Exploring Feedforward and Long Short Term Memory Neural Networks to Stock Markets Forecasting

João Sintra

Abstract—Artificial Neural Networks as a part of Machine Learning have been gaining *momentum* in the context of financial prediction. Deep Neural Networks developments allow an increasing level of abstraction over the problem's application, without loss of effectiveness and allowing for state-of-the-art performances. This thesis has a framework for creating Deep Neural Networks models applied to financial forecasting, using Feedforward Neural Networks and Long Short-Term Neural Networks. The framework is configured with parameters that define all aspects, ranging from the data inputs, data pre-processing, model training and backtesting, producing a predicting model. This model is used in association with a defined strategy with the results presented by a backtest and compared against a Buy and Hold strategy. The results show that it is possible to create simple profitable strategies from securities with different behaviors by incorporating past data and technical indicators along with the price data. The semi-automated framework to train models and apply an investment strategy is able to beat the benchmark by 20% in the selected model and able to generate a model that reaches a cumulative return in the end of the period of 40% for the IBM security. For the AAPL security, an accuracy of 66% for the selected model is registered. The FLKS shows consistent high returns of 200% with less opened positions.

Index Terms—Finance, Deep Learning, Machine Learning, LSTM, Feedforward.

1 Introduction

In the last years, Computer Finance research has seen big growth, motivated by the growth of financial markets, bigger access to information and also because financial prediction problems have a great practical and theoretical interest, while still being extremely difficult to get a good solution for them [Heaton et al., 2016].

Financial markets study and literacy is an interesting research topic because these markets are a possible revenue stream for financial institutions, private companies and even private households. As [Hsiao and Tsai, 2018] points out, for the latter, it is particularly interesting because it can give rise to increased social welfare through the transactions between individuals.

Financial markets encompass a series of different instruments, namely Capital Markets, Commodity Markets, etc - paper thesis will be more focused in a subcategory of the Capital Markets, the Stock Markets.

The prediction of Stock Markets movements is considered to be a very complex task and most models are postulated given the assumptions of the Random Walk Theory and the Efficient Market Hypothesis [Lee et al., 2007] [Attigeri et al., 2015]. The Random Walk Theory applied to fto financial markets, it states that past information of a stock cannot be used to predict it's future behaviour. According to [Malkiel, 2003], the logic of the random walk is that the information flow is even across the market players and the occurrence of new information is random an so it is not possible to predict stock price change. The Efficient Market Hypothesis [Fama, 1970] is deeply connected with the random walk theory. The stock market landscape is, at any moment, a mirror of all the available public information. Every time a new piece of information is made public, the market enters an unbalanced state, assimilating the new information that is translated into the price, restoring the balance.

In recent years, a number of evidences point out to the ability of Machine Learning techniques to capture complex market behaviour and translate into successful predictions [Cavalcante et al., 2016], [Attigeri et al., 2015], [Nelson et al., 2017]. In Machine Learning, Deep Learning techniques extended the previous known capabilities of Artificial Neural Networks and allied to it the capability to extract useful representations from data without extreme processing contributed to the renewed interest in it [Liu et al., 2017].

In this paper, a framework for the creation of Deep Neural Networks applied to financial instruments is described. This framework accounts for all the steps needed from the data pre-processing, data transformation, model training and backtesting using simple investment strategies.

2 Related Work

Substantial literature can be found on the application of machine learning to financial markets price prediction. In it, several methods can be distinguished that range from supervised learning to unsupervised methods.

In the supervised methods are included more simple regression techniques [Siew and Nordin, 2012], support vector machines [Kewat et al., 2017] [Fischer and Krauss, 2017], random forests [Krauss et al., 2016] and neural networks. In the neural networks can be distinguished the use of feedforward neural networks in [Krauss et al., 2016] [Fischer and Krauss, 2017], long-short term memory neural networks [Arévalo et al., 2016] [Li and Tam, 2017] [Fischer and Krauss, 2017]

In the literature, two main approaches can be distinguished to the market. The first relies in the use of fundamental analysis [Siew and Nordin, 2012], the second focus on the technical analysis [Kewat et al., 2017] [Fischer and Krauss, 2017] [Krauss et al., 2016] [Li and Tam, 2017] . The literature focus more on the use of technical indicators. An explanation can be the convenience in the creation of these indicators by using the the open, high, low and close prices. The fundamental indicators, because are related with the financial situation of a company, that is not readily obtained neither has the same data granularity as the stock prices have- the fundamental indicators make use of data that is made available in a plurianual fashion. There are even some companies that do not make public available this information.

The assessment of the methods performance is done in several ways. In [Siew and Nordin, 2012] [Li et al., 2007] [Arévalo et al., 2016] it's measured how well the fit of the values is done. This is measured with in several ways like the Root Mean Square (RMSE), Mean Absolute Error (MAE) and Mean Squared Error (MSE), that measure the difference between the predicted values and the real values.

In [Kewat et al., 2017] [Arévalo et al., 2016] [Li and Tam, 2017] the performance is evaluated in term of directional accuracy and by developed specific performance metrics derived from the accuracy.

The last method is by performing backtesting of a developed strategy over the predictive model [Krauss et al., 2016] [Li and Tam, 2017]. Backtesting is used to simulate a real market and allows to see how the model would've truly performed. In the backtesting, several metrics like the returns, sharpe ratio and annualized returns are developed.

In terms of the object of these studies, they focus on different financial instruments. In [Siew and Nordin, 2012] the focus in on stocks from the Bursa Malasia in a period between 2003 and 2011, in [Kewat et al., 2017] is on a South Korea index, in [Li et al., 2007] focus on two indexes, the S&P500 and the Nasdaq Composite and also on a company stock IBM.

These studies focus on different financial instruments like companies stock and indexes. Some focus on constituents of the indexes like [Siew and Nordin, 2012] [Krauss et al., 2016] [Li and Tam, 2017] , other focus only on the indexes [Kewat et al., 2017] and others focus on both indexes and a single company to compare results [Li et al., 2007]. Also [Arévalo et al., 2016] focus only on a company stock.

In table 1, a synthesis of the selected studies is made.

3 Implementation

3.1 Architecture

A focus on the forecast of a financial instrument in the form of the stock price or the stock return is taken. The input can be raw information like the Opening or Closing price of a given stock or a calculated return. Additionally, a group of indicators can also be selected to be calculated to further enrich the input information. After a prediction is made, an investment strategy is applied and the a backtest is performed. The overall solution is divided in two main phases, Prediction and Investment Strategy Backtest.

To achieve this goal, a partially automated pipeline was developed, where given a set of inputs, a number of stages is successively applied, yielding a prediction model and then using it to input data to investment strategies. The simplified architecture of the solution is showed in figure 1. As can be seen, five modules were developed, the *Runner*, *DataManager*, *DataAnalysis*, *Models* and the *Backtest*.

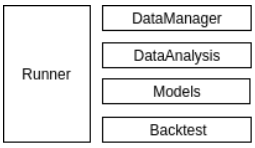

Fig. 1: Architecture Blocks

The ***Runner*** module is the controller module, receiving the input data of the stock and the information for executing the run.

The ***DataManager*** module is responsible for accessing the inputs and transform them into a machine-readable format.

The ***DataAnalysis*** in this module where the input stock data is prepared to be fed to the models.

3.1.0.1 Example: The objective is to predict the opening (Open) price of the stock "X" and the features are the variables Open, High, Low prices for each timestep. We can choose to use only the value from the immediate previous timestep $t-1$ or values from n previous timesteps- this number of days is the called backoff. The dataset that would be created from the input variables would be, each of the cases, represented in equation 1 and equation 2, respectively.

Ref	Year	Method	Data	Period	Financial Instrument	Algorithm Performance
[Siew and Nordin, 2012]	2012	SMO Regression	Stock Prices (daily)	2003-2010	13 Bursa Malasia Companies	0.6079 (R^2)
[Kewat et al., 2017]	2017	SVM and ANN	Index prices (daily)	2007-2016	KOSPI	0.57 (SVM) and 0.54 (ANN) (Prediction Performance)
[Li et al., 2007]	2007	RL and ANN	Index prices (daily), Stock Prices (daily)	1984-2004	S&P500, NASDAQ Composite, IBM	60,90% (S&P500), 63.4%(NASDAQ), 60,2%(IBM) (DA)
[Arévalo et al., 2016]	2016	DNN	Stock Prices (by minute)	September 2nd to November 7th 2008	AAPL Stock	0.086 (MSE), 64,8% (DA)
[Krauss et al., 2016]	2016	DNN	Index prices (daily)	1989-20015	S&P500	0.33% (Return)
[Li and Tam, 2017]	2017	DNN and SVM with PCA	Index prices (daily)	2012-2015	Shangai Stock Exchange	51.59% (SVM), 50,01% (DNN), 51.43% (SVM-PCA), 50.43% (DNN-PCA) (Accuracy)

TABLE 1: Relevant studies results

$$Open(t-1), High(t-1), Low(t-1) \longrightarrow Open(t+1) \quad (1)$$

$$Open(t-n), High(t-n), Low(t-n), ..., Open(t-1), High(t-1), Low(t-1) \longrightarrow Open(t+1) \quad (2)$$

The ***Models*** module receives the datasets required to train a network and, after training, make a prediction. Two types of predictions are developed, one using raw price data and the other using simple price returns. The simple return is calculated as showed in equation 3.

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}} \quad (3)$$

The final module is the ***Backtest*** is used to make a backtesting of the prediction and used investment strategy.

3.2 Pipeline

The term *Pipeline*, when applied to computing, refers to the ability to develop and run a set of instructions is a cascaded and structured way, with several elements connected in series. This elements are the different referred modules.

The execution of the implementation has several cascaded steps and has the objective of predicting a given price of a selected stock by the predictions of a model and assessing the prediction performance and associated returns.

3.2.1 Input Data

There are two kinds of input data: stock market data regarding a ticker - Stock Data - and Control Data, used to define how the runner will proceed.

The **Stock Data** is taken from the website *YahooFinance*, with daily frequency and date intervals selected corresponding to the entire available time span. An example of the data is shown in table 2.

The **Control Data** is a set of user defined input parameters that configure the pipeline execution. This inputs are received in a YAML file - YAML stands for YAML Ain't Markup Language. YAML is a human readable serialized data representation that is used in services configuration like the Google App Engine Cloud. The user places the configurations of the *run* it wants to perform in this file that will be copied to the correspondent *run* directory and used in the several *run* stages. An example of a YAML configuration file is showed in listing 1, only showing the configuration for a Feedforward model use.

3.2.2 Data Preparation

The data preparation or pre-processing happens in the *DataAnalysis* module and turns it into a more meaningful representation that can better assimilated by the neural networks. This step is also used to enrich the stock data with indicators.

Depending on the user input on the YAML, a data normalization technique or a data standardization technique on the features but never on the target. The intuition behind using data normalization techniques is that transforming data to a well defined range or

Date	Open	High	Low	Close	Adj Close	Volume
2000-03-01	4.234375	4.716518	4.232143	4.654018	3.150568	269250800
2000-03-02	4.535714	4.569196	4.310268	4.357143	2.949597	77814800
2000-03-03	4.459821	4.579796	4.285714	4.571429	3.094658	80841600
2000-03-06	4.500000	4.611607	4.464286	4.488839	3.038749	52640000

TABLE 2: Example Stock Market Data Regarding the AAPL ticker

at least a more limited one leads to better and faster training of the neural networks.

```yaml
DataAnalysis: #DataAnalysis commands
  ticker: "IBM"
  train_test_ratio: 0.8
  backoff_days: 15
  indicators:
    - "momentum"
    - "volume"
    - "volatility"
  normalization: "minmax"
  dimensional_reduction: "pca"
  features:
      - "Open"
      - "High"
      - "Low"
  target: "Open"
Model:  #Model commands
  type: "lstm"
  feedforward: #Feedforward Model
    Architecture:
      input_layer:
          activation: 'relu'
      hidden_layer_1:
          neurons: 100
          activation: 'relu'
      hidden_layer_2:
          neurons: 50
          activation: 'relu'
      output_layer:
        neurons: 1
        activation: 'relu'
    Compilation:
      loss: "mean_squared_error"
    Train:
      epochs: 300
   ...
Backtest:# Backtest commands
  portfolio_start: 1
  buy_window: 1
```

Listing 1: YAML configuring the Run

Depending on the activation function of a neuron, it may be helpful to perform some kind of pretreatment - if a logistic sigmoid is used, for example, the variation of the variables has to be adapted to the function maximum output range that is between 0 and 1 [Sola and Sevilla, 1997].

Convergence can be faster when the average of each input variable is closer to 0. Scaling the inputs can be also beneficial when it is known that some inputs are of less significance than others and so can be useful to scale down the inputs in other to make them less visible to the training process [LeCun et al., 2012]. The Min-Max normalization technique, also known as feature scaling, is used and it's given by equation 4, where X' is the result of the normalization, X is the input value, X_{min} is the minimum value of the series of values and X_{max} is the maximum value of the series. The result of this normalization is a normalized value between $[0, 1]$.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{4}$$

3.2.3 Models

After the data is prepared, it is fed to the models. The models were developed using the Keras framework for it's convenience and flexibility to implement in the framework. The chosen models to make predictions over a given stock developed are the Feedforward Neural Network and the LSTM Neural Network. Each Neural Network has a number of common hyper-parameters, parameters that define it's architecture, components and behaviour. For each model, several hyperparameters are chosen and fixed, while others have series of values that are used in several runs. In these NN parameters are included the number of layers, number of neurons per layer, number of training epochs, among others.

The architecture of the Neural Networks it's a sequence of layers, exemplified in figure 2.

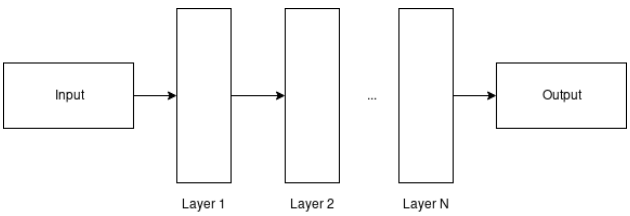

Fig. 2: Deep neural network architecture

In the case of the Feedforward Neural Networks, a more detailed architecture is showed in figure 3.

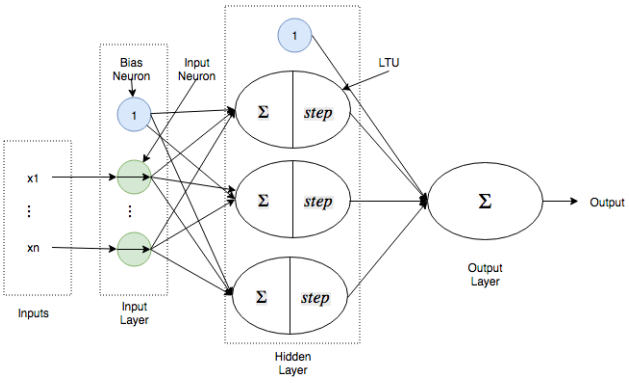

Fig. 3: Multi-layer perceptron representation

The LSTM Neural Networks are a subset of Recurrent Neural Networks (RNN), where, at each timestep, the information of previous states is used to adjust the weights in combination with the timestep features.

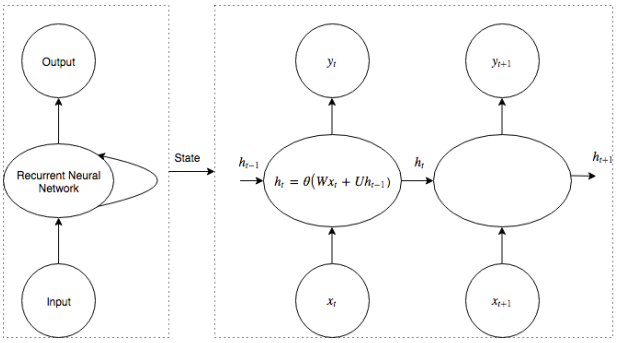

Fig. 4: Recurrent Neural Network representation

To evaluate the obtained models, a number of metrics are developed. There are two groups of metrics, the ones that focus on the prediction capability of the model and the fit of the data - Root Mean Squared Error (RMSE), Mean Absolute Error and R-squared - and the ones that focus on the results of the investment strategies based in the model. From the first group, only the RMSE is detailed in the results.

The ***Backtesting*** module uses the prediction made by the model as input to a simple strategy generating a vector with a given action for each prediction. Depending on the type of prediction that is made, two strategies are considered:

- *PriceStrat*- This strategy, in equation 5, is applied when the prediction is made for the stock price and it relies only on the prediction of the model. This can be interpreted as a directional prediction.
- *RetStrat*- this strategy looks at the predicted return and to the actual return and it's given by equation 6 where $PredReturn(t)$ is the predicted return at instant t and $RealRet(t-1)$ is the real return at $t-1$.

A final strategy is used, the Buy and Hold, where a position is taken by the purchase of a security, in this case, at the first prediction timestep. The security is sold in the last prediction timestep and the profit arises from the valuation of the stock through time (or vice-versa). This strategy is used as a benchmark.

4 Experiments and Results

Three case studies were developed, focused on stocks with different characteristics behaviour:

1) International Business Machines (IBM)
2) Apple Inc. (AAPL)
3) Flex Pharma (FLSK)

The search for the best model for a given security was made by performing *runs* with different values for some hyperparameters and using the results to make a choice over which the networks to use. The objective is, for each stock, to find the model for predicting a stock price or return that gives the biggest profit.

For each possible configuration, five runs are conducted and the results are averaged.

$$\text{Action(t)}=\begin{cases}1, & \text{if } round(Predicted(t-1)) < round(Predicted(t)),\\ -1, & \text{if } round(Predicted(t-1)) > round(Predicted(t)),\\ 0, & \text{otherwise.}\end{cases} \tag{5}$$

$$Action(t) = \begin{cases}1, & \text{if } round(\frac{PredReturn(t)}{RealRet(t-1)} - 1) * 100) > 1,\\ -1, & \text{if } round(\frac{PredReturn(t)}{RealRet(t-1)} - 1) * 100) < 1,\\ 0, & \text{otherwise.}\end{cases} \tag{6}$$

For a given stock, two architectures are chosen, each one per model developed. This choice is done by looking both to the training RMSE and test accuracy and picking the one that yields best results. From these two selected architectures, the one with best training metrics is chosen and compared with the architecture that best performed in the out-of-sample data. The *selected* model is the one that would be selected in a semi-automatic fashion; the *best performing* model is the model obtained that would've performed the best, although it may not be the one chosed.

4.1 Case Study I - IBM

The train data starts in $2000-01-03$ and goes to $2012-11-30$ and the test data starts at $2012-12-03$ and ends in $2018-06-12$. This stock has a sideways behaviour in most the training dataset with a uptrend in the end. In the testing data, the stock exhibits mostly a down trend with the stock ending loosing almost half of it's price. This two factors makes this a curious study, because of the dicotomy in the behaviour for which the network is trained and on which the network is tested.

4.1.1 Hyperparameters Search for Returns Prediction

The results of the search for hyperparameters are shown in table 3. To help define the performance of the trained networks, it is chosen, for each value of backoff days, the best performing network by the obtained training metrics.

Model	Days	Architecture	RMSE		Accuracy	
			Train	Test	Train	Test
FF	5	(10,5)	1,22	1,75	0,62	0,6
	10	(500,100)	1,79	1,23	0,55	0,57
	20	(50,20)	1,81	1,25	0,57	0,58
LSTM	5	(500,100)	2,12	1,53	0,55	0,51
	10	(10,5)	1,74	1,22	0,57	0,56
	20	(10,5)	2,82	2,15	0,65	0,51

TABLE 3: Architectures with highest accuracy for the IBM stock Returns Prediction

In figure 5 the cumulative return for the selected FeedForward models is shown and in figure 6, the cumulative results for the LSTM Network selected models. In both figures the architectures are named in the fashion AX-(Y,Z), where A is the model type (F for Feedforward and L for LSTM), X is the number of backoff days, Y is the number of neurons in the first layer and Z is the number of neurons in the second layer. Also, in both figures, the Benchmark for comparison is showed. At this stage, considering the architectures in table 3, two models are chosen for analysis. By analysis of table 3, the model to be selected is the L20-(10,5). The second model to be chosen is called the "best performing" from now on, and it is chosen based on the test metrics and only for comparison with the selected one. In this case, it's the F20-(50,20).

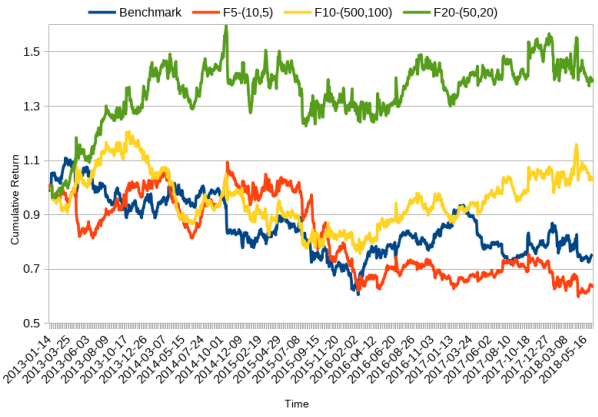

Fig. 5: Cumulative returns for Feedforward NN model selected architectures

The performance of these models is showed in figure 7, where it can be seen that the performance of the selected model is not the as good as the performance of the (rightly named) best performing model. This can be explained by looking at the accuracy values in the test, with the selected model having a value of 51% and the best performing method having a value of 58%. Nevertheless, the selected model ends up beating the benchmark.

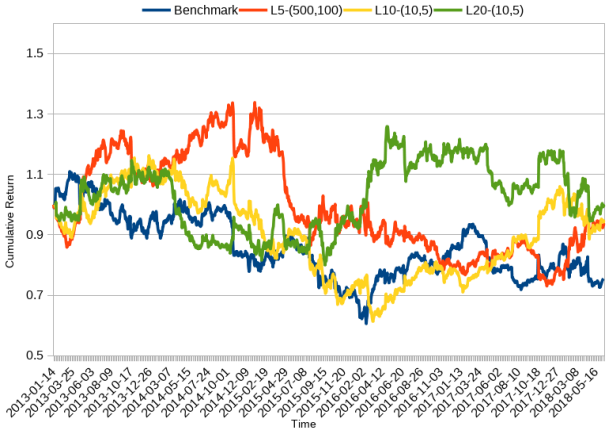

Fig. 6: Cumulative returns for LSTM NN model selected architectures

4.1.2 Hyperparameters Search for Price Prediction

The same search is conducted to a model that predicts the next day price. The resulting models are shown in table 4, with the obtained metrics in training and testing. The performance of the models in the backtest is shown by the cumulative returns obtained in figure 8.

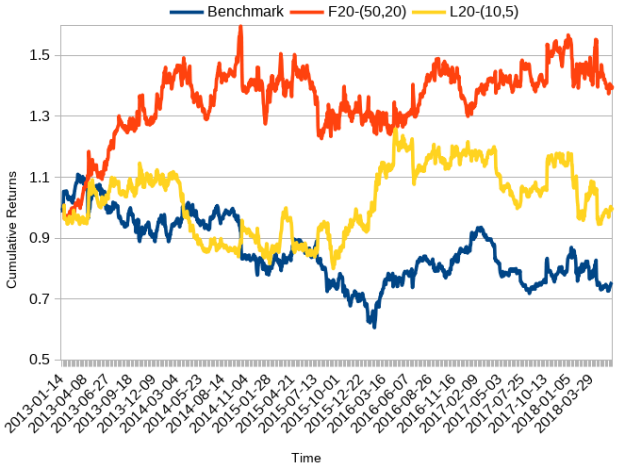

Fig. 7: Cumulative returns for the L20-(10,5) (selected) model and F20-(50,20) (best performing) model - IBM Returns Prediction

By comparison with table 3 it can be seen that the values for the accuracy in both models are considerably lower. Also, by inspection of figure 8, it can be seen that the performance of both models is far off from the performance obtained in the returns prediction. Nevertheless, both models beat the benchmark, although the same problem arises - the network with the best training metrics is not the best performing

	Model	Days	Architecture	RMSE		Accuracy	
				Train	Test	Train	Test
Selected	LSTM	5	(100,50)	24.16	73.12	0.51	0.5
Best	LSTM	20	(50,20)	70.44	150.09	0.5	0.52

TABLE 4: Selected Model vs Best Model - IBM Price Prediction

one.

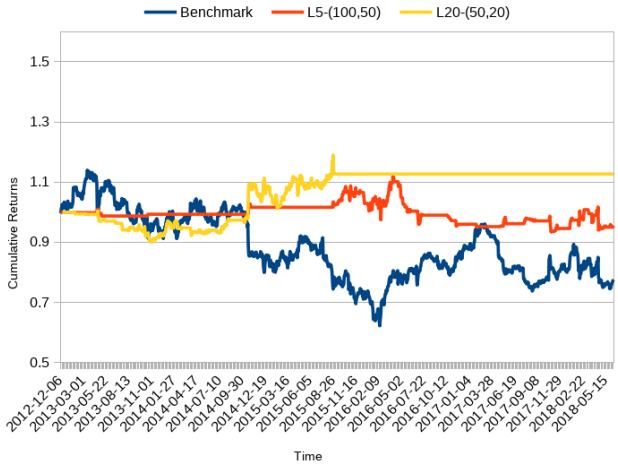

Fig. 8: Cumulative returns for the L5-(100,50) (selected) model and L20-(50,20) (best performing) model - IBM Price Prediction

4.1.3 Analysis

For the returns prediction, both the selected model and the best performing model have a final return on the period that beats the benchmark by 20% and 60%, respectively. In the case of the price prediction, the selected model beats the benchmark by 15% and the best performing model beats by 30%.

The reason for this difference may be the possibility of the prices that are fed to the model not being stationary [Adhikari and Agrawal, 2013]. Typically, a time series is considered non-deterministic, which means that it cannot be made any kind of prediction of the future behaviour with certainty. It is also assumed that it follows a certain probability distribution model [Hansen, 2017] that describes the joint distribution of the random variable x_t. Although, it is known that they can follow a more or less regular pattern for long periods. The stationarity can be viewed as the absent of trend or seasonal components, where the variables x_t follow a statistical equilibrium. In this equilibrium, properties like mean and variance of the stationary process do not depend upon time.

Finally, the Augmented Dickey-Fuller test (ADF) was applied to the price and returns data. This is used to test the null hypothesis stating that a unit root is present in the time series. Upon rejection of the null hypothesis, it suggests that a unit-root is not present in the series. The ADF test yields the results showed in table 5.

	ADF Statistic	P-value
Price	-1.32	0.62
Returns	-15,73	0

TABLE 5: ADF Results for the IBM Stock

For the price data, the ADF statistic yields a negative value, above any of the critical values. Also, the p-value is much bigger than 0.05, which means that we can accept the H0, meaning that the data isn't stationary and has a time dependent structure (unit-root) and so it is not stationary.

Regarding the returns data, the same test was applied, with very different results. For the returns, the ADF statistic is highly negative and the p-value is very close to zero - this allows us to reject the null hypothesis H0 and to accept that a unit-root is not present and so stationary characteristics are present.

4.2 Case Study II - AAPL

The data used for the AAPL stock was divided in training data and testing data, ranging from $2000-03-01$ to $2012-11-23$ and from $2012-11-26$ to $2018-05-08$, respectively.

In this case study, it can be seen that during the training period, some price drops occur but in most of period the price has an increasing trend. This is the same behaviour that occurs in the overall of the test data, yielding a dataset where the training and testing have similar characteristics.

Regarding the AAPL stock, the ADF test was applied to see whether the price data has stationary properties. The results pointed to the non-stationarity of the price data and so, only the returns prediction was analyzed.

The selected model, considering the training metrics, would be the F5-(100,50), because it presents the higher accuracy for the lowest value of RMSE. The model to be selected based on the best performance of the test would be the L20-(500,100). Both models metrics are shown in table 6.

In figure 9, it is presented the benchmark for this stock, along with the selected model and the best performed model. It can be seen that the selected model behaves poorly for this stock, compared with the benchmark, despite achieving a 66% accuracy in the test. By contract, the best performing model

accuracy's is much lower, only 50%. Thus, it ends up performing significatevely better than the selected model.

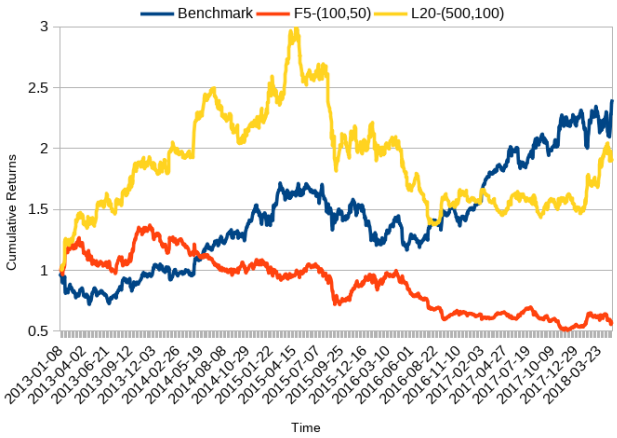

Fig. 9: Cumulative returns for the F5-(100,50) (selected) model and L20-(500,100) (best performing) model - AAPL Returns Prediction

To better understand the low performance of the selected model despite the testing metrics appear to be good, figure 10 containing the strategy signals. The histogram of the returns of this model is also shown in figure 11.

Fig. 10: Strategy signals of the F5-(100,50) (selected) model - AAPL Returns Prediction

In this figure it's plotted the true price of the stock during a fraction of the period corresponding to the test. On top of the price, two types of points, with distinct colors, are marked, to point the action taken at each timestep. In blue, the *Buy* signal is showed and in red the *Sell*. This signal is drew from the action that is taken by the strategy and defined depending on the strategy. It can be seen sometimes that, when the stock price rallies to local *maxima*, the selected action is to sell, instead of buy. These situations contribute to leading to the pour returns of the strategy, comparing to the benchmark.

Another factor that leads to a bad performance of the strategy can be seen by the histogram of the returns in figure 11. The curve resembles a normal curve with a mean slightly deviated to the right of 0. Nevertheless, the curve isn't symmetric, having a bigger right tail (which implies having a bigger magnitude of positive returns than negative returns) but having a bigger amount of negative returns in the range $[-2.5\%, 0]$.

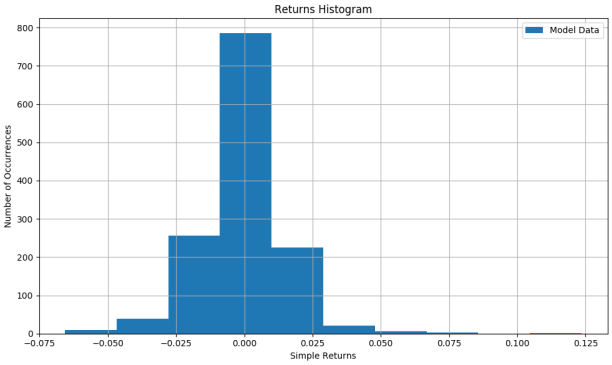

Fig. 11: Histogram of the AAPL selected model returns prediction

4.3 Case Study III - FKLS

In the FLKS stock, the stock price has the opposite behaviour of the AAPL, with the price increasing in the beginning of the training period and then falling abruptly almost through out all of the rest of the period. In the test period the stock goes sideways, with a peak that almost doubles the value and then falling again. Another thing that is particular in this stock is that it is listed in 2015 and not in the beginning of 2000, which determines the size of the datasets for training and testing were much smaller, which has the potential to compromise the training process.

The results of the hyperparameter search for this stock are presented in table 7. In this case, the best performing model and the selected model are the same.

The cumulative returns of the model is displayed in figure 12, along with the benchmark. It can easily be seen that the models beat the benchmark in a very surprising way, performing a steadier walk with fewer positions and almost never losing money. The best model L20-(10,5), on the other hand, has a more sinuous walk, losing some money abruptly but ending up by compensating the losses.

	Model	Days	Architecture	RMSE		Accuracy	
				Train	Test	Train	Test
Selected	FeedForward	5	(100,50)	2.64	1.59	0.66	0.66
Best	LSTM	20	(500,100)	4.34	2.52	0.66	0.5

TABLE 6: Selected Model vs Best Model - AAPL Returns Prediction

	Model	Days	Architecture	Accuracy	
				Train	Test
Selected & Best	FeedForward	20	(50,20)	0.68	0.55

TABLE 7: Selected Model vs Best Model - FLKS Returns Prediction

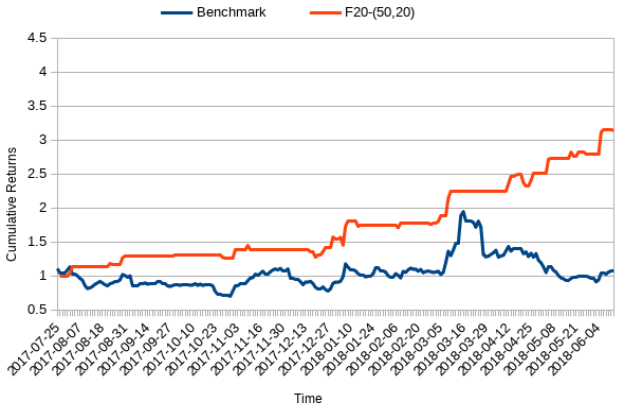

Fig. 12: Cumulative returns for the F20-(50,20) (selected best) model - FLKS Returns Prediction

Both models have accuracy in training no much bigger than the accuracy provided by a coin toss strategy. To understand the extraordinary performance of them, both the returns histogram for both models are displayed, for comparison.

For the selected model, figure 13 shows there are a very large number of small negative returns, but also there are some big positive returns that give an asymmetry to the right and end up by defining the strategy performance.

This model is the example of a strategy where, besides the accuracy being relatively low, when the correct action is taken, a big return compensates for the small time losses, that represent half of the actions in the testing.

4.4 Discussion

The metrics of all the analyzed models (selected and best performing) are condensed in table 8.

Regarding the backoff days, it is interesting to see that, for most of the models ($6/8$), the hyperparameter search yielded the models with 20 of backoff days, indicating that past values up to a month (the market closes in the weekends) are useful to make predictions on the security.

As for the values for the accuracy, the values of the selected models are, in general, lower when compared with the values for the best model for each stock. In case of the IBM stock, comparing with the results obtained in [Li et al., 2007], the value for the accuracy is lower in both the selected model and in the best performing one. In the latter, the value (58%) is very close to the one obtained in [Li et al., 2007] (60%). As for the AAPL stock, in [Arévalo et al., 2016] the obtained value for the directional accuracy is lower (64.8%) than the selected model (66%) but it is higher than the best performing model that, in this case, only gets an accuracy of 50%.

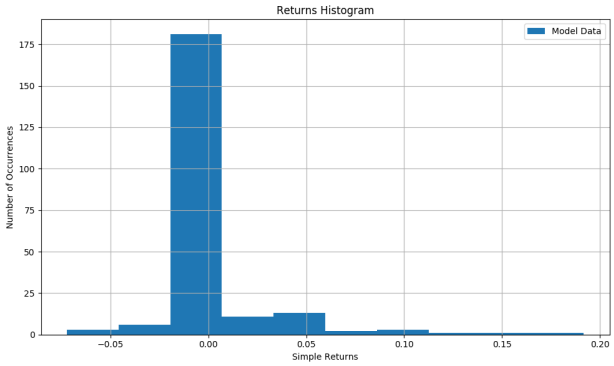

Fig. 13: Histogram of the F20-(50,20) (selected) model - FLKS Returns Prediction

Regarding the number of neurons per layer, the results in table 8 do not point to a winning architecture, with different architectures being selected for different securities.

The annualized returns of the different created models is also shown in table 8. The two of the three analyzed stocks have achieve annualized returns better that the benchmark. Also, it can be seen that, despite the best performing models being generated by the framework, they aren't selected because it's training metrics do not give such information. The result is selected models with worst performances than the best ones. Only the FLKS is an exception to this, where the best model is the selected model. This is also the stock for which the returns is the highest, with values of accuracy in testing and training very high. Regarding the the AAPL stock, it can be seen that the accuracy metrics of the selected model are also very good. Despite this, the annualized return is much

Stock	Prediction		Model	Days	Architecture	RMSE		Accuracy		Annualized Return [%]	
						Train	Test	Train	Test	Benchmark	Model
IBM	Price	Selected	LSTM	5	(100,50)	24.16	73.12	0.51	0.5	-0.09	-0.01
		Best	LSTM	20	(50,20)	70.44	150.09	0.5	0.52		0.05
	Returns	Selected	LSTM	20	(10,5)	2.82	2.15	0.65	0.51		0
		Best	FeedForward	20	(50,20)	1.81	1.25	0.57	0.58		0.13
AAPL	Returns	Selected	FeedForward	5	(100,50)	2.64	1.59	0.66	0.66	0.38	-0.18
		Best	LSTM	20	(500,100)	4.34	2.52	0.66	0.5		0.27
FLKS	Returns	Selected & Best	FeedForward	20	(50,20)	-	-	0.68	0.55	0.17	0.97

TABLE 8: Condensed metrics of the models performance for the case studies

worst than the benchmark. A simple explanation for this can be seen in the number of actions that the strategy associated to the returns does in this case, which can be seen in figure 10. The strategy accuracy in the test period is 66% but, with so much actions placed, the returns get diluted and, in fact, ends up having a great number of misses of high magnitude, which goes against the returns accumulated by the hits. In fact, the implemented strategies are so simple that do not contemplate the differences between the stocks.

5 Conclusion

The system is able to generate models that are able to beat the benchmark that can be selected using training metrics. Despite this, it's still not able to select the model that best performs in out-of-sample data using the training metrics The Backtesting charts developed, namely the strategy actions over the variation of the price and the returns histogram are specially useful to diagnose and assess the behaviour of the model. On the first, it can be seen how the model operates under volatility and the impact of trends on the actions. On the latter, it can be perceived the type of returns that are more common.

References

[Adhikari and Agrawal, 2013] Adhikari, R. and Agrawal, R. (2013). An introductory study on time series modeling and forecasting. *arXiv preprint arXiv:1302.6613*.

[Arévalo et al., 2016] Arévalo, A., Niño, J., Hernández, G., and Sandoval, J. (2016). High-frequency trading strategy based on deep neural networks. In *International conference on intelligent computing*, pages 424–436. Springer.

[Attigeri et al., 2015] Attigeri, G. V., M, M. P. M., Pai, R. M., and Nayak, A. (2015). Stock market prediction: A big data approach. In *TENCON 2015 - 2015 IEEE Region 10 Conference*, pages 1–5.

[Cavalcante et al., 2016] Cavalcante, R. C., Brasileiro, R. C., Souza, V. L., Nobrega, J. P., and Oliveira, A. L. (2016). Computational intelligence and financial markets: A survey and future directions. *Expert Systems with Applications*, 55:194 – 211.

[Fama, 1970] Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *The journal of Finance*, 25(2):383–417.

[Fischer and Krauss, 2017] Fischer, T. and Krauss, C. (2017). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*.

[Hansen, 2017] Hansen, L. P. (2017). Time-series econometrics in macroeconomics and finance. *Journal of Political Economy*, 125(6):1774–1782.

[Heaton et al., 2016] Heaton, J. B., Polson, N. G., and Witte, J. H. (2016). Deep learning in finance. *CoRR*, abs/1602.06561.

[Hsiao and Tsai, 2018] Hsiao, Y.-J. and Tsai, W.-C. (2018). Financial literacy and participation in the derivatives markets. *Journal of Banking Finance*, 88:15 – 29.

[Kewat et al., 2017] Kewat, P., Sharma, R., Singh, U., and Itare, R. (2017). Support vector machines through financial time series forecasting. In *Electronics, Communication and Aerospace Technology (ICECA), 2017 International conference of*, volume 2, pages 471–477. IEEE.

[Krauss et al., 2016] Krauss, C., Do, X. A., and Huck, N. (2016). Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. Technical report.

[LeCun et al., 2012] LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. (2012). Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer.

[Lee et al., 2007] Lee, J. W., Park, J., O, J., Lee, J., and Hong, E. (2007). A multiagent approach to q-learning for daily stock trading. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 37(6):864–877.

[Li et al., 2007] Li, H., Dagli, C. H., and Enke, D. (2007). Short-term stock market timing prediction under reinforcement learning schemes. In *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 233–240.

[Li and Tam, 2017] Li, Z. and Tam, V. (2017). A comparative study of a recurrent neural network and support vector machine for predicting price movements of stocks of different volatilites. In *Computational Intelligence (SSCI), 2017 IEEE Symposium Series on*, pages 1–8. IEEE.

[Liu et al., 2017] Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., and Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11 – 26.

[Malkiel, 2003] Malkiel, B. G. (2003). The efficient market hypothesis and its critics.

[Nelson et al., 2017] Nelson, D. M. Q., Pereira, A. C. M., and de Oliveira, R. A. (2017). Stock market's price movement prediction with lstm neural networks. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 1419–1426.

[Siew and Nordin, 2012] Siew, H. L. and Nordin, M. J. (2012). Regression techniques for the prediction of stock price trend. In *Statistics in Science, Business, and Engineering (ICSSBE), 2012 International Conference on*, pages 1–5. IEEE.

[Sola and Sevilla, 1997] Sola, J. and Sevilla, J. (1997). Importance of input data normalization for the application of neural networks to complex industrial problems. *IEEE Transactions on Nuclear Science*, 44(3):1464–1468.