FidelFolio Investments: Deep Learning models for Financial Market Capitalization Growth Prediction

Team Bitmask

April 18, 2025

Abstract

FidelFolio Investments initiated this project to explore relationship between fundamental financial indicators and future market performance using deep learning methodologies. The primary objective was to develop robust predictive models capable of forecasting market capitalization growth for Indian listed Companies over multiple time horizons (1-year, 2-year, and 3-year). Such models aim to enhance investment research and support data-driven decision-making processes. Four distinct modeling approaches were investigated: a Baseline Long Short-Term Memory (LSTM) model, an Improved LSTM variant with enhanced regularization and training stability, a Multi-Layer Perceptron (MLP) model and an Encoder-Decoder LSTM architecture. This report details the dataset, preprocessing pipeline, model architectures, experimental setup, evaluation strategy, and comparative results of these approaches.

Contents

Chapter 1

Introduction

1.1 Background and Motivation

Predicting financial market movements, particularly metrics like market capitalization growth, is a challenging yet crucial task for investment firms. Traditional financial modeling often relies on linear assumptions or simpler time-series techniques, which may not fully capture the intricate dynamics present in financial data.

The primary motivation behind this work is to develop a predictive tool that can assist investment analysts and portfolio managers by providing quantitative forecasts of market capitalization growth over short-to-medium term horizons (1, 2, and 3 years). Accurate predictions can significantly enhance strategic investment decisions, risk assessment, and portfolio allocation.

1.2 Objectives

The core objectives of this project were:

- To curate and preprocess a comprehensive dataset of financial indicators for Indian listed companies over a significant time period.
- To design, implement and evaluate performace of multiple deep learning models capable of handling sequential financial data and predicting multi-horizon market capitalization growth.
- To provide a detailed technical overview of the best-performing models and their potential application in investment research.

Chapter 2

Dataset and Preprocessing

2.1 Dataset Overview

The dataset employed in this study comprises financial information for 2,796 unique publicly listed Indian companies. The data spans the period from 1999 to 2024, resulting in a total of 24,751 observations (company-year pairs). Each observation contains a set of unnamed fundamental financial features for a specific company in a given fiscal year.

The target variables for prediction are the 1-year, 2-year, and 3-year forward market capitalization growth rates, calculated relative to the current year's market capitalization.

Significant variations in the range of certain features were noted, potentially indicating the presence of outliers or features with inherently large scales, which could impact model reliability if not addressed.

Table 2.1: Representative Statistics of Key Features

Statistic	Feature 1	Feature 2	Target 1Y	Target 3Y
Count	20914	21999	22934	19606
Mean	148.094	17.857	15.418	34.888
Std Dev	1985.689	15.149	87.923	168.034
Min	0.01	-109.44	-178.49	-344.27
Median	52.000	14.830	-4.255	-13.685
Max	227428.00	444.80	988.17	995.22
Range	227427.99	554.24	1166.66	1339.49

2.2 Preprocessing Pipeline

A multi-step preprocessing pipeline was implemented to prepare the data for deep learning models. The key steps are outlined below:

1. **Data Cleaning**: Initial cleaning involved handling inconsistent column names (if any) and ensuring all relevant financial features were converted to numeric types.
2. **Missing Value Imputation**: Missing values were addressed using a two-stage approach suitable for panel data:
    - Forward Fill: Within each company's time series, missing values were filled using the last observed valid value (forward fill) to preserve temporal consistency.
    - KNN Imputation: Any remaining missing values (e.g., at the beginning of a company's series) were imputed using a K-Nearest Neighbors (KNN) imputer with $k = 5$, leveraging information from similar company-year instances.

3. **Outlier Handling**: To mitigate the impact of extreme values, Winsorization was applied at the 1st and 99th percentiles. This approach caps outliers at these percentile values (equivalent to capping at approximately ±3 z-scores for normally distributed data), reducing their influence without removing the observations.

4. **Feature Normalization**: Although initial analysis suggested features were somewhat normalized, Standard Scaling (z-score normalization) was applied consistently across all features to ensure a mean of 0 and a standard deviation of 1, which benefits many gradient-based optimization algorithms. The formula for z-score normalization is:

$$z = \frac{x - \mu}{\sigma}$$

where x is the original feature value, μ is the mean of the feature, and σ is its standard deviation.

5. **Dimensionality Reduction (PCA)**: Principal Component Analysis (PCA) was applied to the feature set to reduce dimensionality, remove multicollinearity, and potentially reduce noise. Components were retained to explain 95% of the variance in the original feature space. This significantly reduced the number of input features from 56 to 6 principal components.

6. **Sequence Preparation**: For sequence models (LSTMs), the data was restructured into fixed-length sequences. This involved:

    - Sorting the data chronologically by company and year.
    - Creating overlapping lookback windows (sequences) of financial data (features or PCA components) for each company. The length of the lookback window (`max_seq_len`) is a hyperparameter.
    - Aligning each input sequence with its corresponding future target values (1Y, 2Y, 3Y growth rates). Sequences that did not have valid future targets were handled appropriately (e.g., excluded or padded).

7. **Categorical Feature Encoding**: Company identifiers (e.g., names or IDs) were label encoded into numerical format. These numerical IDs were then used as input to embedding layers within the models to learn company-specific representations.

8. **Train-Validation Split**: The dataset was split into training,test and validation sets using 70:15:15 ratio. Crucially, this split was performed chronologically to prevent data leakage from the future into the past, mimicking a real-world forecasting scenario. The validation set typically consisted of the most recent years in the dataset prior to the test periods used in walk-forward validation.

Chapter 3

Methodology and Model Architectures

3.1 Overall Approach

To address the prediction task, we explored four distinct deep learning architectures. The core idea behind using multiple models was to explore and evaluate the trade-offs between sequence modeling capabilities, computational efficiency, and predictive accuracy for this specific financial dataset, and ultimately find the best model for the task.

A common theme across several models was the use of a hybrid approach combining:

- **Company-specific Embeddings**: Learned representations for each unique company, capturing static, intrinsic characteristics not fully reflected in the time-varying financial features.
- **Dense (Fully Connected) Layers**: Used for feature transformation, integration of different input streams (e.g., sequence output and embeddings), and final prediction generation.

This hybrid structure allows models to potentially learn both cross-sectional patterns (differences between companies via embeddings) and longitudinal patterns (temporal trends within a company's history via LSTMs).

3.2 Model 1: Baseline LSTM

The first approach served as a baseline, utilizing a standard LSTM architecture. This model aims to capture temporal patterns directly from the sequence of historical financial features. It incorporates a company ID embedding, concatenated with the LSTM output, to provide static context about the specific company being predicted.

3.2.1 Architecture

The model takes two primary inputs:

- **Sequence Input**: A tensor of shape `(batch_size, max_seq_len, num_features)` representing the time series of financial features (or PCA components).
- **Company ID Input**: A tensor of shape `(batch_size, 1)` containing the label-encoded company IDs.

The architecture consists of the following layers:

1. **Company Embedding Layer**: Maps the input company IDs to dense vectors of dimension 10. `Embedding(input_dim=num_companies, output_dim=10)` followed by `Flatten()`.

2. **LSTM Layer**: Processes the input sequence. The number of LSTM units is a hyperparameter (e.g., 64 or 128). It returns the output sequence or just the final hidden state. `LSTM(units=...)`.

3. **Concatenation Layer**: Merges the flattened company embedding vector with the output of the LSTM layer.

4. **Dense Layer**: A fully connected layer with ReLU activation (e.g., 32 units) for further feature transformation. `Dense(32, activation='relu')`.

5. **Output Layer**: A dense layer with 3 units (no activation function for regression) to predict the 1Y, 2Y, and 3Y growth rates simultaneously. `Dense(3)`.

3.3 Model 2: Improved LSTM

This model builds upon the baseline LSTM by incorporating several enhancements aimed at improving training stability and generalization. Key improvements include target variable scaling, dropout regularization, and a potentially adjusted learning rate. Target scaling helps normalize the loss landscape, while dropout prevents overfitting by randomly setting a fraction of neuron activations to zero during training.

3.3.1 Architecture

The core architecture is identical to the Baseline LSTM, with the following additions and modifications:

- **Target Scaling**: The target variables (1Y, 2Y, 3Y growth rates) are scaled using 'StandardScaler' before training. Predictions are inverse-transformed back to the original scale for evaluation.

- **Dropout Layers**: Dropout layers (with rate = 0.25) are inserted after the LSTM layer and the intermediate Dense layer to provide regularization. 'Dropout(0.25)'.

- **Potentially Tuned Hyperparameters**: Learning rate might be lowered (e.g., to 0.0001), and training epochs/patience for early stopping might be increased compared to the baseline.

3.4 Model 3: MLP

This model represents a departure from recurrent architectures, adopting a Multi-Layer Perceptron (MLP) approach commonly used for tabular data. Separate models are trained for each prediction horizon (1Y, 2Y, 3Y).

3.4.1 Architecture

The model takes two inputs per sample:

- **Flattened Sequence Input**: The sequence of PCA components over the lookback window is flattened into a single vector `(batch_size, max_seq_len * num_pca_components)`. Padding is likely used for sequences shorter than `max_seq_len`.

- **Company ID Input**: Processed through an embedding layer, similar to the LSTM models, but potentially with a different dimension (e.g., 16). `Embedding(input_dim=num_companies, output_dim=16)` followed by `Flatten()`.

The architecture for *each* target horizon model is:

1. **Input Concatenation**: The flattened sequence vector and the flattened company embedding vector are concatenated.
2. **Dense Layer 1**: Fully connected layer with ReLU activation (e.g., 128 units) followed by Dropout (e.g., rate 0.3). `Dense(128, activation='relu')`, `Dropout(0.3)`.
3. **Dense Layer 2**: Fully connected layer with ReLU activation (e.g., 64 units) followed by Dropout (e.g., rate 0.3). `Dense(64, activation='relu')`, `Dropout(0.3)`.
4. **Output Layer**: A single dense unit (no activation) predicting one specific target (1Y, 2Y, or 3Y growth). `Dense(1)`.

3.5 Model 4: Encoder-Decoder LSTM

This architecture uses an LSTM as an encoder to create a fixed-size context vector representing the historical information. This context vector is then combined with the company embedding and fed into dense layers to make the prediction. Similar to the MLP model, separate models are trained for each target horizon.

3.5.1 Architecture

Inputs are similar to the baseline LSTM:

- **Sequence Input**: Time series of PCA-reduced financial features `(batch_size, max_seq_len, num_pca_components)`. Masking is likely used to handle padded steps.
- **Company ID Input**: Processed via an embedding layer.

The architecture consists of an Encoder and a Decoder part for *each* target horizon model:

1. **Encoder LSTM**: An LSTM layer (e.g., 128 units) processes the sequence input. It's configured to return only the last hidden state (and possibly the last cell state). `LSTM(128, return_state=True)`. The final hidden state is the context vector.
2. **Company Embedding Layer**: As in other models, maps company IDs to vectors and flattens them.
3. **Decoder Input Preparation**: The context vector from the encoder is concatenated with the flattened company embedding vector.
4. **Decoder Dense Layer 1**: Fully connected layer (e.g., 64 units) with ReLU activation, followed by Dropout (e.g., 0.25). `Dense(64, activation='relu')`, `Dropout(0.25)`.
5. **Decoder Output Layer**: A single dense unit (no activation) to predict the specific target (1Y, 2Y, or 3Y growth). `Dense(1)`.

Chapter 4

Experimental Setup

4.1 Training Configuration

While specific hyperparameters might vary slightly between models, a general training configuration was adopted, with notable deviations highlighted per model.

4.1.1 Common Settings

- **Optimizer**: Adam optimizer with its default learning rate initially (0.001) and a lower rate (e.g., 1e-4 or 0.0001) for more stable convergence in improved models.
- **Loss Function**: Mean Squared Error (MSE) as the loss function.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

  where N is the number of samples, y_i is the true value, and $\hat{y}_i$ is the predicted value.
- **Evaluation Metric**: Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) were used for monitoring performance during training and for final evaluation. RMSE is reported on the original scale of the target variables.

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

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

- **Batch Size**: A batch size of 32 was employed.
- **Epochs and Early Stopping**: Models were trained for a substantial number of epochs (e.g., 50, 60, 75) with early stopping implemented to prevent overfitting.

4.1.2 Model-Specific Settings

- **Baseline LSTM**: Adam (LR=0.001), MSE loss, MAE metric, Batch=32, Epochs=50, Patience=7, Validation Split=15% during initial training/tuning phase.
- **Improved LSTM**: Adam (LR=0.0001), MSE loss, MAE metric, Batch=32, Epochs=60, Patience=10, Dropout=0.25, Target Scaling applied.

- **MLP**: Adam, MSE loss. Specific LR, Epochs, Patience, Dropout=0.3 mentioned. Trained separately per target.
- **Encoder-Decoder LSTM**: Adam (LR=1e-4), MSE loss, RMSE metric, Batch=32, Epochs=75, EarlyStopping(Patience=12), ReduceLROnPlateau(Patience=6). Trained separately per target.

4.2 Evaluation Strategy: Walk-Forward Validation

Given the time-series nature of the data, a standard k-fold cross-validation is inappropriate as it can lead to training on future data to predict the past. Instead, an expanding window walk-forward validation approach was employed.
The process works as follows:

1. Start with an initial training period (e.g., data up to year Y).
2. Train the model(s) on this initial period.
3. Validate on a small hold-out set if needed for hyperparameter tuning (e.g., year $Y + 1$).
4. Test the trained model on the next time period (e.g., year $Y + 2$). Record performance metrics (RMSE, MAE) for this fold/year.
5. Expand the training window to include the test period data (i.e., train on data up to year $Y + 2$).
6. Retrain the model(s) on the expanded dataset.
7. Test on the subsequent period (e.g., year $Y + 3$). Record metrics.
8. Repeat steps 5-7, progressively expanding the training window and testing on the next unseen year, until the end of the dataset is reached.

This ensures that predictions are always made on data that occurs chronologically after the training data, mimicking a realistic deployment scenario.

Final performance metrics (RMSE, MAE) are averaged across all valid test folds (years), excluding any folds where predictions could not be made (e.g., due to insufficient data leading to NaNs). For models involving target scaling (like Improved LSTM and Encoder-Decoder), predictions and actuals were inverse-transformed back to their original scale before calculating the final RMSE and MAE metrics for reporting.

Chapter 5

Results

Performance is primarily reported using Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) on the original, unscaled target variables (1Y, 2Y, 3Y market capitalization growth).

5.1 Model 1: Baseline LSTM Results

The Baseline LSTM model predicted all three target horizons simultaneously. The overall performance, aggregated across all valid test years in the walk-forward validation, was:

- **Target 1 (1Y Growth)**:
  - Final Overall RMSE: 129.8627
- **Target 2 (2Y Growth)**:
  - Final Overall RMSE: 280.0713
- **Target 3 (3Y Growth)**:
  - Final Overall RMSE: 464.2194

5.2 Model 2: Improved LSTM Results

The Improved LSTM, incorporating target scaling and regularization, also predicted all targets simultaneously but performance was reported per target after inverse transformation:

- **Target 1 (1Y Growth)**:
  - Final Overall RMSE: 126.5355
  - Final Overall MAE: 57.1987
- **Target 2 (2Y Growth)**:
  - Final Overall RMSE: 276.1869
  - Final Overall MAE: 108.4222
- **Target 3 (3Y Growth)**:
  - Final Overall RMSE: 472.0414
  - Final Overall MAE: 181.0718

A significant improvement over the baseline is observed, particularly evident when comparing individual target errors (though the baseline combined them). The target scaling and regularization appear effective.

5.3 Model 3: MLP with PCA & Feature Differences Results

The MLP model, trained separately for each target horizon using flattened PCA components and engineered features, yielded the following overall results:

- **Target 1 (1Y Growth)**:
  - Final Overall RMSE: 1402.8884
  - Final Overall MAE: 285.0588
- **Target 2 (2Y Growth)**:
  - Final Overall RMSE: 2674.4482
  - Final Overall MAE: 583.3290
- **Target 3 (3Y Growth)**:
  - Final Overall RMSE: 3930.2122
  - Final Overall MAE: 952.1766

The performance of this MLP approach appears significantly worse than both LSTM models, indicated by substantially higher RMSE and MAE values across all horizons. This suggests that flattening the time series and using a standard MLP might lose crucial temporal information captured effectively by LSTMs for this dataset.

5.4 Model 4: Encoder-Decoder LSTM Results

The Encoder-Decoder LSTM, also trained separately for each target horizon, achieved the following overall performance:

- **Target 1 (1Y Growth)**:
  - Final Overall RMSE: 126.9015
  - Final Overall MAE: 58.1066
  - Total Valid Prediction Pairs: 15044
- **Target 2 (2Y Growth)**:
  - Final Overall RMSE: 277.8013
  - Final Overall MAE: 111.4502
  - Total Valid Prediction Pairs: 15044
- **Target 3 (3Y Growth)**:
  - Final Overall RMSE: 478.0079
  - Final Overall MAE: 187.8028
  - Total Valid Prediction Pairs: 15044

The performance of the Encoder-Decoder LSTM is highly competitive and very close to that of the Improved LSTM model across all three horizons.

5.5 Comparative Analysis

To facilitate comparison, the key performance metrics (RMSE) are summarized in Table 5.1.

Table 5.1: Summary of Model Performance (Overall RMSE on Original Scale)

Model	RMSE (1Y Growth)	RMSE (2Y Growth)	RMSE (3Y Growth)
Baseline LSTM (Combined)	129.8627	280.0713	464.2194
Improved LSTM	126.5355	276.1869	472.0414
MLP (PCA + Feat. Diffs)	1402.8884	2674.4482	3930.2122
Encoder-Decoder LSTM	126.9015	277.8013	478.0079

Key observations from the comparison:

- The MLP model performed significantly worse than the LSTM-based approaches, highlighting the importance of sequential modeling for this task.
- Both the Improved LSTM and the Encoder-Decoder LSTM showed substantial improvements over the Baseline LSTM.
- The Improved LSTM and Encoder-Decoder LSTM achieved very similar performance levels, suggesting both are viable architectures. The Improved LSTM has the potential advantage of predicting all horizons simultaneously, while the Encoder-Decoder might offer slightly more flexibility by having separate models per horizon.
- As expected, prediction error (RMSE/MAE) generally increases for longer prediction horizons (3Y $>$ 2Y $>$ 1Y) across the better-performing models.

Chapter 6

Conclusion

6.1 Summary of Findings

This project successfully developed and evaluated multiple deep learning models for predicting 1-year, 2-year, and 3-year forward market capitalization growth for Indian listed companies, utilizing a dataset spanning from 1999 to 2024.

The key findings are:

- Recurrent architectures (LSTMs) significantly outperformed a non-recurrent MLP approach that relied on flattened sequences and feature engineering, emphasizing the value of explicitly modeling temporal dependencies in financial data.
- Enhancements to the baseline LSTM, such as target variable scaling, dropout regularization, and careful hyperparameter tuning (as seen in the Improved LSTM), led to substantial performance gains.
- The Encoder-Decoder LSTM architecture also proved highly effective, yielding results comparable to the Improved LSTM. This demonstrates the utility of using an LSTM encoder to create a condensed representation of historical context for prediction.

These models demonstrate the potential of deep learning techniques to provide valuable quantitative insights for investment decision-making processes at FidelFolio Investments.

6.2 Limitations

Several limitations should be acknowledged:

- **Feature Space**: The specific nature of the unnamed financial features was not detailed in the source material; understanding their exact definitions could allow for more targeted feature engineering or interpretation. The high range of some features also warrants caution.
- **Market Regimes**: The models were trained across a long period (1999-2024) which includes various market conditions. Model performance might vary significantly under different market regimes not equally represented in the training data.

- **Questionable Data Quality**: The data on which the models have been trained might not be representative of actual company markers, and hence performance on real data might be different. Additionally, the dataset used has features with very high variance, which means that number of outliers in the data is high, distorting the true dependence of target on feature.
- **Exogenous Factors**: The models rely solely on historical fundamental data and company IDs. They do not explicitly incorporate macroeconomic indicators, market sentiment, news events, or other external factors that can influence market capitalization.
- **Interpretability**: While effective, deep learning models, particularly LSTMs, can be complex to interpret fully ('black box' problem), potentially hindering trust or diagnostic capabilities compared to simpler models.
- **PCA Information Loss**: While PCA reduces dimensionality, retaining 95% variance means 5% of the variance is discarded, which might contain subtle predictive signals.