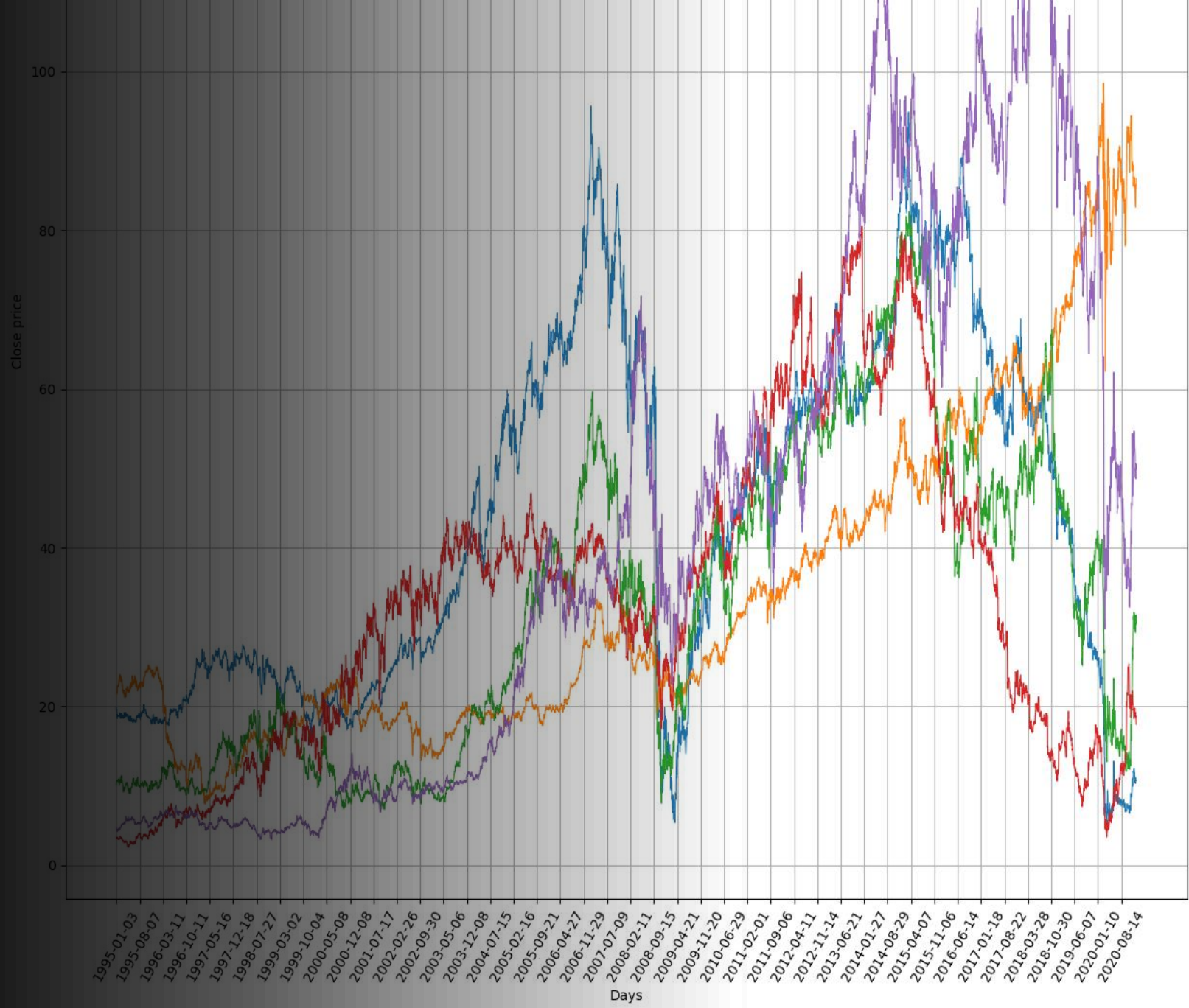
Stock market forecasts with ML

Comparison of different ML algorithms

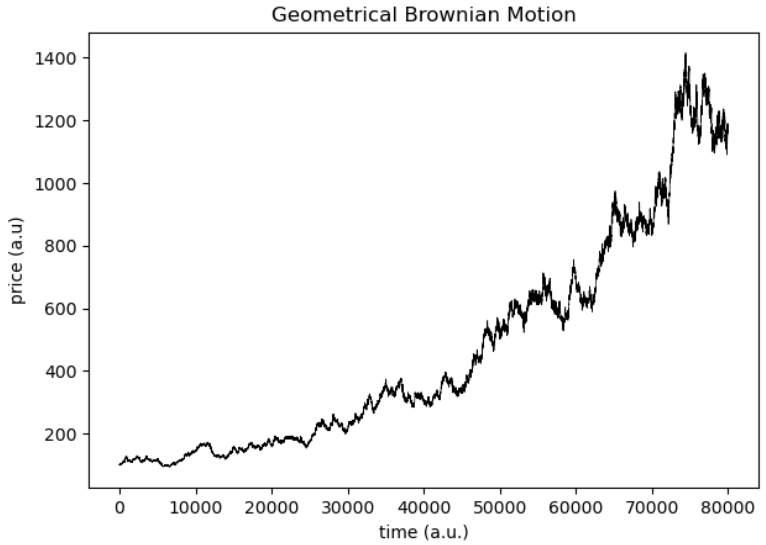


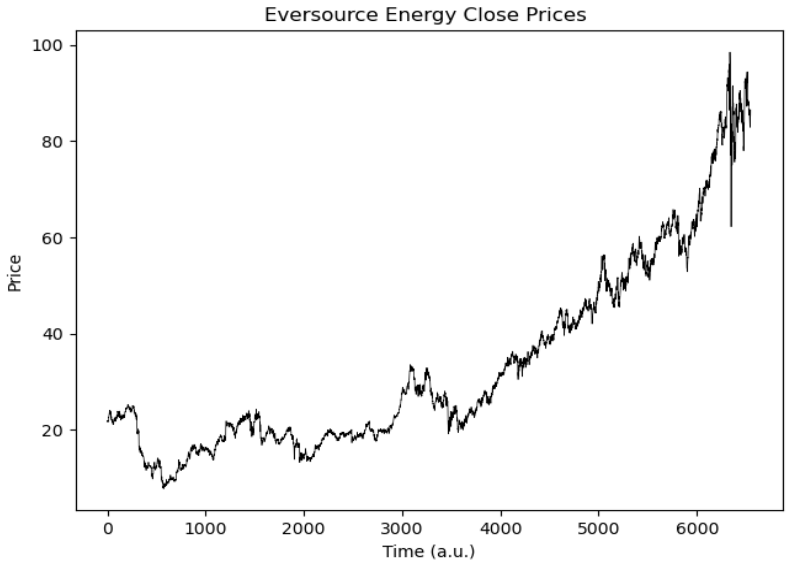
Summary

- Introduction
- Data
- Procedure
- Algorithms
- Pre Processing
- Results

Introduction

- Efficient Market Hypothesis → Prices follow a random path
- Evidence shows that this is not the case.
- Can ANNs capture the non linear information and predict future prices?
- How can we use results from ANNs to set a winner trading strategy?
- To answer these questions we used four different algorithms: LSTM, RAF, DNN, CNN





Data

- Data refer to close prices of 365 companies from S&P500 in starting from 01/01/1999 up to 31/12/2020.
- From this data we obtained the dataframe of m period returns:

$$R_t^{m,s} = \frac{P_t^s}{P_{t-m}^s} - 1$$

with

s = company
m = period returns
t = time

- We set our model as a classification one instead of a regression one: for each day, every entry in the returns dataframe becomes 0 if it is less than the median of returns of that day and 1 otherwise.

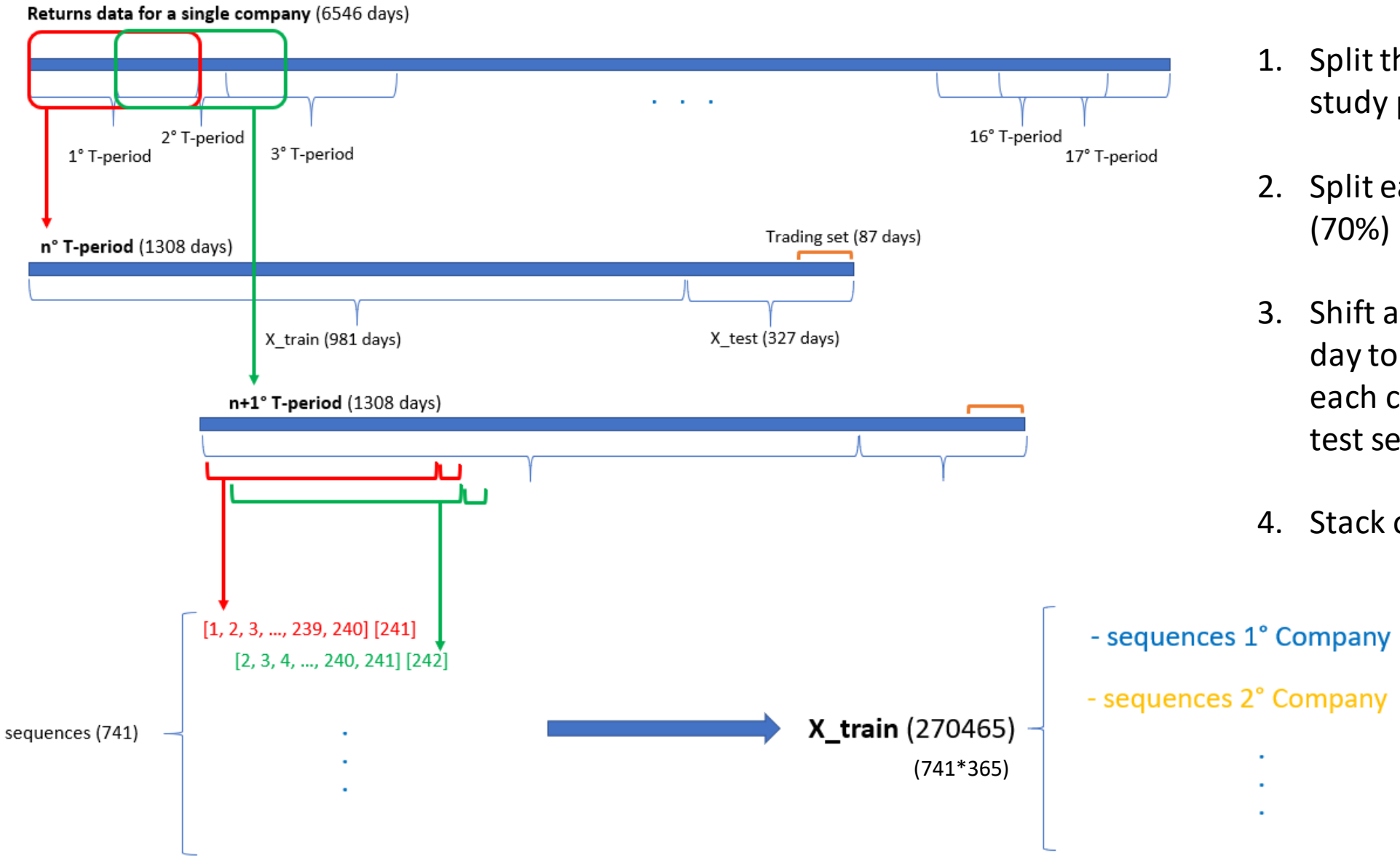
	ALL	SIVB	XRX	AJG	PNC	CAT	FDX	SCHW
0	-0.005155	-0.018868	0.007732	0.007937	0.034483	-0.002273	-0.002070	0.015038
1	0.000000	0.028846	0.003836	-0.023622	0.016667	0.013667	0.010373	-0.003704
2	0.010363	0.009346	0.005096	-0.012097	-0.032787	0.017978	-0.002053	0.026022
3	0.015385	0.000000	-0.002535	0.000000	-0.005650	0.019868	0.004115	0.018116
4	0.000000	-0.009259	0.019060	-0.008163	-0.005682	0.010823	-0.002049	0.014235

5 rows × 365 columns

	ALL	SIVB	XRX	AJG	PNC	CAT	FDX	SCHW
0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	1.0
1	0.0	1.0	1.0	0.0	1.0	1.0	1.0	0.0
2	1.0	1.0	1.0	0.0	0.0	1.0	0.0	1.0
3	1.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0
4	0.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0

5 rows × 365 columns

Data



1. Split the data in 17 overlapping study periods.
2. Split each period in training set (70%) and test set (30%).
3. Shift a 240 days-window by m=1 day to construct all sequences of each company both for train and test set.
4. Stack one over the other.

Methodology

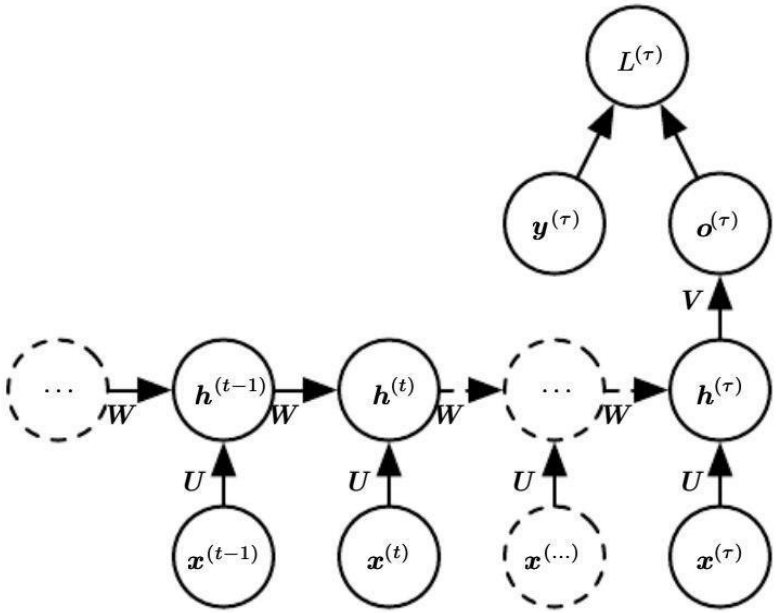
- Set the hyperparameters for the particular algorithm in exam.
- Train the model on 80% of the train set and use the remaining 20% as validation set for each study period.
- The best model (set of hyperparameters) is selected looking for the best validation accuracy and losses.
  a. Since we are dealing with a classification problem, loss function is set to binary cross entropy.
- Use the trained model to make prediction on the test set and compute the ROC curve.
- Use these predictions (class label of the following days) to create the portfolios over the actual 87 days of trading in each period.
  a. Portfolios are created sorting the probabilities given by the models and going short on the first 10 companies and long on the last 10 companies.
- Check the performance in terms of accumulative returns, at this stage also a randomness test (Welch's t-test) is computed.
  a. The randomness test is performed by sampling randomly 10 companies for shorting and 10 for longing. This is done 500 times (our number of monkeys).

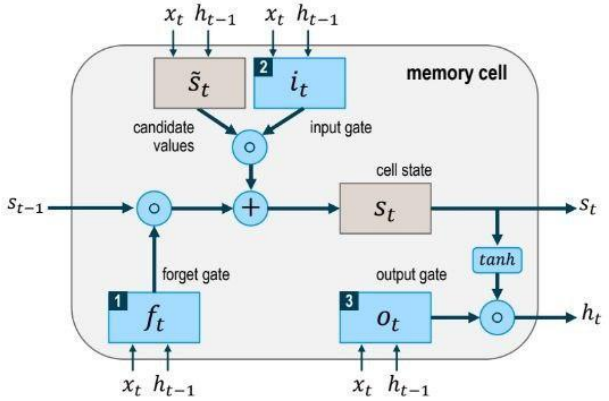
Accumulative returns ⟹ $m_{t+1} = m_t + \frac{1}{20}\sum_{i=1}^{20} 0.1 \times m_t \times r_{ti}$

m_t amount of money at time t
$\mathbf{r_t}$ vector of all the 20 companies returns at time t

Algorithms – Long Short-Term Memory

- Long Short-Term Memory networks belong to the family of Recurrent Neural Networks (RNN).
- It has the capability of reducing the problem of gradient explosion or vanishing.
- The main characteristic of these networks can be found in the hidden layer consisting of the so called memory cells.
- A sequence of data passes through hidden units that can store past information until the last one that outputs a vector of predefined shape.





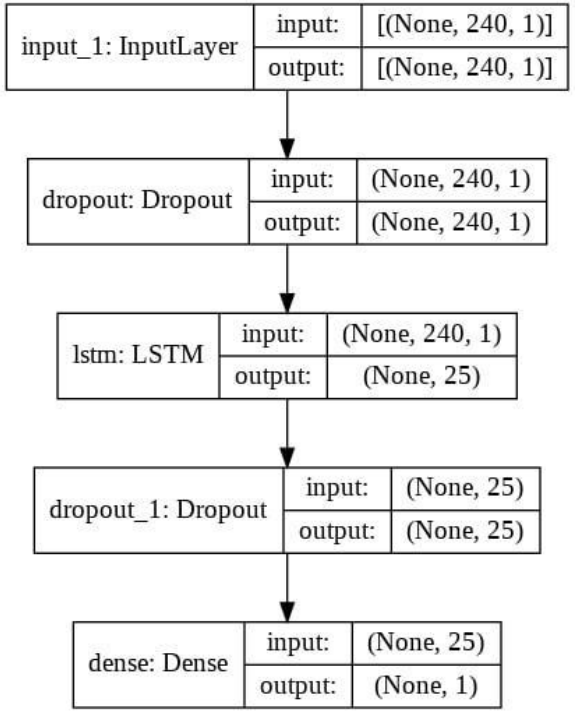
1 Forget gate:
Defines which information to remove from the memory (cell state)

2 Input gate:
Defines which information to add to the memory (cell state)

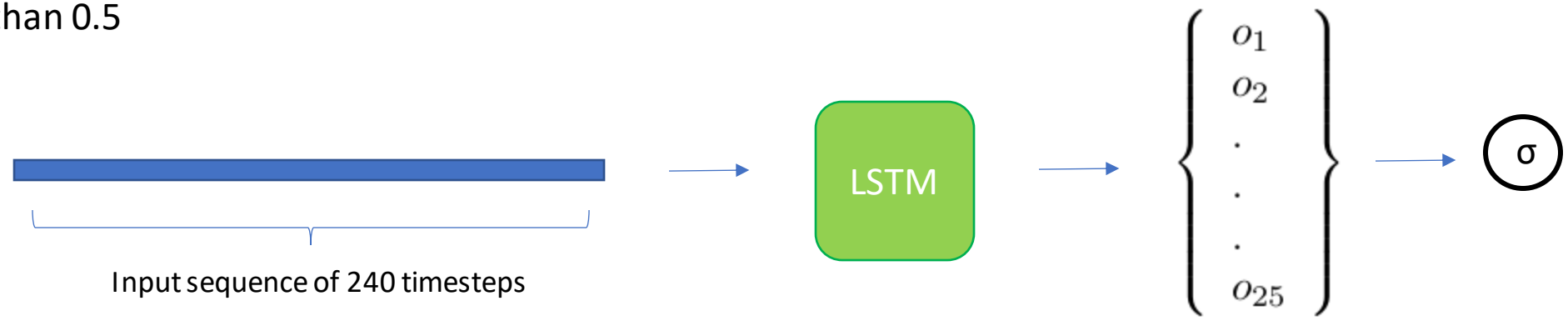
3 Output gate:
Defines which information from the memory (cell state) to use as output

Algorithms – Long Short-Term Memory

- The model we chose is composed of:
  - An input layer of shape (None, 240, 1)
  - A dropout layer set to 0.2
  - A LSTM layer composed by 25 hidden neurons
  - A dropout layer set to 0.2
  - Dense layer with 1 node and activation function set to sigmoid

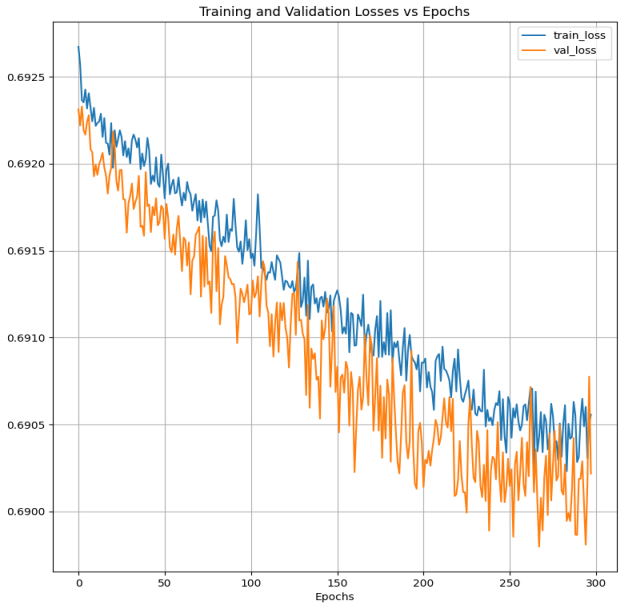


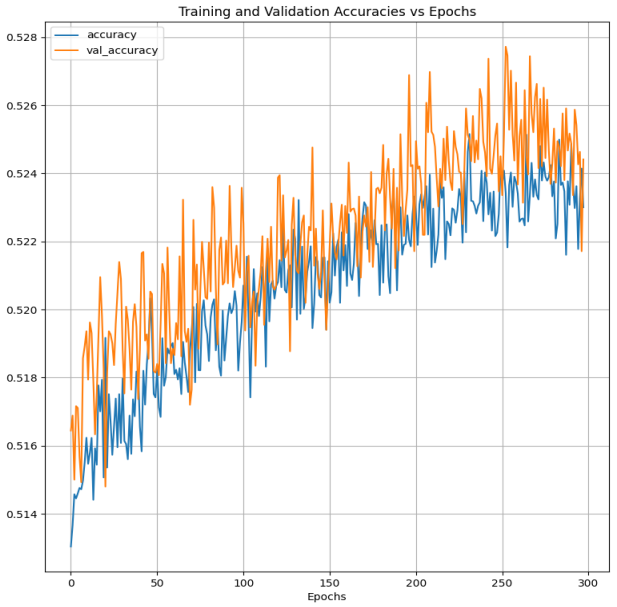
- RMSprop optimizer with learning rates 0.005 and momentum 0.5 gave better results that Adam with same learning rate.
- We had to set clipvalues value to prevent gradient explosion (None value as accuracy and loss): 0.3 better than 0.5

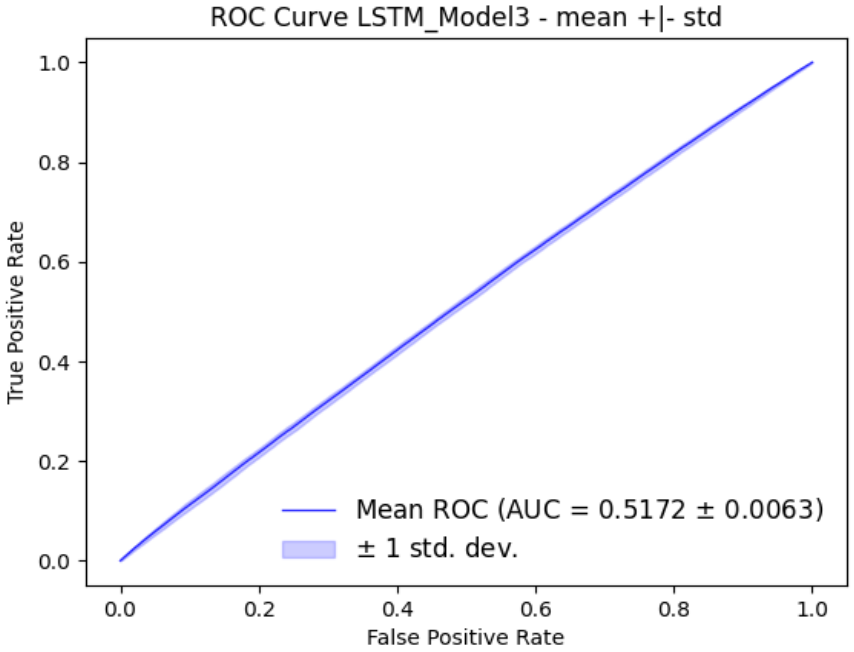


Algorithms – Long Short-Term Memory

- The model was fitted through a maximum value of 800 epochs. Such value was never reached due to the patience set to 30.
- Batch size value was 768: good tradeoff between noise and number of gradients updatings.

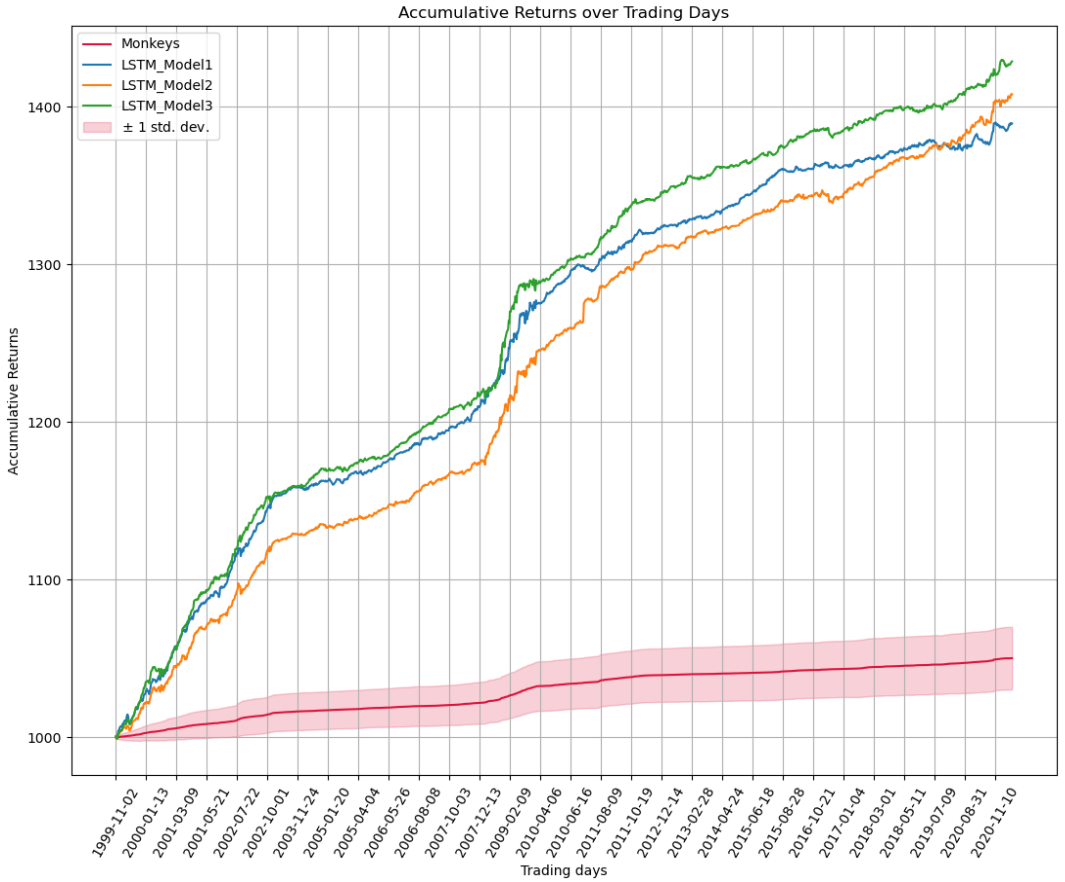


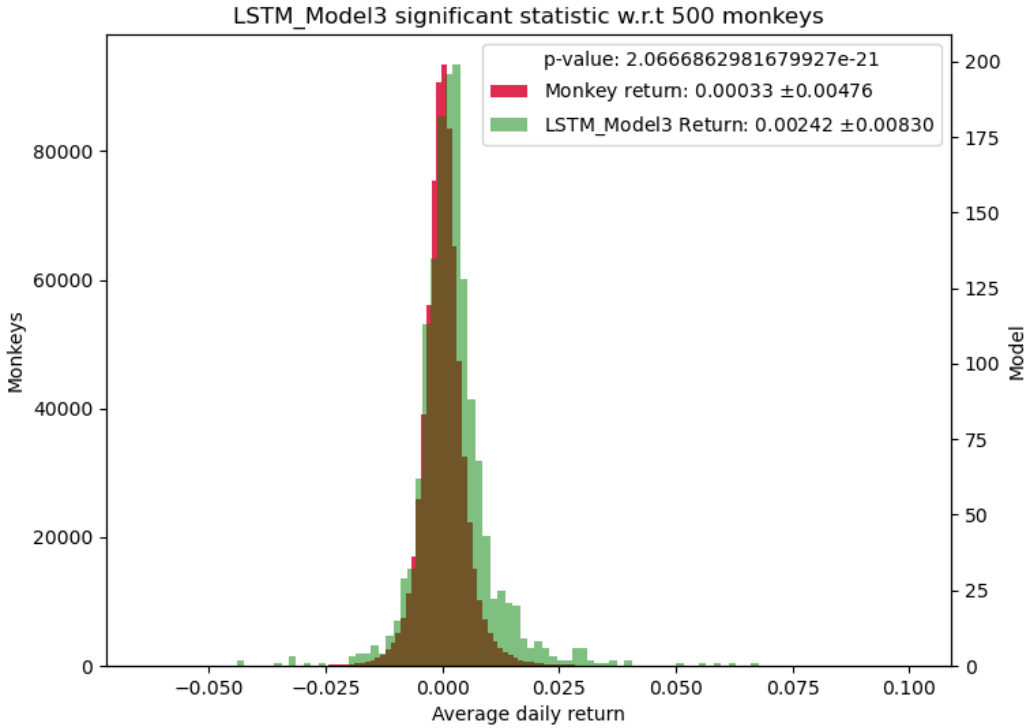




Algorithms – Long Short-Term Memory

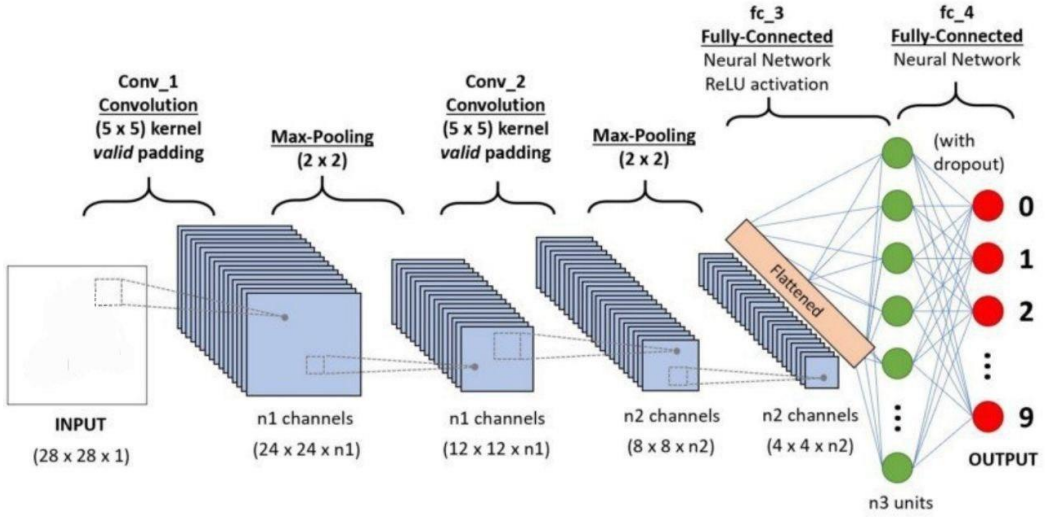
- Better results were obtained with a clipvalue of 0.3, probably due to the ability of storing most equally long-term dependecies and short–term ones.
- Starting from 1000$, the model was able to end with ≈ 1420$

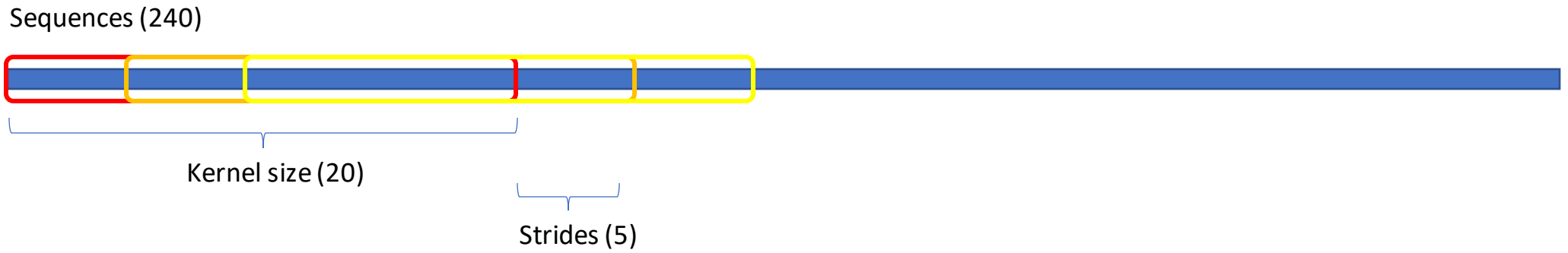




Algorithms – Convolutional Neural Network

- CNN allows us to extract local features inside the input data.
- The hyperparameters of Convolutional1D were set following the "nature" of the problem: kernel size is almost one month of trading days (20) and the stride was set to one week (5)

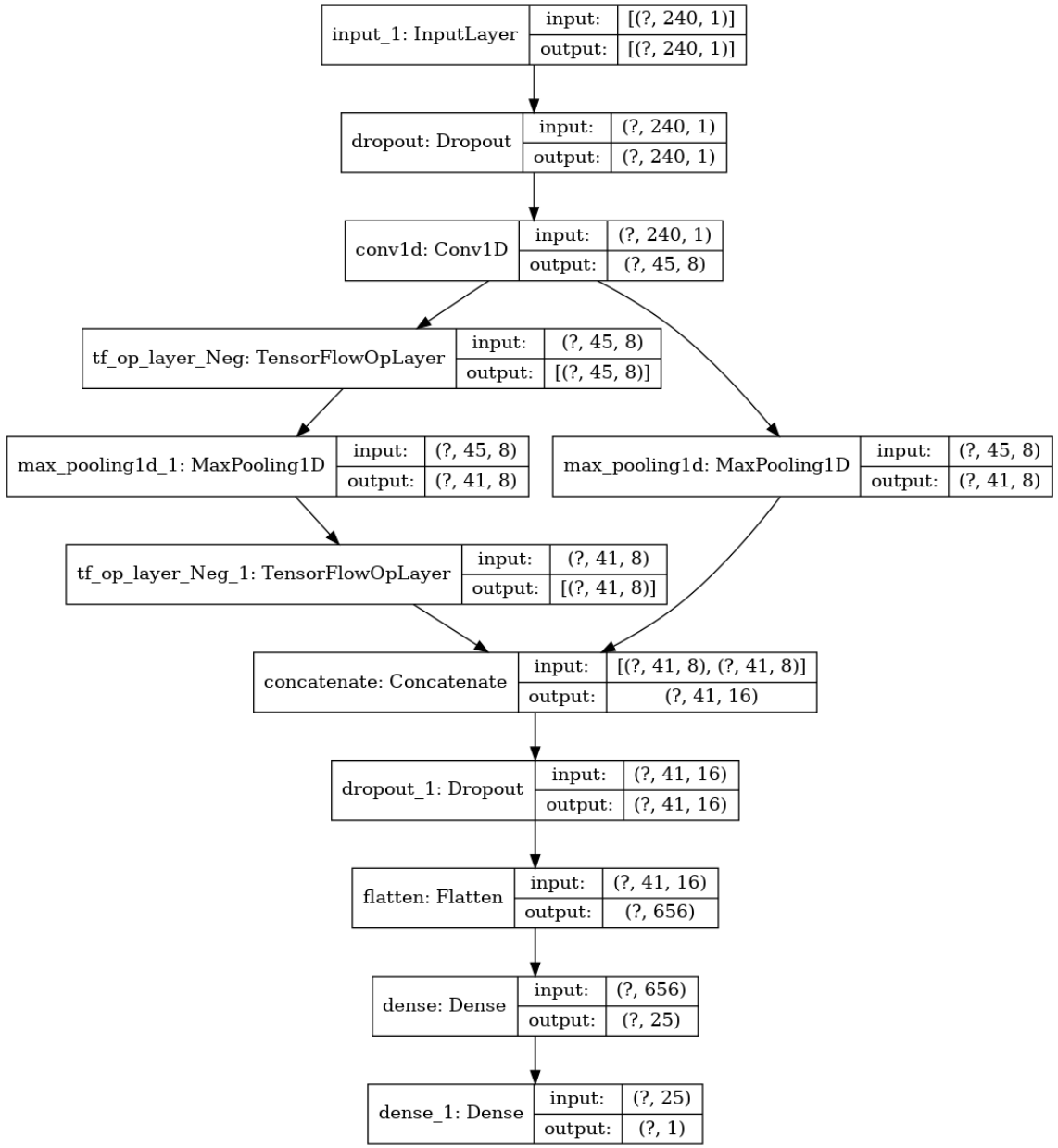




- Maxpooling pooling layer is connected with the convolutional one to allow the model to learn the maximum value inside the kernel.

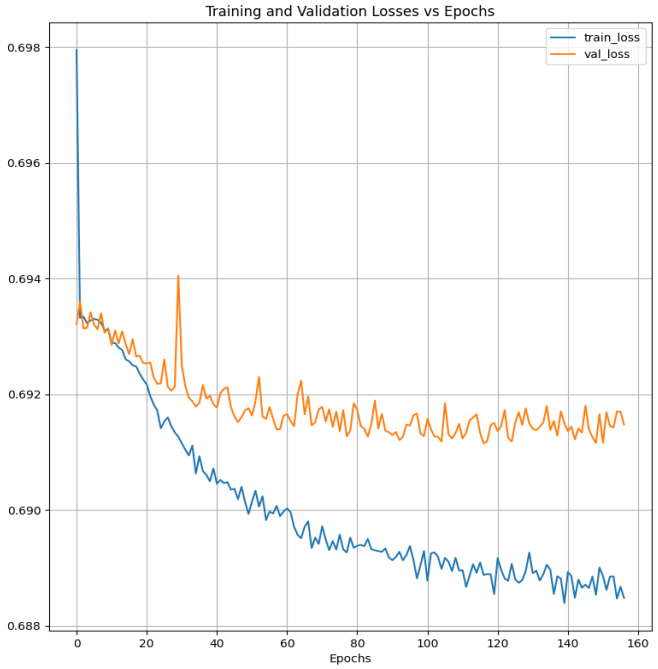
Algorithms – Convolutional Neural Network

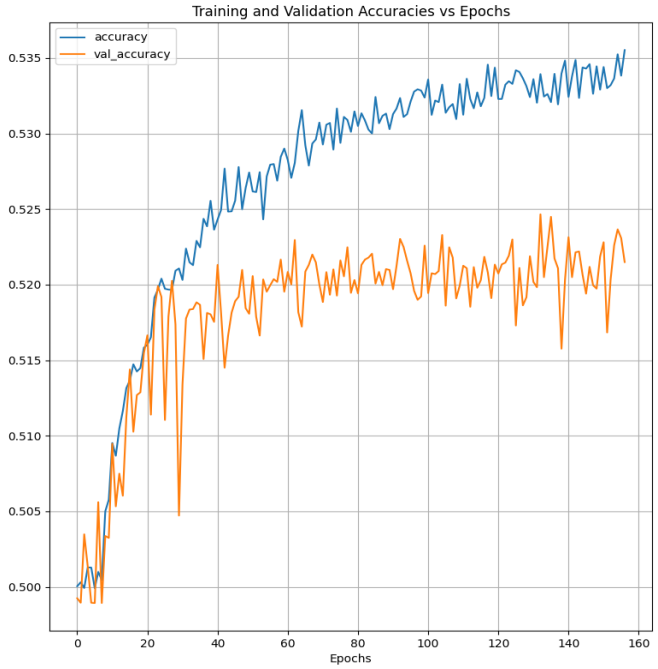
- The models have the following structure:
  1. Input layer: shape = (None, 240, 1)
  2. Conv1D layers: activation="tanh", filters = 8, 15
  3. Pooling layer/s: one or two pooling layers (kernel = 5)
  4. Flatter
  5. Dense layer: 25 nodes and "tanh" as activation function
  6. Output layer: single node with "sigmoid" as activation

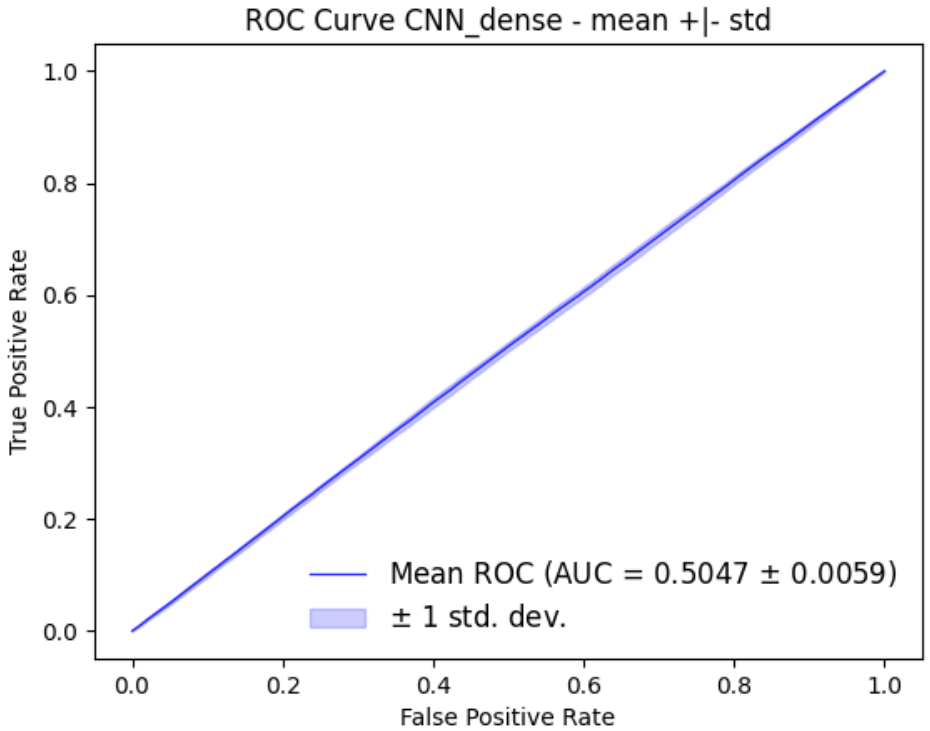


Algorithms - Convolutional Neural Network

- Models were trained on each training set of the T-study periods for 400 epochs, patience set to 40 and batch size 518 together with Adam as optimizer.

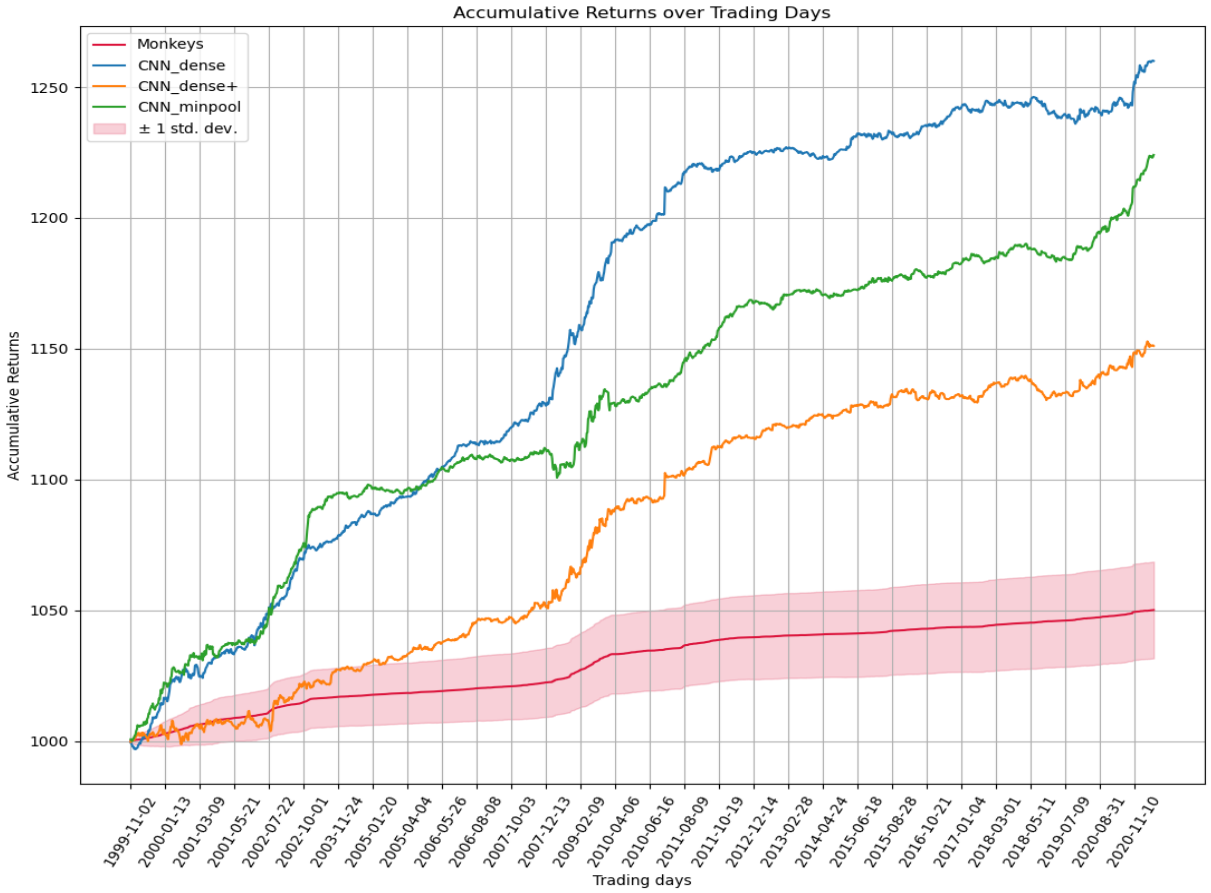


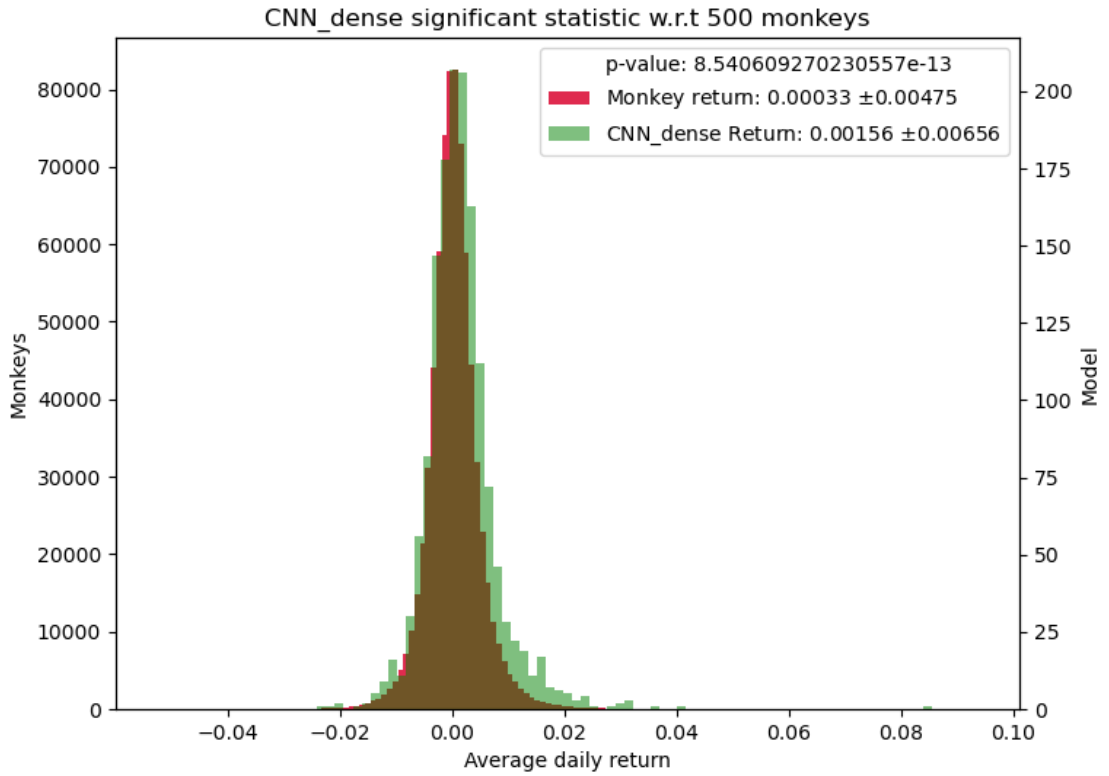




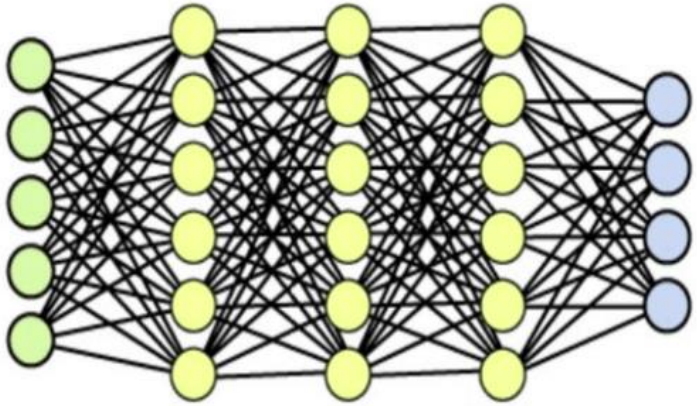
Algorithms – Convolutional Neural Network

- Trading period consists of the 17 tests set
- Starting from 1000$, the model was able to end with ≈ 1250$





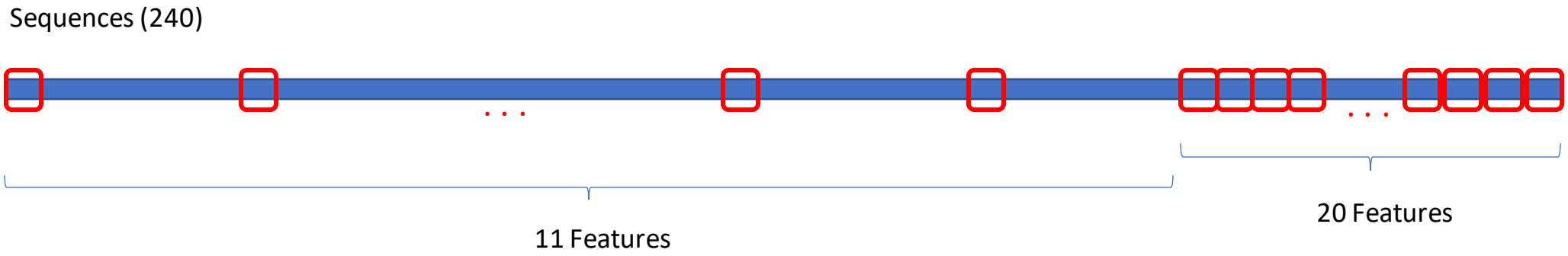
Algorithms – Deep Neural Network



- DNN is composed by several full-connected layers with different numbers of nodes

- From the input sequences of 240 values we extracted 31 features according to an importance rule.

- The nearest 20 returns (almost one trading month) to the trading day plus more past values that are selected with a separation of 20 days.

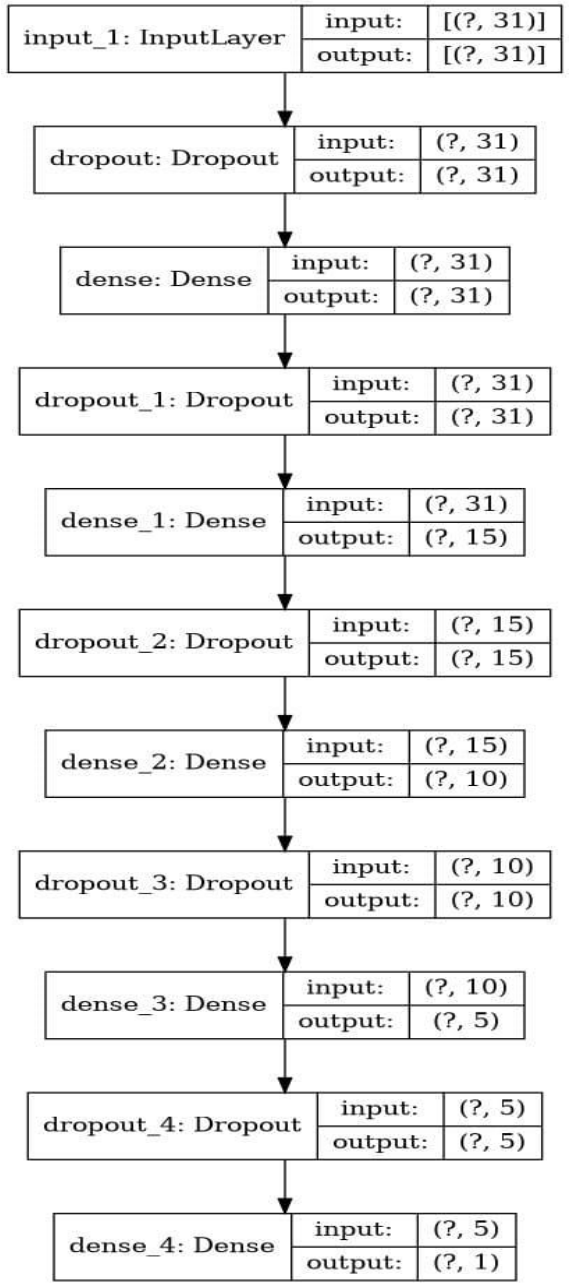


Algorithms – Deep Neural Network

- The models have the following structure:
  1. Input layer: shape = (None, 31)
  2. Hidden layers: 4 hidden layers with "tanh" as activation ad number of nodes in decrescent way
  3. Output layer: single node with "sigmoid" as activation

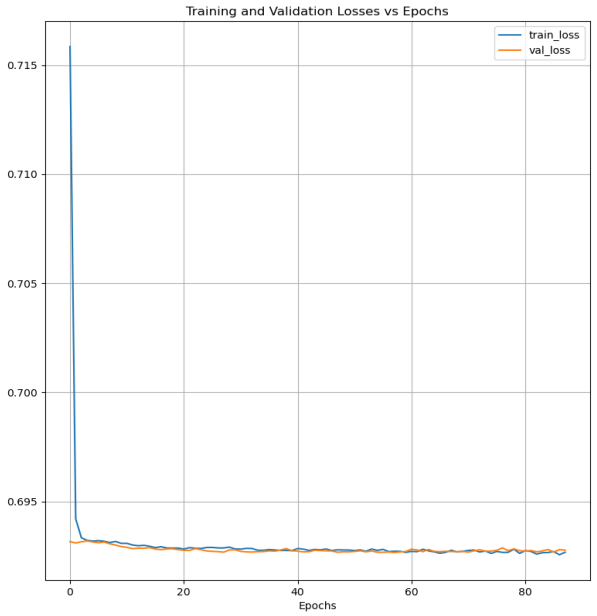
- A dropout layer (0.1 for input and 0.5 for the other) is added between each layers to minimize the noise during training and validation losses.

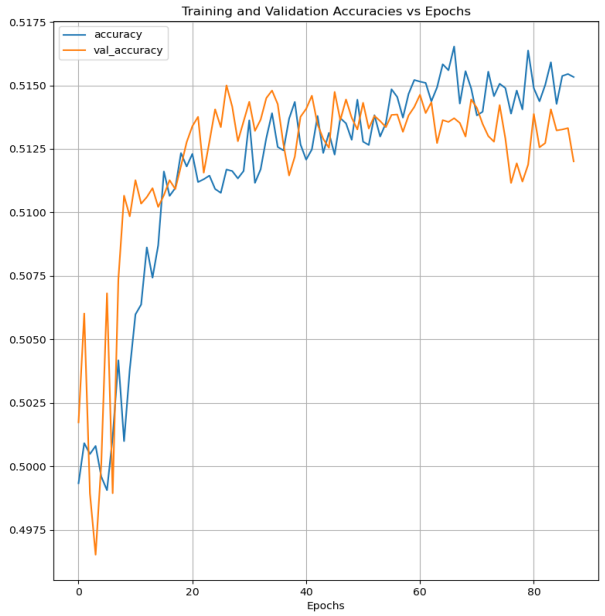
- Model 1: [31, 15, 10, 5]
- Model 2: [150, 50, 10, 5]
- Model3: [150, 80, 15, 5]

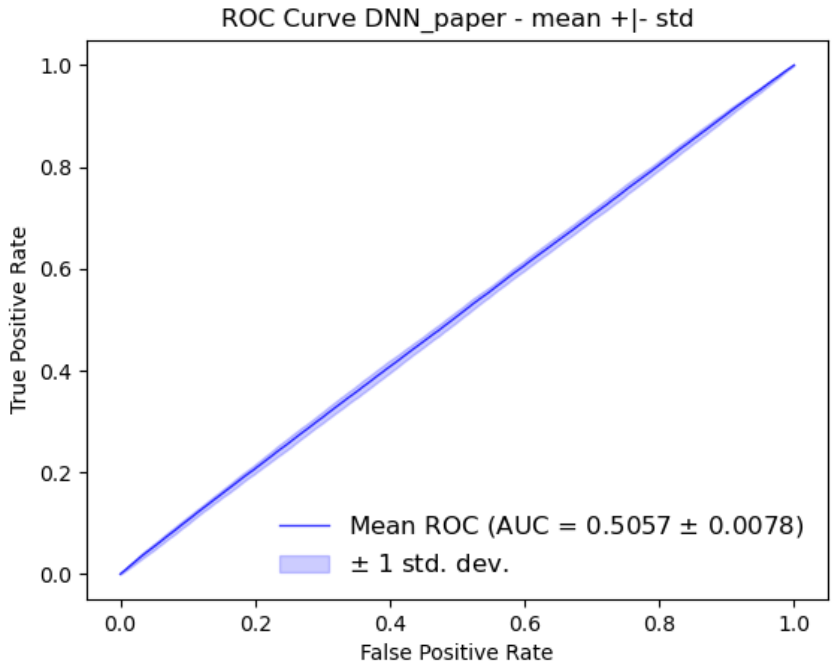


Algorithms – Deep Neural Network

- The models are trained on each training set of the T-study periods for 400 epochs, patience was set to 30 and batch size 256.

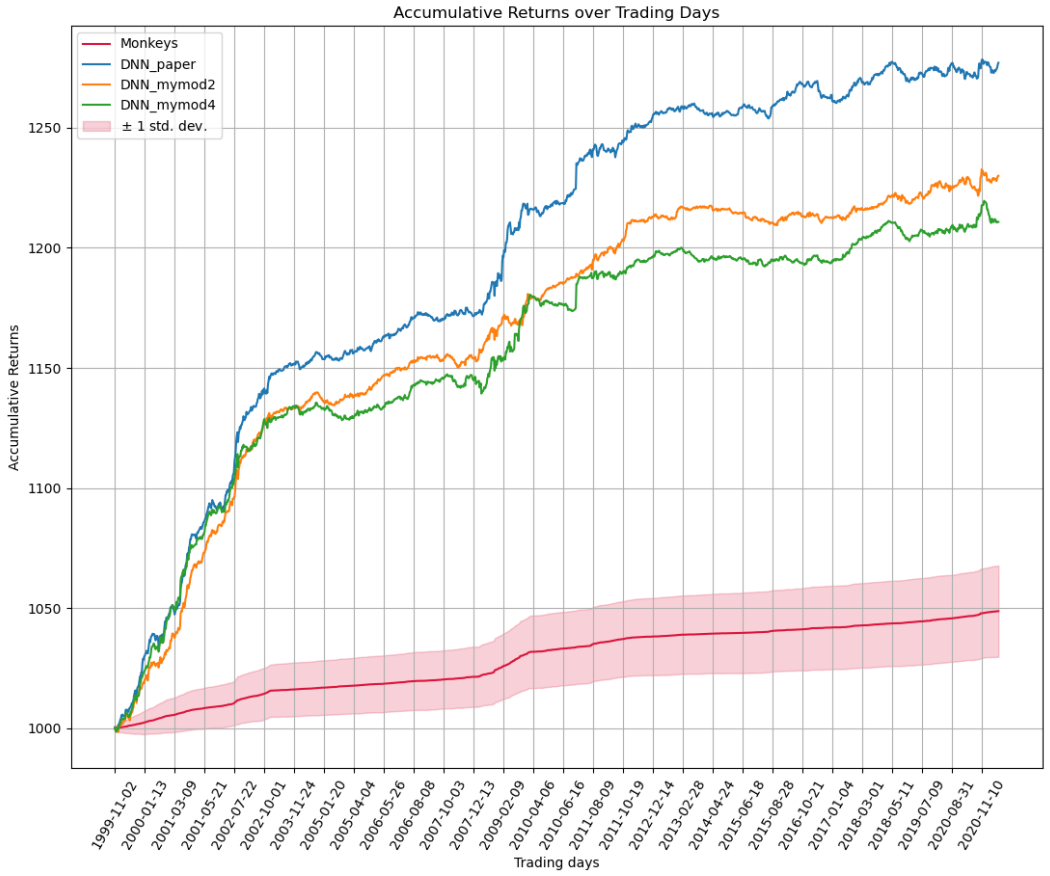


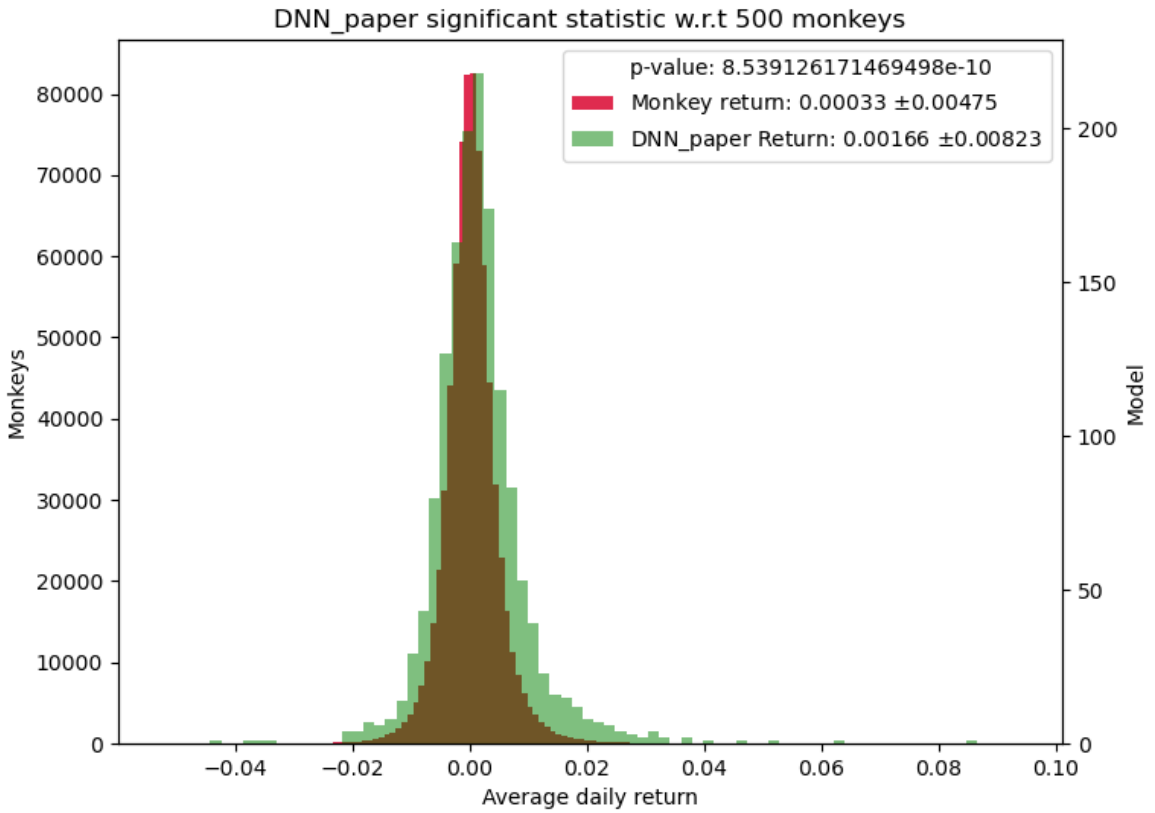




Algorithms – Deep Neural Network

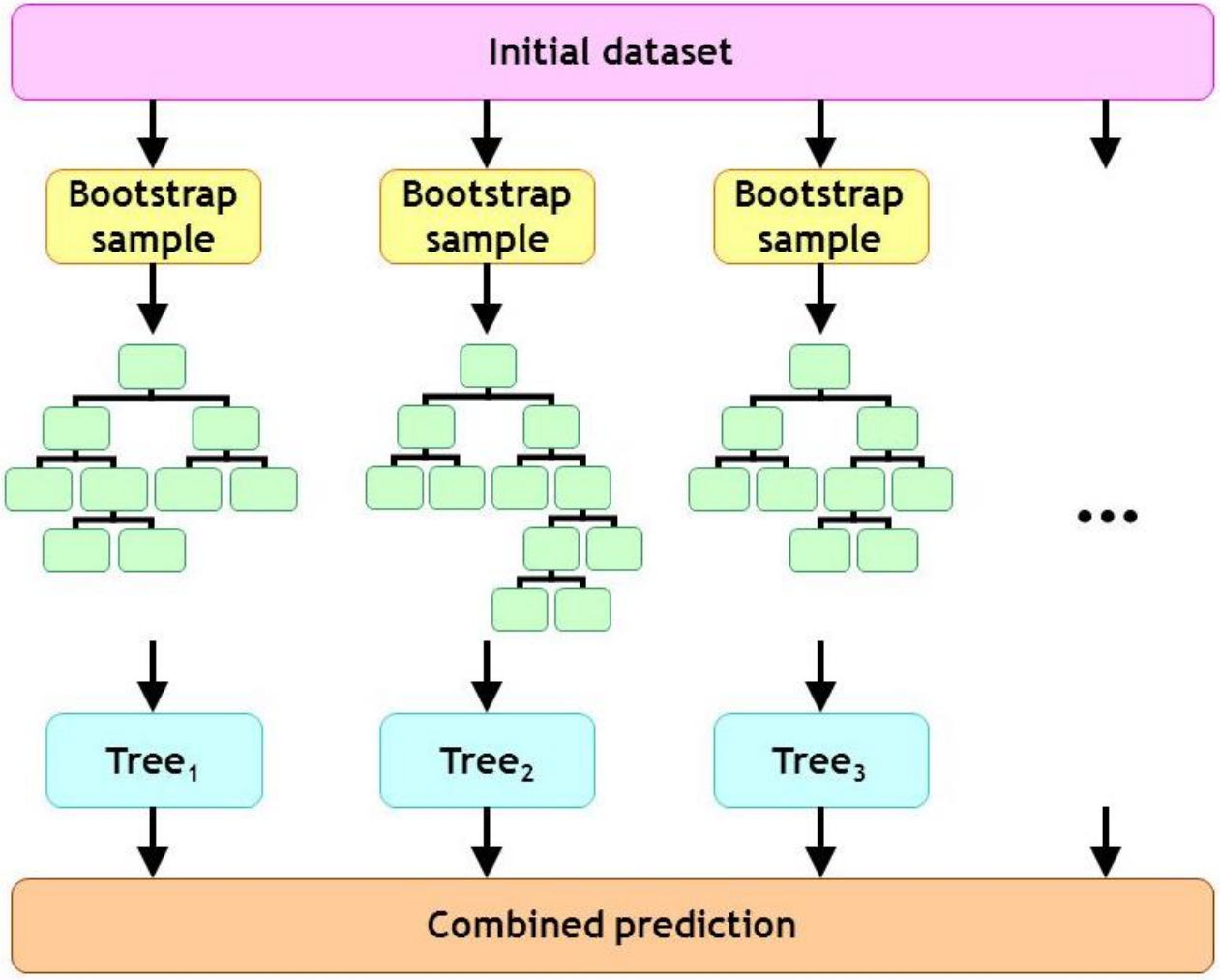
- Trading period consists of the 17 tests set
- Starting from 1000$, the model was able to end with ≈ 1250$





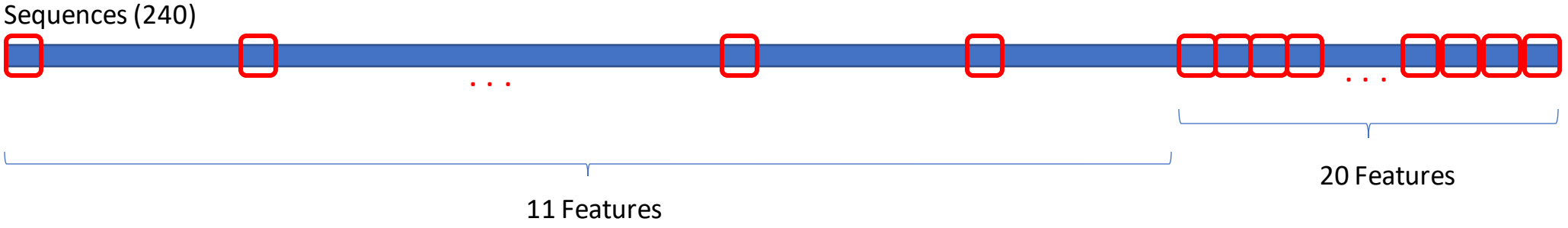
Algorithms – Random Forest

- Random Forest are an ensemble learning method for classification or regression that operates by constructing a multitude of decision trees
- Normally trees that are grown very deep tend to learn highly irregular patterns: they overfit their training sets.
- To avoid this, RAF are a way of averaging multiple deep decision trees, with the goal of reducing the variance.
- RAF is invariant under scaling and various other transformations of feature values and is robust to inclusion of irrelevant features.



Algorithms – Random Forest

- As DNN, from the input sequences of 240 values we extracted F = 31 features according to an importance rule. The features were the m period returns with m in [1, 2, ..., 20] and [40, 60, ..., 240].

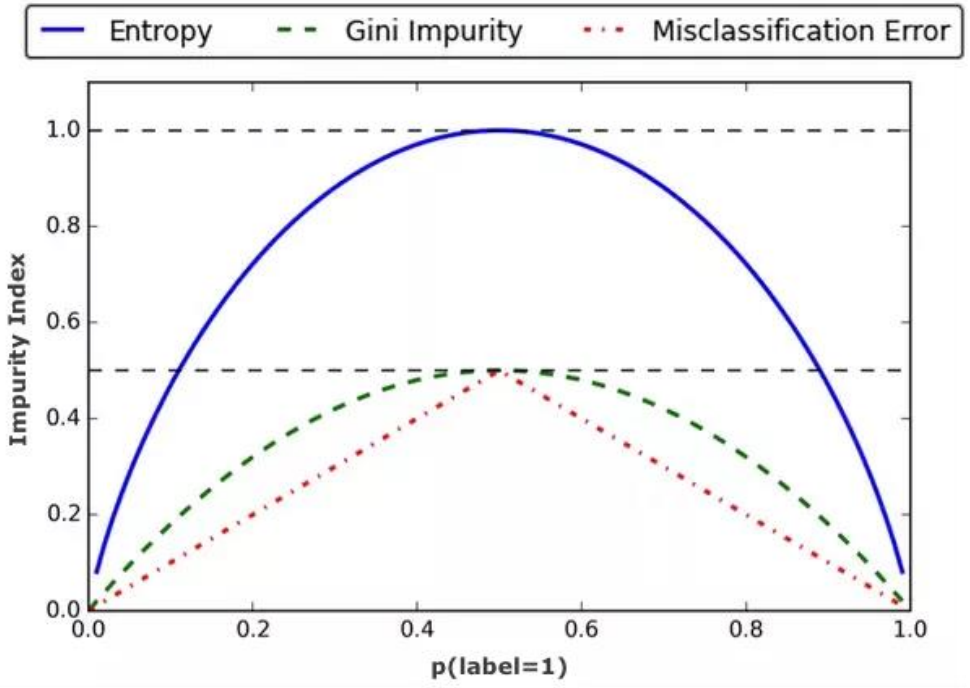


The chosen parameters were:

- The maximum depth of every tree: J = [15, 20, 20, 25, 25]
- The number of estimators: T = [1000, 1000, 500, 1000, 2000]
- The value of the random feature selection: $m = \sqrt{F}$
- The function that measure the quality of the split: Gini Impurity index.

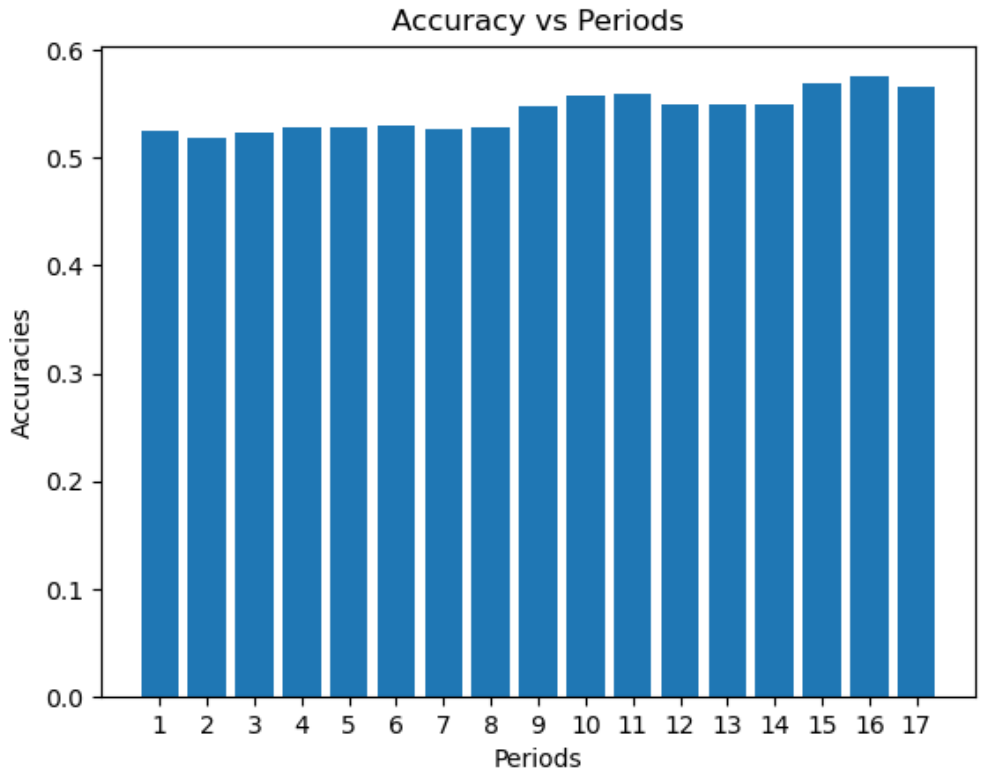
$$Gini(t) = 1 - \sum_{i=1}^{j} P(i|t)^2$$

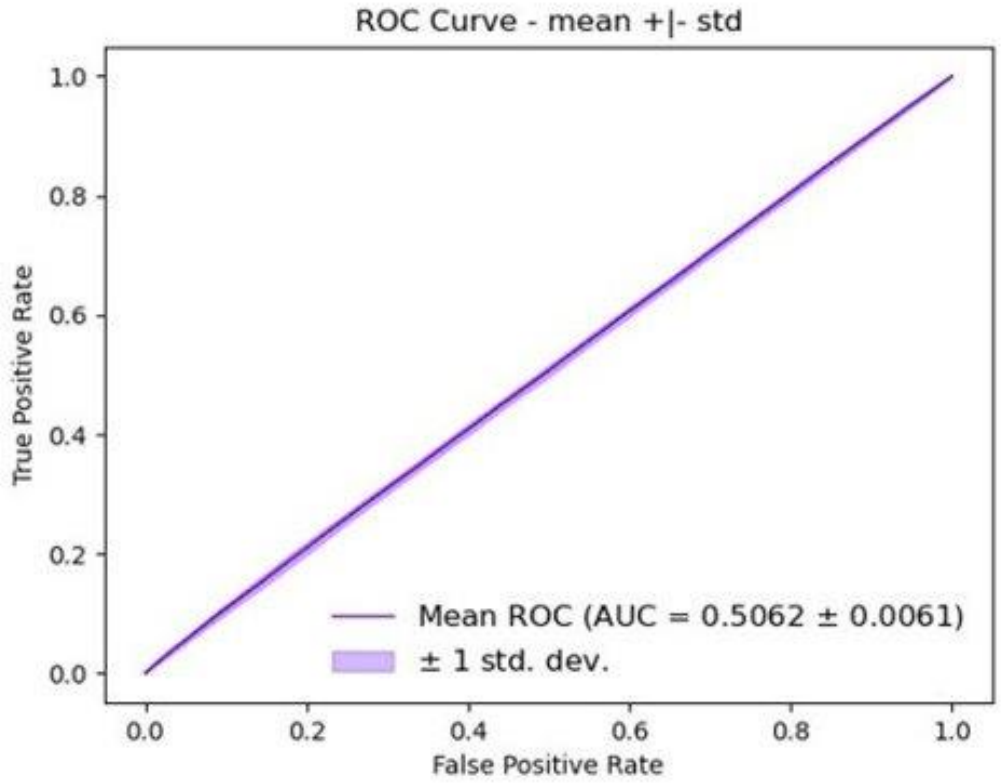
To build a tree, we need to calculate the information gain for every possible split. The best split is the one that provides the most information gain. This process is repeated for each impure node until the tree is complete.



Algorithms – Random Forest

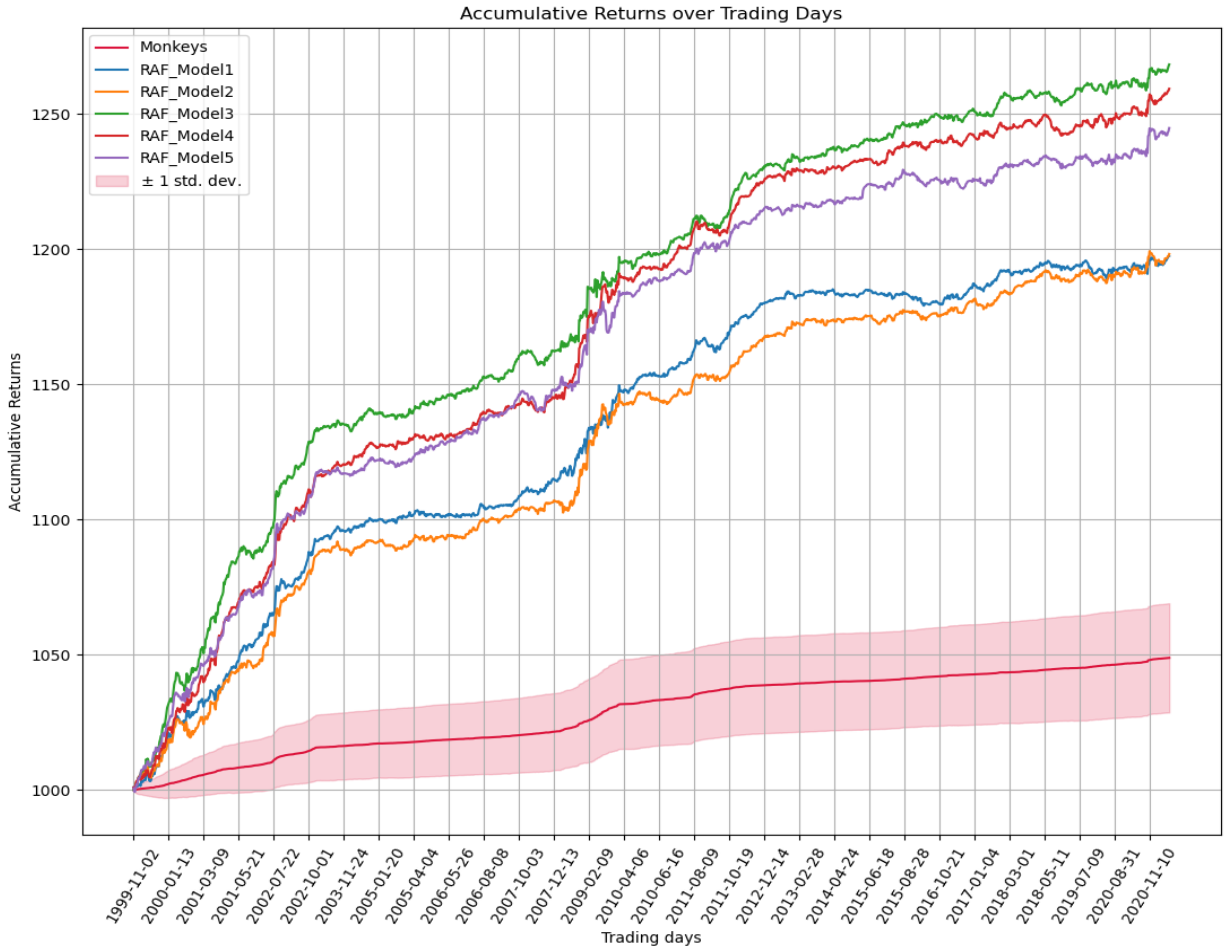
- The best model was fitted over the 17 periods with 1000 trees, a maximum depth of 15, a minimum number of samples required to split an internal node of 2 and a minimum number of samples required to be a leaf node of 1.

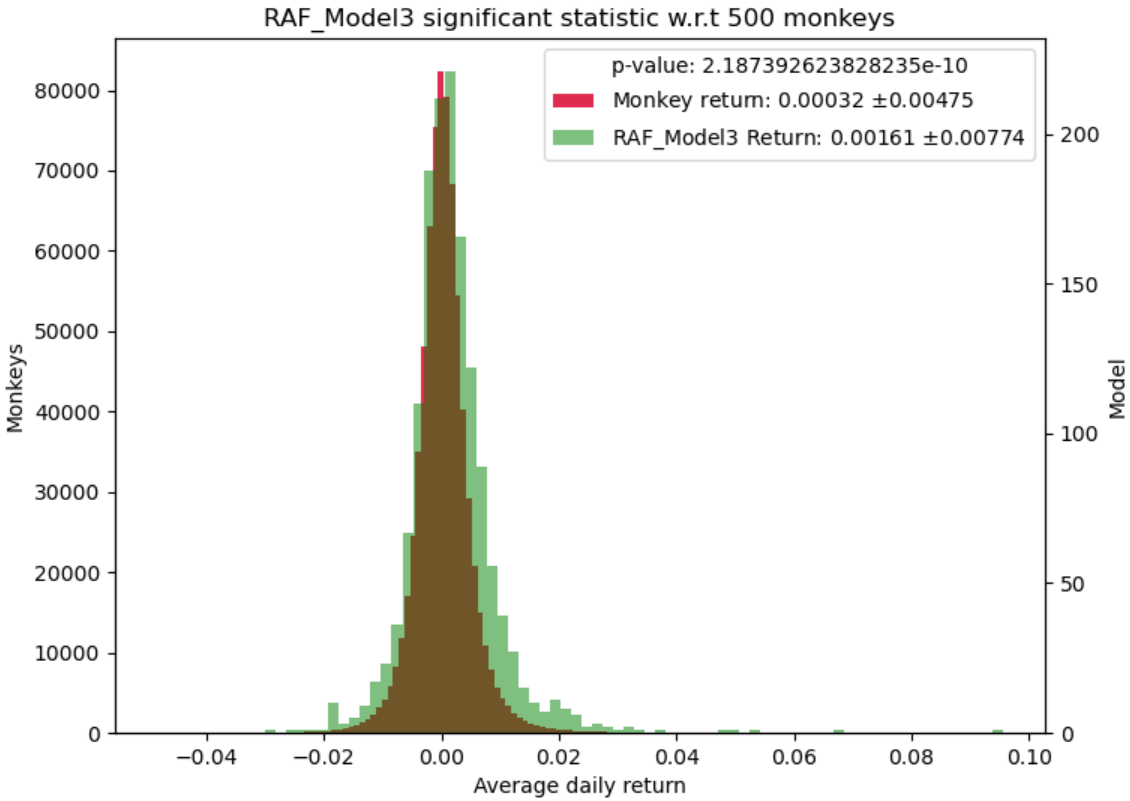




Algorithms – Random Forest

- Trading period consists of the 17 tests sets.
- Starting from 1000$, the model was able to end with ≈ 1250$





Preprocessing – PCA (Principal Component Analysis)

- Even if the results were quite good, we asked ourselves how to handle the data before putting them into the algorithms.

- The purpose is: to decrease computational time and improve the results (expecially in term of AUC).

- Until this moment we dealt with a huge amount of data basically due to the large number of companies (365). The simplest idea in order to decrease computational time is to reduce the data, therefore the number of companies.

- PCA allowed us to reduce the dimension of dataframe keeping most of its variance through a projection onto a less-dimensional space composed by the eigeinvectors of covariat matrix (so-called Principal Components):

1. Kept the first 250 principal components (preserved variance = 0.94)

$$\text{Preserved Variance} = \frac{\lambda_1 + \cdots + \lambda_q}{\lambda_1 + \cdots + \lambda_m}$$

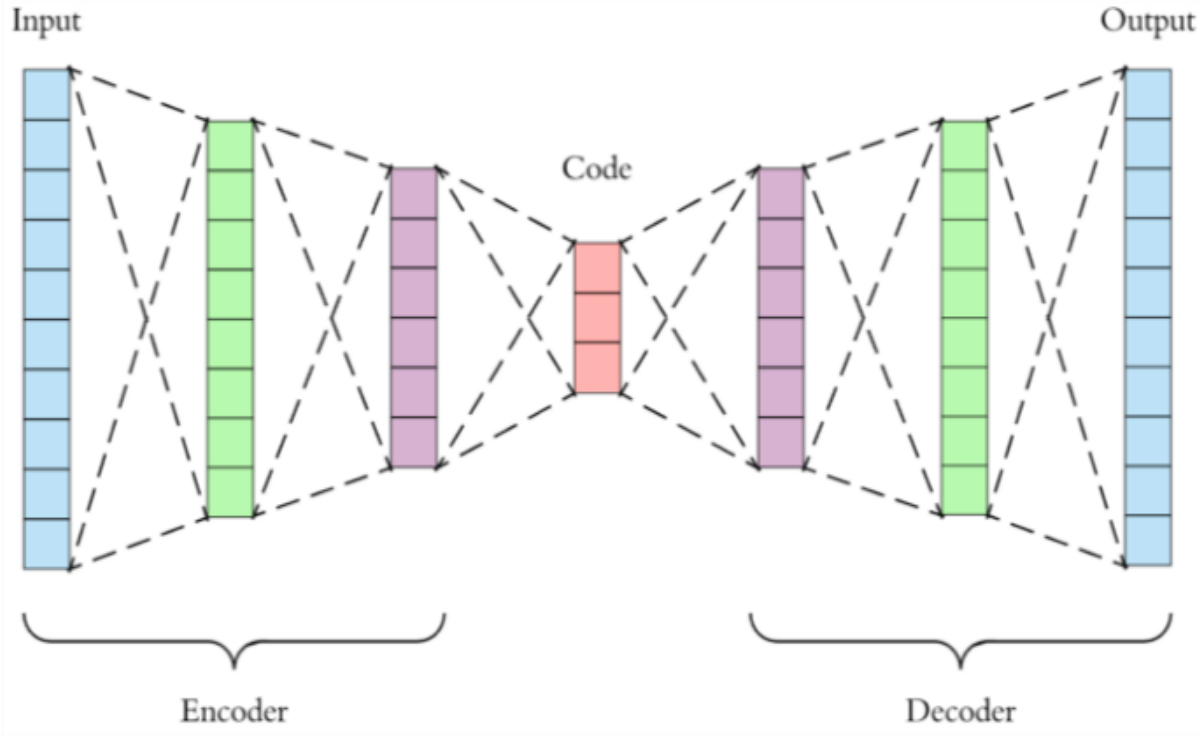
- $\lambda_1 \geq \cdots \geq \lambda_m$ eigenvalues of the covariance matrix.
- q is the dimension of the space onto data is projected.
- m is the original dimension of the space data.

2. Selected the features that most contributed to the principal components: 175 companies.

Preprocessing - Autoencoder

- To improve the results of RAF and DNN, the "by hand" features selection was substituted by an automatic one.
- Autoencoder allowed to reduce the shape of the input keeping most of its "information".

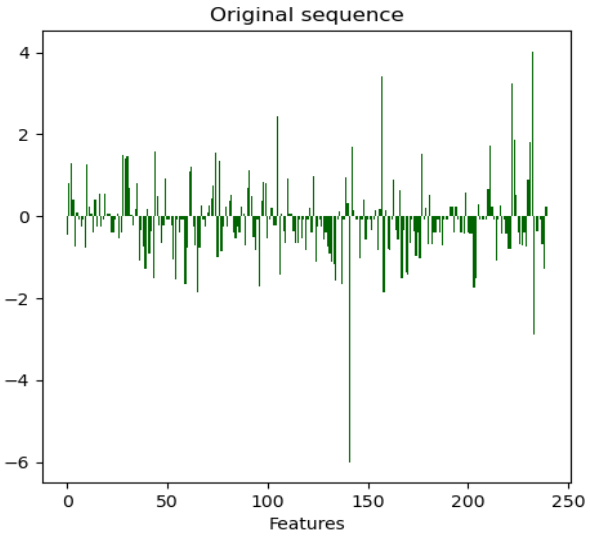
- The input passes through the encoder to produce the code.
- The decoder, which has a similar structure, produces the output only using the code.
- The goal is to get the output as close as possibile to the input and to do so, the input shape and the output shape must be the same.

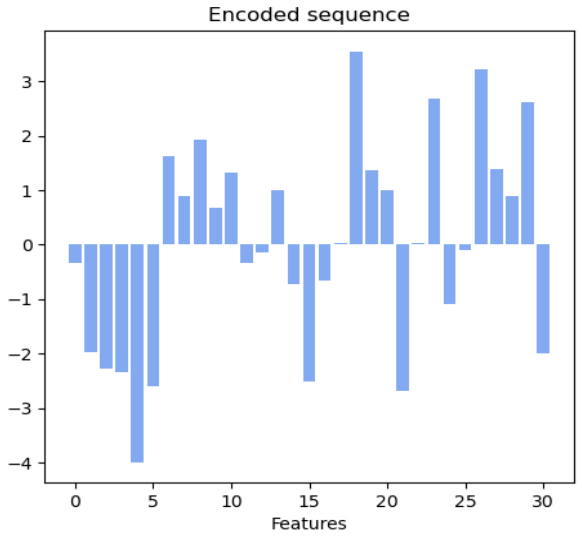


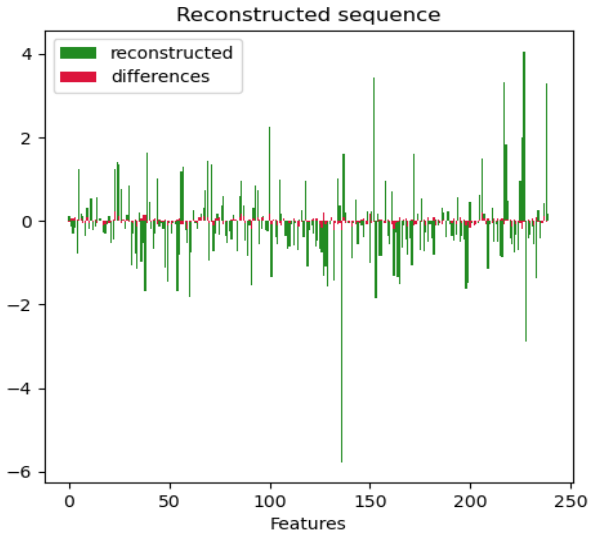
Preprocessing - Autoencoder

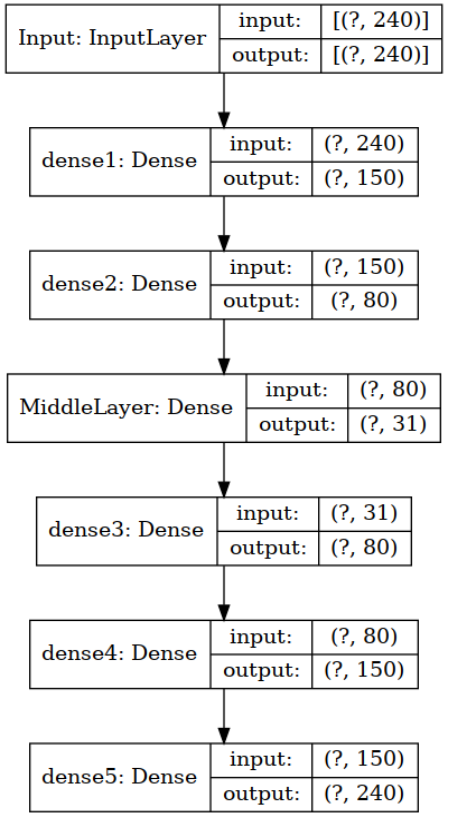
- 2 layers of encoded and decoded with 150 and 80 nodes
- Code nodes in the Middle Layer: 31
- Loss function: Mean Square Error
- Training: 40 epochs

- Computational issue: each sequence of training set (about 127000) and test set (about 15000) passes through the autoencoder and needs about 22 h to complete only one period of train and 2 h for one period of test.





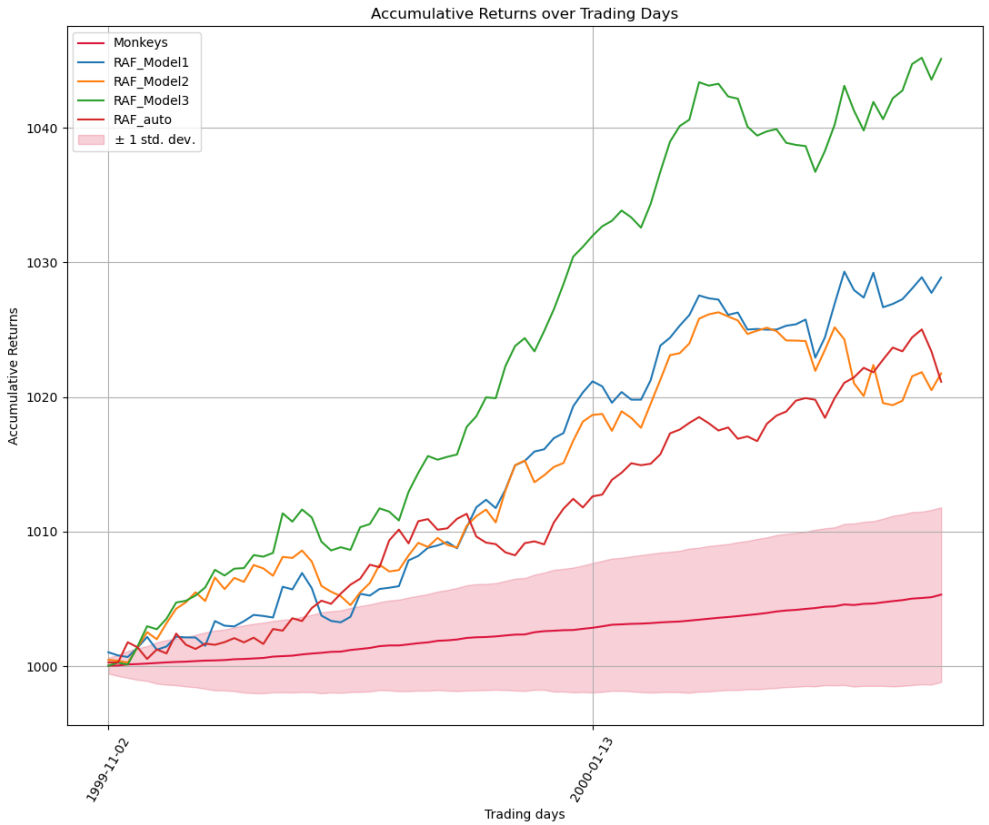


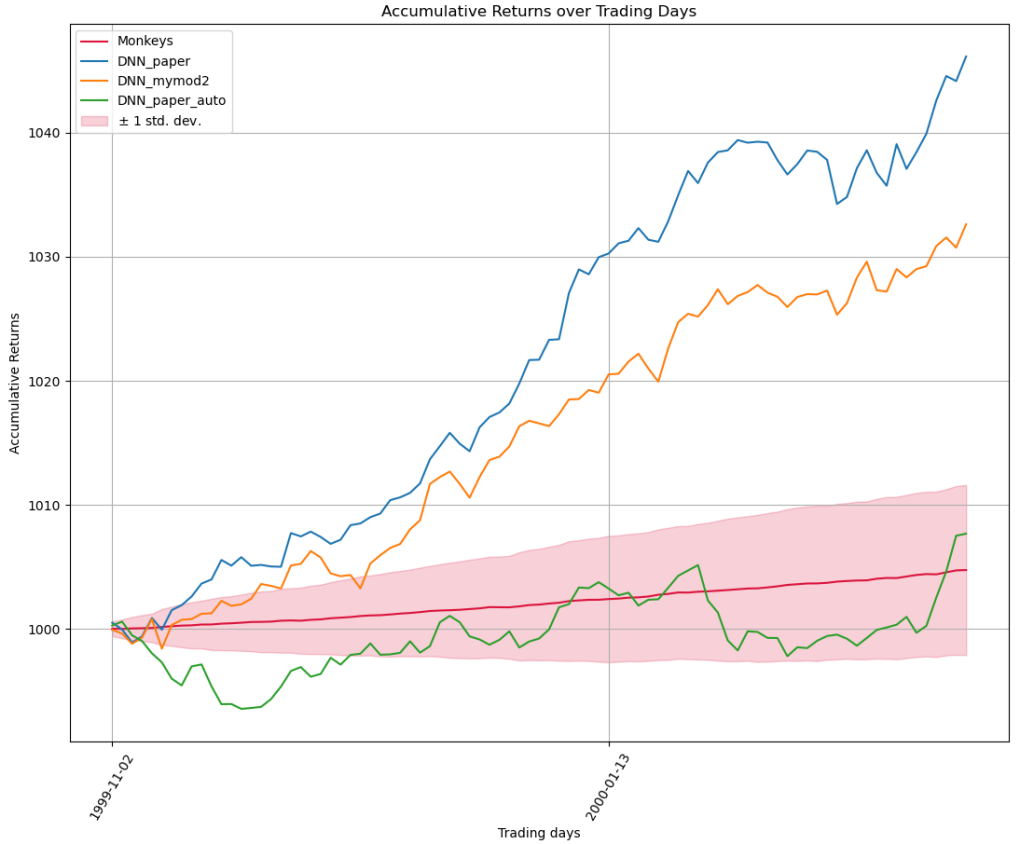


- The differences between reconstructed and original signal are small compared to the original one.

Results – PCA + Autoencoder + RAF/DNN

- The accumulative returns obtained with the autoencoder were worst then the previous models, in particular for DNN in which it totally lies in the randomness region.

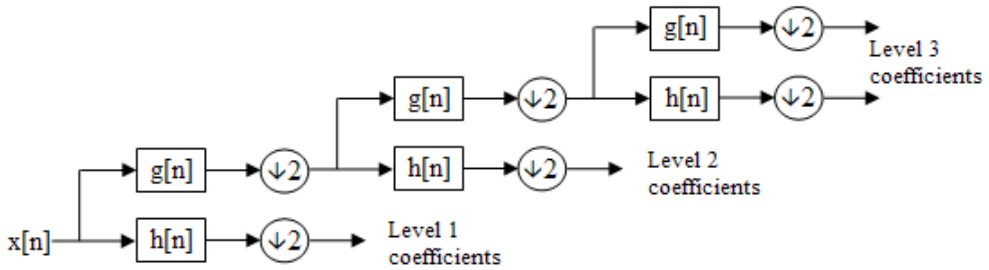




Preprocessing - Wavelet

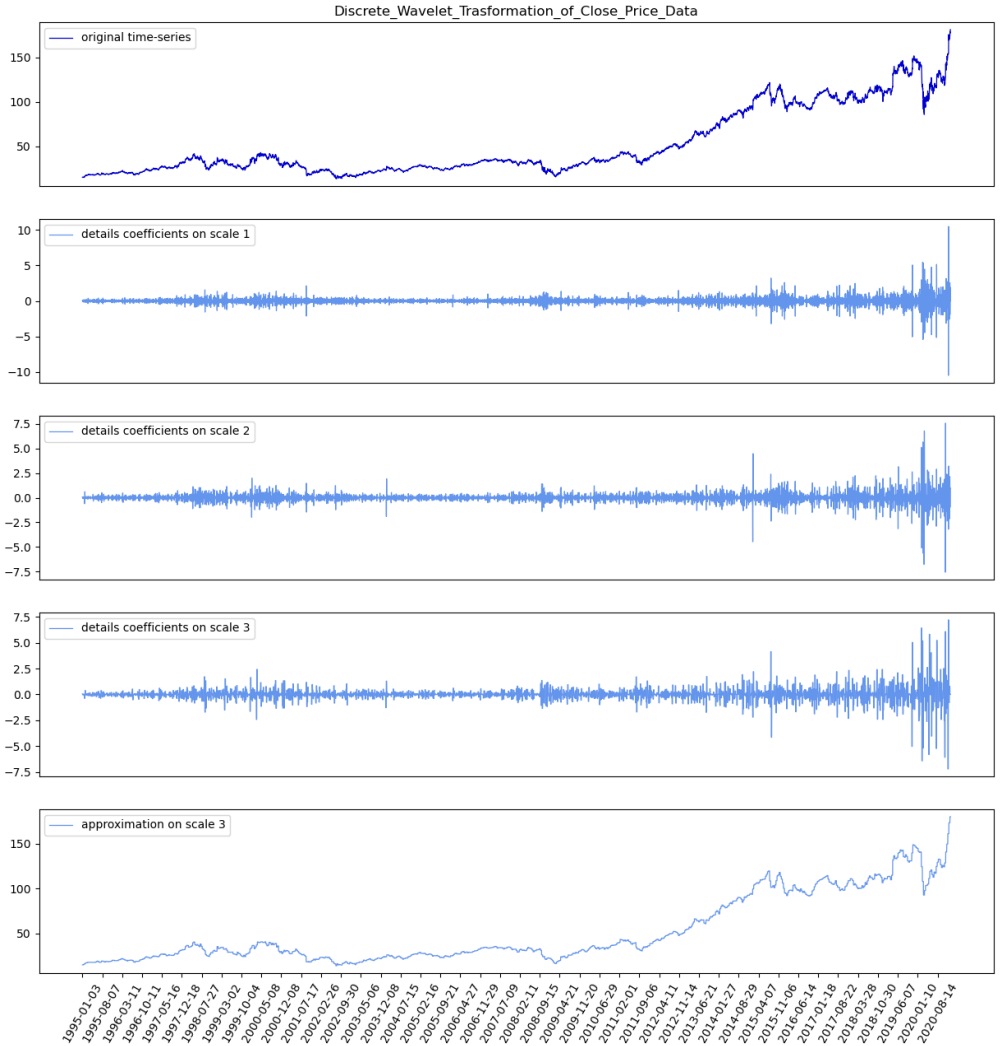
- Discrete Wavelet Transform (DWT) is a kind of time series decomposition that is able to extract information from both time and frequency domain simultaneously at different scales.

- DWT can be thought as an application of a filterbank composed by a low-pass filter and a high-pass filter. Respectively they produces the so-called details coefficients and smooth coefficients



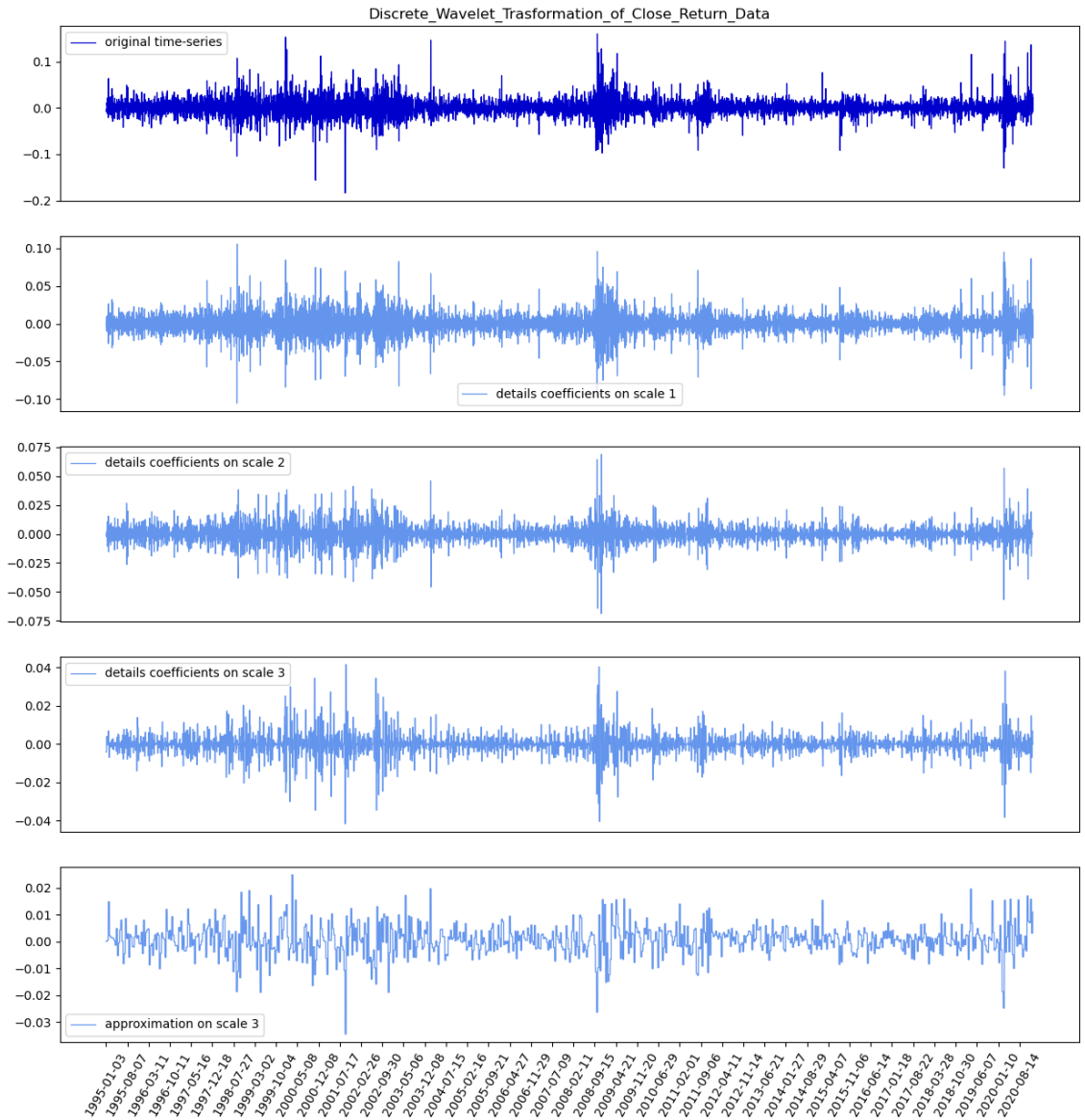
- The signal is therefore decomposed as follow:

$$\mathbf{X} = \sum_{j=1}^{J} D_j + S_J$$



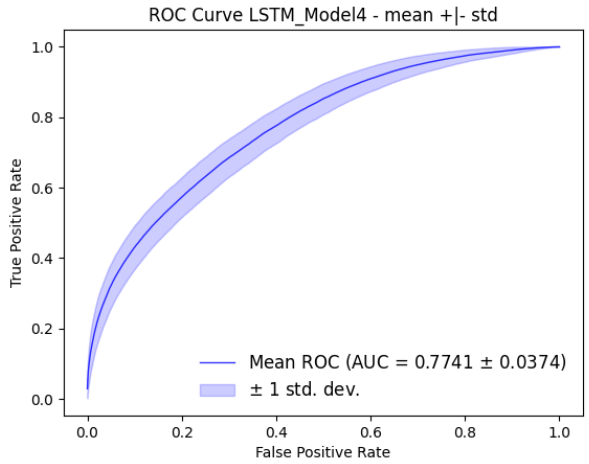
Preprocessing - Wavelet

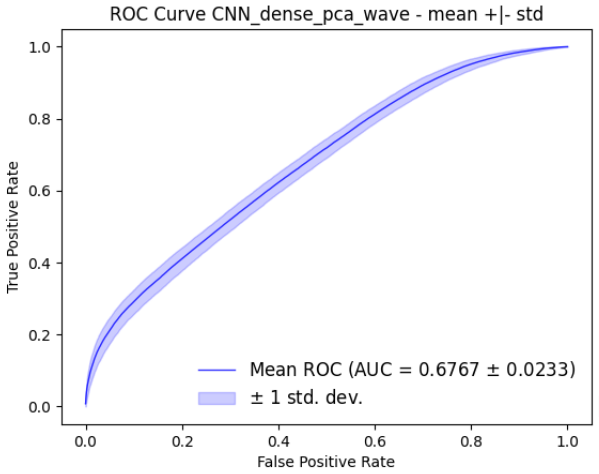
- We applied DWT together with haar function on the returns data.
- This gave us information about differences of values on different time scales.
- Since the objective is to forecast the returns data of the following day, we chose 3 time scales in order to keep only the “short-scale” information.
- We used the first three details coefficients (high frequency information on their scale) and the third approximation (containing all the remaining signal)

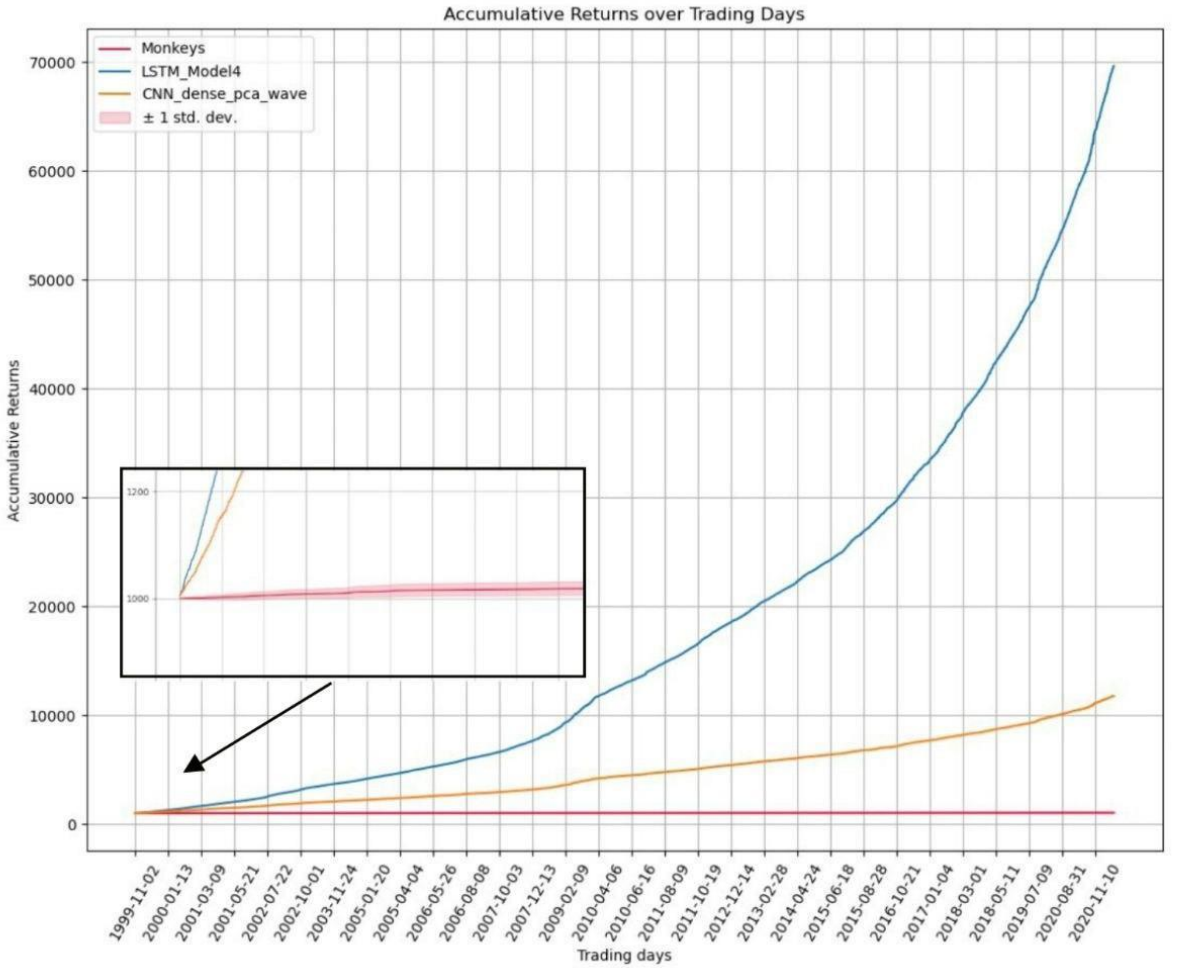


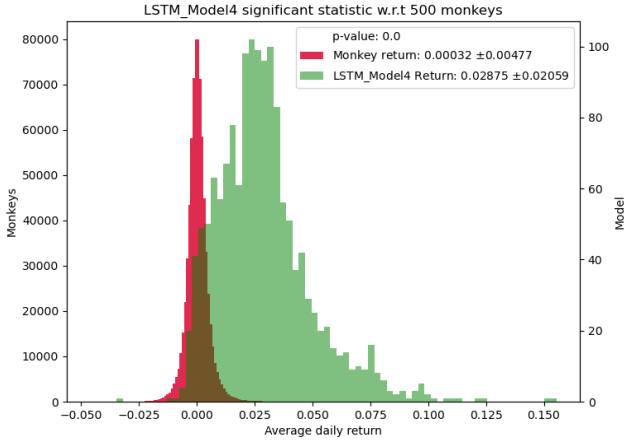
Results – PCA + DWT + LSTM/CNN

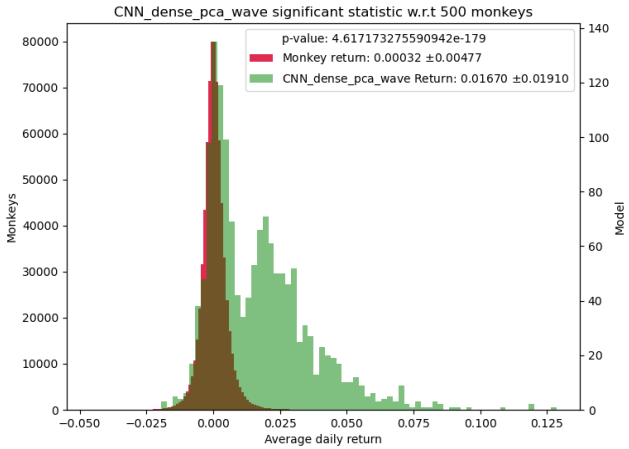
- Since DWT enlarges the dataset fed into the algorithms, the implementation of PCA reduces at the beginning the amount of data but with the advantage of maintaining the most of variance of them.











Bibliography

(1) Omer Berat Sezer, Mehmet Ugur Gudelek, Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning : A systematic literature review: 2005–2019. Applied Soft Computing, Volume 90, 2020, 106181, ISSN 1568-4946, https://doi.org/10.1016/j.asoc.2020.106181

(2) Graves, A., 2014. Generating sequences with recurrent neural networks. arXiv preprint. arXiv:1308.0850.

(3) Thomas Fischer, Christopher Krauss. Deep learning with long short-term memory networks for financial market predictions. European Journal of Operational Research, Volume 270, Issue 2, 2018 Pages 654-669, ISSN 0377-2217, https://doi.org/10.1016/j.ejor.2017.11.054

(4) Christopher Krauss, Xuan Anh Do, Nicolas Huck. Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. European Journal of Operational Research, Volume 259, Issue 2, 2017, Pages 689-702, ISSN 0377-2217, https://doi.org/10.1016/j.ejor.2016.10.031

(5) Jothimani, Dhanya and Yadav, Surendra S. and Shankar, Ravi, Discrete Wavelet Transform-Based Prediction of Stock Index: A Study on National Stock Exchange Fifty Index (2015). Journal of Financial Management and Analysis, Vol. 28(2), 2015, Available at SSRN: https://ssrn.com/abstract=2769529