Part I , a magazine paper analysis

Introduction

A group of researchers from the Nepalese Mathematical Society in the USA published a paper, 'Predicting Stock Market Index Using Long Short-Term Memory Architecture (LSTM),' in the Machine Learning with Applications journal Nr 9, 2022 (Bhandari et al., 2022). The paper has been cited 271 times since then. It is well known that predicting future returns is a challenging task due to the noisy, non-linear, and dynamic nature of stock markets. Nevertheless, the researchers recognized the need to develop a model that captures various dimensions of stock market behavior while maintaining simple architecture. LSTM, an advancement of Recurrent Neural Networks (RNNs), addresses the vanishing gradient problem, present in earlier architectures while preserving the ability to capture both long- and short-term trends in time series data, making it suitable for stock market price prediction (Hochreiter, 1998).

Methodology

The study focused on predicting the next day's closing price of the S&P 500 index as the target variable, explained by 11 raw and generic features (Appendix A). These were processed using an LSTM neural network architecture (Appendix B). The optimal algorithm configuration was determined using a two-loop approach:
1. The first loop optimized batch sizes, choice of optimizers, and learning rates as tuning parameters, while input observations, time steps, and features were tested (Algorithm 1, Appendix C).
2. The second loop fine-tuned the number of layers and neurons while using the input variables and hyperparameters determined in the first loop (Algorithm 2, Appendix D).
The researchers evaluated model performance using RMSE(Root Mean Squared Error), MAPE (Mean Absolute Percentage Error), and R(correlation coefficient) metrics.

Data

The dataset, spanning from 2006 to 2020, was obtained from public sources such as FRED and Yahoo. The choice of each feature was supported by cited research works. The study incorporated fundamental, macroeconomic, and technical indicators to capture the multidimensional nature of stock markets.

The selected timeframe was justified by the researchers, as it included extreme market conditions such as bull and bear markets and financial crises (2008, 2020), thus ensuring model robustness through the inclusion of a complete stock market cycle(p.5). However, the primary critique of the study relates to data preparation, which is addressed in a separate section.

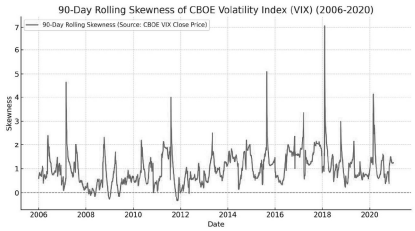
Findings and Analysis of Results

According to test results, the best-performing model was a single-layer LSTM with 150 neurons, the Adam optimizer, and a batch size of 8, achieving RMSE (40.4574), MAPE (0.7989), and R (0.9976). Across all tested models, the minimum R value was 0.9903. Given the noisy and stochastic characteristics of stock market prices, such a high explanatory power raises concerns of potential overfitting.

Further analysis indicates that data preparation methods in the study were inconsistent with best practices for time series modeling. In Section 6 (Ethics and Implications, p. 11), the authors state that the data 'was not modified except as explained in the paper.' A Min-Max transformation was applied to normalize the entire dataset while preserving its dynamic structure and proven to be optimal for LSTM(Kim et al., 2025). However, as stated on p. 7, this normalization was performed before splitting the dataset into training and test sets, leading to information leakage (Apicella et al., 2024). Additionally, the study does not mention time-series purging, which prevents cross-contamination between training, validation, and test sets (BBK SL, L7, Liu et al., 2022). This oversight contributed to data leakage and likely explains the overly optimistic R values, whereas performance on unseen data would likely be weaker.

Apart from these issues, the study contains smaller errors and omissions. The target and most explanatory variables, except UNRATE and UMCSENT, are daily. The researchers applied forward filling; a method typically used when a variable is assumed to remain unchanged until the next update. However, given that these macroeconomic rates change gradually rather than overnight, a log extension of the difference to daily intervals or another filling method might have been more appropriate. This is particularly important since the final correlation heatmap shows that the coefficient between UMCSENT and Close is one of the highest accepted, with 0.8 as the rejection threshold. Additionally, the paper states that the coefficient between UMCSENT and Close is 0.067(4.4, p.6) , but the published heatmap indicates 0.51(Appendix E). Looks like the final heatmap was omitted from the paper.

Appendix A includes time series data for 9 variables, including the target, yet the paper states that the LSTM model used 11 features (p. 6, before section 4.5). Page 8 features a graph (Appendix F) displaying the joint S&P 500 index Close price, 50-day, and 200-day moving averages, suggesting that these might be the omitted features. However, replicating the study or contacting the authors would be necessary to confirm the discrepancy.
The paper does not explicitly mention treatment of outliers, which can impact model performance (Saha et al., 2022). Similarly, it does not address data skewness, which is primarily caused by right skewness of the VIX (CBOE Volatility Index, colloquially – Fear index).



The chosen time frame might enhance model robustness for general market conditions, but the smoothing of extreme coefficients could diminish predictive power in crisis scenarios. Introducing a categorical variable representing different market conditions with a zero baseline might enhance confidence in rare but significant events (Kroujiline et al., n.d.)

The article does not explicitly state the sequence length used in the LSTM model nor justify its selection through research. The sequence length was not included in the model's hyperparameter tuning but was simply used as a time step input (Appendix C, D). However, sequence length is a critical factor in LSTM model performance, as, unlike traditional feedforward neural networks, LSTMs rely on sequence-based backpropagation.

Conclusion

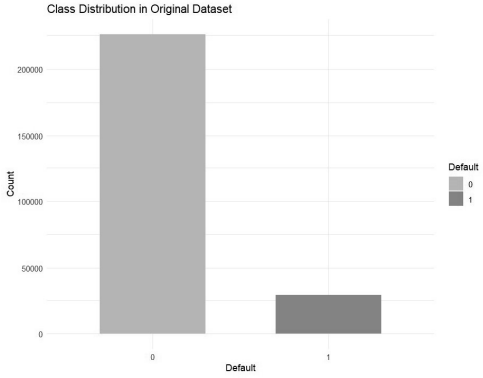
The study's code is not publicly available, making it impossible to verify the omitted details or conduct experiments to correct the errors. As a result, the findings cannot be fully validated, and any replication of the study would reproduce the methodological flaws. While this group of researchers continues to publish in scientific journals, their work should be scrutinized before drawing conclusions.

Part II, Data Analysis.

The dataset originated from Coursera Loan Prediction Challenge, mixed 18 columns, 255,347 rows (*Loan Default Prediction Dataset*, n.d.)

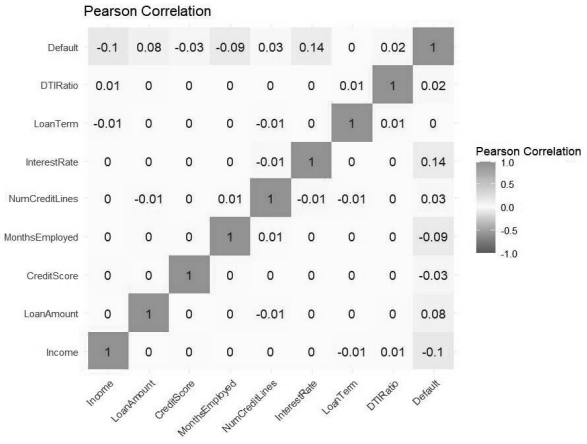
Data preparation and EDA

The original dataset comprises a mixture of numerical and categorical variables, including character strings, integers, and continuous numeric features. The target variable, *'Default'*, is binary and exhibits a significant class imbalance—a characteristic that is frequently encountered in real-world loan default datasets .

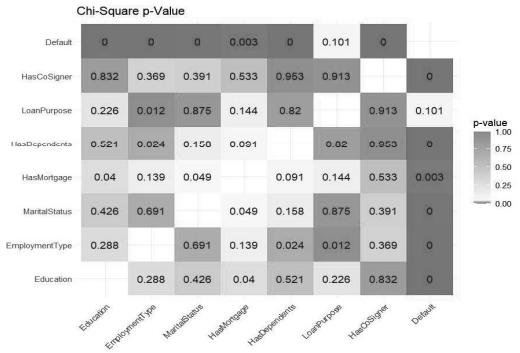


Due to computational limitations, I applied stratified sampling (10%) to preserve the class distribution. Following this, I removed two variables: '*LoanID*', as it serves solely as an identifier, and *'Age'*, excluded on ethical grounds as a protected attribute to mitigate potential bias in model predictions.

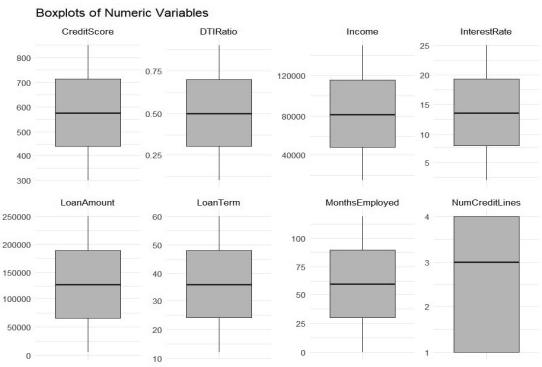
To examine associations among numerical features, including the target, I generated a Pearson correlation heatmap to assess linear dependencies. Although the heatmap revealed weak linear associations between individual predictors and *'Default'*, the predictors appear to be mutually independent.



To evaluate the relevance of categorical variables to the target, I transformed all character columns and Default into factor types. A Chi-Square test of independence, visualized via a heatmap, indicated that all categorical features—except '*LoanPurpose*'—exhibit statistically significant dependency on the target variable, with p-values below 0.001 (i.e., >99.97% confidence). In contrast, *'LoanPurpose'* shows weak dependency overall: its p-value in relation to the target is 0.101, and it has p-values above 0.14 with most other categorical variables, except *'EmploymentType'*, where the p-value is 0.012. Given its limited statistical association and to maintain model parsimony, I excluded *'LoanPurpose'* from further analysis.



Upon inspection, I found that the dataset does not contain any significant outliers, and most numerical variables are symmetrically distributed around the median (as shown in the boxplots).



For subsequent machine learning modeling, I applied one-hot encoding to the remaining categorical features and reverted the target variable to binary numeric format. The resulting dataset comprises 21 variables: 13 derived from categorical features (one dummy omitted per variable to avoid multicollinearity) and 8 original numerical predictors.

Given the heterogeneous nature of the dataset and the presence of both categorical and numerical variables, the choice of algorithm is naturally inclined toward those that can handle non-linear decision boundaries and mixed data types. Suitable candidates include logistic regression, support vector machines(SVM) with a radial basis function (RBF) and polynomial kernel(due to interactions among categorical variables – see the Chi-Square Heatmap), Naïve Bayes, Random Forest, XGBoost and Neural Networks. The observed positive-to-negative class ratio is 0.131 [1]:1 [0]. Since this ratio is above the typical threshold for extreme imbalance (1:10), methods within binary classification algorithms are more suitable than employing an anomaly detection framework. In this work I used Logistic Regression as a base model and SVM as an advanced option. Before transforming the data for model training, I partitioned the dataset (*df_encoded*) into training (80%), and test (20%) sets using a stratified sampling method. This approach preserves the class distribution of the *Default* variable while simultaneously preventing data leakage that would otherwise occur if transformations were applied prior to splitting.

Method 1: Binary classification with Logistic Regression

I computed the model to establish a baseline for the dataset classification task. First, I addressed class imbalance using SMOTE (Synthetic Minority Oversampling Technique) augmentation. I then proceeded with binary classification, optimizing the model for Sensitivity (Recall), as this metric is crucial for predicting loan defaults. Since logistic regression is invariant to monotonic transformations, I used the raw numeric features in the algorithm. The best-performing model used a probability threshold of 0.55, yielding a balanced accuracy of 0.63, with a sensitivity of 0.52 — just over half.

Before fitting Lasso-regularized logistic regression, I standardized the numeric features using a Standard Scaler, as regularization penalties are scale-sensitive. It is important for all numeric features to have zero mean and unit variance for consistent and fair penalization.

$$\min_{\beta} \left\{ \sum (y - X\beta)^2 + \lambda \sum |\beta_j| \right\}$$

The Lasso-regularized model produced slightly worse results (best at a 0.55 probability decision boundary), with a balanced accuracy of 0.6286. Although it showed slightly improved recognition of the minority class, the sensitivity dropped to 0.51. This indicates that the logistic regression algorithm struggles to effectively distinguish between the *'Default'* classes.

The underperformance of both logistic regression models — despite clean, balanced data — combined with the results of the Chi-Square tests, suggests that feature interactions significantly influence the outcome. Therefore, a method that can implicitly learn such interactions, such as an SVM, may yield better results.

Method 2: Binary classification with SVM

To ensure consistency, I use the same input data with a stratified split, along with the same SMOTE-augmented and standardized training set as before. The initial SVM run with an RBF kernel produced results similar to logistic regression. To improve performance, I defined a grid for tuning: *cost = c(0.1, 1, 10, 100), gamma = c(0.001, 0.005, 0.01, 0.05, 0.1)*

Upon testing the model's execution time, I decided to use 5-fold cross-validation. This setup is feasible given my machine's capacity, the sample size, and is sufficient for light tuning and model comparison, though production-level models typically use 10 folds. I saved the best-performing model to a *.rds* file for later retrieval and evaluation.

I conducted two experiments with SVM tuning: one yielded unexpectedly weaker results than the fixed parameters (cost = 1, gamma = 0.1), despite that combination being included in the grid; the other exceeded the projected time by more than double. The best-performing SVM achieved a balanced accuracy of 0.53908—well behind the baseline logistic regression. I then tried a polynomial kernel, which, while theoretically capable of capturing interactions, failed to distinguish between classes.

As the data pipeline appears correct and complete, I suspect the non-linear interactions are too complex and simultaneously too weak for SVM to model effectively. Ensemble methods such as XGBoost may perform better. Dimensionality reduction (e.g., PCA) and further feature engineering could help reduce noise and enhance signal. A more radical approach would be one-class classification, as the minority class is rare and potentially noisy, and I cannot fully trust negative labels. High recall is essential. I look forward to continuing work on this challenge.

Bibliography:

Apicella, A., Isgrò, F., & Prevete, R. (2024). *Don't Push the Button! Exploring Data Leakage Risks in Machine Learning and Transfer Learning*. https://doi.org/10.2139/ssrn.4733889

Bhandari, H. N., Rimal, B., Pokhrel, N. R., Rimal, R., Dahal, K. R., & Khatri, R. K. C. (2022). Predicting stock market index using LSTM. *Machine Learning with Applications*, *9*, 100320. https://doi.org/10.1016/j.mlwa.2022.100320

Hochreiter, S. (1998). The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *6*, 107–116. https://doi.org/10.1142/S0218488598000094

Kim, Y.-S., Kim, M. K., Fu, N., Liu, J., Wang, J., & Srebric, J. (2025). Investigating the impact of data normalization methods on predicting electricity consumption in a building using different artificial neural network models. *Sustainable Cities and Society*, *118*, 105570. https://doi.org/10.1016/j.scs.2024.105570

Kroujiline, D., Gusev, M., Ushanov, D., Sharov, S. V., & Govorkov, B. (n.d.). *Forecasting stock market returns over multiple time horizons*.

Liu, F., Chen, L., Zheng, Y., & Feng, Y. (2022). A Prediction Method with Data Leakage Suppression for Time Series. *Electronics*, *11*(22), Article 22. https://doi.org/10.3390/electronics11223701

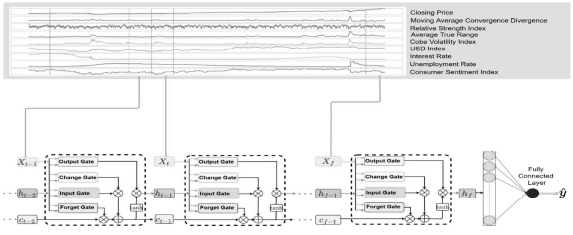
Loan Default Prediction Dataset. (n.d.). Retrieved 18 March 2025, from https://www.kaggle.com/datasets/nikhil1e9/loan-default

Appendix A

Table 1
List of potential features for the model.

Data	Source	Frequency	Abbreviation
Fundamental			
Open price	Yahoo	Daily	···
Close price	Yahoo	Daily	···
Macroeconomic			
Cboe volatility index	Yahoo	Daily	VIX
Interest rate	FRED	Daily	EFFR
Civilian unemployment rate	FRED	Monthly	UNRATE
Consumer sentiment index	FRED	Monthly	UMCSENT
US dollar index	Yahoo	Daily	USDX
Technical indicator			
Moving average convergence divergence	···	Daily	MACD
Average true range	···	Daily	ATR
Relative strength index	···	Daily	RSI

Appendix B



Appendix C

Algorithm 1 *(Pseudo Code for Hyperparameter Tuning Procedure). Input Preparation: Split train and validation data sets and create input of the form [#observations, time step, #features]*
Input: [#observations, time step, #features]; choices of optimizers, learning rates, and batch sizes.
Initialize: Set number of epochs sufficiently large and patience = 5

For *"choice of optimizers"*, **Do**
  For *"choice of learning rates"*, **Do**
    For *"choice of batch sizes"*, **Do**
      For *"range of number of replicates"*, **Do**
        Train the model, monitor validation loss;
        Continue *Until training loss at epoch* $n \leq$ *training loss*
        at epoch $n+1 \leq \cdots \leq$ *training loss at epoch* $n+4$*, Or maximum epochs are reached.*
        Evaluate model on the validation data.
        Calculate RMSE scores.
      End Do.
      Calculate average RMSE scores.
    End Do.
  End Do.
End Do.
Output *Set of best hyperparameters, average RMSEs, best average RMSE.*

Appendix D

Algorithm 2 *(Pseudo Code for LSTM Model after Hyperparameter Tuning). Input Preparation: Split train and test data sets and create input of the form (#observations, time step, #features)*
Input: [#observations, time step, #features]; chosen hyperparameters (optimizer, learning rate, batch size) obtained from Algorithm 1 for each model.
Initialize: Set number of epochs sufficiently large and patience = 5

For *"choice of layers and neurons"*, **Do**
  For *"range of number of replicates"*, **Do**
    Train the model, monitor training loss;
    Continue *Until training loss at epoch* $n \leq$ *training loss*
    at epoch $n+1 \leq \cdots \leq$ *training loss at epoch* $n+4$*, Or maximum epochs are reached.*
    Evaluate model on the test data.
    Calculate RMSE, MAPE and R scores.
  End Do.
  Calculate minimum, maximum, average and standard deviation of RMSE, MAPE and R scores.
  Save key results in respective files.
End Do.

Appendix E