Financial time series forecasting methods

Anna Zinenko*, and *Alena* Stupina

Siberian Federal University, 79, Svobodny Prospect, Krasnoyarsk, 660049, Russian Federation

Abstract. The paper presents the development of time series forecasting algorithms based on the Integrated Autoregressive Moving Average Model (ARIMA) and the Fourier Expansion model. These models were applied to non-stationary time series of stock quotes after bringing these series to a stationary form. In the paper, ARIMA and Fourier Expansion model were constructed, using Python development environment. The developed algorithms were tested on Russian and American stock indices using the Mean Absolute Percentage Error metric.

1 Introduction

Traditional methods of forecasting univariate time series include autoregressive models and smoothing models [1]. Autoregressive models represent the current values of a time series as a regression dependence on past values, and smoothing models consider some moving indicator (for example, a simple moving average) and continue the trend based on it. Models of this kind work well when forecasting time series have an obvious trend. Also, classical forecasting models can take into account seasonality [2].

The following paper describes algorithms for more advanced forecasting methods - the Integrated Autoregressive Moving Average Model (ARIMA) and the Fourier Expansion model. The ARIMA model is a modification of the autoregressive model that takes into account the non-stationarity of the time series and adds random moving average components. The Fourier decomposition decomposes the original time series into the sum of harmonic oscillations and makes a forecast based on the most significant harmonics.

Time series with an obvious trend and with seasonality are non-stationary time series. Time series,stationary in the broad sense, include series with a constant mean and variance [3]. In practice, such series describe random processes, which are even easier to predict; it is enough to calculate the average value and variance, then all actual values will be around the average with a spread of standard deviation.

Financial quotes time series are obviously non-stationary, like most time series observed in real life. Seasonality in financial time series can be detected only throughout the entire history of any instrument and trendiness in this kind of time series is obviously presents.

In this work, we developed algorithms for forecasting methods that assume stationarity of the original series. The time series was brought to a stationary form by taking differences. By taking differences, forecasting was carried out not at the original levels y_k , but on the differences between adjacent levels $y_k - y_{k-1}$. If taking differences does not give a

* Corresponding author: anna-z@mail.ru

stationary time series, then we should take second-order differences $\Delta_k - \Delta_{k-1}$, where Δ_1 are first-order differences. As a rule, taking second-order differences already removes stationarity. In the case of financial time series, first-order differences are sufficient to achieve stationarity. The Dickey-Fuller test allows to determine the stationarity of a time series. This test calculates autocorrelation coefficient between adjacent levels of the series. For stationarity, the value of this coefficient must be less than one. The Dickey–Fuller test then checks the correlation coefficient between the first-order difference and the adjacent level. To achieve stationarity, this coefficient must not be equal to zero. The null hypothesis of non-stationarity of the time series is rejected when the p-value of the Dickey Fuller test becomes less than or equal to 0.05 [4].

2 Methods

The ARIMA (*p*,*d*,*q*) model is an autoregressive moving average model (ARMA(*p*,*q*)), which is applied not to the values of series levels, but to differences of order d (for financial time series, d=1 is sufficient). The ARMA (p, q) model is shown by the following equation

$$y_t = \sum_{i=1}^{p} \alpha_i y_{t-i} + \varepsilon_t + \sum_{i=1}^{p} \beta_i \varepsilon_{t-i} , \quad (1)$$

where y_t – current level, y_{t-i} – level, laggard by lag i, ε_t – moving average forecast error, ε_{t-i} – moving average forecast error laggard by lag i. The first sum of the model is the autoregressive component, the second is the moving average component. For the ARIMA model, formula (1) is transformed as follows

$$\Delta_t^d = \sum_{i=1}^{p} \alpha_i \Delta_{t-i}^d + \varepsilon_t + \sum_{i=1}^{p} \beta_i \varepsilon_{t-i} , \quad (2)$$

where Δ_t^d is the difference of d-order required to achieve stationarity [5].

The series levels partial autocorrelation plot determines the order of autoregression, and the autocorrelation plot determines the order of the moving average. The X-axis shows time lags, and the Y-axis shows autocorrelation or partial autocorrelation between the current level and the level lagging behind the lag [6].

Figure 1 shows plots of autocorrelation and partial autocorrelation. The order of the autoregression (moving average) increases to such a lag size as long as the correlation coefficients go beyond the shaded area.

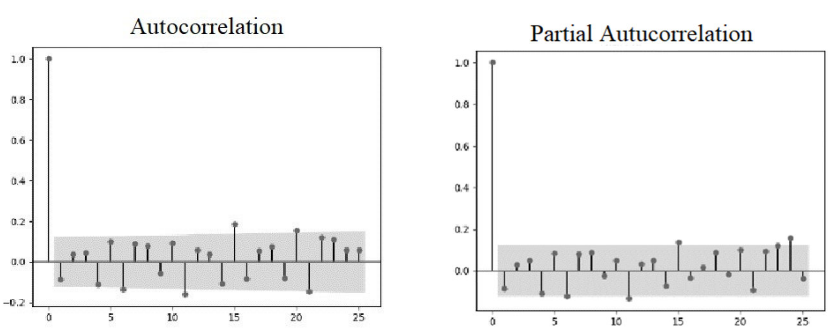

Fig. 1. Autocorrelation and partial autocorrelation plots.

Parameters p and q can take values from 0 to 2, and their sum should not exceed 3. Software packages such as Python, R Studio, Mathlab and others have built-in functions for an ARIMA model construction and forecasting for a given period [6]. Python shows the best results in statistical modeling. For construction an ARIMA model in Python it is nessesary onlyto define the model parameters p, d, q. If the order of taking the differences is different from zero, then the model makes the forecast based on the differences, so the forecast should be carried out on an accrual basis.

However, in practice, for stock time series, the ARIMA function built into Python Statsmodels produces a straight line as a forecast when the moving average is not equal to zero, despite the fact that the presence of a moving average model implies a noise component. In frequent cases, the forecast is given as a line parallel to the X axis, which makes it inadequate.

While constructing the ARMA model and the moving average model, the problem of determining the noise components ε_i appears. In references, these components are defined as Gaussian white noise with zero mathematical expectation and standard deviation σ. The question arises of how to determine the dispersion of noise components. The proposed algorithm offers the following solution.

After taking first-order differences, we can determine the autoregression parameters through the Yule-Walker equations.

$$\begin{gathered} r_1 = \alpha_1 + \alpha_2 r_2 + \cdots + \alpha_p r_{p-1} \\ r_2 = \alpha_1 r_1 + \alpha_2 + + \cdots + \alpha_{p-1} r_{p-2} \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ r_p = \alpha_1 r_{p-1} + \alpha_2 r_{p-2} + \cdots + \alpha_p, \end{gathered} \tag{3}$$

where r_i – autocorrelation coefficients, α_i – autoregression model parameters [7]. Thus, the model parameters can be written through autocorrelation coefficients: for $p = 1$:

$$\alpha_1 = r_1,$$

for $p = 2$:

$$\begin{gathered} \alpha_1 = \frac{r_1(1-r_2)}{1-r_1^2}, \\ \alpha_2 = \frac{r_2-r_1^2}{1-r_1^2}. \end{gathered} \tag{4}$$

Autocorrelation coefficients can be determined using Python's Statsmodels library. Putting the parameters into the autoregression formula

$$\begin{gathered} y_t = \alpha_1 y_{t-1}, \text{ or} \\ y_t = \alpha_1 y_{t-1} + \alpha_2 y_{t-2}, \end{gathered} \tag{5}$$

we got calculated values of series levels. The standard deviation of white noise ε_t can be specified as the standard deviation of the differences between the actual series levels and the series levels calculated by formula (5). We should also set the noise ε_{t-1} and ε_{t-2} standard deviations (for the first- and second-order moving average components) as the standard deviations of the differences without the last one and last two actual and estimated levels of the series.

Having the actual values of the time series levels, as well as white noise equal to $N(0, \sigma)$, where σ is the calculated standard deviation of the actual values from predicted by the autoregression model, it is possible to determine the parameters α_i, β_i of the ARMA models using the method of least squares which minimizes the sum of squared differences. To perform the minimization task in the Python Scipy library, adding restrictions is required, because the model parameters should not exceed 1 in absolute value.

The algorithm assumes three ARIMA (p, d, q) models: ARIMA (1,1,1), ARIMA (1,1,2), ARIMA (2,1,1). More often, the ARIMA (1, 1, 1) model is used. After defining α_i, β_i parameters the algorithm performs recurrent forecasting using the formulas

$$\begin{gathered} \Delta_{t+1}^1 = \alpha_1 \Delta_t^1 + N(0, \sigma_t) - \beta_1 N(0, \sigma_{t-1}) \\ \Delta_{t+2}^1 = \alpha_1 \Delta_{t+1}^1 + N(0, \sigma_t) - \beta_1 N(0, \sigma_{t-1}) \\ \Delta_{t+l}^1 = \alpha_1 \Delta_{t+l-1}^1 + N(0, \sigma_t) - \beta_1 N(0, \sigma_{t-1}), \end{gathered} \tag{6}$$

where $N(0, \sigma_t)$ is a normal distribution with zero expected value and standard deviation of the differences between the actual values and the values obtained from the autoregressive model, l is the forecast period. Formulas (6) illustrate the forecast for most used ARIMA (1, 1, 1) model. Formulas (6) predict first-order differences, and not the values of series levels, therefore, for a correct forecast, the accumulated sum method should be used.

$$y_i = y_{i-1} + \Delta_i^1, \quad (7)$$

where y_0 is the last actual series level.

The Fourier transform matches a function with its frequency spectrum, in other words, shows the dependence of amplitude and phase on frequency [8]. The Fourier transform is described by integral formulas and complex variables, but to develop an algorithm for forecasting time series (which can also be a periodic function), its discrete interpretation is quite enough.

The Fourier series expansion of a function divides a periodic function into a sum of sine and cosine waves with known frequencies, amplitudes, and phases. Thus, if we decompose a time series into the sum of known functions, it will not be difficult to make a forecast. It is important that the decomposition is applied to first-order differences, since with a short-term forecast it is incorrect to highlight periodicity in financial time series and in the same time, first order differences are periodic. The Fourier expansion is shown by formula (8).

$$y_t = \bar{y_t} + \sum_{i=1}^{N/2}(a_i\ cos\omega_i t + b_i\ sin\omega_i t), \quad (8)$$

where y_t is predicted time series value, $\bar{y_t}$ is average time series value, ω_i is angular frequency of the i-th harmonic (the first frequency corresponds to the period of the function, the rest are multiples of it), N is number of series levels (period), a_i, b_i are model parameters, which we need to find.

The Fourier expansion coefficients can be estimated by following formulas

$$a_1 = \frac{2}{N}\sum_{t=0}^{2\pi(N-1)/N} y_t \cos t,\ \ b_1 = \frac{2}{N}\sum_{t=0}^{2\pi(N-1)/N} y_t \sin t \quad (9)$$

for first harmonic,

$$a_2 = \frac{2}{N}\sum_{t=0}^{2\pi(N-1)/N} y_t \cos 2\, t,\ \ b_2 = \frac{2}{N}\sum_{t=0}^{2\pi(N-1)/N} y_t \sin 2\, t \quad (10)$$

for second harmonic etc.

In practice, not all terms are used, but the most significant harmonics [9]. So, those harmonics whose coefficients a and b have a small absolute value can be discarded. We propose to use a discrete Fourier transform to determine the number of harmonics in the forecast. Fourier transform is "spectral portrait" of the function which shows the amplitudes of oscillations depending on frequencies. The discrete Fourier transform is represented by the following formula

$$X_k = \sum_{n=1}^{N-1} y_n\left(cos\frac{2\pi kn}{N} - i\ sin\frac{2\pi kn}{N}\right) \quad (11)$$

$$X_k = A_k * N,$$

where X_k is complex amplitude of k-th harmonic, A_k is amplitude of k-th harmonic, y_n is n-th series level, i is imaginary unit [10].

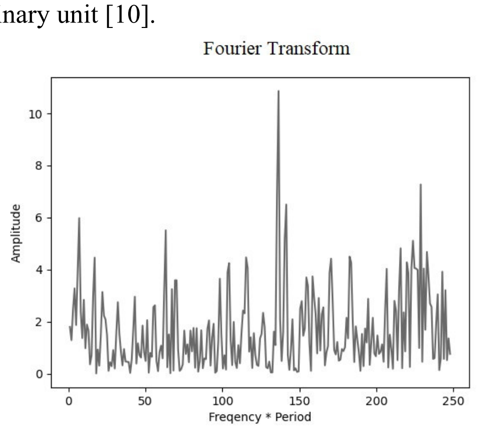

Fig. 2. Discrete Fourier transform plot.

X_k is is a complex number whose modulus is the amplitude multiplied by the period, and whose argument is the oscillation phase. To determine the largest amplitude, the real part is sufficient. Figure 2 shows the frequency portrait of the first-order differences in the time series. Note that the amplitudes are symmetrical, so we should look for significant harmonics at the frequencies of the interval $\frac{N}{2}$.

Thus, after taking first-order differences, the algorithm finds the real parts of the amplitudes and returns a list of frequencies with the highest amplitudes. Emission values were calculated with the formula

$$(A_i - \bar{A})/\sigma > 3, \quad (12)$$

where A_i is the i-th amplitude, $\bar{A}$ is the average value of amplitudes, σ is the standard deviation [11].

The next step of the algorithm is to find the coefficients aa_i, b_i from formulas (9) and (10) using the frequencies determined in the previous step. Before substituting into formulas (9), (10) frequencies should be divided by period N. Then the resulting coefficients were substituted into formula (8) and the predicted values of the differences were found. The last step is the same as in forecasting using the ARIMA method - by adding a number of forecast differences to the last actual level on a cumulative basis.

3 Results and discussion

For testing the algorithms, it is necessary to determine on what data the test is performed, divide the original time series into a training and test set, and determine the criterion for the adequacy of the forecast. We took daily stock quotes of the Moscow Exchange index (43 items), the New York Stock Exchange index (533 items), NASDAQ (231 items) and the American Stock Exchange (259 i items) for a one year time period. The division into training and testing sets was carried out in a ratio of 80:20. Model coefficients were found from the training sample, and the test sample was compared with the predicted results. As an error metric we chose the mean absolute percentage error (MAPE). This metric is convenient because it allows to evaluate the deviation of forecast data from actual data as a percentage. The MARE metric is calculated using formula (13).

$$MAPE = \frac{1}{n}\sum_{i=1}^{n} |\frac{y_i - \hat{y}_i}{y_i} * 100|, \quad (13)$$

where n is the number of observations in the test set, y_i ¬ is the actual value of the parameter in the test set, $\hat{y}_i$is the predictive value of the parameter [12].

The received results are given in Table 1.

Table 1. MAPE forecasts based on ARIMA models and Fourier decomposition (period 1 year).

Index	MAPE, ARIMA	MAPE, Fourier decomposition
MOEX (44 items)	7.4%	7.1%
NYSE (533 items)	8.6%	7.3%
NASDAQ (231 items)	8.9%	7.9%
AMEX (259 items)	12.9%	11.3%

From the results, we can conclude that both forecasting algorithms perform approximately the same, with the Fourier expansion showing a slightly smaller error. When we compared the developed ARIMA algorithm with the built-in Python Statsmodels, our algorithm showed the best results. Also, the built-in algorithm failed during automatic testing on a large number of tools. Figure 3 shows the visualization of the forecast using ARIMA and Fourier Decomposition methods.

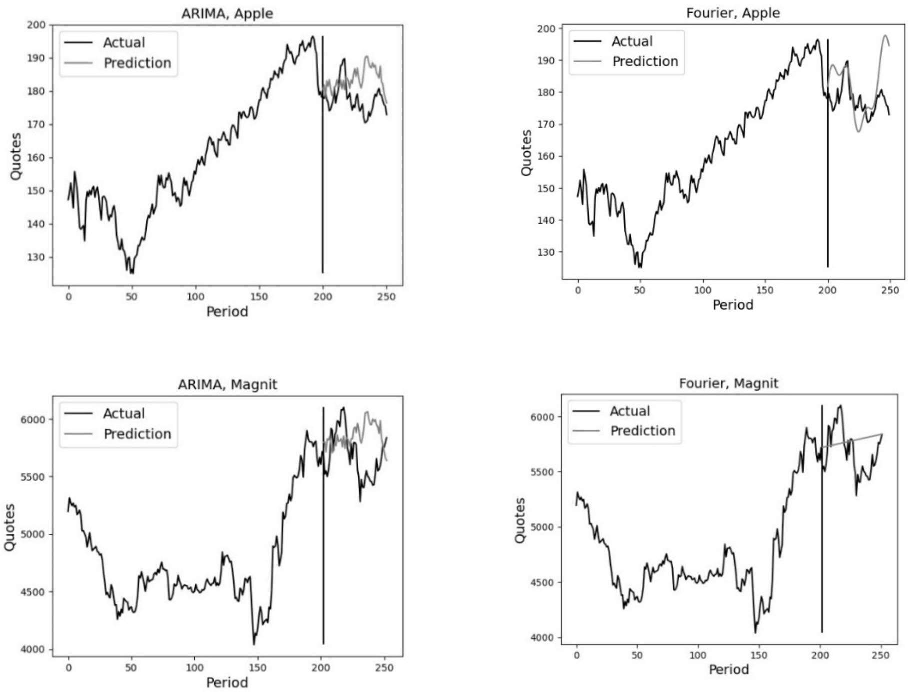

Fig. 3. Forecasts of some stocks visualization.

4 Conclusion

The presented algorithms were developed taking into account the features of financial time series. Unlike algorithms built into software packages, we can adjust them if verification fails on large data sets. Also, the developed forecasting algorithms show good results in predicting the prices of Russian and foreign stock quotes. In the future, a similar check on large arrays is planned to be carried out on fiat and cryptocurrency quotes. Another advanced forecasting algorithm – Singular Spectral Analysis – is also in development.

This work was supported by the Ministry of Science and Higher Education of the Russian Federation (Grant No.075-15-2022-1121).

References

1. E. Nielsen, *Practical analysis of time series: forecasting with statistics and machine learning* (Dialectics LLC., 2021)
2. V.N. Aphanasyev, *Time Series Analyses and Forecasting* (AiPiEr Media, 2020)
3. K.O. Kizbikenov, *Forecasting and time series* (Federal State Budgetary Educational Institution of Higher Education "AltSPU", 2017)
4. V.V. Vityazev, *Uneven Time Series Analysis* (SPbFU, 2001)
5. J. Zhang, H. Liu, W. Bai, X. Li, North Am. J. Econ. Finance **69**, 102022 (2024)
6. Jun Luo, Yaping Gong, Atmos. Pollut. Res. **14** iss. 16, 101761 (2023)

7. G.G. Kantorovich, HSE Ec. J. **1**, 85 (2002)
8. G. Box, G. Jenkins, *Time series analysis, forecast and management* (Mir, 1974)
9. V.P. Kandidov, S.S. Chesnokov, S.A. Shlenov, *Discrete Fourier Transform* (Faculty of Physics, Moscow State University, 2019)
10. A.D. Chernishev, V.V. Gorainv, Th. & App. Sc. **1(33)**, 137 (2016)
11. G. Khodamipour, S. Khorashadizadeh, M. Farshad, ISA Trans. **138**, 63 (2023)
12. F. Cholet, *Deep Learning with Python* (Peter, 2018)