FIN 285 A

FINAL RESEARCH REPORT

Comparative Analysis of Machine Learning and Time Series Forecasting Algorithms in Stocks Price Prediction

Vicky Ke, Jianying Zhu

Instructor: Professor Ying Becker

Fall 2021

TABLE OF CONTENTS

1. Introduction 1

2. Literature Review 1

- 2.1. Returns 1
- 2.2. Performance Measures 2
- 2.3. Value-at-Risk 3
- 2.4. Expected Shortfall 4

3. Data and Methodology 4

- 3.1. Data Description 4
- 3.2. Model Selection 7

4. Analysis and Interpretation 10

- 4.1. Model Results 10
- 4.1.1 Machine Learning Model and Its Best Accuracy 10
- 4.1.2 Time Series Model and Its Best Accuracy 11
- 4.2 Select the Best Predicted Model, Determine the Forecast Length and Make Predictions 12
- 4.3 VaR and Expected ShortFall Forecast 13

5. Conclusion and Recommendations 14

Reference 15

Appendix - Code 1 16

Appendix - Code 2 17

Appendix - Code 3 18

Appendix - Code 4 20

Appendix - Code 5 21

Appendix - Code 6 22

1. Introduction

This report aims to examine two innovative methods of stock price analysis - using machine learning and forecasting algorithms to forecast one month on future stock price and returns, and to estimate its VaR and Expected Shortfall value. Our target stock is the S&P 500 index, and we are going to use the other three stocks from three different sectors to assist with the prediction: Walmart - from consumer staple sector; Microsoft - from technology sector; Pfizer - from pharmaceutical sector. The forecasting period is one month, or 21 trading days.

The rest of the report is structured as follows. In the next section, we briefly present a literature review on the calculation methodologies on the four aspects - stock returns, performance measures, VaR, and Expected Shortfall. In the Third section, we perform a descriptive analysis of the stock price and introduce the models we use for forecasting. We then interpret the forecasting results, make predictions, and calculate VaR and Expected Shortfall value. In the last section or Fifth section, we analyze the results and provide the investment suggestions to investors.

By reading the report, the audience should be able to know the predicted price of S&P 500, Walmart, Microsoft, and Pfizer in the next month, and the recommendation we provided is based on the forecasting results, such as the expected risk and losses of purchasing, and other practical considerations.

2. Literature Review

There are several concepts we use in the report for assessing the companies' stock price. A large variety of metrics has been documented dating back to history and has gradually formed important credentials taken as reference by institutional professionals to give clients greater perspective. Before proceeding further, we believe that it is necessary to understand both mathematical and statistical computational methods and the logic behind our visualizations.

2.1. Returns

The first concept is the returns of a stock. The three main methods of calculating returns are Arithmetic Returns, N-Period Returns, and Continuously Compounded returns. All these methods use different variable types and periods of observations as inputs. These three approaches estimate the stock returns as following:

The basic return, called Arithmetic Returns, is very simple and the format is a 1-period return as the left side of Equation 1 . T is the sample length. P_t is Price at time t as in $P_1, P_2, ... P_T$

The second type is compounded return over n periods, which shows on the right side of Equation 1, is the N-Period Returns.

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}} = \frac{P_t}{P_{t-1}} - 1 \qquad R_t(n) = (1+R_t)(1+R_{t-1})\dots(1+R_{t-n+1}) - 1$$

$$R_t(n) = \frac{P_t}{P_{t-n}} - 1$$

Equation 1

The third one is called continuously compounded returns, which is used to calculate geometrically or log returns over n periods. Equation 2 illustrates the calculation process.

$$r_t = \log(P_t/P_{t-1}) = \log(P_t) - \log(P_{t-1}) = \log(1+R_t)$$

$$\begin{aligned} r_t(n) &= \log(1+R_t(n)) \\ &= \log(1+R_t) + \log(1+R_{t-1}) + \dots + \log(1+R_{t-n+1}) \\ &= \log(P_t) - \log(P_{t-1}) + \log(P_{t-1}) - \log(P_{t-2}) + \dots - \log(P_{t-n}) \\ &= \log(P_t) - \log(P_{t-n}) \end{aligned}$$

Equation 2

Since we run the models use the historical data and forecast over a certain time period, we choose the continuously compounded returns to calculate the stock returns. We use Adj Close as stock price, and use np.log(Adj Close(t)) - np.log(Adj Close(t-21)) to calculate daily stock return over 21 trading days period, where Adj Close(t-21) = Adj Close(t)/Adj Close(t-21) - 1 and we have dropped the values of first 21 days.

2.2. Performance Measures

Performance measures are also known as Error Metric, which is a way to quantify the performance of a model and provide a criterion for the predictors to quantitatively compare different models. When considering the performance of any forecasting model, the prediction values it produces must be evaluated, so they could give us a way to more objectively gauge how well the model executes its tasks. We primarily introduce two popular metrics: Root Mean Squared Error and Mean Absolute Percentage Error.

Root Mean Squared Error (RMSE) measures the sample standard deviation of the difference between actual and predicted values. These differences that were used for estimates are called residuals during the in-sample calculation and are called prediction errors in the out-of-sample calculation. So Root Mean Squared Error (RMSE) usually can be interpreted as the standard deviation of the unexplained variance.

Root Mean Squared Error (RMSE) is a good measure of how accurate the model predicts and is the most important criterion for fit. It indicates the absolute fit of the model to the data–how close the observed data points are to the model’s predicted values. Lower values of RMSE indicate how accurate the model results are. Equation 3 is used to compare the obtained results from models:

$$\mathrm{RMSE} = \sqrt{\sum_{i=1}^{N} \frac{(\mathrm{Experimenta\ l_i} - \mathrm{Model\ \ prediction\ s_i})^2}{N}}$$

Equation 3

Mean Absolute Percentage Error(MAPE) is a type of percentage error and is also very useful because they are scale-independent, so they can be used to compare forecasts between various data series. The disadvantage of it is that it cannot be used in a series that has zero value. The calculation format is as follows:

$$\mathrm{MAPE} = \frac{1}{N} \sum_{t=1}^{N} \left| \frac{\mathrm{Actual}_t - \mathrm{Model\ Prediction}_t}{\mathrm{Actual}_t} \right|$$

Equation 4

We will compare the model performance with these error metrics and discuss further how we can pick up the best model in Section Four.

2.3. Value-at-Risk

Value at Risk (VaR) now is a valuable measure to evaluate historical volatility. The history of VaR as a required measure has been dated back to 1922 (Vasileiou, 2017). The first crude VaR measure was published by Leavens (Leavens, 1945). In the following years, the VaR progress that is worth highlighting is JP Morgan’s attempt under the name Risk Metrics to standardize the VaR estimation process (Morgan, 1996), which boosted the VaR significance not only for practitioners but for regulators also (Basle Committee on Bank Supervision, European VaR Directive (Vasileiou, 2017). Since then, the VaR measure has drawn the attention of many scholars, financial analysts, and regulators.

VaR, as a statistical financial metric, is defined to estimate the amount of potential loss that could happen in an investment portfolio over a specified period of time for a given confidence interval. VaR estimation has several methodologies. In this report, we mainly use historical simulation because we use real historical data over 21 years and recalculated the company portfolio’s returns in the previous step for the last 21 observations, assuming that the following 21 days will be similar to the previous 21 days, and from these 21 returns, we set p, the significance value, equals to 5%.

Our first step is to find the future price P_{t+1} with the np.exp function, as the statistical format: $P_{t+1} = 100 \times e^{R_t}$. We then use np.percentile function to calculate P^* that is simply the p quantile of the portfolio valuation 21 days in the future. The formula looks like this: $P^* = q_p(P_{t+1})$. Equation 5 is the equation of VaR based on the value of P^* .

$$\mathrm{VaR}(p) = 100 - P^*$$

Equation 5

2.4. Expected Shortfall

We then move to Expected Shortfall which produces better incentives for traders than VaR. Expected Shortfall, also known as conditional value at risk or cVaR, is another alternative measure to qualify the expected loss(average value) of tail risk beyond VaR, which is used in optimizing portfolios for effective risk management.

The expected return is noted as $E(R_t)$. The stock price at period t+ 1 is noted as P_{t+1} .

Our first step is to define the future value of the stock, V_{t+1} , as the following equation showed $V_{t+1} = 100 \times (1 + V_t)$. We then use np.percentile function to calculate V^* , which estimates p quantile of the portfolio distribution over 21 days in the future. The calculation format is the following: $V^* = q_p(V_{t+1})$. The next step is to estimate conditional mean $\bar{V}$ for prices at or below V^* . Equation 6 shows the final result of the Expected shortfall.

$$\bar{V} = E(V_{t+1} | V_{t+1} \leq V^*)$$
$$\text{ES}(p) = -(\bar{V} - V_t) = V_t - \bar{V}$$

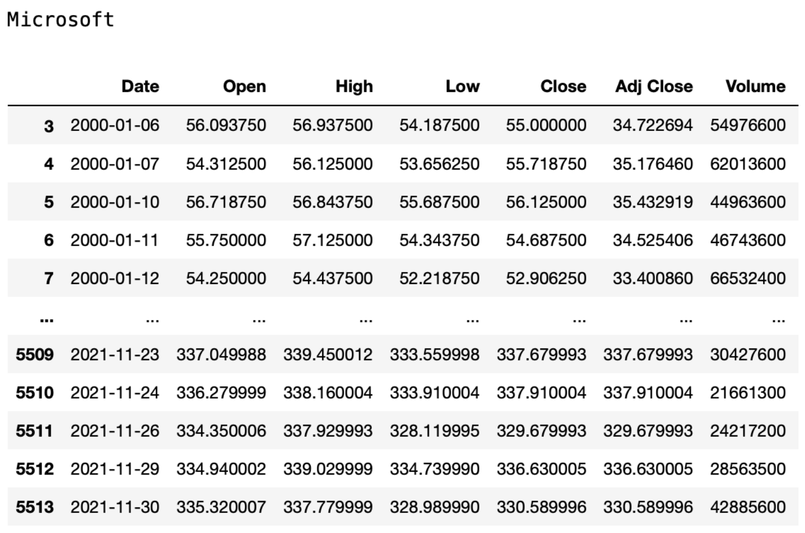
Equation 6

Expected shortfall measures could be very sensitive to the inclusion or exclusion of very low probability extreme events. Expected shortfall measures are likely to be much less stable than corresponding VaR measures. Because Expected Shortfall is sensitive to extreme events, there is no way to backtest Expected Shortfall without making additional assumptions about the distribution of losses beyond the VaR threshold.

3. Data and Methodology

3.1. Data Description

The raw datasets we loaded are the four stocks data: Walmart, Microsoft, Pfizer, and S&P 500 index, starting from the year 2000/01/01 to 2021/11/30. The data was retrieved from yahoo finance by reading into python using yfinance package. Each dataset has 7 columns and 5531 rows, which columns are including Date, Open, High, Low, Adj Close, and Volume. We use Adj Close as the stock price to make forecasts and calculate daily returns. Given the Adj Close as stock price, we use Adj Close(t)/Adj Close(t-1) - 1 to calculate daily stock return for each stock. Figure 1 is an example of the stock data we use in this report.



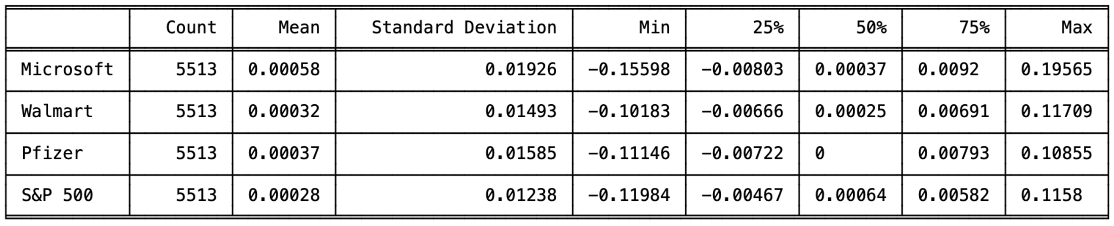
```
Microsoft
```

	Date	Open	High	Low	Close	Adj Close	Volume
3	2000-01-06	56.093750	56.937500	54.187500	55.000000	34.722694	54976600
4	2000-01-07	54.312500	56.125000	53.656250	55.718750	35.176460	62013600
5	2000-01-10	56.718750	56.843750	55.687500	56.125000	35.432919	44963600
6	2000-01-11	55.750000	57.125000	54.343750	54.687500	34.525406	46743600
7	2000-01-12	54.250000	54.437500	52.218750	52.906250	33.400860	66532400
...	...	...	...	...	...	...	...
5509	2021-11-23	337.049988	339.450012	333.559998	337.679993	337.679993	30427600
5510	2021-11-24	336.279999	338.160004	333.910004	337.910004	337.910004	21661300
5511	2021-11-26	334.350006	337.929993	328.119995	329.679993	329.679993	24217200
5512	2021-11-29	334.940002	339.029999	334.739990	336.630005	336.630005	28563500
5513	2021-11-30	335.320007	337.779999	328.989990	330.589996	330.589996	42885600

Figure 1

3.1.1 Descriptive Analysis

Figure 2 is the table that summarizes all statistics. From the table, we can see that Microsoft has the highest average return, but also the highest standard deviation and the largest maximum to a minimum range. S&P 500 has the lowest average return and the lowest standard deviation. This is saying that Microsoft stock is a higher risk stock whereas S&P 500 index is a lower risk stock.



	Count	Mean	Standard Deviation	Min	25%	50%	75%	Max
Microsoft	5513	0.00058	0.01926	-0.15598	-0.00803	0.00037	0.0092	0.19565
Walmart	5513	0.00032	0.01493	-0.10183	-0.00666	0.00025	0.00691	0.11709
Pfizer	5513	0.00037	0.01585	-0.11146	-0.00722	0	0.00793	0.10855
S&P 500	5513	0.00028	0.01238	-0.11984	-0.00467	0.00064	0.00582	0.1158

Figure 2

3.1.2 Violin Plot

Figure 3 is a violin plot of Adj Close and returns of four stocks from Year 2000 to Year 2021. Violin plot can be regarded as a combination of the box plot and the kernel density plot. In the Violin plot, we find information about: median, interquartile range, and the distribution of the data. It is clear to see that the stock price distributions of all four stocks are skewed to the right, and returns are nearly normally distributed. Microsoft's stock price and returns have a larger tail compared to the other three stocks, which means that Microsoft has higher volatility. Pfizer's stock is not very separated from the mean, which means that the value of the stock does not grow too much across 20 years.

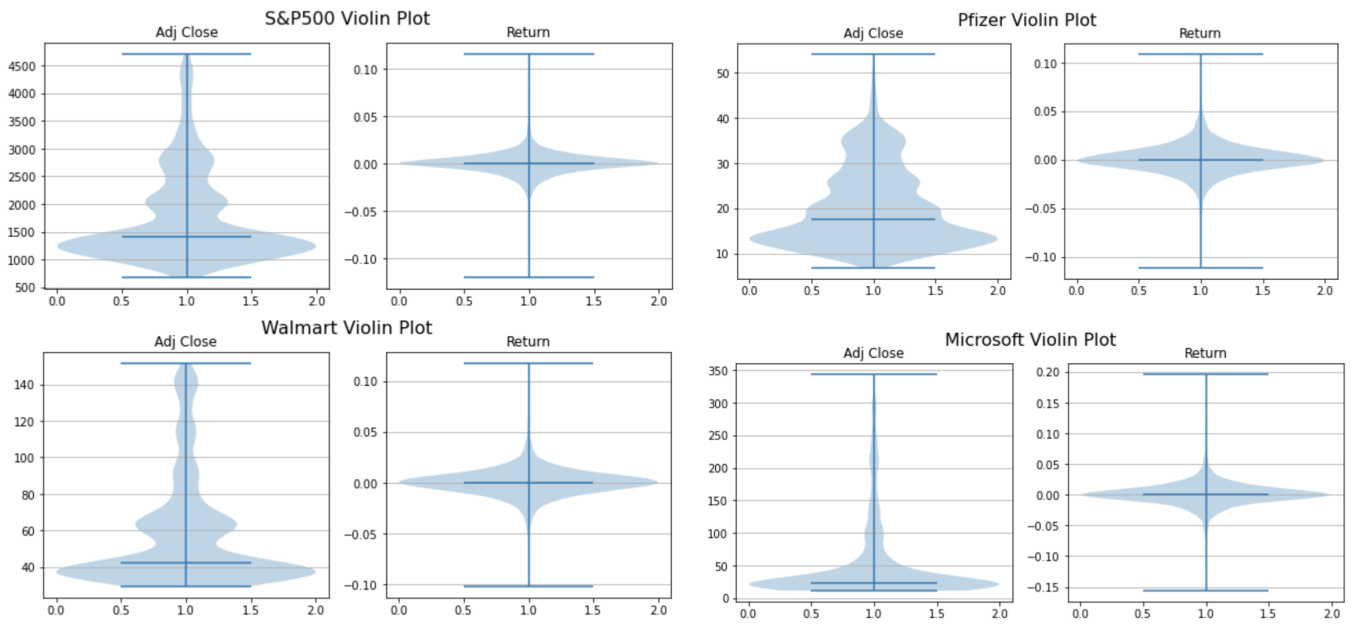


Figure 3

3.1.3 Scatter Plot

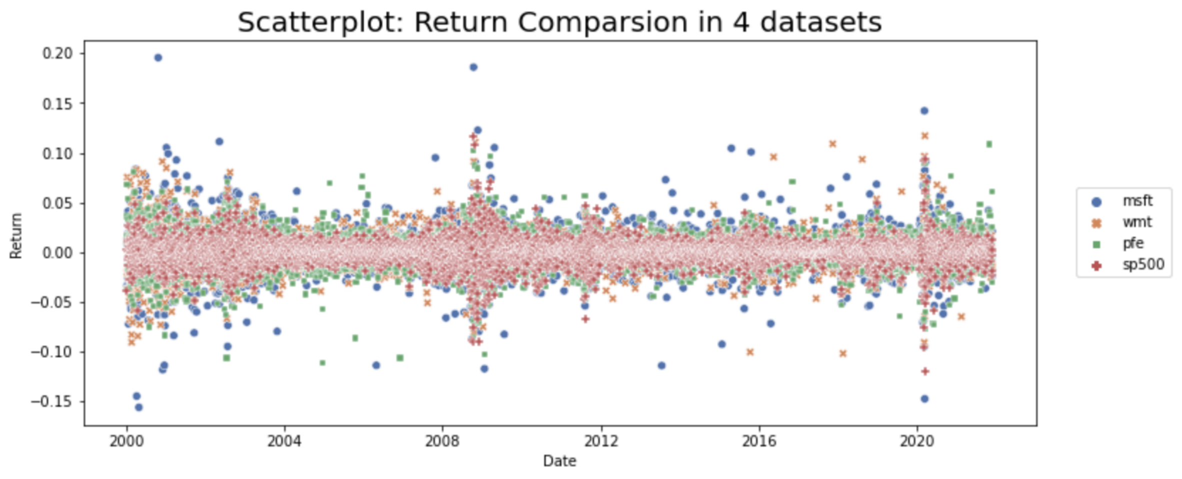


Figure 4

Figure 4 is a scatter plot of returns of all four types of stocks. From the plot above, we can see that the blue dots are in the outer layer most of the time, followed by green and orange. Red dots are predominantly in the inner layer, or in the middle of the plot. This information shows that Microsoft stock value (in blue dots) has the greatest fluctuation, either increasing or decreasing, and S&P 500 index value (in red dots) has the smallest fluctuation among the four. This infers that the people who buy Microsoft stock will have a huge gain and huge losses, while people who buy the S&P 500 index will have relatively steady gains and fewer losses.

3.2. Model Selection

In this section, we will briefly illustrate the models we ran in predicting the S&P 500 returns, including the models, the parameters, and model applications to the stock return data.

We use two types of models for the prediction: machine learning models, and time series forecasting models.

3.2.1 Machine Learning Model and Its Best Parameter Selection

Among all machine learning models, we have picked four models that are better suited for stock return prediction: Lasso Regression, Ridge Regression, Random Forest Regressor, and Gradient Boosting Regressor.

The first step is to know what are the independent and the dependent variables of the model. The main goal of this is to use the stock returns of Microsoft, Walmart, and Pfizer to predict the stock returns of the S&P 500. Therefore, the X variable or independent variables we use are the stock returns of Microsoft, Walmart, and Pfizer, and the Y variable or dependent variables are the stock returns of the S&P 500.

The second step is to do the train test split. In machine learning modeling or other data modeling, train test split is a necessary preparation before modeling. We use the training dataset to train the data, and use the testing dataset to test the model's accuracy. This step can also help to determine the overfitting or underfitting problem. To do the train test split, we use the train_test_split function from the sklearn python library, and set 80% as training data, 20% as testing data (refer to code 1 in Appendix).

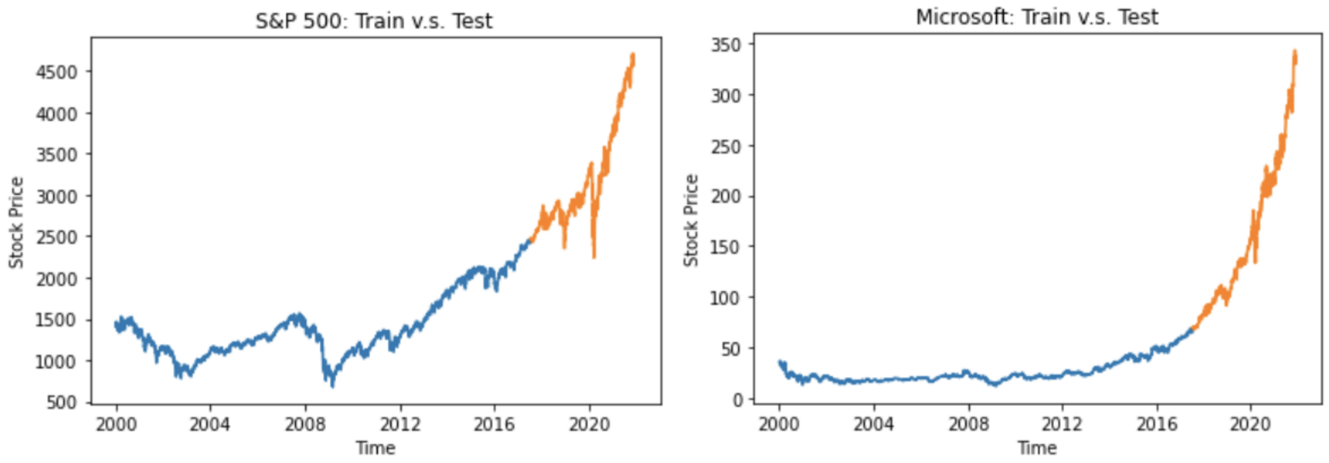


Figure 5

For the time series split, we do not split by random order. Instead, we split by time order. We set 80% of the data as training data, that is all the stock returns before 2017/07/16, and set later 20%, that is all the stock returns after 2017/07/16. Figure 5 is the visualization of the data after train test splitting, where training data is in the blue line, and testing data is in the orange line.

The last step is to determine the best parameters of the model. We use the grid search method with a for loop in python to find the best parameters sets (refer to code 2 in Appendix). We first set a range of values for the parameter, then we run a for loop to run through all the combinations of parameters to find the set with the highest training accuracy score. The model with the highest accuracy score is the best model, which will be used in modeling later.

3.2.2 Time Series Model and Its Best Parameter Selection

The second type of model is the time series forecasting model. The model we used is the ARIMA model, which stands for Autoregressive Integrated Moving Average (ARIMA) model. This type of model makes the prediction based on the historical data or performance.

The first step is to determine the target data for forecasting and followed by the train test splitting. Different from the target data in machine learning modeling, we use stock price as the target for forecasting. Usually, a statistics time series model, such as ARIMA, requires the target data to be stationary in order to be effective. A stationary time series data is a type of data whose properties do not depend on the time at which the series is observed. However, as we know, the price of the stock is increasing year over year, because of the increase of the company's value and inflation.

The second step is checking for stationarity of the time target data. Figure 6 below shows both Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots of the residuals from the ARIMA model. With the below ACF plot, the lags total 35 and above. We can see that the correlation scale is from -1 to 1, which represents a scale from zero correlation to full correlation. With there being most lags with a positive correlation above 0, this time series will possibly need a significant level of differencing to achieve stationarity.

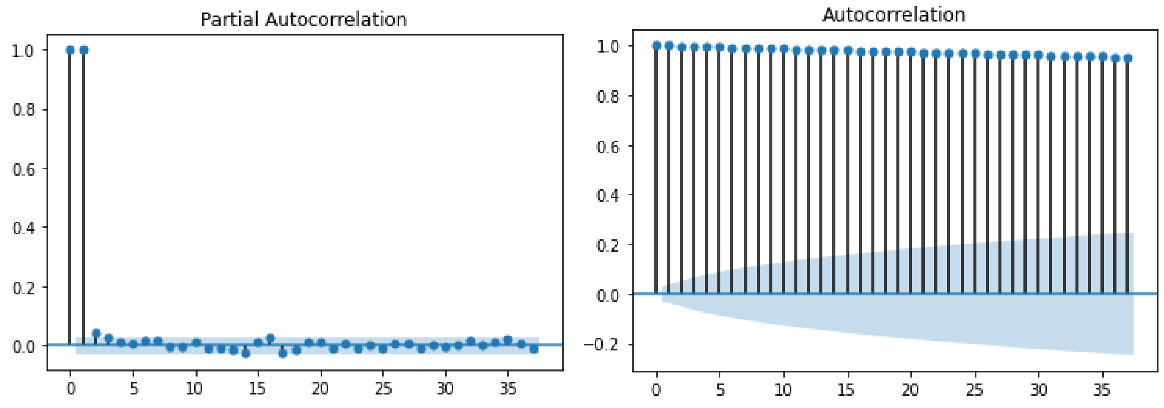
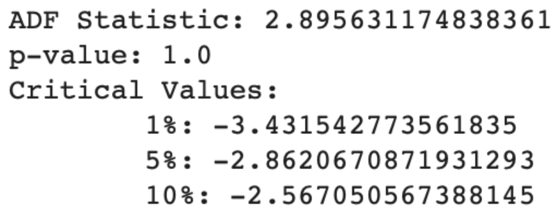


Figure 6

On the other hand, Partial Autocorrelation summarizes the relationship between an observation in a time series and observations at previous time steps, but with the relationships of intervening, the observations are removed. This is the critical difference between Autocorrelation and Partial Autocorrelation - the indirect correlations are removed. Therefore, in the PACF plot in figure 6, we see this happen. Because the indirect correlations are not calculated, the correlation "legs" dwindle completely to 0.

With the inspection of the PACF, we can learn how many AR terms we need to use to explain the autocorrelation pattern in a time series. Overall, the ACF and PACF plots can help us to verify that the Time Series is stationary.

We also doubly confirmed by running an Augmented Dickey-Fuller test to check for the stationarity of the stock price data. To run an Augmented Dickey–Fuller test in python, we use the adfuller function from statsmodels python library. We run the adfuller function on stock price and check for the ADF statistics. If the ADF statistics is smaller than the absolute value of the critical value at 5% or 1% significant level, we reject the null hypothesis - the time series is non-stationary, and conclude that the stock time series is stationary and does not need the transformation. Figure 7 is the result of the Dickey-Fuller Test for S&P 500 stock price. We observe that the 2.8956 (ADF statistics) is greater than the 2.8620 (the absolute value of critical value under 5% significant level), so we cannot reject the null hypothesis and conclude that the time series is non-stationary.



```
ADF Statistic: 2.895631174838361
p-value: 1.0
Critical Values:
        1%: -3.431542773561835
        5%: -2.8620670871931293
        10%: -2.567050567388145
```

Figure 7

The third step is generating a stationary time series for the purposes of forecasting. To transform a non-stationary to a stationary time series, we usually do the log-differencing transformation. Basically, we first do a log transformation on the original stock price, then we calculate the difference between log price at t and log price at t-1. Figure 8 is an example of the log-differencing transformation of the S&P 500 stock price. The left side is before the transformation versus the right side is after the transformation. The data on the right graph now becomes stationary and ready for the ARIMA forecasting.

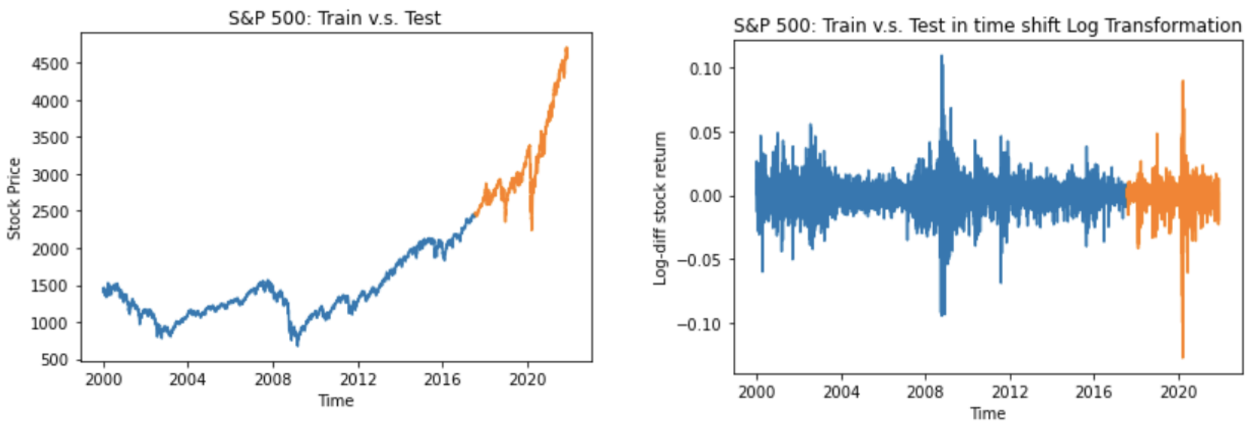


Figure 8

The last step is to find the best ARIMA parameter. The ARIMA model has three key parameters, and it is usually written as ARIMA(p,d,q). To determine the value for p,d, and q, we apply the auto_arima function from pmdarima python library. This function will automatically search for all possible p,d, and q values, and output the best combination of the parameters (see code 3 in Appendix for more details on the function and output). After running the function and finding out the best combination of parameters, we are ready to put the parameters into the model and train the data.

4. Analysis and Interpretation

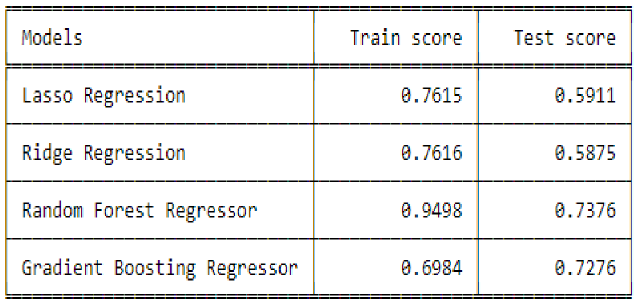
4.1. Model Results

In this section, we will evaluate the model performance that we described in the previous section and we then compare the results to select the best accuracy score of the same type of models.

Still, we focus on two types of models: machine learning models, and time series forecasting models.

4.1.1 Machine Learning Model and Its Best Accuracy

Figure 9 visualizes the accuracy score through all four machine learning models.



Models	Train score	Test score
Lasso Regression	0.7615	0.5911
Ridge Regression	0.7616	0.5875
Random Forest Regressor	0.9498	0.7376
Gradient Boosting Regressor	0.6984	0.7276

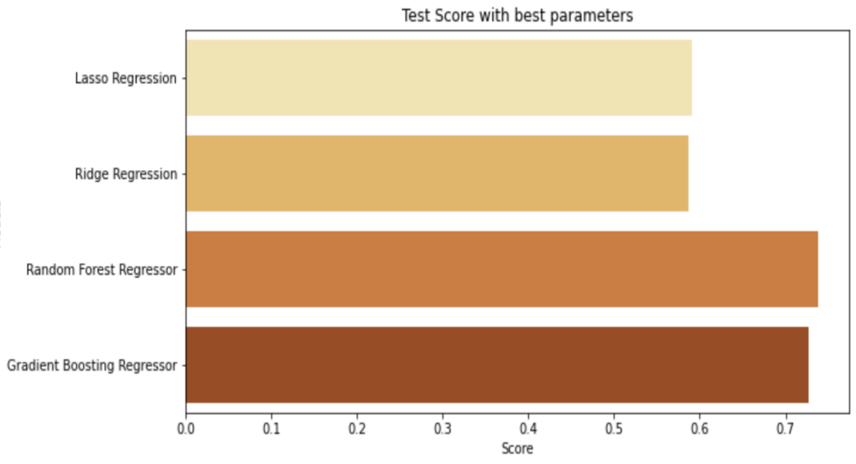


Figure 9

As shown in figure 9, Random Forest has the highest training score of 94.98% and the highest testing score of 73.76%. From the graph, we observe that Random Forest could have the problem of overfitting. We see the training score is about 94% accuracy, but once it applies to the new testing data, the accuracy score drops dramatically to 73%. It happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. This means that the noise or random fluctuations in the training data is picked up and learned as concepts by the model. Because of this, Random Forest fits the noise in the stock data, which negatively impact the Random Forest Model's ability to generalize new results. It creates completely untrustworthy results which appear to be statistically significant.

This problem can be addressed by either drawing a random sample that is large enough to handle all of the terms that we expect to include in the model or pruning a tree after it has learned in order to remove some of the detail it has picked up.

Consequently, we selected the Gradient Boosting Model, as it has the second-highest accuracy score of 69.84% in training and 72.76% in testing as our best model of forecasting.

4.1.2 Time Series Model and Its Best Accuracy

After running the best model with the log differencing transformation price, we then transform it back to the original price using the cumsum and np.exp functions (refer to code 4 and 5 in Appendix).

Figure 10 visualizes the actual testing data (in blue line) overlapped with the predicted data (in orange line). We see that the orange line is very close to the blue line, which shows that our ARIMA forecast model can create a satisfying result.

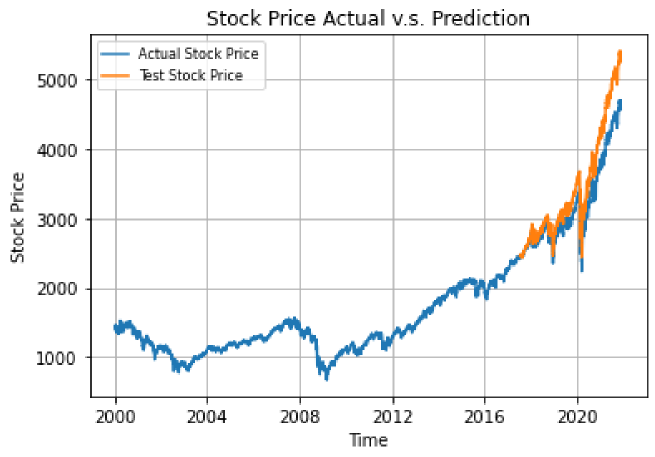
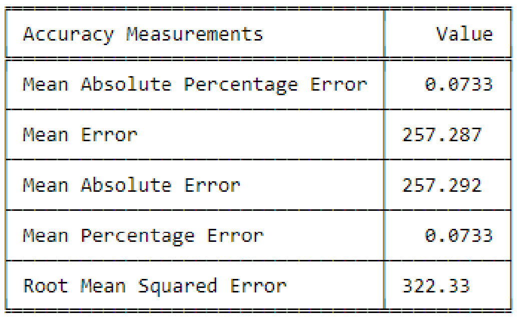


Figure 10

After reviewing the results, we assess the accuracy of the models by applying multiple accuracy measurements to quantify the performance of the prediction. Figure 11 is the table of five key accuracy measurements of the S&P 500 index. Root Mean Squared Error (RMSE) has the highest predicted error of 322.33, which is about an average of 6% of prediction error or 94% of prediction accuracy. Besides, Mean Absolute Percentage Error (MAPE) has the exactly same number as Mean Percentage Error (MPE). Mean Absolute Percentage Error (MAPE) is another commonly used measure for forecast accuracy, which is the average of the percentage errors. A Mean Absolute Percentage Error (MAPE) value of 7.33% means that the average difference between the forecasted value and the actual value is 7.33%. The lower the value for MAPE, the better a model is able to forecast values. Mean Percentage Error (MPE) is also the computed average of percentage errors between actual and forecasted values. Mean Error(ME) and Mean Absolute Error(MAE) are close to each other. However, Mean Absolute Error(MAE) is less sensitive to the occasional large error as it does not square the errors in the calculation.



Accuracy Measurements	Value
Mean Absolute Percentage Error	0.0733
Mean Error	257.287
Mean Absolute Error	257.292
Mean Percentage Error	0.0733
Root Mean Squared Error	322.33

Figure 11

In brief, all of the different types of error measuring techniques we explained above are useful when dealing with time-series data. We decide to pick RMSE and MAPE as our best indicators that assure our models to get an accurate and unbiased forecast.

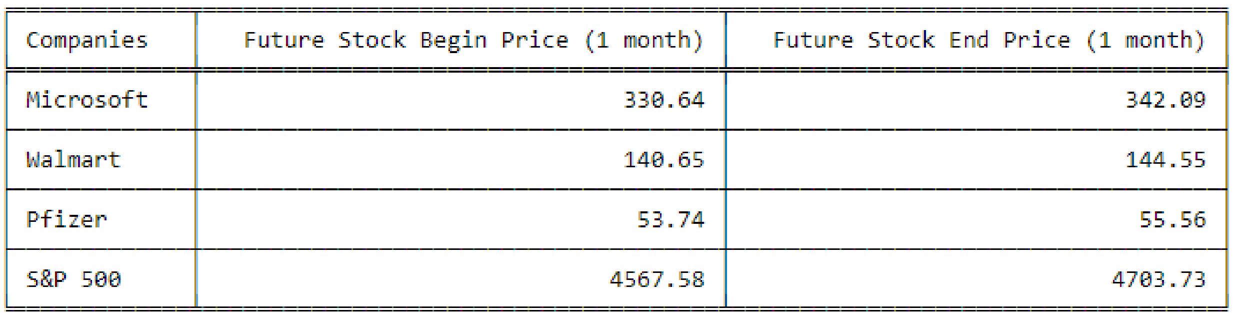
4.2 Select the Best Predicted Model, Determine the Forecast Length and Make Predictions

After reviewing the accuracy of both machine learning models and ARIMA model, we make a comparison between these two, and figure out that the ARIMA model results in the highest accuracy.

Given the property of stocks, we set the forecast length as one month. Since the stock market only opens on weekdays, which means, there are a total of 21 weekdays in a month, we set our forecasting length at 21 days to forecast the stock price next month.

Plugin the timing variable, we make a 21-day forward forecast on the stock price for all four types of stocks. Figure 12 shows the results of the forecast using the ARIMA model.

With the figure below, we see that all the stocks' prices will increase in the next 21 days. S&P 500 will be $4703.73, Microsoft will be $342.09, Walmart will be $144.54, and Pfizer will be $55.56.



Companies	Future Stock Begin Price (1 month)	Future Stock End Price (1 month)
Microsoft	330.64	342.09
Walmart	140.65	144.55
Pfizer	53.74	55.56
S&P 500	4567.58	4703.73

Figure 12

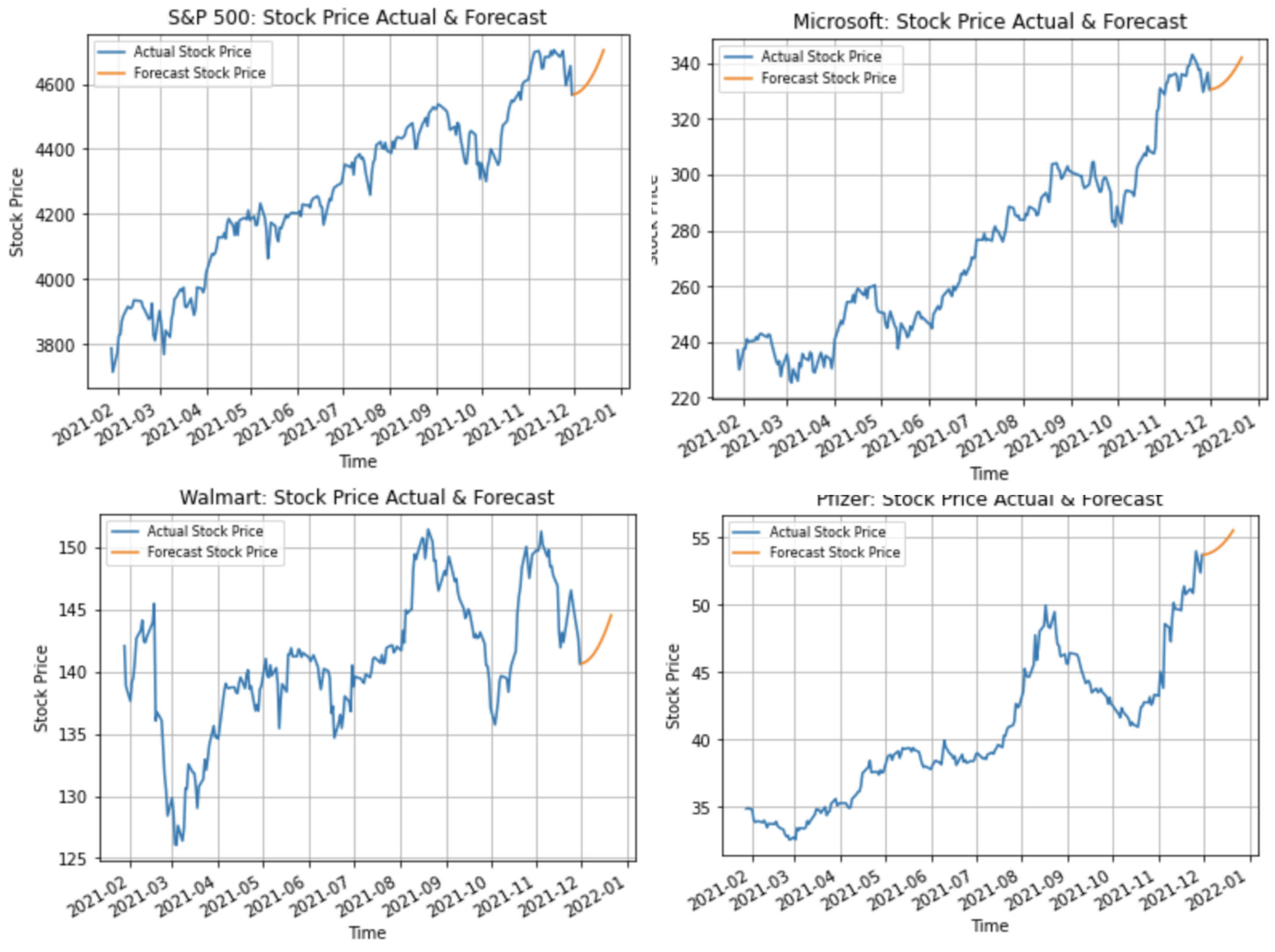
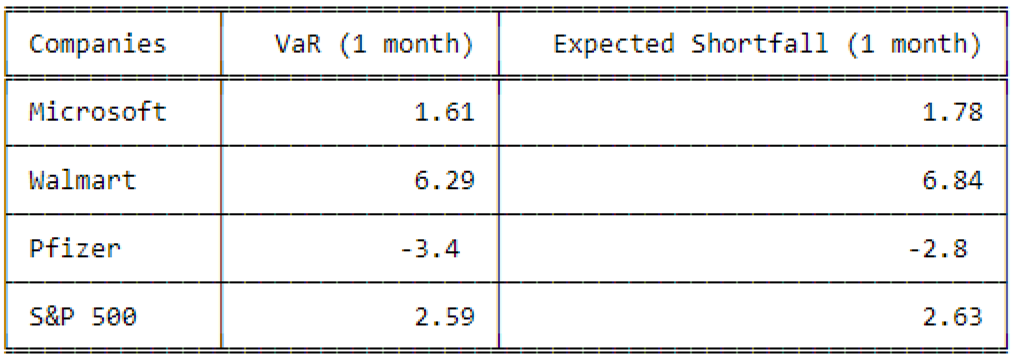


Figure 12

4.3 VaR and Expected ShortFall Forecast

The next step is to measure the risk of buying the stock today and to hold it for one future month. This is usually measured by VaR and Expected Shortfall (ES). To calculate VaR and ES, we use functions that are explained in Section Two, with t or time of 21 days and p or significance level as 0.05 (see code 6 in Appendix for more details on the function and output).

Figure 13 summarizes the results of Value at Risk (VaR) and Expected Shortfall (ES) for all four types of stocks: Microsoft, Walmart, Pfizer, and S&P 500. From the results, we see that Walmart has the highest VaR and ES among all fours, which shows that it might have the largest potential losses and the downside risk at 5% of the significance level. The data shows that Walmart will have 5% of probability to fall in value by more than $6.29 over a 21 days period. Another stock, Pfizer, has a negative VaR and ES, or the smallest among all fours. This means that it is 95% confident that the price won't lose more than $3.40 during the given period.



Companies	VaR (1 month)	Expected Shortfall (1 month)
Microsoft	1.61	1.78
Walmart	6.29	6.84
Pfizer	-3.4	-2.8
S&P 500	2.59	2.63

Figure 13

5. Conclusion and Recommendations

In this report, we applied machine learning models and time series forecasting models in stock price prediction. We compare the accuracy score of each model and select the ARIMA model, which has the highest accuracy score as the best model for forecasting. By applying the ARIMA model with data on four stock prices, we predict that in the next one month, the stock price of Walmart, Microsoft, Pfizer, and S&P 500 will increase. However, there are risks associated with this. Based on our VaR and ES calculation, we observe that Walmart has the greatest value of risk to loss, whereas Pfizer has the smallest value of risk to loss at a 5% confidence level.

By viewing all the performance and predictions of the stocks, we highly recommend that investors should invest in Pfizer in the next month. After one month, the stock price will have a slight rise from $53.74 to $55.56, and investors may have a chance to earn about $1.83 per share. We also have 95% confidence that by holding Pfizer for over one month, investors' losses will not go beyond $3.40 per share.

We highly do not recommend investing in Walmart in the next month. We predict Walmart's stock value will also have a slight rise from $140.65 to $144.55, which is about a $3.9 increase per share in value, but investors may have a 5% chance of losing in value by more than $6.29 per share within one month.

Reference

Aydin, E.S., Yucel, O., & Sadikoglu, H. (2018). Exergetic, Energetic and Environmental Dimensions. Chapter 2.6 - Numerical Investigation of Fixed-Bed Downdraft Woody Biomass Gasification. Academic Press, 323-339. doi: 10.1016/B978-0-12-813734-5.00018-4.

(https://www.sciencedirect.com/science/article/pii/B9780128137345000184)

Basle Committee on Banking Supervision (1996) Overview of the Amendment to the Capital Accord to Incorporate Market Risks. Bank for International Settlements.

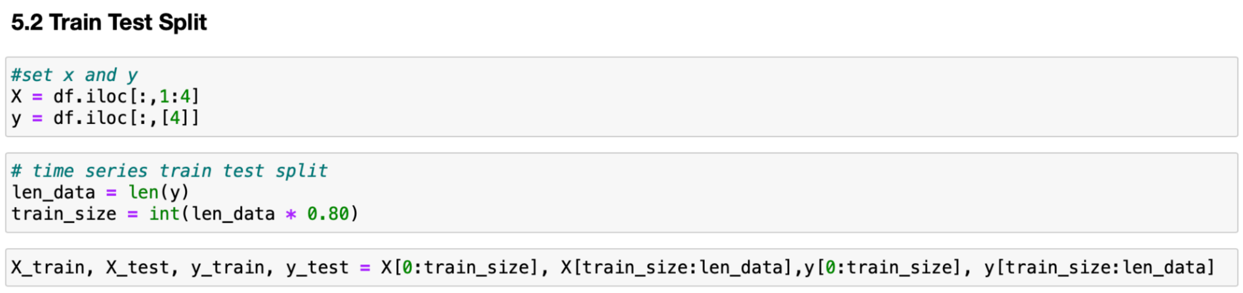
Danielsson, J. (2011). Financial risk forecasting: The theory and practice of forecasting market risk with implementation in R and Matlab (Vol. 588). John Wiley & Sons.

Leavens, D.H. (1945). Diversification of Investments. Trusts and Estates, 80, 469-473.

Morgan, J.P. (1996). Riskmetrics: Technical Document. Morgan Guaranty Trust Company of New York.

Vasileiou, E. (2017). Value at Risk (VaR) Historical Approach: Could It Be More Historical and Representative of the Real Financial Risk Environment?. Theoretical Economics Letters, 7, 951-974. doi: 10.4236/tel.2017.74065.

Appendix - Code 1



5.2 Train Test Split

```
#set x and y
X = df.iloc[:,1:4]
y = df.iloc[:,[4]]
```

```
# time series train test split
len_data = len(y)
train_size = int(len_data * 0.80)
```

```
X_train, X_test, y_train, y_test = X[0:train_size], X[train_size:len_data],y[0:train_size], y[train_size:len_data]
```

Code 1

Appendix - Code 2

5.3.1 Lasso

```python
alpha_lasso = np.arange(start=0.0001, stop=0.01, step=0.0001)
length_lasso1 = len(alpha_lasso)
lasso_trainScore1 = []
lasso_testScore1 = []

for i in range(length_lasso1):
    lasso = Lasso(alpha=alpha_lasso[i])
    lasso.fit(X_train, y_train)
    #store train accuracy
    lasso_trainScore1.append(lasso.score(X_train, y_train)
    #store test accuracy
    lasso_testScore1.append(lasso.score(X_test, y_test))

print("Lasso Regression: ")
print("Full sample R-squared:", lasso.score(X, y))
print(" ")

dataframe_lasso = {'Alpha': alpha_lasso,
                   'Train Set Score': lasso_trainScore1,
                   'Test Set Score': lasso_testScore1}

table_lasso = pd.DataFrame(dataframe_lasso)
table_lasso.index += 1

table_lasso = table_lasso.sort_values("Test Set Score", ascending=False)
display(table_lasso)
```

```
-------------------------------
Lasso Regression:
-------------------------------
Full sample R-squared: -5.2690468363092435e-05
```

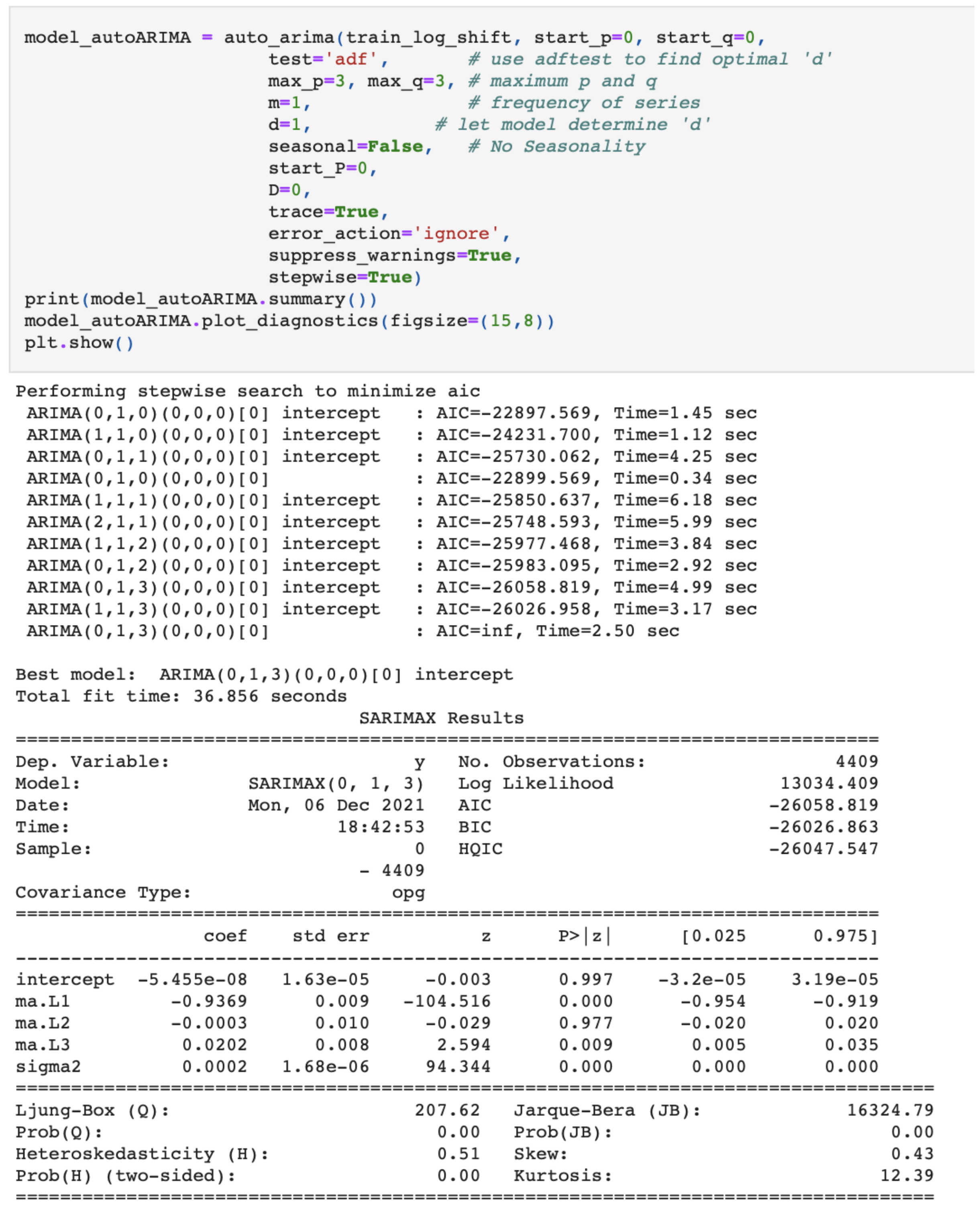
	Alpha	Train Set Score	Test Set Score
1	0.0001	0.637062	0.705361
2	0.0002	0.636704	0.703712
3	0.0003	0.636107	0.701851
4	0.0004	0.635272	0.699777
5	0.0005	0.634199	0.697490
...	...	...	...
85	0.0085	0.000000	-0.001238
84	0.0084	0.000000	-0.001238
83	0.0083	0.000000	-0.001238
82	0.0082	0.000000	-0.001238
99	0.0099	0.000000	-0.001238

99 rows × 3 columns

Code 2

Appendix - Code 3

6.1.4 Run Auto ARIMA find the best model

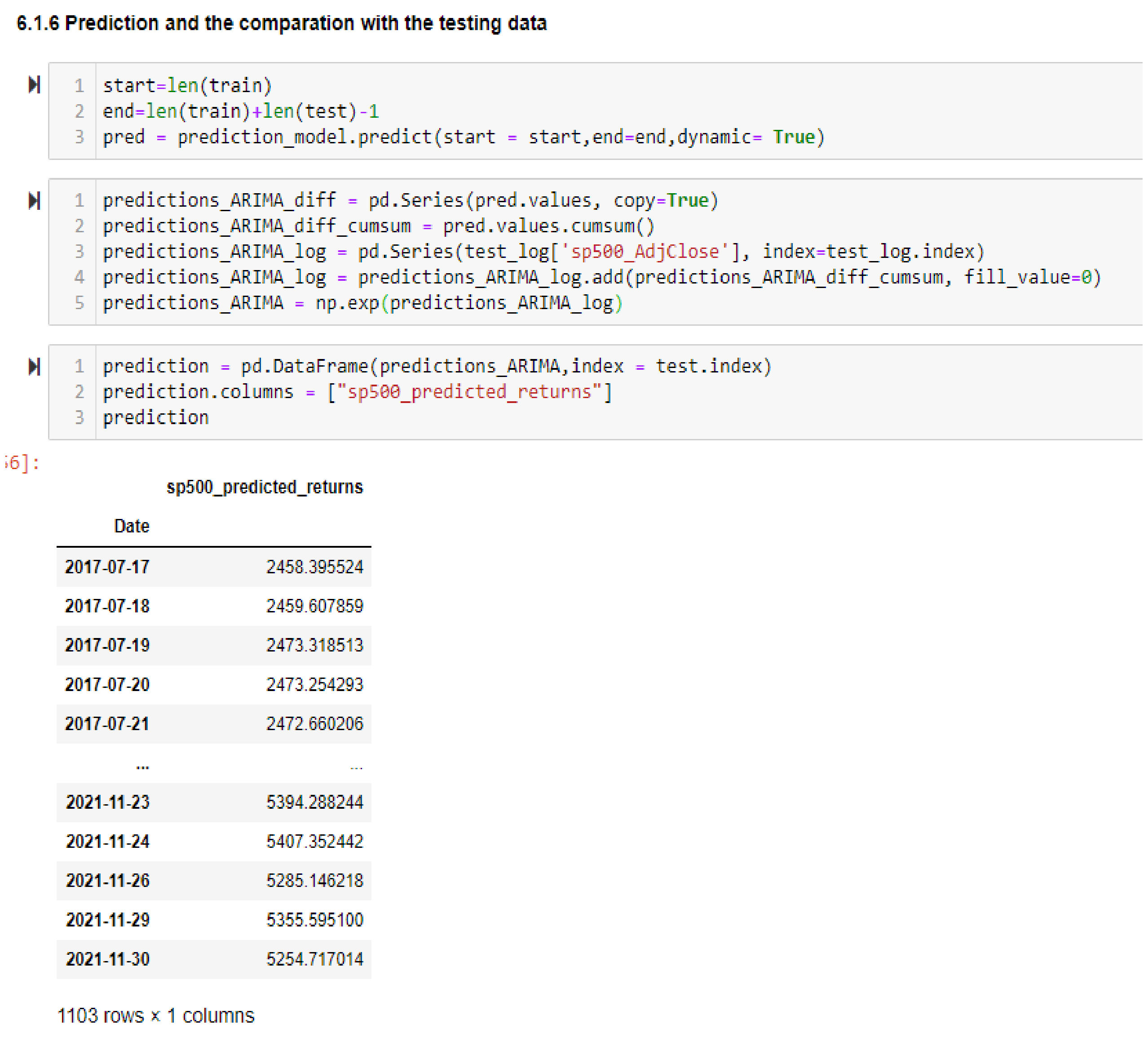


```
model_autoARIMA = auto_arima(train_log_shift, start_p=0, start_q=0,
                      test='adf',       # use adftest to find optimal 'd'
                      max_p=3, max_q=3, # maximum p and q
                      m=1,              # frequency of series
                      d=1,           # let model determine 'd'
                      seasonal=False,   # No Seasonality
                      start_P=0,
                      D=0,
                      trace=True,
                      error_action='ignore',
                      suppress_warnings=True,
                      stepwise=True)
print(model_autoARIMA.summary())
model_autoARIMA.plot_diagnostics(figsize=(15,8))
plt.show()
```

```
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=-22897.569, Time=1.45 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=-24231.700, Time=1.12 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=-25730.062, Time=4.25 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=-22899.569, Time=0.34 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=-25850.637, Time=6.18 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=-25748.593, Time=5.99 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=-25977.468, Time=3.84 sec
 ARIMA(0,1,2)(0,0,0)[0] intercept   : AIC=-25983.095, Time=2.92 sec
 ARIMA(0,1,3)(0,0,0)[0] intercept   : AIC=-26058.819, Time=4.99 sec
 ARIMA(1,1,3)(0,0,0)[0] intercept   : AIC=-26026.958, Time=3.17 sec
 ARIMA(0,1,3)(0,0,0)[0]             : AIC=inf, Time=2.50 sec

Best model:  ARIMA(0,1,3)(0,0,0)[0] intercept
Total fit time: 36.856 seconds
                               SARIMAX Results                                
==============================================================================
Dep. Variable:                      y   No. Observations:                 4409
Model:               SARIMAX(0, 1, 3)   Log Likelihood               13034.409
Date:                Mon, 06 Dec 2021   AIC                         -26058.819
Time:                        18:42:53   BIC                         -26026.863
Sample:                             0   HQIC                        -26047.547
                               - 4409                                         
Covariance Type:                  opg                                         
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept  -5.455e-08   1.63e-05     -0.003      0.997    -3.2e-05    3.19e-05
ma.L1         -0.9369      0.009   -104.516      0.000      -0.954      -0.919
ma.L2         -0.0003      0.010     -0.029      0.977      -0.020       0.020
ma.L3          0.0202      0.008      2.594      0.009       0.005       0.035
sigma2         0.0002   1.68e-06     94.344      0.000       0.000       0.000
===================================================================================
Ljung-Box (Q):                      207.62   Jarque-Bera (JB):              16324.79
Prob(Q):                              0.00   Prob(JB):                          0.00
Heteroskedasticity (H):               0.51   Skew:                              0.43
Prob(H) (two-sided):                  0.00   Kurtosis:                         12.39
===================================================================================
```

Code 3

Appendix - Code 4



6.1.6 Prediction and the comparation with the testing data

```python
start=len(train)
end=len(train)+len(test)-1
pred = prediction_model.predict(start = start,end=end,dynamic= True)
```

```python
predictions_ARIMA_diff = pd.Series(pred.values, copy=True)
predictions_ARIMA_diff_cumsum = pred.values.cumsum()
predictions_ARIMA_log = pd.Series(test_log['sp500_AdjClose'], index=test_log.index)
predictions_ARIMA_log = predictions_ARIMA_log.add(predictions_ARIMA_diff_cumsum, fill_value=0)
predictions_ARIMA = np.exp(predictions_ARIMA_log)
```

```python
prediction = pd.DataFrame(predictions_ARIMA,index = test.index)
prediction.columns = ["sp500_predicted_returns"]
prediction
```

6]:

	sp500_predicted_returns
Date	
2017-07-17	2458.395524
2017-07-18	2459.607859
2017-07-19	2473.318513
2017-07-20	2473.254293
2017-07-21	2472.660206
...	...
2021-11-23	5394.288244
2021-11-24	5407.352442
2021-11-26	5285.146218
2021-11-29	5355.595100
2021-11-30	5254.717014

1103 rows × 1 columns

Code 4

Appendix - Code 5

7.1 S&P 500

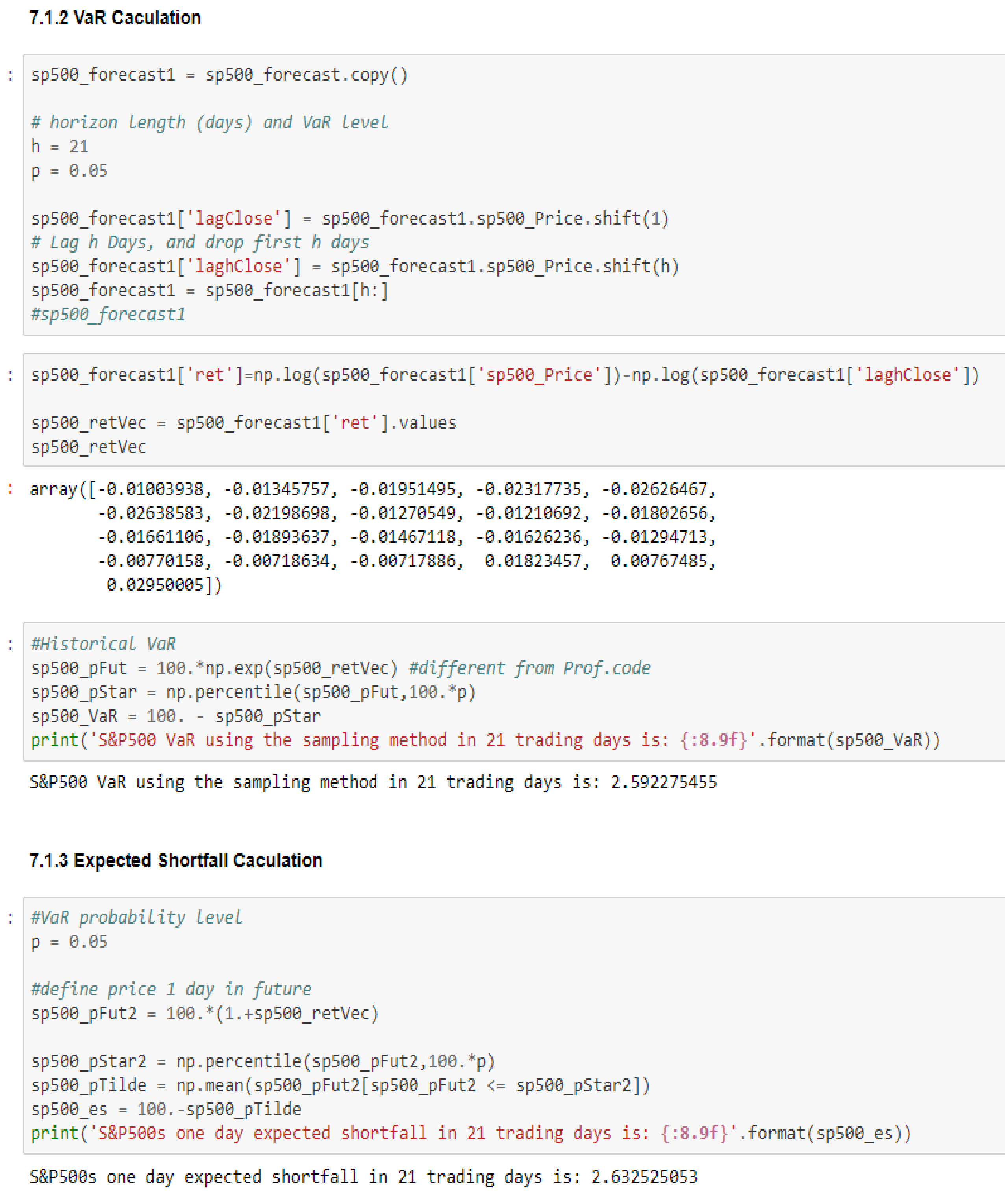
7.1.1 reform a new dataset: 21 days from historical price + 21 future price

```
start_date = '2021-11-01'
end_date = '2021-11-30'

#Filter data between two dates in historical data
last21 = y.loc[(y.index >= start_date) & (y.index <= end_date)]
#Remove time portion of DateTime index
last21.index = last21.index.date
#last21
one_month_prediction.columns = last21.columns
sp500_forecast = last21.append(one_month_prediction)
#OR sp500_forecast= pd.concat([last21, one_month_prediction], axis=0)
sp500_forecast = sp500_forecast.rename(columns={"sp500_AdjClose": "sp500_Price"})

# Display
sp500_forecast
```

	sp500_Price
2021-11-01	4613.669922
2021-11-02	4630.649902
2021-11-03	4660.569824
2021-11-04	4680.060059
2021-11-05	4697.529785
2021-11-08	4701.700195
2021-11-09	4685.250000
2021-11-10	4646.709961
2021-11-11	4649.270020
2021-11-12	4682.850098
2021-11-15	4682.799805
2021-11-16	4700.899902
2021-11-17	4688.669922
2021-11-18	4704.540039
2021-11-19	4697.959961
2021-11-22	4682.939941
2021-11-23	4690.700195
2021-11-24	4701.459961
2021-11-26	4594.620117

Code 5

Appendix - Code 6



7.1.2 VaR Caculation

```
sp500_forecast1 = sp500_forecast.copy()

# horizon length (days) and VaR level
h = 21
p = 0.05

sp500_forecast1['lagClose'] = sp500_forecast1.sp500_Price.shift(1)
# Lag h Days, and drop first h days
sp500_forecast1['laghClose'] = sp500_forecast1.sp500_Price.shift(h)
sp500_forecast1 = sp500_forecast1[h:]
#sp500_forecast1
```

```
sp500_forecast1['ret']=np.log(sp500_forecast1['sp500_Price'])-np.log(sp500_forecast1['laghClose'])

sp500_retVec = sp500_forecast1['ret'].values
sp500_retVec
```

```
array([-0.01003938, -0.01345757, -0.01951495, -0.02317735, -0.02626467,
       -0.02638583, -0.02198698, -0.01270549, -0.01210692, -0.01802656,
       -0.01661106, -0.01893637, -0.01467118, -0.01626236, -0.01294713,
       -0.00770158, -0.00718634, -0.00717886,  0.01823457,  0.00767485,
        0.02950005])
```

```
#Historical VaR
sp500_pFut = 100.*np.exp(sp500_retVec) #different from Prof.code
sp500_pStar = np.percentile(sp500_pFut,100.*p)
sp500_VaR = 100. - sp500_pStar
print('S&P500 VaR using the sampling method in 21 trading days is: {:8.9f}'.format(sp500_VaR))
```

```
S&P500 VaR using the sampling method in 21 trading days is: 2.592275455
```

7.1.3 Expected Shortfall Caculation

```
#VaR probability level
p = 0.05

#define price 1 day in future
sp500_pFut2 = 100.*(1.+sp500_retVec)

sp500_pStar2 = np.percentile(sp500_pFut2,100.*p)
sp500_pTilde = np.mean(sp500_pFut2[sp500_pFut2 <= sp500_pStar2])
sp500_es = 100.-sp500_pTilde
print('S&P500s one day expected shortfall in 21 trading days is: {:8.9f}'.format(sp500_es))
```

```
S&P500s one day expected shortfall in 21 trading days is: 2.632525053
```

Code 6