“ADVANCED TRADING STRATEGIES: DEEP LEARNING”

ITESO

ITESO, Universidad Jesuita de Guadalajara

MICROESTRUCTURA Y SISTEMAS DE TRADING

LUIS FELIPE GÓMEZ ESTRADA

IVANNA HERRERA IBARRA 744614

ANA SOFÍA HINOJOSA BALE 742594

28 DE OCTUBRE DE 2025

Table of contents

Introduction

In this project, a systematic trading strategy was developed using 15 years of daily Lululemon stock price data. Multiple neural network architectures were trained to predict trading signals, such as buy, sell and hold, based on engineered time series features from technical indicators. The strategy is evaluated in a realistic backtesting environment, which considers commissions, borrow rate, long and short positions without leverage, and uses different metrics to measure its performance. The data were divided into train, test, and validation sets in order to avoid overfitting and to evaluate the model's performance across different time periods.

Strategy Overview

Deep learning approach to trading signal prediction

This strategy uses two trained deep learning architectures, Multi-Layer Perceptron (MLP) and Convolutional Neural Network (CNN), to generate trading signals; buy, sell, and hold. These models use the normalized engineered features from momentum, volatility, and volume technical indicators to learn patterns and make predictions.

Why multiple architectures

Using multiple architectures is important because each one captures different type of patterns depending on its structure and design. By training both MLP and CNN models, the strategy can use the strengths of each model, see which one works better, and choose the most accurate for predicting trading signals. Using more than one model also makes the strategy more reliable and helps understand how the features influence the predictions.

MLP

A Multi-Layer Perceptron (MLP) is a type of feedforward artificial neural network composed of at least three layers: an input layer, one or more hidden layers, and an output layer. MLPs are trained using a supervised learning method called backpropagation. In this strategy, the MLP captures complex relationships between normalized features without considering temporal sequences. It is effective at learning interactions between momentum, volatility, and volume indicators, identifying patterns that are not strictly time dependent.

CNN

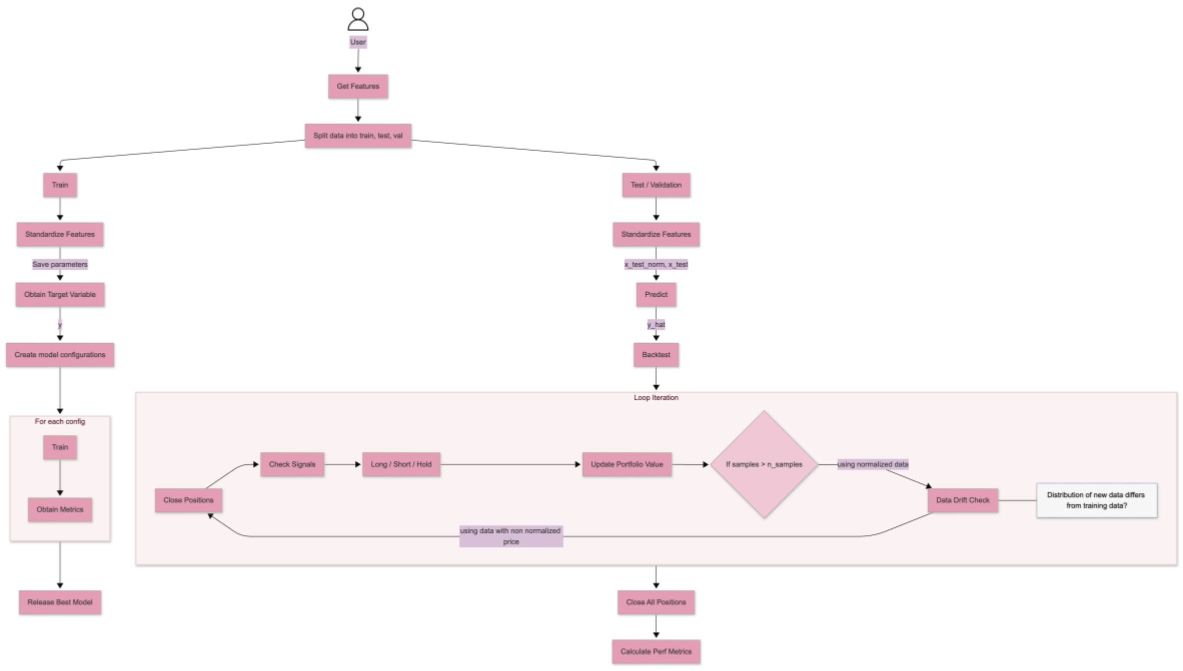
A Convolutional Neural Network (CNN) is a type of deep learning model designed to detect local patterns in sequential data. It consists of convolutional layers that automatically extract features by applying filters to the input data, followed by pooling layers that reduce dimensionality while preserving important information. In this strategy, the CNN analyzes short-term trends and identifies temporal structures in the engineered features that may indicate upcoming price movements. It is effective at learning how recent fluctuations in momentum, volatility, and volume indicators relate to directional changes in price.

Expected advantages and limitations

Using both MLP and CNN architectures provides complementary strengths for trading signal prediction. The MLP effectively captures nonlinear relationships between features, while the CNN recognizes short-term temporal patterns that may indicate market momentum shifts. Combining both approaches helps the model adapt better to different market conditions and detect various types of patterns in the data.

However, deep learning models also have limitations. They require large amounts of data and computational resources to train effectively, and their performance can be sensitive to hyperparameter choices. In addition, these models can sometimes overfit the training data, which means they might perform well during training but fail to generalize to unseen market conditions.

Methodology Overview



The process begins with feature collection, where input variables that describe the market or system are gathered. These features will serve as inputs for the machine learning model. Once collected, the dataset is split into three subsets: a training set for learning model parameters, a validation set for tuning hyperparameters and selecting the best configuration, and a test set for evaluating the model's final performance on unseen data. Before training, all features are standardized, ensuring that no single feature dominates due to scale differences. The normalization parameters from the train normalization are then saved to apply the same transformation later to test and validation data. The target variable is defined next, representing what the model aims to predict, the trading signals. Multiple model configurations are then created. Each configuration is trained on the training data, and key metrics such as accuracy and loss are recorded. Afterward, performance across models is compared, and the best-performing model is selected and released based on validation results.

During validation or testing, the saved normalization parameters are applied to ensure consistency, and the trained model is used to generate predictions or trading signals. These predictions are evaluated through a backtest, where a simulated trading strategy buys or sells according to the model's signals, tracking the portfolio's value and performance over time. Once deployed, the model undergoes continuous monitoring to check signal accuracy and assess performance. As new data accumulates, the system performs a data drift check using the Kolmogorov–Smirnov test, which compares the distribution of new data against that of the original training data. Finally, after each test or backtest cycle, all positions are closed, and key performance metrics, such as total return, Sharpe ratio, maximum drawdown, and win rate are computed.

Feature Engineering

Feature engineering was applied to transform raw price and volume data into predictive momentum, volatility, and volume indicators. We created 21 features, which were normalized for them to be comparable and to make sure the model does not give more importance to higher values, since each one has its own scale.

Indicators

Momentum

Momentum indicators are used to determine the strength or weakness of a stock's price trend and indicate the rate at which the price changes. They help identify trends, potential reversals, and overbought/oversold conditions. The momentum indicators used for this strategy are the following:

- Relative Strength Index (RSI): Measures the speed and strength of price changes. Three RSI were computed, with windows of 10, 14 and 20, to have short-term, medium-term, and slightly longer-term perspective on market momentum, allowing for more detection of overbought or oversold conditions across different time horizons.
- Kaufman Adaptive Moving Average (KAMA): Follows trends adjusting its sensitivity to price movement based on market volatility.
- Stochastic Oscillator: Compares the current closing price to the high-low range over a period. Three stochastic oscillators were computed, with windows of 10, 14 and 20, to have short-term, medium-term, and slightly longer-term insights into price momentum.
- Rate of Change (ROC): Measures the percentage change in price over a given period. Three rates of change were computed, with windows of 10, 12 and 20, to have short-term, medium-term, and longer-term perspectives on price momentum.
- Williams %R: Shows overbought/oversold levels relative to the highest and lowest prices of a given period. Three Williams %R indicators were computed, with windows of 7, 14 and 2, to have short-term, medium-term, and short-term perspectives on market momentum.

Volatility

Volatility indicators are used to measure the intensity of a stock's price fluctuations. They help understand risk and possible breakout points. The volatility indicators used for this strategy are the following:

- Bollinger Bands: Upper and lower bands that widen when a stock's price becomes more volatile and contract when it is more stable, which indicate overbought/oversold levels.
- Keltner Channels: Upper/lower bands and central moving average that reflect price volatility. The bands represent the range in which prices typically move, expanding during high volatility and contracting when volatility decreases.
- Donchian Channels: Upper and lower bands that represent the highest and lowest prices over a given period.

Volume

Volume indicators are used to measure market participation during a specific time window, providing information for trading decisions. The volume indicators used for this strategy are the following:

- On-Balance Volume (OBV): Measures positive and negative volume flow and analyzes the trading direction. It is represented with a single line that helps investors identify buy or sell points.
- Accumulation/Distribution Line (A/D): Determines a stock's trend using the relation between the stock's price and volume flow. Accumulation is the level of buying (demand) and distribution is the level of selling (supply).
- Chaikin Money Flow (CMF): Calculates how much money moves into a market, relative to the amount of money that leaves the market. A positive value indicates net buying pressure, while a negative value indicates net selling pressure.
- Ease of Movement (EOM): Measures the relationship between price and volume. That relationship is shown as an oscillator that fluctuates above and below a zero line.
- Force Index (FI): Measures how much power is used to move an asset's price by combining price change and volume.

Normalization approach and feature scaling

Range scaling

Indicators with ranges from 0 to 100 were divided by 100 to keep them on a 0–1 scale, preserving their relative interpretation within a common range. The indicators normalized this way are RSI and Stochastic Oscillator.

Range shift

Williams %R indicator values oscillate within a range from -100 to 0, therefore, we normalized them using $\frac{x+100}{100}$, which rescales the values to a positive 0-1 range, making them more interpretable and comparable.

Min-max normalization

Indicators with variable ranges that depend on their own minimum and maximum values, such as KAMA and CMF, were normalized using $\frac{x-min}{max-min}$, which transforms them to a 0-1 range. In this scale 0 corresponds to the minimum value and 1 corresponds to the maximum value.

Z-score standardization

Indicators like ROC, OBV, A/D, EOM, FI, and Close price have unbounded variable values, so they were normalized using $\frac{x-\mu}{\sigma}$, which transforms the original values to a distribution with mean 0 and standard deviation 1.

Relative position scaling

Indicators with variable ranges that depend on their own lower and upper bands, such as Bollinger Bands, Keltner Channels, and Donchian Channels, were transformed into a relative position using $\frac{Close-Lower}{Upper-Lower}$, which rescales the price into a 0-1 range. In this scale 0 means the price is near the lower bound and 1 means the price is near the upper bound.

Class imbalance strategy

The trading signals generated by this strategy are naturally imbalanced, with hold signals occurring more frequently than buy or sell signals. To address this, Bayesian optimization was used to determine the optimal alpha threshold for signal generation. The optimization process evaluates different alpha values to find the one that produces the most balanced distribution of long, short, and hold signals in the training data, which is 1/3 for each class. By tuning alpha in this way, the strategy ensures that the model is trained on sufficient examples of each class, improving its ability to learn meaningful patterns and reducing bias toward the majority hold class.

Target Variable Definition

Trading signal labels

The trading signal labels used in this strategy were long (1), hold (0), and short (-1). Since neural networks require non-negative class labels, all values were shifted by +1 before training, resulting in long (2), hold (1), and short (0). After generating predictions, the labels were converted back to their original form to ensure correct interpretation and functionality in the backtesting process.

Threshold logic for class assignment

The trading signals generated for this strategy (buy, sell, and hold) are not evenly distributed. Most of the signals tend to be hold because small price changes do not meet the threshold for buy or sell. A 5-day shift was chosen instead of a 1-day shift to reduce noise from daily fluctuations, capture more meaningful short-term trends, and produce signals that are more actionable for trading. An alpha threshold of 0.027 was used to generate the signals, as this value was given by Optuna, which produces a reasonable balance between buy, sell, and hold classes. If the future price increases by more than 2.7%, the signal is long, if it decreases by more than 2.7%, the signal is short, otherwise, the signal is hold. A smaller alpha would generate too many buy or sell signals from minor price changes, while a larger alpha would result in mostly hold signals.

Distribution of signals in training data

The distribution of signals in training data before being processed by the neural networks was as follows:

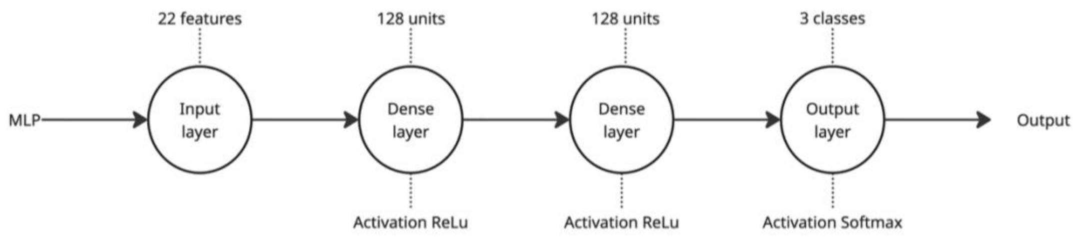
1	908
0	751
-1	605

Model Architecture and Design

MLP: Architecture diagram, layer descriptions, parameters

Component	Model 1	Model 2	Model 3	Model 4	Model 5
Input layer	Number of features	Number of features	Number of features	Number of features	Number of features
Hidden dense layers	2	3	2	3	2
Number of units	128	64	64	128	256
Output layer	Rectified Linear Unit	Hyperbolic Tagent	Sigmoid	Rectified Linear Unit	Hyperbolic Tagent
Optimizer	Adam	Adam	Adam	SDG	Adam
Output layer activation	Softmax	Softmax	Softmax	Softmax	Softmax
Loss function	Sparse categorical crossentropy	Sparse categorical crossentropy	Sparse categorical crossentropy	Sparse categorical crossentropy	Sparse categorical crossentropy
Metric	Accuracy	Accuracy	Accuracy	Accuracy	Accuracy

The highlighted model is the one that was chosen after comparing the performance of all models, the following diagram shows the implementation of the chosen model:



All five models share the same general architecture structure consisting of an input layer, multiple hidden dense layers, and an output layer configured for three-class classification. Each model begins with an input layer that receives the features. The hidden layer configuration varies between models: Models 1 and 3 include two dense layers, while Models 2 and 4 use three, and Model 5 contains two dense layers with a higher number of neurons. The number of units per layer also differs, Model 1 employs 128 neurons, Models 2 and 3 use 64, Model 4 uses 128, and Model 5 has 256 neurons per layer. Activation functions vary across models to explore nonlinear transformations: ReLU is used in Models 1 and 4, the hyperbolic tangent (Tanh) in Models 2 and 5, and the Sigmoid function in Model 3. Each network concludes with an output layer of three neurons activated by the Softmax function to perform multiclass classification. The models are trained using either the Adam optimizer (Models 1–3 and 5) or stochastic gradient descent (Model 4), with sparse categorical crossentropy as the loss function and accuracy as the evaluation metric. Epochs used to train all models was set to 100.

CNN: Convolutional structure, kernel sizes, pooling strategy

Component	Model 1	Model 2	Model 3	Model 4	Model 5
Input layer	Number of features	Number of features	Number of features	Number of features	Number of features
Convolutional layers	2	3	2	3	2
Convolutional filters	32	64	32	64	128
Pooling	2	2	2	2	2
Dense units	64	32	64	128	64
Activation	Rectified Linear Unit	Hyperbolic Tangent	Sigmoid	Rectified Linear Unit	Hyperbolic Tangent
Kernel size	3	3	3	3	3
Output layer activation	Softmax	Softmax	Softmax	Softmax	Softmax
Loss function	Sparce categorical crossentropy	Sparce categorical crossentropy	Sparce categorical crossentropy	Sparce categorical crossentropy	Sparce categorical crossentropy
Metric	Accuracy	Accuracy	Accuracy	Accuracy	Accuracy

The highlighted model is the one that was chosen after comparing the performance of all models.

The compared five neural network models have identical input and output structures but differing in convolutional and dense layer configurations, activation functions, and filter sizes. All models use the same input features, a pooling size of 2, kernel size of 3, softmax output activation, sparse categorical crossentropy function for loss, and accuracy as the evaluation metric. Variations arise in the number of convolutional layers (2 or 3), convolutional filters (ranging from 32 to 128), dense layer units (32 to 128), and activation functions (ReLU, Sigmoid, or Tanh). Model 2 and Model 4 employ three convolutional layers with smaller (Model 2) or medium (Model 4) filter sizes, while Model 5 uses the highest number of filters in a two-layer convolutional setup. The diversity in activation functions and dense units was intended as an exploration of different non-linear transformations and network capacities to balance feature extraction and classification performance. Epochs used to train all models was set to 100.

Training specifications

Loss function: Sparce categorical cross entropy

Sparse Categorical Cross entropy is used when you there is a need for multi-class classification with integer labels instead of one-hot encoded labels. The formula is as follows:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} \log(p_{i,y_i})$$

Where:

$$N = number\ of\ samples$$
$$y_i = true\ class\ label\ for\ sample\ i$$
$$p_{i,y_i} = predicted\ probability\ for\ the\ true\ class\ y_i for\ sample\ i$$

It is used for MLP and CNN because it compares the predicted probabilities to integer class labels without needing one-hot encoding, which is necessary in the case of the three classes for buy, sell and hold.

Optimizer: Adam

The Adam optimizer, short for Adaptive Moment Estimation, is an algorithm used to train neural networks efficiently. It combines the benefits of momentum and adaptive learning rates: momentum helps the optimizer move smoothly in the direction of consistent gradients, while adaptive learning rates adjust how big each step is for every parameter based on how large or small its recent gradients have been. Adam keeps track of the first moment (the mean of past gradients) and the second moment (the variance of past gradients) and uses these to scale the parameter updates. It also applies bias correction to ensure the estimates are accurate at the start of training. The result is an optimizer that often converges faster, handles noisy or sparse gradients well, and requires minimal tuning, making it especially suitable for deep models like MLPs and CNNs.

Optimizer: SGD

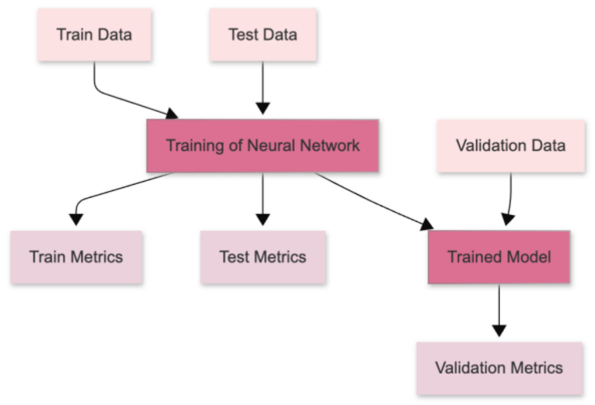
The Stochastic Gradient Descent (SGD) is an optimization algorithm for training neural networks. It works by updating model parameters in the direction that reduces the loss, using the gradient computed from a small batch of data rather than the entire dataset. This approach introduces some noise in the updates, which can help the model escape shallow local minima and find better solutions. It is memory-efficient, easy to implement, and forms the foundation for more advanced optimizers, making it suitable for models like MLPs and CNNs where iterative weight updates based on smaller batches are required.

Batch size

Batch size is a key hyperparameter in training neural networks that determines the number of training examples the model processes before updating its weights. Using a small batch size can make training noisy but helps the model generalize better, while a large batch size gives more stable and accurate gradient estimates but requires more memory and may lead to slower generalization.

MLFlow Experiment Tracking

Experiment setup and logging structure



For each of the models implemented, both MLPs and CNNs, the training is carried out using the previously defined training dataset, while the testing dataset is used to evaluate performance and monitor improvement across epochs. This process yields performance metrics for both the training and testing sets, such as accuracy, loss, and F1 score. Once training is completed, the model is evaluated on completely unseen data, the validation dataset, and the corresponding metrics (validation accuracy, loss, and F1 score) are obtained. It should be noted that, in MLFlow, due to the way variables were assigned, the test metrics appear as validation accuracy, loss, and F1 score, while the actual validation metrics were logged as test accuracy, loss, and F1 score.

Parameters tracked for each model

The parameters tracked for both the MLP and CNN models were:

- Batch size: The batch size defines the number of training samples processed before the model's internal parameters are updated. Smaller batch sizes allow for more frequent updates and can improve generalization but may introduce noise during training. Larger batches provide more stable gradient estimates but require greater computational resources and may converge to less optimal minima.
- Epochs: An epoch refers to one complete pass through the entire training dataset. During each epoch, the model processes all training samples (in batches), updating its weights to minimize the loss function. Increasing the number of epochs allows the model to learn more complex patterns but also increases the risk of overfitting if training continues beyond the optimal point.
- Optimizer: The optimizer determines how the model's weights are updated based on the computed gradients from the loss function. It controls the learning process by adjusting the magnitude and direction of weight changes. Only Adam and SGD were used.

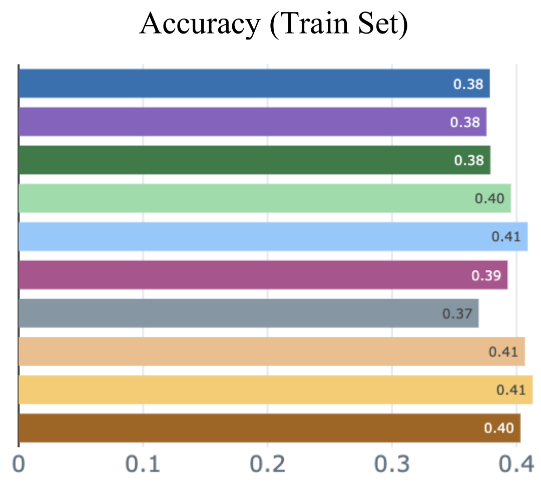
Metrics recorded

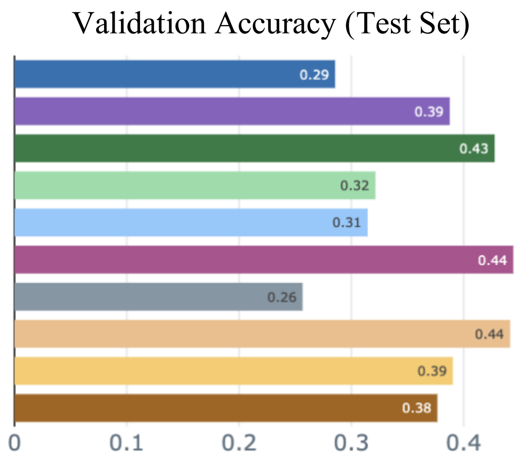
For each recorded metric, ten models are displayed in different colors. It should be noted that only five distinct parameter configurations were used for both the MLP and CNN models. The ten models appear because each parameter set was trained twice, once for 50 epochs and once for 100 epochs, to assess whether increasing the number of epochs produced any significant differences in performance. The following graphs show the performance of each of the ten models across different metrics, for both MLP and CNN architectures.

Training/validation accuracy

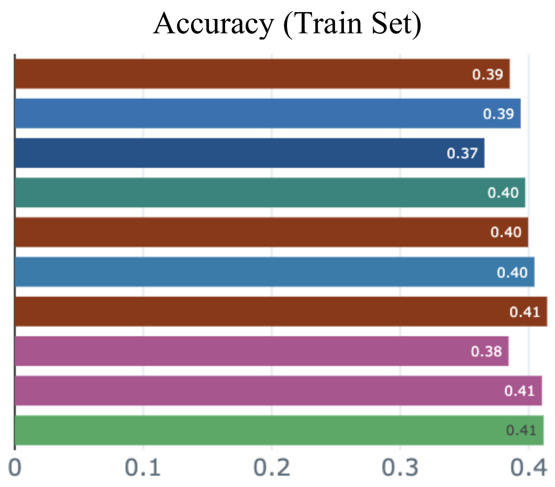
Training accuracy measures how well the model fits the data it was trained on, showing whether it can capture patterns in the features. Validation accuracy measures performance on unseen data, providing insight into the model's ability to generalize beyond the training set.

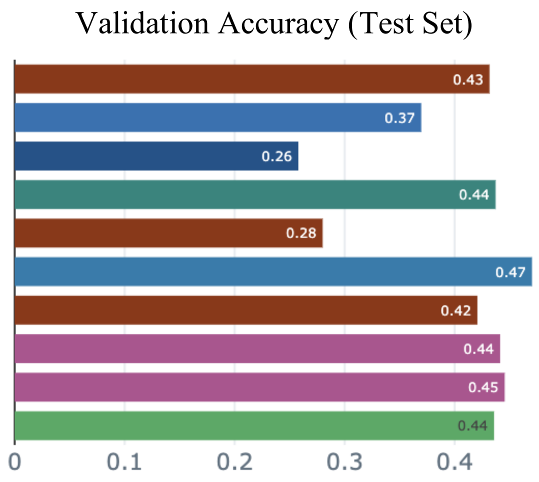
MLP





CNN

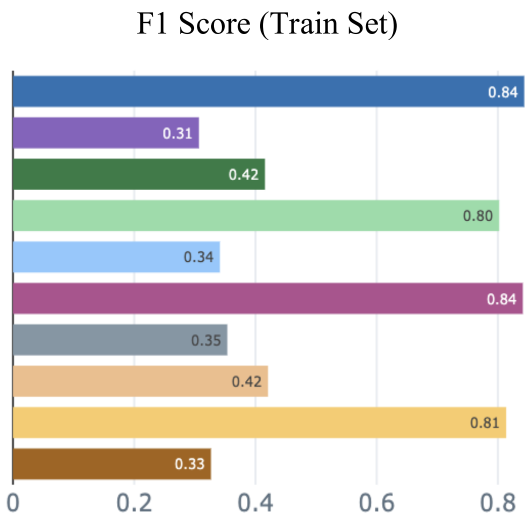


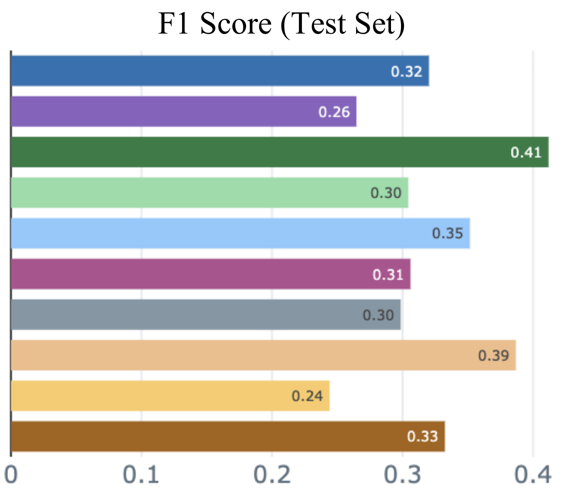


F1-score

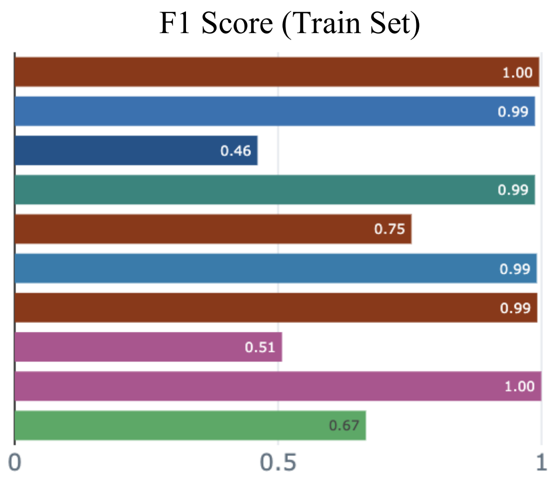
F1-score is a machine-learning evaluation metric that measures a model's accuracy by combining the precision and recall scores of a model, providing a balanced measure of classification performance. This is especially important for imbalanced classes like buy, sell, and hold signals, because it ensures the model performs well across all signal types, not just the majority class.

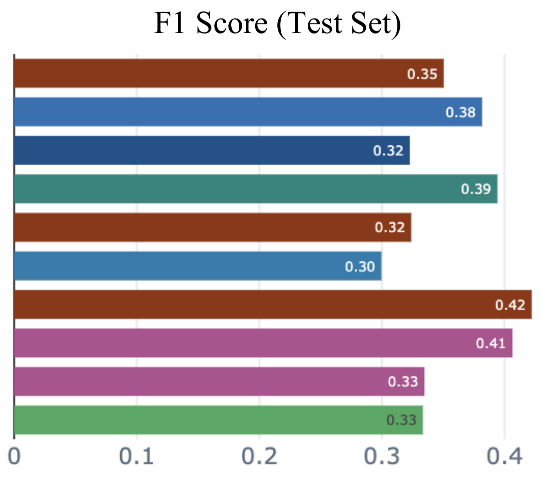
MLP





CNN

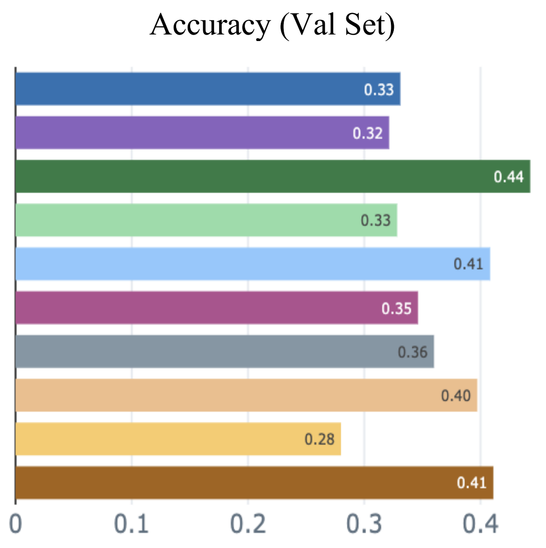


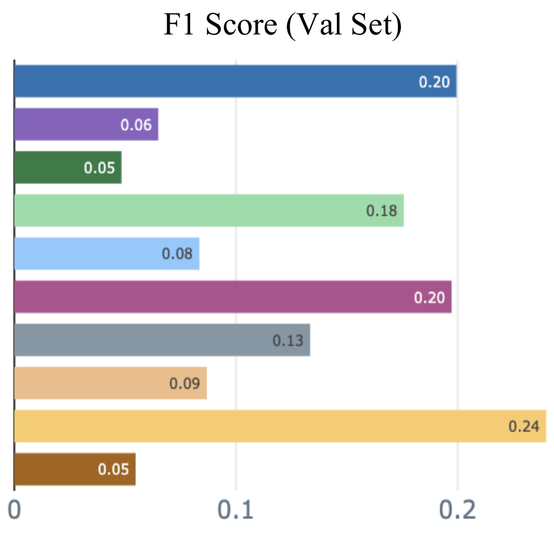


Test performance

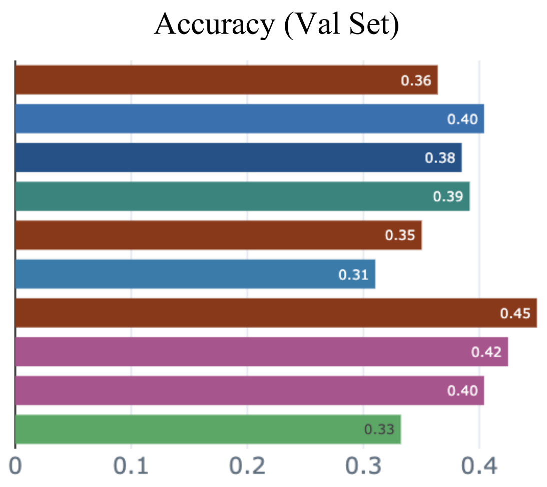
Test performance evaluates the final model on completely unseen data, the validation dataset, to measure its real predictive ability. This metric provides an unbiased evaluation of how well the model is likely to perform when generating trading signals in practice.

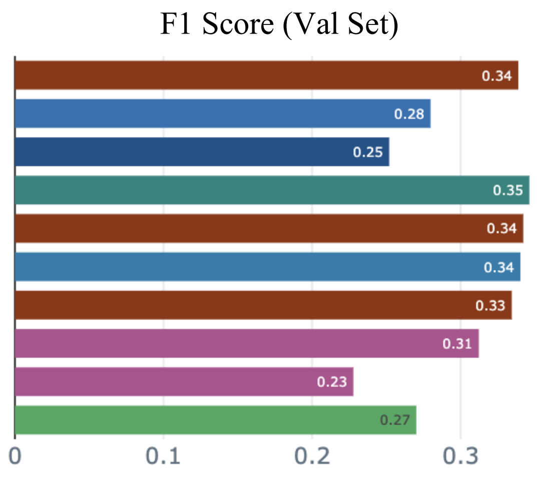
MLP





CNN





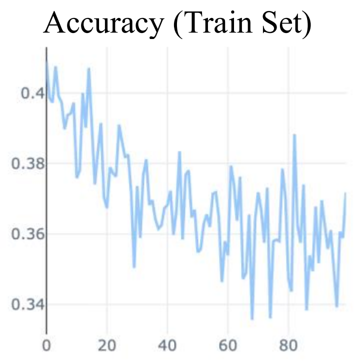
Model comparison table: MLP vs CNN results

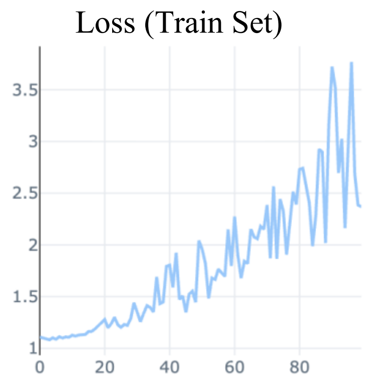
Metric	**MLP**	**CNN**
Selected model	2 dense layers 128 dense units Relu activation	3 conv layers 64 conv filters 128 dense units Relu activation
Accuracy on Train Set	0.4089	0.4045
Accuracy on Test Set	0.4087	0.4502
Accuracy on Val Set	0.4083	0.4497
F1 Score on Train Set	0.3414	0.992
F1 Score on Test Set	0.3516	0.4222
F1 Score on Val Set	0.0835	0.3344

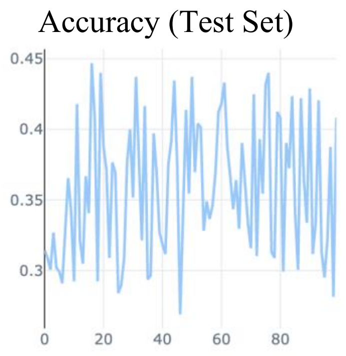
Selected model justification and performance on test set

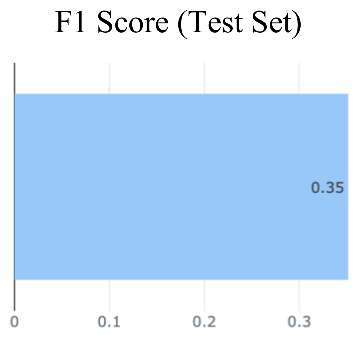
MLP

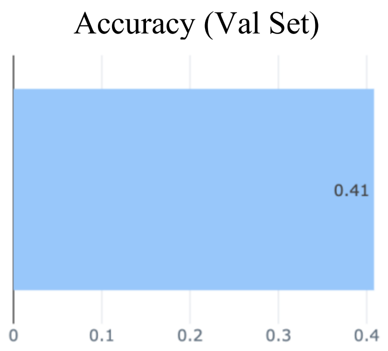
The selected MLP model that was chosen for backtest was the one with parameters: 2 dense layers, 128 dense units, Adam optimizer and ReLu activation. This model was chosen because it represented a good balance between accuracy in all three datasets.

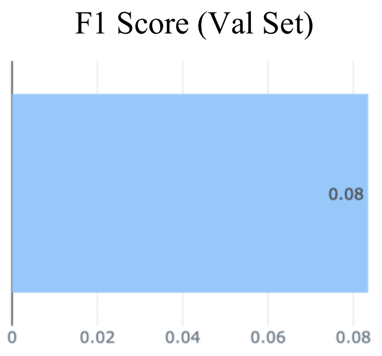






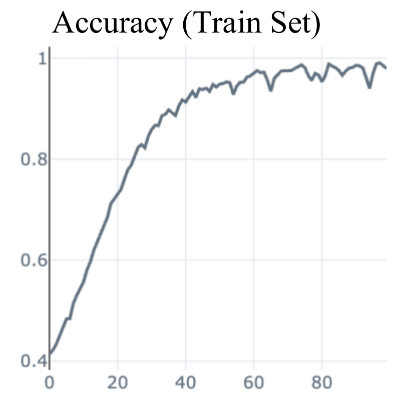


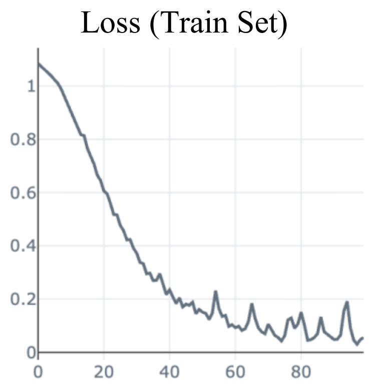


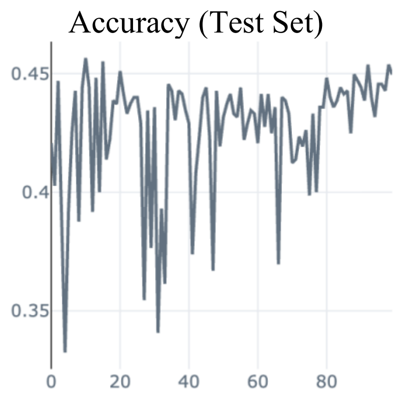


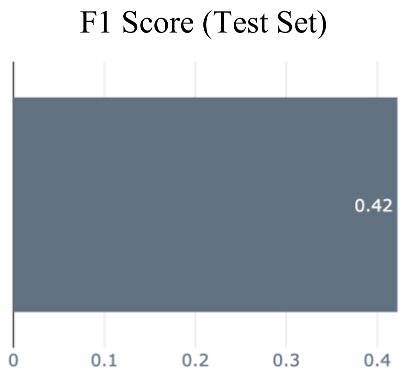
CNN

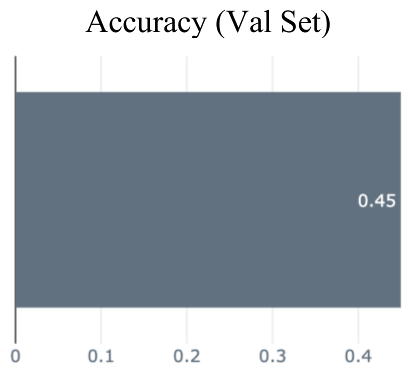
The selected CNN model that was chosen for backtest was the one with parameters: 3 convolutional layers, 64 filters, 128 dense units, Adam optimizer and ReLu activation. This model was chosen because it represented a good balance between accuracy in all three datasets.

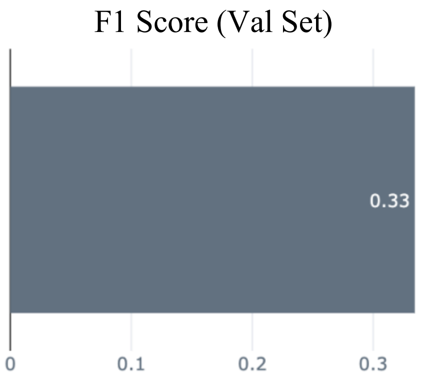






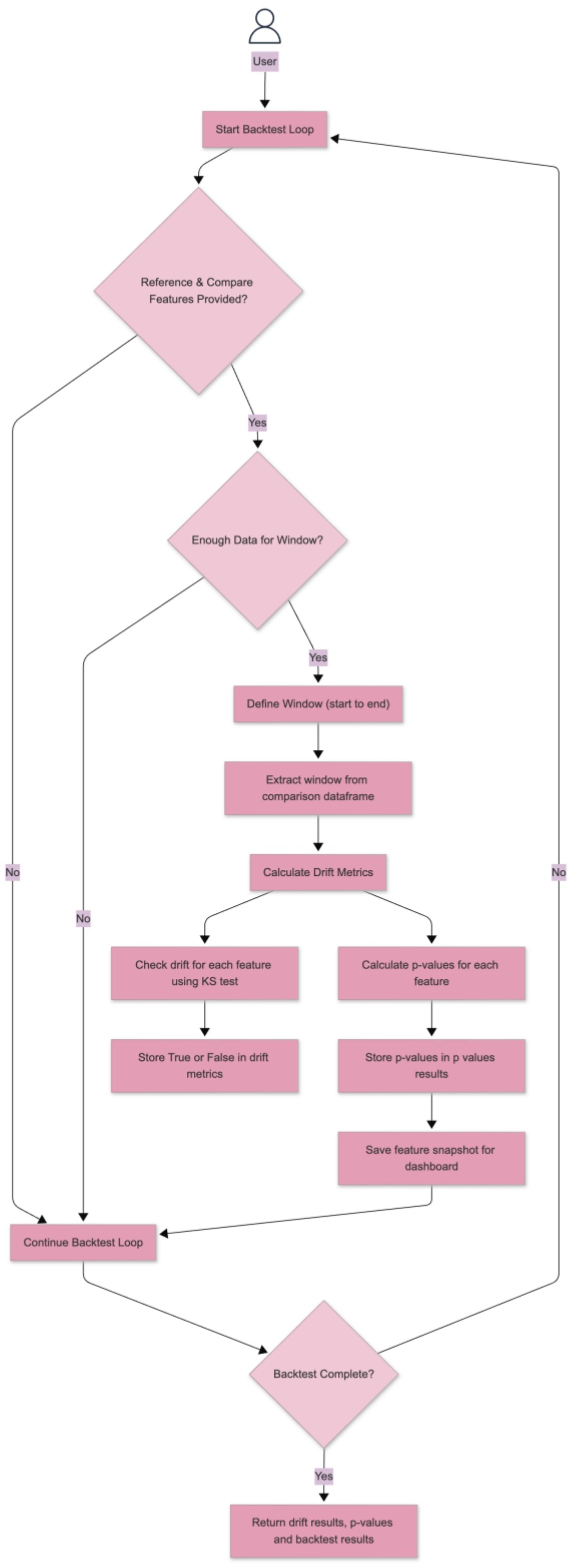






Data Drift Monitoring

Drift monitoring methodology


User
Start Backtest Loop
Reference & Compare Features Provided?
Yes
No
Enough Data for Window?
Yes
No
Define Window (start to end)
Extract window from comparison dataframe
Calculate Drift Metrics
Check drift for each feature using KS test
Calculate p-values for each feature
Store True or False in drift metrics
Store p-values in p values results
Save feature snapshot for dashboard
Continue Backtest Loop
Backtest Complete?
No
Yes
Return drift results, p-values and backtest results

The drift detection process is implemented as a structured backtest loop that iteratively evaluates distributional changes across time windows. The loop begins by verifying whether reference and comparison features have been provided. If sufficient data is available to define a valid window, the system proceeds to extract the corresponding segment from the comparison dataframe. Within each window, drift metrics are computed for every feature using the Kolmogorov–Smirnov (KS) test. The results of these tests, binary drift flags and associated p-values, are stored in dedicated metrics for downstream analysis. Additionally, a snapshot of the feature distributions is saved to support further diagnostics or model retraining. The loop continues until all windows have been processed, at which point the system returns a complete set of drift flags, p-values, and backtest results for interpretation and visualization. A window of 90 days with a window step of 30 is implemented.

Drift detection dashboard description

A dashboard that enables users to monitor statistical drift in features used by neural network models for financial forecasting was created. Drift detection is crucial for identifying when the data distribution changes over time, which can degrade model performance and indicate the need for retraining or adaptation.

Key components:

1. User Input Panel (Sidebar)

    - Ticker Input: Specify the stock or asset ticker to analyze, by default, the ticker shown is the one for Lululemon.
    - Run Button: Execute drift analysis on the selected ticker.

2. Feature Distribution Analysis

    - Overlayed Histograms: Compare feature distributions across datasets with purple for the reference distribution of the training set, blue for the comparison distribution of the test set and finally,
    - Purpose: Visualize how feature distributions evolve over time and detect shifts relative to the reference set.

3. P-Value Plots (Time Windows)

    - Show p-values computed over sequential time windows for both Test and Validation sets.
    - Red Shaded Area: Indicates p-values below 0.05, signaling statistically significant drift.
    - Horizontal Dashed Line at 0.05: Marks significance threshold.
    - Purpose: Identify specific periods where features significantly drift from the training distribution.

4. Statistics Tables

   - Summarize average p-values and drift flags for each feature across Test and Validation sets.
   - Highlight which features experience significant drift, helping prioritize monitoring or retraining efforts.

5. Drifted Windows Timeline

   - Visualizes the number of features drifted per time window, with red, yellow and green bands indicating high, moderate, and low drift intensity.
   - Purpose: Pinpoint exact windows where drift is most pronounced.

6. Top 5 Most Drifted Features

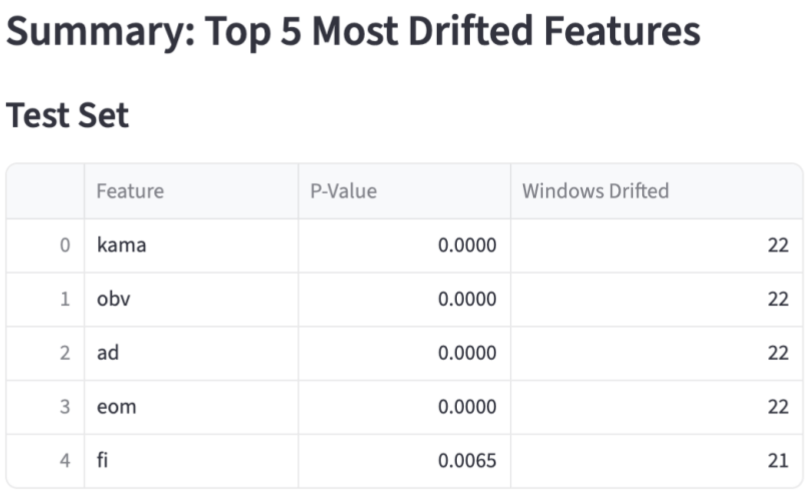
   - Shows the features with highest drift, including the average p-value and the number of windows in which drift occurred
   - Purpose: Identify which indicators are most unstable and may affect model predictions.

7. Explanatory Sections

   - Expanders provide detailed explanations for:

     - How histograms and p-values are interpreted.
     - How statistics tables are computed.
     - Understanding drifted windows and feature-specific behavior.

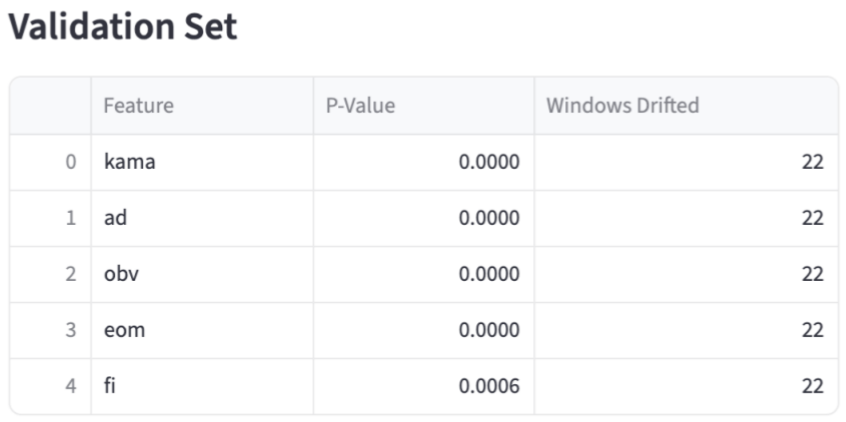
Features showing significant drift



Summary: Top 5 Most Drifted Features

Test Set

	Feature	P-Value	Windows Drifted
0	kama	0.0000	22
1	obv	0.0000	22
2	ad	0.0000	22
3	eom	0.0000	22
4	fi	0.0065	21



Validation Set

	Feature	P-Value	Windows Drifted
0	kama	0.0000	22
1	ad	0.0000	22
2	obv	0.0000	22
3	eom	0.0000	22
4	fi	0.0006	22

Statistical drift detection results are summarized for five indicators across two datasets: Test Set and Validation Set.

- P-Value: Measures the statistical significance of drift (lower values indicate more significant drift).
- Windows Drifted: The number of time windows in which drift was detected.

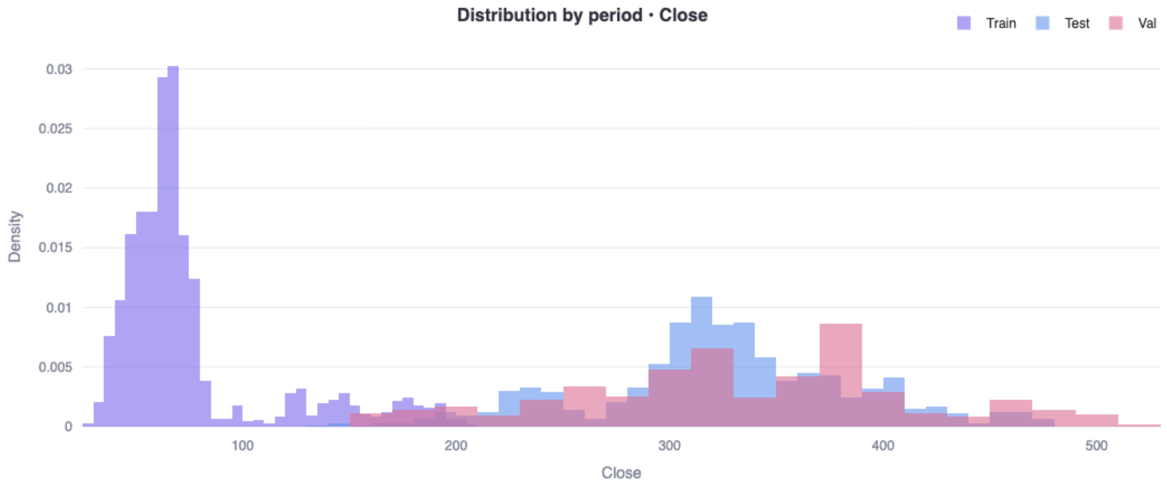
What These Indicators Measure

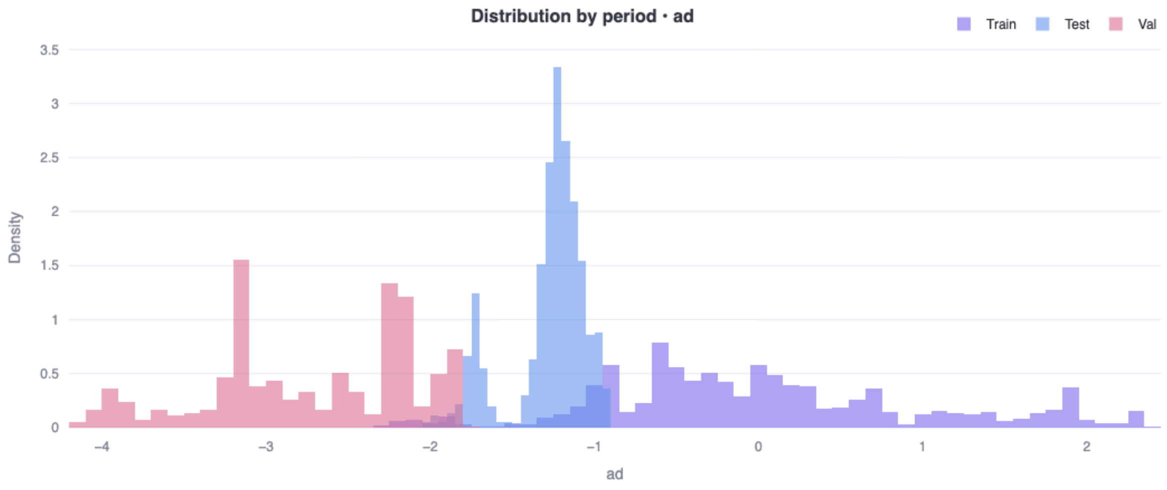
- KAMA (Kaufman Adaptive Moving Average): Adapts to market volatility, drift suggests changing volatility regimes.
- OBV (On-Balance Volume) and AD (Accumulation/Distribution): Volume-based indicators, drift implies altered buying or selling pressure.
- EOM (Ease of Movement): Reflects how easily price moves, drift signals shifts in liquidity or volatility.
- FI (Force Index): Combines price and volume, drift indicates changing momentum dynamics.

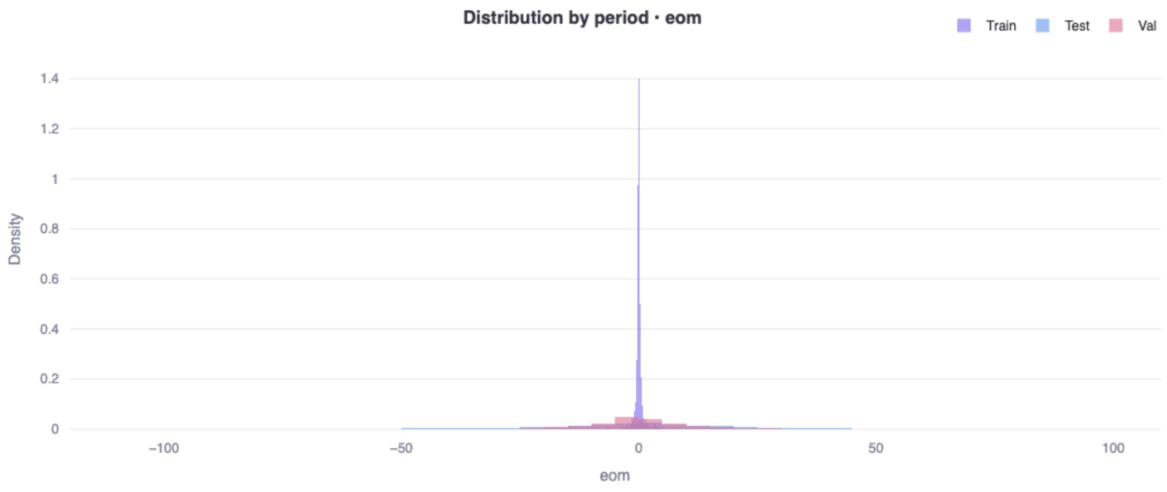
Why So Much Drift?

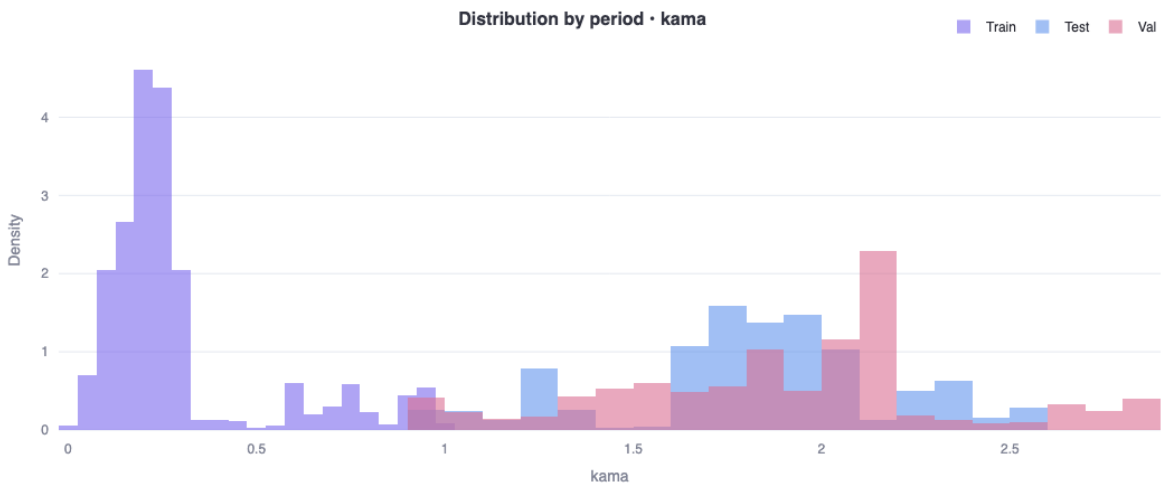
- Market Regime Change: Structural shifts may have altered price–volume relationships.
- Feature Sensitivity: These indicators are highly reactive to volatility and volume changes.
- Window Size: Shorter windows amplify moderate fluctuations.

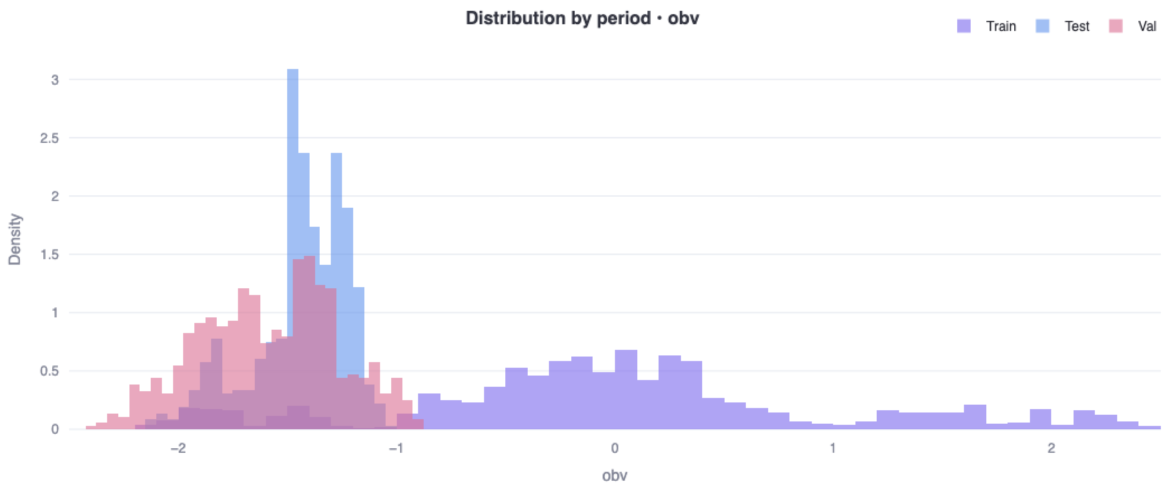
The features that drifted all the windows in test and validation test were the following:





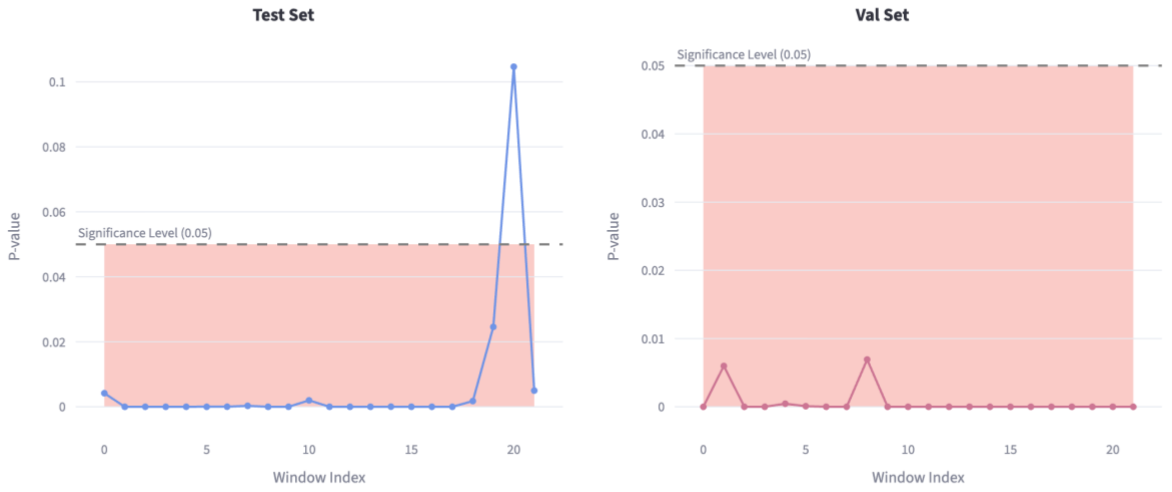




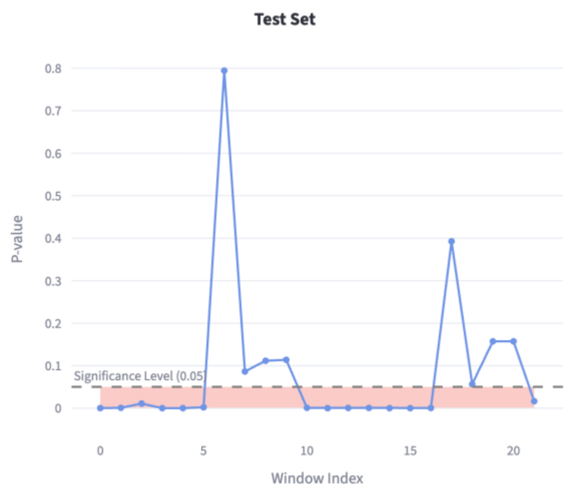


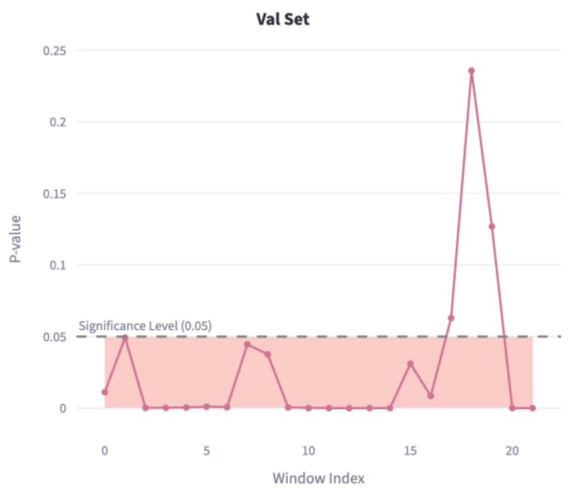
Other significant drifted features were the following:

Feature: Force Index

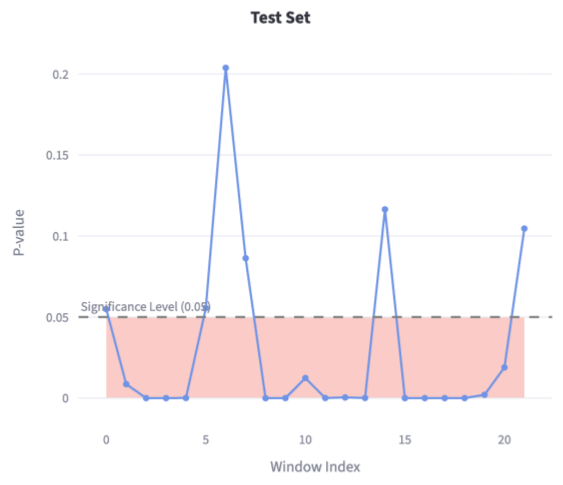


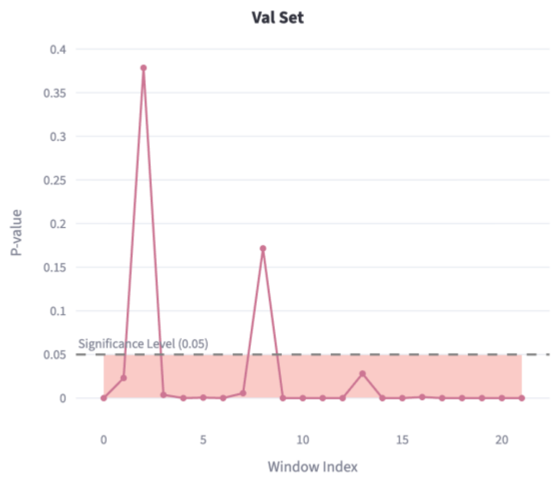
Feature: Chaikin Money Flow





Feature: Rate of Change with window of 20





Three overlayed histograms were generated per feature, followed by p-value plots for both the Test and Validation sets.

Overlayed Histograms

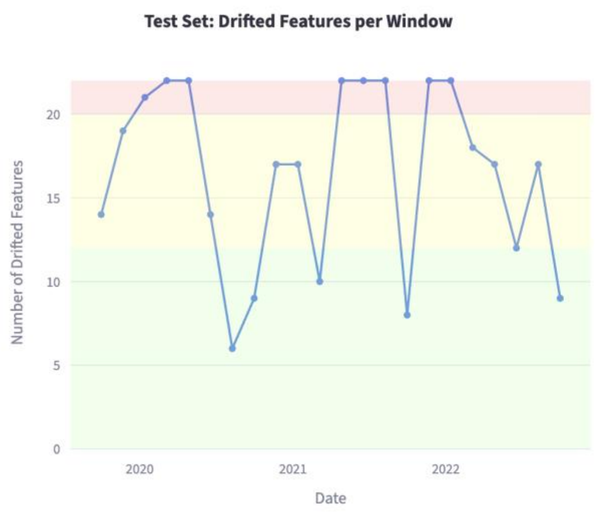
- Purple Histogram: Distribution of the feature in the Training set (reference).
- Blue Histogram: Distribution of the feature in the Test set (comparison).
- Pink Histogram: Distribution of the feature in the Validation set (comparison).

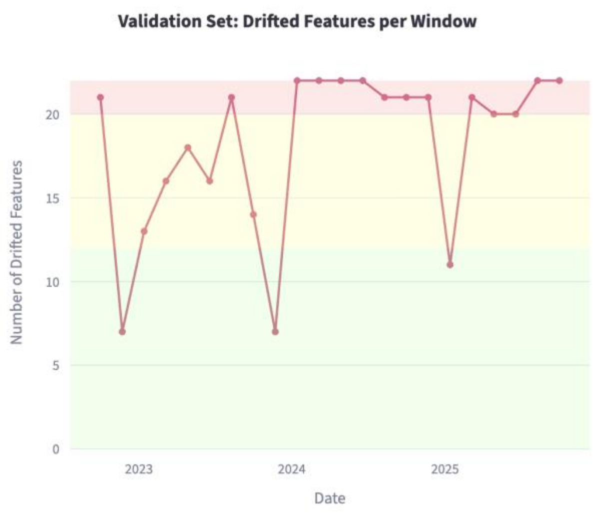
P-Value Plots

- Each plot shows the p-values computed over sequential time windows for the Test and Validation sets.

- The shaded red area indicates p-values below the significance threshold of 0.05, suggesting drift.
- A horizontal dashed line marks the 0.05 significance level for easy reference.

Timeline of drift periods with market context





These graphs show how many features drifted in each time window for the Test Set and Validation Set.

Graph Components

- X-Axis: Time windows (sequential segments of the dataset).
- Y-Axis: Number of features detected as drifted in that window.
- Colored bands:
    - Red: High drift
    - Yellow: Moderate drift
    - Green: Low drift

Interpretation

- Test Set: Drift fluctuates across windows, with occasional peaks in the red zone and lower values in the yellow. This pattern suggests intermittent instability in the financial indicators during this period, likely reflecting temporary market fluctuations, seasonal effects, or short-lived changes in trading behavior rather than sustained shifts. The model may still perform reasonably well during stable windows but could be sensitive to sudden volatility spikes.
- Validation Set: Drift is consistently high, with most windows in the red zone. This indicates persistent distribution shifts in the asset's financial indicators over the last three years, potentially due to structural changes in the market, evolving consumer behavior, or shifts in volatility patterns.

The sustained high drift signals that models trained on the earlier period may underperform unless retrained or adapted to the new regime.

Why Is This Important?

- High drift across windows can degrade model performance and signal regime changes.
- Consistent drift in the validation set may require retraining or feature re-engineering.
- Visualizing drift per window helps pinpoint when and where the data distribution changes most.

Market context

Market context provides essential insight into the causes of data drift observed across financial and trading-related datasets for LULU. Product and brand developments can significantly influence merchandising, marketing, and consumer engagement metrics, introducing volatility into features that were previously stable. Seasonal drops and new collections may shift inventory dynamics and demand cycles, while influencer campaigns or brand collaborations often trigger sudden spikes in traffic and conversions, distorting historical patterns. Financial and operational factors also contribute: quarterly earnings reports can cause abrupt changes in sentiment and trading volume, and store expansions or closures may alter regional sales and foot traffic metrics. Additionally, supply chain adjustments, whether due to delays, cost fluctuations, or vendor changes, can have downstream effects. Finally, external macroeconomic conditions further amplify the risk of drift by affecting consumer behavior, market sentiment, and overall trading activity.

More detailed and complete information in the features' data drift can be seen on the dashboard streamlit app.

Backtesting Methodology

Signal generation from model predictions

The backtest uses the predicted signals generated by the trained neural network models (MLP and CNN). Each prediction corresponds to a trading action: buy (1), hold (0), or sell (-1). A buy signal opens a long position, a sell signal opens a short position, and a hold signal keeps the current position unchanged. The signals were applied to the historical test data to replicate how the model would have traded in real time.

Position sizing and management rules

The backtest starts with an initial capital of $1,000,000 and uses a fixed position size of 100 shares per trade. No leverage is applied. Each long or short trade is opened when a corresponding signal appears and remains active until either a take profit of 14% or a stop loss of 7% is triggered. Transaction costs are included using a commission of 0.125% and an anual borrow rate of 0.25% for short positions, which is converted to a daily rate and charged on each trading day. The portfolio value is updated at each step, considering cash, open positions, and borrowing costs. Win and loss counts are recorded to calculate the win rate and total number of trades at the end of the simulation.

Assumptions and limitations of your backtest implementation

The backtest implementation assumes a fixed trade size for all positions, with each trade consisting of a predetermined number of shares. Transaction costs are included as a uniform commission percentage for all trades, and short positions have a daily borrowing cost. Stop-loss and take-profit levels are applied uniformly across all trades, and it is assumed that positions can be closed exactly at these levels. The backtest relies entirely on the final signal column to generate trades, interpreting signals as buy, sell, or hold. Trades occur only if sufficient cash is available and are executed at daily closing prices, which are also used to value open positions. Long positions are not leveraged beyond the available capital.

This approach has several built-in limitations. Risk management is simplified, with fixed position sizing that does not adapt to volatility, drawdowns, or portfolio-level exposure. Short selling is assumed to be unrestricted, with borrowing always available and a daily borrow cost applied. Furthermore, margin calls and other practical constraints are ignored.

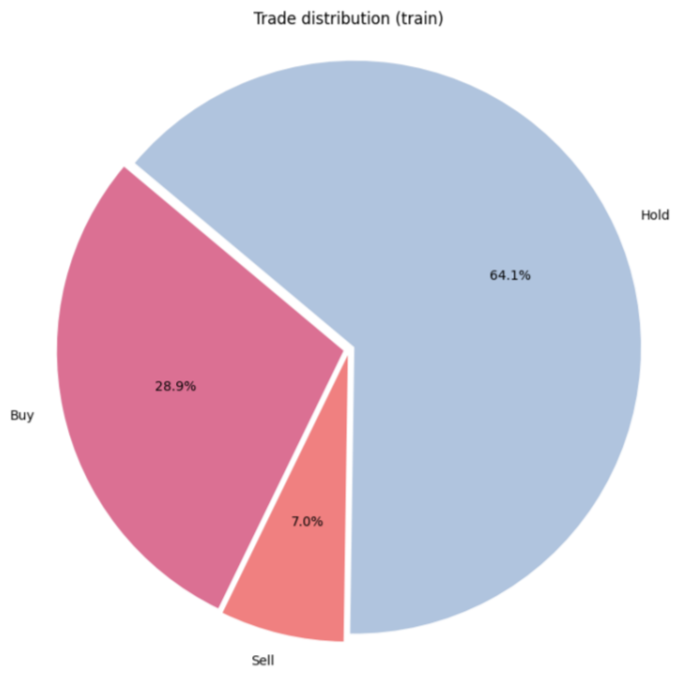
Finally, while the backtest includes data drift detection using a fixed window size and step, this analysis does not influence trade execution. Performance metrics are limited to win rate, counts of buy, sell, and hold actions, and portfolio value over time, without considering drawdowns, risk-adjusted returns, or other more advanced performance measures. The results are constrained by the available historical data, and past performance may not necessarily reflect future outcomes.

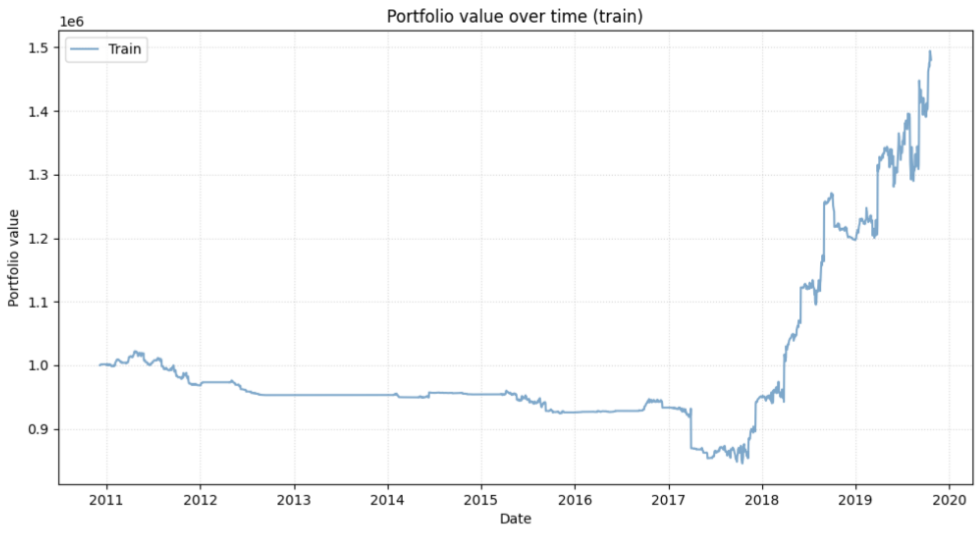
Results and Performance Analysis

Equity curve plots and trade distribution graphs

MLP

Train

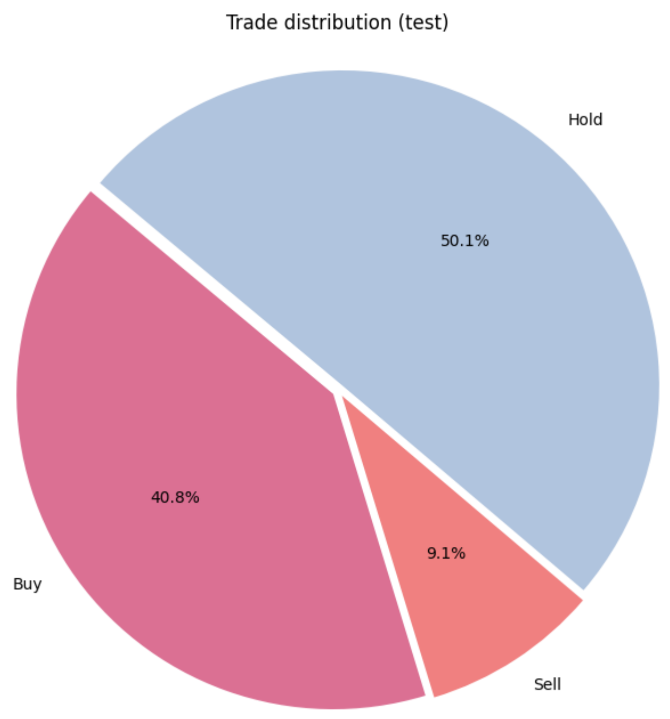


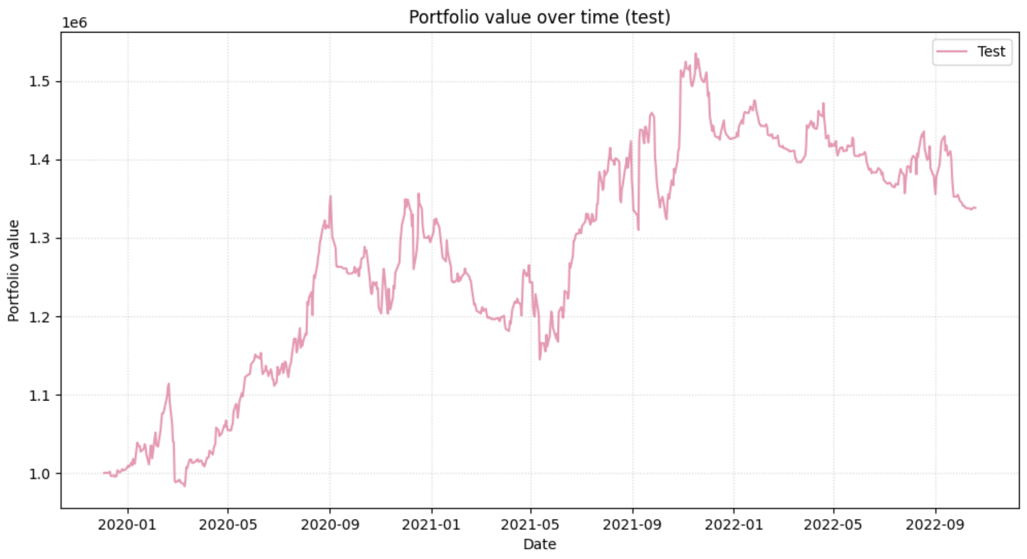


As we can see, the MLP strategy on the training set shows a clear predominance of hold signals, which account for 64.1% of the total signals. This indicates that the strategy tends to remain passive most of the time, avoiding unnecessary trades during periods of low market momentum. Buy signals represent 28.9%, suggesting that the model identifies upward opportunities with moderate frequency, while sell signals account for only 7.0%, implying that bearish opportunities are less frequently detected.

Examining the portfolio value over time, performance remains relatively stable from 2011 to around 2017, with minor downward movements indicating small or flat losses during some periods. From late 2017 onward, the portfolio experiences a strong upward trend, increasing sharply and peaking close to $1.5 million by 2020. This growth phase shows that the MLP strategy was able to capture favorable market movements effectively.

Test

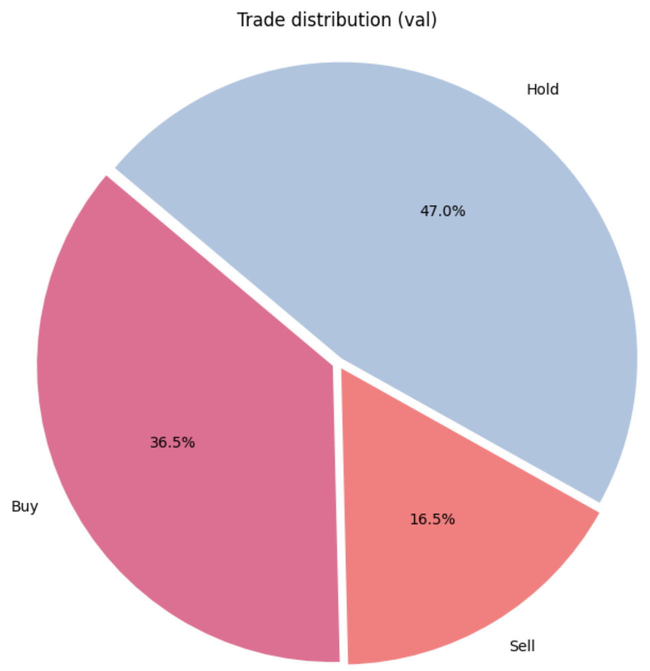


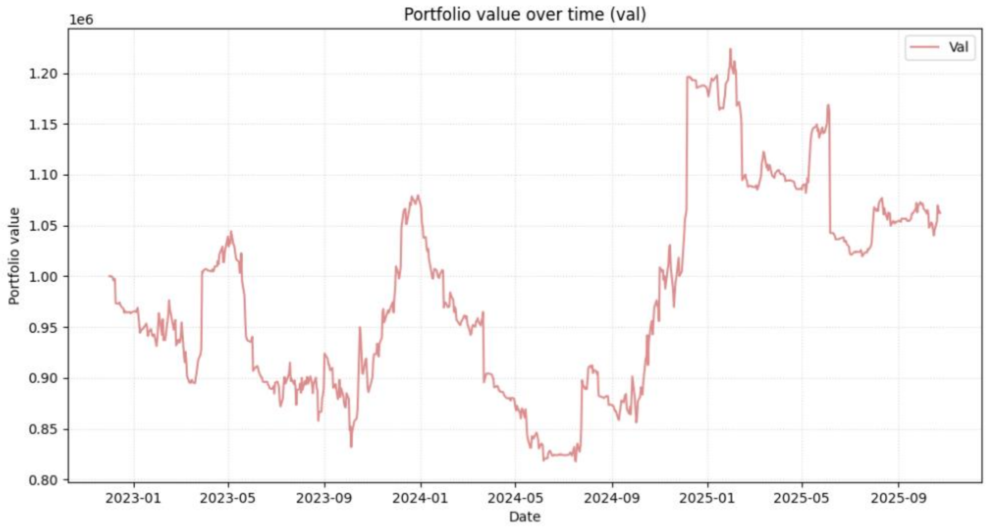


As we can see, the MLP strategy on the test set presents a more balanced trade distribution compared to the training period. Hold signals account for 50.1% of the total, indicating moderate inactivity but less than during training. Buy signals increase to 40.8%, reflecting a more active approach in identifying bullish opportunities, while sell signals remain relatively low at 9.1%, showing that the strategy continues to favor long positions over short ones.

Examining the portfolio value over time, the value follows a consistent upward trajectory with some downward movements. We can observe that even between 2021 and 2022, it reached a value of more than $1.5 million. This indicates strong initial performance and the strategy's ability to capture profitable trends. From late 2021 onward, however, fluctuations and a gradual decline appear, suggesting that market conditions became less favorable or that the model's predictive strength weakened under more volatile conditions. Despite this, the final portfolio value remains around $1.35 million, demonstrating that the MLP strategy maintained profitability and robustness during the testing period.

Validation

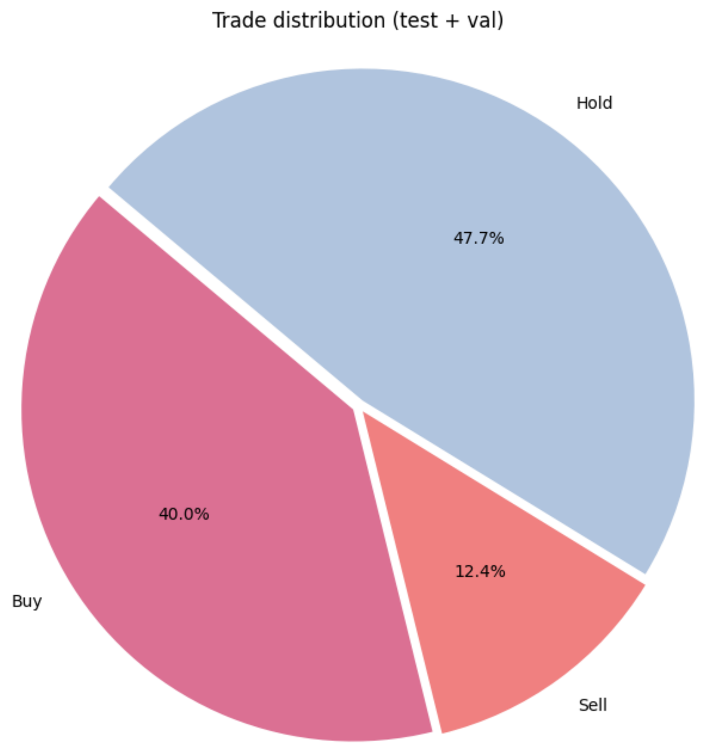


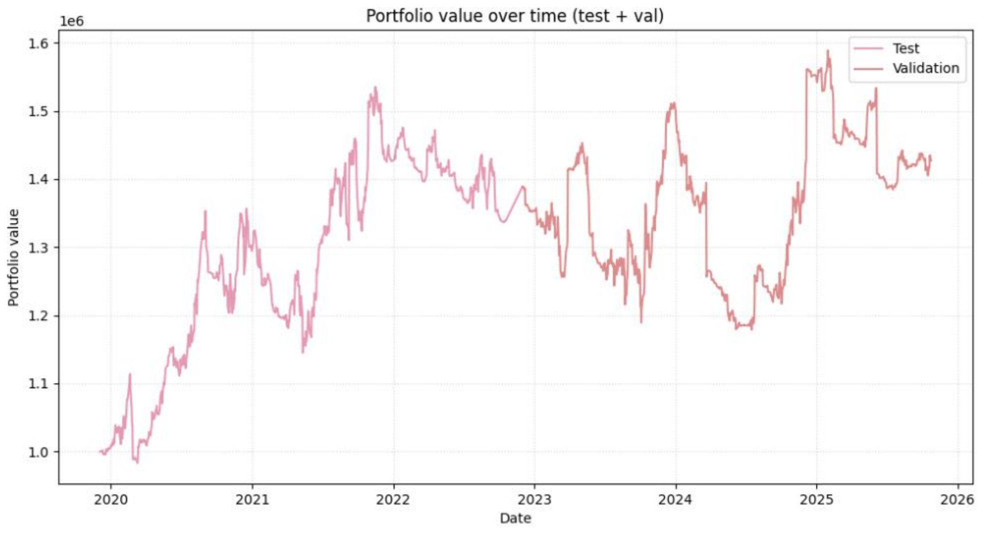


As we can see, the MLP strategy on the validation set exhibits a more diversified trade distribution, with hold positions representing 47.0% of total actions. This reduction in holding frequency compared to previous periods suggests that the model became more reactive to market signals. Buy signals account for 36.5%, reflecting a continued emphasis on upward opportunities, while sell signals increase to 16.5%, indicating that the model is now more responsive to bearish trends and seeks to capture profits during downward movements. This more balanced distribution reflects a shift toward a flexible and adaptive trading behavior under unseen market conditions.

Analyzing the portfolio value over time, the strategy experiences several drawdowns, reaching lows around $0.82 million during 2024. However, a strong recovery occurs in early 2025, with the portfolio climbing above $1.2 million before stabilizing near $1.05 million toward the end of the period. This pattern suggests that while the MLP model is capable of recovering from adverse conditions, it also encounters periods of heightened volatility. Overall, the strategy maintains a neutral outcome on the validation set, demonstrating resilience and adaptability, though with limited consistent profitability.

Test and validation



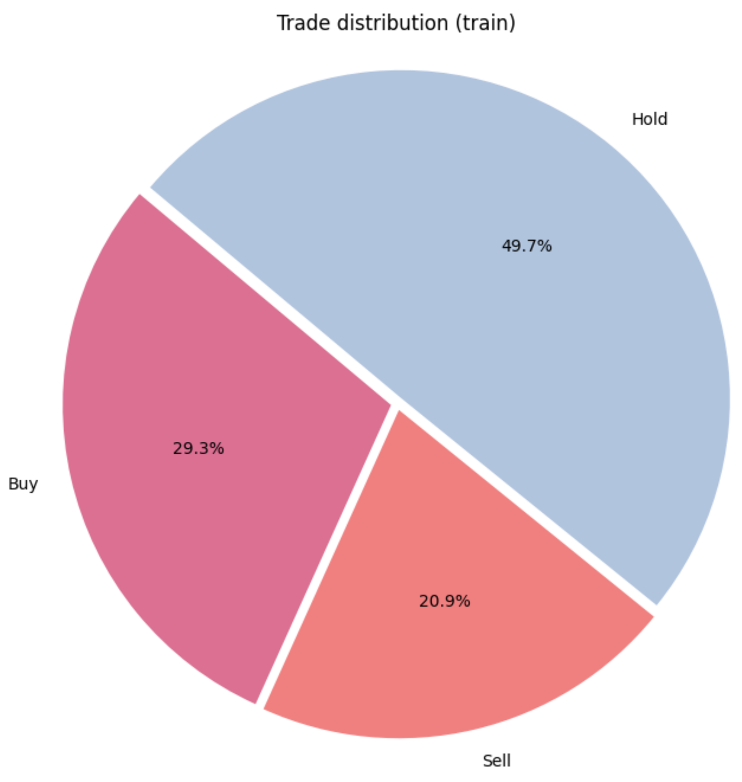


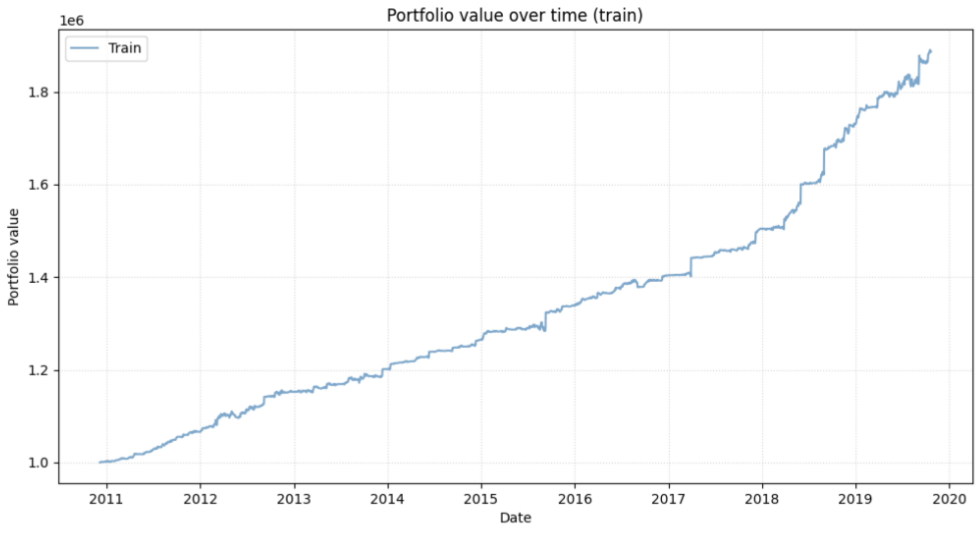
As we can see, when combining the test and validation periods, the MLP strategy maintains a relatively consistent trade distribution. Hold signals represent 47.7% of total actions, showing that the model preserves a moderately conservative stance. Buy signals remain strong at 40.0%, confirming the model's continued focus on capturing bullish opportunities, while sell signals account for 12.4%, indicating that although the model includes bearish positioning, its primary exposure remains long. This balanced mix suggests that the MLP retains its trading behavior across different market regimes without overfitting to a specific period.

Regarding portfolio performance, the combined curve shows a clear continuation of the upward trend seen in the test phase, followed by more pronounced fluctuations during the validation period. The portfolio peaks at approximately $1.55 million in early 2022, and then goes through several cycles of decline and recovery. Despite the increased volatility between 2023 and 2025, the value stabilizes around $1.4 million toward the end. Overall, this combined performance indicates that the MLP strategy sustains profitability over extended horizons, demonstrating robustness and adaptability under varying market conditions, though with some sensitivity to high-volatility phases.

CNN

Train

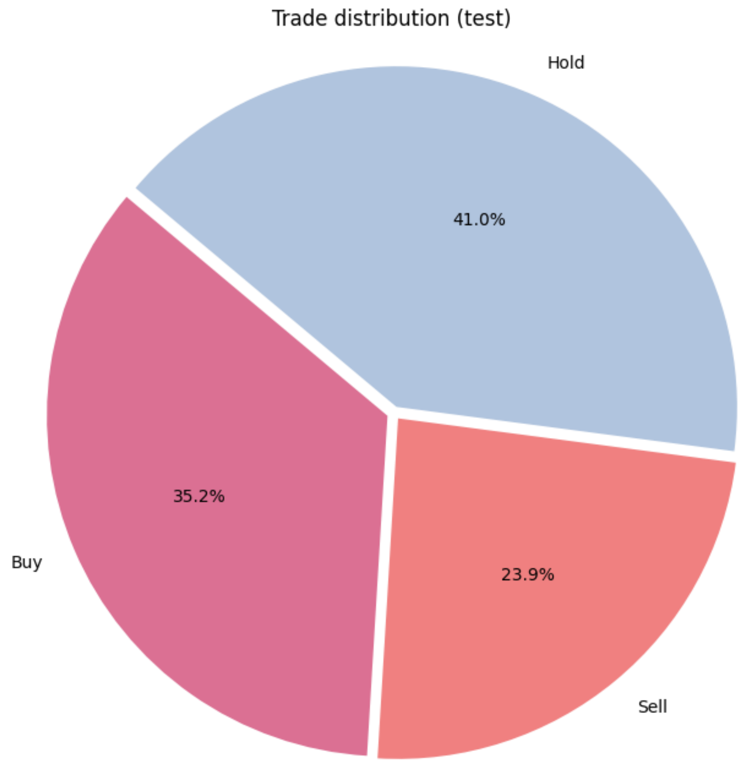

Trade distribution (train)
Hold
49.7%
Buy
29.3%
Sell
20.9%

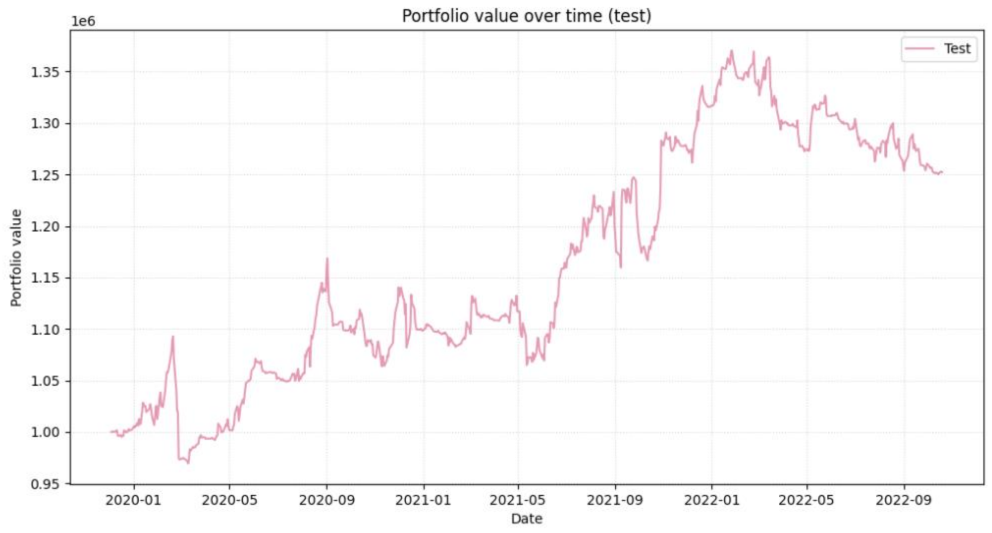

Portfolio value over time (train)
1e6
Train
1.8
1.6
1.4
1.2
1.0
Portfolio value
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
Date

As we can see, the CNN strategy on the training set exhibits a well-balanced trade distribution. Hold signals represent 49.7% of total decisions, indicating that the model maintains a moderate level of activity while avoiding excessive trading. Buy signals account for 29.3%, reflecting a healthy level of long exposure, while sell signals make up 20.9%, suggesting that the model actively engages in short positions when necessary. This distribution highlights a versatile strategy capable of adapting to both bullish and bearish conditions rather than favoring one side of the market excessively.

When analyzing the portfolio value over time, the CNN model demonstrates a remarkably consistent upward trajectory from 2011 through 2020, growing steadily and surpassing $1.9 million by the end of the training period. The curve displays minimal drawdowns and a smooth compounding effect, indicating strong model stability and effective trade execution. Overall, the CNN strategy achieves robust and sustained profitability during training, suggesting that it effectively captures long-term trends and manages risk efficiently within this dataset.

Test

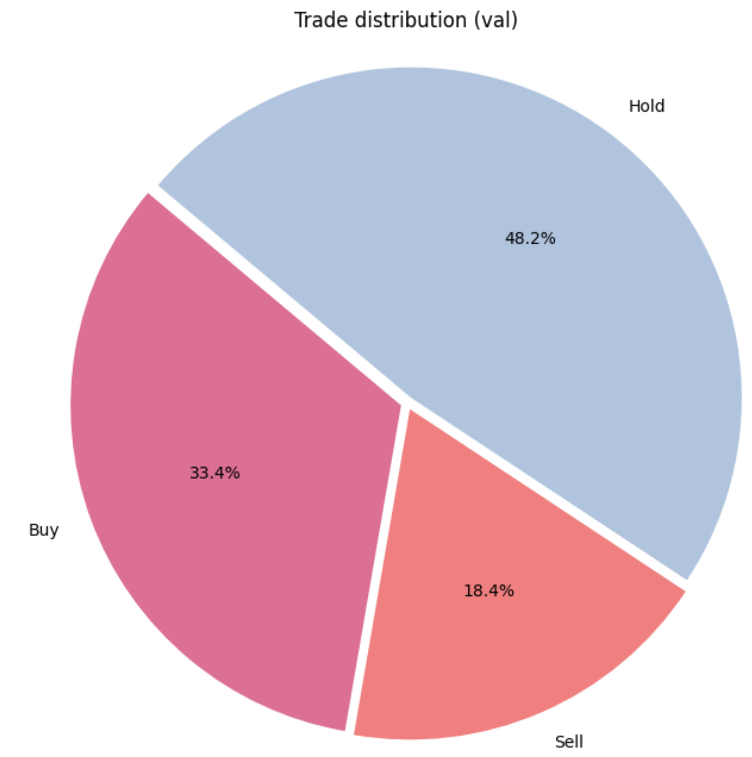


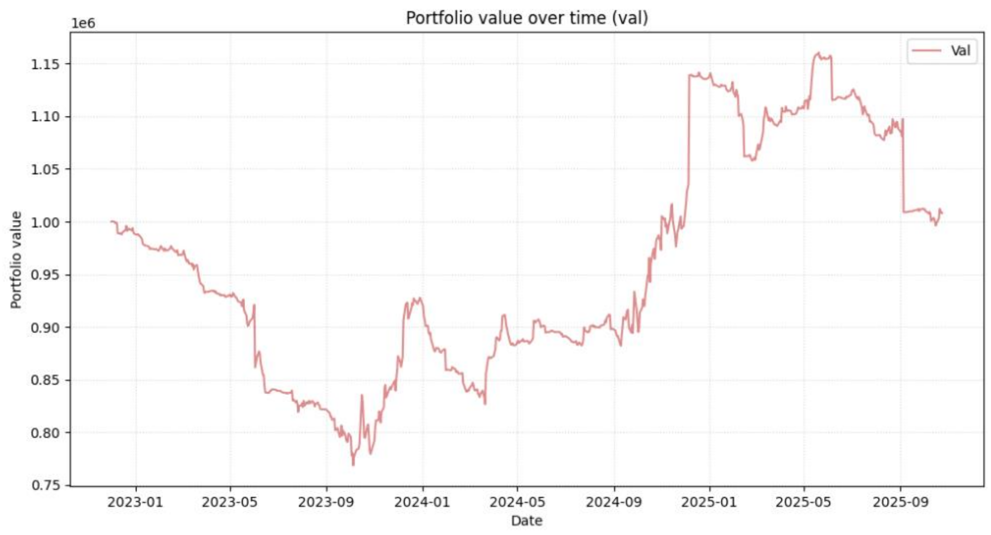


As we can see, the CNN strategy on the test set presents a more active trading profile compared to the training phase. Hold signals represent 41.0% of total decisions, reflecting a decrease in inactivity and suggesting that the model reacts more dynamically to new market information. Buy signals account for 35.2%, indicating balanced exposure to upward movements, while sell signals rise to 23.9%, showing stronger participation in short positions. This trade distribution suggests that the CNN model adapts well to out-of-sample data, maintaining flexibility and identifying both bullish and bearish opportunities.

Regarding portfolio performance, the strategy displays a clear upward trajectory, reaching a peak of approximately $1.35 million by early 2022. Afterward, the portfolio experiences moderate drawdowns and periodic fluctuations, stabilizing around $1.25 million toward the end of the test period. Despite these variations, the overall outcome remains solidly profitable, confirming that the CNN model continues to generate returns and generalize effectively beyond the training sample, preserving capital growth while keeping volatility under control.

Validation

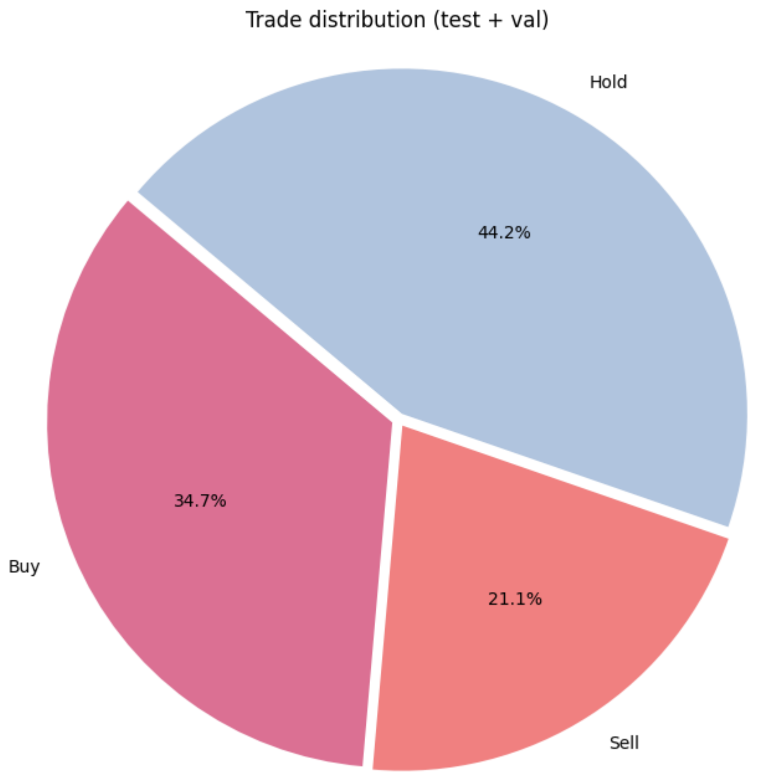


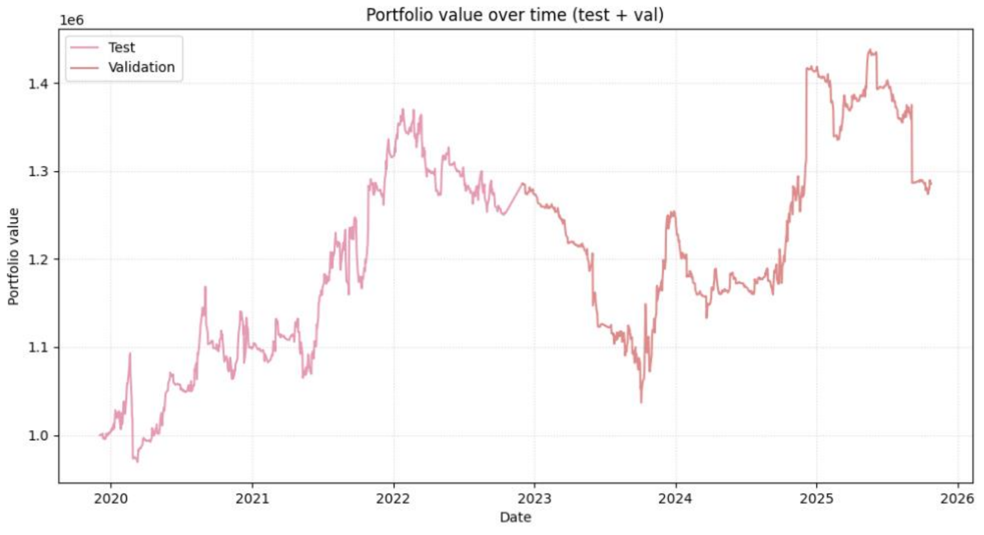


As we can see, the CNN strategy on the validation set maintains a balanced trading behavior, with hold signals representing 48.2% of total decisions. This reflects a moderate level of market participation and consistent signal confidence. Buy signals account for 33.4%, indicating that the model continues to prioritize long opportunities, while sell signals comprise 18.4%, showing active but less frequent engagement in short positions. This distribution suggests that the CNN model retains a structured decision-making pattern, avoiding overtrading while remaining responsive to both bullish and bearish market conditions.

In terms of portfolio performance, the strategy experiences an initial decline during 2023, reaching approximately $780,000 by the end of that year. From early 2024 onward, however, the model shows a strong recovery, climbing to a peak around $1.15 million by mid-2025 before slightly decreasing toward $1 million at the end of the period, ending like it started. This trajectory highlights the model’s ability to rebound from adverse phases and capture favorable trends once conditions improve. Overall, the CNN strategy demonstrates resilience and adaptability during the validation phase, maintaining a neutral-to-positive outcome despite market volatility.

Test and validation





As we can see, when combining the test and validation periods, the CNN strategy maintains a coherent and balanced trading profile. The hold signals represent 44.2% of total actions, indicating a moderate trading frequency that allows the model to stay selective while remaining responsive to market changes. Buy signals account for 34.7%, reflecting a consistent long bias and the model's tendency to capture upward price trends, while sell signals represent 21.1%, confirming a steady but secondary focus on short positions. Overall, this trade distribution suggests that the CNN model preserves its decision-making structure across different data segments, maintaining consistency under varying market conditions.

Analyzing the combined portfolio performance, the strategy shows steady growth throughout the test phase, reaching a peak of approximately $1.38 million by early 2022. This is followed by a contraction period during 2023, where the portfolio drops close to $1.05 million, before a strong rebound in 2024–2025 that lifts it back toward its previous highs, over $1.4 million. Although minor declines appear toward the end of the validation window, ending at $1.3 million, the overall trajectory remains positive. This behavior highlights the CNN model's robustness and ability to recover from drawdowns while sustaining profitability over extended horizons.

Performance metrics

To evaluate the effectiveness of the strategy, several metrics were used to provide a more comprehensive view of its performance over several periods. The implemented metrics were as follows:

Sharpe Ratio

This metric measures the risk adjusted return relative to its volatility. The higher the value, the better the relationship between return and risk. It is calculated as follows:

$$Sharpe\ Ratio = \frac{Average\ return}{Volatility}$$

Sortino Ratio

This metric is similar to Sharpe Ratio but focuses on downside volatility by considering only negative returns. The higher the value, the better protection against losses. It is calculated as follows:

$$Sortino\ Ratio = \frac{Average\ return}{Downside\ volatility}$$

Calmar Ratio

This metric measures the return relative to the maximum loss risk. The higher the value, the better the risk-return efficiency. It is calculated as follows:

$$Calmar\ Ratio = \frac{Average\ return}{Max\ Drawdown}$$

Max Drawdown

This metric indicates the largest drop that can occur from a peak to a trough. The lowest the value, the lower the exposure to risk. It is calculated as follows:

$$Max\ Drawdown = \frac{Minimum\ value - Maximum\ value}{Maximum\ value}$$

MLP

Train

Sharpe Ratio	Sortino Ratio	Max Drawdown	Calmar Ratio
0.595938	1.095923	0.172984	0.273815

As we can see, the MLP strategy on the training set obtained a Sharpe ratio of 0.6, which being greater than 0.5 indicates that the strategy delivers positive returns relative to the risk taken. The Sortino ratio was 1.1, which, being greater than 1, shows that losses are relatively small compared to the returns. The maximum drawdown was 0.17, indicating that the largest peak-to-trough loss was moderate but still significant, and that the strategy preserves capital reasonably well. Finally, the Calmar ratio was 0.27, which, being below 1, suggests that the returns do not sufficiently exceed the maximum loss. However, this is not concerning given that the maximum loss itself is not that high.

Test

Sharpe Ratio	Sortino Ratio	Max Drawdown	Calmar Ratio
0.712801	1.216054	0.156061	0.731778

As we can see, the MLP strategy on the testing set obtained a Sharpe ratio of 0.71, which being greater than 0.5 indicates that the strategy delivers positive returns relative to the risk taken. The Sortino ratio was 1.22, which, being greater than 1, shows that losses are relatively small compared to the returns. The maximum drawdown was 0.16, indicating that the largest peak-to-trough loss was moderate but still significant, and that the strategy preserves capital reasonably well. Finally, the Calmar ratio was 0.73, which, being closer to 1, indicates that the returns are fairly high relative to the maximum loss, showing improved risk-adjusted performance compared to the training set.

Validation

Sharpe Ratio	Sortino Ratio	Max Drawdown	Calmar Ratio
0.20379	0.33753	0.242779	0.165452

As we can see, the MLP strategy on the validation set obtained a Sharpe ratio of 0.2, which being below 0.5 indicates that the risk-adjusted returns are relatively low. The Sortino ratio was 0.34, showing that losses are relatively high compared to gains and downside risk is not well-controlled. The maximum drawdown was 0.24, indicating that the largest peak-to-trough loss was significant, suggesting the strategy struggled to preserve capital during this period. Finally, the Calmar ratio was 0.17, which, being below 1, suggests that the returns do not sufficiently exceed the maximum loss, highlighting weaker performance on the validation set compared to training and test.

Test and validation

Sharpe Ratio	Sortino Ratio	Max Drawdown	Calmar Ratio
0.452176	0.742174	0.232327	0.326083

As we can see, the MLP strategy on the combined test and validation set obtained a Sharpe ratio of 0.45, which being below 0.5 indicates that the risk-adjusted returns are relatively low. The Sortino ratio was 0.74, showing that downside risk is somewhat controlled but losses are still relatively high compared to gains. The maximum drawdown was 0.23, indicating that the largest peak-to-trough loss was significant, suggesting the strategy struggled to preserve capital during this period. Finally, the Calmar ratio was 0.33, which, being below 1, suggests that the returns do not sufficiently exceed the maximum loss.

CNN

Train

Sharpe Ratio	Sortino Ratio	Max Drawdown	Calmar Ratio
2.362248	6.847423	0.014664	4.91459

As we can see, the CNN strategy on the training set obtained a Sharpe ratio of 2.36, which is well above 1, indicating excellent risk-adjusted returns. The Sortino ratio was 6.85, showing that downside risk is extremely low compared to gains, meaning losses are minimal relative to the positive returns. The maximum drawdown was 0.01, indicating that the largest peak-to-trough loss was very small, so the strategy preserves capital exceptionally well. Finally, the Calmar ratio was 4.91, which is very high, meaning that returns are far greater than the maximum drawdown, which itself is very low, reflecting outstanding performance on the training set.

Test

Sharpe Ratio	Sortino Ratio	Max Drawdown	Calmar Ratio
0.748934	1.213909	0.113005	0.749554

As we can see, the CNN strategy on the testing set obtained a Sharpe ratio of 0.75, which being greater than 0.5 indicates that the strategy delivers positive returns relative to the risk taken. The Sortino ratio was 1.21, showing that downside risk is reasonably controlled and losses are relatively small compared to gains. The maximum drawdown was 0.11, indicating that the largest peak-to-trough loss was moderate and that the strategy preserves capital reasonably well. Finally, the Calmar ratio was 0.75, which, being closer to 1, indicates that the returns are fairly high relative to the maximum loss, reflecting solid performance on the test set.

Validation

Sharpe Ratio	Sortino Ratio	Max Drawdown	Calmar Ratio
0.088197	0.140909	0.231746	0.051714

As we can see, the CNN strategy on the validation set obtained a Sharpe ratio of 0.09, which being below 0.5 indicates very low risk-adjusted returns. The Sortino ratio was 0.14, showing that downside risk is high relative to gains, meaning losses are significant compared to positive returns. The maximum drawdown was 0.23, indicating that the largest peak-to-trough loss was significant, suggesting the strategy struggled to preserve capital during this period. Finally, the Calmar ratio was 0.05, indicating that returns are far below the potential losses, reflecting weak performance on the validation set compared to training and test.

Test and validation

Sharpe Ratio	Sortino Ratio	Max Drawdown	Calmar Ratio
0.435649	0.706986	0.243376	0.206472

As we can see, the CNN strategy on the combined test and validation set obtained a Sharpe ratio of 0.44, which being below 0.5 indicates that the risk-adjusted returns are relatively low. The Sortino ratio was 0.71, showing that downside risk is somewhat controlled but losses are still relatively high compared to gains. The maximum drawdown was 0.24, indicating that the largest peak-to-trough loss was significant, suggesting the strategy struggled to preserve capital during this period. Finally, the Calmar ratio was 0.21, indicating that returns are low relative to the potential losses.

Trade statistics

Total trades

This statistic represents the total number of trades executed by the strategy, including both sells and buys. A higher number of trades indicate more activity or a shorter holding period. It is calculated as follows:

$$Total\ trades = Sells + Buys$$

MLP

Train

Buy signals	Sell signals	Hold signals	Total trades
909	219	2016	1128

As we can see, the MLP strategy on the training set generated a total of 1,128 trades, consisting of 909 buy signals, 219 sell signals, and 2,016 hold signals. This indicates that the strategy favored holding positions most of the time, with buying actions occurring more frequently than selling. The higher number of buy signals compared to sell signals suggests that the strategy identifies more opportunities to enter long positions, while the predominance of hold signals reflects that the strategy avoids unnecessary trades during periods of low market momentum.

Test

Buy signals	Sell signals	Hold signals	Total trades
500	111	614	611

As we can see, the MLP strategy on the testing set generated a total of 611 trades, consisting of 500 buy signals, 111 sell signals, and 614 hold signals. Compared to the training set, the total number of trades decreased, reflecting a more selective trading approach. The number of buy signals remains higher than sell signals, indicating that the strategy continues to favor entering long positions. Meanwhile, the predominance of hold signals shows that the strategy avoids unnecessary trades during periods of low market momentum, similar to the behavior observed in the training set.

Validation

Buy signals	Sell signals	Hold signals	Total trades
413	187	532	600

As we can see, the MLP strategy on the validation set generated a total of 600 trades, consisting of 413 buy signals, 187 sell signals, and 532 hold signals. Compared to the training and testing sets, the total number of trades slightly decreased, reflecting an even more conservative approach. The number of buy signals remains higher than sell signals, indicating a continued preference for long positions. The predominance of hold signals demonstrates that the strategy consistently avoids unnecessary trades during periods of low market momentum across all datasets.

Test and validation

Buy signals	Sell signals	Hold signals	Total trades
961	298	1146	1259

As we can see, the MLP strategy on the combined test and validation set generated a total of 1,259 trades, consisting of 961 buy signals, 298 sell signals, and 1,146 hold signals. The total number of trades is higher than in the individual test or validation sets, reflecting the accumulation of trading activity across both periods. Buy signals remain more frequent than sell signals, indicating that the strategy predominantly favors long positions. The predominance of hold signals shows that the strategy continues to avoid unnecessary trades during periods of low market momentum, ensuring trades are made only when significant opportunities arise.

CNN

Train

Buy signals	Sell signals	Hold signals	Total trades
928	662	1573	1590

As we can see, the CNN strategy on the training set generated a total of 1,590 trades, consisting of 928 buy signals, 662 sell signals, and 1,573 hold signals. This indicates that the strategy favored holding positions most of the time, with buying actions occurring more frequently than selling. The higher number of buy signals compared to sell signals suggests that the strategy identifies more opportunities to enter long positions, while the predominance of hold signals reflects that the strategy avoids unnecessary trades during periods of low market momentum.

Test

Buy signals	Sell signals	Hold signals	Total trades
393	267	458	660

As we can see, the CNN strategy on the testing set generated a total of 660 trades, consisting of 393 buy signals, 267 sell signals, and 458 hold signals. Compared to the training set, the total number of trades decreased, reflecting a more selective trading approach. The number of buy signals remains higher than sell signals, indicating that the strategy continues to favor entering long positions. Meanwhile, the predominance of hold signals shows that the strategy avoids unnecessary trades during periods of low market momentum, similar to the behavior observed in the training set.

Validation

Buy signals	Sell signals	Hold signals	Total trades
359	198	518	557

As we can see, the CNN strategy on the validation set generated a total of 557 trades, consisting of 359 buy signals, 198 sell signals, and 518 hold signals. Compared to the training and testing sets, the total number of trades slightly decreased, reflecting an even more conservative approach. The number of buy signals remains higher than sell signals, indicating a continued preference for long positions. The predominance of hold signals demonstrates that the strategy consistently avoids unnecessary trades during periods of low market momentum across all datasets.

Test and validation

Buy signals	Sell signals	Hold signals	Total trades
767	465	976	1232

As we can see, the CNN strategy on the combined test and validation set generated a total of 1,232 trades, consisting of 767 buy signals, 465 sell signals, and 976 hold signals. The total number of trades reflects the accumulated trading activity across both periods. Buy signals remain more frequent than sell signals, indicating that the strategy predominantly favors long positions. The predominance of hold signals shows that the strategy continues to avoid unnecessary trades during periods of low market momentum, ensuring trades are made only when significant opportunities arise.

Win rate

This statistic measures the number of profitable trades relative to the total number of trades. The higher the value, the more consistent the strategy. It is calculated as follows:

$$Win\ Rate = \frac{Positive\ trades}{Positive\ trades + Negative\ trades}$$

MLP

Train

Win Rate
52.04%

As we can see, the MLP strategy on the training set achieved a win rate of 52.04%. This indicates that slightly more than half of the trades were profitable, reflecting a modestly successful performance in capturing favorable market movements. While the win rate is above 50%, it suggests that the strategy still experiences a considerable number of losing trades, emphasizing the importance of risk management to preserve capital.

Test

Win Rate
46.15%

As we can see, the MLP strategy on the testing set achieved a win rate of 46.15%. This indicates that less than half of the trades were profitable during this period, reflecting a decline in performance compared to the training set. The lower win rate suggests that the strategy faced more challenging market conditions or that its predictive signals were less effective on unseen data, highlighting the importance of careful risk management.

Validation

Win Rate
41.67%

As we can see, the MLP strategy on the validation set achieved a win rate of 41.67%. This indicates that slightly less than half of the trades were profitable during this period, reflecting a further decline in performance compared to the training and testing sets. The lower win rate suggests that the strategy faced more challenging market conditions or that its predictive signals were less effective on unseen data, highlighting the importance of careful risk management.

Test and validation

Win Rate
44.00%

As we can see, the MLP strategy on the combined test and validation set achieved a win rate of 44.00%. Combining both periods provides a broader view of the model's performance across different market conditions, showing that while profitability is moderate, the strategy maintains a consistent behavior and stability over time.

CNN

Train

Win Rate
65.85%

As we can see, the CNN strategy on the training set achieved a win rate of 65.85%. This demonstrates a strong ability to capture favorable market movements and generate consistent gains under the training conditions. The relatively high win rate indicates that the model effectively identifies profitable opportunities.

Test

Win Rate
42.42%

As we can see, the CNN strategy on the test set achieved a win rate of 42.42%, indicating that less than half of the trades were profitable. This drop in performance compared to the training set suggests that the model may struggle to generalize effectively to unseen data. The lower win rate highlights the importance of improving the model's adaptability under different market conditions.

Validation

Win Rate
38.78%

As we can see, the CNN strategy on the validation set achieved a win rate of 38.78%, indicating that a smaller portion of trades were profitable compared to the training and testing sets. This decline in performance suggests that the model's predictive power weakens further on unseen data, possibly due to overfitting or changing market dynamics.

Test and validation

Win Rate
40.99%

As we can see, the CNN strategy on the combined test and validation set achieved a win rate of 40.99%. Combining both periods provides a broader view of the model's performance across different market conditions, showing that while profitability is moderate, the strategy maintains a consistent behavior and stability over time.

Cash and portfolio value

MLP

Train

Cash	Portfolio value
$ 1,478,967.86	$ 1,480,347.56

The MLP strategy on the training set ended with a cash balance of $1,478,967.86 and a total portfolio value of $1,480,347.56. This reflects stable growth during training, suggesting that the model effectively managed trades and preserved capital under known conditions.

Test

Cash	Portfolio value
$ 1,337,644.75	$ 1,338,238.49

On the testing set, the MLP strategy resulted in a cash balance of $1,337,644.75 and a total portfolio value of $1,338,238.49. The decrease compared to the training results indicates reduced profitability when applied to unseen data, which may reflect more volatile or less predictable market conditions.

Validation

Cash	Portfolio value

$ 1,061,551.01	$ 1,062,130.06

The validation set shows a cash balance of $1,061,551.01 and a total portfolio value of $1,062,130.06, representing a further decline in performance from both the training and testing sets. This suggests that the strategy's effectiveness weakens when facing new data, reinforcing the importance of model generalization.

Test and validation

Cash	Portfolio value
$ 1,426,592.95	$ 1,427,172.00

When combining the test and validation periods, the MLP strategy achieved a cash balance of $1,426,592.95 and a total portfolio value of $1,427,172.00, which is higher than either individual period but still below the training performance. These results provide a broader view of the model's performance across unseen data, indicating moderate profitability and consistent behavior across different market conditions.

CNN

Train

Cash	Portfolio value
$ 1,885,766.43	$ 1,886,481.83

The CNN strategy on the training set achieved a cash balance of $1,885,766.43 and a portfolio value of $1,886,481.83, reflecting strong profitability and effective capital utilization during the training phase. This indicates that the model successfully captured favorable price movements.

Test

Cash	Portfolio value
$ 1,251,607.65	$ 1,252,275.61

On the testing set, the CNN strategy produced a cash balance of $1,251,607.65 and a portfolio value of $1,252,275.61. The decrease compared to the training set highlights challenges in maintaining performance on unseen data, suggesting reduced adaptability under different market dynamics.

Validation

Cash	Portfolio value
$ 1,007,736.84	$ 1,008,070.91

The validation set shows a cash balance of $1,007,736.84 and a portfolio value of $1,008,070.91, indicating a further decline in profitability from both training and testing results. This reinforces the idea that the strategy may be overfitted to the training data or less effective under changing market conditions.

Test and validation

Cash	Portfolio value
$ 1,285,193.22	$ 1,285,527.29

For the combined test and validation periods, the CNN strategy achieved a cash balance of $1,285,193.22 and a total portfolio value of $1,285,527.29, which is higher than either individual period but still below the training performance. This reflects moderate profitability and consistent capital preservation, providing a more comprehensive view of the model's robustness across different market environments.

Model accuracy vs. strategy profitability analysis

MLP

The Multi-Layer Perceptron (MLP) models were initially evaluated based on validation accuracy recorded in MLFlow. Subsequently, the models were tested using the backtesting function to assess actual trading performance across the three datasets. Although higher validation accuracy generally indicated a stronger model, some MLP models with slightly lower accuracy or higher loss produced better profitability. This suggests that accuracy alone does not necessarily correspond to higher trading returns.

CNN

Convolutional Neural Network (CNN) models were evaluated in a similar manner. Initial assessments focused on models with the highest validation accuracy; however, backtesting indicated that models with lower accuracy occasionally outperformed those with top validation metrics in actual trading scenarios. This reinforces that real-world profitability depends on factors beyond validation performance alone.

Comparison: Does higher model accuracy means better trading returns?

Not every time did the models with the highest validation accuracy give the best results in practice. After the model metrics were collected in MLFlow, we tested several models using the backtesting function to see how they actually performed. Even though we started by testing the models that had the highest validation accuracy, it was sometimes found that other models, those with lower accuracy or higher loss, gave better results when applied to the three datasets. This shows that the model that looks best during training does not always perform best in real scenarios, and it is important to test multiple models to find the one that works most reliably.

Metric	MLP	CNN
Accuracy on Train Set	0.4089	0.4045
Accuracy on Test Set	0.4087	0.4502
Accuracy on Val Set	0.4083	0.4497
Sortino Ratio	0.742174	0.706986
Win Rate	44.00%	40.99%
Portfolio value in Test + Val	$1,427,172.00	$1,285,527.29

From the table, it is clear that higher model accuracy does not necessarily translate into better trading performance. While the CNN model achieved higher accuracy across the test and validation sets (0.4502 and 0.4497, respectively), its trading metrics, including Sortino ratio, win rate, and portfolio value, were inferior compared to the MLP model. Specifically, the MLP achieved a Sortino ratio of 0.742 and a portfolio value of $1,427,172, outperforming the CNN despite lower accuracy scores.

This contrast highlights that predictive accuracy alone cannot guarantee profitability or stable returns in trading strategies.

Conclusions

Key findings and strategy viability

The implementation of deep learning architectures for trading signal prediction demonstrated that both Multi-Layer Perceptron (MLP) and Convolutional Neural Network (CNN) models are capable of identifying profitable patterns in Lululemon's historical data, showing consistent portfolio growth under varying market conditions. Across training, testing, and validation periods, both strategies showed consistent behavior and effective capital preservation. The MLP strategy produced stable results with moderate profitability and controlled drawdowns, while the CNN model exhibited higher profitability during training and testing, though with weaker performance on unseen data due to higher sensitivity to market regime changes. These results demonstrate that both deep learning models were capable of learning meaningful patterns from momentum, volatility, and volume engineered features and generating consistent trading signals over extended historical periods.

Across all datasets, the Sharpe and Sortino ratios indicated that both strategies generated positive risk-adjusted returns in training and testing, though performance weakened under validation conditions, reflecting reduced adaptability to new market regimes. The data drift analysis confirmed significant distributional changes in key indicators, particularly in the ones related to market volatility and liquidity, which likely reduced predictive accuracy over time. Despite these limitations, the overall evidence supports the viability of deep learning for trading applications, provided that continuous monitoring and retraining mechanisms are in place.

Model selection and performance summary

The CNN model achieved superior performance during training and testing, attaining higher Sharpe and Sortino ratios with lower drawdowns compared to the MLP, its strong drop in both validation profitability and risk-adjusted metrics indicated overfitting and lower resilience to distributional shifts. However, during the validation phase, the MLP exhibited greater stability and robustness, maintaining more consistent risk-adjusted returns despite changing market conditions. Therefore, while the CNN provided stronger profitability in historical contexts, the MLP model demonstrated superior generalization capabilities, making it more suitable for deployment in dynamic trading environments.

After comparing both architectures, the MLP model emerged as the most balanced and reliable approach. It achieved the highest overall cash on combined test and validation data ($1,426,592.95) and a stronger Sortino ratio (0.74), outperforming the CNN, which ended with $1,285,193.22 and a Sortino ratio of 0.71. The MLP also showed a slightly higher win rate of 44%, compared to 41% for the CNN, confirming its more robust behavior across different datasets.

Whether strategy is profitable after transaction costs

After accounting for transaction costs, including a 0.125% commission per trade and an annual borrow rate of 0.25% for short positions, both strategies remained profitable during training and testing phases, though net returns were reduced. The MLP ended the combined test and validation period with a portfolio value of $1,427,172, while the CNN achieved $1,285,527.29.

This confirms that despite transaction costs, both models generated positive cumulative returns and maintained capital growth. However, the profit margins decreased significantly during the validation period, highlighting that future implementations must include dynamic cost adjustment and improved risk management to sustain profitability under real market conditions.

Recommended improvements or extensions

To enhance strategy robustness and profitability, several extensions are recommended:

- Combine predictions from MLP and CNN architectures to capture both temporal and nonlinear dependencies, improving stability and signal confidence. Recurrent models such as LSTM or Transformers could also be included.
- Incorporate macroeconomic variables, sentiment indicators, and alternative data sources to improve predictive coverage.
- Replace fixed trade sizes with volatility-adjusted position sizing and dynamic stop levels to optimize exposure, improve capital efficiency, and adapt to varying market conditions.

In conclusion, the deep learning-based trading framework developed in this project successfully demonstrated predictive capacity and profitability under realistic assumptions. With further enhancements in adaptability, data diversity, and risk control, the strategy has strong potential for real-world application in systematic trading systems.

References

Documentation — Technical Analysis Library in Python 0.1.4 documentation. (n.d.). https://technical-analysis-library-in-python.readthedocs.io/en/latest/ta.html#momentum-indicators

Documentation — Technical Analysis Library in Python 0.1.4 documentation. (n.d.). https://technical-analysis-library-in-python.readthedocs.io/en/latest/ta.html#volume-indicators

Documentation — Technical Analysis Library in Python 0.1.4 documentation. (n.d.). https://technical-analysis-library-in-python.readthedocs.io/en/latest/ta.html#volatility-indicators

ks_2samp — SciPy v1.16.2 Manual. (n.d.). https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ks_2samp.html

Segal, T. (2025, April 14). *What is a momentum indicator? Definition and common indicators*. Investopedia. https://www.investopedia.com/investing/momentum-and-relative-strength-index/

Finserv, B. (2025, July 5). *Volatility indicators*. www.bajajfinserv.in. https://www.bajajfinserv.in/volatility-indicator

Brown, B. (2025, March 31). *How to use stock volume to improve your trading*. Investopedia. https://www.investopedia.com/articles/technical/02/010702.asp

Ushman, D. (2023, December 22). *What is the Kaufman Adaptive Moving Average?* TrendSpider. https://trendspider.com/learning-center/what-is-the-kaufman-adaptive-moving-average/

Babypips.com. (2025, May 30). *How to use Williams %R (Williams percent range)*. https://www.babypips.com/learn/forex/williams-r

Thompson, C. (2025, September 3). *Understanding Bollinger Bands: a key technical analysis tool for investors*. Investopedia. https://www.investopedia.com/terms/b/bollingerbands.asp

Team, C. (2024, September 20). *Keltner Channel*. Corporate Finance Institute. https://corporatefinanceinstitute.com/resources/career-map/sell-side/capital-markets/keltner-channel/

Chen, J. (2025, August 21). *Understanding Donchian channels: formula, calculation, and trading uses*. Investopedia. https://www.investopedia.com/terms/d/donchianchannels.asp

Hayes, A. (2025, August 10). *On-Balance Volume (OBV): how it works and how to use it*. Investopedia. https://www.investopedia.com/terms/o/onbalancevolume.asp
Team, C. (2024, June 4). *Accumulation/Distribution Indicator (A/D)*. Corporate Finance Institute. https://corporatefinanceinstitute.com/resources/equities/accumulation-distribution-indicator-a-d/

Chen, J. (2025, October 21). *Understanding the Chaikin Oscillator: Definition, formula, and trading insights*. Investopedia. https://www.investopedia.com/terms/c/chaikinoscillator.asp

Ease of movement (EOM). (n.d.). TradingView. https://www.tradingview.com/support/solutions/43000502256-ease-of-movement-eom/

Palmer, B. (2023, December 26). *How to use the Force Index*. Investopedia. https://www.investopedia.com/articles/trading/03/031203.asp

Uniqtech. (2018, December 22). *Multilayer Perceptron (MLP) vs Convolutional Neural Network in Deep Learning*. Data Science Bootcamp. https://medium.com/data-science-bootcamp/multilayer-perceptron-mlp-vs-convolutional-neural-network-in-deep-learning-c890f487a8f1

Kundu, R. (n.d.). F1 Score in Machine Learning: Intro & Calculation. *V7*. https://www.v7labs.com/blog/f1-score-guide