# Forecasting Currency Risk Premiums with Artificial Neural Networks in Dow, Ibex35, Stoxx and MCSIW stock markets

Miquel Perelló, E-mail: miquel.perello.nieto@est.fib.upc.edu

**Abstract**—In this work I tried to predict the future currency risk premiums of four stock markets; Dow, Ibex35, Stoxx and MCSIW. The models used for this objective are a Single-layer perceptron, a multilayer perceptron and an Elman neural network. We will evaluate the performance of each one and will get an out-of-sample error for the model that performs better in validation data.

**Keywords**—Single-layer perceptron, Multylayer Perceptron, Elman neural Network, timeseries, prediction, risk premiums

## INTRODUCTION

ARtificial Neural Networks (ANN) have been used in multiples application for prediction and forecasting, since the backpropagation was introduced into the training phase for ANN their use spread rapidly.

In this work, we will use different ANN architectures to forecast the the future currency risk premiums for the Dow, Ibex35, Stoxx and MCSIW stock markets.

## 1 DATASET

The dataset we are going to fit is a timeserie of risk premiums, concretely Dow Jones, Ibex-35, MCIW and STOXX. The values to predict are computed with the Gordon and Shapiro equation.

The value ranges of the dataset are very different and in the special case of BPAe12 are very high. It must made the neural networks not to train correctly, for that reason is necessary to standardise the data. In addition, it is usual to find peaks in economic data, for that reason I prefer to do the log of the data expecting to improve the predictions.

$$X' = \frac{\log(X)}{\sum \log(X)} \tag{1}$$

Where X is a matrix with one row for each sample and one column for each variable. And the sum and division is done for each column. The resulting dataset is shown in the figure 1.

## 2 EMPIRICAL METHODOLOGY

### 2.1 Linear Regression

In some problems the best fit can be done using a linear regression. If it is the case, is always better than a nonlinear model because the last can produce an overfitting of the data.

#### *Single-layer Perceptron*

It is possible to generate a linear regression using an ANN without hidden units. This is the first ANN designed by McCulloch and Pitts (1943). The equation is a sum of all the input weights and values plus a bias. It can be seen in the next equation.

$$h_{sw}(x_j) = \beta_0 + w_1 x_{j,1} + \cdots + w_n x_{j,n} = \beta_0 + \sum_{i=1}^{n} w_i x_{j,i} \tag{2}$$

A representation of this network is shown in the figure 2 in the special case of our dataset. It contains the eight input values, the four output values, the bias B and the weights $w_{i,j}$ connecting each input and output units.

This model will be the base to compare the results of more complex ones.

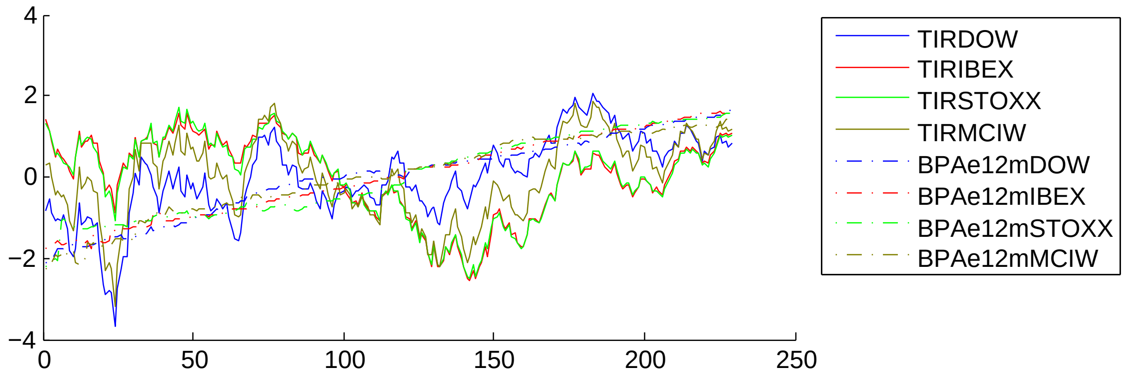

Fig. 1. **Standarized** - only input values

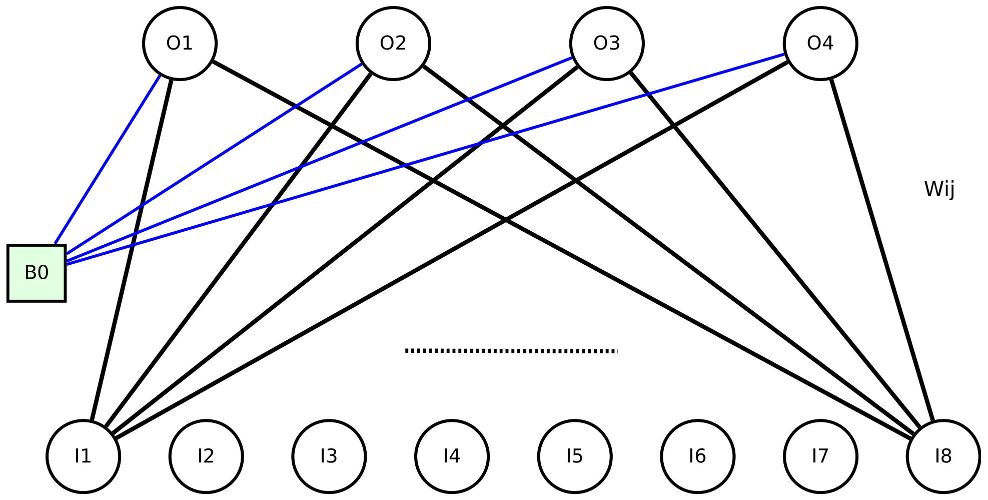

Fig. 2. **Neural Network without hidden layer** - this structure performs a linear regression

### 2.2 Nonlinear Models

These models performs nonlinear fittings of the data, that raises to better predictions in some cases, but they may overfit the data. For that reason it is necessary to cross-validate the solutions to discard models that performs well in the training, but bad in the validation set.

The nonlinear models used in this work are Multilayer Perceptron (MLP) and Elman neural networks. Both are ANN with hidden units, in that case with one hidden layer.

#### *Multilayer Perceptron*

MLP was the natural step for the ANN to achieve nonlinear solutions. With one hidden layer it is possible to create any function, the problem was to train the network. But it was solved with the back-propagation algorithm. The equation for the MLP allows to implement any threshold function (must be derivable) and any number of hidden layers. In our case, with only one hidden layer it uses the next equation:

$$r_t = \beta_0 + \sum_{j=1}^{q} \beta_j g \left( \sum_{i=1}^{p} \phi_{ij} r_{t-i} + \phi_{0j} \right) + \epsilon_t \quad (3)$$

The graphical representation of the MLP can be seen in the figure 3, where there is represented the eight input values, N hidden units and four output values.

#### *Elman Neural Network*

This is a feedforward neural network with a recurrent layer. It allows to memorise time series and may be useful on pattern recognition tasks. The hidden layer is replicated for the next input data and given as another input values. In that case, the order of training is important, for that reason, it must be trained with a real sequence of the dataset and validated with another one.

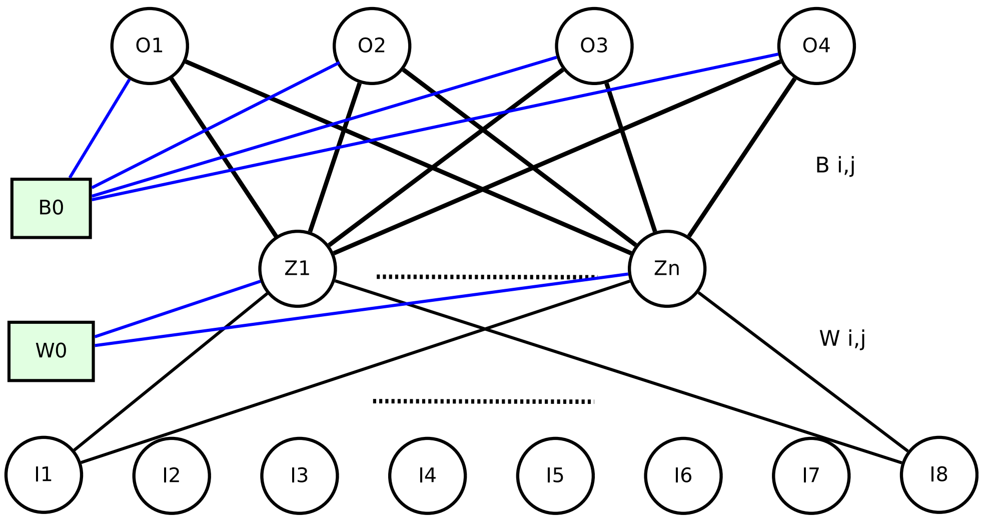

Fig. 3. **Multilayer Perceptron** - with one hidden layer and N hidden units

The resulting equation is similar to the MLP, except for the previous hidden unit values in the input. The next equation is for our case of one hidden layer.

$$
\begin{aligned}
r_t &= \beta_0 + \sum_{j=1}^{q} \beta_j z_{j,t} + \epsilon_t \\
z_{j,t} &= g\left(\sum_{i=1}^{p} \phi_{ij} r_{t-i} + \phi_{0j}\right)
\end{aligned}
\tag{4}
$$

In the figure 4 there is a representation of our case, with the eight input values, the hidden layer, the previous hidden layer values as a inputs, and the four output neurons.

## 3 Empirical Analysis

Once standarized the dataset with the method explained in section 1, the values to predict are shifted one position to the past, then I divided the dataset in two portions. The first $85\%$ (229 samples) is used to train and validate, and the last $15\%$ (41 samples) is used to estimate an out-of-sample error.

With the $85\%$ portion of the dataset is divided randomly in a $80\%$ for training and $20\%$ for validation ten times per model parameters modification. In the case of the Elman neural network, the division is not taken absolutely random, instead is taken by sequences.

Notice that this method is not crossvalidation, it is not implemented in the *Matlab* toolbox I am using. Instead it, this method tries to average the ten errors to minimise the possible changes in the performance.

For the MLP and Elman NN the number of hidden units is changed from 1 to 30 neuron. In the MLP the maximum number of epoch is set to one thousand, in the Elman NN the number of epoch has been reduced to one hundred, this is because when training if it have a larger number of epoch it usually crashes in my *Matlab* version.

Regarding to the results, I selected Mean Squared Error (MSE) and Sum of Squared Error (SSE) as a training function and for forecast evaluation. These two

*MSE with eight variables*

| model | h.u. | performance |
|---|---|---|
| Perceptron | 0 | 0.2146 |
| MLP | 3 | 0.1695 |
| Elman | 1 | 0.4831 |

When evaluating the MSE on the eight variables to do the predictions in all the cases the Elman is the worse model. On the other side the The Perceptron performs very well compared with the others. Finally the MLP performs slightly better in some cases with four, five and six neurons. In the other cases it overffits the training data and performs worse in the validation set (see figure 5).

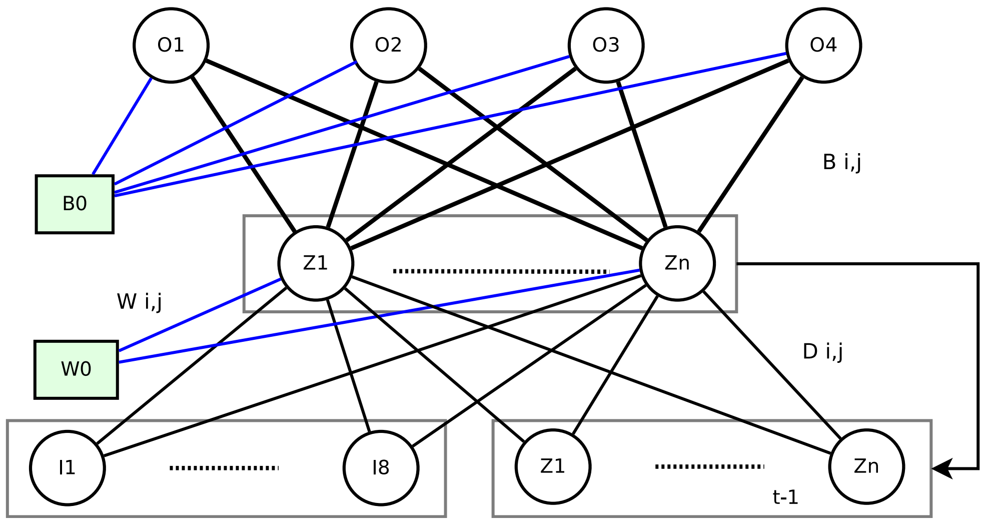

Fig. 4. **Elman Neural Network** - with one hidden layer, N hidden units and one time delay

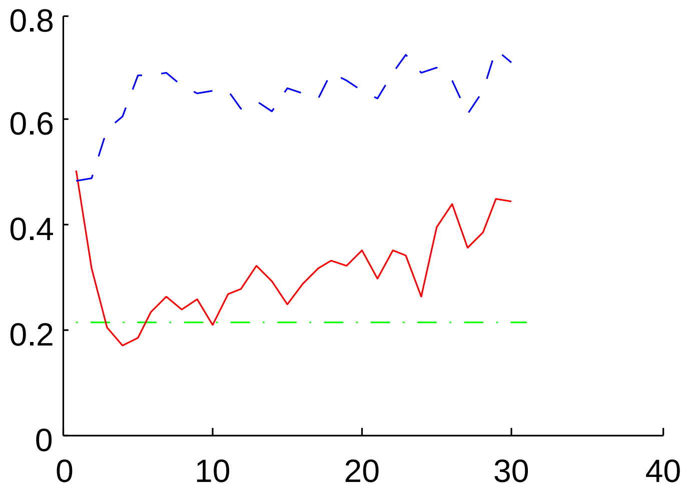

Fig. 5. **Performance on MSE** for the three models and eight input variables

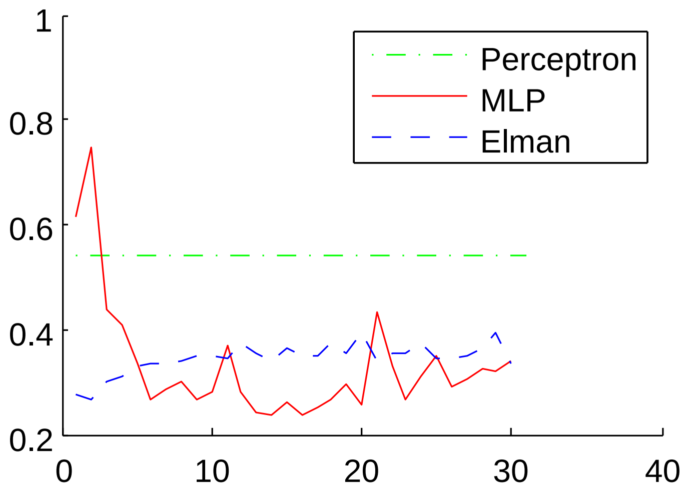

Fig. 6. **Performance on MSE** for the three models with the actual values as inputs to predict

*MSE with twelve variables*

| model | h.u. | performance |
|---|---|---|
| Perceptron | 0 | 0.565 |
| MLP | 14 | 0.223 |
| Elman | 2 | 0.2677 |

If we add to the input data the actual currency risks for some reason the Perceptron perfomes worst than the rest. It is possible that the complexity on the number of variables or the kind of these values do not let to perform a linear fitting. The Elman networks is still worse than the MLP but with a reduced number of hidden units (one to six) seems to be better. On the other hand, MLP have the best results with fourteen and sixteen hidden units (see figure 6). Compared with the previous case the complexity of the model is able to afford best results.

*SSE with eight variables*

| model | h.u. | performance |
|---|---|---|
| Perceptron | 0 | 26.2118 |
| MLP | 4 | 33.0766 |
| Elman | 1 | 4.2235e+03 |

When computing the performance with SSE the values increases exponentially, and in the case of Elman networks the performance is extremely worse (see figure 7).

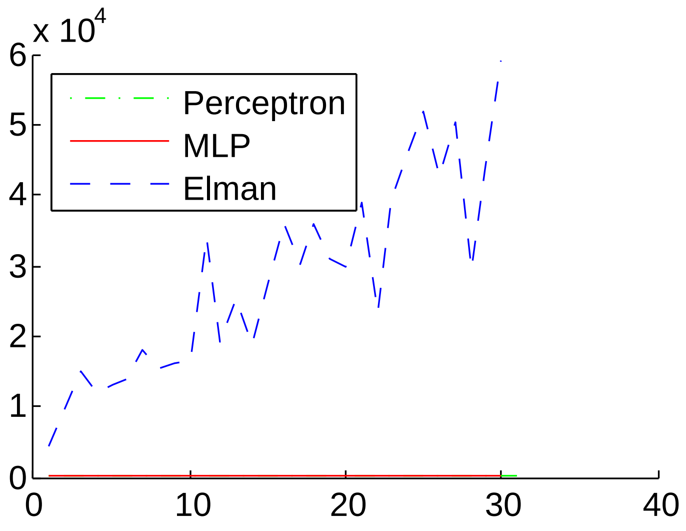

Fig. 7. **Performance on SSE** for all the models and eight input variables

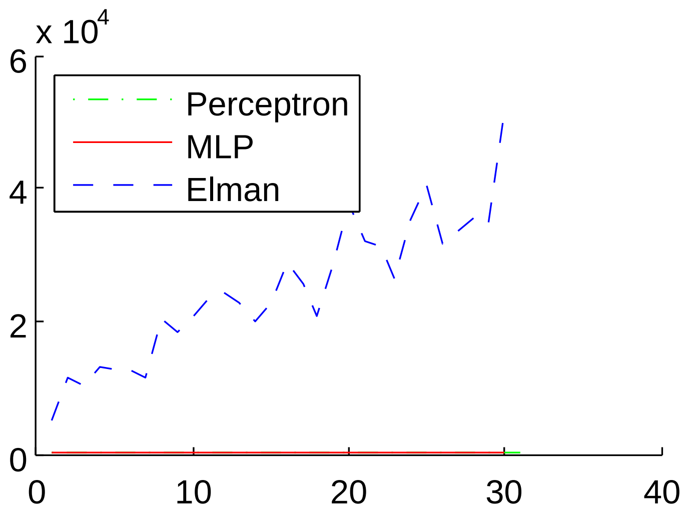

Fig. 9. **erformance on SSE** for all the models and twelve input variables

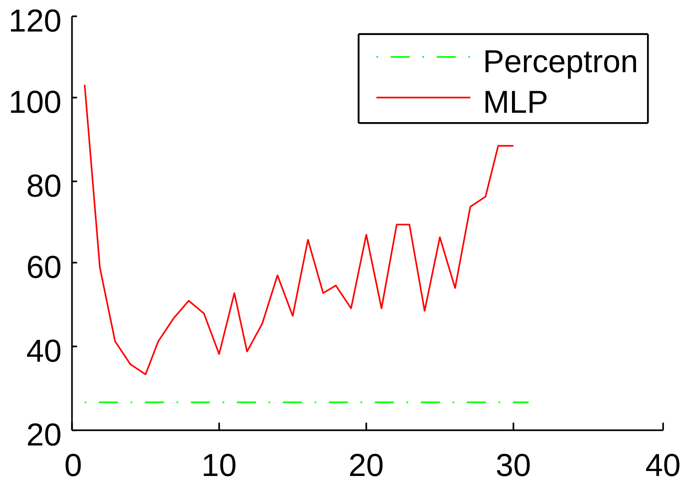

Fig. 8. **Performance on SSE** only with perceptron and MLP, and eight input variables

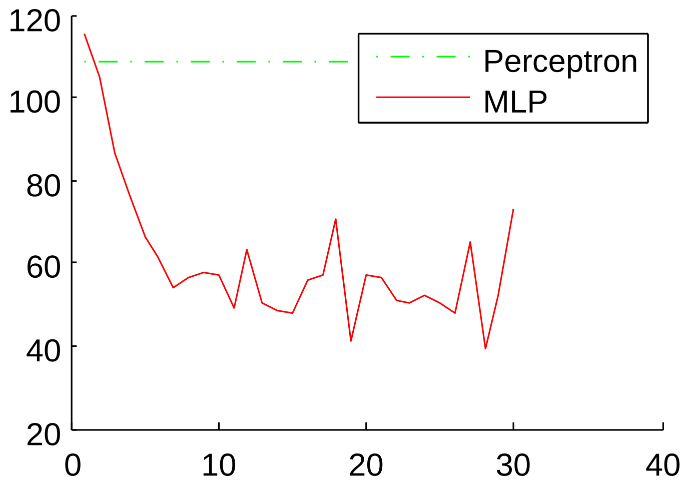

Fig. 10. **erformance on SSE** only with perceptron and MLP, and twelve input variables

If we take a look at perceptron and MLP, the first performs better no matter the number of hidden units the MLP has (see figure 8).

*SSE with twelve variables*

| model | h.u. | performance |
|---|---|---|
| Perceptron | 0 | 109.803 |
| MLP | 28 | 38.624 |
| Elman | 1 | 5.0332e+03 |

## 3.1 Test out-of-sample error

The selected model is MLP with three hidden units that performed in the validation error 0.1695 MSE. Finally I executed the out-of-sample error test with the $15\%$ of the data to make the prediction of all the values. When predicting only the next sample the error is 0.2694. That is similar to the validation error, but for only one prediction it can be so high. The other error is for all the test data with 47 instances, the performance is 0.8186 MSE.

| model | h.u. | predict | performance |
|---|---|---|---|
| MLP | 3 | one | 0.2694 |
| MLP | 3 | 47 | 0.8186 |

## 4 Conclusion

This work shown that it is possible to perform a prediction of the forecasting currency risk premiums with a MLP and an out-of-sample error of MSE. We saw that perceptron performs well in certain situations with eight variables, but it is not better than a MLP with the good parametrisation. On the other side, the Elman network is in the majority of the cases not good for this problem.

## References

[1] Pérez-Rodríguez, Jorge V., Salvador Torra, and Julian Andrada-Felix. "Are Spanish Ibex35 stock future index returns forecasted with non-linear models?." Applied Financial Economics 15.14 (2005): 963-975.

[2] Pérez-Rodríguez, Jorge V., Salvador Torra, and Julian Andrada-Félix. "STAR and ANN models: forecasting performance on the Spanish "Ibex-35" stock index." Journal of Empirical Finance 12.3 (2005): 490-509.

[3] Russell, Stuart Jonathan, et al. Artificial intelligence: a modern approach. Vol. 74. Englewood Cliffs: Prentice hall, 1995.

## Appendix 1

**Are Spanish Ibex35 stock future index returns forecasted with non-linear models?**

*Jorge V. Pérez-Rodríguez, Salvador Torra and Julian Andrada-Félix*

In this report, the authors made an analysis of the Spanish Ibex35 stock index returns. In order to extract the conclusions, it has been made a BDS test to search nonlinearities (Brock et al. 1987), a forecasting performance by using statistical and economic criteria.

For the BDS test the series is transformed into logarithms to compute continuous returns, with the next expression : $r_t = log(F_t/F_{t-1})$, the results applying this technique shows that the series contains nonlinear structures with a $5\%$ significance level.

On the forecasting performance by using statistical criteria, the authors used the Smooth Transition Autoregressive Models (STAR), Artificial Neural Networks (ANN) and Nearest Neighbours (NN) models. Several variants of these models were tested to perform this analysis, linear autoregressive (AR), logistic STAR (LSTAR), exponential STAR (ESTAR), Multilayer Perceptron (MLP), Jump Connection Nets (JCN) and a recurrent network (Elman).

For each of these models has been computed the mean absolute error (MAE), mean absolute percentage error (MAPE), root mean squared error (RMSE), U-Theil, the proportion of times the sign of the prediction was correct and Pesaran and Timmermann test (DA, 1992).

With all these data tests the DA indicates that AR, STAR and JCN models rejects the independence of the forecasts and realizations (with a $10\%$ significance level). Furthermore the STAR, ANN and NN models performs better than random walk and AR models in terms of goodness of forecasts.

Another test was performed with the forecasting performance by using economic criteria. In that case the performance is evaluated using different trading strategies. The different equations to perform the analysis are not explained in this summary but the conclusions of the report shows that Elman and NN are the best models improving the mean returns.

Finally, with all these tests is concluded that out-of-sample ANN and NN forecasts are more accurate than AR and STAR models. Furthermore it is shown that these models can predict the returns on the Ibex35 stock future index.

## Appendix 2

**STAR and ANN models: forecasting performance on the Spanish "Ibex-35" stock index**

*Jorge V. Pérez-Rodríguez, Salvador Torra and Julian Andrada-Félix*

The purpose of this paper is to improve the prediction of Spanish Ibex-35 stock index returns with nonlinear regression models. It is focused in smooth transition autoregression (STAR) models and artificial neural networks (ANN).

The problem is divided in three forecast horizons for short to long predictions (100, 200 and 500 trading days). The methodology to analyse the performance of the different models is based in the statistical assessment of the out-of-sample error.

They made three types of tests: comparing the Signs and P-values of the Pesaran and Timmermann test for all models; it shows that ANN techniques performs better than the others for 100-forecast periods and one-step-ahead forecast method. Moreover the linear AR model performs better when the forecast horizon is extended.

Secondly is made the Chong and Hendry's forecast encompassing test, that performs a competition between the different models in a one-versus-one test. On this test the MLP and JCN networks performs slightly better, most in the special case of one-step-ahead.

Finally the Diebold and Mariano forecast accuracy test is performed, with the MAE loss function. It computes the difference of errors for each pair of models to perform statistical significant analysis. It concludes that random walk performs worst for all the cases with a $5\%$ significance level. A part of that, there is no evidences of any model performing better than the others in that test.

On the other hand, they performed a forecast performance with economic criteria in a simple trading strategy. The method is to compare a strategy with a trading rule based in the predictions versus a buy-and-hold strategy. It concludes that in general there is not a significant improvement in the mean Sharpe risk with respect to buy-and-hold approach. Indeed, the MLP produces an even higher Sharpe ratio in the cases of 500 and 200 forecast-horizons.