Contrastive Learning for Financial Market Regime Prediction

by

Abhipal Sharma(as20410)

Akshat Singh(as20255)

Palak Gupta(pg2820)

Computer Science Engineering

New York University, Tandon School Of Engineering

Fall, 2024

Contents

Chapter 1

Introduction

1.1 Background

1.1.1 History

Financial markets have long been a cornerstone of global economic systems, facilitating capital flow, wealth creation, and economic stability. However, these markets are highly dynamic, driven by a complex interplay of macroeconomic factors, geopolitical events, and human behavior. Predicting market trends has always been challenging due to this inherent complexity.

Financial markets often operate within distinct regimes, each with unique characteristics and implications for economic activity. **Bull markets** are marked by optimism, rising prices, and economic growth, while **bear markets** reflect pessimism, declining prices, and economic contraction. **Volatile markets** are characterized by erratic price movements and heightened uncertainty, presenting challenges for investors and analysts alike. Traditionally, market analysis has relied on approaches such as **technical analysis** (e.g., chart patterns,

moving averages) and **fundamental analysis** (e.g., financial ratios, earnings reports). While these methods provide valuable insights, they are often limited by their inability to handle large, noisy datasets or uncover hidden patterns. Over the past two decades, **machine learning** has transformed financial analysis by providing powerful tools to analyze complex datasets and make accurate predictions. However, supervised ML models rely heavily on labeled data, which is often incomplete, expensive, or unavailable in financial contexts. This limitation has driven the exploration of **unsupervised learning techniques**, which can derive meaningful representations directly from raw data, offering a promising solution to the challenges of modern financial market analysis.

1.1.2 Significance

The ability to predict market regimes has significant real-world applications:

- For Investors: Identifying market trends allows for strategic investment decisions, such as when to buy, hold, or sell.
- For Risk Managers: Understanding market regimes helps mitigate financial risks during periods of volatility or decline.
- For Policymakers: Monitoring market behavior enables timely interventions to stabilize economies and protect stakeholders.

However, predicting market regimes comes with several challenges like:

- Incomplete or Noisy Data: Financial datasets often contain gaps or irregularities.
- High Dimensionality: There are numerous variables (e.g., prices, volumes, macroeconomic indicators) to consider simultaneously.

- Changing Dynamics: Markets are non-stationary, meaning the statistical relationships between variables evolve over time.

1.1.3 The Role of Unsupervised Learning

Unsupervised learning is uniquely suited to tackle the complexities of financial data:

- Contrastive Learning (e.g., SimCLR):
  - Learns to differentiate between similar and dissimilar samples by comparing "positive pairs" (e.g., augmented versions of the same data) and "negative pairs" (e.g., different data points).
  - Reduces reliance on labeled data, which is ideal for financial datasets where labeled examples are scarce or expensive to obtain.
- Representation Learning:
  - Instead of reducing the dimensionality of the data, this project increases it to create richer and more complex feature representations. These high-dimensional embeddings help capture subtle patterns in the data, making them more robust for distinguishing between various market regimes.
  - By focusing on expanding the feature space, the approach emphasizes feature enrichment, which allows the models to handle complex financial datasets more effectively leading to better accuracy and reliability in predicting market trends.
  - By integrating these techniques, it is possible to create a scalable framework that handles the complexities of financial data and gives out accurate market predictions.

1.1.4 Our Goal

This project explores the potential of unsupervised contrastive learning frameworks for predicting market regimes by learning embeddings from financial data without the need for labels. The primary goal is to improve the accuracy and robustness of market regime classification, even when the data is incomplete or noisy, and to test whether these embeddings can generalize across various market conditions. By combining the strengths of unsupervised and supervised learning, the project aims to provide a comprehensive solution to market analysis. Using advanced techniques like **SimCLR**, **LSTM**, and Transformer models, meaningful patterns are extracted from raw financial data, and the resulting embeddings are evaluated through downstream tasks using classifiers such as **Random Forest** and **XGBoost**. This approach demonstrates the power of cutting-edge machine learning techniques applied to real-world financial challenges. Moreover, the implications of this work extend beyond finance, as the methods for handling noisy, incomplete data and generating robust embeddings can be applied to other fields, including healthcare, supply chain optimization, and environmental modeling. By showcasing the feasibility and effectiveness of these techniques, the project contributes to a broader understanding of how unsupervised learning can address challenges in complex, high-dimensional datasets.

1.2 Problem Statement

This project explores the potential of unsupervised contrastive learning embeddings in addressing the market regime prediction problem. In this approach, meaningful representations of the data are learned without the use of labels. The project seeks to assess whether unsupervised contrastive learning can effectively capture latent structures in financial data, improving classification performance in the supervised setting when data is incomplete or missing.

1.3 Methodology

The methodology involves leveraging a deep learning-based framework to learn unsupervised embeddings from financial data, followed by training supervised classifiers using these embeddings for market regime prediction. The steps are as follows:

- Data Preparation
  - The financial dataset is preprocessed, including normalization and preparation for augmentation.
  - Labels are included for downstream supervised tasks but are not used during the unsupervised embedding learning phase.
- Deep Learning-Based Framework
  - The framework comprises:
    - Encoder: A feedforward neural network with a hidden layer and ReLU activation to extract feature representations.
    - Projection Head: A linear layer mapping encoded features to a lower-dimensional embedding space.
    - Decoder: A network reconstructing the original input from the encoder's latent representation to encourage the model to preserve meaningful patterns.
  - A corrupted view generator is employed to create augmented data. This involves randomly masking a fraction of features (based on a corruption rate) and replacing them with noise to create two distinct views of the same data point.
- Loss Function

  The total loss function combines:

  - Contrastive Loss (NT-Xent): Encourages similar embeddings for different views of the same data point while ensuring dissimilarity between embeddings of different data points.
  - Reconstruction Loss: Uses Mean Squared Error (MSE) to measure the accuracy of input reconstruction for both views.

  A weighting factor balances the two losses.

- Training the Deep Learning Framework

  Each training epoch consists of:

  - Generating augmented views (x_i and x_j)
  - Passing both views through the framework to obtain embeddings (z_i and z_j) and reconstructions ($reconstructed_i$ and $reconstructed_j$).
  - Computing the combined loss and performing backpropagation to update model weights.

  The model outputs embeddings capturing latent patterns in the data.

- Supervised Classification

  - The learned embeddings from the deep learning framework are used as input features for three supervised classifiers: Logistic Regression, Random Forest, and XGBoost.
  - Two experimental setups are evaluated:
    * Classifiers trained on embeddings alone.
    * Classifiers trained on a combination of embeddings and original features.
  - In both experimental setups, the model is tested on data where 40% of the test data is randomly masked, and the missing values are imputed using reconstructions of the learned embeddings.

- Evaluation The classifiers are evaluated using standard metrics, such as accuracy, precision, recall, and F1-score, to assess their ability to predict market regimes.

Chapter 2

Data Preparation

2.1 Data Acquisition

Historical price data for a selection of ETFs and stocks was obtained using the yfinance library. The dataset includes ETFs such as SPY, GLD, QQQ, and IWM, alongside stocks like AAPL, AMZN, MSFT, and TSLA. Data was collected for the period between January 2, 2015, and January 1, 2022. Each asset's data was stored as individual CSV files for further processing.

2.2 Data Cleaning and Preprocessing

The ticker column was removed to prepare the data for analysis, and the datasets were aligned to ensure consistency. Missing values were handled by dropping incomplete rows, and exploratory data analysis (EDA) was performed to visualize key features, including adjusted close prices, trading volume, and feature correlations. These steps ensured that the dataset was robust and free from anomalies.

2.3 Feature Engineering

Three important features were engineered to capture meaningful patterns in the data:

- Daily Returns: Percentage changes in adjusted closing prices were computed to capture short-term price movements.
- Volatility: A 20-day rolling standard deviation of daily returns was calculated to quantify market fluctuations.
- Market Regimes: Market conditions were labeled into distinct regimes using specific criteria:
  - Bull Market: A 20-day price increase exceeding 5
  - Bear Market: A 20-day price decrease exceeding 5
  - Sideways Market: Price changes within ± 5
  - Volatile Market: Volatility exceeding 2
  - Crisis Market: Daily returns dropping below -10

Rows labeled as Unknown were excluded to ensure the integrity of the regime classification. The Regime column was then encoded into numerical labels, enabling compatibility with machine learning models. Finally, all the individual CSV files for the various assets were concatenated into one dataframe.

2.4 Data Splitting

To prepare the data for training and evaluation, numerical columns were standardized using a StandardScaler, ensuring uniform feature scaling. The dataset was split into training

and testing subsets using an 80-20 split. Stratified sampling was applied to maintain the distribution of market regimes across both subsets, ensuring a balanced evaluation of model performance.

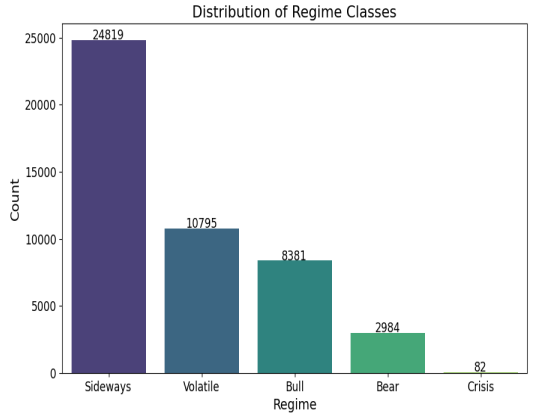

Figure 2.1: Distribution of Regime Classes

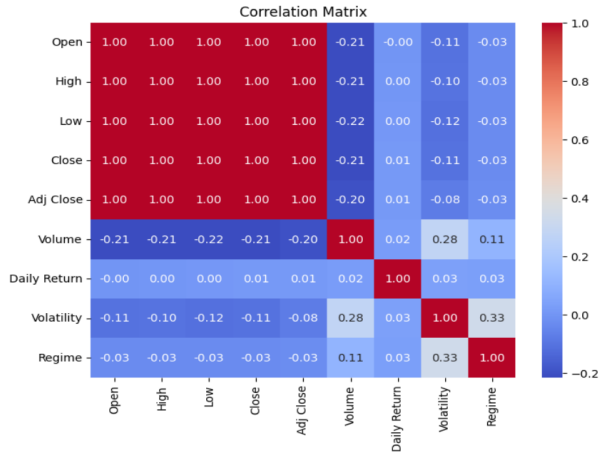

Figure 2.2: Correlation matrix of dataset features

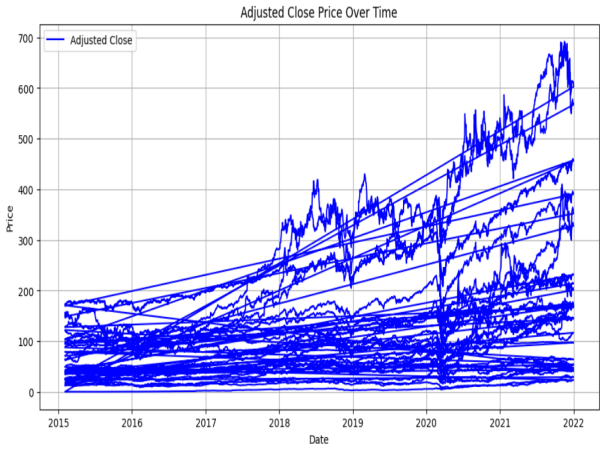

Figure 2.3: Adjusted close over time

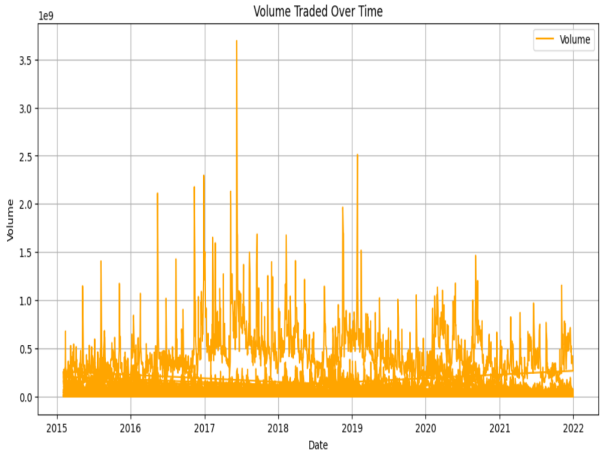

Figure 2.4: Volume traded over time

Chapter 3

Model Training and Evaluation

3.1 Base Model

3.1.1 Model Architectures

- Logistic Regression:

  A simple logistic regression model is employed with a class weight adjustment to handle class imbalance.

- Random Forest Classifier:

  A Random Forest classifier is used with class weight adjustment to handle the class imbalance in the dataset.

- XGBoost Classifier:

  The XGBoost classifier is used with mlogloss as the evaluation metric to optimize the model during training.

3.1.2 Training Process

All the base models are trained on a complete set of features for the training data and labels obtained from section 2.4.

3.1.3 Testing Process

The testing phase involves evaluating the trained models on a dataset with 40% of the test data randomly masked. The masked values are imputed with the column-wise mean of the respective features in the dataset.

3.1.4 Results and Analysis

Model	**Accuracy**	**F1**
Base Logistic Regression	0.35	0.41
Base Random Forest	0.67	0.61
Base XGBoost	0.58	0.61

Table 3.1: Accuracy and F1 across models

Model	**Recall**	**Precision**
Base Logistic Regression	0.35	0.71
Base Random Forest	0.67	0.67
Base XGBoost	0.65	0.59

Table 3.2: Recall and Precision across models

Model	Sideways	Volatile	Bull	Bear	Crisis
Base Logistic Regression	0.35	0.70	0.30	0.15	0.48
Base Random Forest	0.76	0.76	0.15	0.09	0.79
Base XGBoost	0.69	0.75	0.33	0.15	0.35

Table 3.3: Classwise F1

3.2 SimCLR Model

3.2.1 Model Architecture

The SimCLR model consists of three primary components: an encoder, a projection head, and a decoder.

- The encoder is a fully connected layer followed by a ReLU activation to capture features from the input data.
- The projection head is a linear layer that maps the encoded features into a lower-dimensional embedding space.
- The decoder reconstructs the input data from the encoded features to enforce useful feature learning.

3.2.2 Training Process

The SimCLR model was trained for 100 epochs with a batch size of 64 using the contrastive learning methodology described in Section 1.4 on a complete set of features for the training data and labels.

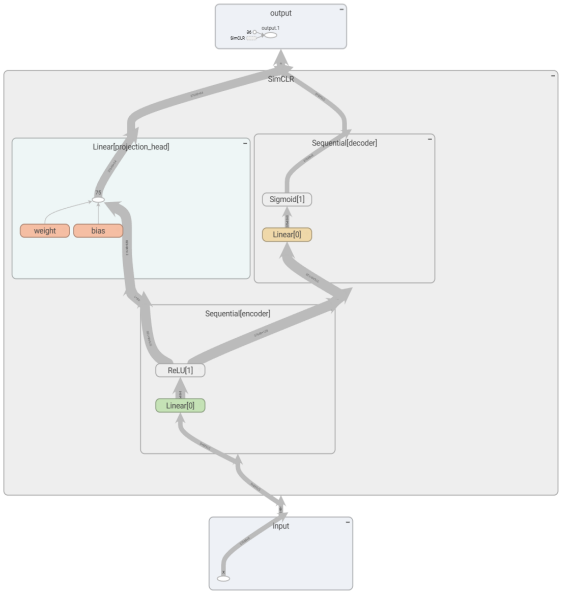

Figure 3.1: SimCLR Model architecture

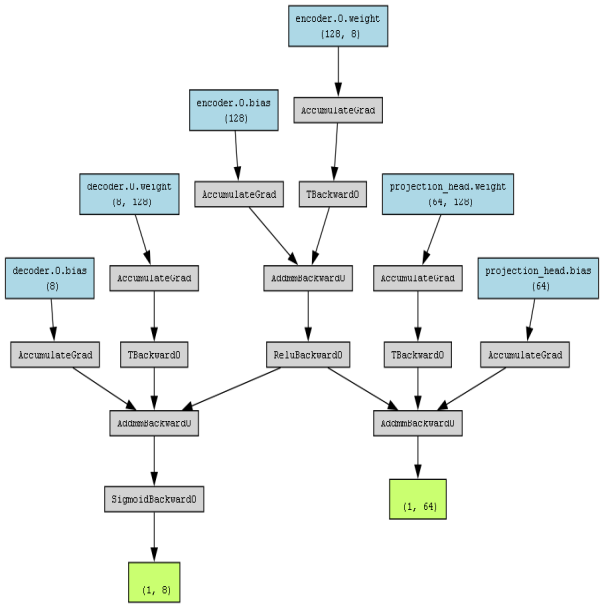

Figure 3.2: SimCLR Computational Graph

3.2.3 Testing Process

- The masked test data is passed through the trained SimCLR model which returns the higher dimension test embeddings along with the reconstructed values that match the dimension of the test data.
- The masked test data is imputed by replacing the masked values with the reconstructed values from the SimCLR model.
- The embeddings produced by the SimCLR model are also captured for use in the downstream task.

3.2.4 Supervised Downstream Classifier

Two categories of classifiers are considered for comparison:

- The trained base Logistic Regression, Random Forest, and XGBoost models are eval-

uated on the imputed test data.

- New Logistic Regression, Random Forest, and XGBoost models are trained on a concatenation of original train data features supplemented with the learned train embeddings and tested on the imputed test data supplemented with the embeddings obtained for the test data.

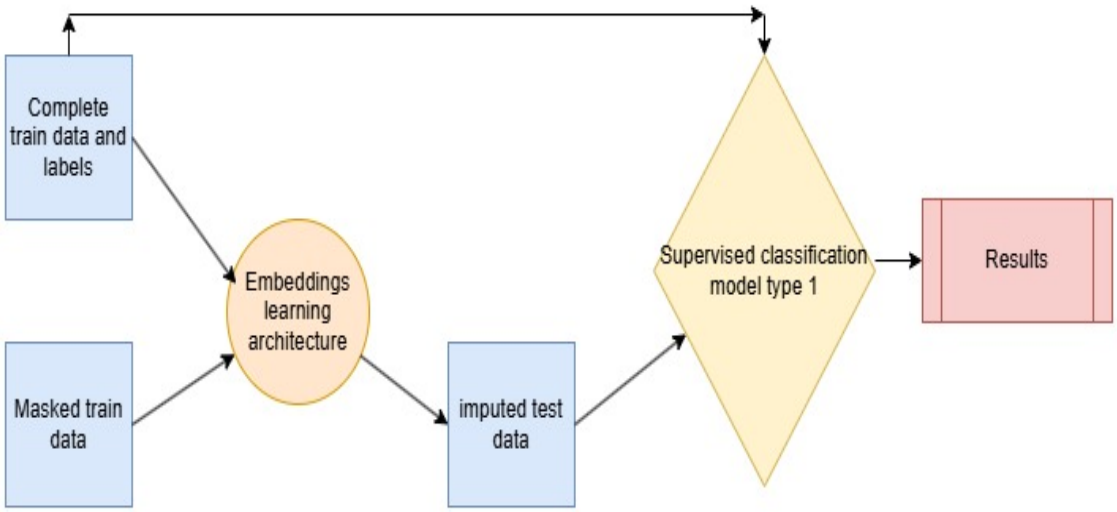

Figure 3.3: Supervised paradigm on imputed test data

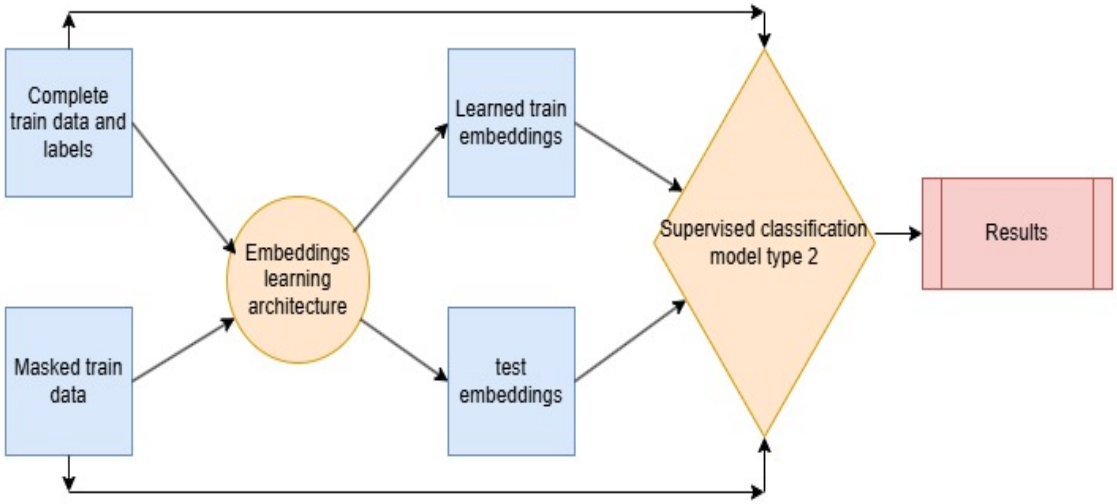

Figure 3.4: Supervised paradigm on imputed test data + embeddings

3.2.5 Results

Model	Accuracy	F1
Base Logistic Regression	0.35	0.41
Base Random Forest	0.67	0.61
Base XGBoost	0.58	0.61
Logistic Regression on imputed test data	0.38	0.45
Random Forest on imputed test data	0.67	0.62
XGBoost on imputed test data	0.59	0.61
Logistic Regression trained on embeddings + original train data	0.42	0.49
Random Forest trained on embeddings + original train data	0.75	0.68
XGBoost trained on embeddings + original train data	0.68	0.68

Table 3.4: Accuracy and F1 across models

Model	Recall	Precision
Base Logistic Regression	0.35	0.71
Base Random Forest	0.67	0.67
Base XGBoost	0.65	0.59
Logistic Regression on imputed test data	0.38	0.71
Random Forest on imputed test data	0.67	0.66
XGBoost on imputed test data	0.60	0.65
Logistic Regression trained on embeddings + original train data	0.43	0.71
Random Forest trained on embeddings + original train data	0.76	0.71
XGBoost trained on embeddings + original train data	0.69	0.68

Table 3.5: Recall and Precision across models

Model	Sideways	Volatile	Bull	Bear	Crisis
Base Logistic Regression	0.35	0.70	0.30	0.15	0.48
Base Random Forest	0.76	0.76	0.15	0.09	0.79
Base XGBoost	0.69	0.75	0.33	0.15	0.35
Logistic Regression on imputed test data	0.38	0.71	0.29	0.15	0.58
Random Forest on imputed test data	0.76	0.76	0.15	0.08	0.79
XGBoost on imputed test data	0.72	0.75	0.28	0.14	0.42
Logistic Regression trained on embeddings + original train data	0.47	0.80	0.26	0.18	0.57
Random Forest trained on embeddings + original train data	0.81	0.99	0.12	0.05	0.79
XGBoost trained on embeddings + original train data	0.74	0.99	0.32	0.16	0.56

Table 3.6: Classwise F1

3.3 LSTM Model

3.3.1 Model Architecture

The LSTM-based model incorporates Long Short-Term Memory (LSTM) layers to process sequential data. The architecture consists of:

- An LSTM encoder to capture temporal dependencies in the input sequence.
- Followed by an embedding layer that projects the LSTM output into a lower-dimensional embedding space.
- The decoder is another LSTM layer that takes the embeddings and reconstructs the sequence.
- Finally, a reconstruction layer is used to map the decoder output back to the input space.

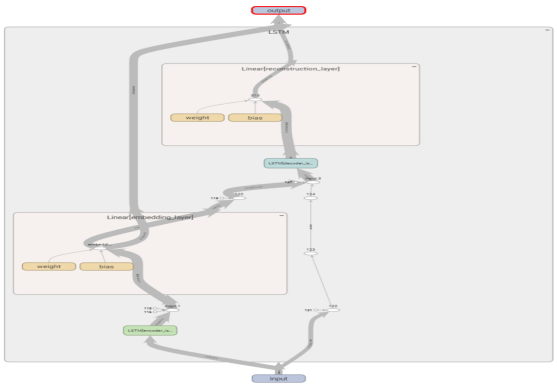

Figure 3.5: LSTM Model architecture

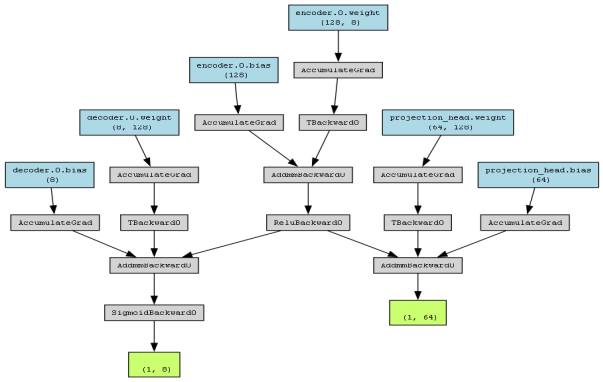

Figure 3.6: LSTM Computational Graph

3.3.2 Training Process

The LSTM model was trained for 100 epochs with a batch size of 64 using the contrastive learning methodology described in Section 1.4 on a complete set of features for the training data and labels.

3.3.3 Testing Process

- The masked test data is passed through the trained LSTM model which returns the higher dimension test embeddings along with the reconstructed values that match the dimension of the test data.
- The masked test data is imputed by replacing the masked values with the reconstructed values from the LSTM model.
- The embeddings produced by the LSTM model are also captured for use in the downstream task.

3.3.4 Supervised Downstream Classifier

Two categories of classifiers are considered for comparison:

- The trained base Logistic Regression, Random Forest, and XGBoost models are evaluated on the imputed test data.
- New Logistic Regression, Random Forest, and XGBoost models are trained on a concatenation of original train data features supplemented with the learned train embeddings and tested on the imputed test data supplemented with the embeddings obtained for the test data.

3.3.5 Results

Model	**Accuracy**	**F1**
Base Logistic Regression	0.35	0.41
Base Random Forest	0.67	0.61
Base XGBoost	0.58	0.61
Logistic Regression on imputed test data	0.36	0.43
Random Forest on imputed test data	0.67	0.61
XGBoost on imputed test data	0.60	0.61
Logistic Regression trained on embeddings + original train data	0.42	0.41
Random Forest trained on embeddings + original train data	0.75	0.68
XGBoost trained on embeddings + original train data	0.69	0.68

Table 3.7: Accuracy and F1 across models

Model	Recall	Precision
Base Logistic Regression	0.35	0.71
Base Random Forest	0.67	0.67
Base XGBoost	0.65	0.59
Logistic Regression on imputed test data	0.37	0.70
Random Forest on imputed test data	0.67	0.66
XGBoost on imputed test data	0.61	0.64
Logistic Regression trained on embeddings + original train data	0.43	0.71
Random Forest trained on embeddings + original train data	0.76	0.71
XGBoost trained on embeddings + original train data	0.69	0.68

Table 3.8: Recall and Precision across models

Model	Sideways	Volatile	Bull	Bear	Crisis
Base Logistic Regression	0.35	0.70	0.30	0.15	0.48
Base Random Forest	0.76	0.76	0.15	0.09	0.79
Base XGBoost	0.69	0.75	0.33	0.15	0.35
Logistic Regression on imputed test data	0.40	0.74	0.30	0.15	0.61
Random Forest on imputed test data	0.76	0.78	0.17	0.09	0.79
XGBoost on imputed test data	0.70	0.77	0.32	0.16	0.42
Logistic Regression trained on embeddings + original train data	0.47	0.80	0.26	0.18	0.57
Random Forest trained on embeddings + original train data	0.81	0.99	0.12	0.05	0.79
XGBoost trained on embeddings + original train data	0.74	0.99	0.32	0.16	0.56

Table 3.9: Classwise F1

3.4 Transformer Model

3.4.1 Model Architecture

The Transformer model consists of three primary components: an encoder, a projection head, and a decoder.

- The encoder is a feed-forward network with 2 attention heads to capture features from the input data.
- The projection head is a linear layer that maps the encoded features into a lower-dimensional embedding space.
- The decoder reconstructs the input data from the encoded features to enforce useful feature learning.

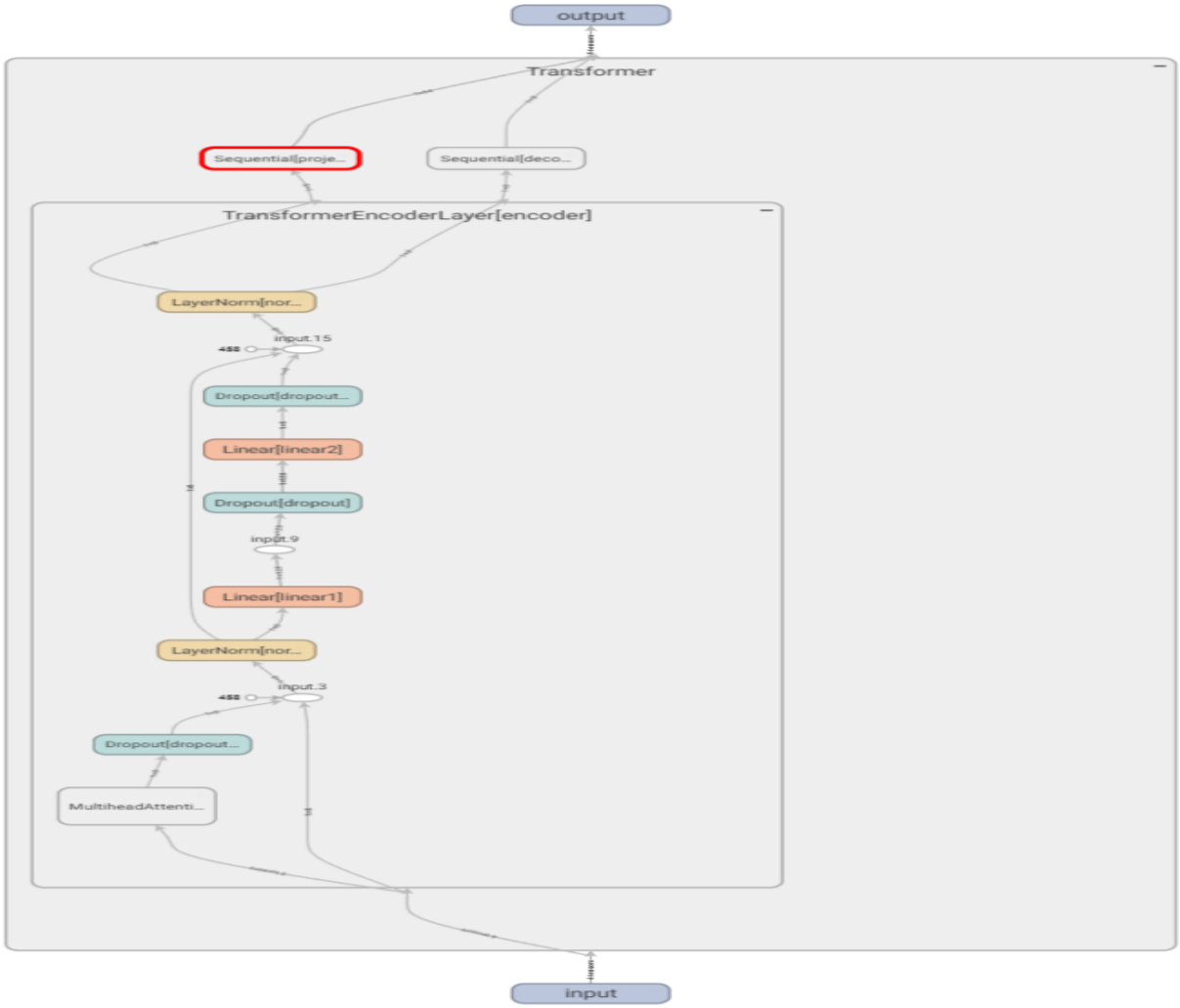

Figure 3.7: Transformer Model architecture

3.4.2 Training Process

The Transformer model was trained for 100 epochs with a batch size of 64 using the contrastive learning methodology described in Section 1.4 on a complete set of features for the training data and labels.

Figure 3.8: Transformer Computational Graph

3.4.3 Testing Process

- The masked test data is passed through the trained Transformer model which returns the higher dimension test embeddings along with the reconstructed values that match the dimension of the test data.
- The masked test data is imputed by replacing the masked values with the reconstructed values from the Transformer model.

- The embeddings produced by the Transformer model are also captured for use in the downstream task.

3.4.4 Supervised Downstream Classifier

Two categories of classifiers are considered for comparison:

- The trained base Logistic Regression, Random Forest, and XGBoost models are evaluated on the imputed test data.
- New Logistic Regression, Random Forest, and XGBoost models are trained on a concatenation of original train data features supplemented with the learned train embeddings and tested on the imputed test data supplemented with the embeddings obtained for the test data.

3.4.5 Results

Model	Accuracy	F1
Base Logistic Regression	0.35	0.41
Base Random Forest	0.67	0.61
Base XGBoost	0.58	0.61
Logistic Regression on imputed test data	0.36	0.34
Random Forest on imputed test data	0.54	0.59
XGBoost on imputed test data	0.49	0.47
Logistic Regression trained on embeddings + original train data	0.47	0.54
Random Forest trained on embeddings + original train data	0.76	0.69
XGBoost trained on embeddings + original train data	0.69	0.69

Table 3.10: Accuracy and F1 across models

Model	Recall	Precision
Base Logistic Regression	0.35	0.71
Base Random Forest	0.67	0.67
Base XGBoost	0.65	0.59
Logistic Regression on imputed test data	0.36	0.58
Random Forest on imputed test data	0.54	0.58
XGBoost on imputed test data	0.49	0.57
Logistic Regression trained on embeddings + original train data	0.48	0.70
Random Forest trained on embeddings + original train data	0.76	0.72
XGBoost trained on embeddings + original train data	0.70	0.69

Table 3.11: Recall and Precision across models

Model	Sideways	Volatile	Bull	Bear	Crisis
Base Logistic Regression	0.35	0.70	0.30	0.15	0.48
Base Random Forest	0.76	0.76	0.15	0.09	0.79
Base XGBoost	0.69	0.75	0.33	0.15	0.35
Logistic Regression on imputed test data	0.29	0.56	0.28	0.15	0.75
Random Forest on imputed test data	0.62	0.59	0.14	0.04	0.83
XGBoost on imputed test data	0.52	0.59	0.29	0.16	0.18
Logistic Regression trained on embeddings + original train data	0.55	0.77	0.31	0.20	0.75
Random Forest trained on embeddings + original train data	0.81	1.00	0.13	0.07	0.83
XGBoost trained on embeddings + original train data	0.76	1.00	0.30	0.21	0.76

Table 3.12: Classwise F1

Chapter 4

Results

4.1 Visualization of Results

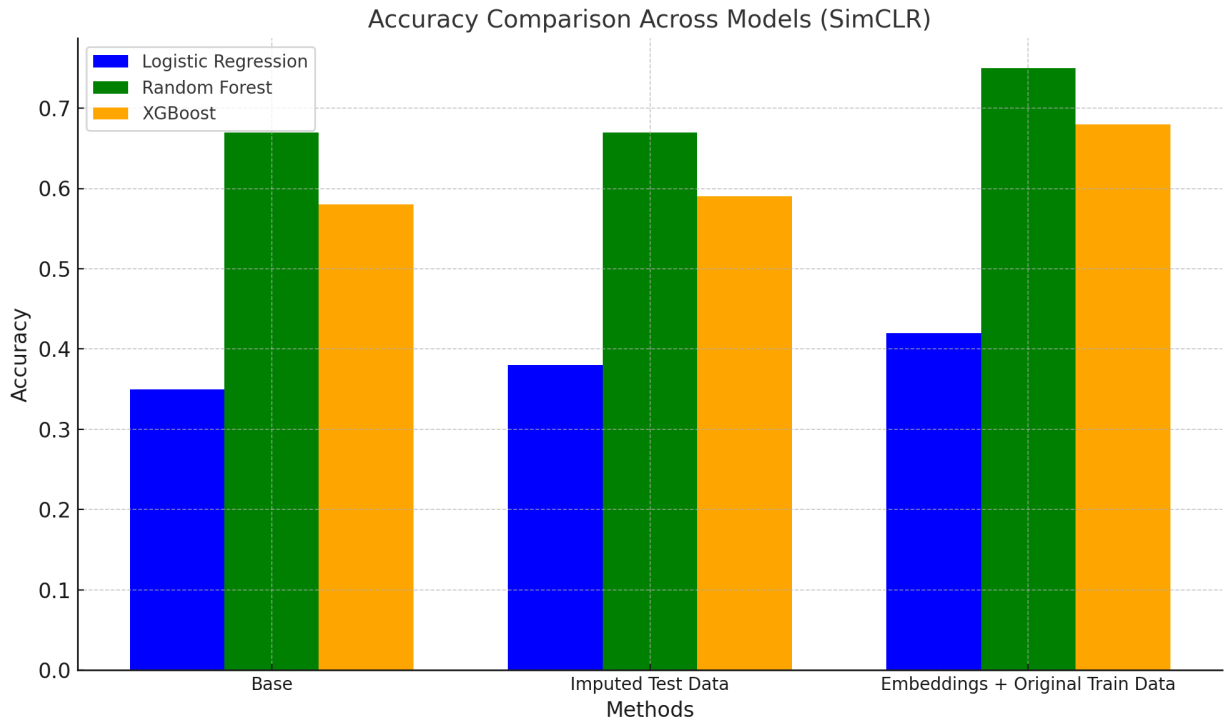

Figure 4.1: Model accuracies comparison for SimCLR embeddings

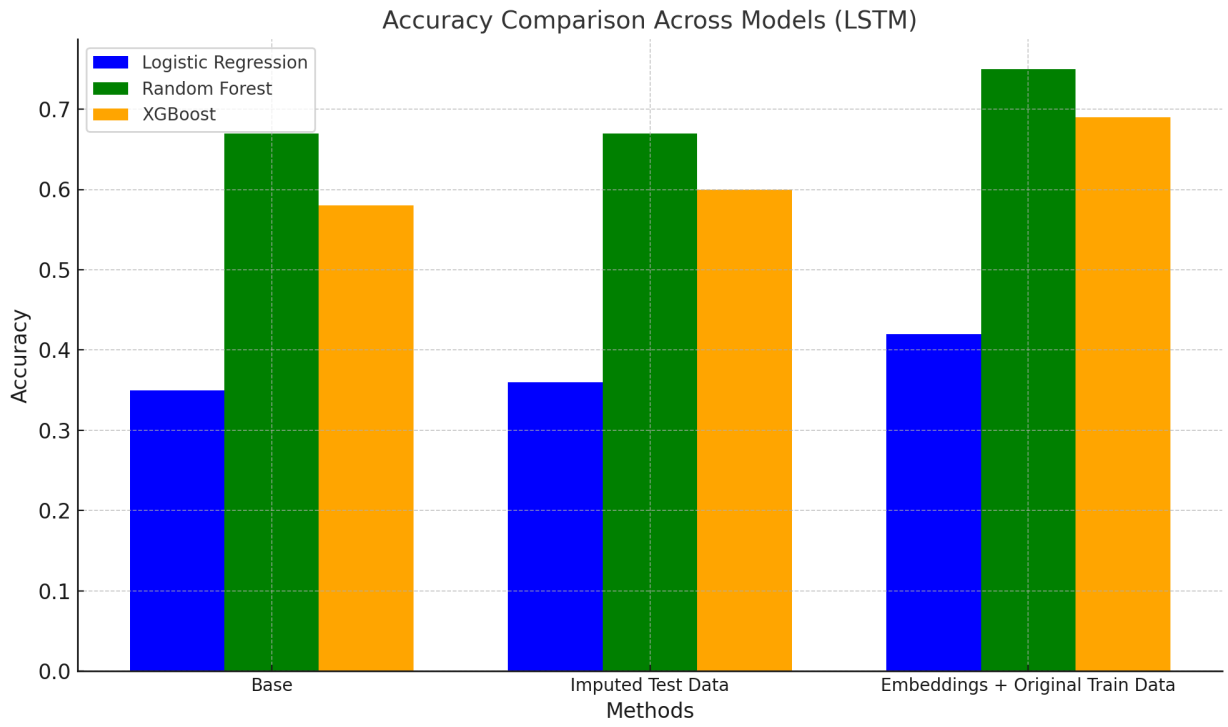

Figure 4.2: Model accuracies comparison for LSTM embeddings

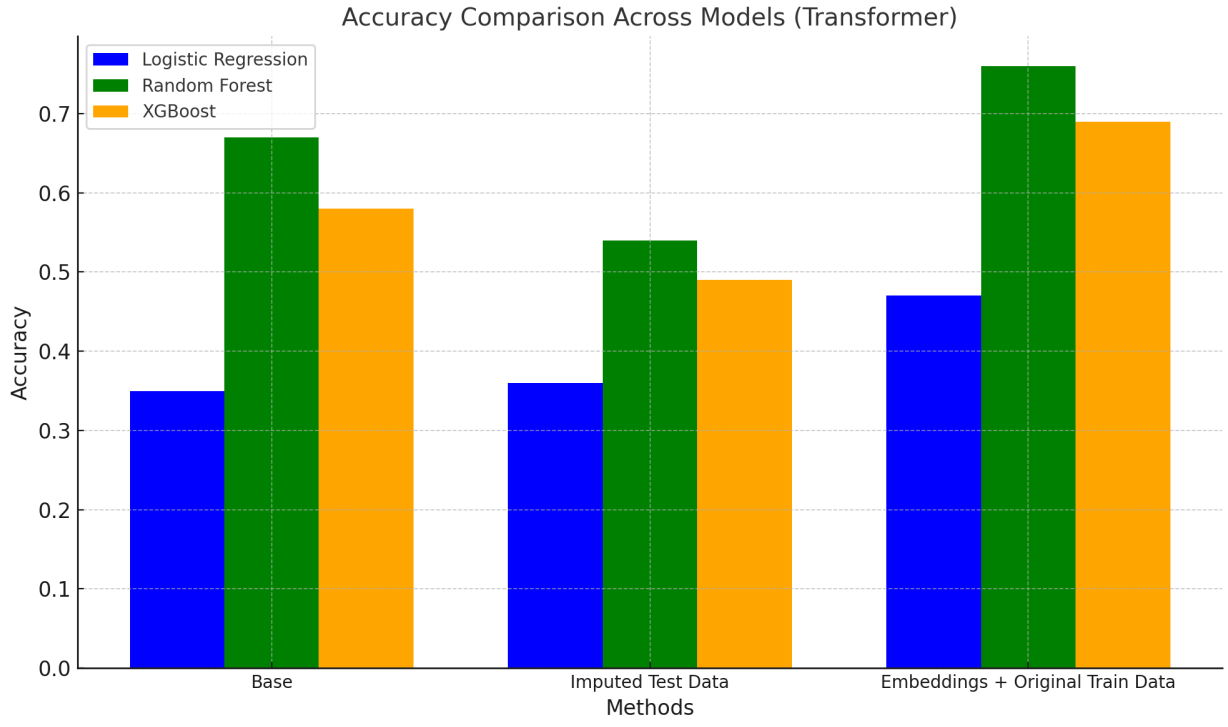

Figure 4.3: Model accuracies comparison for Transformer embeddings

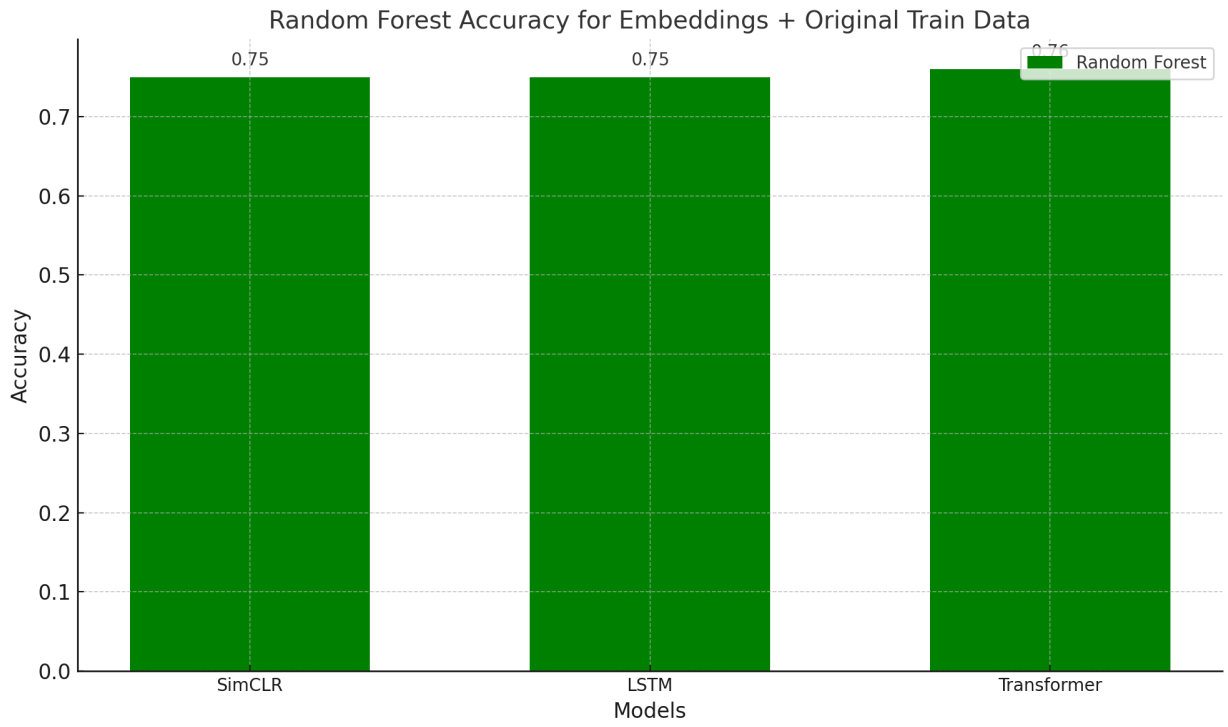

Figure 4.4: Accuracies comparison for the best model of each embedding method

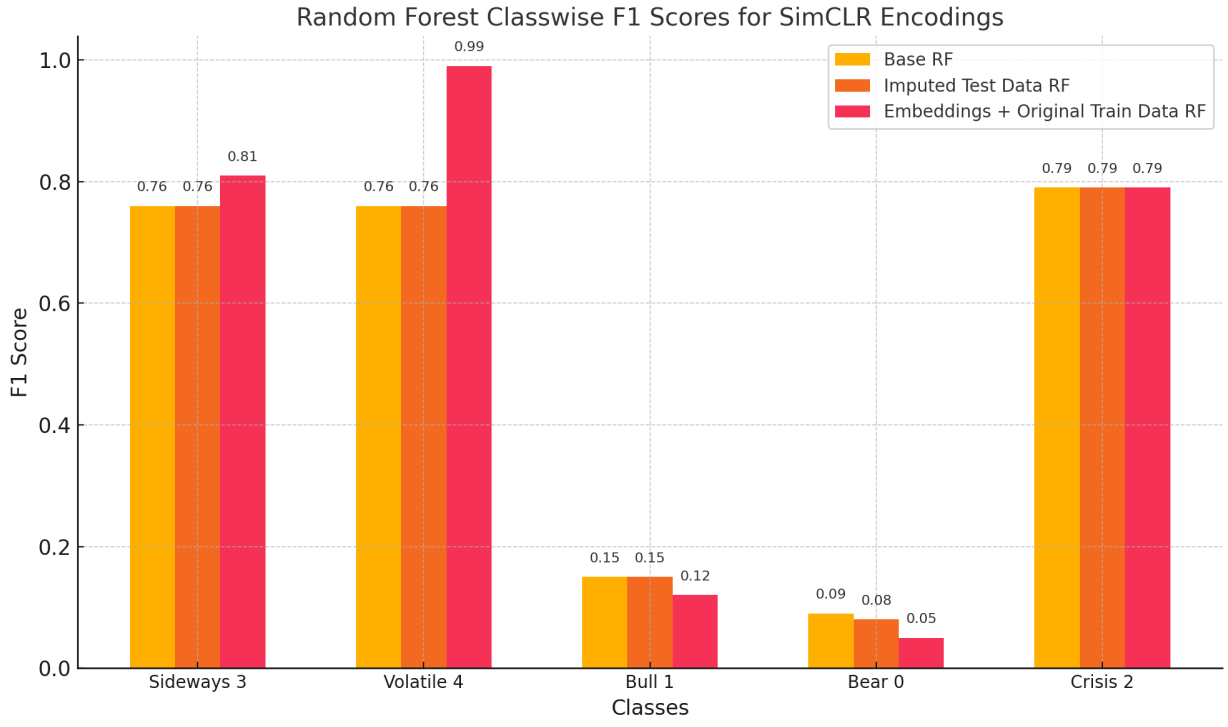

Figure 4.5: Classwise F1 scores for Random Forest for SimCLR embedding method

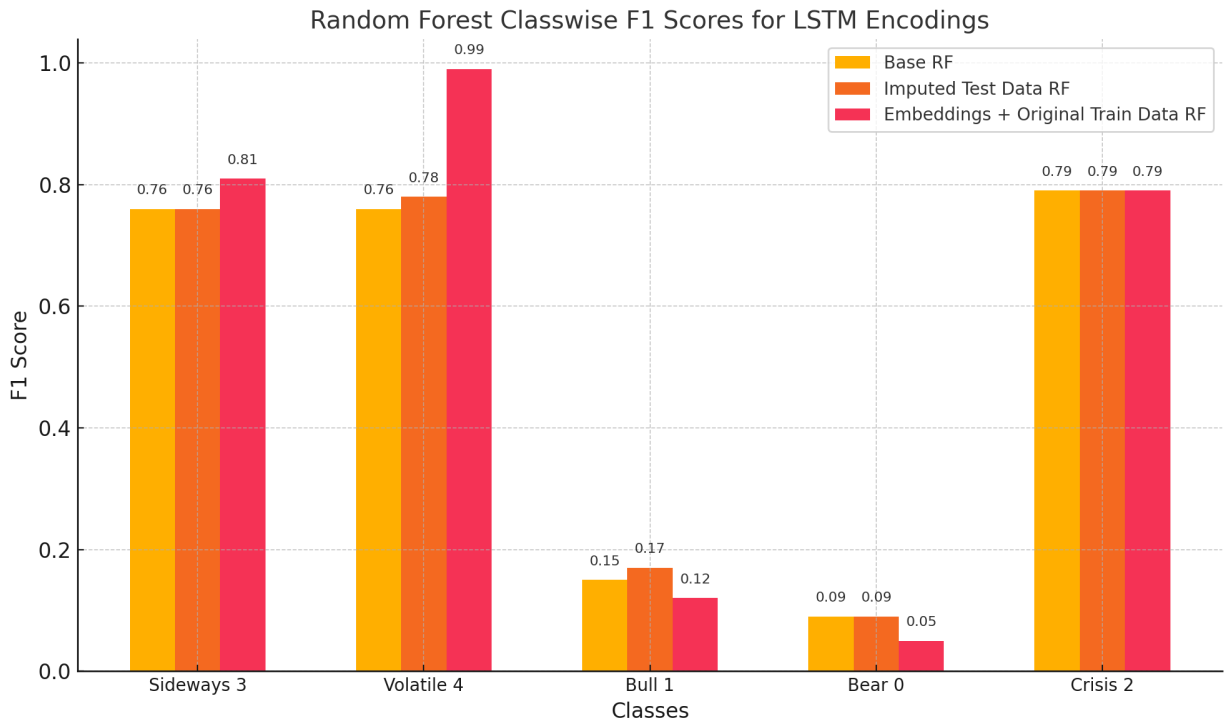

Figure 4.6: Classwise F1 scores for Random Forest for LSTM embedding method

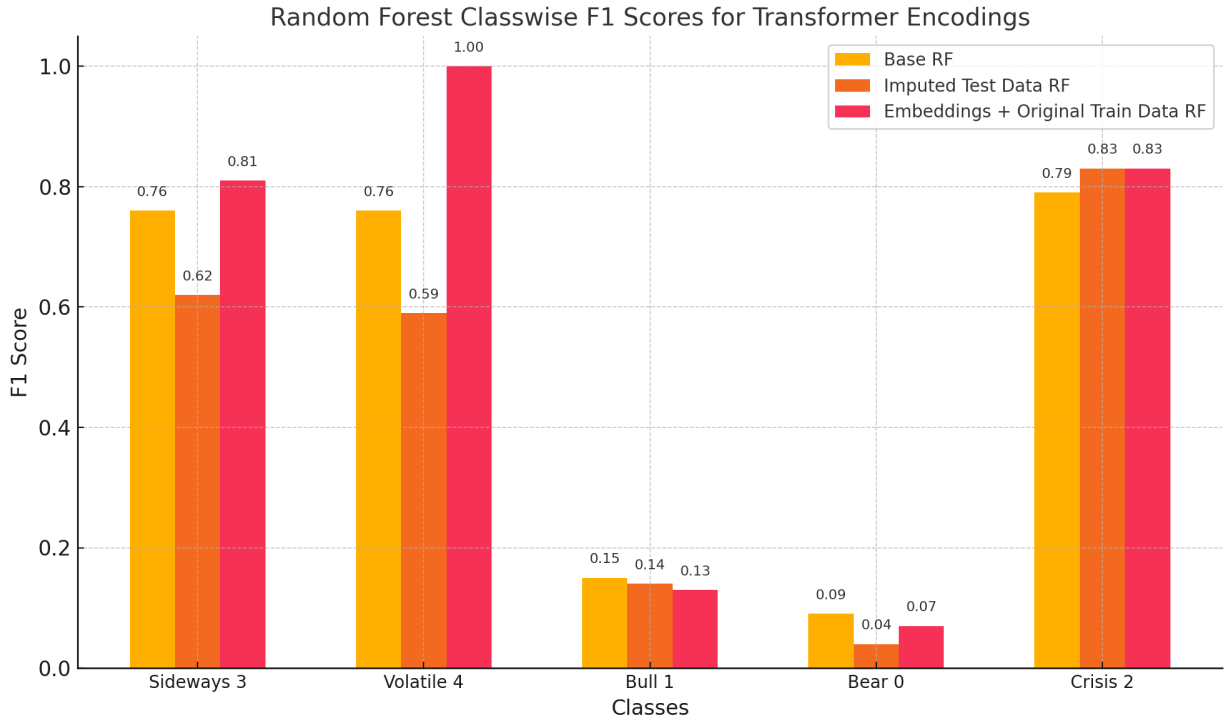

Figure 4.7: Classwise F1 scores for Random Forest for Transformer embedding method

4.2 Analysis of Visualizations

Figures 4.1-4.4 help illustrate the following points:

- Imputing the masked entries using the SimCLR and LSTM embeddings during test time perform comparable but slightly better than the base models where a mean-imputation technique is used.
- Imputing the masked entries using the Transformer embeddings during test time performs significantly worse than the base models where a mean-imputation technique is used.
- Training a classifier with original feature set supplemented with train embeddings and testing on embeddings imputed test data supplemented by test embeddings shows a marked increase in performance for all three embedding learning techniques across supervised classification models.
- The best classification model achieved using all three embedding learning techniques achieves similar results.

Figures 4.5-4.7 help illustrate the following about the scope of embeddings learned:

- The overall improvement in performance metrics is primarily driven by significant enhancements in the prediction accuracy for majority classes, which dominate the dataset.
- However, the prediction performance for underrepresented minority classes decreases when our methodology is applied, highlighting the challenges of imbalanced data.

Chapter 5

Discussion and Conclusion

5.1 Summary of Findings

This project explored advanced machine learning techniques to learn embeddings without relying on labeled data. It used methods like SimCLR for contrastive learning, along with sequential models like LSTMs and Transformers, to uncover meaningful patterns and representations for predicting financial market regimes. The results highlighted the effectiveness of embedding-based approaches in uncovering complex patterns within financial data, particularly in challenging conditions such as incomplete or noisy datasets. SimCLR and LSTM embeddings proved to be the most impactful, significantly boosting the performance of downstream supervised models like Random Forest and XGBoost. By combining these embeddings with original features, the models achieved greater accuracy and robustness in market classification tasks.

From a real-world perspective, this research has tangible implications for the financial industry. The ability to predict market regimes with higher accuracy aids in developing smarter investment strategies, enabling portfolio managers to make better-informed decisions dur-

ing bull, bear, or volatile market conditions. Improved classification of market regimes can also enhance risk management frameworks, helping institutions prepare for sudden market changes or financial crises. Furthermore, the findings contribute to the broader adoption of machine learning in finance, demonstrating how cutting-edge models can provide actionable insights from raw data, ultimately driving efficiency and innovation in financial systems.

5.2 Limitations

While the project demonstrated promising results, it also revealed several limitations. A key challenge was the class imbalance in the dataset, which affected the model's ability to accurately predict minority regimes like bear markets. This resulted in a bias toward dominant classes, impacting recall and F1-scores for less frequent market scenarios. Additionally, although SimCLR and LSTM embeddings performed well, Transformer embeddings struggled to capture the underlying patterns in financial data, potentially due to insufficient optimization or the inherent complexity of the dataset. The static labeling of market regimes was another limitation, as it did not fully account for the dynamic and fluid nature of financial markets. Lastly, while the models were effective for specific asset classes, their generalizability to broader financial datasets or other markets was not tested extensively, which could limit their applicability in global contexts.

5.3 Future Work

Future efforts could address the identified limitations and expand the scope of this research while enhancing its real-world applicability. To tackle class imbalance, techniques such as oversampling, weighted loss functions, or synthetic data generation (e.g., SMOTE) could be employed to improve the prediction of minority market regimes. This could directly benefit

financial institutions by providing more reliable predictions for rare but impactful market scenarios, such as bear markets or crises, helping them prepare for potential downturns or volatility.

The underperformance of Transformer embeddings could be revisited through hyperparameter tuning, larger datasets, or the use of pre-trained models specifically designed for financial time series analysis. Improved Transformer models could unlock deeper insights into complex financial sequences, enabling more accurate predictions for portfolio adjustments, algorithmic trading, and credit risk assessments.

A dynamic market regime labeling system could be introduced to better reflect the continuously changing conditions in financial markets, making the approach more applicable to real-time decision-making. For example, dynamic labeling could help asset managers adjust their strategies in response to evolving market conditions, enhancing both returns and risk management.

Advanced imputation techniques, such as generative models or sequence-to-sequence architectures, could further enhance the handling of missing data, which is common in real-world financial datasets. This would allow institutions to make more informed decisions even with incomplete data, such as during rapid market fluctuations or crises. Finally, scaling this approach to include a broader range of asset classes and global financial markets would test the robustness of the methodology and its potential to provide actionable insights across diverse financial systems. For instance, it could be applied to commodities, fixed-income securities, or emerging markets, providing investors and policymakers with tools to analyze and respond to market dynamics across geographies and asset types. Such advancements would not only improve predictive analytics but also contribute to more stable and efficient financial markets worldwide.

Bibliography

[1] Tianqi Chen and Carlos Guestrin. *XGBoost Documentation*, 2024.

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. Simclrv2 - big self-supervised models are strong semi-supervised learners. https://github.com/google-research/simclr, 2020.

[3] MathWorks. Deep learning in quantitative finance: Transformer networks for time series prediction. https://blogs.mathworks.com/finance/2024/02/02/deep-learning-in-quantitative-finance-transformer-networks-for-time-series-predictio 2024.

[4] Author Names Redacted. Explainable stock price movement prediction using contrastive learning. In *Proceedings of the ACM International Conference on Artificial Intelligence*. ACM, 2024.

[5] Yahoo Finance. Yahoo finance - business finance, stock market, quotes, news. https://finance.yahoo.com/?guccounter=1, 2024.

[6] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Long Short-Term Memory (LSTM) — Dive into Deep Learning 1.0.3 documentation*. Self-published, 2020.