AI FOR FINANCE: FINAL PROJECT

Depedri Kevin

13-06-2022

Index

General introduction

In this final project a quantitative analysis over 5 companies (“AMZT.MI”, “ENI.MI”, “AGL.MI”, “LDOF.MI”, “ENEI.MI”) is carried out using different type of predictive models. The goal is to show how different methods can give us a prediction of the stock price percentual variation of the next quarter and assessing the quality of the generated predictions. To do so, we are using as input data information regarding the company fundamentals, the price and the opinion of the main analysts.

The project’s script is divided in sections, each one has a specific focus as follows:

1) Data retrieval from Eikon Refinitiv Workspace using the Eikon API for Python and data cleaning. The list of the extracted variables and the cleaning procedure that has been adopted are described accurately in the section ‘Data description’.

2) Computation of a VAR model. This model is used to get a first glance over the correlation between the available variables.

3) Computation of prediction models based on OLS regression. In this section different types of OLS models are created and the relative predictions are saved into a data-frame, to then compare them using metrics such as MSE and RMSE. The OLS models that have been computed are:
    - In-Sample with all variables
    - In-Sample with significant variables only
    - Out-of-Sample with all variables
    - Out-of-Sample with significant variables only
    - Recurrent-out-of-Sample with all variables
    - Recurrent-out-of-Sample with significant variables only

4) Computation of prediction models based on a Recurrent-Neural-Network (RNN), more specifically on a Long Short-Term Memory RNN that from now will be called LSTM-RNN. In this section different types of models are created and the corresponding predictions are saved into the data-frame created in the previous point. All the LSTM-RNN models use always all the available input variables. The generated predictions are then compared between them using metrics such as MSE and RMSE. The LSTM-RNN models that have been computed are:
    - In-Sample
    - Out-of-Sample
    - Recurrent-out-of-Sample

In this report the results for the analyzed company (AZMT.MI) will be presented for each of the previous points. The results are expressed both numerically using tables and graphically showing the behavior of the generated predictions and comparing it to the real changes of price.

Furthermore, the analysis carried out for this company and for all the other companies previously cited is attached this report. To see all the numerical and graphical results for the other companies please refer to the section ‘Appendix: Codes’.

Description of the forecasting model

In this project two forecasting model has been chosen to generate prediction about the price percentual change for the next quarter. The chosen models are OLS and LSTM-RNN.

OLS – Ordinary Least Squares

Ordinary Least Squares (or OLS) is a type of linear least squares method for estimating the unknown parameters in a linear regression model. OLS chooses the parameters of a linear function of a set of explanatory/independent variables (which are our input variables) by the principle of least squares: minimizing the sum of the squares of the differences between the observed dependent variable (which is the variable that we want to predict, and so the price percentual change) in the given dataset and those predicted by the linear function of the independent variable (and so predicted by our OLS model).

As previously seen, OLS has been computed six times, using different settings, in particular:

- The 'In-Sample with all variables' model is trained using all the independent variables (also the not statistically significant ones) over all the available dataset. This means that the prediction generated by this model will be generated for values that it has already seen. Therefore, it will be one of our most accurate models (together with the 'In-Sample with significant variables only' model) and it will reach an accuracy and an R^2 value that cannot be reached in real forecasting where the value that we want to forecast has never been seen before.

- The 'In-Sample with significant variables only' model is computed as the model above, but using as independent variables only the statistically significant ones, which has been extracted from the set of all the independent variables through an iterative procedure that removed at each iteration the least significant variables with a p-value > 0.05. Again, also in this case the model is trained with all the available dataset.

- The 'Out-of-Sample with all variables' model is trained using all the independent variables (also the not statistically significant ones) but only over a certain portion of the dataset (80%). After this point, the model is never re-trained and the prediction are based on the knowledge extracted by the first 80% of the information present in the dataset.

- The 'Out-of-Sample with significant variables only' model is computed as the model above, but using as independent variables only the statistically significant ones.

- The 'Recurrent-out-of-Sample with all variables' model is trained using all the independent variables (also the not statistically significant ones) but only over a certain portion of the dataset (80%). At this point, the first prediction is done, and then the real value of that period is added to the train dataset, and the model is re-trained before giving another prediction. This procedure of training/prediction is iterated until all the predictions have been computed.

- The 'Recurrent-out-of-Sample with significant variables only' model is computed as the model above, but using as independent variables only the statistically significant ones.

LSTM - Recurrent Neural Network

Long Short-Term Memory Recurrent Neural Network (LSTM-RNN) is an evolution of the vanilla RNN artificial neural network that has been designed to process entire sequences of data. Due to this peculiarity, LSTM networks are well-suited to make predictions based on time series data, also if there are lags of unknown duration between important events in a time series.

As previously seen, LSTM-RNN has been trained three times, using different settings, in particular:

- The 'In-Sample' model is trained using all the independent variables over all the available dataset. Again, also in this case this model will be the most accurate between all the RNN models, since it has been trained using all the available data. In particular this model shows us the ability of the LSTM-RNN to learn complex patterns, indeed, the prediction done by this model follows almost perfectly the real price percentual variation.

- The 'Out-of-Sample' model is trained using all the independent variables but only over a certain portion of the dataset (80%). After this point. the model is never re-trained and the prediction are based on the knowledge extracted by the first 80% of the information present in the dataset.

- The 'Recurrent-out-of-Sample' model is trained using all the independent variables but only over a certain portion of the dataset (80%). At this point, the first prediction is done, and then the real value of that period is added to the train dataset, and the model is re-trained before giving another prediction. This procedure of training/prediction is iterated until all the predictions have been computed.

Data description

To train the predictive models, we used different types of data coming from the platform of Refinitiv Eikon called Refinitiv Workspace. In this way we ensured a full understanding of the different ongoing dynamics by our models. In particular, we decided to use data coming from the company fundamentals, the price and the opinions of the main Refinitiv analysts. The idea is that these main factors should reflect the company current situation and so they should be useful to determine some possible changes in the price of the stock.

Descriptive statistics

The list of the variables used with their description is presented here below.

Price variables:

- Price Close, it is the price to which the stock closed on a specific day. This data has been introduced since it allows to understand the general trend of the price.

Company fundamentals variable:

- Revenue, it represents the total revenue of the company. This data allows to understand how much activity and how many deals the company closed in the interested period.
- Operating Profit, it represents the total revenue minus the cost that the company needs to sustain to run its business. This data allows to understand how the cost are impacting the business.
- Net Income after Tax, it represents the final net income of the company after that all the expenses, interests and taxes have been paid. This data shows how much the business is profitable.
- Total Assets, it represents the total assets owned by the company. This data is an indicator of the physical value of the company.
- Total Liabilities, it represents the total debt owned by the company. This data is an indicator of how much of the company assets have been acquired through the use of debt.
- Shareholder Equity, it represents the difference between assets and liabilities. The grater it is, the more valuable is the company.
- Net cash, it represents the reserve of money of the company. This data is an indicator of the health of the company since more money implies more possibilities and less risks.

Refinitiv analysts' variables:

- Revenue - Median, it represents the median of the analysts' expected revenue for that specific period. This data allows us to anticipate the performance of the company in the next period.
- Price Target, it represents the mean of the analysts' expected price target for that specific period. This data allows us to anticipate the price that the company's stock could reach in the next period. This price is estimated by analysts and it should already encompass considerations around all the structure of the company.

Data cleaning

Once all the previously listed data have been downloaded, we performed some simple operations of data cleaning to ensure that all the data were correctly stored in our data-frame.

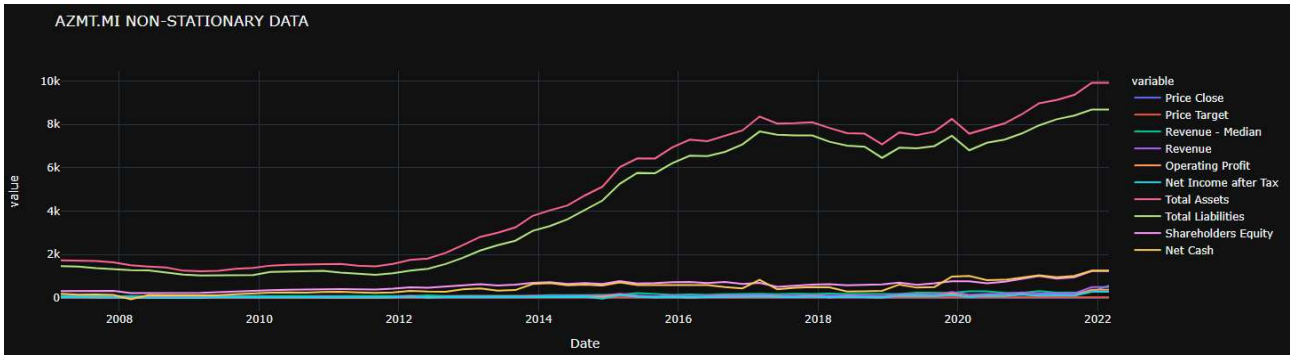
We identified two main problems in the downloaded data:

- Some of the used variables presented some missing data here and there along all the data-frame, but in total these missing values were just a few.
- Some of the used variables started later than other, leading to some long interval of missing data at the beginning of the data-frame (and so in the oldest years).

To address the first problem (missing values inside the data-frame) we decided to replace the missing data with the previous value of the data. In this way, we built a model that has all the necessary data and that will not lead to errors.

To fix the second problem we decided to remove all the rows where the first data cleaning procedure could not replace the values. In this way, we removed only the oldest lines for which there was not a precedent value that could have been propagated to the missing cells.

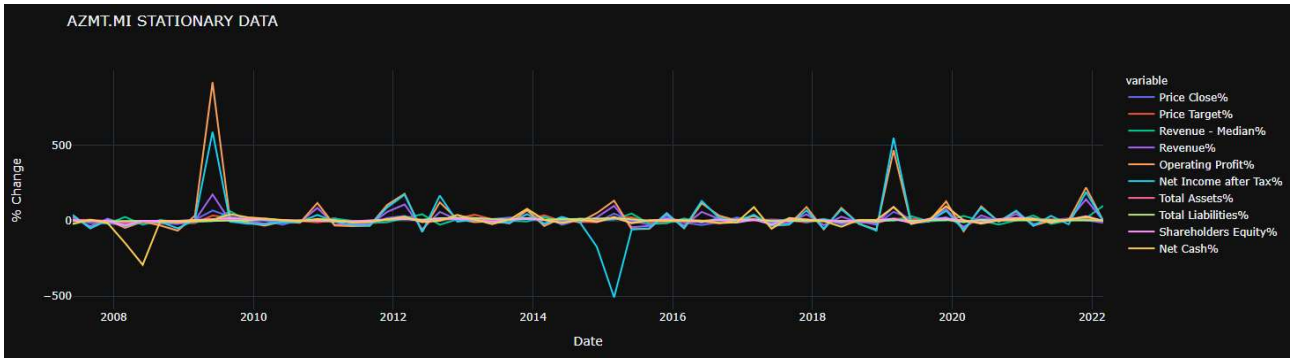
At this point, we obtained our final (non-stationary) data, which are plotted here below:



In the chart above the values are expressed in thousands (k) of millions of euros.

After the two previous steps the data cleaning procedure has been completed and so we created a new data-frame which substitutes the absolute values of each variable with its percentual change with respect to the previous value. In this way, we transformed the data from non-stationary (and so trending data) to stationary (and so ranging data). This is a fundamental step since the linear models that we will use before using the RNN-LSTM (like VAR and OLS) require stationary data to work correctly.

The obtained stationary data are plotted here below:



Econometric analysis

In this section the results obtained by the forecasting model based on OLS (which has been introduced in the section 'Description of forecasting models - OLS') are presented and discussed.

The econometric analysis is composed of four models that have been ran sequentially. To have a more ordered presentation of the results, this section is divided in four parts as follows.

VAR model

First, we computed a VAR model using the stationary data obtained in the previous step. The model has been set-up in a way that allows it to automatically define the best-lag parameters, which in our case resulted to be four, meaning that it is better to use the data from the previous four period to have the best correlations to describe the data for the next period.

```
VAR model results based on best lag value of 4 lags:
  Summary of Regression Results
==================================
Model:                         VAR
Method:                        OLS
Date:           Wed, 08, Jun, 2022
Time:                     16:03:40
--------------------------------------------------------------------
No. of Equations:         10.0000    BIC:                    65.3612
Nobs:                     56.0000    HQIC:                   56.2817
Log likelihood:          -1799.52    FPE:                4.93824e+23
AIC:                      50.5327    Det(Omega_mle):     2.03112e+21
--------------------------------------------------------------------
```

Looking at the results of the VAR model we were interested in the 'Correlation matrix of residuals' which shows how the different variables are correlated between them. In particular we are interested in seeing how the variable 'Price Close%' (which is the one that we want to predict) is correlated to the other variables. The result obtained by the VAR model for 'Price Close %' is shown here below in tabular form:

Price Close %	Price Target %	Revenue -Median %	Revenue %	Operating Profit %	Net Income after Tax %	Total Assets %	Total Liabilities %	Shareholders Equity %	Net Cash %
1.000	0.778	-0.099	0.261	0.223	0.330	0.655	0.535	0.446	0.501

The results of the 'Correlation matrix of residuals' shows how the variables 'Price Close %' is highly correlated with the variables: 'Price Target %', 'Total Assets %' and 'Total Liabilities %'. We expect these terms to be of interest moving on into the econometric analysis.

In-Sample OLS model

After computing the VAR model, we created the first OLS model which is 'In-Sample', meaning that it uses all the available data to build the model. The first run of the model (without removing any variable which is not statistically significant) led to the results shown here on the right.

OLS Regression Results

Dep. Variable:	Price Close%	R-squared:	0.163
Model:	OLS	Adj. R-squared:	-0.011
Method:	Least Squares	F-statistic:	0.9343
Date:	Wed, 08 Jun 2022	Prob (F-statistic):	0.511
Time:	16:03:40	Log-Likelihood:	-258.95
No. Observations:	59	AIC:	539.9
Df Residuals:	48	BIC:	562.8
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1.0902	3.530	0.309	0.759	-6.007	8.187
Price Close%	-0.0638	0.238	-0.268	0.790	-0.542	0.414
Price Target%	-0.5292	0.314	-1.687	0.098	-1.160	0.102
Revenue - Median%	-0.1589	0.178	-0.894	0.376	-0.516	0.198
Revenue%	-0.1692	0.132	-1.285	0.205	-0.434	0.096
Operating Profit%	0.0969	0.052	1.857	0.069	-0.008	0.202
Net Income after Tax%	-0.0267	0.033	-0.805	0.425	-0.093	0.040
Total Assets%	0.6085	2.232	0.273	0.786	-3.879	5.096
Total Liabilities%	0.2208	1.715	0.129	0.898	-3.228	3.670
Shareholders Equity%	0.0639	0.412	0.155	0.877	-0.764	0.892
Net Cash%	0.0586	0.071	0.831	0.410	-0.083	0.201

Omnibus:	2.764	Durbin-Watson:	2.006
Prob(Omnibus):	0.251	Jarque-Bera (JB):	1.883
Skew:	0.315	Prob(JB):	0.390
Kurtosis:	3.607	Cond. No.	251.

As we can notice all the variables are present in the model, but unfortunately none of these has statistical significance in predicting the next value of 'Price Close %'.

At this point we ran an algorithm that iteratively:

1. Removes the most statistically insignificant variable from the model
2. Re-compute the model
3. If other statistically-insignificant variables are present then it iterates again until all the remaining variables are statistically-significant, or until one only variable remains in the model. Then it stops.

We ran that algorithm to see if removing the most statistically-insignificant variables could lead to an increase in our model accuracy, and maybe in finding some statistical-significance in the other variables.

Between the performed iterations, some of these led to a slight improvement in accuracy and in the p-values, but statistically insignificant variables were still present, so the algorithm kept on iteratively removing those variables until it eventually remained with only one variable 'Operating Profits %', which still was not statistically significant (p-value of 0.167), but which was not removed since otherwise the model would be based only on the constant. The model is shown here on the right.

OLS Regression Results

Dep. Variable:	Price Close%	R-squared:	0.033
Model:	OLS	Adj. R-squared:	0.016
Method:	Least Squares	F-statistic:	1.958
Date:	Wed, 08 Jun 2022	Prob (F-statistic):	0.167
Time:	16:56:29	Log-Likelihood:	-263.20
No. Observations:	59	AIC:	530.4
Df Residuals:	57	BIC:	534.6
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	2.0458	2.863	0.714	0.478	-3.688	7.780
Operating Profit%	0.0268	0.019	1.399	0.167	-0.012	0.065

Omnibus:	7.026	Durbin-Watson:	2.089
Prob(Omnibus):	0.030	Jarque-Bera (JB):	6.212
Skew:	0.655	Prob(JB):	0.0448
Kurtosis:	3.901	Cond. No.	154.

As we can notice, the model is composed only of the constant and the previously cited variable.

At this point, we used both the models showed here (the initial one with all the variables and the final one with only one variable) to compute predictions for the variable 'Price Close %'. The results are shown in the section 'Results of the econometric analysis', together with the one obtained by the other OLS models.

Out-of-Sample OLS model

Next to the ‘In-Sample’ OLS model, we created the ‘Out-of-Sample’ one, meaning that it uses a part of the available data to build the model (80% of the data), while the remaining 20% is left for the testing and to generate out-of-sample prediction. The first run of the model (without removing any variable which is not statistically significant) led to the results shown here on the right.

OLS Regression Results

Dep. Variable:	Price Close%	R-squared:	0.258
Model:	OLS	Adj. R-squared:	0.052
Method:	Least Squares	F-statistic:	1.253
Date:	Wed, 08 Jun 2022	Prob (F-statistic):	0.293
Time:	17:13:41	Log-Likelihood:	-205.98
No. Observations:	47	AIC:	434.0
Df Residuals:	36	BIC:	454.3
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.2985	4.183	-0.071	0.944	-8.783	8.186
Price Close%	0.0983	0.293	0.335	0.739	-0.496	0.693
Price Target%	-0.8195	0.377	-2.173	0.036	-1.585	-0.055
Revenue - Median%	-0.1202	0.220	-0.545	0.589	-0.567	0.327
Revenue%	-0.0767	0.169	-0.455	0.652	-0.419	0.266
Operating Profit%	0.0753	0.060	1.252	0.219	-0.047	0.197
Net Income after Tax%	-0.0112	0.038	-0.297	0.768	-0.088	0.066
Total Assets%	0.3718	2.417	0.154	0.879	-4.530	5.274
Total Liabilities%	0.5649	1.815	0.311	0.757	-3.117	4.247
Shareholders Equity%	0.0075	0.523	0.014	0.989	-1.054	1.069
Net Cash%	0.0878	0.078	1.132	0.265	-0.069	0.245

Omnibus:	6.879	Durbin-Watson:	1.845
Prob(Omnibus):	0.032	Jarque-Bera (JB):	5.873
Skew:	0.684	Prob(JB):	0.0531
Kurtosis:	4.062	Cond. No.	244.

As we can notice, all the variables are present in the model and in this case the variable ‘Price Target %’ has statistical significance already from the beginning in predicting the next value of ‘Price Close %’.

At this point we ran the iterative algorithm previously described. The algorithm kept on iteratively removing variables until it eventually remained with only one variable ‘Operating Profits %’, which still was not statistically significant (p-value of 0.079), but which was not removed since otherwise the model would be based only on the constant.

The model is shown here on the right. As we can notice the model is composed only of the constant and the previously cited variable.

OLS Regression Results

Dep. Variable:	Price Close%	R-squared:	0.067
Model:	OLS	Adj. R-squared:	0.046
Method:	Least Squares	F-statistic:	3.236
Date:	Wed, 08 Jun 2022	Prob (F-statistic):	0.0788
Time:	17:13:46	Log-Likelihood:	-211.37
No. Observations:	47	AIC:	426.7
Df Residuals:	45	BIC:	430.4
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1.5758	3.300	0.478	0.635	-5.070	8.222
Operating Profit%	0.0404	0.022	1.799	0.079	-0.005	0.086

Omnibus:	9.429	Durbin-Watson:	1.894
Prob(Omnibus):	0.009	Jarque-Bera (JB):	8.664
Skew:	0.957	Prob(JB):	0.0131
Kurtosis:	3.871	Cond. No.	150.

At this point, we used both the models showed here (the initial one with all the variables and the final one with only one variable) to compute predictions for the variable ‘Price Close %’. The results are shown in the section ‘Results of the econometric analysis’, together with the one obtained by the other OLS models.

Recurrent-out-of-Sample OLS model

Next to the 'Out-of-Sample' OLS model, we created the 'Recurrent-out-of-Sample' one, meaning that it uses a part of the available data to build the initial model (80% of the data), while the remaining 20% is left for the iterative procedure of testing and re-training. In this way, we are able to generate out-of-sample predictions, while also gradually updating and re-training the model as we would do in the reality.

The results of the first run of the model (without removing any variable which is not statistically significant) correspond with the ones of the 'Out-of-Sample' model (since it uses only 80% of the data). For this reason, it will not be attached here.

In the same way, the results of the latest run of the model (without removing any variable which is not statistically significant) correspond with the ones of the 'In-Sample' model (since it uses 100% of the data). Again, for this reason, it will not be attached here.

One interesting pattern that we can notice observing the different runs in sequence (without removing any variable which is not statistically significant) is that at each run the R-squared of the model decreases.

```
The evolution of R-squared for the 13 iterations is:
 [0.258, 0.251, 0.238, 0.232, 0.197, 0.177, 0.175, 0.171, 0.169, 0.167, 0.157, 0.156, 0.163]
```

This means that as the number of data used to train the model increases, the correlation between the data decreases. It could be due to the fact that the latest 12 quarters (which is our training period) starts from the Q1 of 2019 up to now (Q1 of 2022). Just 3 quarters after the start of the testing period we had the Covid-19 pandemic, which caused strong movement in the stock markets without a correct response by the fundamental indicators, since obviously they could not predict a phenomenon like this one. This could be one of the reasons for which as we move on into our Recurrent-out-of-Sample model we assess decreasing values of R-squared.

After computing the recurrent results using all the variables as shown above, we decided to add our algorithm that removes the statistically insignificant variables. The algorithm has been implemented into the iterative procedure to generate models which are only composed of statistically significant variables (or that are composed of only 1 variable if it is the last one that remains).

Again, also in this case the first iteration of the model corresponds with the final results of the 'Out-of-Sample' model shown above (since it uses only 80% of the data). In the same way, the latest iteration of the model corresponds with the final results of the 'In-Sample' model shown above (since it uses 100% of the data). For this reason, the models will not be attached here.

Observing the R-square values for the sequence of runs (where each run is based only on statistically significant variables, or on the last remaining variable) we can notice that, in this case as well, the R-squared of the model decreases for each iteration.

```
The evolution of R-squared for the 13 iterations is:
 [0.067, 0.222, 0.068, 0.064, 0.049, 0.045, 0.045, 0.043, 0.044, 0.043, 0.042, 0.042, 0.033]
```

We can see how the values here are generally lower than the ones seen above, since the model is composed of less variables and so they are able to explain with a lower accuracy the changes in the dependent variable. The only outlier is the second iteration, in which we have an R-squared value of 0.222, which is more than the triple of all the other R-squared. In this case the R-squared resulted so high because the iterative algorithm has been able to retain 3 variables: 'Price Target %', 'Operating Profit %' and 'Total Assets %'. Unfortunately, we have three statistically significant variables only in that case, which leave us with low values of R-squared over all the other iterations.

The results of all the single iterations discussed here above can be checked using the attached codes, see 'Appendix: Codes' for more information.

Results of the econometric analysis

Once that the econometric analysis has been completed, we need to compare the predictions generated by our six models (three with all the variables, and three with a number of variables iteratively reduced by our algorithm) with the real price percentual change for each given period.

First of all, we saved all the generated predictions into a common data-frame, where for each period we can see the 'Real_Price%' change at the end of that period, and the predictions generated by all our models for the end of that specific period.

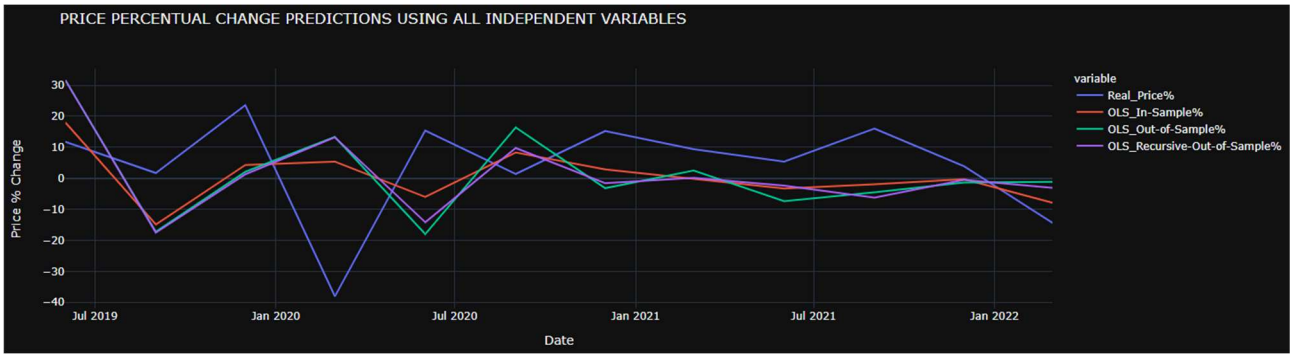
The data-frame is reported here below, where for each type of model we have two columns, the first one for the model with all the variables, and the second one for that same model but with only statistically significant variables (these columns are denoted by a final '_Sig').

Date	Real_Price%	OLS_In-Sample%	OLS_In-Sample_Sig%	OLS_Out-of-Sample%	OLS_Out-of-Sample_Sig%	OLS_Recursive-Out-of-Sample%	OLS_Recursive-Out-of-Sample_Sig%
2019-06	11.749	17.99536	14.585677	31.606671	20.468103	31.606671	20.468103
2019-09	1.713	-14.919447	1.610438	-17.254116	0.919927	-17.507407	-15.17535
2019-12	23.577	4.322158	1.914863	2.110421	1.378565	1.289806	1.351007
2020-03	-38.064	5.403216	5.482439	13.383961	6.753387	13.215737	6.666419
2020-06	15.402	-5.968736	0.158039	-17.961397	-1.268222	-14.20475	-1.062559
2020-09	1.381	8.313606	4.58665	16.406679	5.403814	9.754506	4.606015
2020-12	15.240	2.844068	1.874423	-3.184148	1.31764	-1.605106	1.368834
2021-03	9.370	-0.254441	3.737763	2.483133	4.124902	0.111054	3.78818
2021-06	5.377	-3.284648	1.171286	-7.354531	0.258311	-2.338557	0.930727
2021-09	16.016	-1.962519	2.058869	-4.535488	1.595522	-6.235539	2.04321
2021-12	3.872	-0.264645	2.113254	-1.341746	1.677456	-0.478006	2.363094
2022-03	-14.384	-7.865874	7.941166	-1.128815	10.457645	-3.111684	8.968937

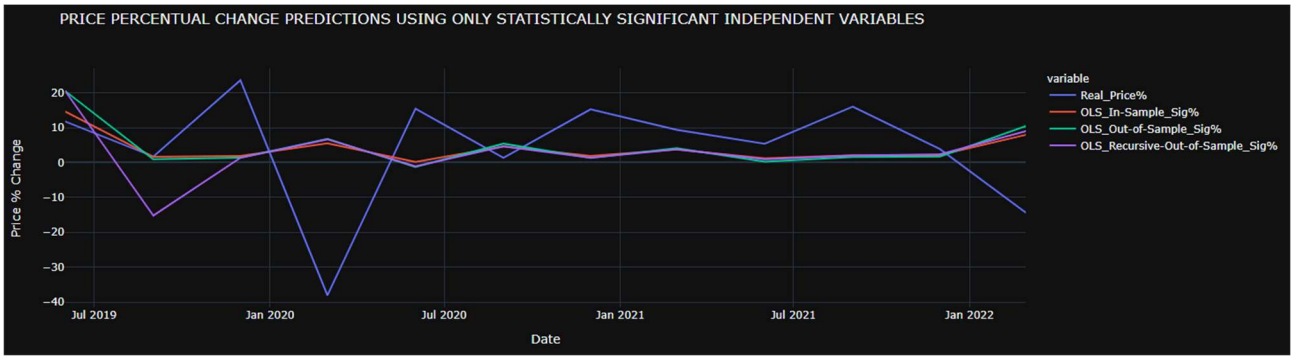
Already from the first glance it is possible to notice how the columns related to the models with all the variables show much bigger values in absolute sense (both positive and negative). This is due to the fact that having a lot of different variables in the model (of which the most are not statistically significant) led us to generate predictions that are very noisy. The columns related to the models with a reduced set of variables show instead more steady and conservative predictions, which only seldomly go beyond the threshold of 10%. This is due to the fact that these predictions are (in most of the cases) based only on the variable 'Operating Profit %', which as we saw in the previous sections, was the only variable to be statistically significant in the models (or at least it was the variable that was left as last one in the procedure of remotion of the insignificant variables). Therefore, if the value of this variable is steady, also the generated predictions that rely on it will be steady and will have a low variance.

The results discussed here above have been represented graphically through the following charts.

In this first chart the predictions generated by the models which uses all the independent variables are compared with the 'Real_Price%' change. We can notice how the predictions are not excessively accurate, but anyway they are able to follow the general trend of the real changes.



In this same way, in this second chart the predictions generated by the significant models (which uses only statistically significant variables) are compared with the 'Real_Price%' change. As we have already mentioned above, we can notice how their predictions tend to be much more steady and less noisy than the previous ones. This is good from the point of being less noisy, but this also means that for these models it will be impossible to follow the real changes in high variance scenarios.



Finally, after comparing the predictions both using tables and charts, we computed some metrics that allow us to understand in a more precise way how much our predictions are accurate from the mathematical point of view.

To accomplish this task, we decided to use two metrics that are often used in this field. In particular, we choose to use the Mean Squared Error (MSE) and the Root Mean Squared Error (RMSE), which is just the square root of the first one.

The results obtained by this procedure are attached here on the right, both for the predictions based on all variables, and for the predictions based on few selected variables (through algorithm).

From the first attachment we can notice how the MSE has its lowest value for the 'In-Sample' model, and this is what we expected since this model has been trained on all the available data, and consequently cannot be used in the reality to generate real predictions for the future. The 'Recurrent-out-of-Sample' model is the second best, while the 'Out-of-Sample' is the worst one, this is in line with what we expected since, the former has been updated retrained different times, while the latter has never been retrained.

```
Metrics with all variables:

In sample MSE: 315.393
Out of sample MSE: 531.241
Recurrent out of sample MSE: 492.241

In sample RMSE: 17.759
Out of sample RMSE: 23.049
Recurrent out of sample RMSE: 22.187
```

Looking at the second attachment we can notice how the MSE again has its lowest value for the 'In-Sample' model, but here the second best is the 'Out-of-Sample' model. While the 'Recurrent-out-of-Sample' model is the worst one. This makes sense and is aligned with what we noticed before, descending R-squared values as the iteration kept on rising. Generally, we would expect that as the iteration increases, the model becomes more stable and reliable, but in this case, it is the opposite. As the iterations increase, the model becomes less accurate and, as we said before, this could be due to the unusual move that the Covid-19 pandemic brought on the stock markets from 2020 moving on.

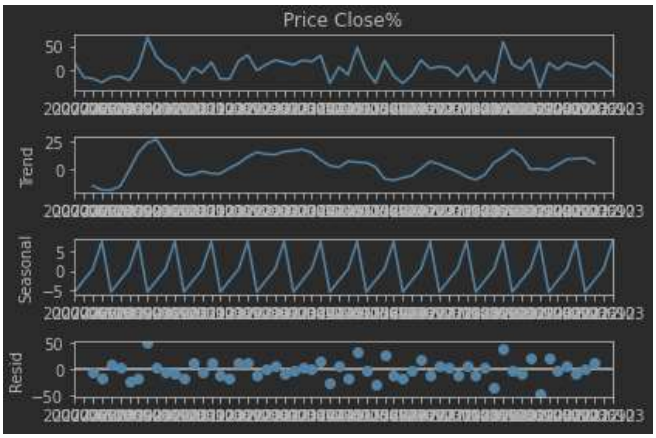
```
Metrics with selected variables:

In sample MSE: 295.051
Out of sample MSE: 329.126
Recurrent out of sample MSE: 343.645

In sample RMSE: 17.177
Out of sample RMSE: 18.142
Recurrent out of sample RMSE: 18.538
```

Neural network analysis

In this section the results obtained by the forecasting model based on LSTM - RNN (which has been introduced in the section 'Description of forecasting models – LSTM-RNN') are presented and discussed.

Before running the neural network analysis, we decided to perform a seasonality analysis to extract the following information from our input data:

- Trend
- Seasonality
- Residual



The obtained results are attached on the right.

The first window shows the input data 'Price_Close%' (that we also called 'Real_Price%'), while the second window puts together these data to give us some information about general trend, which in this case seems more or less sideways instead of trending.

The information in which we are interested the more is the one showed in the third window, here we can see the seasonality of the input date. We can see how, according to this computation, there is a good seasonal pattern in this data, meaning that our LSTM-RNN should be able to extract this recurrent behavior, and to generate some interesting predictions. Obviously, trend and seasonality cannot explain in full the behavior of the price, for this reason in the fourth window we have the residual component, which represents what could not be explained by the previous two data.

The econometric analysis is composed of three models that have been ran sequentially. These models have been trained with some common specifications:

1. First, they all uses the same number of features (equal to the number of independent variables introduced in the section 'Data description – descriptive statistics'). Therefore, in this forecasting models we will not have the distinction between models based on all the independent variables and model based only on statistically significant ones, because for neural networks all the data are important in the same way.

2. Secondly, we will have the same number of inputs for all the models, that is the number of previous values (for each one of the input features) that will be used to compute the next prediction. The number of inputs used in the following models is equal to 4 quarters.

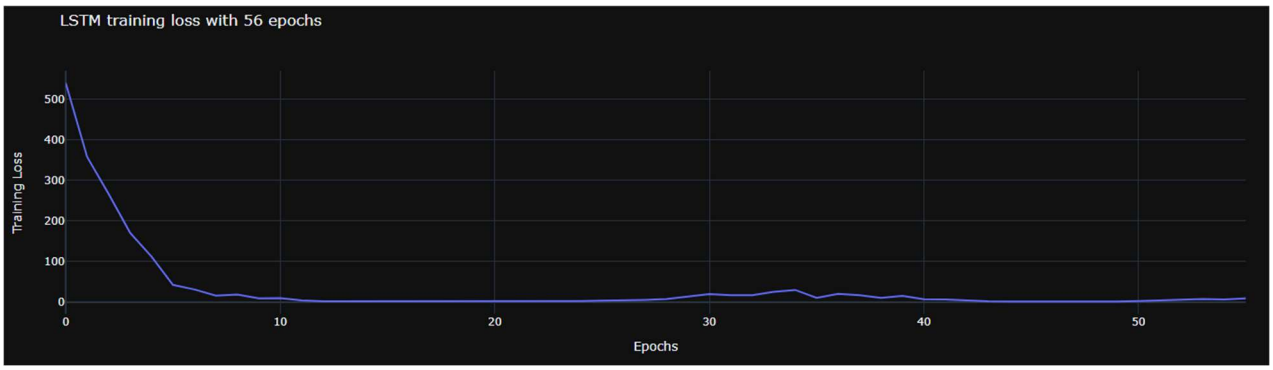
3. In addition, all the LSTM-RNNs have been trained using the Adam optimizer and they rely on MSE as loss metric.

To have a more ordered presentation of the results, this section is divided in three parts, where the results of each model are discussed in detail.

In-Sample model

After performing the seasonality analysis, we run the training procedure for the first LSTM-RNN model which is 'In-Sample', meaning that it uses all the available data to build the model.

The evolution of the training loss during the optimization procedure is showed here below.



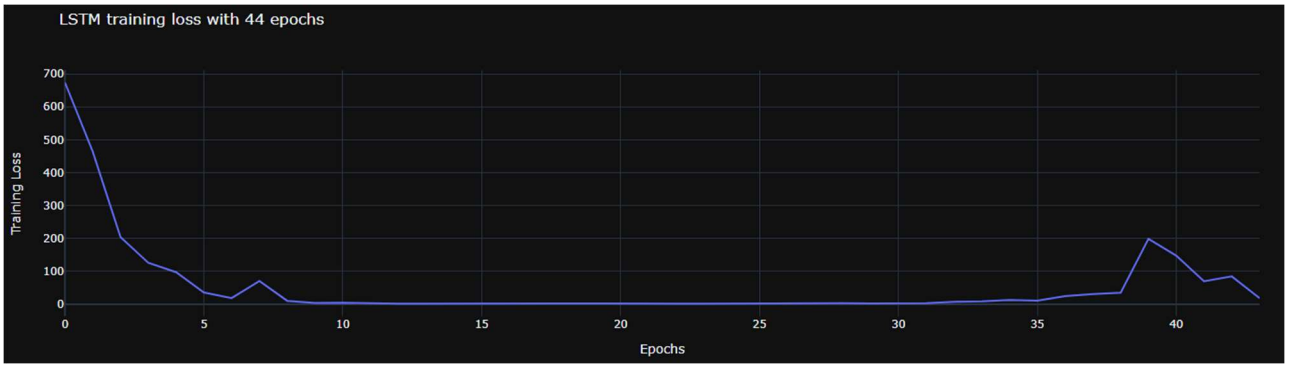
As we can notice the big part of the learning for our model happened in the first 10 iterations, while for the remaining 46 iterations we got initially a slight improvement, and then an increase in the loss around the 35th iteration, to then see the value reduce again asymptotically toward zero.

Once the training procedure has been terminated, we used that model to compute predictions for the variable 'Price Close %'. The results are shown in the section 'Results of the neural network analysis', together with the ones obtained by the other LSTM-RNN models.

Out-of-Sample model

After performing the In-Sample training and getting the respective prediction we run the training procedure for the second LSTM-RNN model which is 'Out-of-Sample', meaning that it uses a part of the available data to build the model (80% of the data) while the remaining 20% is left for the testing and to generate out-of-sample prediction. This model is never retrained.

The evolution of the training loss during the optimization procedure is showed here below.



As we can notice, also in this case the big part of the learning for our model happened in the first 10 iterations, while for the remaining 34 iterations we got initially a slight improvement, and then an increase in the loss around the 40th iteration, to then see the value reduce again toward zero.

Once the training procedure has been terminated, we used that model to compute predictions for the variable 'Price Close %'. The results are shown in the section 'Results of the neural network analysis', together with the ones obtained by the other LSTM-RNN models.

Recurrent-out-of-Sample model

After performing the 'Out-of-Sample' training and getting the respective prediction, we run the training procedure for the third and latest LSTM-RNN model which is 'Recurrent-out-of-Sample', meaning that it uses a part of the available data to build the initial model (80% of the data) while the remaining 20% is left for the iterative procedure of testing and re-training. In this way, we are able to generate out-of-sample predictions while also gradually updating and re-training the model as we would do in the reality.

The first iteration of this model will have 44 epochs (as the 'Out-of-Sample' one), while the last iteration will have 56 epochs (as the 'In-Sample' one). For this reason, the charts will not be attached here, but the numerical evolution of the training losses is available for each iteration and it can be seen in detail using the attached codes, see 'Appendix: Codes' for more information.

Once the training procedure has been terminated, we used that model to compute predictions for the variable 'Price Close %'. The results are shown in the section 'Results of the neural network analysis', together with the ones obtained by the previous LSTM-RNN models.

Results of the neural network analysis

Once that all the neural network analysis' have been completed, we need to compare the predictions generated by our three models with the real price percentual change for each given period.

First of all, we saved all the generated predictions into a common data-frame, where for each period we can see the 'Real_Price%' change at the end of that period, and the predictions generated by all our models for the end of that specific period.

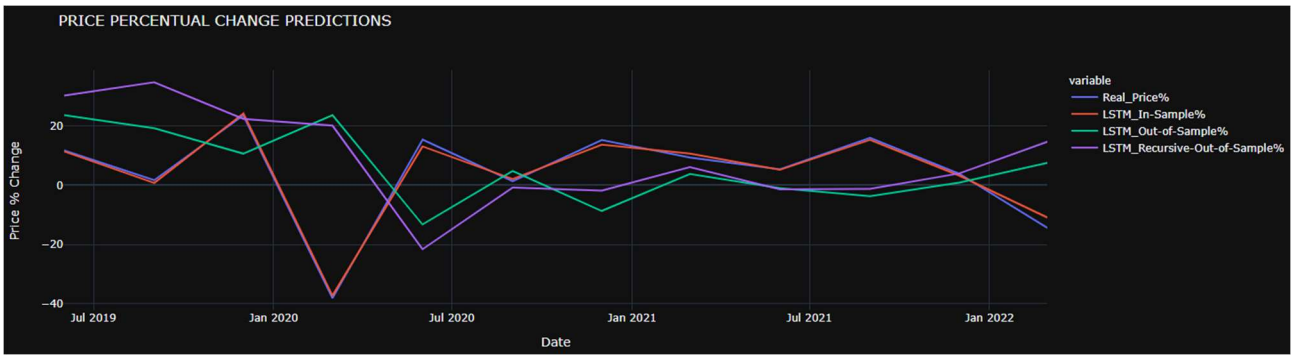
The data-frame is reported here below, where for each LSTM-RNN model we have one column.

Date	Real_Price%	LSTM_In-Sample%	LSTM_Out-of-Sample%	LSTM_Recursive-Out-of-Sample%
2019-06	11.749	11.393000	23.643999	30.264999
2019-09	1.713	0.774000	19.247999	34.730999
2019-12	23.577	24.240999	10.668000	22.391001
2020-03	-38.064	-37.201000	23.657000	20.156000
2020-06	15.402	13.107000	-13.216000	-21.575001
2020-09	1.381	2.099000	4.827000	-0.796000
2020-12	15.240	13.664000	-8.643000	-1.823000
2021-03	9.370	10.675000	3.772000	6.125000
2021-06	5.377	5.250000	-0.982000	-1.363000
2021-09	16.016	15.312000	-3.652000	-1.237000

Already from the first glance it is possible to notice how the results generated by all the three models are more homogeneous between them, with predictions that go from 0% to 37% (in absolute value). This is due to the fact that, differently from the OLS models seen in the previous section, here all the models exploit all the variables, and this allows the models to have more data and more relations to use in the generation of the predictions.

Another characteristic of these predictions that we can notice is that the latest two models tend to have much lower (in absolute sense) and steady prediction in the second part of the data-frame, while the 'In-Sample' model keeps on predicting important changes in the prices also for these periods.

The results discussed here above have been represented graphically through the following chart, where the prediction generated by all the models are compared with the 'Real_Price%' changes.



We can notice how the predictions, also in this case, are not excessively accurate, indeed, they are able to follow the general trend of the real changes in price, but in different occasion the prediction is directed in the opposite way with respect to the value 'Real_Price%'.

One interesting thing that can be noticed is how the prediction 'LSTM_In-Sample%' is almost completely overlapped with the 'Real_Price%'. This is not a good sign because it clearly shows us that the training data have been overfitted, and so that we are able to generate a prediction with a really high accuracy only if we have already witnessed that scenario (which is not possible in the

reality). This offers us some valuable hints, telling us that improvements need to be done on the network architecture to reduce this overfitting. Indeed, improvements over this aspect could also lead to better generalization capabilities and so better results in the 'Out-of-Sample' and 'Recurrent-Out-of-Sample' scenario.

Finally, after comparing the predictions both using tables and charts, we computed some metrics that allow us to understand in a more precise way how much our predictions are accurate from the mathematical point of view.

To accomplish this task, we used again the Mean Squared Error (MSE) and the Root Mean Squared Error (RMSE), which is just the square root of the first one.

The results obtained by this procedure are highlighted here on the right.

```
Metrics with all variables (LSTM-RNN):

In sample MSE: 2.106
Out of sample MSE: 564.551
Recurrent out of sample MSE: 640.457

In sample RMSE: 1.451
Out of sample RMSE: 23.76
Recurrent out of sample RMSE: 25.307
```

From the attachment we can notice how the MSE has its lowest value for the 'In-Sample' model, and this is what we expected since this model has been trained on all the available data. The point here is that the MSE and the RMSE are excessively low compared with the ones of the other training procedures. This is in accordance with the comments that have been done above, where the first sign of overfitting has been witnessed.

Astoundingly, the 'Out-of-Sample' model is the second best, while the 'Recurrent-out-of-Sample' is the worst one. This is not in line with what we expected since, the former has never been retrained, and so it generated predictions for the latest 20% of the dataset while being trained only on the first 80% of it. The latter instead has been updated and retrained different times and so theoretically it should have better performances with respect to the 'Out-Of-Sample'.

Unfortunately, it is difficult to make other considerations on the results that we have got in the neural network analysis since, as it is known, neural networks suffer from the so called 'black-box problem' which is also called 'explainability-problem'. This is due to the fact that neural networks are based on automatic algorithm which studies the inputs and the outputs looking for relationships and patterns that connect them. The training procedure starts from a set of initial random parameters and optimizes the values of these parameters to get a lower loss and so better predictions. For each time that the training procedure is started we do not know which are the starting parameters (and for this reason each time we will obtain a different model, with different accuracy and different results). Furthermore, we do not know which are the most important features in the neural network, since it is a completely free choice let to the training algorithm. For these reasons we are not able to explain why we got these particular results, and how we could fine tune them. The only approach that we can adopt is trying different architectures and input options to see if we are able to get better and more accurate results for our specific task.

Conclusion

In this report all the procedure followed from the extraction of the data to computation of the final results has been treated and explained in great detail for the Italian company with ticker 'AZMT.MI'.

The same procedure has been then repeated also for other 4 companies and the corresponding results have been attached together with the main analysis over 'AZMT.MI'.

The single results obtained by the 'Econometric analysis' and by the 'Neural network analysis' have already been discussed alone, but they have not been fully compared in the previous sections.

To compare in an analytical way the results obtained by the two analysis we will now rely on the metric results that have been presented previously (MSE and RMSE), which are now all represented in a singular table to give a more complete overview of the final results.

Model	Type of training/testing	MSE	RMSE
OLS	In-Sample	315,39	17,76
	In-Sample-Sig	295,05	17,18
LSTM-RNN	In-Sample	2,11	1,45
OLS	Out-of-Sample	531,24	23,05
	Out-of-Sample-Sig	329,13	18,14
LSTM-RNN	Out-of-Sample	564,55	23,76
OLS	Recurrent-out-of-Sample	492,24	22,19
	Recurrent-out-of-Sample-Sig	343,64	18,54
LSTM-RNN	Recurrent-out-of-Sample	640,46	25,31

Looking at the table we can easily see how:

- Generally, the OLS models are always better than the LSTM-RNN models (excluding the 'In-Sample' case, in which due to overfitting the LSTM-RNN exhibits an extremely low MSE and RMSE).
- Between the OLS models, the significant ones (the one based only on statistically significant variables, or on the latest remaining variable) are always better by a great margin with respect to the not significant ones.

If to these previous considerations we add the 'explainability-problem' (that has been discussed in the previous section) to which the neural networks are subjected, we can easily avow that the OLS model is a better forecasting model. This due both to the better performances, and also to the transparency that it guarantees. Obviously, as previously stated, the performance of the LSTM-RNN could be improved through trials on different architectures and settings, but as far as the results presented in this report, there is no doubt that the OLS model is superior and that it offers more advantages and a greater accuracy with respect to the used neural network.

Appendix: Codes

In this report some of the results for the company with ticker ‘AZMT.MI’ have already been shown.

To see the full results for that company, open the file ‘AZMT.MI – As from Report’ with an editor that supports Python Notebook (like PyCharm, Jupyter-Notebook). Do not run the code otherwise the results will be erased (and they will not be re-computed since the connection to Refinitiv Eikon Workspace will not be established). The results are already computed and showed under each section of the code.

Notice that the plot of charts is based on the library ‘Plotly’, which allows to create dynamic charts on which it is possible to interact (like selecting which lines we want to see, zoom-in and out, …). Be sure that Plotly is correctly installed on the IDE on which the Notebook will be opened, otherwise almost all the charts will not be shown. This could create problems when using online solutions like Google Colab, for this reason a local IDE with Plotly installed is a better alternative to load the codes.

Furthermore, the analysis carried out for the other companies has been attached as well. Please open the respective folder and file ‘COMPANY.TICKER’ to see the results for that specific company.