Statistical Models and Machine Learning Algorithms to Forecast Future Prices in the Stock Market

Ana Rita Silveira da Costa
Instituto Superior Técnico,
Avenida Rovisco Pais, 1049-001 Lisboa, Portugal
Email: ana.rita.costa@tecnico.ulisboa.pt

***Abstract*—Stock prices can be interpreted as time series that can be forecasted in order to improve the returns of a trader. Several methods including statistics and artificial intelligence are being developed in order to turn this prediction more accurate and reliable. Forecasting a time series is a regression problem since it is a continuous variable that is being forecasted. This thesis applies a statistical method, ARIMA, and two machine learning models, K-Nearest Neighbors and Support Vector Regression, in order to forecast future stock prices. The presented work shows predictions in a daily, weekly, and monthly range using different stocks with different characteristics. The three studied models are compared in each of these situations considering the error of the forecasted values, the returns of a strategy that relies on these predictions and the risk and accuracy of that strategy. The data sets that were used for the training period correspond to 4 years of data of a clear trend stock and a sideways stock in order to present data with different characteristics. The test period corresponds to 1 year of the same stocks. The best result obtained was by the ARIMA model in a monthly forecast, reaching returns of 40% and an accuracy of 90.9%. The K-Nearest Neighbors and Support Vector Regression algorithms are more precise in a sideways stock being superior to the ARIMA solution. Both machine learning approaches benefit from the introduction of a retraining during the test period, in some cases decreasing the error in 10 times. *Index Terms*—Stock Market, Forecast, ARIMA, K-Nearest Neighbors, Support Vector Regression**

I. Introduction

Stock market refers to the collection of markets and exchanges where trading securities takes place and it is considered one of the most vital components of a free-market economy. It is known that the first stock exchange happened in 1531 in Belgium, even though the concept of "stock" has changed over time, in the beginning it was similar to a financier partnership that produced income like stocks do. The stock exchange started in London officially in 1973, and 19 years after was the first New York Stock Exchange. Stock market news are everywhere: newspapers, TV news, and there are complete websites dedicated to this matter. The reason why the stock market is so important is that allows companies to raise money by offering part of their equity, letting the investors participate in their financial achievements. Also, stock market serves as an economic barometer since share prices rise and fall depending largely on economic factors. For example, share prices tend to increase when the economy shows signs of growth, and in the other hand tend to brutally decrease, sometimes leading to a stock market crash, during economic recession, depression or financial crisis. Having a good knowledge about the stock indexes serves as a reference to the general trend in the economy, influencing decisions from the average family to the wealthiest executive

Going back to the fact that stock market gives the opportunity to small investors participate of the company financial achievements, the idea of owning shares of a big company is very appealing, leading loads of people to invest in this market. Taking this into account, the investor wants to own shares of a wealthy company with expectations of future returns, and not of a company that will decrease its value in the future. This lead to the need of pondering the decision of which stock one should invest. To solve this impasse, the concept of predictive analysis, also known as forecast, stated to appear in the financial field. The definition of "forecast" is to predict or estimate a future event or trend based on past and present information. In this case, this future event is a future value of a share, in order to decide either to buy it or not. The idea of learning from the past in order to predict the future gained popularity in the last years and a lot of techniques concerning this topic are now being used and tested to make a good prediction helping those involved in this market.

There are statistical models to do this predictive task and they started to give very good results. Statistics deals with the collection, classification, analysis, and interpretation of numerical data and provides tools for forecasting through statistical models. Statistical models started to be almost all from the class of linear models but the data behavior, especially financial data, caused a particular interest in non-linear models. Also because of the non-linearity of the data, artificial intelligence methods gained a lot of popularity in the forecasting field. The term Machine Learning is being widely used in financial computation, although the origins of machine learning are from the 50s. In 1952, Arthur Samuel wrote the first computer learning program, a checkers game where the IBM computer improved the more it played, studying which moves made up winning strategies and using it into its program to win. It was one of the first time that was created a kind of an artificial intelligence. In 1957, Frank Rosenblatt designed the first neural network, the beginning of one of the most powerful

machine learning algorithms used nowadays. During the 90s, machine learning shifts from a knowledge-driven to a data-driven approach. Programs were being created for computers to analyze large amounts of data and draw conclusions or learn from the results. Today we can even talk about deep learning, the ability to see and distinguish objects and text in images and videos [1]. Computers abilities to see, understand, and interact with the world around them is growing at a remarkable rate, and a lot of traders are using this ability to forecast in the stock market.

This work is then motivated by the challenge of conducting predictive analysis in the stock market using statistic and machine learning algorithms to improve the returns of a trader.

Briefly, the main objectives are: 1) study time series and their role in the stock market; 2) introduce the use of forecast tools in financial computation; 3) implement one statistical algorithm to forecast future prices of a share in the stock market; 4) implement two machine learning algorithms to forecast future prices of a share in the stock market and 5) compare the behavior of the three different models for a daily, weekly and monthly forecast.

II. Related Work

This section presents some works related to the forecast topic in the financial field.

A. Statistical Approach

Rounaghi and Zadeh [2] tried to model and forecast the stock value of 350 firms listed in London Stock Exchange and S&P 500 from 2007 until the end of 2013 using ARMA model. They applied monthly and yearly forecasting in both London Stock Exchange and S&P 500 Index. To measure the quality of the proposed ARMA model, researchers used MAE (Mean Absolute Error), MAPE (Mean Absolute Percentage Error), MDAPE (Median Absolute Percentage Error), SMDAPE (Symmetric Median Absolute Percentage Error), and MASE (Mean Absolute Scaled Error). The results show that medium and long-term forecasting of time series is possible in S&P 500 and London Stock Exchange at the error level of 1%. Both markets are considered efficient and with financial stability during periods of boom and bust. Vantuch et al. [3] also tried to predict future prices of the Microsoft stock (MSFT) using the ARIMA model. The AIC and BIC values of this model were compared with models chosen without the help of the Genetic Algorithm. A Genetic Algorithm is used to find the best model. The results considering GA-ARIMA $(12, 2, 8)$ showed a BIC of 458.6266 and an AIC of 400.4396 while an ARIMA $(2, 1, 3)$ without the Genetic Algorithm showed a BIC of 434.0470 and an AIC of 408.1205. The GA-ARIMA model did not show significantly better results, as it is observable, being the GA-ARIMA AIC only slightly inferior to the AIC of the ARIMA. Also, the tests with PSO optimization did not prove that the estimation of the coefficients by PSO has significant importance in the ARIMA results, maybe because of the low number of PSO iterations. PSO is more popular in parallel computing, where it can obtain better results. Wu & Lu [4] used Neural Networks to predict future values of the S&P 500 Index. Their paper compares Neural Networks performance against an ARIMA model and the Neural Network model outperformed the ARIMA model only in a stable market. When dealing with volatile markets, the Neural Network system only showed an accuracy of 23% and the ARIMA's accuracy was 42%. The same comparison was conducted by Kamruzzaman & Sarker [5] but applied to exchange rates. The Neural Networks were trained with back-propagation, scaled conjugate gradient, and back-propagation with Bayesian regularization. The algorithm used technical indicators and outperformed the ARIMA model, having an impressive accuracy of 80%. Considering this results, Gerlein et al. [4] stated that an accuracy of 80% must be considered with care since only the best results are reported and in general machine learning techniques do not present such high levels of accuracy.

B. Machine Learning Approach

Mandziuk et al. [6] used Neural Networks to train data in order to do a prediction for a 5-day period of EUR/USD trading. In order to choose a suitable subset of input variables to train with Neural Networks, Mandziuk et al. [6] used GA to perform the selection process out of a large pool of diverse data sources available. Neuro-evolutionary model proved to have more than 56% of correct decisions, 30% more than wrong ones. It has way more activity than the rest of the algorithms in comparison. The weighted version has more than 111% of profit corresponding to more than 25% of annual profit. Yoo et al. [7], in their survey on machine learning techniques for stock market prediction, talk about the use of Neural Networks, Support Vector Machines, and Case Based Reasoning. The researchers admit that Neural Networks are gaining a lot of popularity in this field of study but they have some related issues, such as the black box problem, meaning that it is not known the significance of each variable neither it is possible to understand how the network produces future prices. Another problem with Neural Networks is the overfitting problem since Neural Networks fit the data too well and lose the ability of generalization. This can be due to many nodes in the networks or long periods of training. Yoo et al. [7] also assume Support Vector Machines to be very interesting when applied to classification and regressions tasks in time series prediction related to financial applications. Unlike Neural Networks, Support Vector Machines are resistant to overtraining achieving a high generalization performance and one of the main advantages is that it is equivalent to solving a linear quadratic problem, having a unique and globally optimal solution while Neural Networks have the danger of getting stuck at local minima. Kim [8] also stated some of the limitations of Neural Networks, such as the overfitting problem and the local optimal solution, and tried Support Vector Machines to solve the problem of predicting future prices in the stock market. In this solution, Support Vector Machine outperforms Backpropagation Neural Networks and Case Based Reasoning, with a hit ratio of 57.8313%, Neural

Networks with 54.7332%, and CBR with 51.9793%. This study ends with a proposal of a Support Vector Machine hyper-parameters optimization and also with the conclusion that low accuracies are a common and expected result when dealing with capital markets since there is no single model perfectly suited in all market conditions. Tay and Cao [9] compared Support Vector Machines against back propagation Neural Networks to forecast future contracts and the Support Vector Machines solution obtained better accuracy (47.7%) than the Neural Networks solution (45.0%). The same happened in the work reported by Chen and Shih [10] where the two techniques were compared when applied to six Asian indices, with an accuracy of 57.2% for the Support Vector Machines and 56.7% for the Neural Networks. Putting Neural Networks aside and entering with K-Nearest Neighbors, Chen and Hao [11] proposed a hybridized framework of the feature weighted Support Vector Machine and feature weighted K-Nearest Neighbors to predict stock market indices. The proposed model was applied to the Chinese stock market (the Shanghai and Shenzhen stock indices) and the results were slightly better than for regular SVM-KNN approach and sometimes even equal. For Shanghai composite index, FWSVM-FWKNN is better than SVM-KNN for time horizons of 1, 5, 15, and 30 days, even though the results have very similar values of MAPE and RMSE. For a time horizon of 10 and 20 days, the values of MAPE are the same. When observing the Shenzhen composite index results, the SVM-KNN and FWSVM-FWKNN are even more similar.

III. Implementation

The proposed solution intends to predict the future values of a share taking into consideration the past values of the same share. One statistical algorithm and two machine learning algorithms will be used in order to solve the proposed problem of forecasting in the stock market.

This section describes the architecture design of the proposed solution for forecasting future prices in the stock market. The module that is developed in this work can be divided into three layers as shown in figure 1.

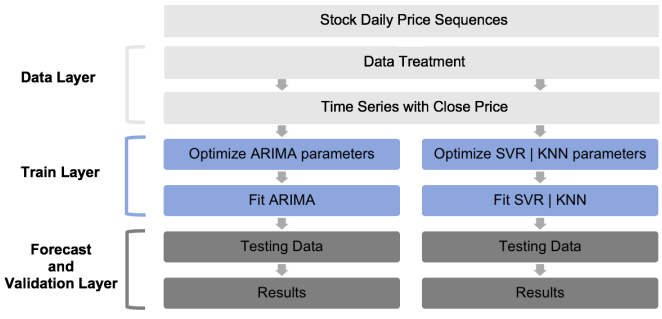

Fig. 1. Proposed architecture

A. Data Layer

Data Layer is the one responsible for treating the price sequences that will serve as an input to the Train Layer. The two main reasons for this layer to exist are: 1) transformation of the .csv files that contain the daily open, close, highest and lowest prices of a stock into a time series with the close price; 2) separation of the data into training and test sets. When having the data in a time series format, this data is divided into training and test datasets. The reason behind this separation is because the algorithms cannot be evaluated with the same data that they were trained since it would lead to an overfitting problem and lost of generality. Overfitting is the non-ability of a model to generalize to out-of-data samples. It usually happens when too many features are considered and the model fits the training set with a cost function of almost zero. In the opposite, underfitting is when the model does not follow the behavior of the training set. The training dataset is then used for learning and tunning the hyperparameters of each of the algorithms, and the test dataset is used to assess the model performance of the algorithm in question. Taking this into consideration, the percentage chosen for the training set was 80% and for the testing set 20%. This procedure is illustrated in figure 2.

Fig. 2. Data separation

The Data Layer ends when a time series is ready to be used by the algorithms in question. After this steps, the training dataset enters the Train Layer where each of the algorithms is trained using this set of data.

B. Train Layer

The Train Layer, as the name suggests, is responsible for training each of the three algorithms developed in this work using the training dataset. Training a model is applying it to a training dataset so that the model can perceive hidden patterns and mapping relationships that help it perform well during the test period. This layer is divided into two sub-layers, the statistical and the machine learning one. The reason behind this separation is the existence of common steps in the implementation of the machine learning algorithms that do not make sense in the implementation of a statistical model and vice versa. In the statistical sub-layer the ARIMA model is implemented, and in the machine learning sub-layer, the Support Vector Machine and K-Nearest Neighbor are implemented.

1) Statistical Sub-Layer: The only statistical method implemented is an ARIMA (p,d,q) model. The process of training the ARIMA (p,d,q) with the training dataset is described in three steps.

Step 1 is to check for stationarity. Price sequences are not usually stationary and they usually need some transformations to show a stationary behavior. To check for stationarity, a Dickey-Fuller Test is conducted. For this test, the regression model is estimated, $y'_t = \alpha + \beta t + \phi y_{t-1} + \gamma_1 y'_{t-1} + \gamma_2 y'_{t-2} + ... + \gamma_k y'_k$, where y'_t denotes the first-differenced series and k the number of lags to include in the regression. If the original series is non-stationary, then the coefficient ϕ should be approximately zero. The null-hypothesis states that the data is non-stationary. If $\phi < 0$, the hypothesis is rejected meaning that it is stationary. Based on this stationarity test, two things can be done: 1) stabilize the variance and 2) stabilize the mean. To stabilize the variance, a non-linear transformation should be applied. In this case, this transformation is a logarithmic one. To turn the mean a constant value, a first-difference is applied, meaning computing the differences between consecutive observations. This process should be repeated until the series is stationary. Once the series is stationary, the model can be trained using all the past prices until instant *t*.

Step 2 is finding the best order of p, d, and q. When having a series with constant mean and variance, the next step is the choice of the right ARIMA model. Selecting the right order is finding the best combination of the three parameters (p, d, q) for a specific training data set. The values that these parameters can take are described in table I. The search for

TABLE I
ARIMA PARAMETERS

Parameters	Range
p	[1,5]
d	[1,2]
q	[1,5]

the best ARIMA parameters can be done in various ways, but since the values of p and q are not usually greater than 5, a brute force search is used to find the best possible combination of parameters. For each set of parameters, a model evaluation is taken using the mean squared root (MSE) as choice criteria. The best combination is the one with the lowest MSE.

Step 3 it to fit the model. After choosing the best ARIMA model based on the lowest values of MSE, the model is fitted to the data and future prices can be calculated. ARIMA uses all the information (prices) from the past as inputs.

2) Machine Learning Sub-Layer: In this sub-layer, two algorithms are implemented: Support Vector Regression (SVR) and K-Nearest Neighbors (KNN). As the two used techniques are machine learning algorithms that use supervised learning, there are some implementation steps in common. These implementation steps are described below.

Step 1 is choosing features and targets. Initially, the dataset that enters this layer is in a time series format. Taking into consideration that the two algorithms use supervised learning, it is necessary to transform the dataset into a format that matches the supervised learning problem format. The short idea of supervised learning is to have input variables called features (X), output variables called targets (y), and an algorithm to learn the mapping functions from the input to the output. The goal of this mapping function is to maximize the mapping so well that when having new input data (X), it is possible to predict the output variables (y) for that data. Time series data can be phrased as supervised learning, having features and targets, using previous time steps as input variables (features) and use the next time step as the output variable (target). The use of previous time steps to predict the next time step is called the sliding window method and the number of previous time steps is called the window width. This sliding window method is the basis of turning any time series into a supervised learning format, and once a time series is prepared this way, any of the standard linear and nonlinear machine learning algorithms can be applied. This window width corresponds then to the number of features.

Step 2 is tuning the hyper-parameters. For each iteration along the window width range, the machine learning algorithm is trained to find the best set of hyper-parameters for that window width. The search for the best set of hyper-parameters is done by a RandomizedSearchCV, implemented by *sklearn.model_selection.RandomizedSearchCV*. This method is similar to a GridSearchCV, where a python dictionary is created using combinations of the algorithm hyper-parameters, and then the dictionary combinations are tested and scored considering the resulted MSE. The difference between RandomizedSearchCV and GridSearchCV is that in RandomizedSearchCV not all the dictionary combinations of hyper-parameters values are tried out, but rather a fixed number of parameter settings is sampled from the specified distributions reducing the computational effort. The number of parameter settings that are tried out can be chosen.

When calculating the best hyper-parameters, it is important to avoid train and validate the same data since this can cause overfitting. The cross-validation method is used to split the data into training and validation sets. The logic behind this method is simply dividing the data into k-folds, using "k-1" folds for training and the remaining one for validating and cost calculation (MSE). The process is repeated "k" times and the validation folds alternate in each one of the k rounds. Finished the "k" iterations, the average cost of the validation sets is calculated. In the end, there is an average cost for each set of parameters, and the one with the lowest cost is the chosen one. The illustration of this process is described in figure 3. It is important to notice that these training sets are not the same as the one from figure 2. The training set from figure 2 is represented in figure 3 as "All training dataset" and the training and validation sets are partitions of that "All training dataset".

Summarizing and concluding, for each of the window widths the best set of parameters is calculated. At the end of

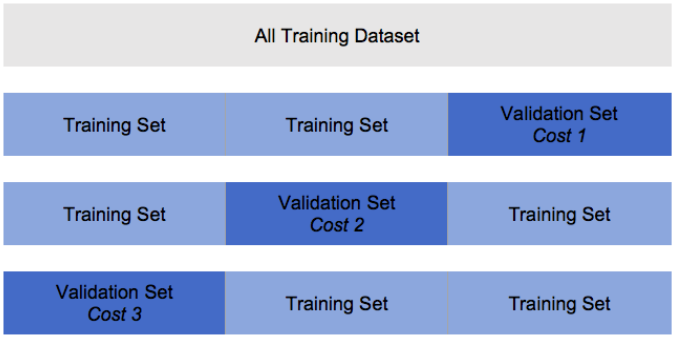

Fig. 3. Cross-Validation with K=3

all the iterations through the window width range, the scores of the best parameters for each of the windows are compared, and the one with the lowest MSE is the chosen one. The result is then the optimal window width (number of features) and the parameter's values.

In Support Vector Regression, the three hyper-parameters are described in table II, as well as the test range of each of the parameters.

TABLE II
SVR PARAMETERS

Parameters	Range
C	$[1, 10, 100, 1000]$
Kernel	Linear, polynomial, RBF
Gamma	$[0.1, 0.01, 0.001, 0.0001]$

The C parameter is called soft margin. A small value of C allows ignoring points close to the boundary, increasing the margin while a large value of C assigns a large penalty to errors and margin error, so the margin is smaller in those cases. The decision boundary is also affected by the kernel that is commonly either linear, polynomial or Gaussian, also known as Radial Basis Function (RBF). The degree of the polynomial kernel and the width parameter of the Gaussian kernel, gamma, influence the flexibility of the decision boundary. In K-Nearest Neighbors, there is only one hyper-parameter to tune and is described in table III, as well as its test range. The value of "K" is the number of nearest neighbors used to predict the value in question.

TABLE III
KNN PARAMETERS

Parameters	Range
K	$[1, 50]$

Step 3 is to fit the model. In this step, the model is fitted using the best window width and hyper-parameters calculated. For example, for support vector machine, if the optimal number of features discovered is 10, and the optimal hyper-parameters discovered are [C=1, gamma = 0.1, kernel='rbf'], SVR will use sequences of the last 10 prices to characterize the next instant, and tomorrow's price will be predicted using a soft-margin of 1 and a Gaussian kernel with an inverse-width parameter of 0.1. For example for k-nearest neighbors, if the optimal number of features discovered is 20, and the optimal hyper-parameter discovered is [K=10], KNN will use sequences of 20 prices to characterize the next instant, and it will use the 10 closer neighbors to predict tomorrow's price.

C. Forecast and validation Layer

The last step for each one of the studied models is the validation of the predictions using the testing dataset. There are four evaluation metrics calculated in this layer: mean absolute error (MAE), return on investment (ROI), Sharpe Ratio (SR) and accuracy.

The mean absolute error takes as an input the actual values and the predicted values, calculates the absolute deviation between each pair of predicted value and actual value, and computes the average of these deviations. To accomplish this calculation, the predicted value of each trade is appended to a list called predictions to serve as an input to the *mean_absolute_error* function. To calculate the Return On Investment, a strategy must be implemented to enter and exit the market. Since the focus of this work is forecasting prices and not optimizing strategies, a very simple strategy is implemented, just to evaluate how well a strategy would work depending on the prediction made. Depending on the prediction for the day after, one of two strategies is created in each trade: 1) if the predicted price is greater than todays price, one should buy shares (go long), and a variable named strategy takes the value 1; 2) if the predicted price is smaller than the todays price, one should short the stock, and a variable named strategy takes the value -1.

After the strategy definition, the profit for that same trade is calculated using todays price as cost of the investment, tomorrows actual price as gain of the investment, and multiplying it by the strategy variable 1 or -1.

The Sharpe Ratio is calculated at the end of the test period using the accumulated profits calculated along the trading period, dividing its mean by its standard deviation. So for that, a list called profits containing all the profits per trade is used as an input to the Sharpe Ratio function. Finally, the accuracy counts the number of times the strategy was in agreement with reality, in other others, the number of times that the strategy chose long or short and the price actually raised or dropped respectively. A right guess generates a positive profit, so the accuracy is the number of positive profits divided by the number of trades.

When validating the models, different stocks are considered as well as different trading periods in order to evaluate the model's performance in different situations.

IV. RESULTS

The performance of the three algorithms in a clear trend stock is discriminated in table IV and a deep comparison is conducted below.

TABLE IV
RESULTS FOR A CLEAR TREND STOCK

		Daily	Weekly	Monthly
MAE	ARIMA	0.695	1.256	2.025
	KNN	10.838	12.073	16.080
	SVR	4.552	2.463	13.004
ROI	ARIMA	34.5%	36.7 %	40.7 %
	KNN	-8.3%	-16.6%	-11.9%
	SVR	23.4%	8.7%	-24.3%
SR	ARIMA	1.906	2.752	3.556
	KNN	-0.591	-1.568	-2.754
	SVR	1.368	0.725	-2.166
ACC	ARIMA	57.8%	67.3%	90.9%
	KNN	44.6%	40%	18.2%
	SVR	57.8%	53.1%	27.3%

It is clear that exists one that stands out: the ARIMA model. ARIMA has the lowest value for MAE in all daily, weekly and monthly forecast, and this difference is more pronounced as the forecast range increases. The higher value of MAE is for KNN when trying to forecast in a monthly range, being 16.400. In general, the KNN errors for this stock are very high. The SVR also has high values of MAE, although they are not so high as the KNN ones. It is observable that for all three cases the error increases with the range of the forecast being higher for the weekly forecast and consequently for the monthly forecast. To better understand the dimension of the errors obtained by the three algorithms, the daily predictions made by each of them are compared against the real actual values of the training data sets. Figure 4 illustrates this comparison.

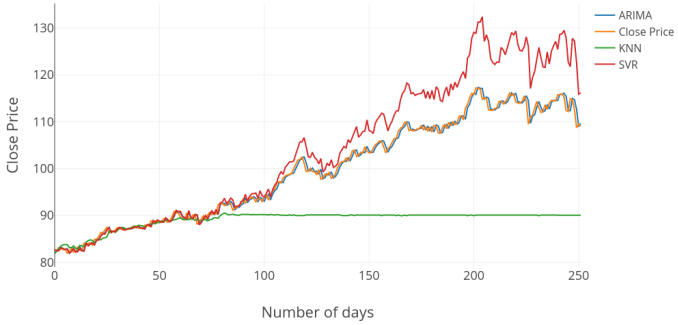

Fig. 4. Comparison of the three algorithms in a clear trend stock

Figure 4 illustrates the behavior of the best daily forecast set obtained for each of the three algorithms during the training period and referenced previously.

The model that is more close the actual price is the ARIMA model, as it was expected since it is the model with the lowest daily MAE for the clear trend, 0.695. Relatively to KNN, at the beginning of the test period, the algorithm seems to fit the model but quickly starts to output bad results. The same happens with SVR. The SVR is able to more or less predict the volatility of the prices but not the actual values, being a little above the actual values, with a MAE of 4.552.

Considering the returns, the comparison term for its evaluation is the B&H strategy which gives a return of 33.2% for this specific stock. This value is only exceeded by all ranges of ARIMA forecast, and KNN and SVR give lower values in all the forecast ranges. Even though these two algorithms do not give such good results as the ARIMA model, the SVR is superior to the KNN in the returns.

The sharpe ratio reflects a good strategy in all ARIMA predictions and also in the daily forecast for SVR, proving again that SVR performs better than the KNN.

The accuracies are very high in the ARIMA case, and not so high in the two machine learning algorithms. Again, even though the SVR does not show good accuracies, it performed better than the KNN model that has accuracies all less than 50%.

Concluding, the ARIMA model performs very well in a clear trend stock, outperforming the B&H strategy and also the two machine learning algorithms. The Support Vector Regression outperformed the KNN taking into account all the evaluation metrics. The KNN is very weak in its predictions and seems to fit the data only in the beginning of the test period.

Table V presents the performance results for the three algorithms in a sideways stock, the BEN stock, that does not show a clear up or down trend. Contrary to the clear trend stock, there is no model or algorithm that stands out in this situation. The errors are similar between the three models and on average they increase with the range of the forecast. The KNN is still the solution that shows higher values of MAE and the lowest error is found again in the daily forecast of the ARIMA model. Figure 5 shows how far the predictions are from the real values, in other words, they illustrate the values of MAE.

TABLE V
RESULTS FOR A SIDEWAYS STOCK

		Daily	Weekly	Monthly
MAE	ARIMA	0.362	0.776	2.230
	KNN	0.810	3.378	5.574
	SVR	0.807	1.422	3.622
ROI	ARIMA	-29.5%	-4.7%	-15.2%
	KNN	-28.4%	-1.5%	-15.2%
	SVR	1.9%	-4.4%	-1.5%
SR	ARIMA	-2.034	-0.318	-0.908
	KNN	-1.704	-0.082	-0.908
	SVR	0.101	-0.227	-0.078
ACC	ARIMA	44.6%	44.9%	36.4%
	KNN	44.6%	40%	36.4%
	SVR	51.8%	50%	63.6%

The ARIMA predictions are very close to the actual values, and the KNN performs a lot better compared to the clear trend stock. The SVR does not show very precise values and it is not obvious just by looking at figure 5 which of the KNN or SVR

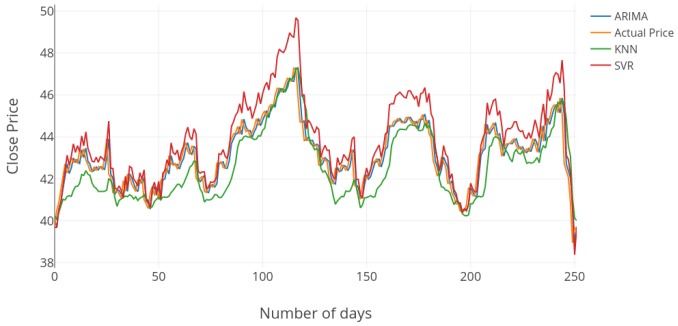

Fig. 5. Comparison of the three algorithms in a sideways stock

is more accurate. Only looking at MAE values is possible to check that SVR outputs values more close to the actual prices.

Concerning the returns, the results may not seem very optimistic. In fact, this specific stock is very volatile, with a very inconstant behavior, being more difficult to invest and have good results. The B&H strategy gives a profit of -0.74% being only exceeded by the daily SVR forecast. In this volatile stocks, even when the errors are small, the returns can be very low since it is difficult to predict if the price will increase or decrease. ARIMA has the lowest value of returns for the daily forecast, -29.5%.

The sharpe ratio is lower than 1 for all situations meaning that even the lowest return values represent a very risky investment. The accuracies are high for the SVR algorithm, and they are very low for the KNN method. ARIMA exhibits average results of accuracy, but all inferior to 50%.

A. *Overall comparison*

Forecasting in the stock market is not an easy task, and forecast results can sometimes be dubious. The great part of the works related to forecasting only use either error metrics or return/profit/accuracy metrics. When dealing with financial data it is extremely important to look at both metric types and learn to find the appropriate trade-off between the two. For example, the daily results for the ARIMA forecast applied to the sideways stock have a very low error value of 0.362, and if this value was evaluated alone without the remaining context it would seem a very good and precise result. The returns for the same daily forecast are very low, being negative and almost -30%. This is just an example that results should be analyzed carefully and inside the context.

To solve the problem of forecasting in the stock market, the ARIMA gives results very close to the actual values and their results are consistent along the test period. This model performs a lot better when applied to a clear trend stock, and it does not fit well when consistent up and down movements are happening. When dealing with the clear trend stock outperforms the two machine learning algorithms. ARIMA can also give very high returns when used to construct a strategy, giving returns way above the current values. ARIMA is also a very simple statistic algorithm and it is not very complex, compensating with its low computational effort. Besides this, for this work purposes, only values inside a range between 0 and 5 were tested since the computational resources were limited.

The two machine learning algorithms performed not as expected, being overpast by the ARIMA model in the clear trend stock. Support Vector Machines performed better than the KNN model, mainly because the KNN is a very simple and lazy-algorithm with a slow learning, and SVR is a more complex one that with the more complex kernels can perceive complex relations between data. Also, KNN is more used in classification tasks, where feature scaling is very common and helps a lot the performance of the algorithms.

Another clear conclusion is that looking at the MAE values, as the range of the forecast raises, also it raises the error, and this happens for almost all situations. Also, the results prove that good values of errors do not mean good returns, and good returns do not mean a good investment since the risk must be taking into account. When dealing with KNN, there are no K values superior to 29, indicating that there is no need to try very high values of K. Considering SVR, the polynomial kernel is the one with better results, even though the RBF is mostly used with complex data such as financial data.

Both the machine learning algorithms did not live up to the expectations. Looking at figures 4 and 5, it seems that both KNN and SVR can fit the data at the beginning much better than they did in the end. This can be caused by the fact that the hyper-parameters start to be out of data and the model needs to be retrained. Taking this into consideration, the retraining of the machine learning algorithms was conducted in order to compare the results and see if it gives more precise values.

B. *Retraining KNN and SVR*

The introduction of a retraining period is due to the fact that, concerning the mean absolute errors, the two machine learning algorithms present good predictions at the beginning of the test period and deteriorate their performance over time. This observation leads to the suspicion that the algorithms may lose their ability since they were trained before the test period and their hyper-parameters can be out of date. It is important to give the opportunity to the models to learn the price trend from the past, but it also relevant that the algorithms use up to date information, not being only influenced by data located too far in the past.

Taking this into consideration, a retraining step is introduced for both KNN and SVR in order to check if their results in terms of mean absolute error improve. This retraining step is incorporated during the test period and the retraining period varies between 5, 10 and 15. A retraining period of 5 means that for each 5 executed trades, the algorithm is retrained and new hyper-parameters are calculated and used until the next retrain.

This approach is used for the KNN and SVR performance in both stocks for a daily forecast and compared with the results obtained in sections 4.2 and 4.3.

The new daily results of the KNN with a 5-period retraining for the clear trend stock are illustrated in figure 6. KNN had the

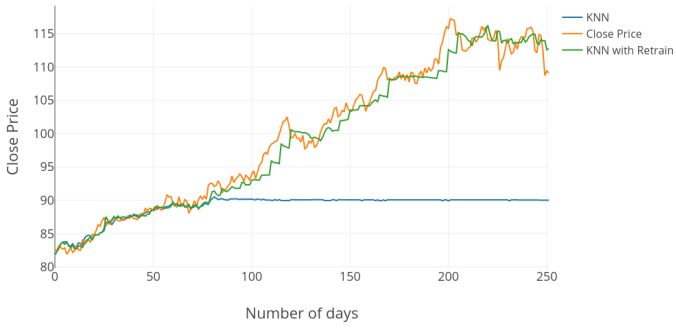

Fig. 6. KNN with Retraining for a Clear Trend Stock

most inadequate results in relation to the actual values for the clear trend forecast, and the retraining every 5 trades improved a lot its error and consequently the returns, shaper ratio and accuracy of the strategy based on KNN predictions. These results were obtained for the same test period and number of features that were used in section 4.2.

Even though the KNN had a better performance in the sideways stock, its results are also improved by the introduction of a retraining period, reducing the error and improving the returns, sharpe ratio and accuracy of the algorithm. The optimal retraining period was again every 5 trades. The improvements are illustrated in figure 7 for the daily results.

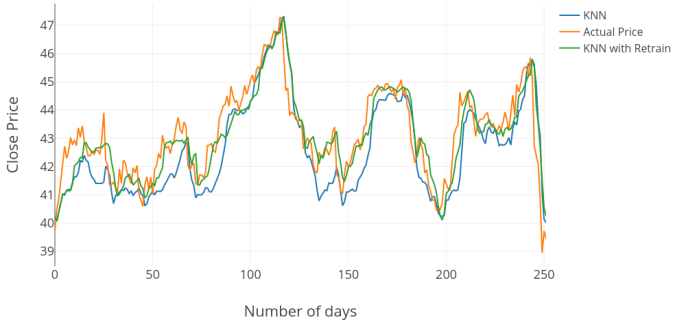

Fig. 7. KNN with Retraining for a Sideways Stock

Also, the SVR was way below the expectations and a retraining was introduced during the test period. The retrain period varies between 5, 10 and 15. For the clear trend, the SVR predicted values that were higher than the actual prices, even though the shape of the output time series was similar. With the introduction of the retraining every 5 trades, the SVR error improved a lot but the same did not happen with the returns, sharpe ratio and accuracy. The results are illustrated in figure 8.

Concerning the sideways stock, the results of the SVR errors are better than without taking the retraining. Again the optimal

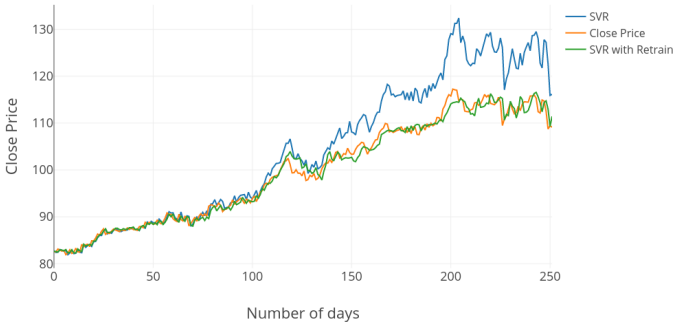

Fig. 8. SVR with Retraining for a Clear Trend Stock

period number is every 5 trades. Even though the error shows an improvement, the rest of the results are not so optimistic. The results are illustrated in figure 9.

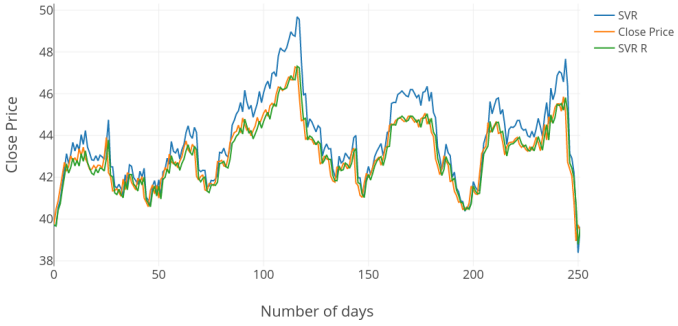

Fig. 9. SVR with Retraining for a Sideways Stock

Concluding, the retraining of the machine learning algorithms is something to take in consideration when forecasting for a long test period and show improvements in every case. The chosen period for all the cases was every 5 trades. This may not work in all situations since it increases the computational effort. This period should be optimized for each algorithm in its specific context.

V. Conclusion

The best performance for the clear trend stock corresponds to the ARIMA model that exceeds the B&H strategy, with returns of 40.7%. For the sideways stock, the SVR was the one with highest returns, even though they were not very good. The two machine learning algorithms demonstrated a good fit in the beginning of the test period and started to degrade their performance over time. The introduction of a retraining period was tested in order to find if the results could be improved. Due to this introduction, both algorithm results were better than the ones obtained before, proving the point that this model retraining should be considered when forecasting for a long test period. There are some points that are worth to

be enumerated in order to have a brief understanding of the reached conclusions:

1) The choice of the evaluation metrics are extremely important, and a reliable comparison between algorithms should not be conducted based only on one metric;
2) The ARIMA model has very good results in a clear trend stock, while in a sideways stock the same does not happen;
3) K-Nearest Neighbors is a very simple algorithm that does not fit the stock data very well due to the complexity of price moves and simplicity of the algorithm, being the weakest implemented algorithm;
4) Support Vector Regression performs better than the K-Nearest Neighbors in modeling and forecasting financial data but it exceeded by the ARIMA model in a clear trend stock;
5) Machine learning algorithms can lose their validity and introduce a retraining along the test period improves a lot the error results.

Concluding, forecast is a very difficult task, and even more in the financial field where the prices can be so unpredictable. The flow of a stock price can have multiple channels of influence, and in this work only past price sequences were used, probably impairing the results. Machine learning algorithms are used in the financial field more often as classifiers than as regressors, which turned the task of forecast a continuous value (close price) more challenging. In the end, the three algorithms show very interesting results even though only price sequences were used as their input, making the point that the idea of forecast future prices as continuous variables can be a very promising tool for investors and traders.

VI. Future Works

For future works, the present thesis should be seen as a starting point in the forecast of stock prices as continuous variables. To continue this work, the following approaches can be conducted: 1) evaluate the models with more refined and complex strategies since the implemented one was very simplistic and only to gain an idea of how useful the predictions were; 2) optimize the retraining periods since only 3 periods were tested; 3) integrate the solution into a Big Data platform in order to process more data in a more quickly way; 4) the use of different algorithms to forecast future prices that can work as regressors; 5) combining price sequences with fundamental analysis in order to enter with more channels of influence.

References

[1] Bernard Marr. "A Short History of Machine Learning". *Forbes*, pages 1–2, 2016.

[2] Mohammad Mahdi Rounaghi and Farzaneh Nassir Zadeh. "Investigation of market efficiency and Financial Stability between S&P 500 and London Stock Exchange: Monthly and yearly Forecasting of Time Series Stock Returns using ARMA model". *Physica A: Statistical Mechanics and its Applications*, 456:10–21, 2016.

[3] Tomas Vantuch and Ivan Zelinka. "ECC 14 - Evolutionary Based ARIMA Models for Stock Price Forecasting". 2014.

[4] Eduardo A. Gerlein, Martin McGinnity, Ammar Belatreche, and Sonya Coleman. "Evaluating machine learning classification for financial trading: An empirical approach". *Expert Systems with Applications*, 54:193–207, 2016.

[5] J. Kamruzzamana and R. A. Sarkerb. "Comparing ANN Based Models with ARIMA for Prediction of Forex Rates". *ASOR BULLETIN*, 22(2):2–11, 2003.

[6] Jacek Mandziuk and Piotr Rajkiewicz. "Neuro-evolutionary system for FOREX trading". *2016 IEEE Congress on Evolutionary Computation, CEC 2016*, pages 4654–4661, 2016.

[7] P.D. Yoo, M.H. Kim, and Tony Jan. "Machine Learning Techniques and Use of Event Information for Stock Market Prediction: A Survey and Evaluation". *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*, 2:835–841, 2007.

[8] Kyoung Jae Kim. "Financial time series forecasting using support vector machines". *Neurocomputing*, 55(1-2):307–319, 2003.

[9] Lijuan Cao and Francis E. H. Tay. "Application of support vector machines in financial time series forecasting". *Omega*, 29(4):309–317, 2001.

[10] Wun Hua Chen, Jen Ying Shih, and Soushan Wu. "Comparison of support-vector machines and back propagation neural networks in forecasting the six major Asian stock markets". *International Journal of Electronic Finance*, 1(1):49, 2006.

[11] Yingjun Chen and Yongtao Hao. "A feature weighted support vector machine and K-nearest neighbor algorithm for stock market indices prediction". *Expert Systems with Applications*, 80:340–355, 2017.