Machine Learning for Time Series Forecasting: A Comparison of New and Traditional Methods on Volatility

Lester Pi
Faculty Advisor: Professor Randall R. Rojas

June 13, 2017

```r
knitr::opts_chunk$set(echo = TRUE)
```

```r
library('xts')
library("quantmod")
library('forecast')
library('dynlm')
library('vars')
library('tseries')
library('glmnet')
library('randomForest')
library('neuralnet')
library('plyr')
library('glarma')
library('caret')
library('png')
library('grid')
```

Abstract

Traditionally econometric models such as autoregressive (AR), moving average (MA), and autoregressive moving average (ARMA) models are used to model, analyze, and forecast time series. However, with the advent of big data and exponential increases in computing power there have been groundbreaking leaps in the field of machine learning. This empirical paper focuses on various time series forecasting by comparing different methods on financial and economic data. The first time series that will be tested is the S&P 500 30-day volatility, a continuous linear time series. The second time series that will be tested is a binary classification time series of a recession indicator. The goal is to either find a model that out performs our traditional econometric methods. The new models that we will be examining are LASSO regressions, decision trees as random forests, and artificial neural networks. We measure performance through the mean absolute percentage error (MAPE), the classification accuracy and confusion matricies, and practical feasibility. After running through rigerous testing methods on our data sets, we propose two new models that feasibly out perform ARIMA. For linear regression forecasting on S&P 500, we propose an artificial neural network model. For recession classification, we purpose a decision tree implemented through a random forests model.

1. Introduction

1.1 Motivation

This paper is motivated by the interdisiplinary studies of applied economics, specifically the application of statistics, computer science, and data science to the field of economics. Traditionally, econometrics dominated the field of applied economics. However, as critisized by Peter Swann [1], econometrics often times provides dissapointing results and calls for action for a larger variety of research methods. Therefore, we would like to compare traditional econometric techniques with machine learning to improve upon the econometric techniques as well as provide an alternative when econometrics does not suffice.

1.2 Data

The following is a list of all external data series used, a description of each, and the data source.

1. S&P 500 - The Standard & Poor's 500 Index is a daily time series representing a weighted index of companies selected by economists and is often used as leading indicator for of U.S. equities and a performance measure of the market. We use this to calculate our first target variable, the 30-day S&P 500 volatility. Source: Yahoo Finance
2. U.S. Recession - A daily time series of U.S. recession binary dummies. A recession is categorized as negative economic growth for at least two consecutive quarters. We use this time series as our second target variable. Source: FRED
3. CBOE VIX - A daily time series index for future 30-day volatility of the S&P 500 calculated by the Chicago Board Options Exchange (CBOE). Often refered to as "The Fear Index". Source: Yahoo Finance
4. U.S. Effective Federal Funds Rate - A daily time series for what is often refered to as "the interest rate". Represents the rate at which the Federal Reserve sets that banks can borrow from each other at. Source: FRED
5. U.S. Presidential Approval Ratings - Data originally adpated from the Gallup Poll that contains approval ratings, disapproval ratings, and unknown ratings for each president. The data is not in a time series format, but was converted to a daily time series of monthly averages. Source: UCSB [from Gallup Poll]

Note that "daily" refers to daily observances for when the stock market is open. Non-stock time series data has been adjusted.

The following is a list of all daily time series created from the external data sources and their descriptions.

1. S&P 500 30-Day Volatility - A measure of how volatile the S&P 500 is over a 30 day period. Our first target variable. Created by taking the 30-day volatility of the S&P 500.
2. S&P 500 30-Day Volatility Lagged Values - Lagged values from L(1) to L(30) of the S&P 500 30-Day Volatility.
3. S&P 500 Lagged Values - Lagged values from L(1) to L(30) of the S&P 500.
4. CBOE VIX Lagged Values - Lagged values from L(1) to L(30) of the CBOE VIX.
5. U.S. Presidential Approval Rating Averages - The daily time series constructed from monthly averages of the external U.S. Presidential Approval Ratings.
6. U.S. Effective Federal Funds Rate Momentum - The momentum of the U.S. Effective Federal Funds Rate calculated from taking the daily difference of the series on itself.

The range of dates used for the econometric models is February 12, 1990 to May 19, 2017. The range of dates used for the machine learning models is a subset of econometric models date range minus the first 30 indicies. This is to hardcode those values into the machine learning models as lagged values opposed to autofitting that the econometric models use.

For the econometric models, we only use the target time series alone to build the ARIMA models. The machine learning models are built off the target variable as well as all our other data, adjusted as needed.

1.3 Data Analysis Tools

We use two tools for data analysis as follows:

1. R - R is a programming language and software environment commonly used by statisticians, economists, and data scientists. We use R to do all of our quantitive data analysis. We import various packages as needed.
2. Microsoft Excel - Excel is a spreadsheet application. We use Excel to do light data manipulatoin such as verifying and compiling our time series before loading them into R. Any other spreadsheet document would suffice as well.

2 Forecasting Methods

Traditionally, economics and finance have relied heavily on econometrics for time series forecasting. Our goal is to see if we can out perform our econometrics techniques with machine learning.

2.1 Econometrics

Basic econometrics revolves around linear regression, which takes the form of the following:

$$y_i = \beta_0 1 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^\top \boldsymbol{\beta} + \varepsilon_i, \qquad i = 1, \ldots, n,$$

Within the time series frame work, we usually work with forms of AR, MA, or ARMA models which are representations of random processes.

2.1.1 ARIMA

An autoregressive integrated moving average model, also known as ARIMA, is a generalized form of the ARMA model that adds an integration term. One of the powers of ARIMA is that it can convert a non-stationary series to stationary series and if this is not necessary it will just apply an ARMA model to the series. Along with this, ARIMA in general is very good at predicting time series [1]. ARIMA models take the for of:

$$\left(1 - \sum_{i=1}^{p} \phi_i L^i\right) (1 - L)^d X_t = \delta + \left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \varepsilon_t.$$

For the previous reasons and the fact that many previous research papers succesfully use and rely on ARIMA for forecasting [2][3][4], we chose to use ARIMA as our benchmark for econometric time series forecasting.

2.1.1.1 Model Fitting

We use the R package "forecast" and the function "auto.arima" which automatically selects the best ARIMA model based on error statistics. The values for the AR, MA, and I term are selected accordingly.

2.2 Machine Learning

Machine learning is an umbrella term for algorithms that allow a computer to "learn" without explicitly being programmed. Although machine learning has been around since around the conception of the ARMA model in the 1950s, recent developments, improvements in technology, and a change from knowledge driven to data driven approaches have propelled machine learning into being widely accepted in mainstream data analysis [5]. We choose three distinct machine learning techniques, LASSO regressions, decision trees implemented through random forests, and artificial neural networks, each with their own advantages and disadvantages.

2.2.1 LASSO Regressions

The goal of LASSO is to shrink coefficients to 0 through a penalty term, lambda. Through cross validation, we can find our optimal level of lambda which in turn gives us the optimal number of shrinkage. LASSO takes the form of:

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| = \mathrm{RSS} + \lambda\sum_{j=1}^{p}|\beta_j|.$$

2.2.1.1 Model Fitting

We use the R package “glmnet” and the function “cv.glmnet”. This function is capable of fitting a LASSO regression that has its parameters selected from cross validation.

2.2.2 Random Forests

The next machine learning method attempted is decision trees. Decision trees are exactly what the name says, they are (computer science) trees that make decisions. We use a random forest setup, which solves the problems of high variance and correlated trees that can arrise from simple decision tree setups. One of the main advantages of decision trees when compared to other methods are that they are genearlly robust to noise and missing values. A potentially large disadvantage of decision trees, and by extension random forests, is computational complexity of large and complex trees.

2.2.2.1 Model Fitting

We use the R package “randomForest” and the similarily named “randomForest” function to fit the model. This function implements Breiman’s random forest algorithm [8].

2.2.3 Artificial Neural Networks

Artificial Neural Networks are modeled after the human brain and solve problems similarly so. Neural networks are able to find complex connections that are otherwise unknown or difficult to find. Neural networks are now on the frontier of machine learning and are in a category of their own, deep learning. They are not perfect as they are prone to overfitting and require vast amounts of data and tuning. However, if tuned properly and with the right data they can provide extremely useful insights.

2.2.3.1 Model Fitting

We use the R package “neuralnet” and the similarily named “neuralnet” function to fit the model. Fitting a neural network is not as simple as fitting a LASSO regression or random forest. We must select hidden layers and nodes per layer, which algorithm to use, learning rate, and many more. Other than determining the number of hidden layers and nodes, which we will cover later in this paper, we use the neuralnet defaults which can be found on the R documentation for the package [9].

2.3 Performance Measures

In order to compare models with each other, we need to devise a set of measurements to quantify which models out perform others. In addition to that, we need to add in a more qualitative measure in feasibility because a perfect solution will not only be quantativley accurate, but also practical.

2.3.1 Mean Absolute Percentage Error

The mean absolute percentage error (MAPE) is a good measure for forecasting accuracy because it is unit neutral in that it returns accuracy in the form of a percentage. The formula for MAPE is as follows:

$$\mathrm{M} = \frac{100}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|,$$

For **A** being the actual value and **F** being the forecasted value. We will use the MAPE to measure our accuracy when we are using a linear regression framework in forecasting the S&P 500 volatility.

2.3.2 Prediction Error and Confusion Matrix

Within the classification setting, we cannot simply use MAPE as a measure. Considering the fact that our target variable is a binary dummy, we use a prediction accuracy which is defined as the average rate of succesful predictions. However, this does not prove to be a very good measure in all situations. Therefore, we also introduce a second classification measure through confusion matricies. The confusion matricies show the number of correctly classified predictions and incorrectly classified predictions and where they have been incorrectly classified. This is very important specifically for our data set in that we only care about recession predictions and do not care about non-recession predictions.

2.3.3 Feasibility

Our last measure, which we will apply to both test settings, is feasibility. We define feasibility as how realistic an implementation is. In this measure, we will factor in computational complexity, flexibility of applications, and if there is a need for the model.

2.4 Testing Schemes

In order to come up with our error statistics, we need to run different testing schemes by splitting the data into a training set of two thirds of the data and a test set of one thirds. We run different testing schemes for econometrics and machine learning. For econometrics we will using backtesting and for machine learning we will use random sampling and cross validation. [6][7]

2.4.1 Econometrics Testing Schemes

We follow testing schemes devised for time series analysis. It follows that we split the data into the first two-thirds as training data and the last two-thirds as test data. We use the training set to train the model and the test set to predict on and calculate the accuracy of prediction. We do this so we can refit the model at each iteration step of testing. However, the way we refit the model differs depending on which type of backtest window we use. [6]

2.4.1.1 Recursive Window Backtest

A recursive window backtest starts by fitting the model to the two-thirds training set and makes a 1-step-ahead forecast. We then store this value as our first predicted value. For the second iteration, we update our training set from the first two-thirds to the first two-thirds plus the first value of the test set. From the updated set, we refit the model and do another 1-step-ahead forecast and store the value again. We repeat until we iterate over the entire test set. With the resulting set of predicted values, we can then find the MAPE by comparing the predicted values with the actual. [6]

2.4.1.2 Rolling Window Backtest

The rolling window backtest follows the same methods as the recursive window backtest except for one change. With every iteration, the window for the training set "rolls" such that when a new entry is added to the the training set, the last (earliest) entry is removed [6].

2.4.2 Machine Learning Testing Schemes

We generally split machine learning training and test sets into sets of two-thirds and one-thirds as well. However, we randomly split to get an accurate sample and so we do not overfit our data to a biased set [7].

2.4.2.1 Cross Validation

We use cross validation to find a specific set of parameters that do not suffer from overfitting. This is achieved by testing the set of parameters over randomized selections of the entire data set. We implement this in a k-fold cross validation with k=5. 5-fold cross validation randomly splits the data set into five equal partitions where we will train our model on and predict over the remaining set that was left out of the training. We can then get the spread of our predictions and see how robust our model is over different data sets.

3. S&P 500 Volatility Forecasting Results

The following is a summary of the results from the different testing schemes on S&P 500 volatility forecasting with a comparrison of the different models at the end.

3.1 Forecasting with ARIMA

3.1.1 Recursive Backtest

MAPE: 2.913536

3.1.2 Rolling Backtest

MAPE: 2.927933

3.2 Forecasting with LASSO

MAPE: 2.743576272 LASSO Shrinkage

3.2.1 Cross Validation

MAPE: 2.879230238 Variance: 0.004070765755

3.3 Forecasting with Random Forests

MAPE: 2.732716043 Importance Plot

3.3.1 Cross Validation

MAPE: 2.673993581 Variance: 0.01175814229

3.4 Forecasting with Nueral Netowrks

Our initial parmeters was 7 hidden layers of 100, 70, 60, 50, 40, 30 and 20 nodes.

MAPE: 4.060531678

3.4.1 Parameter Tuning

The following steps were used to tune the neural network parameters: 1. Reduce the variables to only the non-shrunk variables from the LASSO results. 2. Reduce the number of hidden layers and nodes per layer accordingly. 3. Randomly assign values for layers and nodes to find an optimal hidden layer setup.

After parameter tuning, we converged to a hiddlen layer set up of 3 hidden layers of 20, 13 and 1 node(s) which result in a new and lower MAPE.

New MAPE: 2.288863265

3.4.2 Cross Validation

MAPE: 2.356741945 Variance: 0.01514343111

3.5 Model Comparrison

The machine learning models, besides the untuned neural network, all out perform the ARIMA models. The model with the lowest MAPE score is the tuned neural network with a MAPE of 2.288863265. This is far better than the ARIMA backtesting MAPEs are. However, we must consider feasibility. The machine learning techniques require an abundance of data. This is not a problem if the data is easily accessible, such as the data gathered for this paper. Neural networks in particular have long training times, especially for large and complex networks. However, this is also not an issue for the current data set. If we were to find and integrate more data points, this could drastically increase computational complexity. We did see the current results were enough to out perform ARIMA. The neural network is our preferred method for linear regression time series forecasting.

4. U.S. Recession Prediction Results

The following is a summary of the results from the different testing schemes on U.S. Recession prediction with a comparrison of the different models at the end.

4.1 Predicting with ARIMA

4.1.1 Recursive Window Backtest

4.1.2 Rolling Window Backtest

4.2 Predicting with LASSO

4.2.1 Cross Validation

4.3 Predicting with Random Forests

4.3.1 Cross Valiation

4.4 Predicting with Nueral Netowrks

4.4.1 Parameter Tuning

4.4.2 Cross Validation

4.5 Model Comparrison

5. Conclusion