Appendix F

Code

This appendix explores some practical aspects concerning the code developed in this work, which is publicly accessible in [59]. This project is coded entirely using Python.

F.1 Code organization

The code repository on which this work is based is organized as follows:

- **Classes:**
  - ***class_DataProcessor.py***: contains the required functions to handle the data processing.
  - ***class_ForecastingTrader.py***: contains the required functions to implement the forecasting algorithms as well as the forecasting-based trading model.
  - ***class_SeriesAnalyser.py***: contains the required functions to perform time-series analysis, mainly used for pairs selection.
  - ***class_Trader.py*** - contains functions related with the trading process, namely the standard trading model and the portfolio simulation.
- **Notebooks:**
  - ***PairsTrading-DataPreprocessing.ipynb*** - contains all the steps performed to generate the dataset.
  - ***PairsTrading-Clustering.ipynb*** - contains the code responsible for performing the pairs' clustering according to the three techniques examined in this work.
  - ***PairsTrading-FixedBeta_NoClustering_201x_201x.ipynb*** - contains the simulation results for the standard model when performing no clustering (Research Stage 1).
  - ***PairsTrading-FixedBeta_Sector_201x_201x.ipynb*** - contains the simulation results for the standard model when grouping by sector (Research Stage 1).
  - ***PairsTrading-FixedBeta_OPTICS_201x_201x.ipynb*** - contains the simulation results for the standard model when clustering with OPTICS (Research Stage 1).

- ▷ ***PairsTrading-FixedBeta_20xx_201x.ipynb*** - contains the simulation results for the standard model (Research Stage 2).
- ▷ ***PairsTrading-FixedBeta_Forecasting_2009_2019.ipynb*** - contains the simulation results for the forecasting-based model using ARMA, LSTM and LSTM Encoder-Decoder (Research Stage 2).

- **Data:**
  - ▷ ***commodity_ETFs_long_updated.xlsx*** - contains the tickers for all commodity-linked ETFs available for trading in 2019 (some tickers are repeated).
  - ▷ ***intraday_etfs.xlsx*** - contains only the price series of the commodity-linked ETFs active throughout 2009-2019.
  - ▷ ***intraday_etfs_2014_2019.xlsx*** - contains the additional price series of commodity-linked ETFs active throughout 2014-2019, except the ones already included in the previous file.
  - ▷ **Pickle files:**
    - ▷ ***20xx-201x*** - contain the list of pairs selected in that period. There is a *pikle* file for each clustering technique applied.
    - ▷ ***commodity_ETFs_xxx*** - auxiliary files used for processing the dataset.
    - ▷ ***ticker_catgory_dict*** - dictionary containing the proposed mapping for each ETF category.
    - ▷ ***ticker_segment_dict*** - dictionary containing the proposed mapping for each ETF segment.
- **Training** - this folder contains the python files to run on the Google server, for training the Deep Learning architectures.
- **Forecasting models** - this folder contains the *.pikle* files with the predictions for all the configurations experimented (output collected from the training files).
- **Drafts** - this folder contains a set of code which was implemented as the work progressed but was not used in the final version. Although less documented, several functions may be useful for extending this work in other directions.

Note: The data folder on GitHub contains the link to the files, as they are to large for being uploaded directly on GitHub.

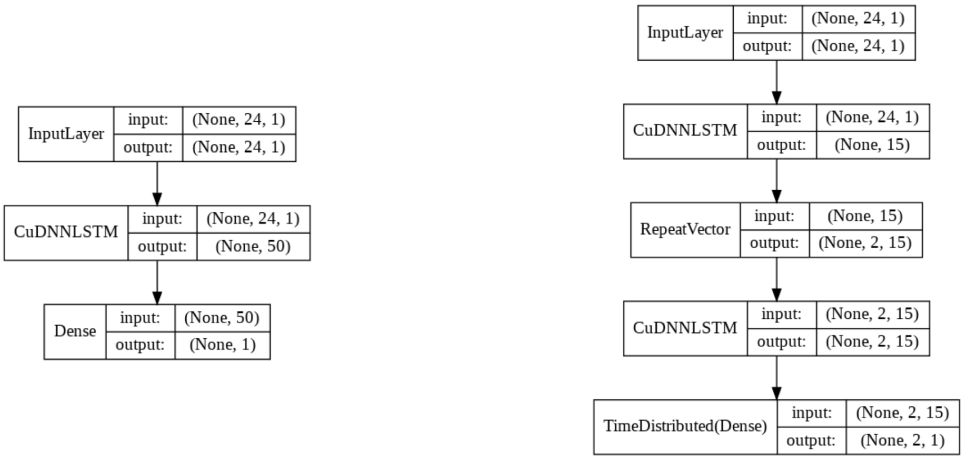
F.2 Running the code

All the results presented in this document are already displayed in the corresponding notebooks' output. Nevertheless, if there is an interest in re-running the notebooks with different configurations, it should be ensured that:

1. The class files are in the same directory as the notebooks.
2. The pickle files are downloaded and inserted in the corresponding data directory.

F.3 Deep Learning models graphs

The *Keras* library employed in this work, for building the Artificial Neural Network models, provides a feature to describe the graph corresponding to a given architecture. Figure F.1(a) illustrates the graph corresponding to the LSTM implementation, and Figure F.1(b) represents the Encoder-Decoder's graph.



(a) LSTM graph.

(b) LSTM Encoder-Decoder graph.

Figure F.1: Graphs of the neural network models with the best forecasting-score, produced with Keras.