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

Integrating Machine Learning and Classical Time Series Modeling

Don Krapohl (Advisor: Dr. Shusen Pu)

2025-11-01

Table of contents

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

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 (Bollerslev 1986). 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 (Sims 1980). 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.

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 σ^2 the average variance for B independent predictors is:

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

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):

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

Partial Autocorrelation Function (PACF):

$$\phi_{kk} = \mathrm{Corr}(y_t - \hat{y}_t(1, \dots, k-1),\ y_{t-k} - \hat{y}_{t-k}(1, \dots, k-1))$$

The Partial Autocorrelation Function (PACF) provides the relationship between the observation and the observation at lag k removing the influence of all shorter lag periods. In viewing the plot the PACF drops off after lag p in the AR(p) process providing an indicator of the order of the AR model.

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 ϕ_i are autoregressive coefficients and θ_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 Methodology for Random Forest with Time Series Hybrid Model

To determine an optimal model and the significant features within it I will need to bring together multiple datasets to build a model that accurately predicts my target variables of market index direction in 5 and 20 days.

Steps:

1. **Collect, combine, and cleanse datasets**: Join pricing and indicator data by trading day
2. **Use time series analysis to diagnose time-based structures**: Run ADF tests, plot ACF/PACF, fit ARCH models with various lags, compare AIC/BIC
3. **Engineer time series features based on diagnostics**: Create lag variables for significant PACF lags, add rolling windows based on ARCH results
4. **Add technical indicators**: Include standard indicators (MACD, RSI, Bollinger Bands) used in the literature
5. **Train Random Forest on augmented feature set**: Build multiple Random Forest models and optimize hyperparameters
6. **Validate with OOB and cross-validation**: Perform model quality measurement
7. **Output feature importance**: Outline the greatest contributors to the model and trim features as appropriate

This approach provides the ability to add some memory of past pricing and effects to the Random Forest model based on extracted patterns instead of using default lags common to indicator algorithm default inputs.

4 Data and Methods

4.1 Data

4.1.1 Data Sources

To ensure transferability of the approaches used in multiple reference papers I am combining two data sources to get comprehensive coverage and generate additional features:

1. **Kaggle "34-year Daily Stock Data"** (Prakash 2024): Provides S&P 500 pricing as well as macroeconomic indicators (VIX, unemployment, interest rates, and geopolitical risk indices)
2. **Yahoo Finance** (Yahoo Finance 2024): Provides OHLC (Open, High, Low, Close) data and volume for the S&P 500

I limit them to an overlapping period January 1990 to February 2024 resulting in a merged dataset of ~9,000 daily observations.

4.1.2 Dataset Incoming Features

The combined dataset from Kaggle and Yahoo result in 19 columns representing the pricing, volume, and macroeconomic features fundamental to establishing pricing patterns. Each row contains a date column indicating the stock trading date and the relevant metrics for that date.

Column Name	Description	Data Source
dt	Date of observation in YYYY-MM-DD format.	Kaggle
vix	VIX (Volatility Index), a measure of expected market volatility.	Kaggle
sp500	S&P 500 index value, a benchmark of the U.S. stock market.	Kaggle
sp500_volume	Daily trading volume for the S&P 500.	Kaggle
djia	Dow Jones Industrial Average (DJIA), another key U.S. market index.	Kaggle
djia_volume	Daily trading volume for the DJIA.	Kaggle
hsi	Hang Seng Index, representing the Hong Kong stock market.	Kaggle
ads	Aruoba-Diebold-Scotti (ADS) Business Conditions Index, reflecting U.S. economic activity.	Kaggle
us3m	U.S. Treasury 3-month bond yield, a short-term interest rate proxy.	Kaggle
joblessness	U.S. unemployment rate, reported as quartiles (1 represents lowest quartile and so on).	Kaggle
epu	Economic Policy Uncertainty Index, quantifying policy-related economic uncertainty.	Kaggle
GPRD	Geopolitical Risk Index (Daily), measuring geopolitical risk levels.	Kaggle
prev_day	Previous day's S&P 500 closing value, added for lag-based time series analysis.	Kaggle
sp500_open	Opening price (USD).	Yahoo
sp500_high	High price for the day.	Yahoo
sp500_low	Low price for the day.	Yahoo
sp500_close	Closing price for the day.	Yahoo
sp500_adj_close	Adjusted closing price (accounting for dividends and splits).	Yahoo
sp500_ohlc_volume	Day trading volume.	Yahoo

#| Table 1: Market and Volume Indicators

Market statistics show 8597 samples from 2007-01-19 to 2024-02-16 and prices ranging from $295.46 to $5029.73 and S&P500 volume ranging from 14,990,000 to 11,456,230,000 shares traded.

Stat	Date	sp500	sp500_volume	djia	djia_volume	hsi	sp500_close	sp500_high	sp500_low	sp500_open	sp500_ohlc_volume
count	8597	8597.00	8.60e+03	8597.00	8597.00	8597.00	8596.00	8596.00	8596.00	8596.00	8.60e+03
mean	2007-01-19	1596.65	2.46e+09	13662.54	183.17	16763.46	1596.11	1605.32	1585.88	1595.90	2.46e+09
min	1990-01-03	295.46	1.50e+07	2365.10	1.59	2736.60	295.46	301.45	294.51	295.45	0.00
50%	2007-01-22	1270.20	2.52e+09	10846.29	177.83	16803.76	1270.09	1277.49	1261.72	1270.04	2.52e+09
max	2024-02-16	5029.73	1.15e+10	38797.90	1022.68	33154.12	5029.73	5048.39	5016.83	5026.83	1.15e+10
std	NaN	1106.24	1.85e+09	9022.86	133.67	7350.10	1105.71	1111.32	1099.28	1105.44	1.85e+09

#| Table 2: Macroeconomic Indicators

Macroeconomic indicators show interesting information with 8597 rows matched to the market indicator dataset. The VIX is a volatility index showing how quickly prices are changing and ranges from 9.14 to 82.69 with 82.69 indicating high volatility. Aruoba-Diebold-Scotti (ADS) Business Conditions Index is a measure of US economic activity and implies the current state of the economy. ADS ranges from -26.42 indicating economic shrinkage and likely recession to a maximum in this dataset of 9.48. The data are indexed roughly to zero showing no economic growth or shrinkage. The US 3 month bond yield ranges from 0% yield to maximum of 8.26% and a mean of 2.69%. Joblessness is unemployment measured in quartiles with lowest quartile being 1, highest 4. EPU is the Economic Policy Undertainty index measuring uncertainty related to US economic policy and ranges from 57.20 to 350.46 in this dataset. GPRD is the Geopolotical Risk Index (Daily) measuring geopolitical risk and ranges from 9.49 to 1045.60 in this dataset with a mean of 109.43.

Stat	vix	ads	us3m	joblessness	epu	GPRD	prev_day
count	8597.00	8597.00	8597.00	8597.00	8597.00	8597.00	8597.00
mean	19.56	-0.16	2.69	2.49	115.56	109.44	1596.11
min	9.14	-26.42	0.00	1.00	57.20	9.49	295.46
50%	17.73	-0.05	2.30	2.00	106.12	96.60	1270.09
max	82.69	9.48	8.26	4.00	350.46	1045.60	5029.73
std	7.90	1.65	2.30	1.12	41.58	64.57	1105.71

4.1.3 Dataset Calculated Indicators

The dataset columns "dt" indicating the trading date and "sp500_close" were used with the pandas_ta library to generate technical indicator values commonly used in the literature review reference papers and cited as common practice (Murphy 1999). The default values were used for all indicator inputs such as moving average type for Middle Bollinger Bands (typically 20-day simple moving average).

Column Name	Description	Data Source
1d_return	One-day absolute return of the S&P 500.	Derived
macd	Moving Average Convergence Divergence (EMA12 − EMA26).	Derived
macd_signal	Signal line for MACD, typically a 9-day EMA of MACD.	Derived
roc	Rate of Change indicator showing percentage price change over a set period.	Derived
rsi	Relative Strength Index, measures recent price strength and momentum.	Derived
stoch_k	Stochastic oscillator %K, compares closing price to recent high-low range.	Derived
stoch_d	Stochastic oscillator %D, a moving average of %K.	Derived
adx	Average Directional Index, measures the strength of a trend.	Derived
obv	On-Balance Volume, cumulative measure of volume flow with price movement.	Derived
atr	Average True Range, measures market volatility based on recent price ranges.	Derived
bb_upper	Upper Bollinger Band, indicating upper volatility threshold.	Derived
bb_middle	Middle Bollinger Band, usually a 20-day simple moving average.	Derived
bb_lower	Lower Bollinger Band, indicating lower volatility threshold.	Derived
ema_12	12-day Exponential Moving Average.	Derived
ema_26	26-day Exponential Moving Average.	Derived
sma_20	20-day Simple Moving Average.	Derived
sma_50	50-day Simple Moving Average.	Derived
sma_200	200-day Simple Moving Average.	Derived

4.1.4 Dataset Correlation Matrix

The clustering of highly correlated features as indicated in the correlation matrix typically show closely related time periods such as previous and next day returns. This is expected as large jumps in a market-wide index is rare. Likewise rolling mean and standard deviations are correlated to price changes over equivalent periods (5 day rolling mean is moderately correlated to the price changes over that period).

The indicator features roc, rsi, and the stochastics show correlation with lagged values at lags 1, 2, and 11. The high correlation of rolling means to these indicators implies these features may be incorporated into those derivative indicators (roc, rsi, and stochastic d and k). This is further supported by the high correlation among these same features.

Within macroeconomic series the VIX (volatility index) is correlated to rolling mean and standard deviations–large rolling standard deviation implies volatility occurred during that period. The EPU and joblessness are moderately correlated with rolling standard deviations implying those indices are likely significant in market-level pricing.

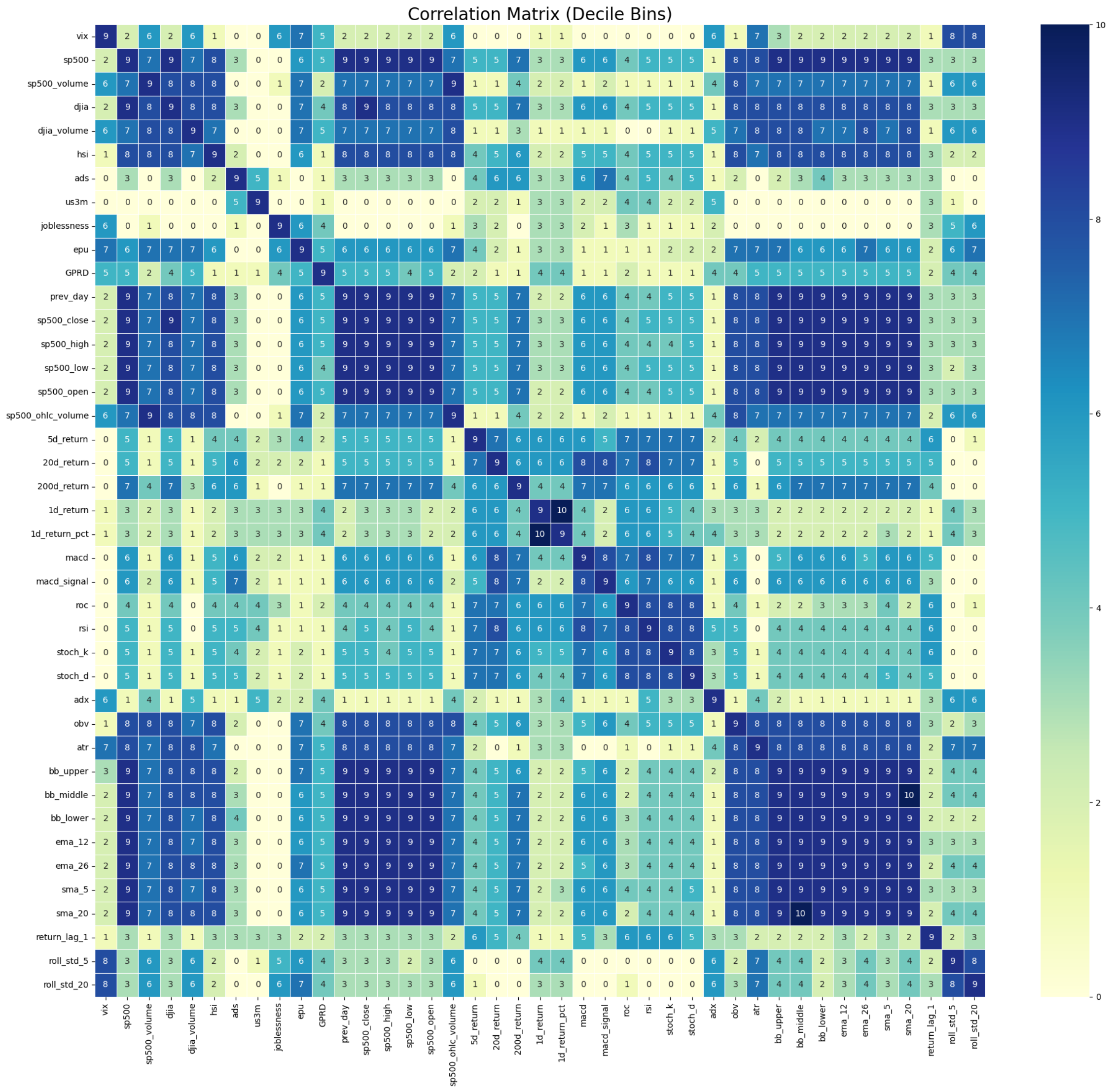


Figure 1: Correlation Matrix

4.1.5 Missing Data Check and Correct

The dataset after supplementary features were added for rolling averages and technical indicators had rows at the beginning of the dataset with empty values. This is because calculating a 20-day rolling average will not have a value until the 20th day from the beginning of the dataset as it requires 19 prior days to calculate the average with that day. Likewise the 5-day rolling average would not appear until day 5. In all the added features 54 rows contained one or more empty columns. For the purposes of this study the beginning dates in the financial time series were unimportant so no imputation was done, the 54 rows were dropped.

4.1.6 Class Imbalance Check and Correct

The incoming data contained 8797 samples. The feature created for each row containing the 5-day future up/down signal had 4921 rows indicating the price would be up in 5 days, 3622 indicating the price would be down. After using Synthetic Minority Oversampling Technique (SMOTE) to balance the classes resulting in 4921 rows for each class.

In SMOTE a random sample in the minority class is selected along with a set of its nearest neighbors of the same class. A new minority class sample is generated from a random point between the selected sample and one or more of the neighbors. This repeats until the number of samples in the minority class matches that of the majority class.

4.2 Feature Engineering

One valuable aspect of Random Forest is that scaling is not required and the algorithm is not sensitive to orders-of-magnitude differences in the input variables. Because time series does require feature scaling all inputs to the Ordinary Least Squarse model produced here to estimate optimal lag must be scaled if they are not within the same range and distribution. In this study we are using only the S&P500 closing price for lag determination so scaling will not be required.

4.2.1 Time Series

4.2.1.1 Time series model {sec-methods-diagnostics-ts-model}

Testing for stationarity with the Augmented Dickey-Fuller (ADF) indicates a first differencing of the S&P 500 closing prices to be stationary with ADF of -15.96 at $p<0.001$. This allows model building with no further differencing. The first determination of optimal lag is to check the ACF and PACF plots using the squared residuals from a basic Ordinary Least Squares (OLS) model.

Note

ACF/PACF of OLS Model Squared Residuals

```
# fit basic regression (OLS)
```

```
X = sm.add_constant(np.ones(len(y_diff)))
ols_model = sm.OLS(y_diff.values, X).fit()
squared_resid = ols_model.resid**2  # need squared residuals for ACF/PACF

# plot acf
plot_acf(squared_resid, lags=40, ax=plt.gca())

# plot pacf
plot_pacf(squared_resid, lags=40, ax=plt.gca())
```

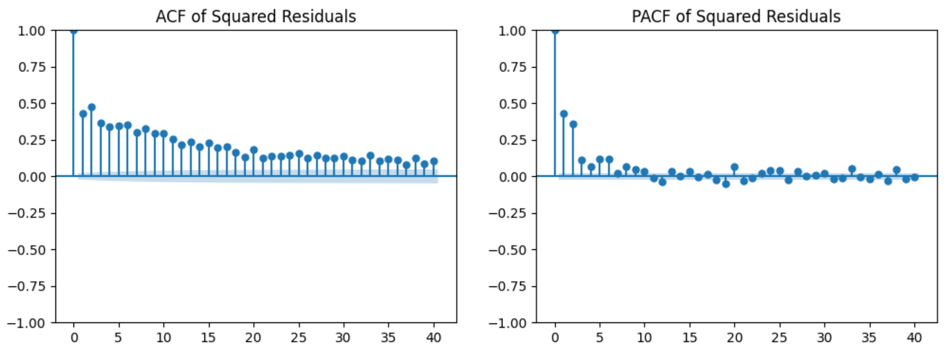


Figure 2: ACF/PACF

The PACF shows a strong spike at lag 1 indicating that as potentially significant with smaller spikes at lags 2 and 3. The ACF shows slow decay with a stationary financial time series indicating financial market volatility clustering. This supports the use of rolling mean and standard deviation to smooth but capture volatility patterns.

Using lag of 1 and verifying through AIC and BIC measures with maximum 20-day lag period the model scores continued to improve up to the maximum (20 day) lag. This indicates an ARCH(1) model may not be sufficient and that a GARCH model may be necessary to capture lag and volatility. This is consistent with the ACF interpretation of volatility clustering.

Despite the poor ARCH(1) model results I generated features for the 1-day absolute and percent return to provide the temporal features to test their viability in the ultimate Random Forest model.

4.2.2 Dataset Engineered Time Series Features

Returns at lag periods identified through time series analysis were added with the AIC and BIC indicating a potentially significant lag of 1. Rolling mean (through 5- and 20-day simple moving averages) and standard deviation were added to capture and smooth periodic trending in the closing prices and supplement the lagged returns with GARCH-aligned volatility metrics

Column Name	Description	Data Source
return_lag_1	One-day lagged return value.	Derived
return_lag_2	Two-day lagged return value.	Derived
return_lag_3	Three-day lagged return value.	Derived
roll_std_5	5-day rolling standard deviation of returns or prices.	Derived
roll_std_20	20-day rolling standard deviation of returns or prices.	Derived

4.2.3 Dataset Engineered Target Variables {#sec-data-dataset-engineered-targets

The 1, 5, and 20 day direction of the closing price was used to generate potential target variables for prediction. These direction features were not included as inputs to the model but reserved for individual Random Forest models designed specifically to predict for that future period. The creation of models at each future duration also provided the ability to compare the OOB score for each to determine the most useful model.

Column Name	Description	Data Source
direction_1d	Direction of 1-day return (1 = up, 0 = down).	Derived
direction_5d	Direction of 5-day cumulative return (1 = up, 0 = down).	Derived
direction_20d	Direction of 20-day cumulative return (1 = up, 0 = down).	Derived

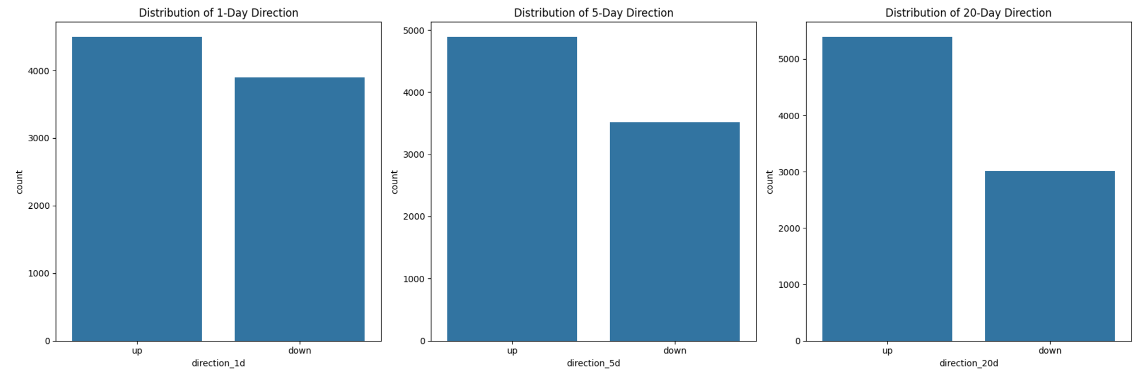


Figure 3: Distribution of Price Direction

4.3 Feature Reduction

4.3.1 Dimensionality Reduction

This dataset began with ~60 features across multiple data sources. Many of the features are either correlated or derived from related algorithms. To reduce the features space we can study how features vary together, which features contribute generally to the variance of the dataset, and which specifically provide the most variance to a general random forest model. After the features are selected we may then proceed with training a final model.

4.3.1.1 Multicollinearity

The measure of how features are correlated in a regression model is measured using the Variance Inflation Factor (VIF). I use it here to provide insight into columns which may be removed as the information they contain are embedded in other predictor variables. For each factor X_i we measure the coefficient of determination (R^2) against the selected feature using the VIF formula:

$$\text{VIF}_i = \frac{1}{1 - R_i^2}$$

If there is no multicollinearity for the factor VIF=1. The more factors or the larger the R^2 with other factors the larger the VIF. By convention a moderate correlation of 5 is borderline acceptable and 10 indicates high multicollinearity. For my analysis I removed all factors with VIF>10.

4.3.1.2 Principal Component Analysis

In addition to removal of columns using VIF I further reduced columns using Principal Component Analysis (PCA). The purpose of PCA is to reduce the dimensionality of a dataset by capturing the maximum variance available in a dataset. The steps to perform PCA are to standardize the data to make the mean zero, calculate a covariance matrix, find the eigenvalues and eigenvectors of the data, sort the discovered eigenvalues and eigenvectors, and calculate the principal components. Steps:

1. Scale data by standard scaling
$$Z_j = \frac{X_j - \bar{X}_j}{s_j}$$
where $\bar{X}_j$ is the mean and s_j is the standard deviation of variable X_j.
2. Perform Singular Value Decomposition

As an alternative to the calculation of eigenvalues and eigenvectors we can use Singular Value Decomposition (SVD) to derive the principal components. SVD

Note

SVD in Python

```
U, S, Vt = np.linalg.svd(X_centered, full_matrices=False)

# for each sample
explained_variance = S**2 / (X_centered.shape[0] - 1)  # sample variance divided by number
explained_variance_ratio = explained_variance/np.sum(explained_variance)  # explained varia
```

3. Determine Variance Contribution for Each Component

Using the Explained Variance Ratio (EVR) formula we divide each eigenvalue by the sum of all eigenvalues to determine how much each factor contributes to the variance. Sorted descending we are then able to select the most significant factors that have the highest explanatory value in the data. EVR can be used to determine which features to preserve in the model with a common cutoff being those features that cumulatively explain 90-95% of the variance. The EVR formula:

$$\text{EVR}_i = \frac{\lambda_i}{\sum_{j=1}^{p} \lambda_j}$$

where: λ_i is the eigenvalue for the i^{th} principal component
p is the total number of components

4.4 Model Training and Evaluation

4.4.1 Model Evaluation Criteria Selection

To optimize use of the data and to leverage the strength of Random Forest Out-Of-Bag (OOB) validation I elected to forego the standard 80/20 train/test split and use OOB validation in the model. As noted previously, OOB validation holds out a random subset of the data each round of training and uses this set of data as validation. This provides unseen data while not making this set a fixed series for use by every tree model trained.

4.4.2 Classic Random Forest Model Training

Pseudocode for random forest model training is above. The process, assuming 100 trees (n_estimators=100), begins with drawing roughly 67% of the samples randomly as training examples. The remaining 33% is used for testing. The samples are random and all rows in dataset are candidates for each tree built. This generates 100 trees from slighly different sets of data. Each tree is built by choosing a subset of features from all features. In my optimal model the number of features at each node split is $log_2(p)$. My final dataset after PCA is composed of 8 features so each node will select $log_2(8) = 3$ 3 of these 8 features. This ensures variety in the choices to be made at any node in the model. The algorithm tests multiple thresholds for each of the features and measures the information gain. In my model I used Gini Impurity. The data are divided into two groups matching the best split threshold for each feature. Each subgroup becomes a node and the same process continues until a stopping criteria is hit. These can include the maximum tree depth, fewer than the set minimum samples per split, each leaf node has fewer than minimum samples per leaf, or all samples in the node are from the same class.

4.4.3 Random Forest Model Prediction

The algorithm harnesses the "wisdom of crowds" phenomenon in which the opinion of a single individual may be low quality but the combination of a group of individuals can be of higher quality than the individuals. Here as well, a single tree at decision time accepts the input features and provides a class prediction. The remaining trees do likewise and the final prediction is the class predicted most frequently with the probability of that class being correct the probability of the output class being selected across all trees.

4.4.4 Hyperparameter Tuning

To try to discover globally optimal sets of hyperparameters I used a random search algorithm. I selected random search over grid search as grid search performs an exhaustive search over a range of values set in a search space that is sensitive to the search space values and computationally intensive. In contrast random search selects values randomly within the range of the hyperparameter values set in the search space making it more computationally efficient to explore a larger range of hyperparameter values.

4.4.5 OOB Score and Best Model

The best model based on out-of-bag score was a model using 100 estimators, maximum tree depth of 18 levels, minimum samples per leaf of 3, minimum samples per split of 4, and log^2 max features.

4.5 Model quality

4.5.1 Model Quality Metrics

To measure the quality of the model we will determine how the predictions are able to correctly characterize future market direction we will measure precision, recall, and F1 score. Generally the higher each of these scores the better the model.

Precision is the measure of the number of correctly predicted positive cases. Precision is calculated:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Where TP is the true positive rate and FP is the false positive rate. In the selected random forest model the precision is 74.7% representing a model that predicts market direction substantially better than random guessing.

Recall (sensitivity, true positive rate) is the measure of how many actual positives are accurately identified. It is calculated:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Where TP is the true positive rate and FN is the false negative rate. The recall for my selected random forest model is 71.6% which also represents the model to be ability of the model to discover true positives.

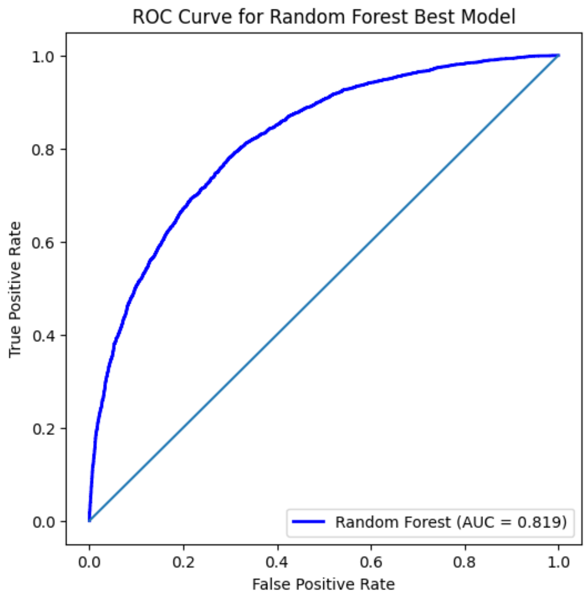
A measure of the balanced view of precision and recall is F1 score. It is the harmonic mean of precision and recall and is calculated as:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The F1 score for the highest OOB-scoring random forest model is 73.1% indicating a well-balanced model that has a fairly high discernment capability between the up and down classes in the dataset.

4.5.2 ROC AUC

The Receiver Operating Characteristics curve captures the balance of the true and false positive rates and provides a visual demonstration of the model predictive power vs random chance. The 45 degree diagonal line represents predictions made randomly over a dataset and represents 50% odds of correctly guessing the sample class. The curved line represents the tradeoff between sensitivity (recall rate or the number of true positives correctly identified) and specificity (the number of true negatives correctly identified). A perfect ROC curve touches the top left corner and indicates a problematic model that is may indicate overfitting. A ROC curve that matches the diagonal line represents a model that is no better than randomly guessing the class label.



This ROC curve demonstrates a good balance of sensitivity and specificity and is in line with similar curves in the litarature.

The area under the ROC curve, commonly called AUC, is the measure of the model to discriminate between classes and serves as a better measure of model quality than accuracy as it measure quality across all classes and thresholds. A generalized form of AUC is the “trapezoidal rule” for using ROC to establish AUC.

$$\text{AUC} = \sum_{i=1}^{n-1} (FPR_{i+1} - FPR_i) \frac{TPR_{i+1} + TPR_i}{2}$$

Where true positive rate is as above and false positive rate is:

$$\text{FPR} = \frac{FP}{FP + TN}$$

The AUC for this random forest model is 0.819 or 81.9% indicating a fairly high probability of being to discern between classes.

4.5.3 Confusion Matrix

The confusion matrix provides the ability to characterize in which direction a model is most/least able to discern or if it is easily confused between classes.

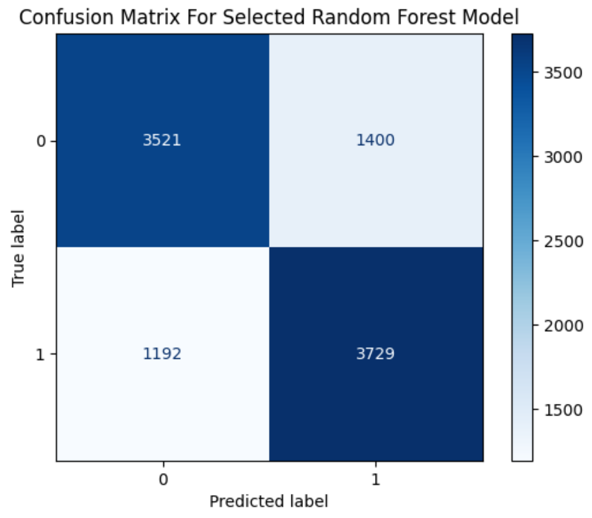


Figure 4: Confusion Matrix

The confusion matrix for this model indicates that there are 3521 true negatives (TN) where the class label was correctly predicted to be “price moves down”. There are 1400 false positives (FP) where the model predicted the future price to be up but it was actually down. There are 1192 false negatives (FN) where the model predicted the market would be down but it is actually up. Finally there are 3729 instances where the model correctly predicted the future market price at 5 days in the future would be up.

5 Interpretation and Conclusion

I initially trained multiple test random forest models to establish baseline model quality for predicting the S&P 500 5-day future price direction (up/down). Those models achieved just over chance at 51-53% OOB score. Adding rolling and lag features discovered through time series analysis did not markedly improve the model and these were eliminated during feature pruning with PCA and manual feature removal based on multicollinearity studies. The final model trained over the pruned set of features achieved 73.1% OOB score indicating the model had improved. This model also yielded a 73.1 F1 score and AUC of 81.9 indicating the model demonstrates reasonably high quality.

The model determined the most important features of the dataset to be the Aruoba-Diebold-Scotti (ADS) Business Conditions Index reflecting U.S. economic activity, the momentum indicator "Rate of Change" (ROC), Relative Strength Index (RSI) capturing the speed and momentum of recent price changes, Average Direction Index (ADX) capturing up/down trend, the volume of the Dow Jones Index component stocks (DJIA Volume), the daily Economic Policy Uncertainty Index (EPU), the VIX volatility indicator, and Hang Seng Index (HSI) representing the Hong Kong stock market levels. Within these we have macroeconomic measures (EPU, ADS), comparative price movements with other indexes (HSI and DJIA volume), measures of trend and trend strength (ROC, ADX), and volatility (VIX).

These findings suggest that the rolling period features were redundant, as their information was already captured by other macroeconomic indicators. This is implied by PCA not removing them and the model indicating high value to these while muticollinearity study showed some of these to be moderately or highly correlated to other features.

The final model is usable for predicting the 5-day direction of the S&P 500 stock market index. Separate manual backtesting was done using a long-only approach wherein a purchase is made when the prediction first says the 5-day price will be up and sell when the prediction first says the price will drop. Not accounting for cost of the trade this model resulted in returns significantly greater than a simple buy-and-hold and could potentially provide one source for the high-risk allocation of an investment portfolio.

6 References

Bollerslev, Tim. 1986. "Generalized Autoregressive Conditional Heteroskedasticity." *Journal of Econometrics* 31 (3): 307–27. https://doi.org/10.1016/0304-4076(86)90063-1.

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.

Murphy, John J. 1999. *Technical Analysis of the Financial Markets*. New York: New York Institute of Finance.

Prakash, Shivesh. 2024. "34-Year Daily Stock Data." Kaggle. https://www.kaggle.com/datasets/shiveshprakash/34-year-daily-stock-data.

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.

Sims, Christopher A. 1980. "Macroeconomics and Reality." *Econometrica* 48 (1): 1–48. https://doi.org/10.2307/1912017.

Yahoo Finance. 2024. "Historical Market Data." https://finance.yahoo.com.