Machine Learning to Forecast Google Stock Price

Brian M Wisniewski

2023-07-31

Introduction

The purpose of this report is to explore the question "How do different machine learning models compare in terms of accuracy and robustness for stock price prediction?". For practical uses, using machine learning to forecast stock price is complex beyond the scope of this report. Practical forecasting should involve consultation with financial experts for domain specific knowledge and the use of more sophisticated modeling techniques. Additionally, to evaluate robustness and accuracy of models is a context dependent and nuanced endeavor. Because of those limitations, this report is limited in it's explanatory power. As such, this report is intended as an intellectual exercise and a proof of concept argument to justify additional in depth analysis.

Data

The data set used in this report consists of historical stock market data for Google (GOOG). The historical data ranges from the past 5 years (7/25/2018 – 7/25/2023). The fields in this data set are Date, Open, High, Low, Close, Adj Close, and Volume. The names of these fields correspond to the conventional usage of those terms in the traditional financial contexts that they are often referenced. I downloaded the data from Yahoo! Finance (https://finance.yahoo.com/). According to them, the "US quotes are real-time for NASDAQ, NYSE, and NYSE American when available from Nasdaq Last Sale and if not available it will appear delayed from the consolidated tape." This financial data is public domain.

Exploratory Analysis

Exploratory analysis was conducted prior to the examination of the machine learning models. For brevity, some analysis has been omitted from this report. Figure 1 shows Google stock prices and volume over time. High co-linearity was observed between price related variables, posing a difficulty for feature importance analysis. Table 1 shows correlation between variables and the closing price. For this data, Volume showed negative correlation (R=-0.263) with the closing price. The other price variables demonstrated high positive correlations (R=~.98-.99). Note that this correlation assessment was performed to understand gross relationships between the variables. Price fluctuations are small and the computation used to assess the relationships is somewhat insensitive to these fluctuations. More precise assessments would be needed to gain a granular understanding of variable relationships. Figure 2 shows the distribution of percent change in price variables from close. That is, it shows the distribution of daily percent deviation from the close price for each of the other price variables. Outliers signal one of the major challenges of stock price forecasting with simple machine learning models, which is market volatility.

Methods

This report examines two machine learning regression models to forecast stock prices for Google: Random Forest Regression and Support Vector Regression (SVR). The data was divided into training and testing sets, and cross-validation with repeated sampling was utilized to evaluate the models' accuracy and robustness. The evaluation metrics used were Root Mean Squared Error (RMSE) and percentage RMSE, which measured the accuracy of the models' predictions. The results showed that the Random Forest Regression model outperformed SVR in terms of accuracy and robustness for short-term stock price prediction, indicating its potential utility for real-time forecasting.

Results

The results of the methods indicate that Random Forest Regression may be superior to Support Vector Regression (SVM), with respect to accuracy and robustness, in this context. The Random Forest model achieved a lower Root Mean Squared Error (RMSE) and a percentage RMSE, indicating better accuracy compared to the SVR model. Because cross validation was performed in each case, my inference is that Random Forest Regression may perform better with unseen data and, therefore, be considered more robust. This alone, however, may be insufficient for completely gauging robustness.

Feature importance was conducted with the Random Forest Model, though co-linearity across pricing variables limited the explanatory power of my results. Table 3 shows the features ranked in order of relative importance, based on their "IncNodePurity" values.

Conclusion

The results of this analysis are too limited in explanatory power to be of much practical utility and more work is needed. Employing hyper parameter tuning, expert knowledge, more training data, and external validation could be used to explore these models further. Other techniques could also be explored, such as Time Series Models.

Despite limitations, the research question appears to have been addressed. In this context, Random Forest Regression is more accurate and robust.

Table 1: Correlation between Variables and Close

Open	0.999
High	0.999
Low	0.999
Volume	-0.263

```
# Create the predictors and the target variable
x <- GOOG[, c("Open", "High", "Low", "Volume")]
y <- GOOG$Close

# Set up cross-validation for random forest
repeat_cv_rf <- trainControl(method = 'repeatedcv', number = 5, repeats = 3)

# Train the Random Forest Regression model with cross-validation
rf_model_cv <- train(
    x = x,
```

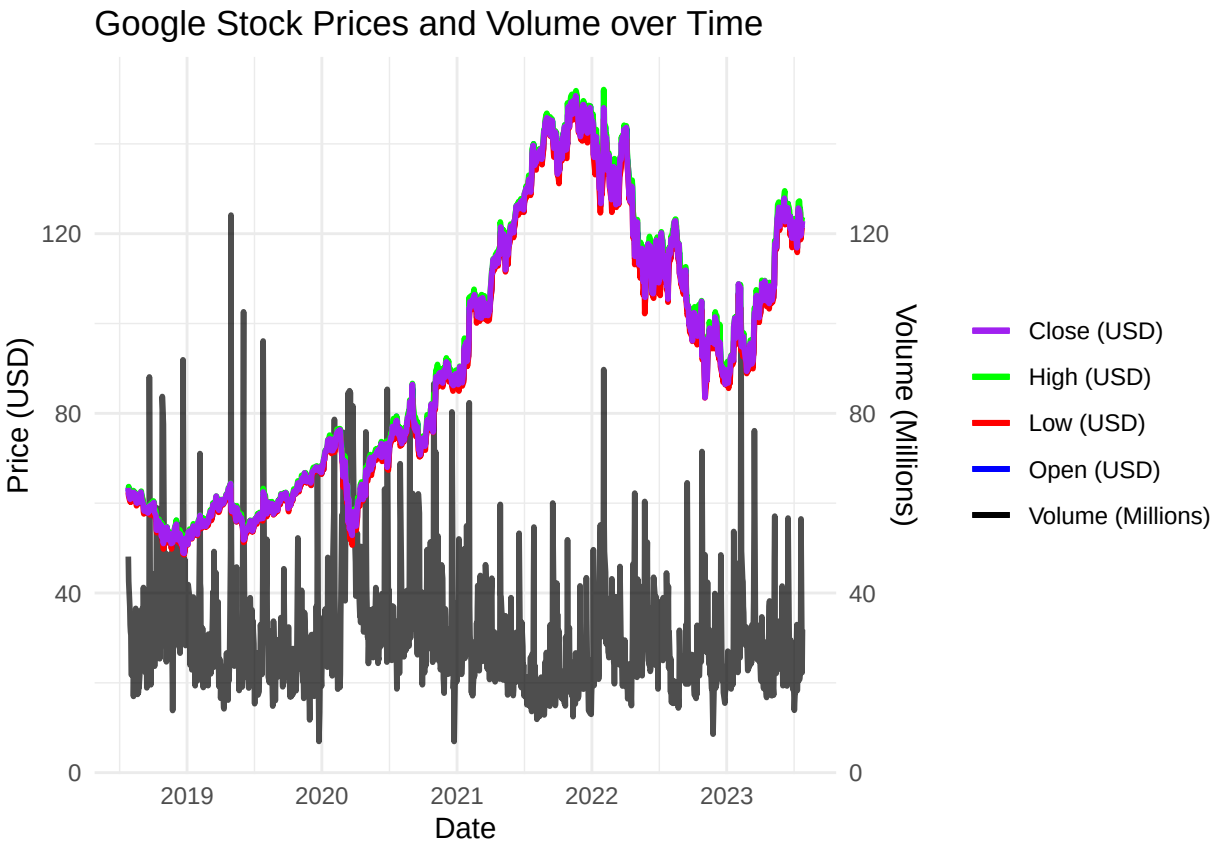

Figure 1: Google Stock Prices and Volume over Time

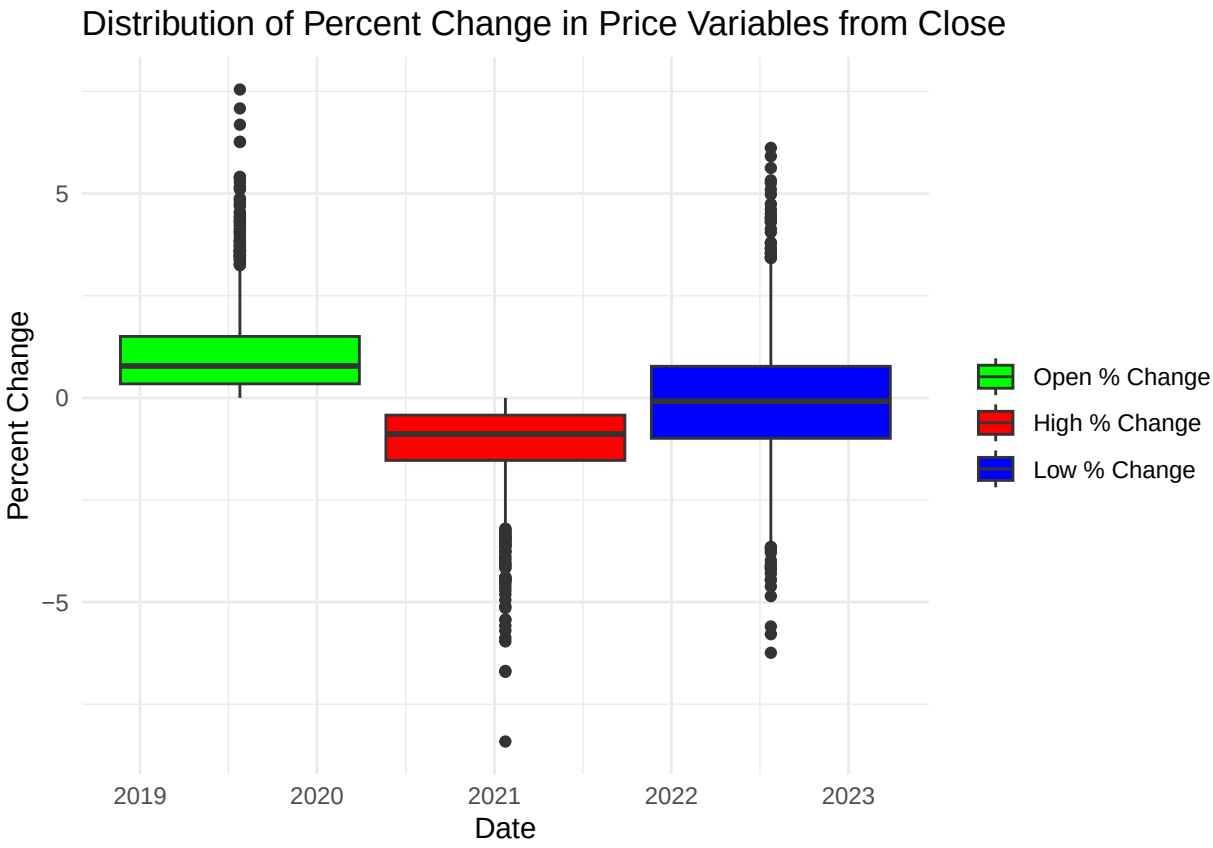

Figure 2: Distribution of Percent Change in Price Variables from Close

```
    y = y,
    method = 'rf',  # Use 'rf' for random forest
    trControl = repeat_cv_rf,
    metric = 'RMSE'
)

# Extract the final random forest model from cross-validation
final_rf_model <- rf_model_cv$finalModel

# Create the test dataset
test_data <- GOOG[1001:nrow(GOOG), ]

# Use the final_rf_model to make predictions on the test dataset (test_data)
test_x <- test_data[, c("Open", "High", "Low", "Volume")]
rf_predictions <- predict(final_rf_model, newdata = test_x)

# Calculate the RMSE
rf_rmse <- rmse(rf_predictions, test_data$Close)

# Calculate the range of closing prices in the test data
range_close_prices <- max(test_data$Close) - min(test_data$Close)

# Calculate the percentage RMSE (%RMSE)
percent_rmse <- (rf_rmse / range_close_prices) * 100

# View the RMSE and %RMSE
cat('Random Forest RMSE: ', rf_rmse, '\n')
```

```
## Random Forest RMSE:  0.4487143
```

```
cat('Random Forest %RMSE: ', percent_rmse, '%', '\n')
```

```
## Random Forest %RMSE:  1.010163 %
```

```
# Calculate the variable importance metric
rf_variable_importance <- importance(final_rf_model)

# Print the importance table
print(rf_variable_importance)
```

```
##        IncNodePurity
## Open       6769.1792
## High     515951.0800
## Low      610691.0029
## Volume      133.3083
```

```
# Load the required libraries
library(e1071)
library(caret)
library(Metrics)
```

```r
# Load the dataset
GOOG <- read.csv("GOOG.csv")

# Create the training dataset
train_data <- GOOG[1:1000, ]

# Create the test dataset
test_data <- GOOG[1001:nrow(GOOG), ]

# Extract the predictors and the target variable from the training dataset
train_x <- train_data[, c("Open", "High", "Low", "Volume")]
train_y <- train_data$Close

# Set up cross-validation for SVR
repeat_cv_svr <- trainControl(method = 'repeatedcv', number = 5, repeats = 3)

# Train the Support Vector Regression model with cross-validation
svr_model_cv <- train(
  x = train_x,
  y = train_y,
  method = 'svmRadial',  # Use 'svmRadial' for SVR
  trControl = repeat_cv_svr,
  metric = 'RMSE'
)

# Extract the predictors from the test dataset
test_x <- test_data[, c("Open", "High", "Low", "Volume")]

# Make predictions using the Support Vector Regression model
svr_predictions_cv <- predict(svr_model_cv, test_x)

# Calculate the RMSE for the cross-validated SVR model
actual_close_prices <- test_data$Close
svr_rmse_cv <- rmse(svr_predictions_cv, actual_close_prices)

# Calculate the range of closing prices in the test data
range_close_prices <- max(actual_close_prices) - min(actual_close_prices)

# Calculate the percentage RMSE (%RMSE) for the cross-validated SVR model
percent_svr_rmse_cv <- (svr_rmse_cv / range_close_prices) * 100

# View the RMSE and %RMSE for the cross-validated SVR model
cat('SVR RMSE (cross-validated): ', svr_rmse_cv, '\n')
```

```
## SVR RMSE (cross-validated):  1.987013
```

```r
cat('SVR %RMSE (cross-validated): ', percent_svr_rmse_cv, '%', '\n')
```

```
## SVR %RMSE (cross-validated):  4.47324 %
```

Table 2: Comparison of RMSE and %RMSE for Different Methods

Method	RMSE	Percent.RMSE
Random Forest	0.449	1.010
Support Vector Regression	1.987	4.473

Table 3: Feature Importance Ranking based on IncNodePurity

Feature	IncNodePurity	Rank
Low	679369.9424	1
High	447838.9155	2
Open	7130.1924	3
Volume	136.2768	4