# Project Work 2

Elias Eskelinen, Jarkko Komulainen, Matti Aalto, Vili Niemelä

November 17, 2025

## 1 Introduction

The aim of this project is to develop a model to predict the forward returns from the US stock market. To do this, we utilize a dataset (Hull et al. (2025)) containing US market data and daily returns data from buying the S&P 500 and selling it the following day.

## 2 Exploratory data analysis

The dataset consists of 9021 observations with 98 features. The features consists of a date id, which acts as an identifier for a single trading day, market dynamics and technical features, macro economic features, interest rate features, price/valuation features, volatility features, sentiment features, momentum features and dummy/binary features. There are also features for forward returns, i.e. the returns from buying the S&P 500 and selling it the following day, risk free rate i.e. the federal funds rate, and market forward excess returns i.e. forward returns relative to expectations.

The data contains plenty of missing data. Figure 1 shows that the first thousand datapoints have 85 missing features and only the last two thousand contain all the features. This is likely due to the data originating from a long period of time and data availability. Missing data must be taken into account in the model training phase and should be dealt with in data preprocessing.

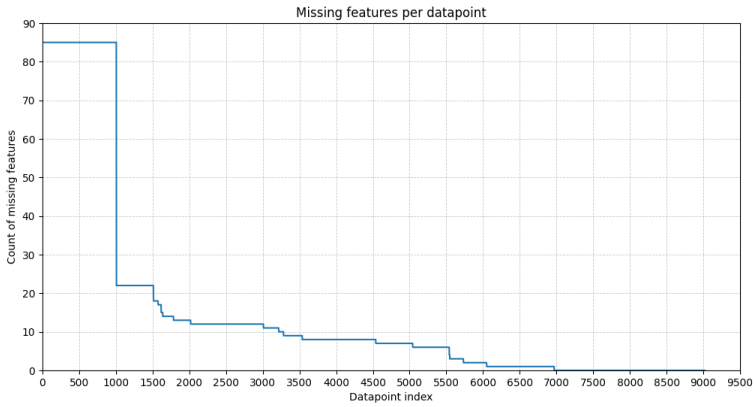


Figure 1: Missing features per datapoint.

Figure (2) shows the forward returns over time. From visual inspection, there does not seem to be any strong trends in the data, or even clear seasonality. However, the data shows that there are clear periods of higher volatility, where the variation in daily returns is high.

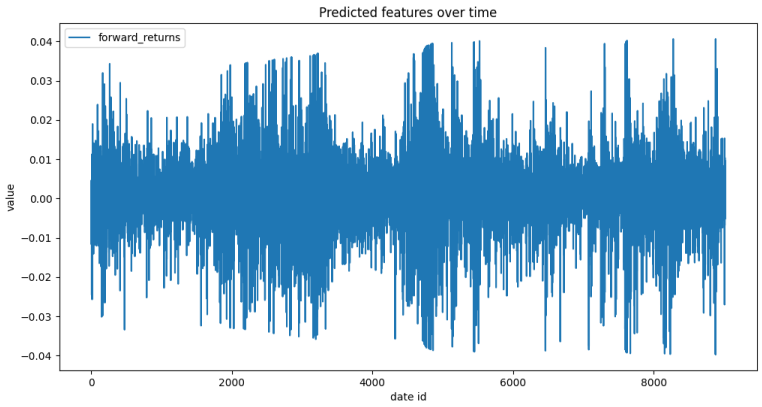


Figure 2: Forward returns time series.

The findings from figure (2) are supported by the distribution of forward returns observations, shown in the histogram (3). The values are distributed symmetrically around zero, with most of the density in the center. The high peak on the mean and symmetric spread around zero tells us, that most of the time the return is close to 0; high or low values are rarer, and getting positive returns is just as likely as getting negative returns.

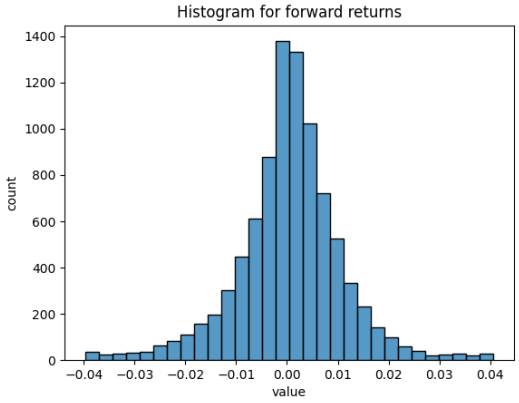


Figure 3: Forward returns histogram.

Figure (4) presents a correlation matrix of selected features in the data. Because of the high number of variables in the data, in order to keep the figure readable, the correlation matrix only shows the correlations for features which have strong ($|\text{corr}(x_i, x_j)| > 0.8,\ i \neq j$) positive or negative correlations with some other feature. The figure shows there are some strong correlations between the features, often between features of the same type. Notably, however, there are no strong correlations between the predictor features and forward returns.

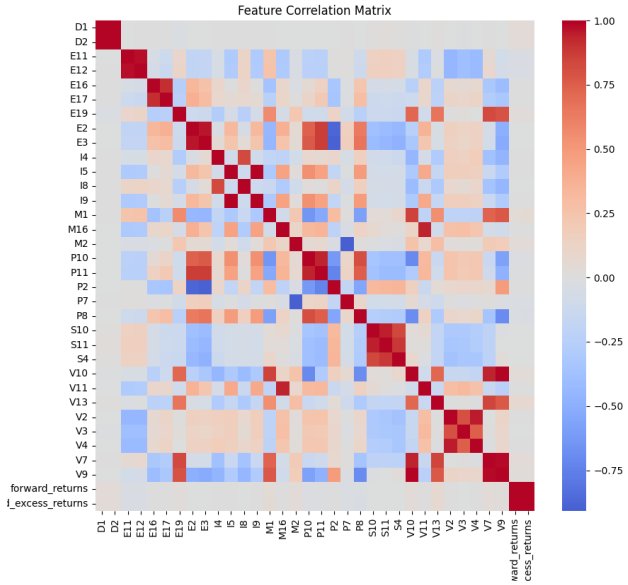


Figure 4: Correlation matrix for features with strong correlations.

## 2.1 Decomposition

Based on the Figure 2, the time series to predict does not seem to contain any clear long-term trend or seasonal pattern. In Figure 5 a zoomed trends of the time series can be seen. Based on the Figure, the time series does not seem to have either clear short-term trend or seasonal patterns. The time series seems to resemble a heteroscedastic Gaussian process.

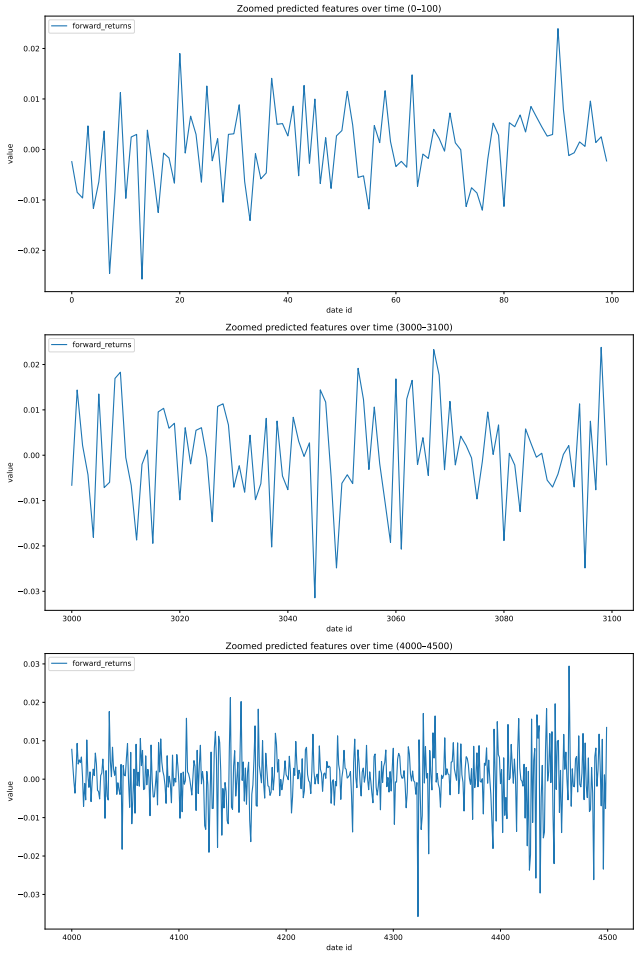


Figure 5: Zoomed forward returns.

We performed a regression analysis to the forward returns time series to confirm whether any linear trend can be identified. The estimated bias and trend coefficient terms are $[1.98082307 \times 10^{-8}, 3.82116136 \times 10^{-4}]$ which indicates that no long-term linear trend exists. As a conclusion, the forward return time series does not seem to have long-term trend or seasonal components to decompose.

## 2.2 Autocorrelation

In figure 6 we can see that the lag plot of forward returns looks like a round cloud and the ACF after lag 0 stays near zero, so yesterday's return doesn't help predict today's. When we fit a simple OLS model that uses yesterday's return to predict today's, the residuals look like noise. Their ACF is flat, so the mean part seems fine. But when we look at the absolute residuals, the ACF shows clear positive correlation that fades slowly. This could be interpreted to mean the volatility clusters. So larger moves and quieter periods tend to come in streaks. So the

findings suggest that returns or their residuals themselves don't show autocorrelation, but the volatility does.

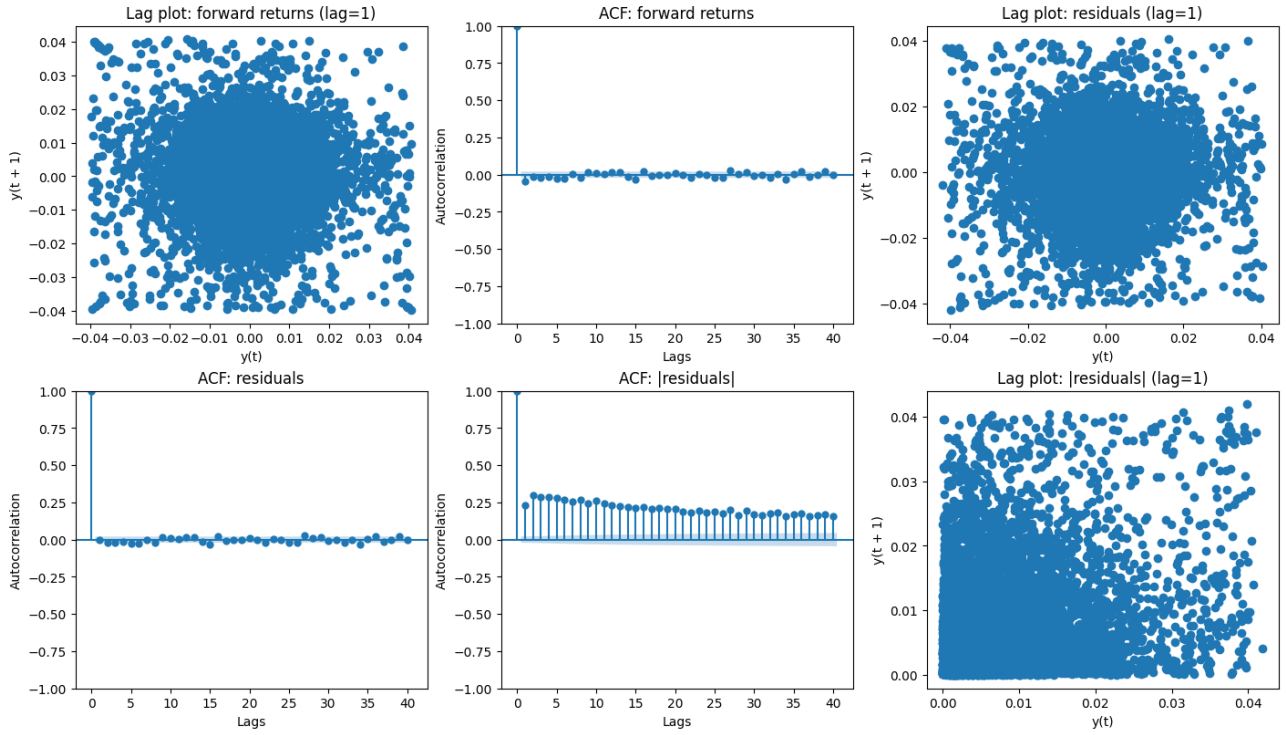


Figure 6: Autocorrelation plots.

## 2.3 Data partitioning for model formation

Partitioning a time-series dataset for model formation has its difficulties as the data has to keep its temporal structure. Our time-series is quite long and the variation seems to stay in a certain region. To perform robust training and validation for our model, we will use cross-validation in the form of a regular time-series split.

In a time-series split, we start with a smaller training data and validate on the next fixed amount of datapoints. Then we add the last validation set to our training set and validate on the next points. In this way, we use the most training data and we mimic the real life situation where we use all historical data to predict the future. The number of validation datapoints will be decided later, but we will start with 500. And the size of the first training data will be around a 1000 datapoints. These can change depending on the computational cost of the model and feature analysis. With this method, there is a possibility that the model learns patterns in the validation data and overperforms. If this seems to happen, we will test a slightly different form of splitting called the blocked time-series split, where the training data is always a fixed size and moves along with the validation data.

Our time-series has 9020 datapoints but many missing values in the beginning as we can see in figure 1. Because of this, we might have to ignore the first thousand datapoints, at least in the beginning. After we have performed an analysis on the features, we can better decide if we can use the first thousand datapoints and if we need to ignore more datapoints or predict some missing features. A little over two thousand of the last datapoints contain all the features and those datapoint will be used in the feture analysis.

# 3 Data pretreatment & baseline models

## 3.1 Data pretreatment

The data we have in our project, is in many ways very easy to work with. since it is not observations from any sensors etc. and the stock market's movement is tracked by numerous different parties, the data is in many way very complete. The absolute first thing we did was to remove the 2 other optional target variables that were provided in the dataset, which were *risk free rate* and *market forward excess returns*.

### 3.1.1 Continuity

The aforementioned facts apply especially continuity-wise. There are no gaps in the time-series, only consecutive days with records for each.

### 3.1.2 Asynchronicity

Again for the previously mentioned facts, the data is synchronous. We did not detect any asynchronicity in the features, everything is 1 observation per day.

### 3.1.3 Missing value handling

The data did have quite a lot of missing values. We observed that the first 1005 observations are missing pretty much all feature data. We assume these were not recorded yet by that time, as the data spans around 35 years. Then additionally there are features, which are missing a lot of consecutive observations, and some that do not have any records prior to around 500 or 1000 newest observations. We assume these are features that were begun to track only recently and cannot be deduced for the historical data.

As our initial approach, we chose to remove the first 1005 observations, and also remove the variables that were missing values. After this process, the data was left with zero missing values and 74 feature variables and 8013 observations.

## 3.2 STL decomposition for outlier detection

There are approximately 252 days in a year where the stock markets are open, so we decided to treat the observations in yearly cycles of 252 observations. We performed STL decomposition for the forward returns variable, to separate the seasonality, trend and residuals from each other.

STL decomposition separates the trend, seasonal and residual parts from the data, as seen in figure 7

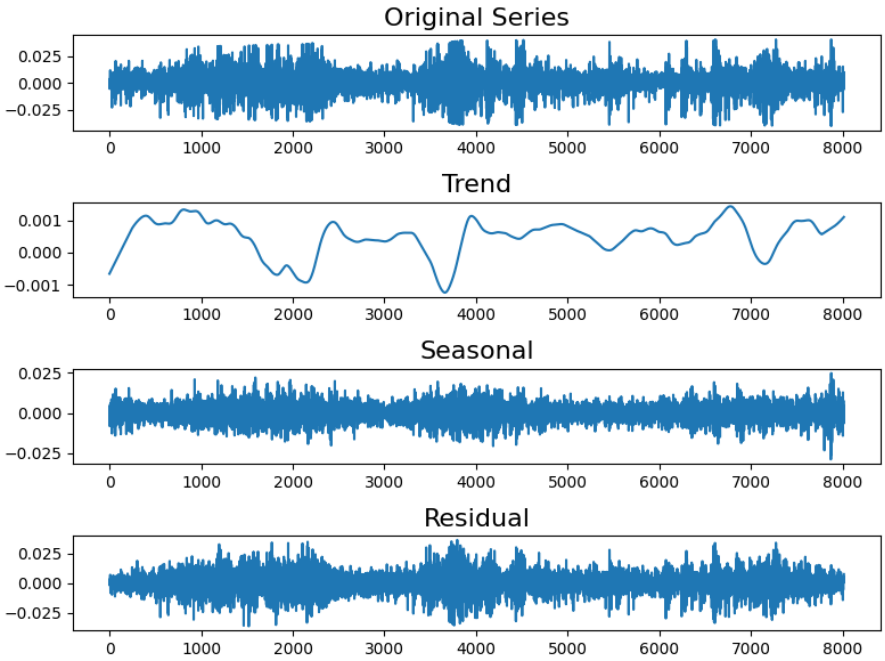


Figure 7: STL decomposition.

From the residuals, we performed anomaly detection by selecting the residuals that surpassed a limit of 3 times the standard deviation of the residuals. From this analysis, 97 anomalies were detected, which represents around 1.21% of all observations. In figure 8 we can see the the detected anomalies plotted in the original data.

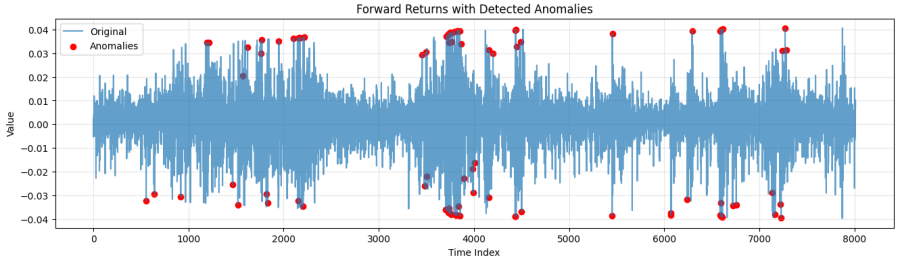


Figure 8: Detected anomalies.

## 3.3 Dimensionality reduction

The preprocessed dataset contains a high number of features (74), therefore we apply dimensionality reduction techniques to try and decrease the dimensionality in the data. We use two different dimensionality reduction to produce two different reduced datasets: linear dimensionality reduction method Principal Component Analysis (PCA), and the non-linear dimensionality reduction method kernel-PCA.

PCA decomposites the data matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$ to matrices $\mathbf{T} \in \mathbb{R}^{n \times k}$ i.e. the *variable loadings*, $\mathbf{P} \in \mathbb{R}^{m \times k}$ i.e. the *object scores* and $\mathbf{E} \in \mathbb{R}^{n \times m}$ i.e. the error and noise according to equation (1). In essence, the method projects the data to a new $k$-dimensional subspace such that the first component vector of $\mathbf{P}$ corresponds to the dimension in the data with the most variance, the second vector to the dimension with the second highest amount of variance

etc. In other words, the matrix $\mathbf{P}$ is the dimensionally reduced $k$-dimensional dataset which maximizes variance. (Salem and Hussein (2019))

$$\mathbf{X} = \mathbf{T}\mathbf{P}^{\top} + \mathbf{E} \tag{1}$$

Figure (9) shows the values of the principal components and the cumulative explained variance by principal components. According to the results, the dataset could be reduced to the first 30-40 component vectors while still retaining approximately 90-95 % of the variance.

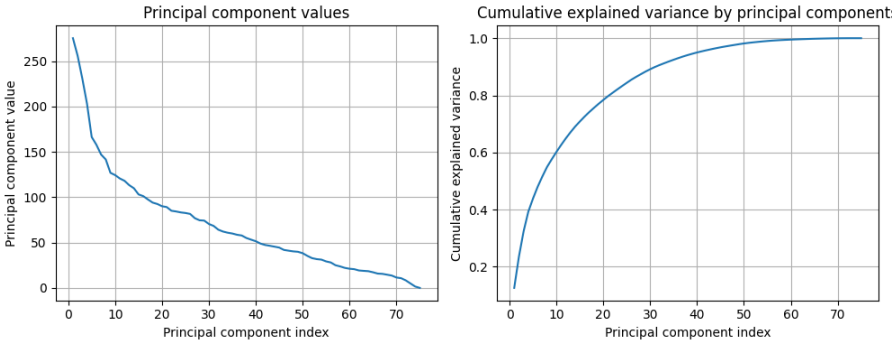


Figure 9: Principal components and cumulative explained variance.

Kernel-PCA (KPCA) extends standard PCA to non-linear spaces by using a kernel function $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_i) \rangle$ to calculate the covariance matrix (Gram matrix) $\mathbf{K}$ where $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. Eigenvalue decomposition is then applied to extract principal components $\lambda$ and component vectors $\mathbf{V}$. (Rosipal et al. (2001)).

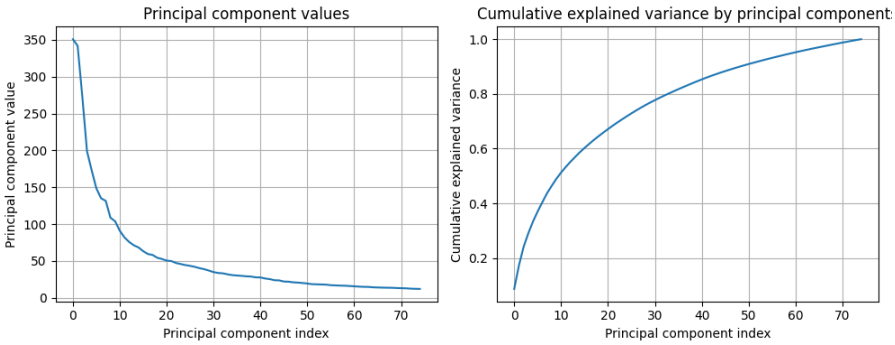


Figure 10: KPCA Principal components and cumulative explained variance.

Figure (10) shows the values of KPCA principal components and the cumulative explained variance by principal components. According to the results, the dataset could be reduced to the first 50-60 component vectors while still retaining approximately 90-95 % of the variance.

## 3.4 Literature review

Subsampling a long time series to smaller sequences is needed for robust results. Many of the methods used for subsampling are based on the fixed-origin and rolling-origin methods. In the fixed-origin method we use a fixed training set up to a certain timestep and validate the model on samples after that timestep. In rolling-origin we add more timesteps to the training data as

we predict further along the timeline, as in if we are predicting sample from t+1 we can use all samples up to t to predict that. In a rolling-window method we drop out older samples from the test set to keep a fixed length set as we move forward. (Tashman, 2000).

Research by Li et al.(2023) suggests that dividing the seasonal timeseries to subsamples which each contain one season, could provide better results. This should be done with different seasonalities on the data, for example weekly, monthly and yearly.

Recurrent neural networks like long short-term memory can model some seasonality as they as they capture long term patterns. However deseasonalization should be used unless the seasonality is clearly homogeneous (Hewamalage et al., 2021). For deseasonalization another model, like seasonal trend decomposition, can be used together with the LSTM to form a more robust hybrid model (Jaiswal et al., 2025).

LSTMs are sensitive to input scale, so the inputs should be standardized or normalized. Some basic methods are min-max scaling and z-score scaling. More advanced methods are normalizing even between the hidden layers of the LSTM. This has can provide better generalization and faster model training times. (Cooijmans et al., 2017).

## 3.5 Baseline model

As a baseline model we decided to use a dynamic autoregressive model (AR-model) which uses 15 previous return values to predict next returns with a rolling window. The AR(15) model can expressed as follows

$$X_t = \beta_0 + \sum_{i=1}^{15} \beta_i X_{t-i} + \epsilon_t, \tag{2}$$

where $X_t$ is the forward return at time index $t$ and $X_{t-i}$ is the $i$:th lag of the forward return. $\beta_0$ is the estimated constant term and $\beta_i$ is the regression coefficient for the $i$:th lagged forward return value.

AR-model assumes that the time series is stationary meaning mean zero and constant variance. From the Figures above, it can be quite clearly seen that the forward returns time series is not stationary since the variance clearly changes with respect to time. However, our assumption is that the forward returns time series is locally stationary, meaning that small sample time series are stationary. We decided to select 100 length time series to test whether randomly selected such time series from the forward returns are stationary. If that is true when training the dynamic autoregressive models with 100 samples, we do not have to stationarize the time series always before training. To study whether our assumption is correct we used Augmented Dickey-Fuller test, which is a statistical stationary test for time series. Based on the test results, randomly selected 100 length sample time series from the forward returns are stationary. Because of the stationary conclusion, we decided to train the dynamic AR-models with 100 length time windows.

The training was done with the rolling time windows in a way that first the model is trained using 100 samples and then tested with the next 50 samples. The next training data starts from the index after the last index of the last training dataset. In Figure 11, the visualization of the used time series splitting technique is presented.

Two baseline models were implemented, one which predicted the forward returns one day ahead and another which predicted the returns 50 days ahead. For the both models, Python's statsmodels toolbox was used. From the Figures 12 and 13, the baseline model predictions one and 50 days ahead can be seen. Based on the Figures, a significant difference in the predictions accuracies cannot be seen which is why the models were evaluated with the root mean squared

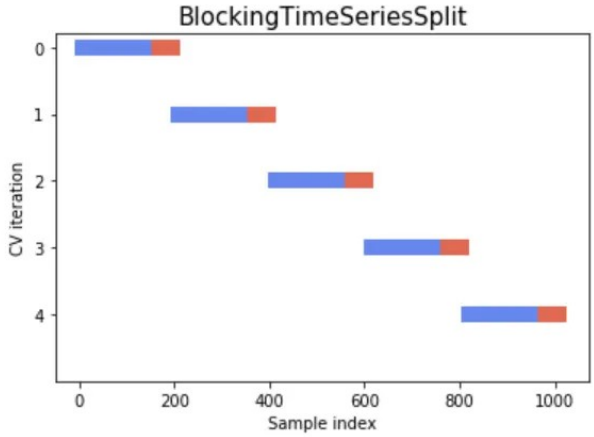


Figure 11: The method to split time series into training and validation datasets visualized.

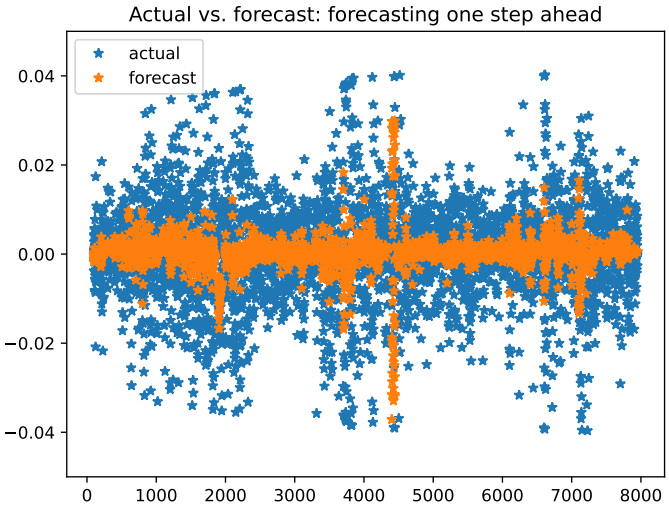


Figure 12: The one day ahead prediction results of the baseline model.

error metric which can be expressed as follows

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i}^{n}(y_{\mathrm{true}} - y_{\mathrm{pred}})^2}, \tag{3}$$

where $n$ is the number of test samples. The corresponding error metrics for the one day and 50 days ahead predictions were 0.0115227 and 0.014618 respectively. Based on the RMSE values, more accurate predictions can be obtained when predicting only one day ahead which was quite predictable.

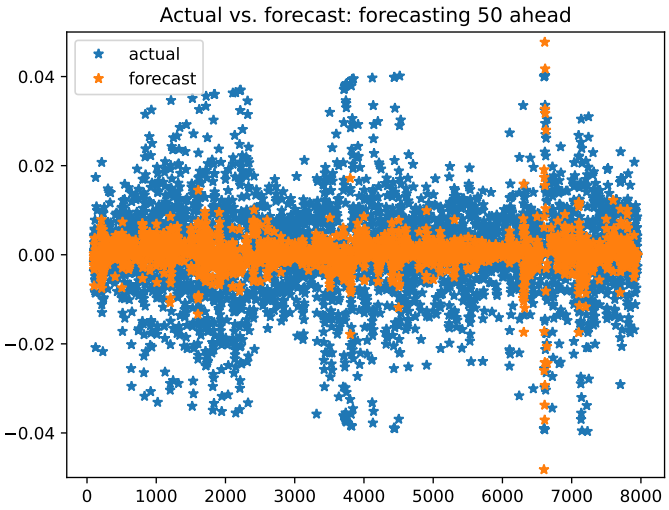


Figure 13: The 50 days ahead prediction results of the baseline model.

# References

Cooijmans, T., Ballas, N., Laurent, C., Çağlar Gülçehre, and Courville, A. (2017). Recurrent batch normalization. *arXiv.org*.

Hewamalage, H., Bergmeir, C., and Bandara, K. (2021). Recurrent neural networks for time series forecasting: Current status and future directions. *International journal of forecasting*, 37(1):388–427.

Hull, B., Bakosova, P., Lanteigne, L., Shah, A., Sinclair, E. C., Fast, P., Raj, W., Janecek, H., Dane, S., and Howard, A. (2025). Hull tactical - market prediction. https://kaggle.com/competitions/hull-tactical-market-prediction. Kaggle.

Jaiswal, R., Jha, G. K., Kumar, R. R., and Choudhary, K. (2025). Stl-lstm hybrid model for forecasting seasonal agricultural price series. *Annals of data science*.

Li, X., Petropoulos, F., and Kang, Y. (2023). Improving forecasting by subsampling seasonal time series. *International journal of production research*, 61(3):976–992.

Rosipal, R., Girolami, M., Trejo, L. J., and Cichocki, A. (2001). Kernel pca for feature extraction and de-noising in nonlinear regression. *Neural computing applications*, 10(3):231–243.

Salem, N. and Hussein, S. (2019). Data dimensional reduction and principal components analysis. *Procedia Computer Science*, 163:292–299. 16th Learning and Technology Conference 2019Artificial Intelligence and Machine Learning: Embedding the Intelligence.

Tashman, L. J. (2000). Out-of-sample tests of forecasting accuracy: an analysis and review. *International journal of forecasting*, 16(4):437–450.