Crypto Currency Funding Rates Prediction

in Cooperation with **Characteristics Matter GmbH**

Nikan Hajialiakbarghomi, Saam Voitto Erkki Ghiasi, Jing Wang, Nasim Zarei

Agenda

- Introduction
- Data Collection
- Data Preprocessing
- Methodology
- Results

Introduction

Cryptocurrency funding rates play a critical role in balancing liquidity, reflecting the cost of holding long or short positions within decentralized finance (DeFi) markets.. Accurate prediction of these rates is vital for traders and investors to optimize strategies, minimize risks, and enhance profitability.

In this project, we explore the application of advanced predictive modeling techniques to forecast cryptocurrency funding rates. By leveraging historical market data, machine learning, and time series analysis, we aim to provide actionable insights into market trends.

What are Funding Rates?

In the cryptocurrency market, borrowing rates and lending rates are key metrics that facilitate the decentralized finance (DeFi) ecosystem and derivatives trading.

- Borrowing Rate:

The borrowing rate is the cost incurred by users to borrow cryptocurrency assets. It is determined by the supply demand dynamics in the market. The rate adjusts based on the utilization rate of the liquidity pool, higher demand leads to higher borrowing costs.

- Lending Rate:

The lending rate is the return earned by users who lend their cryptocurrency assets to liquidity pools. By providing liquidity, lenders enable borrowing and market functionality, earning interest in return.

Data Collection and Preprocessing

Data Collection

APIs:

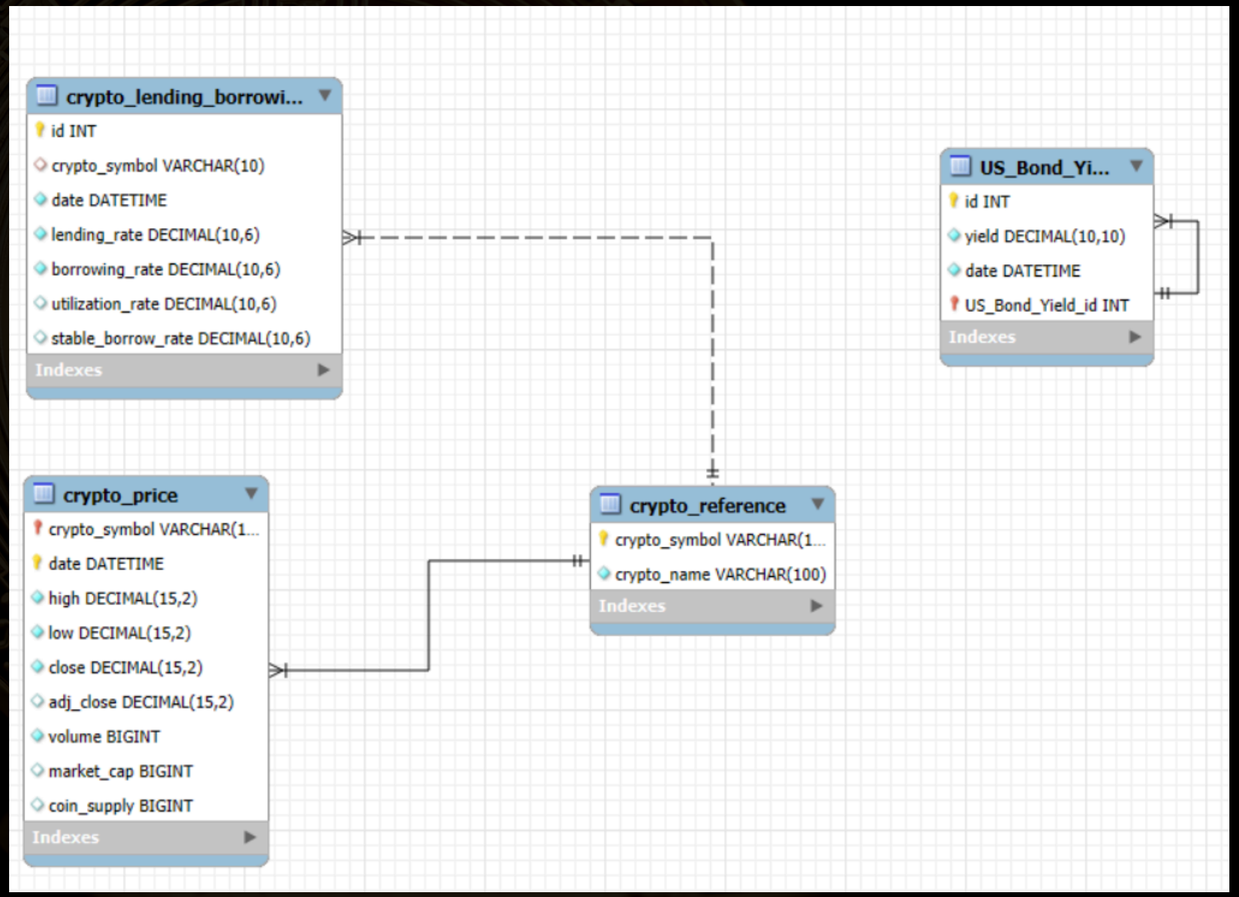
- Binance for prices
- Aave for rates

Market capitalization:

- Scraped from messari.io

Interest rate proxy :

- FRED Economic Data
- US Treasury Bond Monthly Yields



Coins and Data Overview

- Number of coins: 16
- Data frequency: Hourly
- Number of rows: 402,000 valid rows
- Data period: Jan 2021 to Dec 2024

Token Symbol	Name	Description
BAT	Basic Attention Token	Rewards for digital advertising
LINK	Chainlink	Decentralized oracle network
KNC	Kyber Network Crystal	On-chain liquidity protocol token
MKR	Maker	Governance token for the MakerDAO ecosystem
MANA	Decentraland	Token for virtual reality and gaming platform
ZRX	0x	Decentralized exchange protocol
SNX	Synthetix	Token for trading synthetic assets
WBTC	Wrapped Bitcoin	ERC-20 token backed by Bitcoin
ENJ	Enjin Coin	Cryptocurrency for gaming and NFTs
REN	Ren	Protocol enabling cross-chain transactions
YFI	Yearn.Finance	Yield optimization protocol token
UNI	Uniswap	Governance token for Uniswap DEX
CRV	Curve DAO Token	Governance token for Curve protocol
BAL	Balancer	Automated market maker token
ENS	Ethereum Name Service	Token for decentralized domain names
1INCH	1inch	Token for decentralized exchange aggregator

Features and Preprocessing

Rate Related:

- Borrowing Rate and Lending Rate
- Utilization Rate = Total Borrowed / Total Supplied * 100

Sentiment, and Seasonality Related:

- Attention = volume * price
- Time Features to capture seasonality - Cyclical
- Yield (From US Treasury Bond)

Price and Volume Related:

- OHLCV (Open, High, Low, close, volume)
- Hourly returns

Coin Specification Features:

- Name embeddings (model : all-MiniLM-L6-v2)
- Market Capitalization
- Scaling features on each coin and all together

Models

Model Input and Output

The Goal of the project is to predict the trend of a later sequence of Funding rates based on an earlier sequence of features.

Model Input:

A sliding window of sequences, spaced 1 hour apart, with a customizable set of features and sequence length.

Model Output (Option 1):

A sliding window of sequences, spaced 1 hour apart, predicting the lending rate and borrowing rate.

Model Output (Option 2):

The directional movement of the lending rate or borrowing rate for a 1-hour sequence, with customizable length:

- 0: No change
- 1: Upward movement
- 2: Downward movement

Methodology

Model Selection

2-layers LSTM (rate prediction)

2-layers LSTM(Trend Prediction)

1-layer LSTM (Trend prediction)

Encoder-Decoder Architecture (Trend Prediction)

Attention Model (Trend Prediction)

Generalization Methods

IQR for outlier detection (upper and lower bounds by IQR= Q3-Q1)

Training set balancing

Weighting the classes

Loss function regularization

1-Layer LSTM Architecture

- LSTM (80 units) → Dropout (10%) → Dense (3 classes, softmax)

2-Layer LSTM Architecture (Exact rate)

- LSTM (48 units) → Dropout (20%) → LSTM (48 units) → Dropout (20%) → Dense (2*5)

2-Layer LSTM Architecture (Trend)

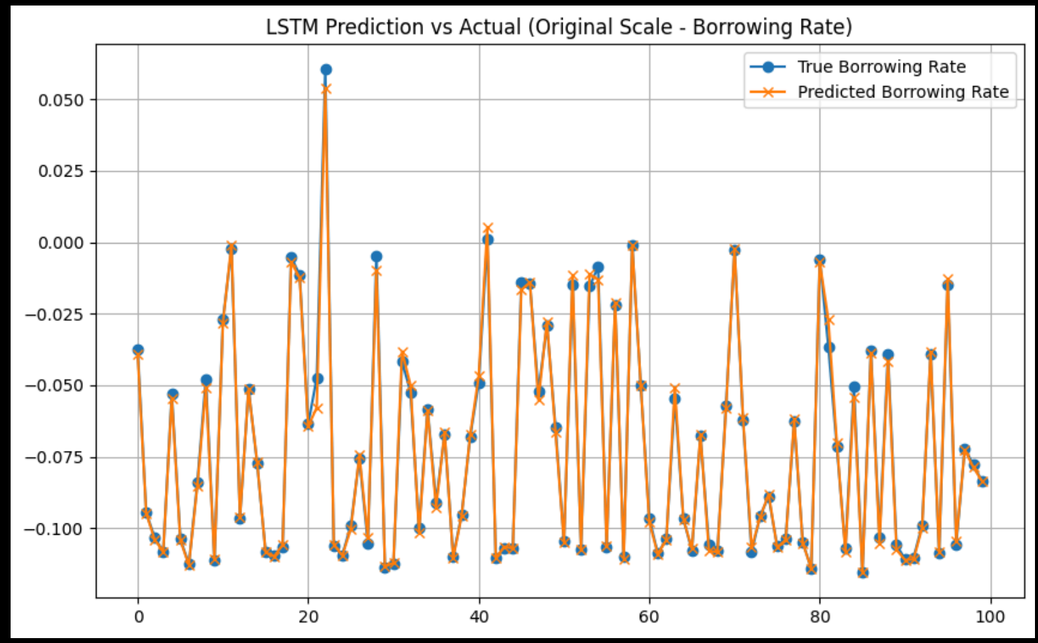
- LSTM (80 units) → Dropout (10%) → LSTM (40 units) → Dropout (5%) → Dense (3 classes, softmax)

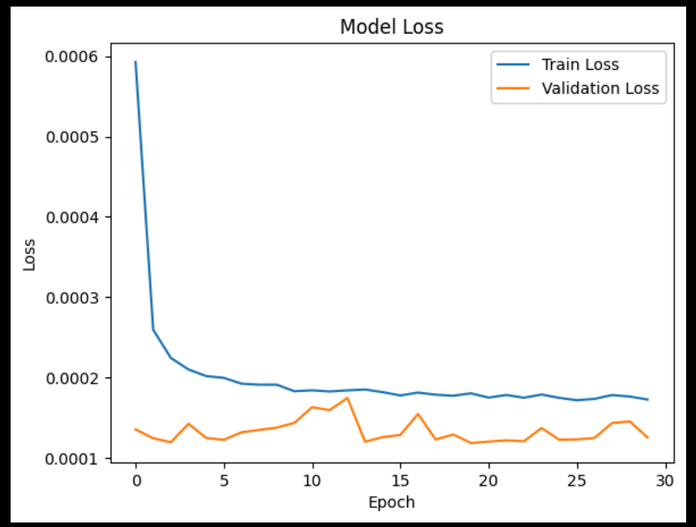
Encoder-Decoder Architecture

- LSTM (80 units) → Dropout (10%) → Latent Dim Layer (10 dims) → LSTM (40 units) → Dropout (5%) → Dense (3 classes, softmax)

Exact Rates Prediction

- **1h Ahead Prediction + 40h History**
- Rates rarely change within an hour, so the model effectively “copies” the last observed value.
- **Performance drops sharply** when predicting for each coin or extending the forecast horizon.
- **Focusing on the overall trend,** rather than exact values leads to more actionable results.





MAE
0.0086

Trend prediction

Benchmark

Random Guessing Performance

- **Goal: Evaluate if the model beats random guessing.**
  - Methodology: Simulated 10,000 random predictions based on training class distribution.
  - Class probabilities:
    - No change: 41.47%
    - Upward: 32.69%
    - Downward: 25.85%
  - Computed Macro F1 Score for each simulation.
  - Input : Output Window => 48 : 48
  - **Results: Mean F1 Score: 32.36%**
  - 95% Confidence Interval: [32.36%, 32.37%]

Grid Search Insights

Feature Importance

- **Lending & borrowing rates** rank highest.
- Rates, returns, yields, market cap, attention, volume matter across models.
- Embeddings & time features stand out in Attention models.

Input/Output Ratios

- **48:48 and 96:8** performed best.
- Performance stable across different data amounts.
- Training curves were more logical.

Overfitting Fixes

- **Data balancing** works best for LSTMs.
- **Class weighting** is most effective for Attention models.
- Outlier thresholds harm performance.

Hyperparameter Tuning

- Under 50 epochs + batch size 200.
- **10% dropout.**
- **LSTM cell size ~40.**

Model Results

Two-Layer LSTM

Goal: Capture complex short- and long-term temporal dependencies.

Why Two Layers? extracting higher-level features.

Single-Layer LSTM

Goal: Simplify the architecture to reduce overfitting.

Why One Layer? Focus on essential patterns, avoiding unnecessary complexity.

Encoder-Decoder (VAE)

Goal: Introduce a probabilistic bottleneck to combat overfitting.

Why VAE? Mapping inputs to a latent Gaussian space forces the model to generalize, discouraging memorization of noise.

Attention Model

Goal: Explore a more flexible mechanism for focusing on different parts of the data.

Why Attention? Particularly effective for static data and offers a novel approach to find key patterns from time series.

Model	Input:Output	F1 Score
2-Layer LSTM	48:48	**38%**
2-Layer LSTM	96:8	**37%**
One-Layer LSTM	48:48	**40%**
One-Layer LSTM	96:8	**38%**
VAE	48:48	**25%**
Attention Model	48:48	**42%**

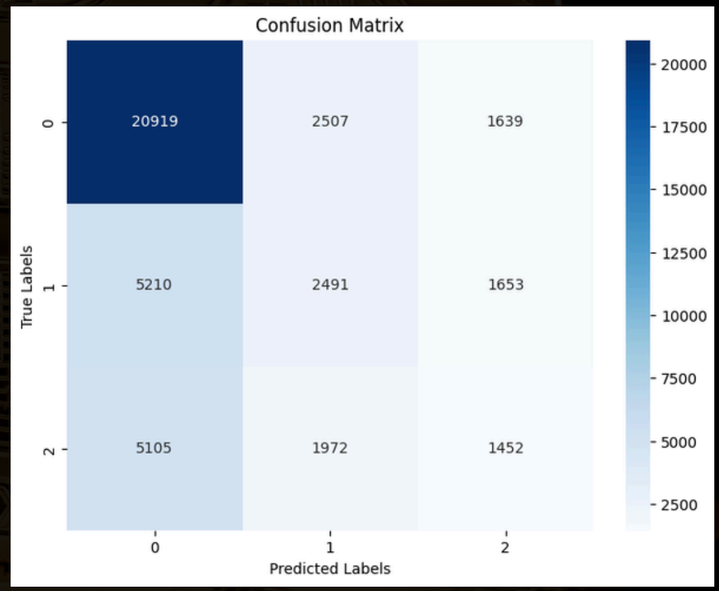
Attention Model

- Layers
  - Layers: Multiattentionhead 1 & 2 (20units per 4 Heads) -> Skip -> Norm
  - Dense Layer with Swish & HE Weight initializer (40 units)
  - Dropout Layer 10%
  - Flatten from 2D->1D (ignoring batch dimesnsion)
  - Dense Layer with Swish & HE Weight initializer (40 units)
  - Dense (3 classes, softmax)
- **Training Data Classes were Weighted**

Input	Output	Epochs	F1 Score
48	48	20	**42%**
48	48	50	**41%**

Motivation Behind Model:

- Handling of static and dynamic features
- Testing different architectures to Sequential for Robustness testing
- Can capture Global Patterns more effectively



Best Performance Until Now

Δ[Benchmark] ≈ +10%

Wrap Up

- **Explored Models:**
  - LSTM, Encoder-Decoder, Attention-based architectures.
  - Used class weighting, feature selection, data balancing, and regularisation.
- **Key Findings:**
  - F1 Scores > 40%, surpassing 33.33% baseline.
  - Best LSTM: RMSprop, 40 cells, 0.1 dropout, data balancing.
  - Best Attention: AdamW, 80->40 cells, 0.1 dropout, class weighting.
  - Best Input-Output Windows: 48:48, 96:8.
- **Challenges:**
  - LSTM issues with embeddings (static data) and time features (implicitly captured through hidden state) .
  - Most Regularisation techniques, early stopping, and outlier removal ineffective.
- **Future Directions:**
  - Test additional features (e.g., equity market indices, sentiment through news or analyst reports).
- **Conclusion:**
  - Models captured meaningful signals, challenging weak form EMH assumptions in Aave platform.

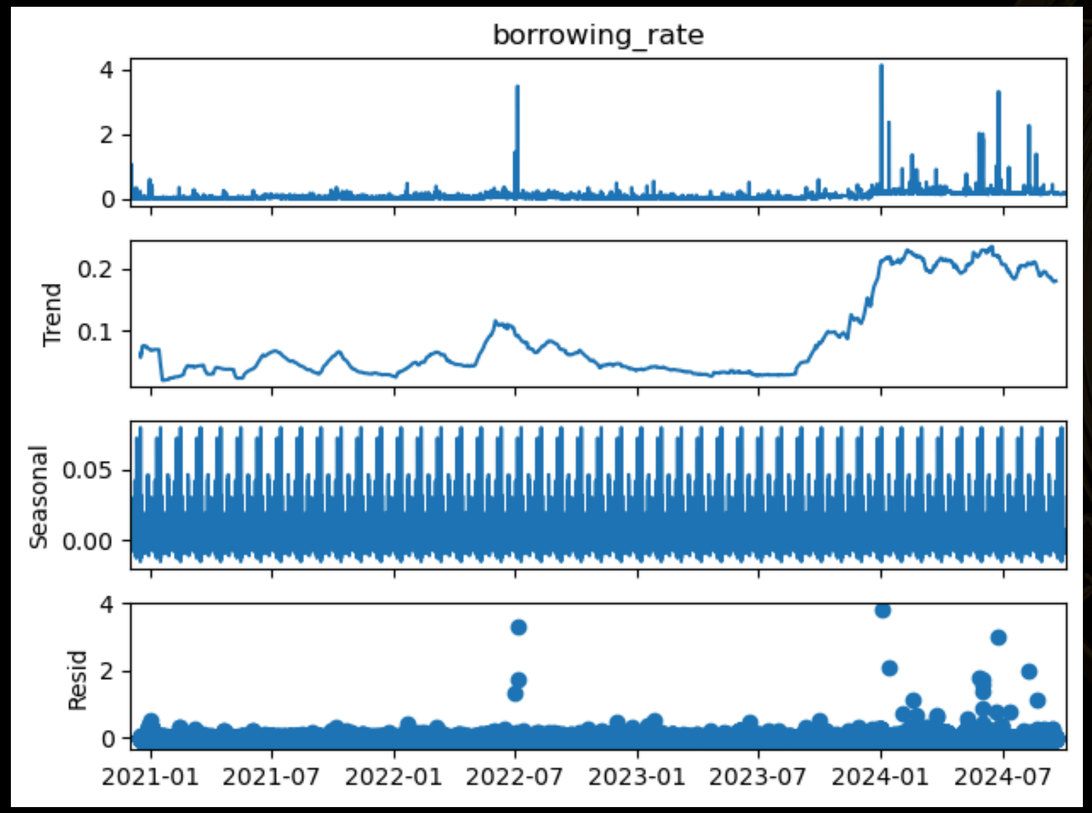
Thank You

FOR YOUR ATTENTION

Github Link: github.com/nikanhg/Funding_rate_Aave

Appendix

Why Cyclical Features?



Seasonal Decompose:

Model Y[t] = Trend[t] + Seasonal[t] + Residual[t]

Borrow Rate Avg. ≈ 0.08

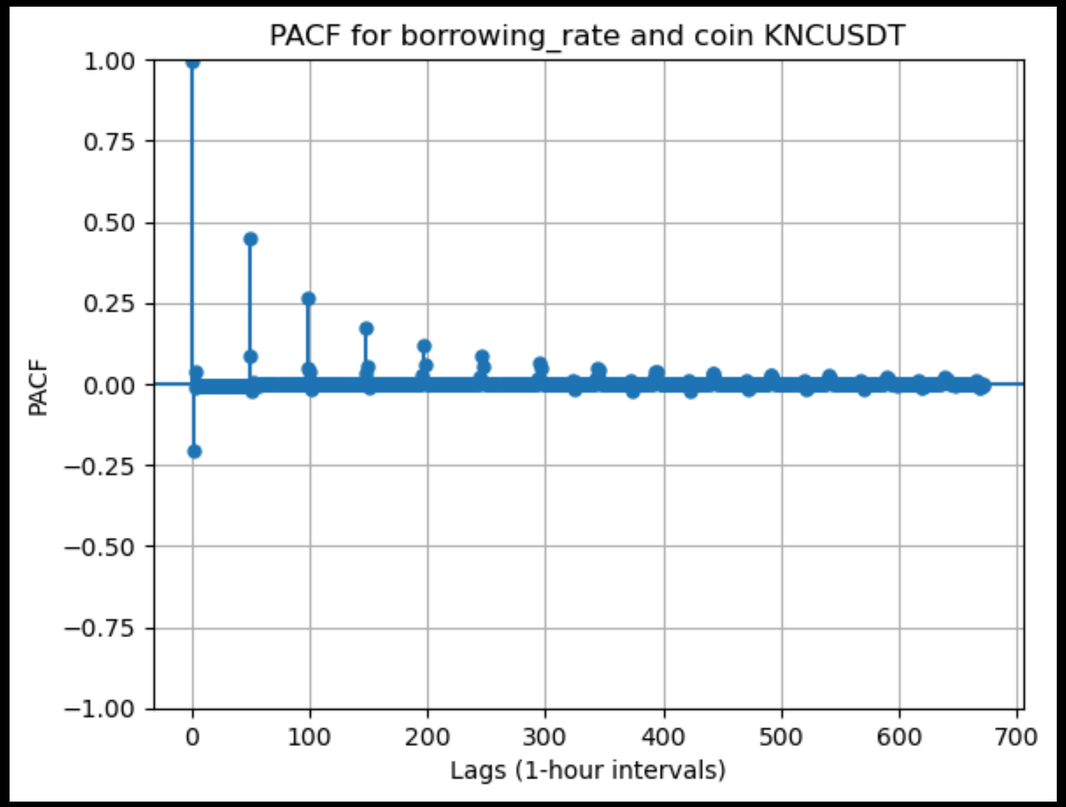
Seasonal Decompose Avg. ≈ 0.05 (Monthly)

- Strong Monthly Effect
- Hourly and Daily is weak
- Weekly & Bi-Weekly is in between

Cyclical features should capture some of these seasonal effects.

Even if cyclical features are not appropriate for hourly or daily data, they are a lighter numerical bias than cardinal encodings.

Testing Autocorrelations



Partial Autocorellations

These were calculated per coin up to 672 lags (28 days) in order to gauge what lagged window could be appropriate.

The coins showed varying results, some of which showed high correlations at very high lags - these were most likely due to numerical instability as they were unbounded.

Most coins like in this example showed significant, but numerically stable lags before 200, hence why we chose to experiment with windows lagged from 8-192 hours.

Encoder-Decoder Model

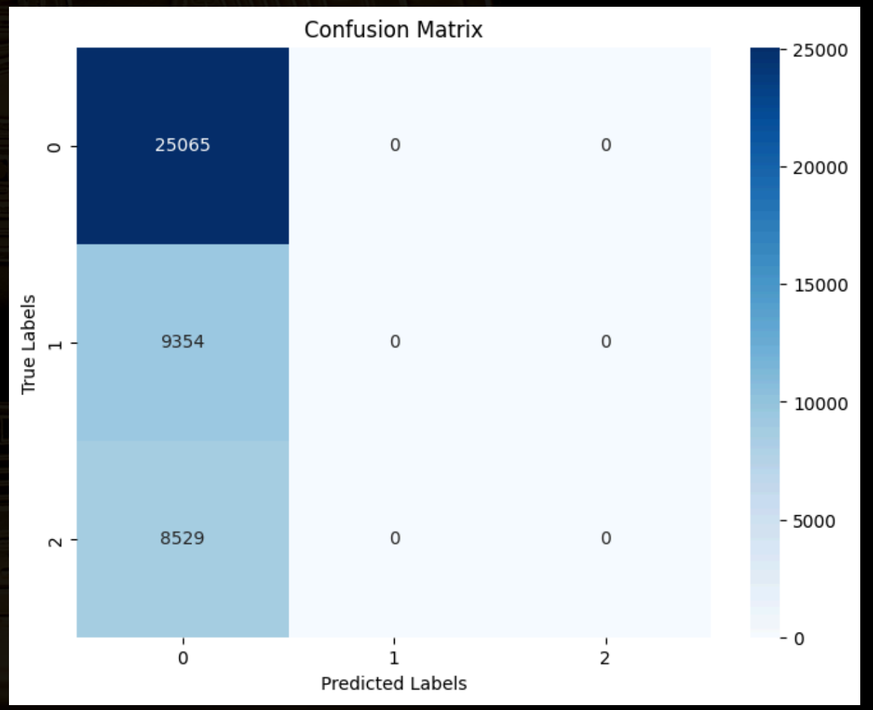
Better generalisation

- Layers
  - Layers: LSTM 1 (Encoder/Decoder) (80 units) → BatchNorm → Dropout (10%)
  - Latent Dim Layer (10 dims)
  - LSTM 2 for Encoder/Decoder) (40 units) → BatchNorm → Dropout (5%)
  - Dense (3 classes, softmax)

Input	Output	Epochs	F1 Score
48	48	20	12%
48	48	50	12%
48	48	100	25%

Motivation Behind Model:

- Tackle overfitting by keeping complexity but forcing everything into a probabilistic bottleneck
- Hoping to get a generalised representation in the latent dim layer

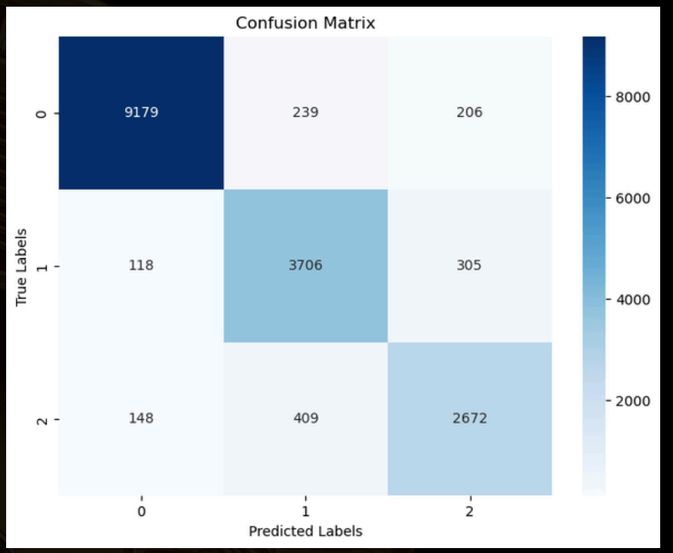


Overfitting Majority Class

2-Layer LSTM Model

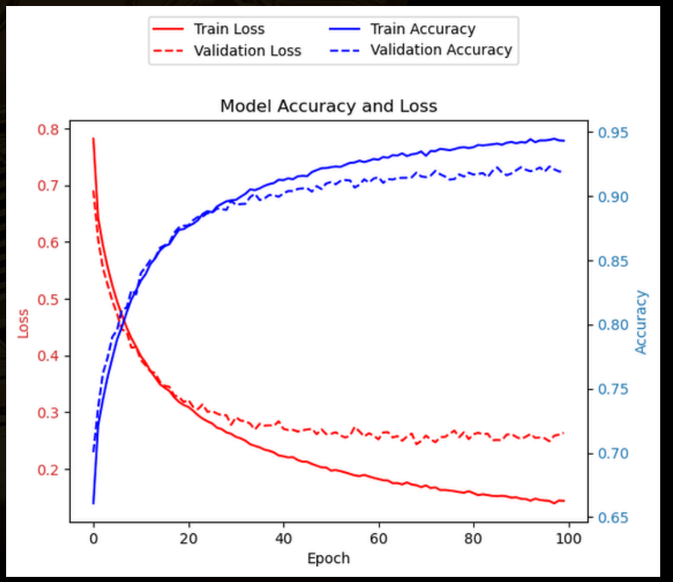
With look ahead Bias

- Layers
  - Layers:LSTM (80 units) → BatchNorm → Dropout (10%)
  - LSTM (40 units) → BatchNorm → Dropout (5%)
  - Dense (3 classes, softmax)
- Data Split: Train (60%), Validation (20%), Test (20%)
- Key Params: 100 epochs, 200 batch size



Best Results (Unbiased by Data Quantity):

Input	Output	Outlier Limit	F1 Score
48	48	None	93%
48	24	None	88%
24	24	3	88%



2-Layer LSTM Model

Without look ahead Bias

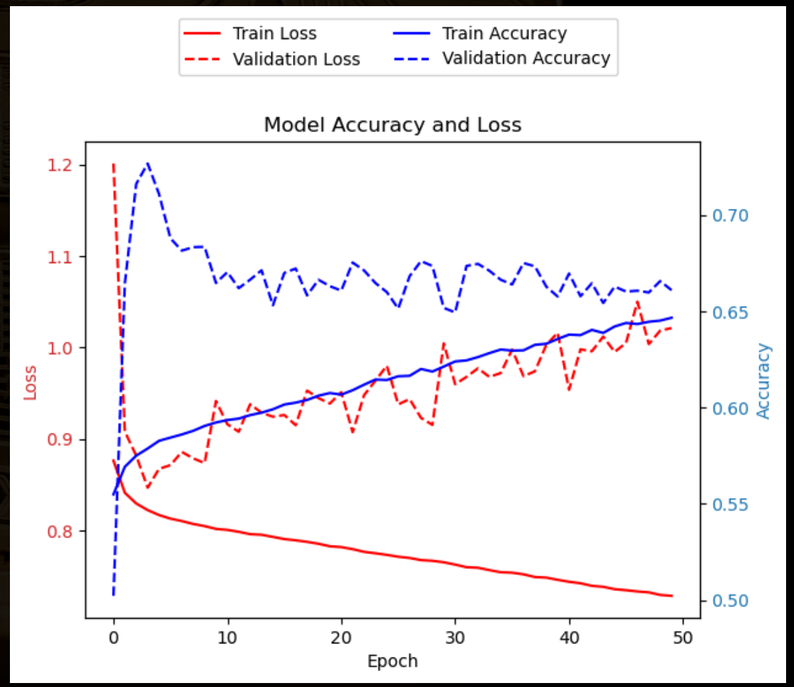
Best Results:

Input	Output	F1 Score
48	48	**38%**
96	8	**37%**

Input : ouput window	**48 : 48 or 96 : 8**
Best Outlier Threshold	**None**
Best Data Balancing Limit	**2 Times allowed**

Motivation Behind LSTM Model:

- **Capturing Temporal Dependencies:**
- **Designed to handle sequential data effectively by updating hidden states, learning short- and long-term dependencies.**
- **Baseline for Complexity:**
- **Using a 2-layer LSTM provides a balanced starting point to evaluate whether the model requires more or less complexity.**



One-Layer LSTM Model

LSTM (80 units) → Dropout (20%) → Dense (3 classes, softmax)

- **embeddings, Time Features and Price** have negative Impacts on the performance.
- Best configuration identified: **RMSprop optimizer, 40 cell size, 0.1 dropout rate**.
- L2 regularization and early stopping did not improve results, hence excluded.
- Class weighing did not Increase the performance

Input	Output	F1 Score
48	48	**40%**
96	8	**36%**

Motivation Behind Model:

- **2 Layer LSTM was overfitting**
- **Reducing complexity as opposed to regularisation techniques**
- **Was more effective**