Machine Learning in Finance

FIN-407

Project

Prof. MALAMUD Semyon
TA. MATERA Giuseppe

Candice BUSSON	394250
Timothé DARD	340944
Marine DE ROCQUIGNY	327489
Christopher SORIANO	326354
Cyprien TORDO	345618

June 11^{th}, 2025

Contents

1 Introduction

This project investigates the potential for predicting the financial returns of stocks within the agro-industrial sector using machine learning techniques. The agro industry is influenced by a wide range of external and internal factors, and our goal is to capture this complexity through a multi-faceted modeling approach.
We focus on three categories of predictive variables:

- **Meteorological data**, which reflect environmental conditions that directly affect agricultural output;
- **Sentiment analysis derived from industry-related news**, capturing the market's perception and expectations;
- **Firm-level characteristics turned into tradable long–short portfolios**, representing cross-sectional financial signals that capture a range of economic themes such as value, momentum, profitability, and investment behavior.

By integrating these three dimensions, we aim to build predictive models that are not only data-rich but also context-aware.

2 Collecting and formatting data

Each of the three data channels (meteorological measures, industry-related news and economic long-short portfolios) required a specific approach to data sourcing, preprocessing, and formatting. In the following subsections, we detail the specific methods and tools used to collect and prepare each dataset for use in our predictive models.

2.1 Returns dataset

From the monthly CRSP stock return dataset provided, we selected stocks relevant to our analysis using their SIC codes, a standardized industry classification system. Specifically, we included stocks from the agriculture sector (SIC codes 0100–0999), grocery stores (5411–5412), miscellaneous food stores (5499–5500), and eating places (5812–5813). Only a few data points were missing, which we manually filled to maximize data reliability. For consistency with the rest of the analysis, we restricted our dataset to the period from January 1, 2000 to December 31, 2024. In the end, the dataset contains the returns of 248 stocks in the agro-industrial sector.

2.2 Predictors datasets

2.2.1 Meteorological data

Meteorological data were collected from the National Oceanic and Atmospheric Administration (NOAA) through its National Centers for Environmental Information portal (https://www.ncei.noaa.gov/access/search/data-search/daily-summaries). The dataset contains a wide range of daily environmental variables, with over 100 meteorological indicators available. We focused our collection on seven US states: Illinois, Kansas, Nebraska, Minnesota, South Dakota, Iowa, and Indiana, as they consistently rank among the top producers of major crops and account for an estimated $\sim 70\%$ of US agricultural output.
A key challenge was determining which meteorological variables to retain. We had to balance predictive usefulness (guided by online resources and personal judgement), data completeness (for example, dew point temperature was dropped due to too many missing values), and computational feasibility. In the end, we selected the following six variables: `TMIN` and `TMAX` (daily minimum and maximum temperature), `PRCP` (total daily precipitation), `AWND` (average wind speed), `SNOW` (daily snowfall), and `SNWD` (snow depth on ground). These variables were collected for each of the seven selected states, resulting in a total of $6 \times 7 = 42$ meteorological predictors.
The raw NOAA data spans from around 1940 to start of 2025. For relevance and consistency with the financial and sentiment data used in our models (more on that later), we restricted our analysis to the period January 1, 2000, to December 31, 2024. We then evaluated missing data. The vast majority of columns had (almost) no missing values (see Figure 1). The only exceptions were the two snow-related measures in Minnesota, containing 10 to 15% of missing values.
Since our modeling approach uses monthly data, the daily meteorological values were aggregated by taking

the mean for each month, which appeared to be the most meaningful approach for weather-related variables and reasonable given the very low proportion of missing values. As a result, the remaining monthly data had no missing values.
This cleaned and structured dataset forms the meteorological component of our model, offering insight into how weather and climate conditions may influence the performance of agro-industry companies.

2.2.2 Industry-related news

We focused on news whose subject rotates around agri-food. To do so we have collected news from several datasets and then have combined them together. During this phase, we encountered one major issue which is the amount of open source news related to agriculture. As finding news dataset is already difficult, adding the constraint of an industry proves to be a supplementary strain. Therefore, we chose to work with several news datasets and filter the articles based on a pre-defined vocabulary set (see Appendix). We successfully filtered and used the following datasets:

1. AG News [1] is a dataset of 17'484 news' articles gathered from other 2000 sources which has been classified by topic. It contains news from late 2021 to mid 2020. Each individual sample are arranged in folder corresponding to the date of publication. The creation of a pandas DataFrame regrouping all of these individual folder has been done in `AGNEWS.ipynb`.

2. AG News [2] is another collection of more than 1 million news articles gathered from more than 2'000 sources, from which we successfully retrieved 13'824 articles related to our sector, from August 2004 to February 2008.

3. The News Category Dataset [3] contains around 210k news headlines from 2012 to 2022 from *HuffPost*. We successfully retrieved 1'537 agriculture-related articles for our work.

4. Daily Financial News [4] contains around 1,5 million news' articles related to over 2'000 different stocks, scraped from Benzinga. From this dataset, 11'982 articles were recognized as agriculture-related from 2010 to 2020.

5. Bloomberg News [5] contains around 500'000 news articles from *Bloomberg*, from which we successfully filtered around 51'000 articles from 2010 to 2014.

6. All the News Dataset [6] is probably the most complete dataset we used as it includes 2.7 million news articles from 27 American publications, spanning from 2016 to early 2020. From this dataset, we successfully filtered around 200'000 agriculture-related articles.

7. Frontpage News [7] is the second most important dataset we used, encompassing around 13 million english news articles from around 90 news outlets and spanning from 2016 to 2020. From this dataset, we successfully filtered 100'000 agriculture-related articles.

Additional datasets from *Hugging Face* were used and cited in our references. However, these datasets were either much smaller or included way less data of interest for our project. As we can observe in Figure 2, the number of rows per day is extremely varying depending on the period. Due to the difficulty of finding new data points, we made the choice of still conducting the analysis with or data, even though our results might be quite biased.

2.2.3 Economic data

To predict stock returns, we intended to use financial and economic predictors, which we considered highly relevant. We initially went for quarterly firm characteristics from Compustat. However, the dataset did not provide sufficient coverage for the agricultural firms in our sample. As a result, we turned to the Jensen, Kelly, and Pedersen (JKP) factors, which offer a broad set of predictors.
The JKP dataset provided includes 153 firm-level factors transformed into tradable long–short portfolios, such as firm age, asset growth, debt-to-market ratio, etc, available for U.S. firms with no missing values. Further details on these factors can be found at https://github.com/bkelly-lab/ReplicationCrisis/raw/master/GlobalFactors/Factor%20Details.xlsx. After removing columns with too much outliers, the dataset contained the returns of 130 long-short portfolios from January 1, 2000 to December 31, 2024.

3 Models

3.1 Labeling news for SA

As our news dataset is not labeled we have performed sentiment analysis ourselves over the full dataset. The code used can be found under `Sentiment_analysis.ipynb`.
For the labeling of the news dataset we have used BERT model [9] called *bert-base-multilingual-uncased-sentiment* [10], that is fine-tuned for sentiment-analysis. It predicts the sentiment of the text with a rating by a number of stars: 1 star means the text is negative and 5 stars mean the text is positive.
We have decided to use this model as it is a small model with high accuracy in english that has already been fine-tuned on rating text.
The Bert-base model is made of a transformer stack containing 12 layers, 768-hidden layers and 12 attention heads. On top we use a classification head made of a single layer and a bias that is tuned to output the 5 logits (used to get the 5 stars). The logits are then passed through a softmax function to turn the vector into a probability distribution.
Since there are many days where there are no news in our dataset, we have filled these days by a 3, corresponding to a neutral rating.

3.2 Exploration with Ridge Regression

To initially assess the influence of our features on stock returns, we applied Ridge Regression to identify potential relationships between the input variables (our features) and the target labels (stock returns). After standardizing the data, we tuned the regularization parameter (alpha) to optimize our model's performance based on both the R^2 score and the Mean Squared Error (MSE). The best-performing model was achieved with an alpha value of 0.0001, resulting in an MSE of 0.835 and an R^2 value of 0.165.
An analysis of the model's coefficients revealed substantial variability in feature importance. Coefficients ranged from 0.667 for `TMIN_IOWA` to as low as 0.000003 for `Change in short-term investments`. The sentiment-based feature had a moderate coefficient of 0.0204. Notably, all feature categories (meteorological, sentiment-based, and firm-level) exhibited high internal variability in their impact. Contrary to our earlier observations (prior to standardization), meteorological and sentiment-based features did not consistently receive the smallest coefficients.
To distinguish between important and less relevant features, we applied a threshold of 0.05 (in absolute value) to the standardized coefficients. Features with coefficients above or equal to this threshold were considered important and those below were considered less influential.
To further investigate the potential of these features in capturing nonlinear patterns, we proceeded to implement a deep learning model, as discussed in the following section.

3.3 Modelling and Prediction Pipeline

3.3.1 Model Architecture and Literature Review: The Temporal Fusion Transformer

For the task of predicting stock returns, we selected the Temporal Fusion Transformer (TFT), a state-of-the-art deep learning architecture. Our choice is grounded in a review of recent literature, which demonstrates the model's superior performance and suitability for complex financial forecasting tasks.
The TFT was introduced by Lim, Arık, Loeff, and Pfister (2021) as a novel attention-based architecture designed for high-performance, interpretable multi-horizon time series forecasting [11]. Their foundational work highlighted several key innovations that are directly applicable to our project:

- **Handles Diverse Feature Types**: The model is explicitly designed to ingest a complex mix of inputs, including static covariates (e.g., a company's sector), time-varying known inputs (e.g., day of the week), and time-varying observed inputs (e.g., past returns, meteorological data). This is an advantage for financial modeling, where returns are influenced by a heterogeneous set of variables.
- **Gated-Residual Architecture**: TFT uses specialized gating mechanisms (Gated Residual Networks) to allow the model to learn how much of a feature to use and to skip over unused components of the architecture, providing adaptive depth and complexity for different datasets.

- **Built-in Interpretability**: Unlike many "black-box" deep learning models, the TFT includes variable selection networks and an interpretable multi-head attention block. This allows for the identification of the most significant features and important past time-steps driving the predictions, a crucial requirement for validating financial strategies.

Subsequent research has validated the TFT's effectiveness in the financial domain. Hu et al. (2021) directly compared the TFT against Support Vector Regression (SVR) and Long Short-Term Memory (LSTM) models for stock price prediction, finding that the TFT achieved the lowest prediction errors [12]. Frank (2023) demonstrated that the TFT produced "very good results" in forecasting the realized volatility of S&P 500 stocks, outperforming LSTMs and Random Forests in turbulent market periods [13]. Furthermore, recent studies have shown the model's power in broader economic forecasting, such as predicting the GDP of 25 OECD countries, where it outperformed traditional regression models, especially during periods of high volatility like the COVID-19 shock [14].
Based on this strong evidence from the literature (its architecture, proven performance on volatile financial and economic data, and unique interpretability features) we selected the TFT as the most promising model for this project. The implementation of our model training and validation logic is contained in `TFT.py`.

3.3.2 Data Structuring for Time-Series Forecasting

To feed our data into the TFT model, we first needed to structure it into a format suitable for time-series forecasting. This step was handled by the script `create_dataloader.py`, which uses the `pytorch-forecasting` library.
The process is as follows:

1. **TimeSeriesDataSet**: We created a `TimeSeriesDataSet` object, which is a specialized data structure that understands the temporal nature of our data.

2. **Encoder-Decoder Structure**: The model is trained to predict future returns based on historical data. We defined an encoder length of 12 months (the model looks at the past year of data) and a prediction length of 3 months (the model predicts returns for the next quarter).

   To generate a large number of training examples from our historical data, the `TimeSeriesDataSet` object effectively creates samples using a rolling window approach. For each company, it slides this 15-month window (12 for input, 3 for target) across its entire timeline, generating many overlapping training instances. This allows the model to learn from a rich variety of historical periods.

3. **Feature Definition**: We defined the roles of our various data columns:

   - `target`: The variable to be predicted, which is `Stock_return`.
   - `group_ids`: The identifier for each individual time series, which is `CUSIP` (representing each company).
   - `time_idx`: The integer representing the time step (month).
   - `time_varying_unknown_reals`: This category includes our primary features. These are all variables whose future values are not known at prediction time, such as company-level economic indicators, meteorological data (e.g., `TMAX_ILLINOIS`, `PRCP_KANSAS`), and the sentiment score.

4. **Data Loaders**: The script creates `train_dataloader` and `val_dataloader` objects, which efficiently load batches of data for training and validation, respectively.

The final configured `TimeSeriesDataSet` object is saved to `training_dataset.pth`, ensuring that the exact data processing pipeline (including normalization and feature encoding) used for training is later available for prediction.

3.3.3 Hyperparameter Optimization with Optuna

The performance of a TFT is highly sensitive to its hyperparameters. To find the optimal configuration, we implemented an automated hyperparameter optimization process using the `Optuna` framework in

hyperpar_opt.py.
The objective was to find the set of parameters that minimized the Mean Absolute Error (MAE) on the validation set. The script explored the following key hyperparameters:

- learning_rate: The step size for the AdamW optimizer.
- hidden_size: The number of neurons in the hidden layers of the network.
- attention_head_size: The number of attention heads in the self-attention layer.
- dropout: The rate of regularization to prevent overfitting.

The optimization process runs for a predefined number of trials. In each trial, Optuna suggests a new combination of hyperparameters, a model is trained for 5 epochs, and its validation loss is reported. The MedianPruner is used to automatically stop unpromising trials early, making the search process highly efficient. The best parameter set found during the study was saved to best_hyperparameters.json for use in final model training.

3.3.4 Final Model Training and Prediction

1. **Training**: The train_tft_model function in TFT.py takes the best hyperparameters from the optimization step and trains the final TFT model on the entire training dataset for 200 epochs. It logs the training and validation loss (MAE) at each epoch to monitor performance and saved the final trained model weights to weights_model.pth.
2. **Prediction**: The predict_returns.py script handles the inference stage. It loads the saved model weights (weights_model.pth) and the dataset configuration (training_dataset.pth). This script performs predictions on a dedicated holdout dataset that was separated from the training data, as defined in main.py. This holdout set consists of the final 15 months of data for each company, ensuring the model's performance is evaluated on completely unseen data. For each stock (CUSIP), the script uses the first 12 months of this holdout data as an input to the model's encoder and generates predictions for the subsequent 3 months. The script then saves these predictions alongside the actual returns into a predictions.csv file for subsequent analysis.

4 Backtesting Investment Strategies

The ultimate goal of predicting stock returns is to inform investment decisions. To evaluate the practical value of our model's predictions, we implemented a backtesting framework in strategies.py. This script uses the predictions.csv file to simulate the performance of several different portfolio allocation strategies, including long-short methodologies.
The historical covariance matrix (Σ) is calculated from the full historical returns data. The backtest iterates through each of the 3 prediction steps, rebalancing the portfolio at each step based on the model's new predictions.

4.1 Portfolio Strategies Implemented

The following strategies were implemented to test the effectiveness of the model's predictions:

1. **Long-Short Equal Weight:** A dollar-neutral strategy that longs assets with predicted returns above 0.001 and shorts assets with predictions below -0.001, with equal weighting in each leg.
2. **Mean-Variance Optimization (MVO):** A portfolio that uses the model's predictions (μ) and historical covariance (Σ) to find the tangency portfolio that maximizes the Sharpe ratio. Weights are determined analytically, allowing for short-selling:
$$w = \frac{\Sigma^{-1}(\mu - r_f)}{\mathbf{1}^\top \Sigma^{-1}(\mu - r_f)}$$
3. **Equally Weighted Benchmark:** A naive benchmark holding an equally weighted portfolio of all assets, regardless of model predictions.

4.2 Backtesting Results

The `run_backtest` function tests our strategies month by month. For each month, it calculates the return that each portfolio (Long-Short, MVO, and Benchmark) would have made. After checking all the months, it calculates the total performance. It adds up the returns for the Long-Short strategy and compounds the returns for the Benchmark strategy. Finally, it calculates metrics like R^2 and the Sharpe Ratio and creates a plot to show the results.

5 Results

5.1 Note on the Benchmark Portfolio

The Benchmark portfolio manages very high returns but that is expected as over 20 years the american market has seen a powerful growth.

5.2 Performance with All Features

We first ran our model using all available features, including meteorological data and industry-related news.

Predictive Accuracy (R^2) The model's predictive power was highly variable and, on the whole, quite poor. While it achieved a positive R^2 for a few specific stocks, the mean R^2 was -112.15 (see Table 5), indicating that for most assets, the model's predictions were less accurate than a simple historical average. The predictions from this model were likely noisy and lacked strong, clear signals.

Portfolio Performance Given the weak predictions, the resulting portfolio performance was underwhelming. As shown in Table 1, both the Long-Short (LS) and Mean-Variance Optimization (MVO) strategies underperformed the benchmark significantly.

5.3 Performance with Restricted Features

Following the Ridge Regression analysis, we trained a second model using a reduced set of more impactful features.

Predictive Accuracy (R^2) Removing the noisy features led to a superior model. Although the mean R^2 was still negative, it improved from -112 to -25.8 (see Table 6), and a higher number of stocks achieved a positive R^2 (see Table 4).The model was able to identify certain opportunities with much higher conviction, achieving an R^2 of up to 0.719 for one stock.

Portfolio Performance The higher-quality predictions from this model had a huge impact. The Mean-Variance Optimization strategy, now acting as an amplifier for these clearer signals, produced an explosive cumulative return of 2,443% (see Table 2). This seemingly paradoxical result, an enormous return paired with a low Sharpe Ratio of just 0.1688, is a sign of a strategy driven by extreme outlier returns. The MVO strategy, with a bit more precise predictions from the better model, must have placed a massive, concentrated bet on an asset that experienced an extraordinary return. This single event was enough to cause the multiplicative cumulative return to skyrocket. However, this same event created immense volatility, which is heavily penalized by the Sharpe Ratio calculation. This tells us that while the strategy had immense potential, its success was not steady but was instead characterized by extreme risk and a dependency on correctly predicting rare events.

5.4 Model Training and Overfitting

Finally, the loss curves for both models provide critical context for these results (see Figure 3 and Figure 4). In both training runs, the training loss steadily decreases while the validation loss remains volatile and significantly higher. This divergence is a clear indication of **overfitting**, which occurred despite the use of dropout, a regularization technique implemented specifically to improve the model's generalization. This

suggests that while dropout helped, it was not sufficient to prevent the model from learning the training data's noise. This underlying issue helps explain why the overall predictive power remained limited and why its success was tied to specific, high-conviction predictions rather than broad, consistent accuracy.

6 Conclusion

This project tried to use an advanced deep learning model, the Temporal Fusion Transformer, to predict stock returns for companies in the agro-food industry. We used different kinds of data—like economic factors, weather, and news sentiment—to try and understand what affects the performance of these companies.

Our results were mixed. The first model we built, which used all the features, did not perform well. This shows that for financial machine learning, just adding more data doesn't always help. After we used Ridge regression to pick a smaller, better set of features, the model's ability to predict returns for certain stocks got much better. This showed how important it is to choose the right features.

The backtesting results were very clear. The improved model, when used with a Mean-Variance Optimization strategy, gave a very high total return of over 2,400%. This wasn't because the model was always right, but because it was very good at finding and betting on a few rare events that had a big impact. The low Sharpe Ratio showed what was really happening: the returns were not steady, and the strategy was very risky and depended on correctly guessing these unusual market movements.

In the end, the model's biggest problem was that it overfit the training data. Even though we used dropout, a technique to help the model work better on new data, the loss curves showed it struggled to generalize. This is why it was successful only sometimes on very specific predictions, and not consistently accurate.

For future work, we suggest three things. First, we need better ways to fight overfitting, like using stronger regularization or a larger set of data. Second, adding better features, especially a good news sentiment dataset that covers a long period of time, could lead to more steady predictions. Lastly, the investment strategies could be changed to include risk limits to stop the MVO strategy from making huge, focused bets.

To sum up, while the idea of a perfect "money-making machine" is still out of reach, this project shows that advanced machine learning models can still be useful in financial predictions, even if they are risky. Their real strength isn't in making steady money, but in finding rare chances that simpler models can't see.

7 References

[1] Zhang, X., Zhao, J. J., & LeCun, Y. (2015). Character-level Convolutional Networks for Text Classification. *Advances in Neural Information Processing Systems (NeurIPS)*. https://papers.nips.cc/paper_files/paper/2015/file/2f2b265625d76a6704dc7d27bcd7ee65-Paper.pdf

[2] Gulli, A. &, A Signorini, A. http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html
Accessed: 2025-06-09

[3] Misra & Rishabh. (2022). News Category Dataset. *Sculpting Data for ML: The first act of Machine Learning*. https://arxiv.org/abs/2209.11429
Accessed: 2025-06-09

[4] Ashraq. *Financial News Dataset*. https://huggingface.co/datasets/ashraq/financial-news
Accessed: 2025-06-09

[5] Gurevich & Vladimir. (2022). *Weakly Labelled Large English NER corpus*. https://huggingface.co/datasets/imvladikon/english_news_weak_ner
Accessed: 2025-06-09

[6] Thompson, A. (2022). https://huggingface.co/datasets/rjac/all-the-news-2-1-Component-one
Accessed: 2025-06-09

[7] AndyReas. https://huggingface.co/datasets/AndyReas/frontpage-news
Accessed: 2025-06-09

[8] Myradeng. https://huggingface.co/datasets/myradeng/cs-230-news-v3 & https://huggingface.co/datasets/myradeng/cs-230-news-v2
Accessed: 2025-06-09

[9] Devlin, J., Chang, M-W., Lee, K. & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://arxiv.org/abs/1810.04805

[10] Google Research. Multilingual BERT — GitHub Documentation. https://github.com/google-research/bert/blob/master/multilingual.md

[11] Lim, B., Arık, S. Ö., Loeff, N., & Pfister, T. (2021). Temporal Fusion Transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4), 1748-1764. https://doi.org/10.1016/j.ijforecast.2021.03.012

[12] Hu, X., Wu, T., & Liu, B. (2021). Stock Price Prediction Based on Temporal Fusion Transformer. *2021 International Conference on Machine Learning and Big Data Analytics (MLBDA)*. https://doi.org/10.1109/MLBDBI54094.2021.00019

[13] Frank, J. (2023). Forecasting realized volatility in turbulent times using temporal fusion transformers. *FAU Discussion Papers in Economics*. https://www.econstor.eu/bitstream/10419/268951/1/1837499578.pdf

[14] Laborda, J., Ruano, S., & Zamanillo, I. (2023). Multi-Country and Multi-Horizon GDP Forecasting Using Temporal Fusion Transformers. *Mathematics*, 11(12), 2625. https://doi.org/10.3390/math11122625

8 Appendix

8.1 Agriculture Keywords

General Agriculture Terms
agriculture
agronomy
agribusiness
farming
cultivation
horticulture
irrigation
harvesting
tillage
soil
sustainable agriculture
organic farming
monoculture
crop rotation

Inputs & Equipment
seed
compost
manure
fertilizer
pesticide
machinery
GPS farming

Crops
crop
corn
maize
rice
wheat
soybean
oats
cotton
sugarcane
tobacco
legumes
vegetables
fruits

Agriculture Markets & Policy
commodity
commodity prices
farm subsidy
rural economy
agricultural exports
biofuel
grain reserves

Politics & Geopolitics
armed conflict
political instability
economic sanctions
trade blockade
embargo
trade war
diplomatic relations
geopolitical crisis
government collapse
political transition

War & Security
war zone
military occupation
supply chain disruption
targeted attacks
invasion
rural exodus
forced displacement
landmines
collateral damage
war crimes

Agricultural Impact & Food Systems
food crisis
food security
agri-food system
crop yields
food prices
supply disruption
import dependency
damaged farmland
fertilizer shortage
labor shortage
commodity shock
agricultural subsidy

Institutions & Policy Concepts
food and agriculture organization
World Bank
World Food Program
system resilience
land reforms
agricultural policy

8.2 Tables and Figures

Portfolio	Cumulative Return	Sharpe Ratio
LS_Return	1.0048	0.0535
MVO_Return	0.7715	0.0844
Benchmark_Return	3.3648	0.1538

Table 1: Cumulative portfolio performance and Sharpe ratios at final step with model trained on all features

Portfolio	Cumulative Return	Sharpe Ratio
LS_Return	0.1167	-0.0033
MVO_Return	24.4313	0.1688
Benchmark_Return	3.3648	0.1538

Table 2: Cumulative portfolio performance and Sharpe ratios at final step with model trained on only important features

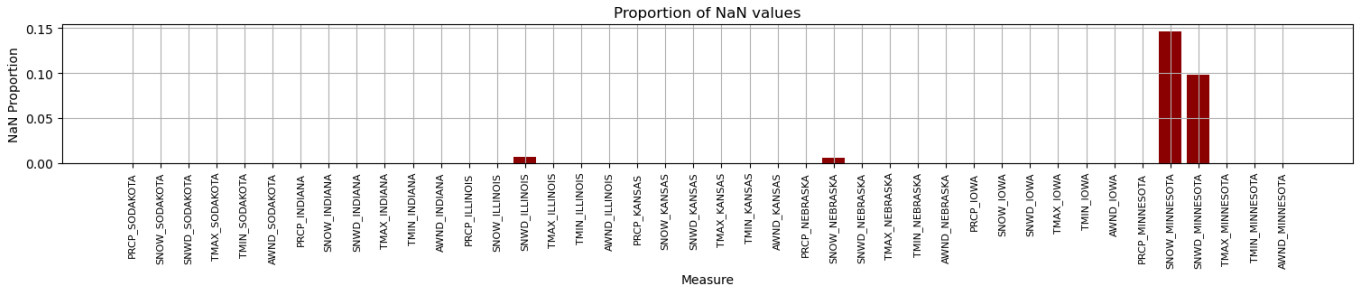

Figure 1: Proportion of missing values in each column before filling

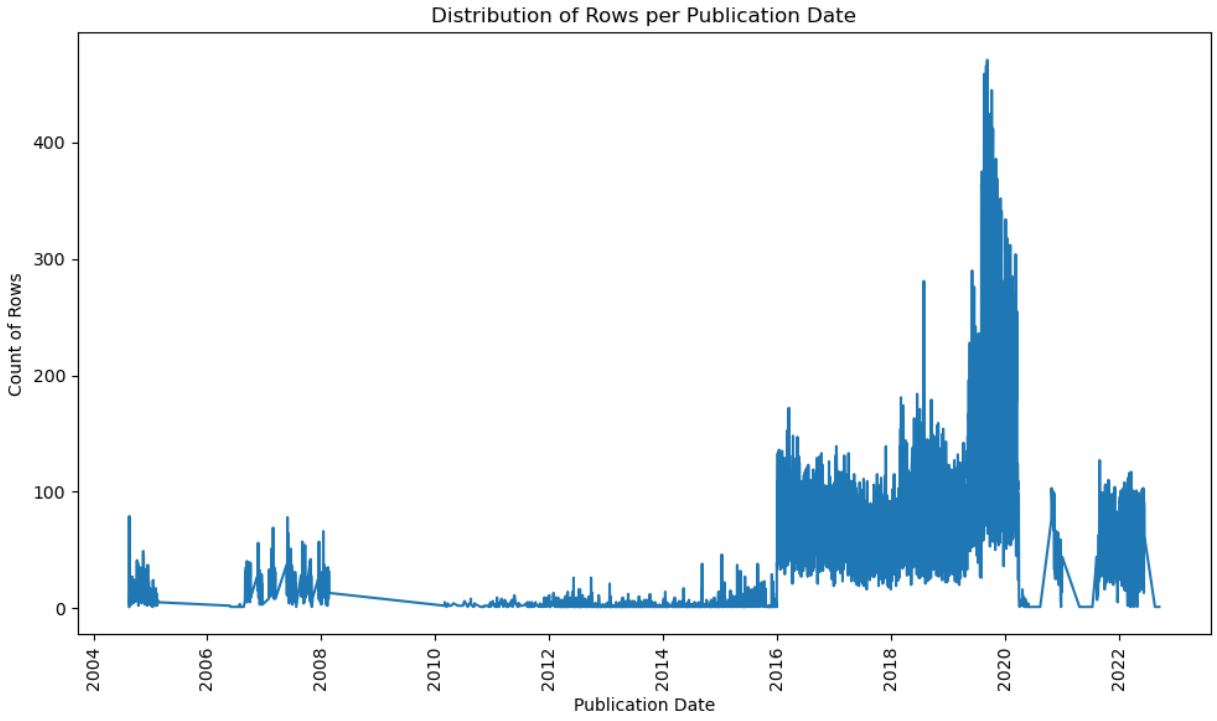

Figure 2: Distribution of news' articles per date

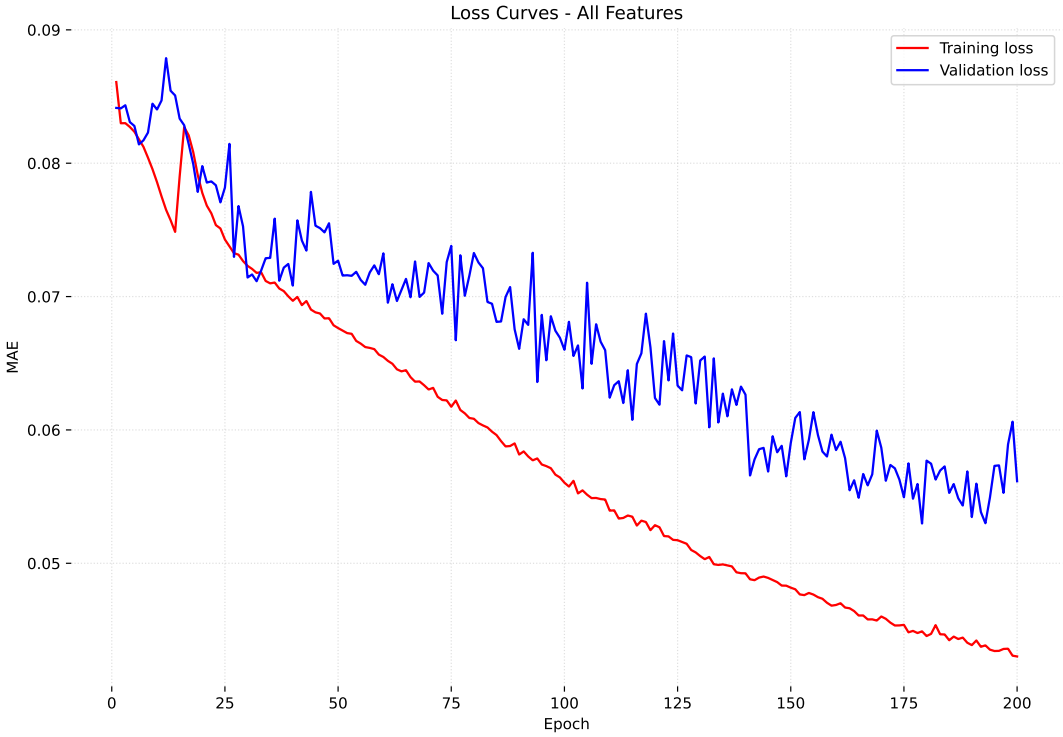

Figure 3: Loss curves for model trained on all features

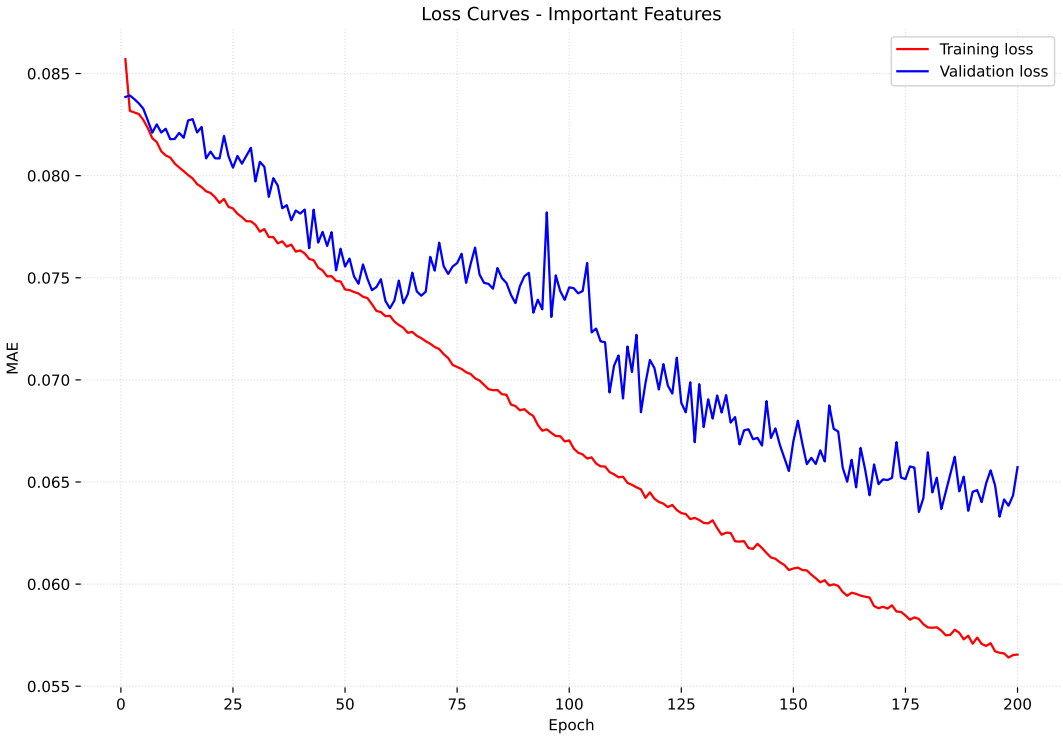

Figure 4: Loss curves for model trained on only important features

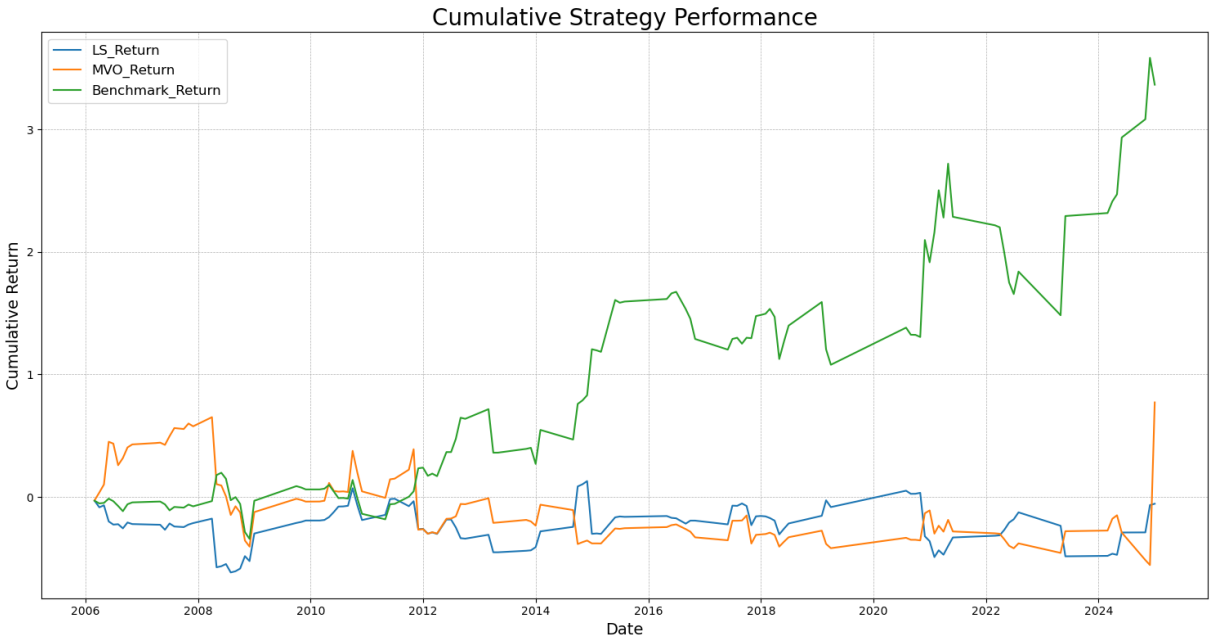

Figure 5: Cumulative performance of strategies with model trained on all features

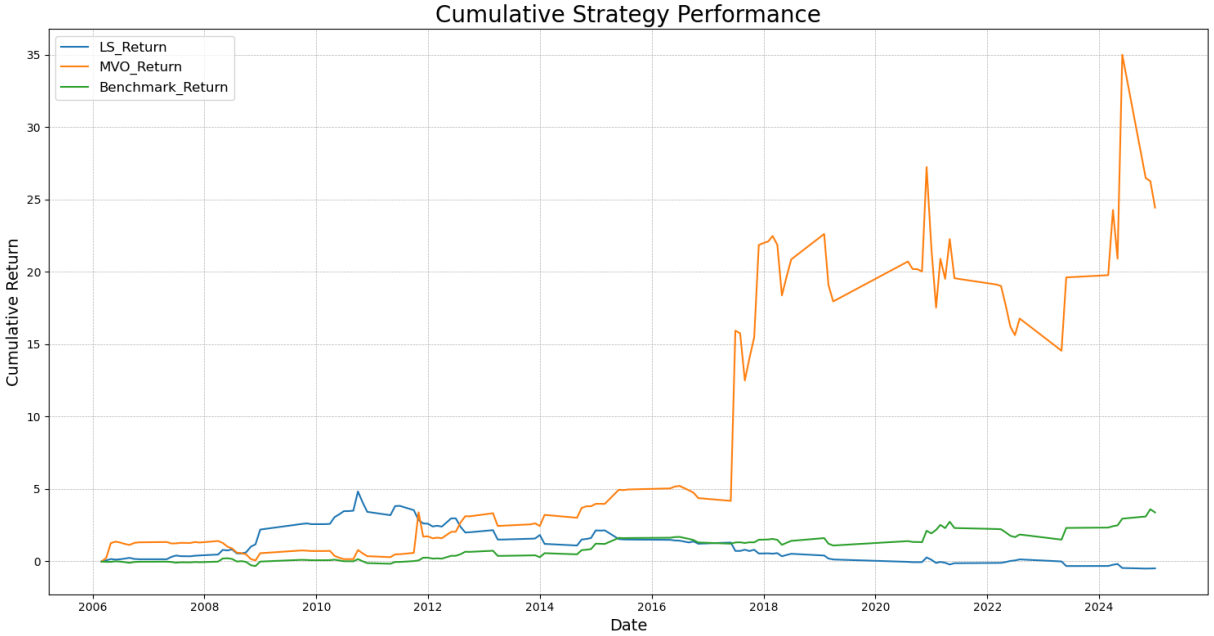

Figure 6: Cumulative performance of strategies with model trained on only important features

CUSIP	R^2
13054D10	0.6267
25475310	0.3456
25754A20	0.4677
28257U10	0.0254
39006410	0.4474
50047H20	0.1691
50104410	0.5372
55307U10	0.5431
63888U10	0.2464
76131D10	0.4409
88706M10	0.6205
...	...

Table 3: R^2 values by CUSIP for the model trained on all features

CUSIP	R^2
09605510	0.7190
13054D10	0.1906
25475310	0.3657
25754A20	0.3325
28257U10	0.0200
40011010	0.0155
46636710	0.0140
55307U10	0.0769
75419810	0.6043
75689M10	0.4599
78118210	0.1083
83175810	0.2902
85208M10	0.0605
87908010	0.0586
88268110	0.6999
95058W10	0.4550
...	...

Table 4: R^2 values by CUSIP for the model trained on only important features

Statistic	**Value**
Mean	-112.1487
Std Dev	628.7993
Min	-5501.7435
Q25	-4.0716
Median	-1.1637
Q75	-0.3122
Max	0.6267

Table 5: R^2 statistics of stocks for model trained on all features

Statistic	**Value**
Mean	-25.7847
Std Dev	133.9835
Min	-1229.1614
Q25	-3.3429
Median	-0.8137
Q75	-0.1277
Max	0.7190

Table 6: R^2 statistics of stocks for model trained on only important features