# Predicting Stock Market Index Direction Using ARIMA-Augmented Quantum Random Forest Models

**Integrating Quantum Computing, Machine Learning and Classical Time Series Modeling**

Don Krapohl (Advisor: Dr. Shusen Pu)

2025-10-15

## Table of contents

**1 Introduction** **2**
- 1.1 Background and Motivation . . . 2
- 1.2 Link to Project Code . . . 2
- 1.3 Research Problem . . . 2
- 1.4 Research Objectives . . . 3
- 1.5 Purpose of Study . . . 3
- 1.6 Scope and Limitations . . . 3
- 1.7 Capstone Project Organization . . . 3

**2 Literature Review** **3**
- 2.1 Overview of Stock Market Prediction . . . 3
- 2.2 SARIMA and ARIMA Time Series Model Development . . . 4
- 2.3 GARCH Models of Volatility . . . 4
- 2.4 Vector Autoregressive (VAR) and Multivariate Time Series Analysis . . . 4
- 2.5 Quantum Random Forest . . . 5

**3 Theoretical Framework** **5**
- 3.1 Random Forest Methodology . . . 5
  - 3.1.1 Decision Tree Introduction . . . 5
  - 3.1.2 Splitting Criteria . . . 6
  - 3.1.3 Ensemble Learning . . . 6
  - 3.1.4 Random Forest Algorithm . . . 7
  - 3.1.5 Out-of-Bag Error Estimation . . . 8

3.1.6 Hyperparameters and Tuning . . . . . 8
3.2 Time Series Analysis . . . . . 9
3.2.1 Stationarity and Unit Root Tests . . . . . 9
3.2.2 Autocorrelation and Partial Autocorrelation . . . . . 9
3.2.3 ARMA Model Structure . . . . . 9
3.2.4 Model Selection Criteria . . . . . 9
3.2.5 GARCH for Volatility . . . . . 10
3.2.6 Trend Removal Through Differencing . . . . . 10
3.3 Random Forest with Time Series Hybrid Model . . . . . 10
3.3.1 Feature Engineering Strategy . . . . . 10

**4 References** **10**

# 1 Introduction

## 1.1 Background and Motivation

Short-term stock market forecasting is a common challenge engaged by many millions of analysts and investors daily. Stock market data is frequently non-linear and is influenced by not only financial drivers but also geopolitical and macroeconomic policies and events. Random Forest has demonstrated the ability to handle non-linear, heterogeneous features while being explainable and resistant to overfitting. One basic issue with Random Forest models are that they do not intrinsically have memory and so can miss opportunities that are based on time-based influences in variables.

## 1.2 Link to Project Code

The Jupyter Notebook is available from **https://github.com/dkrapohl/UWF_DataScience_Capstone/**

## 1.3 Research Problem

The objective is to use my course work, current literature, and intent on future research to classify the market movement as either upward or downward. Because Random Forest has no memory I will use both machine learning, time series modeling, and quantum circuits to identify optimal lags and moving averages and introduce these variables during feature engineering.

### 1.4 Research Objectives

I intend to develop a hybrid methodology combining ARMA feature engineering with Random Forest classification, identify optimal lag structures and moving average windows through systematic time series analysis, evaluate model performance using multiple metrics, and determine feature importance for market direction prediction.

### 1.5 Purpose of Study

I will use my coursework, readings, coding, and statistical knowledge to synthesize an approach to analysis that, although not novel in academia, is new to me. I will not be using any of the tools developed in my coursework to identify, train, tune, and measure the models I build so that I may perform real-world analysis of a type I believe to be relevant to many datasets with which I've worked.

### 1.6 Scope and Limitations

The data will be from the United States Standard & Poor's S&P 500 index covering 1990-2024. The forward-looking limits will be 20 days and the predicted outcome will be binary (up/down).

### 1.7 Capstone Project Organization

This project will consist of a section covering the background, theory, recent research, and explanation of: - Random Forest models - Auto Regressive Moving Average (ARMA) models - Generalized AutoRegressive Conditional Heteroskedasticity (GARCH) models of volatility - Vector Autoregressive (VAR) models and Multivariate Time Series analysis - Hybrid models - Quantum Random Forest

I will being with a literature review, provide a theoretical background, outline my methodology and dataset, state dataset statistical information, perform feature engineering, train and measure my models, review the findings, and discuss their implications.

## 2 Literature Review

### 2.1 Overview of Stock Market Prediction

Stock market prediction prior to the 1960s was based on technical or fundamental analysis, both of which are used today. Technical analysis involves analyzing charts of stock prices to look for long- and short-term cycles and patterns. Fundamental analysis is the use of company

and industry data including balance sheets, contracts, and forecasts to try to determine the current and future value of a company. In the 1960s the Efficient Market Hypothesis (EMH) was the most common theory of how market pricing worked. In this, the price of a stock instantly reflected all information that could affect the price with the implication that constant changes in price are largely random and unpredictable. In the 1980s more computing power and advanced mathematical approaches identified subtle patterns within this "randomness" indicating the movements are not entirely random. Behavioral Economics showed that human and group psychology provided one mechanism by which pricing changes could violate the Efficient Market Hypothesis. The development of Autoregressive Integrated Moving Average (ARIMA) models provided the ability to forecast with more quantitative rigor. In the 2000s computing power and algorithm development advanced further leading to machine learning developments including Random Forest, Support Vector Machines, Recurrent Neural Networks, and Long- Short-term memory (LSTM) models the latter of which benefitted from both temporal memory as well as the ability to "forget" weakly interacting data points.

## 2.2 SARIMA and ARIMA Time Series Model Development

There were some foundational research projects in the 1920s that set the stage for the development of Seasonal Autoregressive Integrated Moving Average (SARIMA), a form of ARIMA in 1970 in part by Box and Jenkins (Box et al. 2015). SARIMA adds seasonality to ARIMA models and tries to find the simplest (most parsimonious) model by identifying the stationarity of data, estimate model parameter values, and checking the validity of the model. The concept of stationarity is the measure of whether a series of data have a trend or seasonality. The removal of trend and seasonality was determined to provide a more robust model(Box et al. 2015). One aspect of these time series models that limit their use is that the data must be able to be rendered stationary for the models to be valid.

## 2.3 GARCH Models of Volatility

Generalized AutoRegressive Conditional Heteroskedasticity (GARCH) is an extension to the 1982 Nobel prize winning AutoRegressive Conditional Heteroskedasticity (ARCH) system that rely on the observation that periods of high volatility tend to cluster together in time. This allows ARIMA models to capture risk over the timeframe of the model and can compensate for limitations of the heterskedasticity assumption of an ARIMA model by allowing variance to be dynamic.

## 2.4 Vector Autoregressive (VAR) and Multivariate Time Series Analysis

The addition of Vector Autoregression (VAR) models were developed in the early 1980s to capture the reality of financial markets, that they are influenced by many internal and external factors such as interest rates, unemployment, current volatility, current pricing levels,

and many others. These factors have complex and dynamic influence on each other. VAR models are designed to capture current values and relationships as well as past values and their relationships. This is in contrast to the commonly 1- or 2-dimensional SARIMA models capturing linear dynamics over time.

### 2.5 Quantum Random Forest

Within the scope of my Random Forest (RF) study, traditional Random Forest uses standard compute approaches. Cloud services such as Amazon Braket provide quantum compute and compute simulators that add quantum compute paradigms to the RF and other machine learning algorithms. One of the key capabilities in quantum RF (QRF) is the ability to move Gini impurity index calculation into a higher dimensional space, which may make the data more separable.

## 3 Theoretical Framework

### 3.1 Random Forest Methodology

#### 3.1.1 Decision Tree Introduction

A decision tree is a structure that captures decisions t0 provide the ability to make decisions about data. Decision paths are constructed mathematically during model building that provide guidance through the structure to the bottom of the tree, the final node of which serves as the prediction. Training begins with a top-level node. Data below this node are split in a manner that increases the "purity" of one class under analysis. The process is repeated recursively with the data below each node increasingly partitioned to favor a single class until a stopping point occurs at which the final "leaf" nodes are composed of only a single class (pure) or a stopping criteria is hit. Stopping criteria may be maximum depth or minimum samples in a node.

Mathematically, during recursion we have a dataset D composed of n samples. We perform a greedy (optimized for the current decision not considering future decisions) search over: - All classes in the dataset: $X_1, X_2, ..., X_p$ - The valid range of values for each class

The objective is to maximize "purity" with a single class dominating each branch until purity or stopping criteria are reached.

### 3.1.2 Splitting Criteria

There are different mathematical criteria possible to use to determine splits at each node but the most common in practice is Gini Impurity, which measures how much each child node focuses on a single class versus the same measure for the node's parent. Entropy and misclassification error are alternatives to Gini Impurity.

Gini Impurity measures the degree of specialization of each child node in comparison to its parent. Gini Impurity is likewise the probability of misclassification if we randomly assign class labels based on class distribution at the node. The formula for Gini Impurity is:

$$G = 1 - \sum_{i=1}^{C} p_i^2$$

}

where $G$ is the Gini impurity measure of the node, $C$ is the number of classes in the dataset, and $p_i$ is the proportion of samples in the node that belong to class $i$.

Entropy is the next most widely used splitting criteria measure and is less computationally efficient than Gini. It uses logarithmic operations to compute how to balance the child nodes. Also, because it is logarithmic an additional rule must be set for cases where the algorithm might attempt to perform log(0) at a split. The measure of Entropy is "bits" with values at the node ranging from 0 indicating the class is pure to log_2(n_classes) indicating an even split among classes (highest uncertainty). The formula for Entropy is:

$$H = -\sum_{i=1}^{C} p_i \log_2(p_i)$$

.

where $p_i$ is the proportion of samples in the node that belong to class $i$.

Gini and Entropy frequently result in analagous tree splits (Raileanu and Stoffel 2004). Because of this exploration of the ideal splitting criteria measure is recommended but Gini is frequently used if the dataset or number of classes is large.

### 3.1.3 Ensemble Learning

Decision trees are very sensitive to the subset of data selected for training and a single tree will have high variance. The insight of the phenomenon of the "wisdom of crowds", that a collection of moderate or poor opinions can be averaged to create a predictor superior to that of the individuals in it provides the basis for the ensemble learning approach used by Random Forest. The concept of Bootstrap Aggregation (Bagging) introduced by Breiman (Breiman et al. 1984) provided the algorithm:

1. Create a dataset for each tree composed of a data subset of roughly equal number of samples, with replacement
2. Fit a decision tree on each subset. By definition each tree will likely be different.
3. At prediction time take the majority vote across all trees for classification, the mean prediction across all trees for regression.

If each tree has variance $\sigma^2$ the average variance for $B$ independent predictors is:

$$\text{Var}(\bar{f}) = \frac{\sigma^2}{B}$$

This means with ensemble learning with 1000 trees the average variance is 1/1000th that of an individual tree.

### 3.1.4 Random Forest Algorithm

In Random Forest the data are sampled $B$ times to produce $B$ trees. Within each tree a subset of features are selected and the split calculated according to the splitting criteria measure (Gini or Entropy as above). Each tree is grown down until purity or stopping criteria are reached. Pruning may occur where nodes are eliminated that provide low value to the prediction.

Random Forest Algorithm Pseudocode

```
# Training:
Input: training data (X, y), number of trees B

For t = 1 to B:
    1. Sample n values from the training data.
    2. Train a decision tree using the sample:
        a. At each split:
            - Randomly select a subset of features (m < total features)
            - Determine the best split among these features
              (unless stopping criteria or full node purity is reached)
        b. Repeat recursively for each child node until:
            - Maximum depth, minimum samples, or purity criterion is met
    3. Save the trained tree T_t

Output: Collection of trained trees {Tree_1, Tree_2, ..., Tree_B}

--------------
# Prediction:
Input: A new observation, the tree collection from training phase
```

```
For classification:
  - Each tree outputs a predicted class
  - The most commonly (majority) class is selected as the output

For regression:
  - Each tree outputs a numeric prediction
  - The output is the average of all predictions
```

### 3.1.5 Out-of-Bag Error Estimation

Error estimation for bagging is performed by using out-of-bag (OOB) samples as cross validation instead of holding out a fraction (70-80% commonly) of the sample in a model validation set. OOB estimation works by using samples that were not included in the training of the specific tree to be tested and using those as a validation sample to estimate the model error. The probability that an observation is not selected in a bootstrap sample is $1 - (1 - 1/n)^n$ or approximately 37%. Assuming the dataset to be composed of independent and identically distributed samples this provides a robust and unbiased validation set to measure error rate of each tree. Further, Breiman also provides an algorithm using OOB predictions and applying permutation to a single feature in the OOB samples and measure the change in error rate thereby measuring the importance of each feature in the tree (Breiman et al. 1984)

### 3.1.6 Hyperparameters and Tuning

Random Forest models are trained and tuned by modifying several hyperparameters that modify the computational complexity and the variance/bias tradeoff. The maximum depth of each tree, early stopping criteria, and the number of trees to train provide tuning opportunities to increase or decrease computational complexity. The number of trees is an important factor to tune as the higher the number of trees the more stable the model becomes as the variance decreases for each additional tree trained. The maximum tree depth controls the complexity of each tree with a low value providing limited predictive power while higher values can capture complex non-linear relationships but risk overfitting.

To add to the tuning of bias and variance, maximum features per split (m) sets the number of features selected at each node for splits with lower m potentially creating higher variability between individual trees while higher m allows a tree to focus on the most important features. Further bias versus variance tuning can be performed by tuning the maximum number of samples per leaf and per node with the former establishing if further splits are required and the latter setting the need to split further. As with maximum tree depth and maximum features per split tuning these can improve model generalization and reduce tree and model error.

## 3.2 Time Series Analysis

### 3.2.1 Stationarity and Unit Root Tests

Augmented Dickey-Fuller (ADF):

$$\Delta Y_t = \alpha + \beta t + \gamma Y_{t-1} + \sum_{i=1}^{p} \delta_i \Delta Y_{t-i} + \epsilon_t$$

### 3.2.2 Autocorrelation and Partial Autocorrelation

Autocorrelation function (ACF):

$$\rho(k) = \frac{Cov(X_t, X_{t-k})}{\sqrt{Var(X_t)Var(X_{t-k})}} = \frac{\gamma(k)}{\gamma(0)}$$

### 3.2.3 ARMA Model Structure

General ARMA(p,q) model:

$$Y_t = c + \sum_{i=1}^{p} \phi_i Y_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \epsilon_t$$

where $\phi_i$ are autoregressive coefficients and $\theta_j$ are moving average coefficients

### 3.2.4 Model Selection Criteria

Note that BIC penalizes model complexity.

AIC and BIC information criteria:

$$AIC = 2k - 2\ln(\hat{L})$$

$$BIC = k\ln(n) - 2\ln(\hat{L})$$

where $k$ is the number of parameters, $n$ is sample size, and $\hat{L}$ is maximum likelihood.

### 3.2.5 GARCH for Volatility

GARCH:

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{p} \alpha_i \epsilon_{t-i}^2$$

### 3.2.6 Trend Removal Through Differencing

Higher-order differencing significantly increases model complexity and risks overfitting. After a first differencing ACF and PACF plots are examined and/or ADF test if performed to establish if the data has been made stationary.

First-order differencing:

$$Y_t' = Y_t - Y_{t-1}$$

## 3.3 Random Forest with Time Series Hybrid Model

### 3.3.1 Feature Engineering Strategy

In feature engineering for this model I will join datasets containing stock market indicators and pricing and supplement with time series features. As Random Forest has no memory (no temporal dependencies) I intend to inject these during feature engineering to produce a hybrid model. The time series aspects I intend to add as features depend on identifying important lag values and adding rolling mean and standard deviation to capture local trends, shocks, and variability.

# 4 References


Box, George E. P., Gwilym M. Jenkins, Gregory C. Reinsel, and Greta M. Ljung. 2015. *Time Series Analysis: Forecasting and Control*. 5th ed. Hoboken, NJ: John Wiley & Sons. https://www.researchgate.net/publication/299459188_Time_Series_Analysis_Forecasting_and_Control5th_Edition_by_George_E_P_Box_Gwilym_M_Jenkins_Gregory_C_Reinsel_and_Greta_M_Ljung_2015_Published_by_John_Wiley_and_Sons_Inc_Hoboken_New_Jersey_pp_712_ISBN_.

Breiman, Leo, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. 1984. *Classification and Regression Trees*. Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software.

Raileanu, Laura E., and Kilian Stoffel. 2004. "Theoretical Comparison Between the Gini Index and Information Gain Criteria." *Annals of Mathematics and Artificial Intelligence* 41 (1): 77–93. https://doi.org/10.1023/B:AMAI.0000018580.96245.c6.