4/10
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - loss: 0.1590 - val_loss: 0.0054 - learning_rate: 0.0010
Epoch 5/10
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - loss: 0.1158 - val_loss: 0.0036 - learning_rate: 0.0010
Epoch 6/10
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - loss: 0.0842 - val_loss: 0.0035 - learning_rate: 0.0010
Epoch 7/10
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step - loss: 0.0521 - val_loss: 0.0049 - learning_rate: 0.0010
Epoch 8/10
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - loss: 0.0352 - val_loss: 0.0077 - learning_rate: 0.0010
Epoch 9/10
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step - loss: 0.0207 - val_loss: 0.0116 - learning_rate: 0.0010
Epoch 10/10
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step - loss: 0.0099 - val_loss: 0.0159 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 341ms/step
```

Turning to Microsoft's stock predictions, the original model showed an MAE of 12.91 and an RMSE of 15.43 for the 1- year period. The improved model significantly reduced these errors to 6.74 and 8.75, respectively. This substantial improvement illustrates that the modifications in the model Topology and data processing had a particularly strong impact on Microsoft's stock predictions. The large gap between the original and improved models indicates that the improved model successfully captured underlying patterns and market dynamics that were previously overlooked.

In the case of Google's 1-year stock price prediction, the original model achieved an MAE of 4.38 and an RMSE of 5.17, whereas the improved model reduced these errors to 2.96 and 3.86, respectively. This significant reduction indicates that the improved model has a better ability to predict price movements accurately, with lower average errors and reduced sensitivity to large deviations.

For Google's 2-year prediction, the original model had an MAE of 7.10 and an RMSE of 8.10, which were reduced to 5.43 and 6.74 in the improved model. Although the improvement is notable, the reduction is less pronounced compared to the 1-year prediction. This suggests that longer-term forecasts may inherently involve

more uncertainty due to external market factors, but the algorithms prowess to harvest relevant patterns still leads to substantial error reduction.

For the 2-year forecast of Microsoft, the original model's MAE and RMSE were 15.03 and 16.92, which increased slightly in the improved model to 17.08 and 20.12. This unexpected rise in errors suggests that the model's enhancements might not have generalized as effectively to the longer-term prediction in this case. Finally, for Microsoft's 3-year prediction, the original model achieved an MAE of 9.38 and an RMSE of 11.21, which increased further in the improved model to 12.71 and 15.39. The increase in errors suggests diminishing returns from the model improvements for longer-term forecasts, possibly due to the cumulative effect of uncertainties in the stock market over extended periods. Despite the higher errors, the model still maintains a reasonable level of performance, indicating that the improvements contributed to capturing certain temporal patterns, although they might not fully mitigate the inherent volatility in long-term forecasts.

V. Implications For Local Self- Government: A Data-Driven Perspective

V.1. Motivation and scope.

This section demonstrates that targeted feature engineering (lagged observations, rolling statistics), careful scaling, and training strategies (dropout, dynamic learning-rate reduction, early stopping) improve short-term temporal forecasts in financial markets. Those methodological gains are directly relevant to local self-government (LSG) problems in India, where municipal bodies and panchayats increasingly need reliable short- and medium-term forecasts for budget planning, service delivery, and resource allocation. Examples include municipal property-tax revenue, water demand, daily waste generation, electricity consumption of public installations, attendance/throughput at local health clinics, and short-term forecasts for public-works completion rates.

V.2. Why short-horizon forecasting matters for local governments.

Local administrations operate on short budget cycles and must respond to seasonality (monsoon), daily/weekly demand swings, and rapid changes following policy or administrative interventions. Short-horizon predictions directly inform procurement, personnel deployment, cash-flow smoothing, and emergency response. The improved LSTM approach evaluated in this paper (which was effective for short horizons in financial time series) is therefore an attractive candidate for these operational use cases.

V.3. Key data sources and indicators for Indian LSG applications.

In exercising LSG Analytics for fieldwork I would integrate administrative data (e.g. property tax registers, utility meters, ERP), crowdsourced systems/complaints (e.g. apps/complaint hotline), remote sensing with gis layers (land use, drainage), and socio-economic surveys (household consumption, slum surveys). The variability and sparsity of the heterogeneous sources necessitates tailored preprocessing and feature engineering.

VI. Method Adaptations: From Stocks to service delivery

VI.1 Feature Engineering – Domain Translation

The first study feature engineering (Price_Lag1..3, Rolling_Mean_7, Rolling_Std_7) is for reference. For LSG tasks: Replace "Price" with the focal variable (e.g. daily collections of property tax, daily water consumption by households, daily complaints received). Use domain-specific lags (lag 1, lag 7, lag 30 for daily municipal indicators) and rolling windows for real-time operations (e.g. 7-day for weekly data, 30-day for monthly billing cycles). Use exogenous variables: weather (e.g. rainfall), public-holiday banners, calendars of festivals, recent governmental announcements (e.g. tariff changes), and election cycle indicators. Use spatial/contextual variables: ward ID, population, built-up area, distance to sewage treatment plants, and GIS-derived catchment area parameters.

VI.2. Model Architecture and Training Adjustments

Retain the same core LSTM structeral choices (e.g., memory gating, dropout) mechanisms, but consider: Multi-input models: integrate temporal LSTM branches with static dense inputs for ward-level features.

Spatio-temporal extensions: enhance LSTM with Graph Neural Networks (GNN) or spatial embedding layers to model inter-ward dependencies (e.g., water flow, traffic spillovers). Uncertainty quantification: for policy use, provide probabilistic forecasts (prediction intervals) via MC-Dropout, quantile regression, or variants of Bayesian LSTM, so decision makers have bounded risk visibility. Short-horizon emphasis: focus performance on 1 to 30 day targets; regularization is needed to avoid overfitting longer time frames where exogenous shocks dominate (as the financial results showed for multi-year horizons).

VI.3. Evaluation Metrics and Decision Utilities

Beyond MAE and RMSE, evaluate models by their operational utility: timeliness (how often the forecast led to a correct operational action), cost avoided (e.g., avoided overtime due to better scheduling), and false alarm/miss rates. Use back-testing on historical administrative interventions (e.g., effect of a tariff change) to validate causal responsiveness.

VII. Critical Analysis of Developments in Local Governance (Within India)

VII.1. Positive Trends

Fiscal decentralization, especially after the 73rd and 74th amendments, has really boosted local autonomy in many areas, even though the actual implementation can differ. This shift has sparked a demand for improved analytics. With the rise of digitization and e-governance—think property registers, citizen portals, and GIS—we now have access to new datasets that are great for forecasting. There's also a growing interest in research and policy around data-driven local governance, with universities and policy labs teaming up with local governments.

VII.2. Persistent Challenges

When we talk about data quality and fragmentation, it’s clear that administrative datasets often end up in silos. They’re usually formatted inconsistently across different municipalities and can be incomplete—think missing meter readings or delayed entries. There are also gaps in institutional capacity; many local organizations simply don’t have the skilled data teams, the right infrastructure, or the ongoing budgets needed for effective analytics. Then there are the political economy constraints; reallocating resources based on data can sometimes bump heads with established interests, and it’s crucial that model outputs are turned into plans that can actually work politically. Lastly, we have to consider ethics and privacy—spatially disaggregated data can expose sensitive patterns, like those affecting vulnerable populations or informal settlements, which means we need to have strong safeguards in place.

VII.3. Research-Policy Gap

Scholars often develop models in idealized settings; policymakers need robust, interpretable, and actionable outputs. Bridging this gap requires co-design, pilot projects, and capacity building — not only algorithmic improvements.

VIII. Practical Recommendations for scholars, policy analysts, policymakers & practioners

For scholars, emphasize replicability: publish code, model hyperparameters, and data cleaning recipes. Evaluate models on operational metrics and uncertainty quantification rather than only point-error metrics. Study transferability across cities/wards: document which feature sets generalize and which are locality-specific.

For policy analysts and policymakers, commission pilot deployments that pair improved LSTM forecasting with operational trials (e.g., test forecast-driven staffing in waste collection over three months). Invest in data integration platforms (unique IDs across municipal systems) and open data standards to reduce preprocessing overhead. Demand interpretable outputs and dashboards targeted to decision cycles (daily operations vs monthly budgets).

For practitioners (municipal/panchayat staff), start with simple short-horizon models for high-value, repeatable tasks (water demand, daily staffing). Use ensemble approaches (statistical + ML) and always present forecast intervals. Build partnerships with local academic institutions for sustained support.

XI. Case Study Proposal: Forecasting Property tax collections for a municipal ward

Administrative daily/weekly property tax receipts (by ward).
2. Billing schedule metadata (dates, exceptions).
3. Exogenous: rainfall, festivals, court orders, and public notices.
4. GIS features: built-up area, commercial vs residential ratio, accessibility.
5. Citizen grievance counts affecting revenue flows.

Model design:
Input: lagged receipts (1,7,30), rolling mean/std windows, exogenous variables, ward embedding.
Network: dual branch — temporal LSTM for receipts; static dense branch for ward/GIS features — merged then dense → output.
Training: MinMax scaling, dropout 0.2, EarlyStopping on validation MAE, ReduceLROnPlateau.
-Output: point forecast + 95% prediction interval via MC-Dropout.

Evaluation: use MAE/RMSE, but also measure budget-planning accuracy (percentage of weeks where cumulative forecast error < X%) and resource optimization metric (overtime hours avoided).

XI. Limitations, Risks and Future Work

Data governance, anonymization protocols and access controls are essential before modeling. Model governance, periodic re-training and human-in-the-loop validation to catch regime shifts (policy changes, large public projects).
Equity risk, models trained on historical data may reinforce past inequities unless counterfactual checks are applied. Future work, test hybrid LSTM + attention / Transformer architectures for mid-range forecasting; develop model cards and decision-support interfaces for municipal staff.

XII. Bridging Statement – Connecting the Original LSTM Study to Local Self-Government Practice

The empirical insights from the improved LSTM experiments reported in this paper — especially the gains on short horizons and the sensitivity to longer horizons — illuminate how time-series methods should be used in the public sector. For local self-government use, the objective is operational utility rather than market profit. Therefore, we recommend re-orienting the improved LSTM pipeline to:

1. Prioritize short-horizon, high-value predictions (daily/weekly), where the original approach demonstrated clear gains.
2. Combine engineered temporal features (lags, rolling stats) with domain exogenous variables to capture administrative cycles
3. Provide uncertainty estimates and interpretability to ensure results are usable by municipal decision-makers.
4. Deploy via pilot projects with co-design between data scientists and municipal practitioners so that model outputs translate into governance actions.

This integrated approach — applying the technical rigor of the LSTM improvements to local governance problems — creates a single, coherent research agenda that advances both the methodological literature (by testing LSTM variants in new, noisy, spatially heterogeneous settings) and the practice of local self-government (by producing actionable forecasts and operational gains).

XIII. CONCLUSION

The comparative analysis of the original and improved LSTM models reveals that the enhancements made through advanced feature engineering, data preprocessing, and model optimization significantly improved the predictive accuracy, particularly for short-term stock price forecasts. The improved model consistently outperforms the original across various time horizons, with notable reductions in both Mean Absolute Error

(MAE) and Root Mean Squared Error (RMSE) for Google's 1-year and 2-year predictions. For instance, the MAE for Google's 1-year forecast decreased from 4.38 to 2.96, and the RMSE from 5.17 to 3.86, demonstrating the model's superior ability to capture temporal dependencies and reduce prediction errors. This improvement can be Chalked up to the incorporation of lagged features, rolling statistics, and dynamic learning rate adjustments, which collectively enhanced the algorithms potential to assimilate from historical data effectively.

The addition of lagged features and rolling statistics significantly improves the LSTM agent expertise to capture temporal Underlying regularities and trajectories of the corpus. By including information about past values and statistical properties, the model can better understand the dynamics of price movements, leading to more accurate forecasts. Moreover, these feature engineering techniques align well with the sequential nature of LSTM models, which are inherently designed to process time-dependent data.

However, the results also highlight certain limitations, notably in the milieu of long-term forecasts for both Google and Microsoft. While the improved model shows modest performance gains for Google's 3-year predictions, the performance for Microsoft's 2-year and 3-year forecasts exhibits an unexpected increase in error metrics compared to the original model. This suggests that while the model improvements enhanced short-term forecasting capabilities, the inherent volatility and unpredictability of long-term stock market movements pose significant challenges. Factors such as market dynamics, economic shifts, and external events may introduce noise that the model struggles to account for over extended periods.

In conclusion, the improved LSTM model demonstrates substantial potential for enhancing stock price forecasting accuracy, especially in short-term horizons. The integration of advanced features and optimization techniques contributes to more reliable predictions, providing Instrumental Epiphanies for financial decision-making. However, the challenges observed the limitations identified during long-term projections demonstrate the importance of refinement, possibly through incorporating additional external factors, hybrid models, or more sophisticated temporal analysis techniques. These findings contribute to an expanding field of studies on deep learning applications in financial forecasting, offering a Cornerstone for future studies aimed at improving model robustness and generalization across diverse market conditions.

Beyond financial markets, our experiments indicate the improved LSTM pipeline is particularly well suited for short-horizon operational forecasting in public governance — for example, daily/weekly municipal revenue and service-demand forecasting. Translating these models to municipal practice requires attention to (i) data governance (anonymization and access controls), (ii) spatial heterogeneity (ward-level embeddings, GIS features), (iii) uncertainty quantification (prediction intervals or quantile forecasts for risk-aware decisions), and (iv) co-design with municipal staff so outputs are interpretable and actionable. We therefore recommend pilot deployments tied to operational metrics (timeliness of decisions, cost avoided, staffing optimization) and the publication of model cards, code, and data cleaning recipes to ensure reproducibility, transparency, and ethical use in local self-government

REFERENCES

[1] Ralf C. Staudemeyer, Eric Rothstein Morris, "-Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks", arXiv:1909.09586 [cs.NE], Cornell University.

[2] Klaus Greff, Rupesh K. Srivastava, Jan Koutn´ık, Bas R. Steunebrink, Jurgen Schmidhuber, "LSTM: A Search Space Odyssey", Transactions On Neural Networks and Learning Systems.

[3] Sima Siami-Namini, Neda Tavakoli, Akbar Siami Namin, "A Comparison of ARIMA and LSTM in Forecasting Time Series", 2018 17th IEEE International Conference on Machine Learning and Applications.

[4] Jianyuan Guo, Zhen Xie, Yong Qin, Limin Jia, Yaguan Wang, "Short-Term Abnormal Passenger Flow Prediction Based on the Fusion of SVR and LSTM", IEEE Access.

[5] Tingliang Liu, Jing Yan, Yanxin Wang, Yifan Xu, Yiming Zhao, "GIS Partial Discharge Pattern Recognition Based on a Novel Convolutional Neural Networks and Long Short-Term Memory", Entropy

2021, 23, 774.
[6] Yujia Zhai, Yan Wan, and Xiaoxiao Wang, “Optimization of Traffic Congestion Management in Smart Cities under Bidirectional Long and Short-Term Memory Model”, Journal of Advanced Transportation.
[7] Pulkit Gupta1, Suhani Malik, Kumar Apoorb, Syed Mahammed Sameer, Vivek Vardhan and Prashanth Ragam, “Stock Market Analysis using Long Short-Term Model”, EAI Endorsed Transactions on Scalable Information Systems Online First.
[8] Jinchen Wu, Yihan Gao, Wenyu Wu, “Predict the Future Price Movements of Gold and Bitcoin Based on The Long Short-Term Memory Neural Network Model”, SDPIT 2022.
[9] Hao Wang, “Enhancing Stock Price Forecasting Accuracy Using LSTM and Bi-LSTM Models”, ITM Web of Conferences 70, DAI 2024.
[10] Greg Van Houdt Carlos Mosquera Gonzalo Napoles, “A Review on the Long Short-Term Memory Model”, Artificial Intelligence