S&P500 Stock Index Visual Analysis and Prediction

Bahram Khanlarov

2023-04-29

```
#install.packages(c("quantmod", "forecast", "tseries", "rugarch", "prophet", "tsfknn", "knitr))
```

```
#libraries
#install.packages("quantmod")
#library(ggplot2)
#library(forecast)
#library(tseries)
#library(rugarch)
#library(prophet)
#library(tsfknn)
#library("quantmod")
#library(knitr)
```

1.Introduction

The S&P 500[1], a stock market index that tracks the performance of 500 large-cap publicly traded companies in the United States. These companies are chosen based on factors such as their market capitalization, liquidity, and industry sector. The S&P 500 is considered a benchmark for the overall health of the US stock market and is widely used to indicate economic performance. In this report,we use retrieved S&P 500 from 2010 until 2023 and explore initially with Moving Average Convergence Divergence and do projection with 3 different methods and finally select the best performing model.

2.Methods

Initially data visually explored with MACD. On the latter stage we applied ARIMA, Prophet and Knn, to forecast the data for the next 30 days. Lastly, the effectiveness of these models evaluated and best model based on RMSE was selected.

2.1 Data collection

The getSymbols() function from the quantmod package is used to download the data for the S&P 500 index ("^GSPC") from Yahoo Finance (`src="yahoo"`) for the date range from January 1st, 2013 to january 1st, 2023.

```
# Load required libraries
library(quantmod)
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: TTR
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(forecast)

# Set start and end dates for data retrieval
start_date <- "2013-01-01"
end_date <- "2023-01-01"

# Retrieve historical stock data for S&P 500 index from Yahoo Finance
getSymbols("^GSPC", src = "yahoo", from = start_date, to = end_date)
```

```
## [1] "GSPC"
```

```
head(GSPC)
```

```
##            GSPC.Open GSPC.High GSPC.Low GSPC.Close GSPC.Volume GSPC.Adjusted
## 2013-01-02   1426.19   1462.43  1426.19    1462.42  4202600000       1462.42
## 2013-01-03   1462.42   1465.47  1455.53    1459.37  3829730000       1459.37
## 2013-01-04   1459.37   1467.94  1458.99    1466.47  3424290000       1466.47
## 2013-01-07   1466.47   1466.47  1456.62    1461.89  3304970000       1461.89
## 2013-01-08   1461.89   1461.89  1451.64    1457.15  3601600000       1457.15
## 2013-01-09   1457.15   1464.73  1457.15    1461.02  3674390000       1461.02
```

The data is organized in a tabular format with columns representing different attributes for each date:

GSPC.Open: The opening price of the S&P 500 on that date. GSPC.High: The highest price reached by the S&P 500 during that trading session. GSPC.Low: The lowest price reached by the S&P 500 during that trading session. GSPC.Close: The closing price of the S&P 500 on that date. GSPC.Volume: The trading volume of the S&P 500 on that date (the total number of shares traded). GSPC.Adjusted: The adjusted closing price of the S&P 500 on that date. The adjusted closing price accounts for factors such as stock splits and dividends to provide a more accurate representation of the stock's performance.

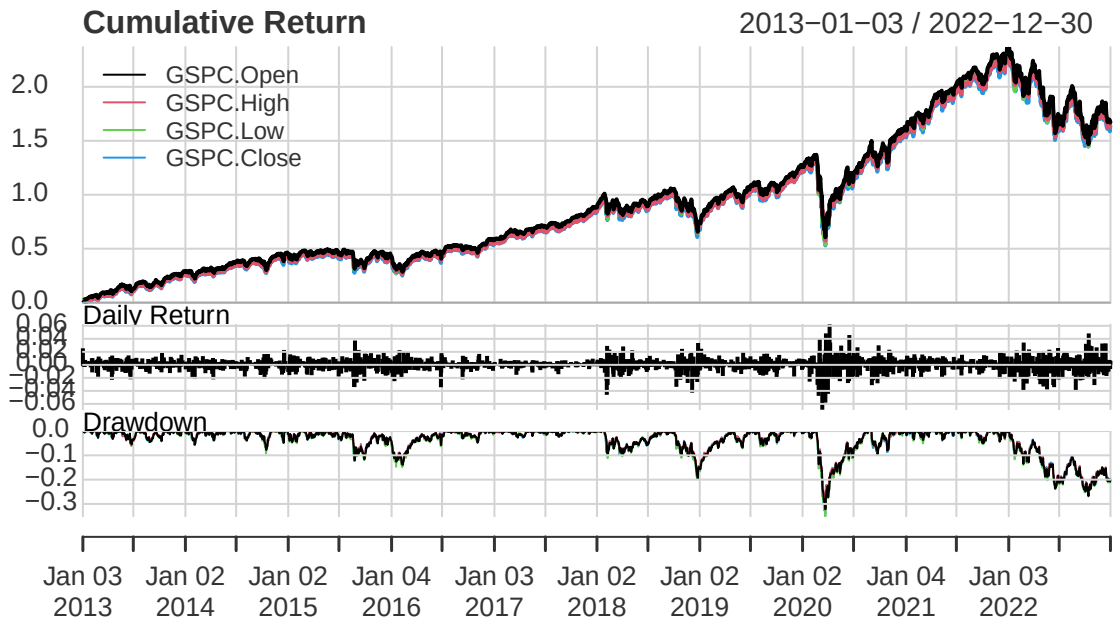
2.2 Descriptive Analysis:

```
plot(GSPC$GSPC.Close,main=paste("Closing prices of GSPC"))
```


```
#install.packages("PerformanceAnalytics")

# Load the PerformanceAnalytics package
library(PerformanceAnalytics)
```

```
##
## Attaching package: 'PerformanceAnalytics'
## The following object is masked from 'package:graphics':
##
##     legend
```

```
charts.PerformanceSummary(ROC(GSPC[, 1:4], n = 1, type = "discrete"), main = "GSPC stock evolution")
```



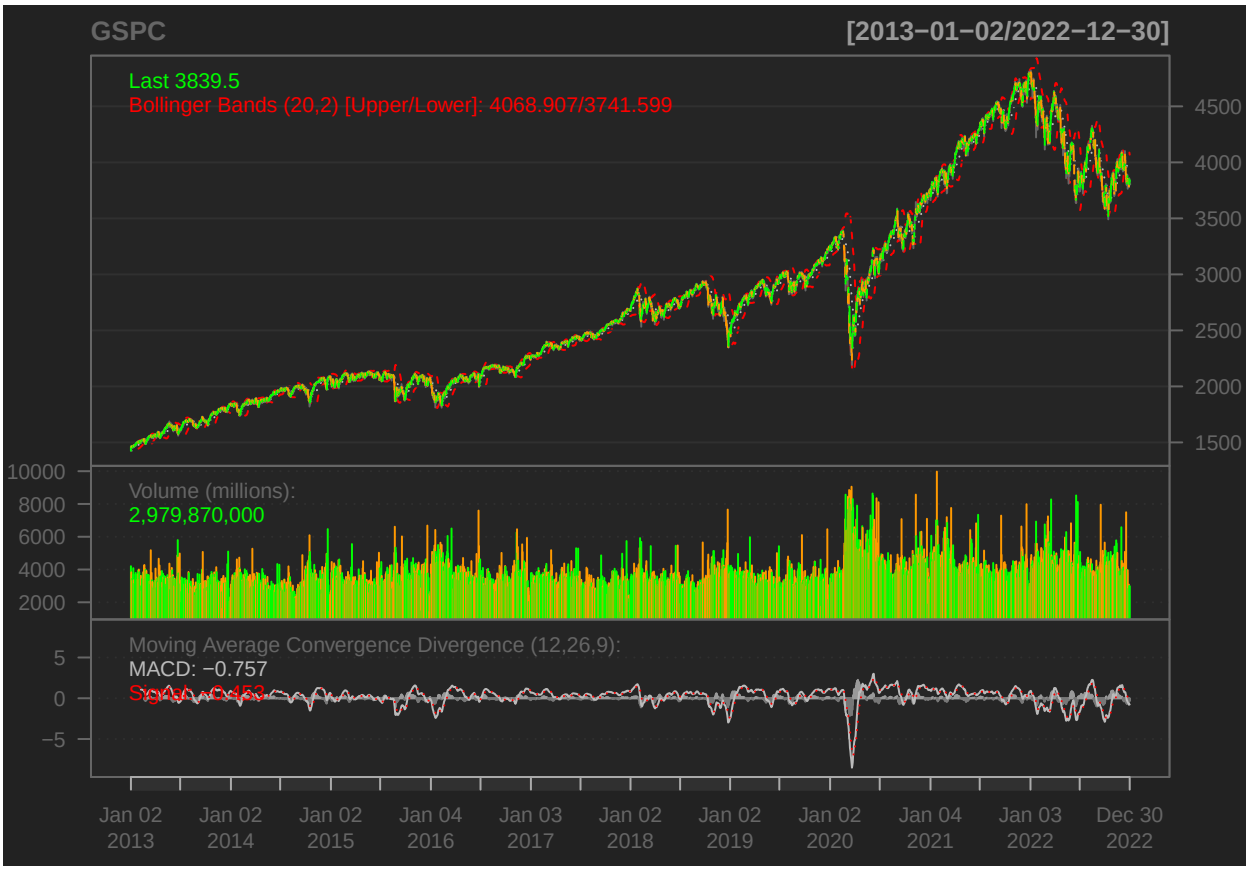
MACD Visualization

We used chartSeries() function from the quantmod package to generate financial charts.To provide a more comprehensive view, the chart integrates three technical indicators: volume (addVo()) [4], Bollinger Bands (addBBands()) [5], and Moving Average Convergence Divergence (MACD) (addMACD()) [6].

Plot below shows Bollinger Bands, which are a technical analysis instrument that comprises a moving average (the middle band) and two standard deviation lines (the upper and lower bands) plotted above and below the moving average [4]. The upper and lower bands represent the range in which the price is anticipated to fluctuate. In our present context, the upper Bollinger Band is positioned at 4068.907 and the lower Bollinger Band is at 3741.599 [3]. Consequently, it is anticipated that the price movements of the index will typically oscillate within this defined range. Any price movements exceeding these boundaries may indicate a potential shift in trend or a continuation of the prevailing trend. By monitoring these Bollinger Bands, investors can gain valuable information about the potential volatility and direction of the market.

The MACD (Moving Average Convergence Divergence) is a momentum indicator that follows trends and displays the relationship between two EMAs (exponential moving average) of an asset's price [4]. It is calculated by subtracting the 26-period EMA from the 12-period EMA [4]. The formula for this is MACD = 12-Period EMA - 26-Period EMA. An EMA gives more weight to recent data points and reacts more significantly to recent price changes than an SMA, whic2h assigns equal weight to all observations in the period [4]. The MACD line is the result of this calculation. A nine-day EMA of the MACD, known as the "signal line", is plotted on top of the MACD line and can act as a trigger for buy and sell signals. Traders may buy when the MACD crosses above its signal line and sell or short when it crosses below [4]. There are several ways to interpret MACD indicators, with crossovers, divergences, and rapid rises/falls being among the most common.

```
chartSeries(GSPC,TA=c(addVo(),addBBands(),addMACD()))
```



The MACD value of -0.757 and signal value of -0.453 that mentioned indicate that the MACD line (grey line) is below the signal line (red one), which is typically interpreted as a bearish signal. In this scenario, the 12-period EMA is positioned below the 26-period EMA, indicating a potential bearish trend. When the MACD line is below the signal line, it can be interpreted as a bearish signal, suggesting that it may be a good time to sell or short the security. While these findings suggest caution and discourage immediate buying, it's essential to remember that MACD is just one among many technical indicators. To make well-informed trading decisions, it is advisable to consider other technical analysis tools and market information, allowing for a comprehensive assessment of the situation.

2.3 Prediction Models

2.3.1 Arima : Autoregressive Integrating Moving Average

The ARIMA (Autoregressive Integrated Moving Average) model is a widely used method for forecasting time series data by leveraging past values to predict future ones. By combining the predictive power of ARIMA with the trend-following and momentum indicators of MACD (Moving Average Convergence Divergence), traders can enhance their decision-making process and gain a more comprehensive understanding of the market.

The ARIMA model consists of three fundamental techniques: auto-regression, differencing, and moving average. The "p" value in the model corresponds to auto-regression, capturing the dependence of the current value on past values. Differencing, denoted by the "d" value, involves removing trends from the time series data to convert non-stationary data into stationary data. Lastly, the "q" value represents the number of lagged values of the error term in the moving average component of the model.

We test for stationarity using the Augmented Dickey-Fuller unit root test. The p-value resulting from the ADF test has to be less than 0.05 or 5% for a time series to be stationary. If the p-value is greater than 0.05 or 5%, we conclude that the time series has a unit root which means that it is a non-stationary process.

In our specific case, the p-value obtained from the ADF test exceeds the threshold, indicating that the time series is non-stationary.

Based on the p-value of 0.3586, we cannot reject the null hypothesis that a unit root is present in the time series sample. This suggests that the GSPC. Close series is non-stationary. We apply diff(log()) to make data stationary now.

```
# Perform the Augmented Dickey-Fuller test on the GSPC.Close column
library(tseries)
adf.test(GSPC$GSPC.Close, alternative = "stationary")
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  GSPC$GSPC.Close
## Dickey-Fuller = -2.5191, Lag order = 13, p-value = 0.3586
## alternative hypothesis: stationary
```

```
GSPC_diff <- diff(GSPC$GSPC.Close)
GSPC_log_diff <- log(GSPC_diff)
```

```
## Warning in log(GSPC_diff): NaNs produced
```

```
# checking Nan and infinite values
GSPC_log_diff <- na.omit(GSPC_log_diff)
any(is.na(GSPC_log_diff))
```

```
## [1] FALSE
```

```
any(is.infinite(GSPC_log_diff))
```

```
## [1] TRUE
```

```
# Remove the infinite values
GSPC_log_diff <- GSPC_log_diff[is.finite(GSPC_log_diff)]
adf.test(GSPC_log_diff, alternative = "stationary")
```

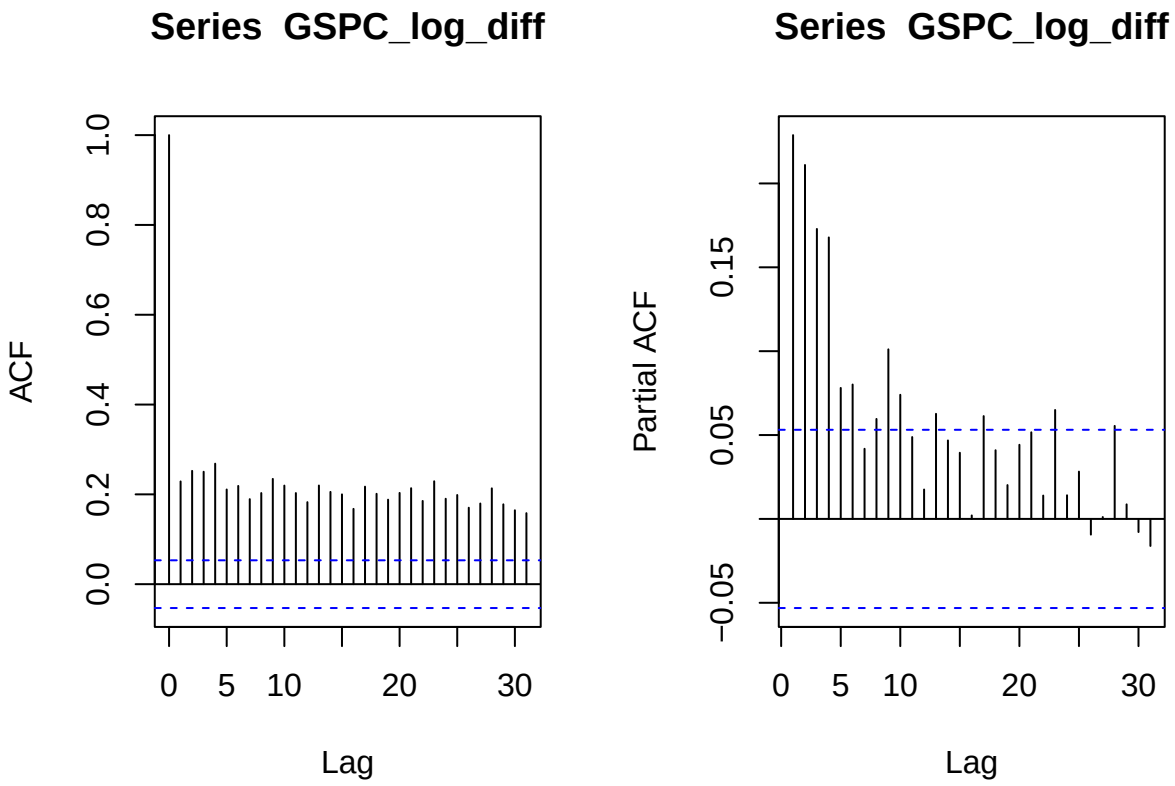
```
## Warning in adf.test(GSPC_log_diff, alternative = "stationary"): p-value smaller
## than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  GSPC_log_diff
## Dickey-Fuller = -7.9136, Lag order = 11, p-value = 0.01
## alternative hypothesis: stationary
```

Before applying the auto.arima function we check how our data looks with ACF and PACF plots. In general, if the ACF plot shows a slow decay and the PACF plot shows a sharp cutoff after a certain lag, it suggests that an Autoregressive (AR) model may be appropriate for the data. If the ACF plot shows a sharp cutoff and the PACF plot shows a slow decay, it suggests that a Moving Average (MA) model may be appropriate. Here a sharp drop in autocorrelation after lag 0 is an indication of stationarity. A stationary time series is one whose properties (mean, variance, autocorrelation structure) do not change over time.A rapid decline in the ACF values after the first lag suggests there's no strong trend or seasonality in the time series. If there were a trend, the ACF would show a more gradual decline. If there were seasonality, there would be spikes at regular intervals corresponding to the seasonality period.A sharp drop-off or truncation in the PACF after a few lags typically indicates an autoregressive (AR) process.

```
## Plot ACF and PACF
```

```r
par(mfrow = c(1, 2))
acf(GSPC_log_diff)
pacf(GSPC_log_diff)
```




```r
par(mfrow = c(1, 1))
```

The auto.arima function, which is part of the forecast package, serves as a powerful tool for automatically selecting the most suitable ARIMA model for our dataset. By employing a stepwise algorithm, this function systematically explores various combinations of AR (Autoregressive), I (Integrated), and MA (Moving Average) terms to identify the optimal model configuration.

```r
## Applying auto.arima() to the dataset
library(forecast)
modelfit <-auto.arima(GSPC_log_diff)
summary(modelfit)
```

```
## Series: GSPC_log_diff
## ARIMA(0,1,1)
##
## Coefficients:
##           ma1
##       -0.9506
## s.e.   0.0105
##
## sigma^2 = 1.43:  log likelihood = -2173.65
## AIC=4351.3   AICc=4351.3   BIC=4361.73
##
## Training set error measures:
##                       ME    RMSE       MAE  MPE MAPE      MASE         ACF1
## Training set 0.03507478 1.19495 0.9021362 -Inf  Inf 0.7448716 -0.002380324
```

The notation ARIMA(p,d,q) is used to describe the characteristics of an ARIMA model, where:

p is the order of the Autoregressive (AR) term. d is the degree of differencing. q is the order of the Moving Average (MA) term. Given ARIMA(0,1,1):

p=0: This indicates that there is no autoregressive term in the model. In other words, the model does not use any of the previous period's values (lags) to predict the current value.

d=1: The series has been differenced once to make it stationary. Differencing a series involves subtracting the current value from the previous value. If a series is differenced once and becomes stationary,d is set to 1. If it needs to be differenced multiple times, d would be the number of times the differencing is performed.

q=1: This means that the model uses one lagged forecast error in a moving average formula to make future predictions. The forecast error is the difference between the actual value and the predicted value from a forecasting model.

In simpler terms, an ARIMA(0,1,1) model suggests that:

The series has been made stationary by differencing it once. The model uses the immediate previous forecast error (from the last period) to predict the future value,without considering any actual previous values of the series itself.

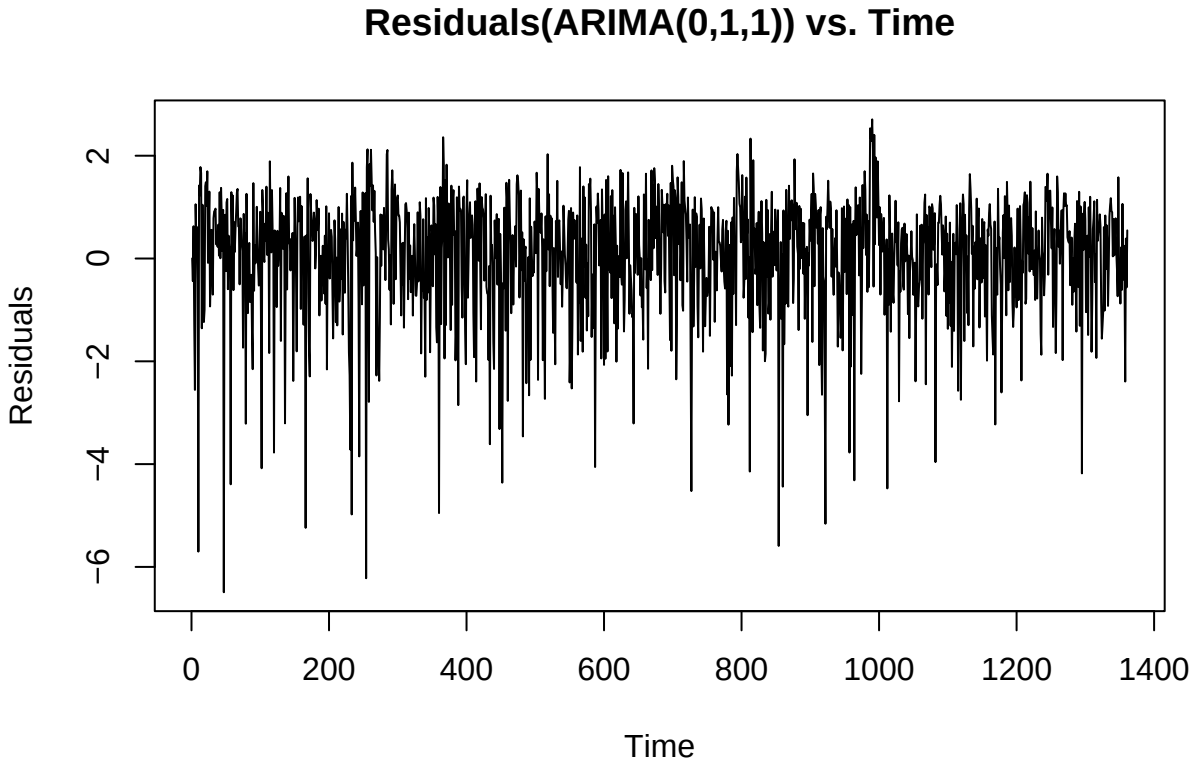
```
# Load required library
library(tseries)

# Perform Augmented Dickey-Fuller test on residuals of fitted model
adf.test(residuals(modelfit))
```

```
## Warning in adf.test(residuals(modelfit)): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  residuals(modelfit)
## Dickey-Fuller = -11.032, Lag order = 11, p-value = 0.01
## alternative hypothesis: stationary
```

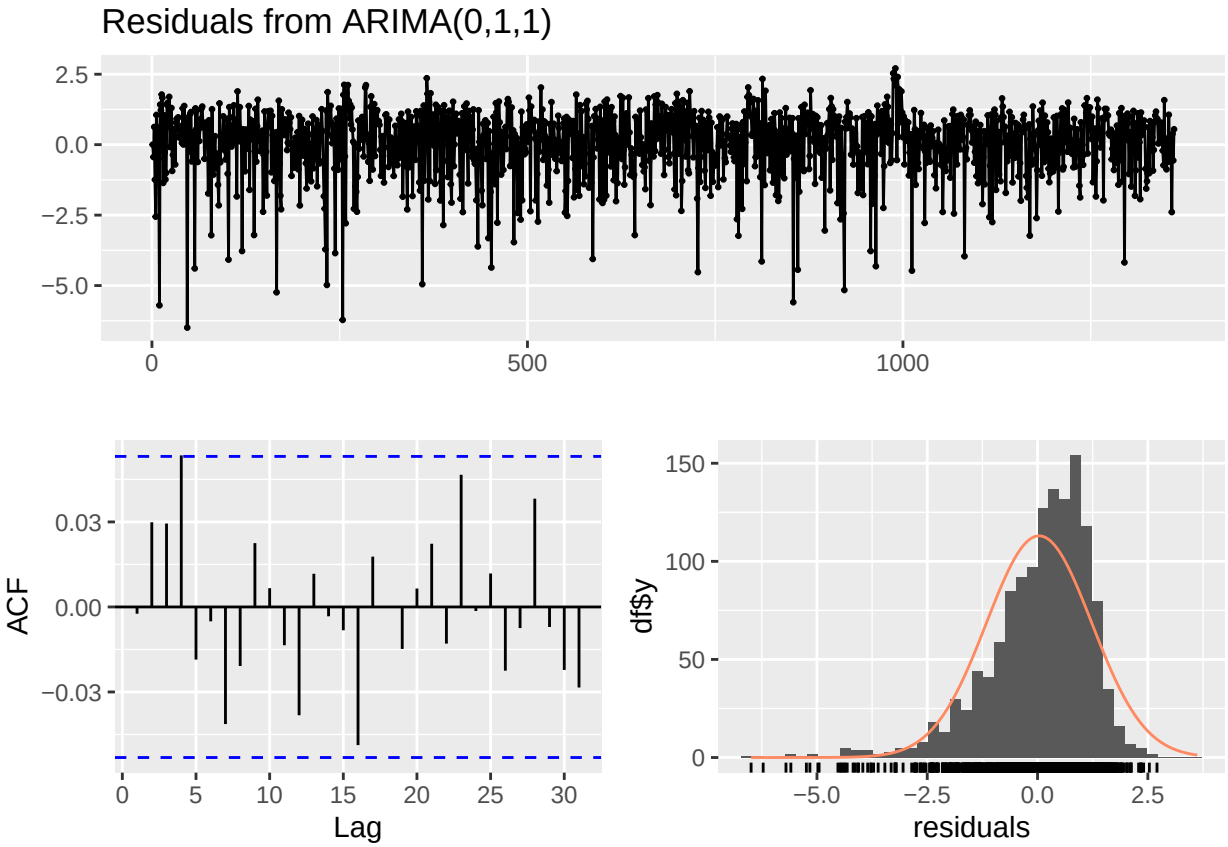
The plot of the residuals can give valuable insights into how well the model fits the data. Ideally, the residuals should be randomly distributed around zero with no obvious patterns or correlations. If the residuals display any patterns or connections, it indicates that the model is not capturing all of the information in the data and that another model may be more suitable.

```
# Diagnostics on Residuals
plot(resid(modelfit),ylab="Residuals",main="Residuals(ARIMA(0,1,1)) vs. Time")
```



From the residual plot we observe residuals randomly scattered around zero, histogram also shows the residuals are following a normal distribution with a mean of zero with slightly left skewed and the ACF plot doesn't show significant autocorrelation beyond the expected range of randomness. We also did Ljung-Box test to assess the presence of autocorrelation in the residuals of TS model. With obtained the p-value is 0.31, which is greater than the significance level of 0.05 we could state, there is no significant evidence to suggest the presence of autocorrelation in the residuals. It indicates that the ARIMA(0,1,1) model has adequately captured the temporal patterns in the data.

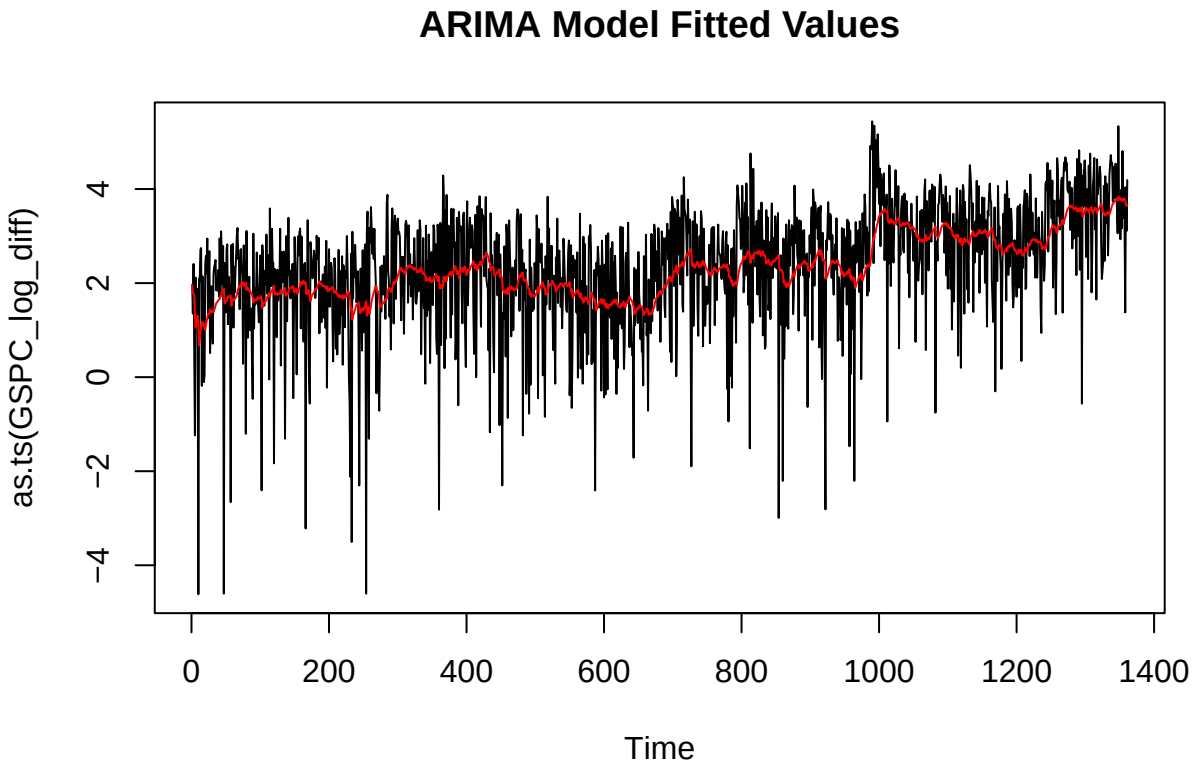
```
library(forecast)
checkresiduals(modelfit)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,1)
## Q* = 10.517, df = 9, p-value = 0.3103
##
## Model df: 1.   Total lags used: 10
```

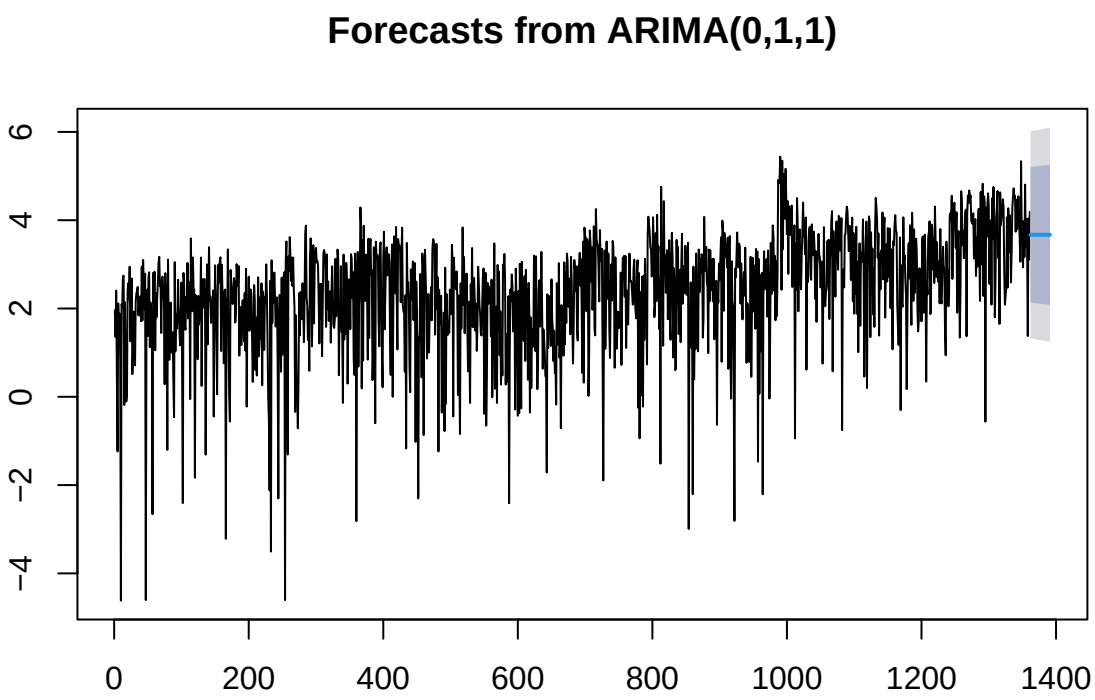
Having our new ARIMA model applied and analyzed we can plot the model prediction in a red line over the real train set stock close price.

```
# Plot the original data
plot(as.ts(GSPC_log_diff), type = "l", col = "black", main = "ARIMA Model Fitted Values")

# Add the fitted values in red
lines(fitted(modelfit), col = "red")
```



After fitting the model, we can forecast future daily close prices. The focus is on predicting the close stock price for the next 30 days or an average month. We can visualize this forecast by plotting the data.

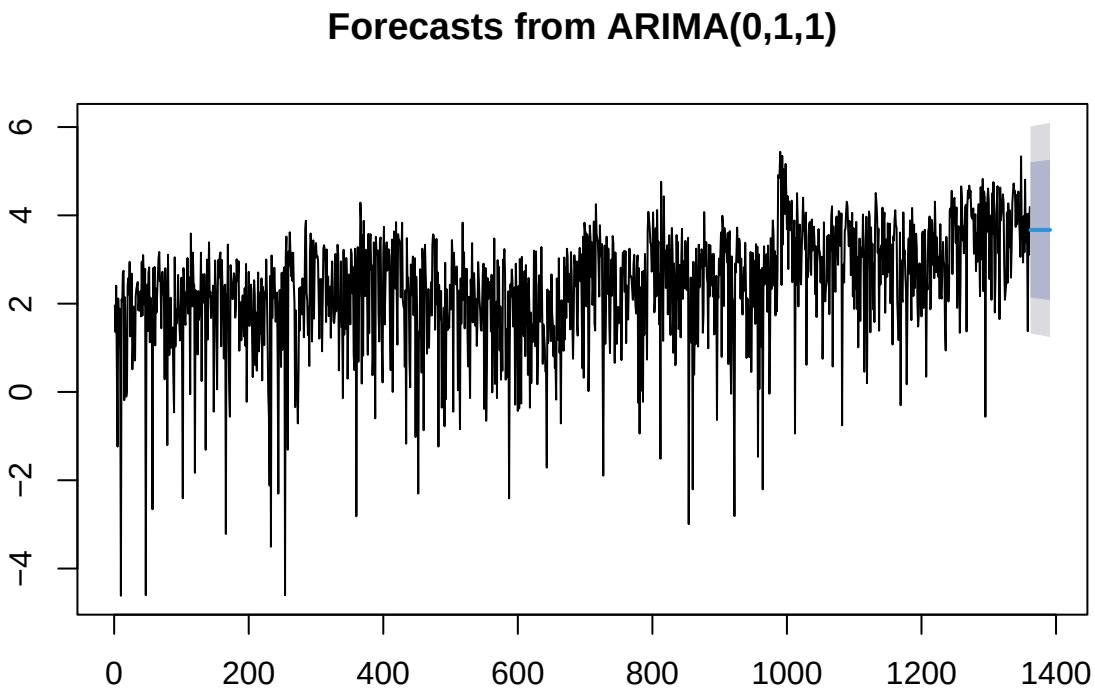
```
plot(forecast(modelfit,h=30))
```



As we can see, we have a blue line that represents the mean of our prediction.

```
# we get the first six forecasted values for the mean of the time series.
price_forecast <- forecast(modelfit,h=30)
```

```r
plot(price_forecast)
```




```r
head(price_forecast$mean)
```

```
## Time Series:
## Start = 1362
## End = 1367
## Frequency = 1
## [1] 3.670212 3.670212 3.670212 3.670212 3.670212 3.670212
```

```r
head(price_forecast)
```

```
## $method
## [1] "ARIMA(0,1,1)"
##
## $model
## Series: GSPC_log_diff
## ARIMA(0,1,1)
##
## Coefficients:
##           ma1
##       -0.9506
## s.e.   0.0105
##
## sigma^2 = 1.43:  log likelihood = -2173.65
## AIC=4351.3   AICc=4351.3   BIC=4361.73
##
## $level
## [1] 80 95
##
## $mean
## Time Series:
## Start = 1362
```

```
## End = 1391
## Frequency = 1
##  [1] 3.670212 3.670212 3.670212 3.670212 3.670212 3.670212 3.670212 3.670212
##  [9] 3.670212 3.670212 3.670212 3.670212 3.670212 3.670212 3.670212 3.670212
## [17] 3.670212 3.670212 3.670212 3.670212 3.670212 3.670212 3.670212 3.670212
## [25] 3.670212 3.670212 3.670212 3.670212 3.670212 3.670212
##
## $lower
## Time Series:
## Start = 1362
## End = 1391
## Frequency = 1
##           80%      95%
## 1362 2.137697 1.326432
## 1363 2.135829 1.323575
## 1364 2.133963 1.320722
## 1365 2.132100 1.317872
## 1366 2.130239 1.315026
## 1367 2.128380 1.312183
## 1368 2.126523 1.309344
## 1369 2.124669 1.306508
## 1370 2.122817 1.303675
## 1371 2.120967 1.300846
## 1372 2.119119 1.298020
## 1373 2.117274 1.295197
## 1374 2.115430 1.292378
## 1375 2.113589 1.289562
## 1376 2.111750 1.286750
## 1377 2.109913 1.283941
## 1378 2.108079 1.281135
## 1379 2.106246 1.278332
## 1380 2.104416 1.275533
## 1381 2.102588 1.272737
## 1382 2.100762 1.269944
## 1383 2.098938 1.267155
## 1384 2.097116 1.264369
## 1385 2.095296 1.261586
## 1386 2.093478 1.258806
## 1387 2.091663 1.256029
## 1388 2.089849 1.253256
## 1389 2.088038 1.250485
## 1390 2.086229 1.247718
## 1391 2.084421 1.244954
##
## $upper
## Time Series:
## Start = 1362
## End = 1391
## Frequency = 1
##           80%      95%
## 1362 5.202728 6.013993
## 1363 5.204596 6.016850
## 1364 5.206462 6.019703
## 1365 5.208325 6.022553
```

```
## 1366 5.210186 6.025399
## 1367 5.212045 6.028242
## 1368 5.213902 6.031081
## 1369 5.215756 6.033917
## 1370 5.217608 6.036750
## 1371 5.219458 6.039579
## 1372 5.221306 6.042405
## 1373 5.223151 6.045228
## 1374 5.224995 6.048047
## 1375 5.226836 6.050863
## 1376 5.228675 6.053675
## 1377 5.230512 6.056484
## 1378 5.232346 6.059290
## 1379 5.234179 6.062093
## 1380 5.236009 6.064892
## 1381 5.237837 6.067688
## 1382 5.239663 6.070481
## 1383 5.241487 6.073270
## 1384 5.243309 6.076056
## 1385 5.245129 6.078839
## 1386 5.246947 6.081619
## 1387 5.248762 6.084396
## 1388 5.250576 6.087169
## 1389 5.252387 6.089940
## 1390 5.254196 6.092707
## 1391 5.256004 6.095471
```

With the blue line explained we can see a darker and light darker areas, representing 80% and 95% confidence intervals respectively in lower and upper scenarios.

Our lower scenario:

```
head(price_forecast$lower)
```

```
## Time Series:
## Start = 1362
## End = 1367
## Frequency = 1
##           80%      95%
## 1362 2.137697 1.326432
## 1363 2.135829 1.323575
## 1364 2.133963 1.320722
## 1365 2.132100 1.317872
## 1366 2.130239 1.315026
## 1367 2.128380 1.312183
```

```
head(price_forecast$upper)
```

```
## Time Series:
## Start = 1362
## End = 1367
## Frequency = 1
##           80%      95%
## 1362 5.202728 6.013993
## 1363 5.204596 6.016850
## 1364 5.206462 6.019703
```

```
## 1365 5.208325 6.022553
## 1366 5.210186 6.025399
## 1367 5.212045 6.028242
```

Finalizing our ARIMA model we do a quick test and train set approach dividing the close price data. We select our train set as the 80 % of our dataset. The test set y the 20% of the dataset.

```
# Dividing the data into train & test sets , Applying the model
N = length (GSPC_log_diff)
n = 0.8*N
train = GSPC_log_diff[1:n, ]
test = GSPC_log_diff[(n+1):N,]
trainarimafit <- auto.arima(GSPC_log_diff)
summary(trainarimafit)
```

```
## Series: GSPC_log_diff
## ARIMA(0,1,1)
##
## Coefficients:
##           ma1
##       -0.9506
## s.e.   0.0105
##
## sigma^2 = 1.43:  log likelihood = -2173.65
## AIC=4351.3   AICc=4351.3   BIC=4361.73
##
## Training set error measures:
##                       ME    RMSE       MAE  MPE MAPE      MASE         ACF1
## Training set 0.03507478 1.19495 0.9021362 -Inf  Inf 0.7448716 -0.002380324
```

```
predlen= length(test)
trainarima_fit <- forecast(trainarimafit, h= predlen)
```

2.3.2 Prophet Prediction

The fable.prophet package, developed by Facebook and introduced by S.J. Taylor and Letham in 2015[7], is a robust forecasting model. Initially designed for forecasting daily data with weekly and yearly seasonality, along with holiday effects, this model has since been expanded to accommodate other types of seasonal data. Its performance shines when applied to time series that exhibit significant seasonality and possess multiple seasons of historical data.

New df has two columns: ds and y. The ds column is assigned the index of the GSPC object, which represents the dates of the time series data. The y column is assigned the fourth column of the GSPC object,GSPC.Close which is first coerced to a numeric vector using the as.numeric function.

```
head(GSPC)
```

```
##            GSPC.Open GSPC.High GSPC.Low GSPC.Close GSPC.Volume GSPC.Adjusted
## 2013-01-02   1426.19   1462.43  1426.19    1462.42  4202600000       1462.42
## 2013-01-03   1462.42   1465.47  1455.53    1459.37  3829730000       1459.37
## 2013-01-04   1459.37   1467.94  1458.99    1466.47  3424290000       1466.47
## 2013-01-07   1466.47   1466.47  1456.62    1461.89  3304970000       1461.89
## 2013-01-08   1461.89   1461.89  1451.64    1457.15  3601600000       1457.15
## 2013-01-09   1457.15   1464.73  1457.15    1461.02  3674390000       1461.02
```

```
df <- data.frame(ds = index(GSPC),
                 y = as.numeric(GSPC[,4]))
head(df)
```

```
##           ds       y
## 1 2013-01-02 1462.42
## 2 2013-01-03 1459.37
## 3 2013-01-04 1466.47
## 4 2013-01-07 1461.89
## 5 2013-01-08 1457.15
## 6 2013-01-09 1461.02
```

```
tail(df)
```

```
##              ds       y
## 2513 2022-12-22 3822.39
## 2514 2022-12-23 3844.82
## 2515 2022-12-27 3829.25
## 2516 2022-12-28 3783.22
## 2517 2022-12-29 3849.28
## 2518 2022-12-30 3839.50
```

We are using the prophet function from the Prophet package to fit a model to our data stored in the GSPC object.We are creating a future dataframe with 30 periods using the make_future_dataframe function and then making predictions using the predict function.

```
#Prophet Forecasting
#Loading time series forecasting prophet package
#install.packages("prophet")
library(prophet)
```

```
## Loading required package: Rcpp
```

```
## Loading required package: rlang
```

```
prophet_pred = prophet(df)
```

```
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
```

```
future = make_future_dataframe(prophet_pred,periods=30)
fcastprophet = predict(prophet_pred,future)
```

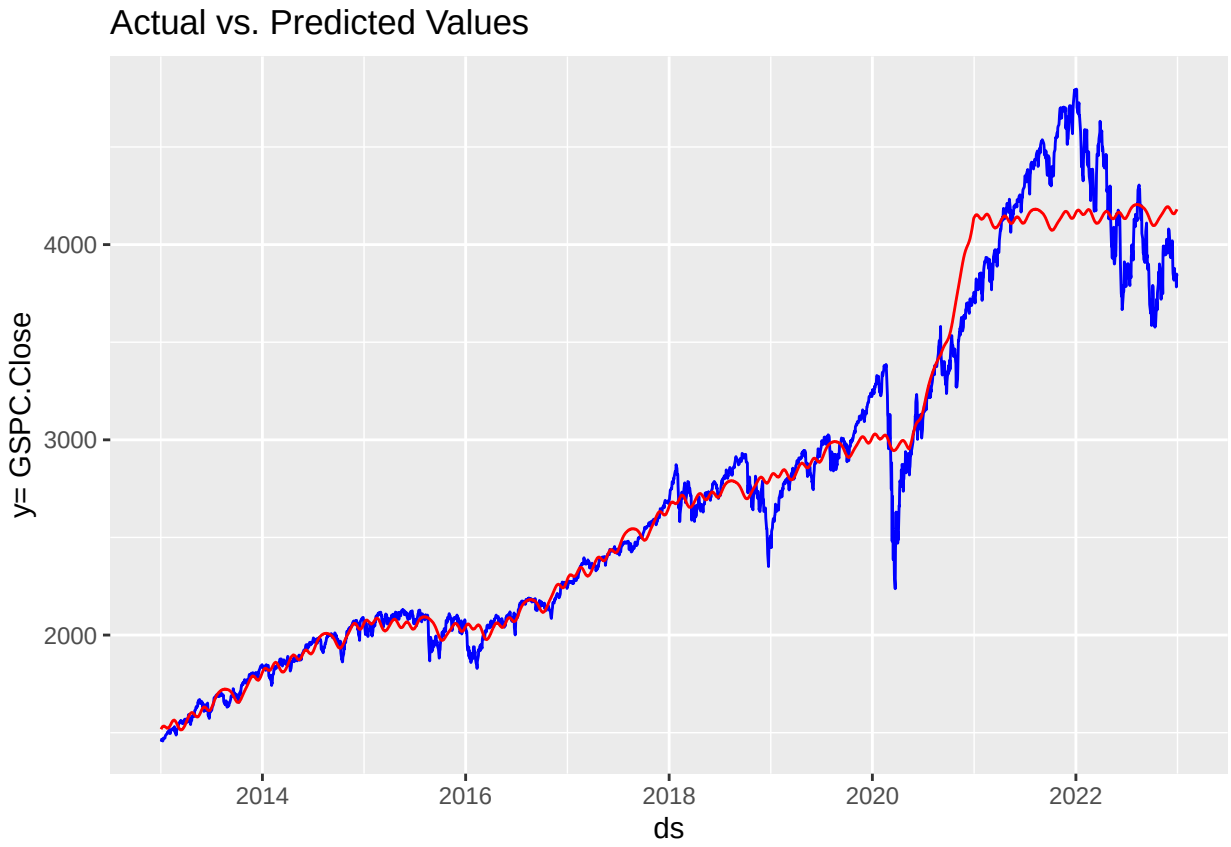
We are creating a new data frame dataprediction with the forecasted values from the fcastprophet object. We are then selecting only the rows of dataprediction that correspond to the length of our training data GSPC$GSPC.Close.

```
#Creating train prediction dataset to compare real data
dataprediction = data.frame(fcastprophet$ds,fcastprophet$yhat)
trainlen = length(GSPC$GSPC.Close)
dataprediction = dataprediction[c(1:trainlen),]
```

```
library(ggplot2)

dataprediction = data.frame(fcastprophet$ds,fcastprophet$yhat)
trainlen = length(GSPC$GSPC.Close)
dataprediction = dataprediction[c(1:trainlen),]

# Plot the actual and predicted values
ggplot() +
  geom_line(aes(x = dataprediction$fcastprophet.ds, y = GSPC$GSPC.Close), color = "blue") +
  geom_line(aes(x = dataprediction$fcastprophet.ds, y = dataprediction$fcastprophet.yhat), color = "red
```

```
  xlab("ds") +
  ylab("y= GSPC.Close") +
  ggtitle("Actual vs. Predicted Values")
```

```
## Don't know how to automatically pick scale for object of type <xts/zoo>.
## Defaulting to continuous.
```



After applying the model and plotting the forecast, we evaluate the model's performance. We use the accuracy function to compare the actual values with the predicted values from the training set. The proper way to do this in Prophet is to perform a cross-validation and analyze the performance metrics of the model.

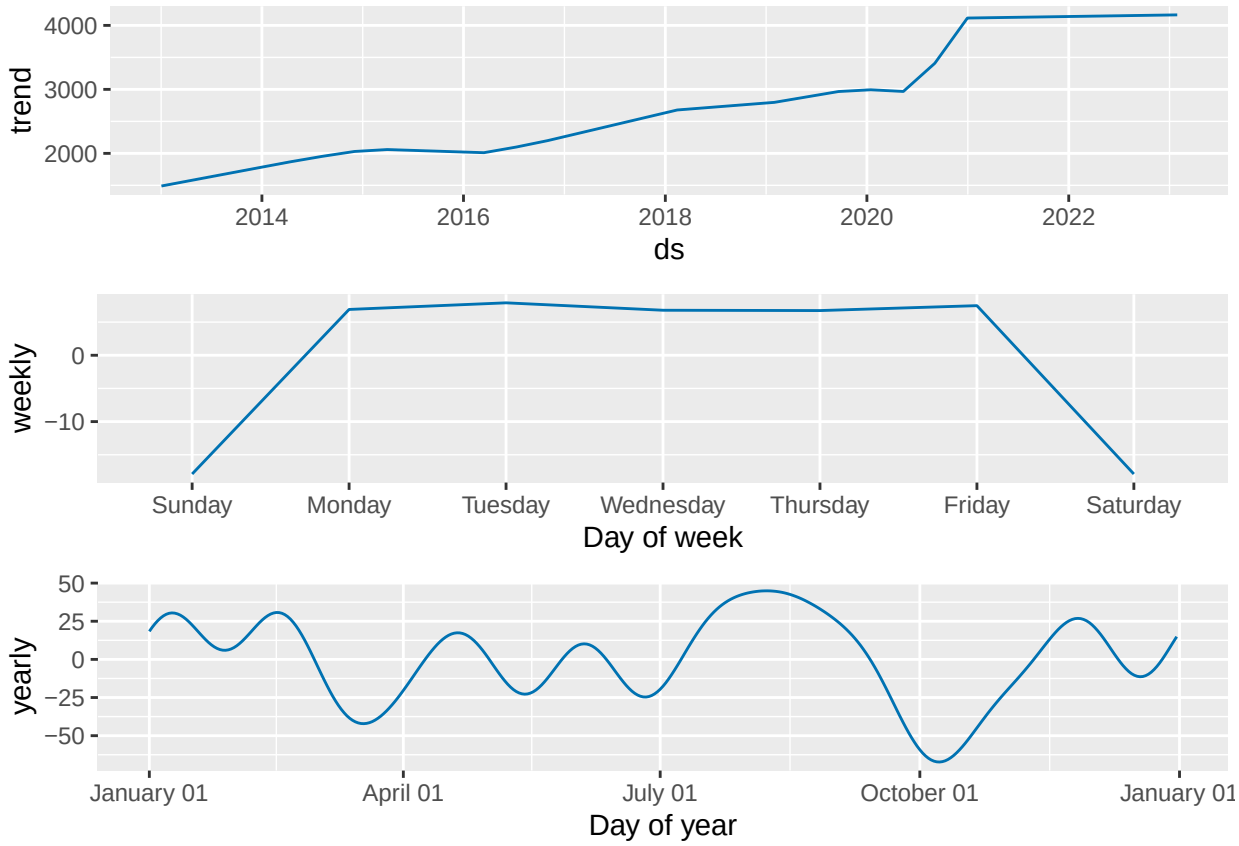
```
#Creating Cross Validation
accuracy(dataprediction$fcastprophet.yhat,df$y)
```

```
##                      ME    RMSE      MAE        MPE     MAPE
## Test set -0.008365584 165.909 105.6513 -0.2400918 3.335202
```

These results show various measures of forecast accuracy for the test set. Lower values for these measures generally indicate better forecast accuracy.

To better understand the data, we can plot the components of the Prophet model, which include a trend component, weekly seasonality, and yearly seasonality.

```
prophet_plot_components(prophet_pred,fcastprophet)
```



2.3.3 KNN Regression Prediction

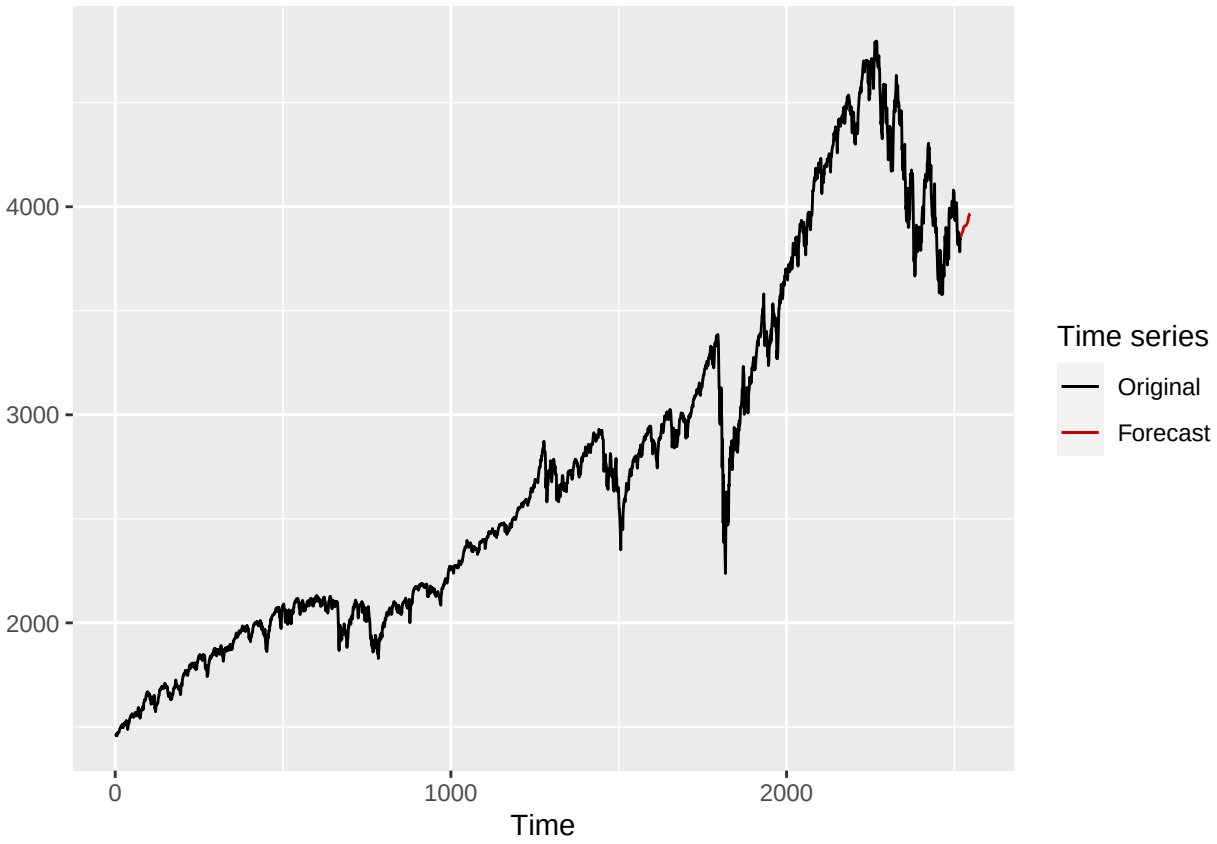
The KNN algorithm is a non-parametric method that predicts the value of a new data point by considering the values of its k nearest neighbors.The features used for prediction were the lagged values of the closing prices. The model was then evaluated based on its ability to accurately predict the future close prices.By comparing the predicted values with the ground truth values, we can observe the model's ability to capture the underlying patterns and trends in the data.

```
#Dataframe creation and model application
#install.packages("tsfknn")
library(tsfknn)
df <- data.frame(ds = index(GSPC),
                 y = as.numeric(GSPC[,4]))

predknn <- knn_forecasting(df$y, h = 30, lags = 1:30, k = 50, msas = "MIMO")
```

```
#Train set model accuracy
ro <- rolling_origin(predknn)
print(ro$global_accu)
```

```
##       RMSE        MAE       MAPE
## 121.575241 104.614426   2.702166
```

```
autoplot(predknn)
```



After applying the KNN forecasting model, the rolling_origin() function is used to evaluate the model's accuracy on the training set. The rolling_origin() function splits the dataset into training and testing subsets and performs rolling origin cross-validation.

3.Results

Based on the RMSE values, the ARIMA(0,1,1) model with 1.19 RMSE was chosen as the best model to forecast future stock values. Overall, the analysis and forecasting of the S&P 500 index provided valuable insights into its historical behavior and future trends. However, it's important to note that stock market prediction is a challenging task, and the results should be interpreted with caution. Additional analysis, consideration of other factors, and regular updates to the model are recommended for more accurate and robust predictions.

References

[1]Kumbure, M. M., Lohrmann, C., Luukka, P., & Porras, J. (2022). Machine learning techniques and data for stock market forecasting: A literature review. Expert Systems with Applications, 197, 116659. https://doi.org/10.1016/j.eswa.2022.116659

[2]Farimani, S. A., Jahan, M. V., & Fard, A. M. (2022). From Text Representation to Financial Market Prediction: A Literature Review. Information, 13(10), 466. https://doi.org/10.3390/info13100466

[3] SP Global. (n.d.). S&P 500® Index Brochure. Retrieved from https://www.spglobal.com/spdji/en/documents/additional-material/sp-500-brochure.pdf

[4] Bollinger, J. (2011). Bollinger Bands. Retrieved from https://www.bollingerbands.com/

[5] Investopedia. (n.d.). Moving Average Convergence Divergence - MACD. Retrieved from https://www.investopedia.com/terms/m/macd.asp

[6] StockCharts.com. (n.d.). MACD (Moving Average Convergence Divergence). Retrieved from https://school.stockcharts.com/doku.php?id=technical_indicators:moving_average_convergence_divergence_macd

[7] Sean J. Taylor & Benjamin Letham (2018) Forecasting at Scale, The American Statistician, 72:1, 37-45, DOI: 10.1080/00031305.2017.1380080