A Comparison of Econometrics and Machine Learning for Time Series Forecasting and Prediction: The S&P 500 and U.S. Recessions

Author: Lester J. Pi
Faculty Advisor: Professor Randall R. Rojas

June 16, 2017

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 paper focuses on various time series forecasting by empirically 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 or show that the econometric models are sufficient. The machine learning 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) and the classification accuracy along with confusion matricies. 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]
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]

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 [2]
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 [3]
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) [4]

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 [5][6][7], 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 [8]. 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 [11].

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 [12].

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. We will compare these measures with each other to determine which model is best suited for each data set.

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. The lower the MAPE, the lower the forecast accuracy.

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.

2.4 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. [9][10]

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. [9]

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. [9]

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 [9].

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 [10].

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

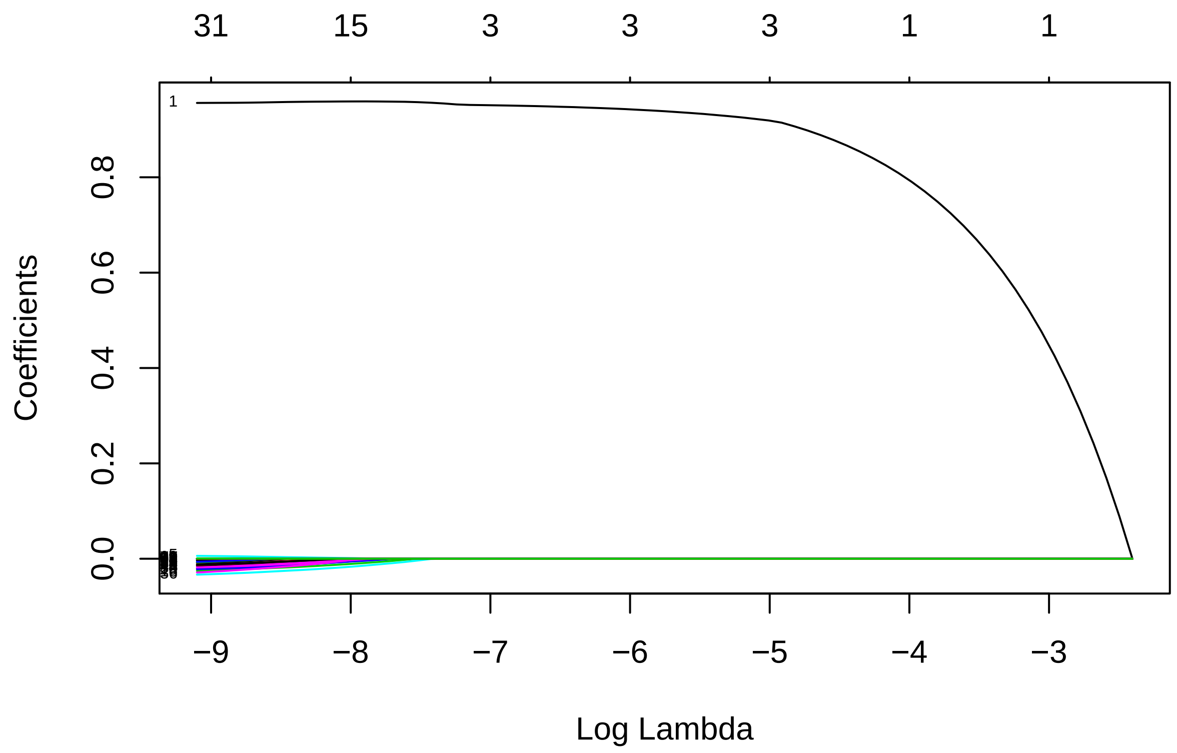
3.1.2 Rolling Backtest

MAPE: 2.927933

3.2 Forecasting with LASSO

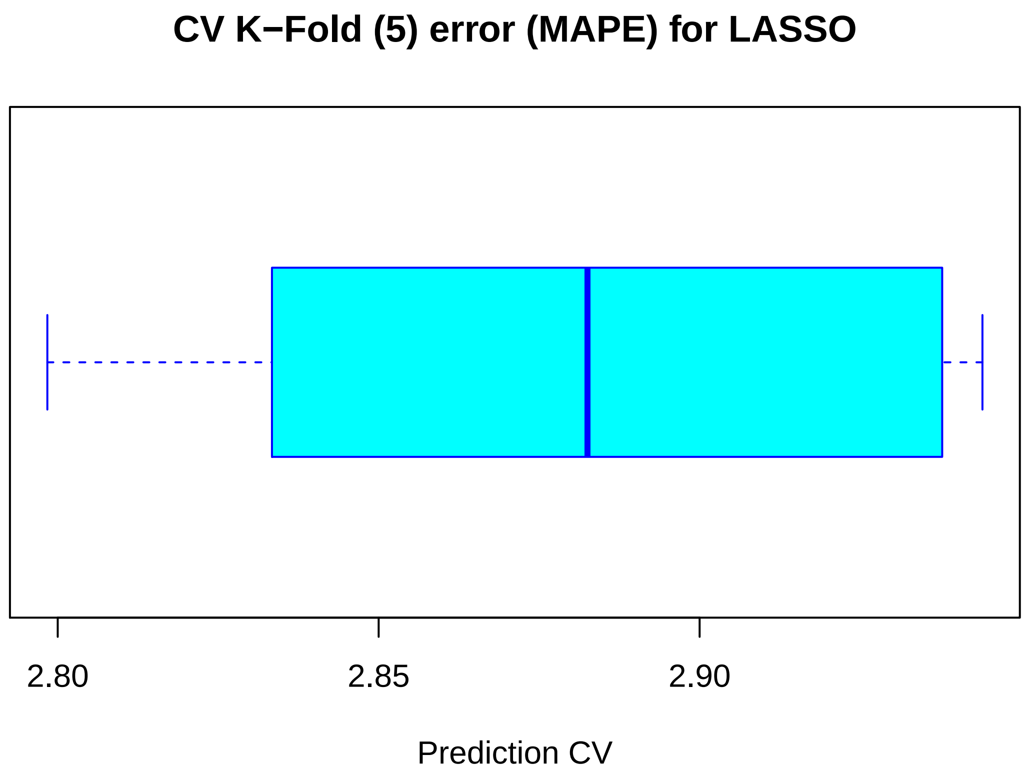
MAPE: 2.743576272

We also generate a shrinkate plot to show how the variables variables increase as lambda increases. Remember that the function used automatically selects the optimal level of lambda for us.



3.2.1 Cross-Validation

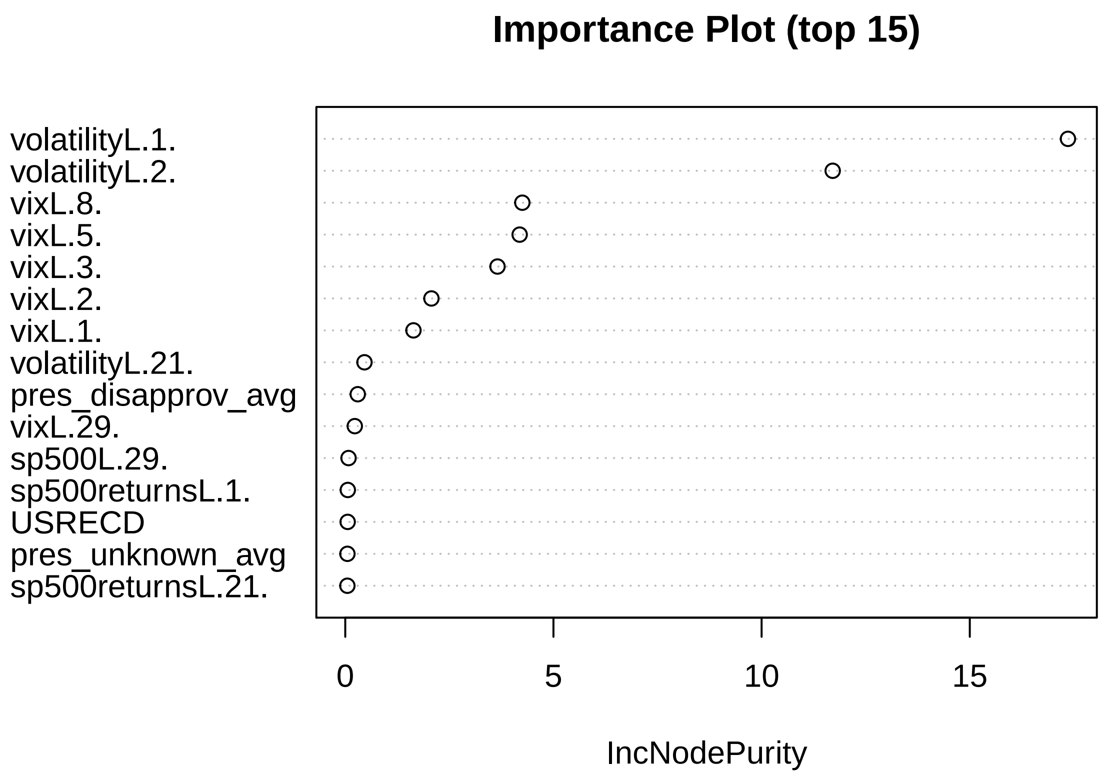
MAPE: 2.879230238 Variance: 0.004070765755



3.3 Forecasting with Random Forests

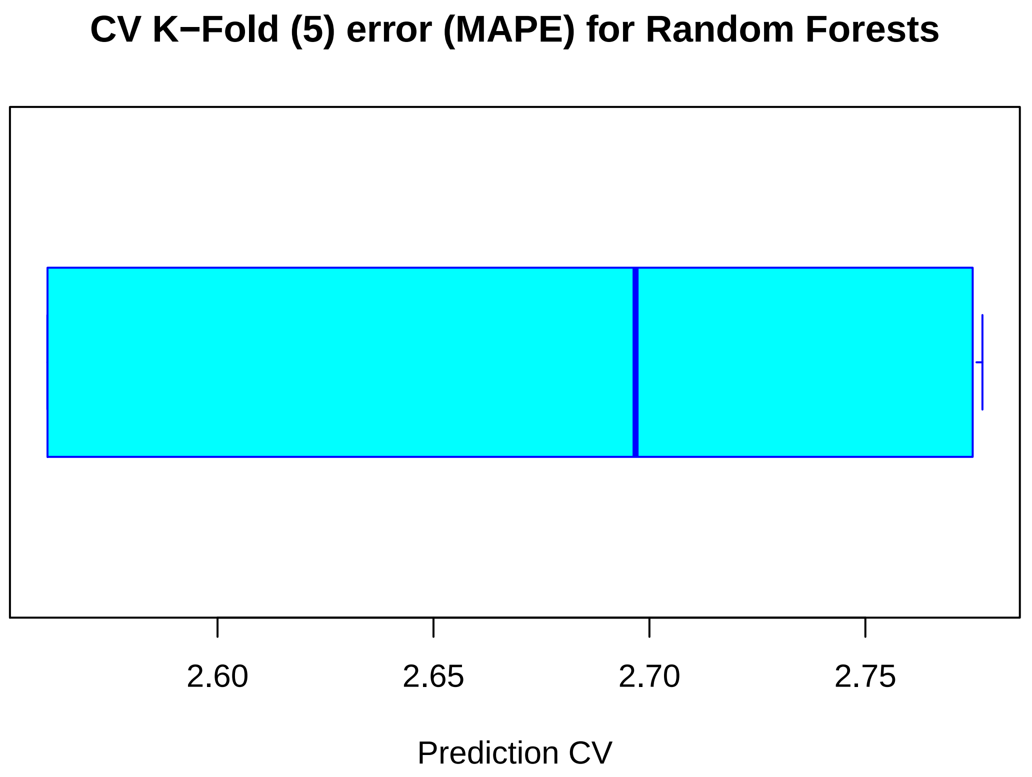
MAPE: 2.732716043

We also generate a variable importance plot. This gives us an idea which variables are important in predicting the outcome. Below is the top 15 important variables.



3.3.1 Cross-Validation

MAPE: 2.673993581 Variance: 0.01175814229



3.4 Forecasting with Nueral Networks

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

MAPE: 4.060531678

3.4.1 Parameter Tuning

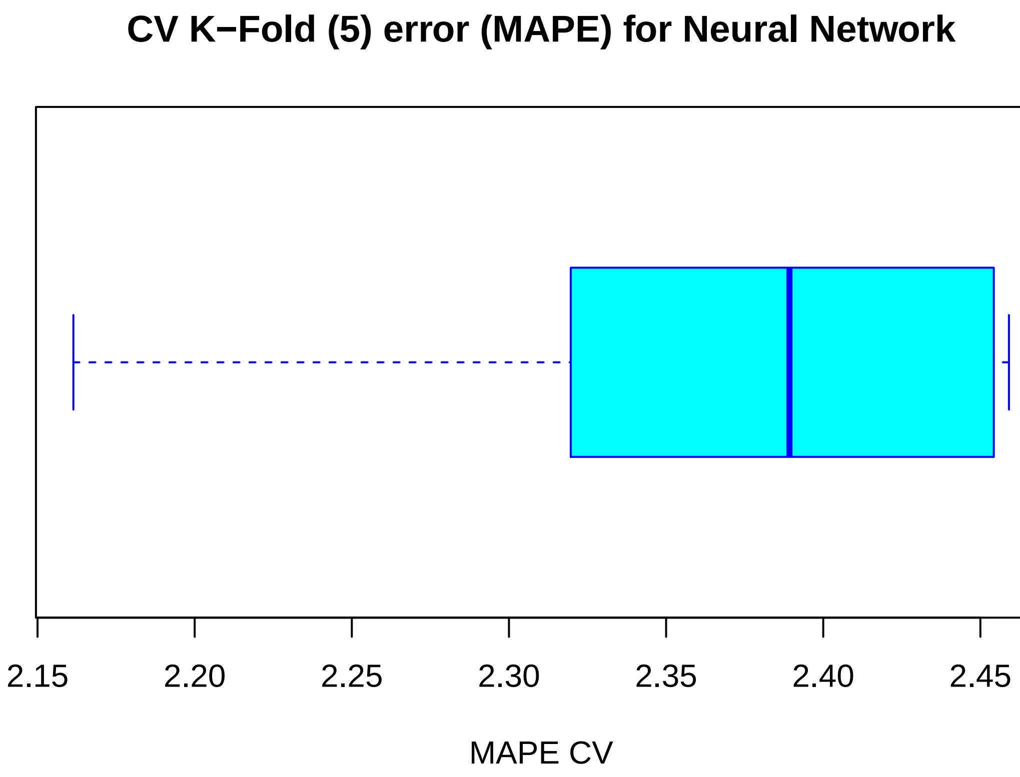
The following steps were used to tune the neural network parameters: 1. Reduce the variables to only the non-shrunk variables from the LASSO results. 2. Reduce the number of hidden layers and nodes per layer accordingly. 3. Randomly assign values for layers and nodes to find an optimal hidden layer setup.

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 random forest model showed us that the most important variables were the lagged values of the S&P 500, which explains why ARIMA is difficult to beat in this setting. 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. 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 setups were enough to out perform ARIMA.

When comparing the machine learning methods with each other, we examine the same metrics as comparing machine learning with ARIMA. After cross-validation, we can see that neural networks still outperform the other models even after considering their variance.

The neural network is our preferred method for linear regression time series forecasting if the neural network can be tuned accordingly.

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. The binary classification prediction section is different than the continuous forecast section in that we would like our model to generate a likilood function to give us a probability of the recession occuring. This is because we would like to know if there is an oncoming recession and how long it will last.

For the machine learning portions, we follow the same procedures in shrinking the data set with LASSO when needed and tuning the neural network in the same way. The cross-validation sections that follow will not include a boxplot and variance of the accuracy because the the confusion matrix is a better representation of performance. We denote "0" as days without a recession and "1" as days with recessions.

4.1 Predicting with ARIMA

4.1.1 Recursive Window Backtest

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1984    0
##          1    1  295
##
##                Accuracy : 0.9995614
##                  95% CI : (0.9975587, 0.9999889)
##     No Information Rate : 0.870614
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.998056
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9994962
##             Specificity : 1.0000000
##          Pos Pred Value : 1.0000000
##          Neg Pred Value : 0.9966216
##              Prevalence : 0.8706140
##          Detection Rate : 0.8701754
##    Detection Prevalence : 0.8701754
##       Balanced Accuracy : 0.9997481
##
##        'Positive' Class : 0
##
```

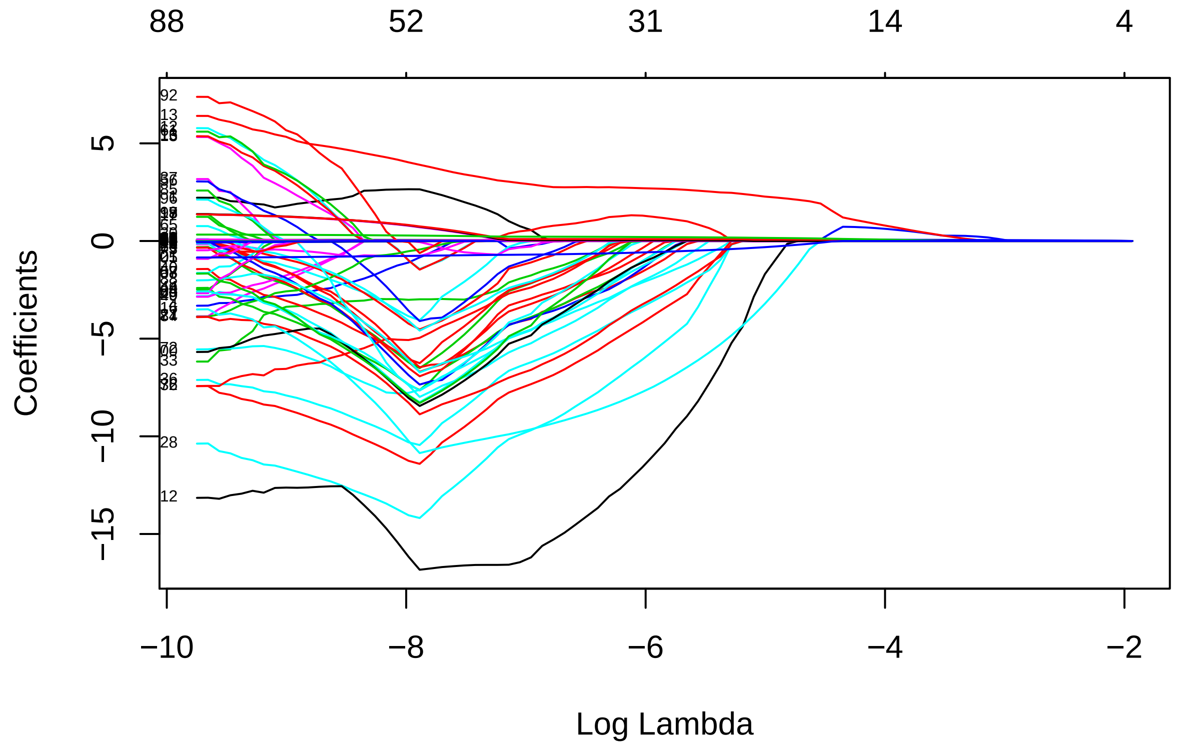
4.1.2 Rolling Window Backtest

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1984    0
##          1    1  295
##
##                Accuracy : 0.9995614
##                  95% CI : (0.9975587, 0.9999889)
##     No Information Rate : 0.870614
##     P-Value [Acc > NIR] : <0.0000000000000002
##
```

```
##                  Kappa : 0.998056
##  Mcnemar's Test P-Value : 1
##
##            Sensitivity : 0.9994962
##            Specificity : 1.0000000
##         Pos Pred Value : 1.0000000
##         Neg Pred Value : 0.9966216
##             Prevalence : 0.8706140
##         Detection Rate : 0.8701754
##   Detection Prevalence : 0.8701754
##      Balanced Accuracy : 0.9997481
##
##       'Positive' Class : 0
##
```

4.2 Predicting with LASSO

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2058  155
##          1   40   74
##
##               Accuracy : 0.9162011
##                 95% CI : (0.9041969, 0.9271447)
##     No Information Rate : 0.90159
##     P-Value [Acc > NIR] : 0.008752342
##
##                  Kappa : 0.3916943
##  Mcnemar's Test P-Value : 0.0000000000000003248895
##
##            Sensitivity : 0.9809342
##            Specificity : 0.3231441
##         Pos Pred Value : 0.9299593
##         Neg Pred Value : 0.6491228
##             Prevalence : 0.9015900
##         Detection Rate : 0.8844005
##   Detection Prevalence : 0.9510099
##      Balanced Accuracy : 0.6520392
##
##       'Positive' Class : 0
##
```



4.2.1 Cross-Validation

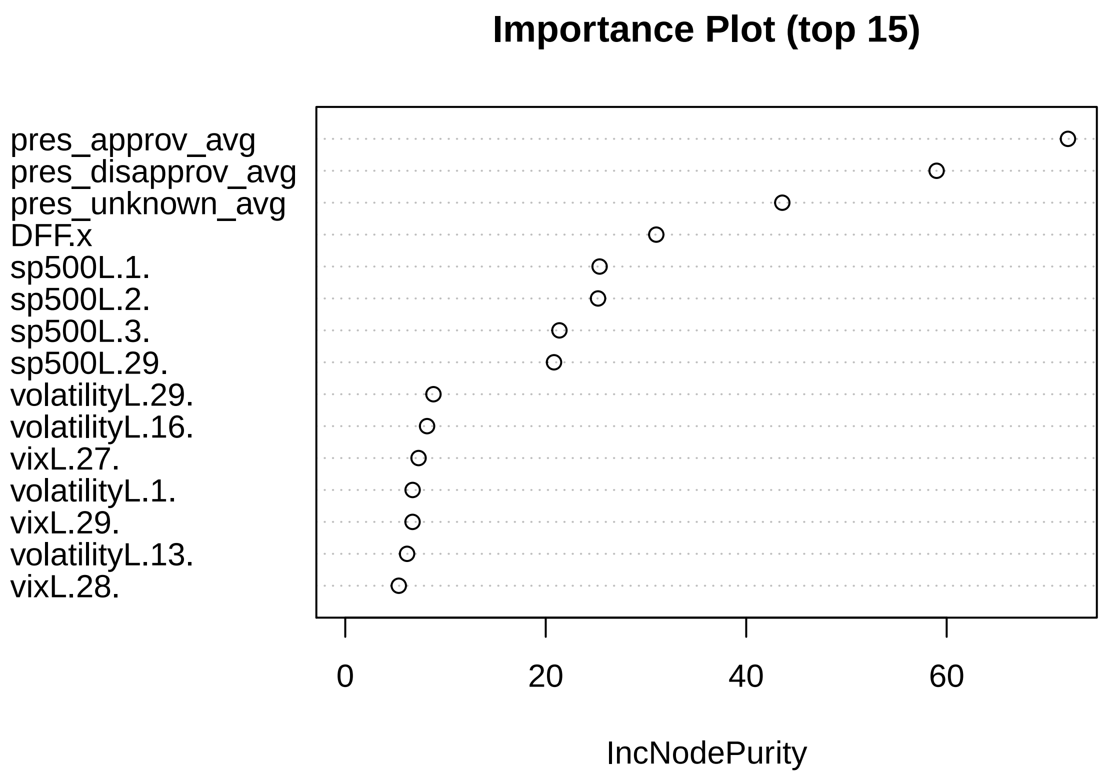
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1202   96
##          1   27   44
##
##                Accuracy : 0.9101534
##                  95% CI : (0.8937461, 0.9247736)
##     No Information Rate : 0.8977356
##     P-Value [Acc > NIR] : 0.06855261
##
##                   Kappa : 0.3739772
##  Mcnemar's Test P-Value : 0.0000000008713454
##
##             Sensitivity : 0.9780309
##             Specificity : 0.3142857
##          Pos Pred Value : 0.9260401
##          Neg Pred Value : 0.6197183
##              Prevalence : 0.8977356
##          Detection Rate : 0.8780131
##    Detection Prevalence : 0.9481373
##       Balanced Accuracy : 0.6461583
```

```
##
##        'Positive' Class : 0
##
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1204  100
##          1   23   42
##
##                Accuracy : 0.9101534
##                  95% CI : (0.8937461, 0.9247736)
##     No Information Rate : 0.8962747
##     P-Value [Acc > NIR] : 0.04833643
##
##                   Kappa : 0.3643927
##  Mcnemar's Test P-Value : 0.000000000007247442
##
##             Sensitivity : 0.9812551
##             Specificity : 0.2957746
##          Pos Pred Value : 0.9233129
##          Neg Pred Value : 0.6461538
##              Prevalence : 0.8962747
##          Detection Rate : 0.8794741
##    Detection Prevalence : 0.9525201
##       Balanced Accuracy : 0.6385149
##
##        'Positive' Class : 0
##
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1203   95
##          1   25   46
##
##                Accuracy : 0.9123448
##                  95% CI : (0.8960998, 0.9267925)
##     No Information Rate : 0.8970051
##     P-Value [Acc > NIR] : 0.03198076
##
##                   Kappa : 0.3920194
##  Mcnemar's Test P-Value : 0.0000000002999405
##
##             Sensitivity : 0.9796417
##             Specificity : 0.3262411
##          Pos Pred Value : 0.9268105
##          Neg Pred Value : 0.6478873
##              Prevalence : 0.8970051
##          Detection Rate : 0.8787436
##    Detection Prevalence : 0.9481373
##       Balanced Accuracy : 0.6529414
##
##        'Positive' Class : 0
```

```
##
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1216   93
##          1   23   37
##
##                Accuracy : 0.9152666
##                  95% CI : (0.8992424, 0.92948)
##     No Information Rate : 0.9050402
##     P-Value [Acc > NIR] : 0.1052756
##
##                   Kappa : 0.3505215
##  Mcnemar's Test P-Value : 0.0000000001489087
##
##             Sensitivity : 0.9814366
##             Specificity : 0.2846154
##          Pos Pred Value : 0.9289534
##          Neg Pred Value : 0.6166667
##              Prevalence : 0.9050402
##          Detection Rate : 0.8882396
##    Detection Prevalence : 0.9561724
##       Balanced Accuracy : 0.6330260
##
##        'Positive' Class : 0
##
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1208  110
##          1   11   40
##
##                Accuracy : 0.9116143
##                  95% CI : (0.8953149, 0.9261199)
##     No Information Rate : 0.890431
##     P-Value [Acc > NIR] : 0.00571802
##
##                   Kappa : 0.3625673
##  Mcnemar's Test P-Value : < 0.0000000000000000222
##
##             Sensitivity : 0.9909762
##             Specificity : 0.2666667
##          Pos Pred Value : 0.9165402
##          Neg Pred Value : 0.7843137
##              Prevalence : 0.8904310
##          Detection Rate : 0.8823959
##    Detection Prevalence : 0.9627465
##       Balanced Accuracy : 0.6288214
##
##        'Positive' Class : 0
##
```

4.3 Predicting with Random Forests

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2010    0
##          1   88  229
##
##                Accuracy : 0.9621831
##                  95% CI : (0.9536142, 0.9695614)
##     No Information Rate : 0.90159
##     P-Value [Acc > NIR] : < 0.00000000000000022204
##
##                   Kappa : 0.8180345
##  Mcnemar's Test P-Value : < 0.00000000000000022204
##
##             Sensitivity : 0.9580553
##             Specificity : 1.0000000
##          Pos Pred Value : 1.0000000
##          Neg Pred Value : 0.7223975
##              Prevalence : 0.9015900
##          Detection Rate : 0.8637731
##    Detection Prevalence : 0.8637731
##       Balanced Accuracy : 0.9790276
##
##        'Positive' Class : 0
##
```



4.3.1 Cross Valiation

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1180    0
##          1   49  140
##
##                Accuracy : 0.9642075
##                  95% CI : (0.9529544, 0.9734054)
##     No Information Rate : 0.8977356
##     P-Value [Acc > NIR] : < 0.00000000000000022204
##
##                   Kappa : 0.8312347
##  Mcnemar's Test P-Value : 0.000000000007025137
##
##             Sensitivity : 0.9601302
##             Specificity : 1.0000000
##          Pos Pred Value : 1.0000000
##          Neg Pred Value : 0.7407407
##              Prevalence : 0.8977356
##          Detection Rate : 0.8619430
##    Detection Prevalence : 0.8619430
##       Balanced Accuracy : 0.9800651
```

```
##
##        'Positive' Class : 0
##
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1175    0
##          1   45  149
##
##                Accuracy : 0.9671293
##                  95% CI : (0.9562618, 0.9759249)
##     No Information Rate : 0.8911614
##     P-Value [Acc > NIR] : < 0.00000000000000022204
##
##                   Kappa : 0.8503843
##  Mcnemar's Test P-Value : 0.00000000005412161
##
##             Sensitivity : 0.9631148
##             Specificity : 1.0000000
##          Pos Pred Value : 1.0000000
##          Neg Pred Value : 0.7680412
##              Prevalence : 0.8911614
##          Detection Rate : 0.8582907
##    Detection Prevalence : 0.8582907
##       Balanced Accuracy : 0.9815574
##
##        'Positive' Class : 0
##
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1168    0
##          1   48  153
##
##                Accuracy : 0.9649379
##                  95% CI : (0.9537796, 0.974037)
##     No Information Rate : 0.8882396
##     P-Value [Acc > NIR] : < 0.00000000000000022204
##
##                   Kappa : 0.8446965
##  Mcnemar's Test P-Value : 0.00000000001170021
##
##             Sensitivity : 0.9605263
##             Specificity : 1.0000000
##          Pos Pred Value : 1.0000000
##          Neg Pred Value : 0.7611940
##              Prevalence : 0.8882396
##          Detection Rate : 0.8531775
##    Detection Prevalence : 0.8531775
##       Balanced Accuracy : 0.9802632
##
##        'Positive' Class : 0
```

```
##
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1189    0
##          1   50  130
##
##                Accuracy : 0.963477
##                  95% CI : (0.9521303, 0.9727727)
##     No Information Rate : 0.9050402
##     P-Value [Acc > NIR] : < 0.00000000000000022204
##
##                   Kappa : 0.8187187
##  Mcnemar's Test P-Value : 0.000000000004218937
##
##             Sensitivity : 0.9596449
##             Specificity : 1.0000000
##          Pos Pred Value : 1.0000000
##          Neg Pred Value : 0.7222222
##              Prevalence : 0.9050402
##          Detection Rate : 0.8685172
##    Detection Prevalence : 0.8685172
##       Balanced Accuracy : 0.9798224
##
##        'Positive' Class : 0
##
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1203    0
##          1   36  130
##
##                Accuracy : 0.9737034
##                  95% CI : (0.963779, 0.9815158)
##     No Information Rate : 0.9050402
##     P-Value [Acc > NIR] : < 0.00000000000000022204
##
##                   Kappa : 0.8638804
##  Mcnemar's Test P-Value : 0.000000005433087
##
##             Sensitivity : 0.9709443
##             Specificity : 1.0000000
##          Pos Pred Value : 1.0000000
##          Neg Pred Value : 0.7831325
##              Prevalence : 0.9050402
##          Detection Rate : 0.8787436
##    Detection Prevalence : 0.8787436
##       Balanced Accuracy : 0.9854722
##
##        'Positive' Class : 0
##
```

4.4 Predicting with Neural Networks

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0  194  120
##          1 1904  109
##
##                Accuracy : 0.1302106
##                  95% CI : (0.1167941, 0.1445653)
##     No Information Rate : 0.90159
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.0965426
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.09246902
##             Specificity : 0.47598253
##          Pos Pred Value : 0.61783439
##          Neg Pred Value : 0.05414804
##              Prevalence : 0.90159003
##          Detection Rate : 0.08336914
##    Detection Prevalence : 0.13493769
##       Balanced Accuracy : 0.28422578
##
##        'Positive' Class : 0
##
```

4.4.1 Parameter Tuning

After following the same parameter tuning methods as in the econometric framework, we converged to the neural network setup of 2 layers with nodes of 19 and 3.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0  862   71
##          1 1236  158
##
##                Accuracy : 0.4383326
##                  95% CI : (0.4180461, 0.4587746)
##     No Information Rate : 0.90159
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0308705
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.4108675
##             Specificity : 0.6899563
##          Pos Pred Value : 0.9239014
##          Neg Pred Value : 0.1133429
##              Prevalence : 0.9015900
##          Detection Rate : 0.3704340
##    Detection Prevalence : 0.4009454
##       Balanced Accuracy : 0.5504119
##
##        'Positive' Class : 0
##
```

4.4.2 Cross-Validation

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 585  59
##          1 644  81
##
##                Accuracy : 0.4864865
##                  95% CI : (0.4596981, 0.5133329)
##     No Information Rate : 0.8977356
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0191381
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.4759967
##             Specificity : 0.5785714
##          Pos Pred Value : 0.9083851
##          Neg Pred Value : 0.1117241
##              Prevalence : 0.8977356
##          Detection Rate : 0.4273192
##    Detection Prevalence : 0.4704164
##       Balanced Accuracy : 0.5272841
##
##        'Positive' Class : 0
##
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 480  64
##          1 750  75
##
##                Accuracy : 0.4054054
##                  95% CI : (0.3792593, 0.4319585)
##     No Information Rate : 0.898466
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.022011
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.39024390
##             Specificity : 0.53956835
##          Pos Pred Value : 0.88235294
##          Neg Pred Value : 0.09090909
##              Prevalence : 0.89846603
##          Detection Rate : 0.35062089
##    Detection Prevalence : 0.39737034
##       Balanced Accuracy : 0.46490612
##
##        'Positive' Class : 0
##
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 594  75
##          1 644  56
##
##                Accuracy : 0.4747991
##                  95% CI : (0.4480599, 0.5016468)
##     No Information Rate : 0.9043097
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.0315141
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.4798061
##             Specificity : 0.4274809
##          Pos Pred Value : 0.8878924
##          Neg Pred Value : 0.0800000
##              Prevalence : 0.9043097
##          Detection Rate : 0.4338934
##    Detection Prevalence : 0.4886779
##       Balanced Accuracy : 0.4536435
##
##        'Positive' Class : 0
##
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1127  109
##          1  103   30
##
##                Accuracy : 0.8451424
##                  95% CI : (0.8248759, 0.8639147)
##     No Information Rate : 0.898466
##     P-Value [Acc > NIR] : 1.0000000
##
##                   Kappa : 0.1346655
##  Mcnemar's Test P-Value : 0.7312965
##
##             Sensitivity : 0.9162602
##             Specificity : 0.2158273
##          Pos Pred Value : 0.9118123
##          Neg Pred Value : 0.2255639
##              Prevalence : 0.8984660
##          Detection Rate : 0.8232286
##    Detection Prevalence : 0.9028488
##       Balanced Accuracy : 0.5660438
##
##        'Positive' Class : 0
##
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 689  87
##          1 521  72
##
##                Accuracy : 0.5558802
##                  95% CI : (0.5290986, 0.5824215)
##     No Information Rate : 0.8838568
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0101818
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.5694215
##             Specificity : 0.4528302
##          Pos Pred Value : 0.8878866
##          Neg Pred Value : 0.1214165
##              Prevalence : 0.8838568
##          Detection Rate : 0.5032871
##    Detection Prevalence : 0.5668371
##       Balanced Accuracy : 0.5111258
##
##        'Positive' Class : 0
##
```

4.5 Model Comparrison

On the surface, the results show show that ARIMA does an excellent job in classifying the recession. On closer inspection, however, the ARIMA model is predicting a recession tomorrow if there is a recession today and similar for non-recessions. The test set was split from the training set in the middle of a recession, therefore it starts with a string of recession days with no other recessions after this one ends. This does not give us a good method to predict oncoming recessions and only tells us we are likely to be in a recession tomorrow if we are in one today. Therefore, ARIMA fails at the task of giving a liklihood estimate and the accuracy rating is misleading.

We can see the results from LASSO have a high accuracy. However, the accuracy is misleading as it was in ARIMA because there is a disproportionate ammount of "0"s compared to "1"s in the data set. It is important to look at the confusion matrix to detect whether there are false positives and false negatives. We can see that LASSO actually misclassified "1"s a lot. It wrongly estimated a "0" 155 times and wrongly estimated a "1" 40 times and only correctly estimated a "1" 74 times. This is still better than ARIMA for giving us a liklihood estimate, but it is not strong enough.

This gives us a very good prediction accuracy, but we must remember that the prediction accuracy is deceiving. Therefore, we must look at the confusion matrix. From the confusion matrix, we see that it does not wrongly predict a “0” when the actual was a “1”. It does however predict a “1” 95 times when it should have been a “0”. It correctly predicts a recession 229 times. This is much better than the LASSO results. Based on these results, we can take this as a cautious approach in estimating a recession prediction model.

After running our “best” setup for neural networks, we still do not reach results as good as the previous methods. It actually has a lower accuracy rating than if we were to predict all “0”s. This could be due to many reasons including, but not limitted to, over/underfitting and not enough data. The neural network does not seem fit for this data.

After cross-validation, we are seeing similar results we saw in the test set. The neural network is performing very poorly. LASSO looks to be doing good, but when only considering correctly and incorrectly classifying “1” (excluding predicting “0” corrrectly), it does not perform well. Decision trees are also overclassifying “1”s, but that is not a bad thing as it does not missclassify a “0” as a “1” and is taking a “cautious”, as defined by being rather safe than sorry, approach.

By looking at each machine learning methods’ confusion matrix and their cross-validation results, the decision trees with a random forest implementation performs the best. It was able to correctly classify most of the recession days and did not classify an actual recession day as being not in a recession. It did overclassify the recession, however, predicting there would be a recession day when in actuallity there was not a recession that day. This makes it a more cautious approach from the human perspective. The alogorithm does not know that “1” is bad and “0” is good which is why we need humans to look at the confusion matrix and conclude that it is a cautious model.

5. Conclusion

We rigerously tested the two time series, S&P 500 volatility and U.S. recession dummies, through the two different frameworks of econometrics and machine learning. Using R, we implemented four different models to model our data.

Our first model, ARIMA, falls into the econometrics framework. We chose ARIMA for the econometrics framework because ARIMA is very powerful for time series forecasting and analysis. It is often the model of choice for time series.

For the machine learning framework we chose three models to test, LASSO regression, decision trees implemented as random forests, and artificial neural networks. All three of these models have their own advantages and disadvantages that have been considered for these data sets. The LASSO regression is able to shrink variables to zero. Decision trees are robust to noise and missing values. Neural networks are very powerful and can find complex relationships that are difficult to find otherwise.

We fit the models on the training data splits. Then we run predictions on the test data split which result in a MAPE score for the continuous time series and a confusion matrix for the binary classification time series. For the machine learning techniques, we also run cross-validation to help us compare between models. By comparing these results, we can see which models perform better than others.

For forecasting of the S&P500, all 3 machine learning techniques were able to achieve better MAPEs through testing and cross-validation with neural networks performing with the lowest (best) MAPE of about .7 percentage points lower than ARIMA.

For forecasting of the U.S. recession dummies, we create predictions and models that are able to give us the liklihoods. We saw that ARIMA’s results were a little misleading and we had to have a closer look as the confusion matrix and accuracy did not show the whole truth. The ARIMA model only told us that if we are in a recession, we will continue to be in one. This allowed us to not consider ARIMA for this setting. The machine learning methods proved much more useful with the decision tree model implemented through random forests providing the best results. This model did not incorrectly predict a non-recession day when it was a recession day. It slightly overclassified on the recession days where it would classify some non-recession days as recessions. In other words, this model had some false positives, but no false negatives. This model also showed us that the largest predictors for recessions are presidential approval ratings.

In conclusion, machine learning outperforms ARIMA for both forecasting S&P 500 volatility and predicting a recession. ARIMA is not as powerful as the machine learning methods we tested for these specific time series. The robustness of the machine learning methods allow us to believe that if these techniques were extended to other time series forecasting and analysis, they would again prove more powerful and better models than ARIMA.

Sources

[1] Swann, G. M. P. (2006) "Putting Econometrics in Its Place: A New Direction in Applied Economics"

[2] "Yahoo! Finance". https://finance.yahoo.com/

[3] "Federal Reserve Economic Data". https://fred.stlouisfed.org/

[4] Gerhard Peters, John T. Woolley. "The American Presidency Project". http://www.presidency.ucsb.edu/data/popularity.php

[5] P. Mondal, L. Shit, S. Goswami. "Study of Effectiveness of Time Series Modeling (ARIMA) In Forecasting Stock Prices."

[6] Arul Earnest, Mark I Chen, Donald Ng, Leo Yee Sin. (2014) "Using autoregressive integrated moving average (ARIMA) models to predict and monitor the number of beds occupied during a SARS outbreak in a tertiary hospital in Singapore"

[7] Ayodele A. Adebiyi, Aderemi O. Adewumi, Charles K. Ayo. (2014) "Stock Price Prediction Using the ARIMA Model"

[8] Marr, Marr. "A Short History of Machine Learning - Every Manager Should Read"

[9] Randall R. Rojas. "Economics 403B Applied Econometrics - Lecture 2 Modeling and Forecasting Trend"

[10] Randall R. Rojas. "Economics 412 Fundamentals of Big Data - Machine Learning Part I"

[11] Leo Breiman. (2001) "Random Forests"

[12] Stefan Fritsch, Frauke Guenther, Marc Suling, Sebastian M. Mueller. (2016) "Package 'neuralnet' "