Comparison of various models for stock prediction

Jonathan Lee

Contest: Financial data utilization

Abstract. The stock forecast of the Korean stock with using the FinanceDataReader provided by Python Module. Eight models were compared to choose the best model. As a result, ARIMA had the best performance.

1 Introduction

Due to the high volatility of the COVID-19 pandemic, interest in stock investment is focused. Also, it is said that the atmosphere is gathering again from the cryptocurrency market to the domestic stock market. In this situation, we looked at which model could more accurately predict the closing price of a stock.

2 Models

We compared eight models for forecasting stock prices. The models: Linear Regression, XGBoost, ARIMA, ES, VAR, LSTM, DeepAR, AR-net

2.1 Linear Regression

In statistics, linear regression is a linear approach for modelling the relationship between a scalar response and one or more explanatory variables (also known as dependent and independent variables). The case of one explanatory variable is called simple linear regression; for more than one, the process is called multiple linear regression. This term is distinct from multivariate linear regression, where multiple correlated dependent variables are predicted, rather than a single scalar variable.

In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models. Most commonly, the conditional mean of the response given the values of the explanatory variables (or predictors) is assumed to be an affine function of those values; less commonly, the conditional median or some other quantile is used. Like all forms of regression analysis, linear regression focuses on the conditional probability distribution of the response given the values of the predictors, rather than on the joint probability distribution of all of these variables, which is the domain of multivariate analysis.

Linear regression was the first type of regression analysis to be studied rigorously, and to be used extensively in practical applications. This is because models which depend linearly on their unknown parameters are easier to fit than models which are non-linearly related to their parameters and because the statistical properties of the resulting estimators are easier to determine.

Linear regression has many practical uses. Most applications fall into one of the following two broad categories:

If the goal is prediction, forecasting, or error reduction,[clarification needed] linear regression can be used to fit a predictive model to an observed data set of values of the response and explanatory variables. After developing such a model, if additional values of the explanatory variables are collected without an accompanying response value, the fitted model can be used to make a prediction of the response. If the goal is to explain variation in the response variable that can be attributed to variation in the explanatory variables, linear regression analysis can be applied to quantify the strength of the relationship between the response and the explanatory variables, and in particular to determine whether some explanatory variables may have no linear relationship with the response at all, or to identify which subsets of explanatory variables may contain redundant information about the response. Linear regression models are often fitted using the least squares approach, but they may also be fitted in other ways, such as by minimizing the "lack of fit" in some other norm (as with least absolute deviations regression), or by minimizing a penalized version of the least squares cost function as in ridge regression (L2-norm penalty) and lasso (L1-norm penalty). Conversely, the least squares approach can be used to fit models that are not linear models. Thus, although the terms "least squares" and "linear model" are closely linked, they are not synonymous.[2]

2.2 XGBoost

XGBoost is an open-source software library which provides a regularizing gradient boosting framework for C++, Java, Python, R, Julia, Perl, and Scala. It works on Linux, Windows, and macOS. From the project description, it aims to provide a "Scalable, Portable and Distributed Gradient Boosting (GBM, GBRT, GBDT) Library". It runs on a single machine, as well as the distributed processing frameworks Apache Hadoop, Apache Spark, Apache Flink, and Dask. It has gained much popularity and attention recently as the algorithm of choice for many winning teams of machine learning competitions.

XGBoost initially started as a research project by Tianqi Chen as part of the Distributed (Deep) Machine Learning Community (DMLC) group. Initially, it began as a terminal application which could be configured using a libsvm configuration file. It became well known in the ML competition circles after its use in the winning solution of the Higgs Machine Learning Challenge. Soon after, the Python and R packages were built, and XGBoost now has package implementations for Java, Scala, Julia, Perl, and other languages. This brought the library to more developers and contributed to its popularity among the Kaggle community, where it has been used for a large number of competitions.

It was soon integrated with a number of other packages making it easier to use in their respective communities. It has now been integrated with scikit-learn for Python users and with the caret package for R users. It can also be integrated into Data Flow frameworks like Apache Spark, Apache Hadoop, and Apache Flink using the abstracted Rabit and XGBoost4J. XGBoost is also available on OpenCL for FPGAs. An efficient, scalable implementation of XGBoost has been published by Tianqi Chen and Carlos Guestrin.

Salient features of XGBoost which make it different from other gradient boosting algorithms include:

Clever penalization of trees A proportional shrinking of leaf nodes Newton Boosting Extra randomization parameter Implementation on single, distributed systems and out-of-core computation Automatic Feature selection

XGBoost works as Newton Raphson in function space unlike gradient boosting that works as gradient descent in function space, a second order Taylor approximation is used in the loss function to make the connection to Newton Raphson method. [4]

2.3 LSTM

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can process not only single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition and anomaly detection in network traffic or IDSs (intrusion detection systems).

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, hidden Markov models and other sequence learning methods in numerous applications [3]

2.4 Exponential smoothing

Exponential smoothing is a rule of thumb technique for smoothing time series data using the exponential window function. Whereas in the simple moving average the past observations are weighted equally, exponential functions are used to assign exponentially decreasing weights over time. It is an easily learned and easily applied procedure for making some determination based on prior assumptions by the user, such as seasonality. Exponential smoothing is often used for analysis of time-series data.

Exponential smoothing is one of many window functions commonly applied to smooth data in signal processing, acting as low-pass filters to remove high-frequency noise. This method is preceded by Poisson's use of recursive exponential window functions in convolutions from the 19th century, as well as Kolmogorov and Zurbenko's use of recursive moving averages from their studies of turbulence in the 1940s. [1]

3 Conclusion

Except for Linear Regression and XGBoost, all predictions were made using only closing price information. Linear Regression, XGBoost 2 model was predicted using the closing price, opening price, high price, and low price. In the case of LSTM, predictions were made using 16 hidden cells. In the case of DeepAR, the performance was rather low and took a long time, so we excluded it from the measurement. The measured evaluation formula is NMAE * 100. Using data from the previous week, we predicted the stock price from November 1, 2021 to November 5, 2021.

Models	FinanceDataReader Korea Stock Forecasting
Linear Regression	3.46
XGBoost	4.45
ARIMA	2.97
ES	2.95
VAR	2.98
LSTM	X
AR-net	2.96

Table 1. NMAE * 100

The model performance results different weekly. In the case of LSTM, the results were not good. In the case of LSTM, it is estimated that more model parameter modifications and pre-processing are required. The performance of ARIMA and ES is almost identical. However, ARIMA was slightly more stable in measurements in about average 8 weeks. Among the ARIMA models, the (1,1,0) selection was able to obtain the highest average score. AR-net had almost the same performance as ARIMA, but because ARIMA was ahead in terms of time, ARIMA(1,1,0) measured according to Ockham's razor principle was considered to be the best model on average.

References

1. contributors, W.: "exponential smoothing." wikipedia
2. contributors, W.: "linear regression." wikipedia
3. contributors, W.: "long short-term memory." wikipedia
4. contributors, W.: "xgboost." wikipedia