Model Evaluation Notebook

```r
# Installing required packages
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
```

```r
library(forecast)
library(tseries)
library(ggplot2)
library(tinytex)
```

```r
# Importing data
data <- readRDS("stock_data_clean.rds")
```

```r
# Finding negative and nan values
sum(data <= 0)
```

```
## [1] 1991
```

```r
# Shift the entire series to make all values positive
shift_value <- abs(min(data)) + 1
stock_shifted = data + shift_value

# Apply the log-plus-one transformation to the shifted data
stock_log = log1p(stock_shifted)
```

```r
# Log transform to stabilize variance
stock_log_data <- log(stock_log)
stock_diff <- diff(stock_log_data)
```

```r
# Fit the ARIMA model. Here, I am using auto.arima to select the best parameters.
fit <- auto.arima(stock_log, seasonal=FALSE)
summary(fit)
```

```
## Series: stock_log
## ARIMA(5,0,1) with non-zero mean
##
## Coefficients:
##           ar1      ar2      ar3     ar4     ar5     ma1    mean
##       -0.7412  -0.0450  -0.0168  0.0339  0.0647  0.7098  0.7877
## s.e.   0.0758   0.0193   0.0192  0.0191  0.0155  0.0745  0.0001
##
## sigma^2 = 8.392e-05:  log likelihood = 13838.84
## AIC=-27661.68   AICc=-27661.65   BIC=-27610.89
##
## Training set error measures:
##                         ME        RMSE         MAE         MPE      MAPE
## Training set -2.456401e-06 0.009153381 0.006368894 -0.01391304 0.8099035
##                    MASE        ACF1
## Training set 0.6926648 0.001509038
```

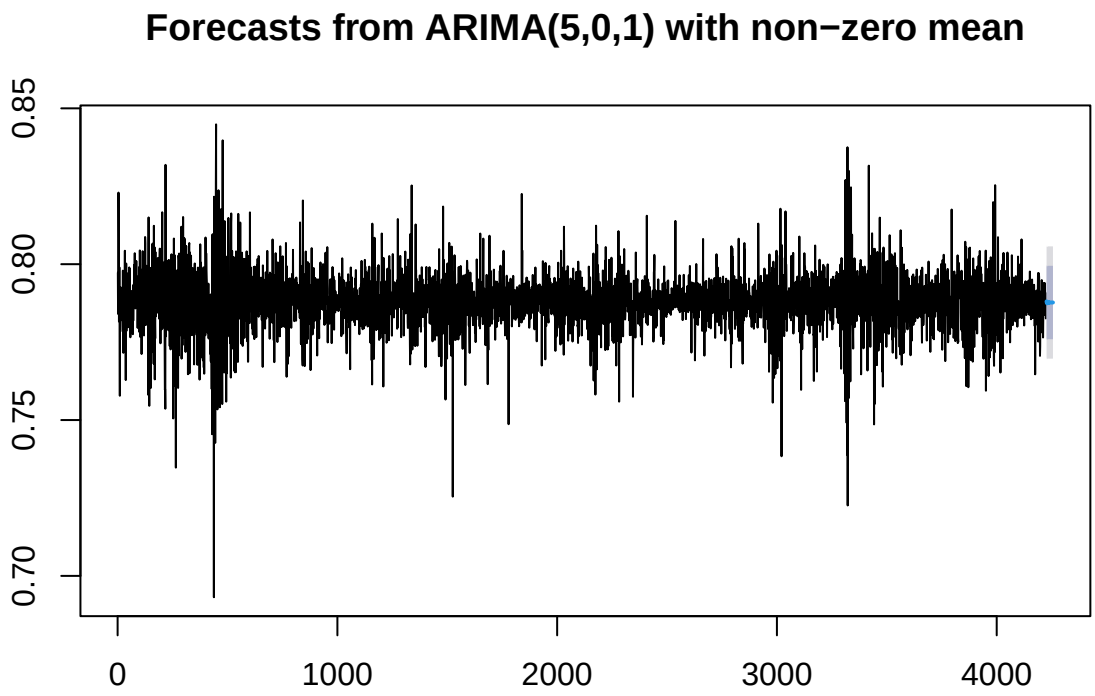
Model Type: ARIMA(5,0,1) with non-zero mean: This suggests that the model chosen is an ARIMA model with order (p=5, d=0, q=1). This means: p=5: The autoregressive part of the model (AR) has 5 lags. d=0: No differencing was applied since d=0. q=1: The moving average part of the model (MA) has 1 lag.

Coefficients and Standard Errors: ar1 to ar5: These are the coefficients for the AR terms from lag 1 to 5. Their values are -0.7412, -0.0450, -0.0168, 0.0339, and 0.0647, respectively. ma1: This is the coefficient for the MA term at lag 1, and its value is 0.7098. mean: The estimated mean of the series is 0.7877. The standard errors (s.e.) for each coefficient are given, which measure the accuracy of the coefficients. The smaller the standard error, the more accurate the estimate.

Model Evaluation Metrics: sigma^2 = 8.392e-05: This is the variance of the residuals (errors). log likelihood = 13838.84: This value represents the log likelihood of the model. The higher this value, the better the model fits the data. AIC, AICc, and BIC: These are information criteria values used to compare the goodness of fit of different models. Lower values suggest a better model fit. Among these: AIC (Akaike Information Criterion): It penalizes the addition of extra parameters. A value of -27661.68 is given. AICc (Corrected Akaike Information Criterion): It's an adjusted version of AIC to account for sample size. It has a value of -27661.65. BIC (Bayesian Information Criterion): It also penalizes the model for the number of parameters but is stricter than AIC. Its value is -27610.89.

Overall, the model seems to be performing reasonably well based on these metrics. The residuals (the differences between the predicted values and actual values) have low autocorrelation, which is a positive sign. The AIC, AICc, and BIC are useful when comparing this model to other potential models to determine which provides the best fit to the data.

```r
# Forecast next 30 days
forecasted_values <- forecast(fit, h=30)
plot(forecasted_values)
```




```
# Split data into training and test sets
train_end <- length(stock_log) - 30
train <- stock_log[1:train_end]
test <- stock_log[(train_end+1):length(stock_log)]
```

```
fit_train <- auto.arima(train, seasonal=FALSE)
forecasted_test <- forecast(fit_train, h=30)
forecasted_test
```

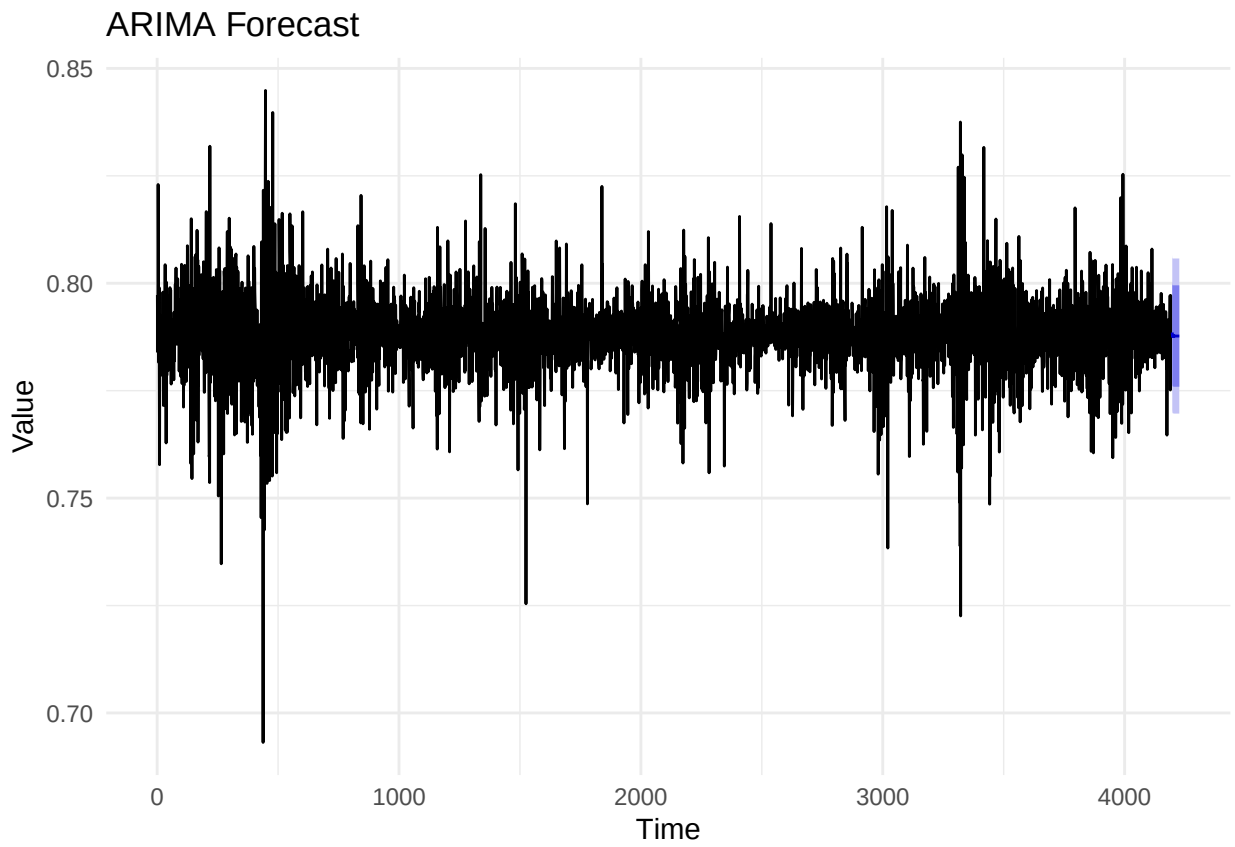
```
##      Point Forecast     Lo 80     Hi 80     Lo 95     Hi 95
## 4197      0.7885416 0.7767783 0.8003049 0.7705512 0.8065320
## 4198      0.7875755 0.7758061 0.7993449 0.7695758 0.8055753
## 4199      0.7879054 0.7761334 0.7996774 0.7699016 0.8059091
## 4200      0.7877964 0.7760244 0.7995684 0.7697927 0.8058001
## 4201      0.7876902 0.7759111 0.7994693 0.7696756 0.8057048
## 4202      0.7877784 0.7759908 0.7995661 0.7697508 0.8058061
## 4203      0.7876682 0.7758743 0.7994621 0.7696310 0.8057054
## 4204      0.7877654 0.7759690 0.7995618 0.7697244 0.8058064
## 4205      0.7876861 0.7758887 0.7994835 0.7696435 0.8057287
## 4206      0.7877385 0.7759401 0.7995369 0.7696944 0.8057826
## 4207      0.7877036 0.7759048 0.7995024 0.7696589 0.8057483
## 4208      0.7877246 0.7759257 0.7995235 0.7696798 0.8057694
## 4209      0.7877133 0.7759144 0.7995122 0.7696685 0.8057582
## 4210      0.7877180 0.7759191 0.7995169 0.7696731 0.8057628
## 4211      0.7877169 0.7759180 0.7995158 0.7696720 0.8057618
```

```
## 4212      0.7877161 0.7759172 0.7995150 0.7696712 0.8057610
## 4213      0.7877177 0.7759187 0.7995166 0.7696728 0.8057625
## 4214      0.7877160 0.7759171 0.7995149 0.7696711 0.8057609
## 4215      0.7877174 0.7759185 0.7995163 0.7696726 0.8057623
## 4216      0.7877163 0.7759174 0.7995152 0.7696714 0.8057612
## 4217      0.7877171 0.7759182 0.7995160 0.7696722 0.8057620
## 4218      0.7877166 0.7759177 0.7995155 0.7696717 0.8057615
## 4219      0.7877169 0.7759180 0.7995158 0.7696720 0.8057618
## 4220      0.7877167 0.7759178 0.7995157 0.7696719 0.8057616
## 4221      0.7877168 0.7759179 0.7995157 0.7696719 0.8057617
## 4222      0.7877168 0.7759179 0.7995157 0.7696719 0.8057617
## 4223      0.7877168 0.7759179 0.7995157 0.7696719 0.8057617
## 4224      0.7877168 0.7759179 0.7995157 0.7696719 0.8057617
## 4225      0.7877168 0.7759179 0.7995157 0.7696719 0.8057617
## 4226      0.7877168 0.7759179 0.7995157 0.7696719 0.8057617
```

```r
# Assuming train data is already loaded and the ARIMA model is already fit

fit_train <- auto.arima(train, seasonal=FALSE)
forecasted_test <- forecast(fit_train, h=30)

# Plot using ggplot2
autoplot(forecasted_test) +
  labs(title = "ARIMA Forecast", x = "Time", y = "Value") +
  theme_minimal()
```



For each time step, the model provides a point forecast and associated confidence intervals. The wider the

interval, the less certain the model is about its forecast for that point in time. We can use these intervals to assess the potential risk and variability in our forecasts. For example, if we're making business decisions based on these forecasts, knowing the range in which the actual values are likely to fall can help with risk management.

The forecasts appear to be fairly stable (around the 0.787 to 0.788 range). There don't seem to be any drastic changes or trends in the forecasted values over these periods. The prediction intervals (both 80% and 95%) seem reasonably narrow, suggesting the model is somewhat confident about its forecasts.

```
# Convert both series to simple numeric vectors
forecast_values <- as.numeric(forecasted_test$mean)
test_values <- as.numeric(test)

# Calculate RMSE
rmse <- sqrt(mean((forecast_values - test_values)^2))
print(paste("RMSE:", rmse))
```

```
## [1] "RMSE: 0.00621773823357338"
```

The smaller the RMSE, the better the model's predictions are matching the actual observed values. An RMSE of 0 would mean the predictions are perfect. Here, an RMSE of 0.0062 indicates that, on average, our forecasted values are off by about 0.0062 units in the log-transformed scale from the actual values.