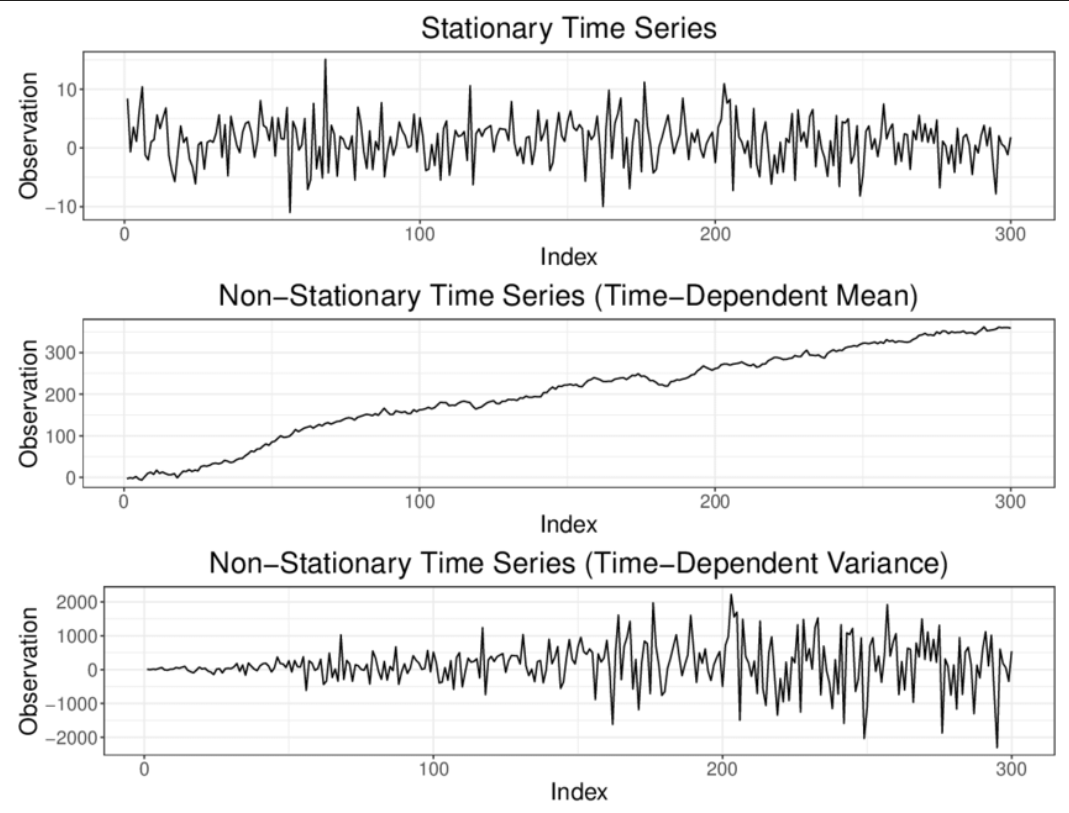
**WIDS**

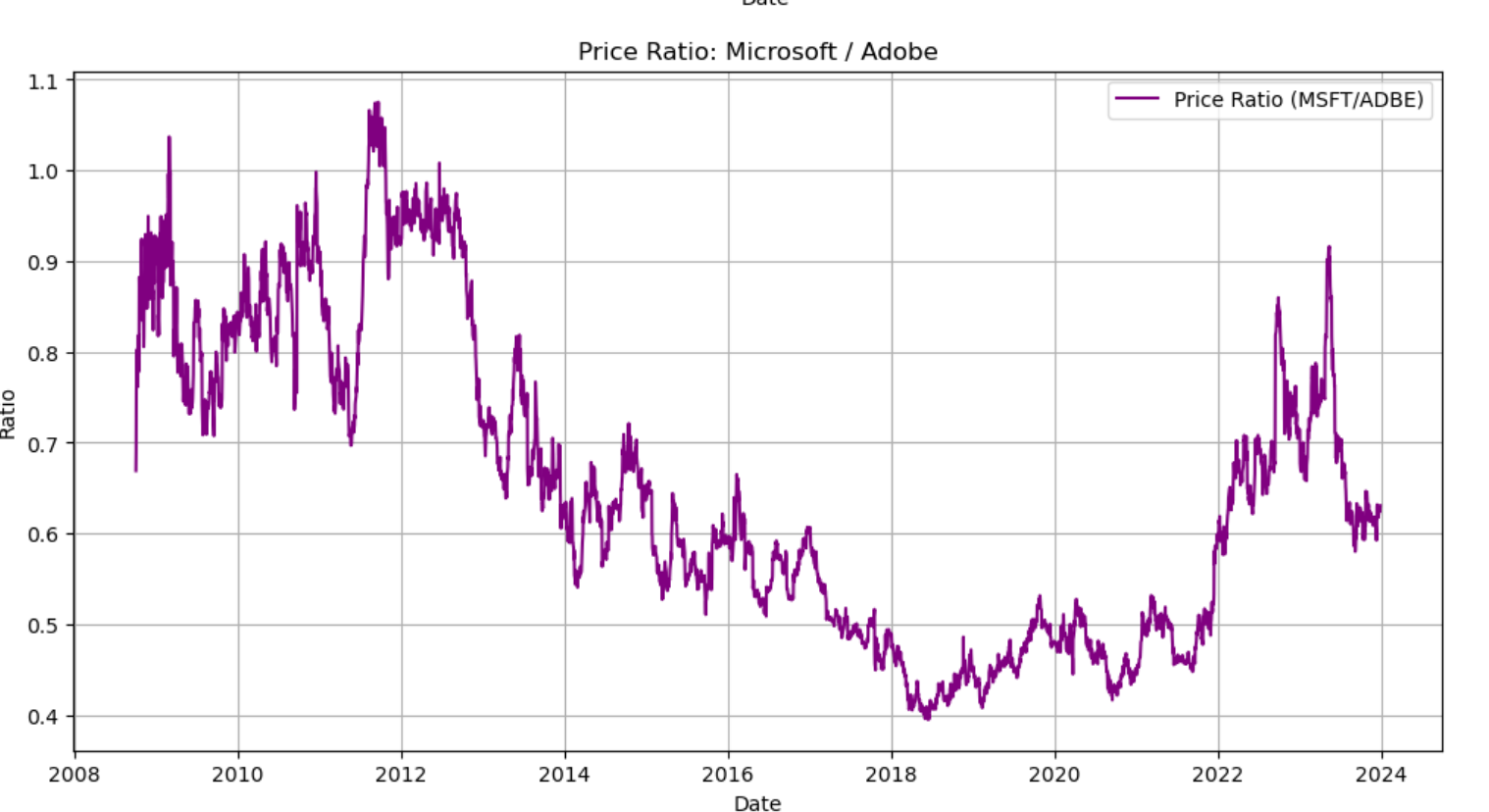
**Week-1 Documentation**

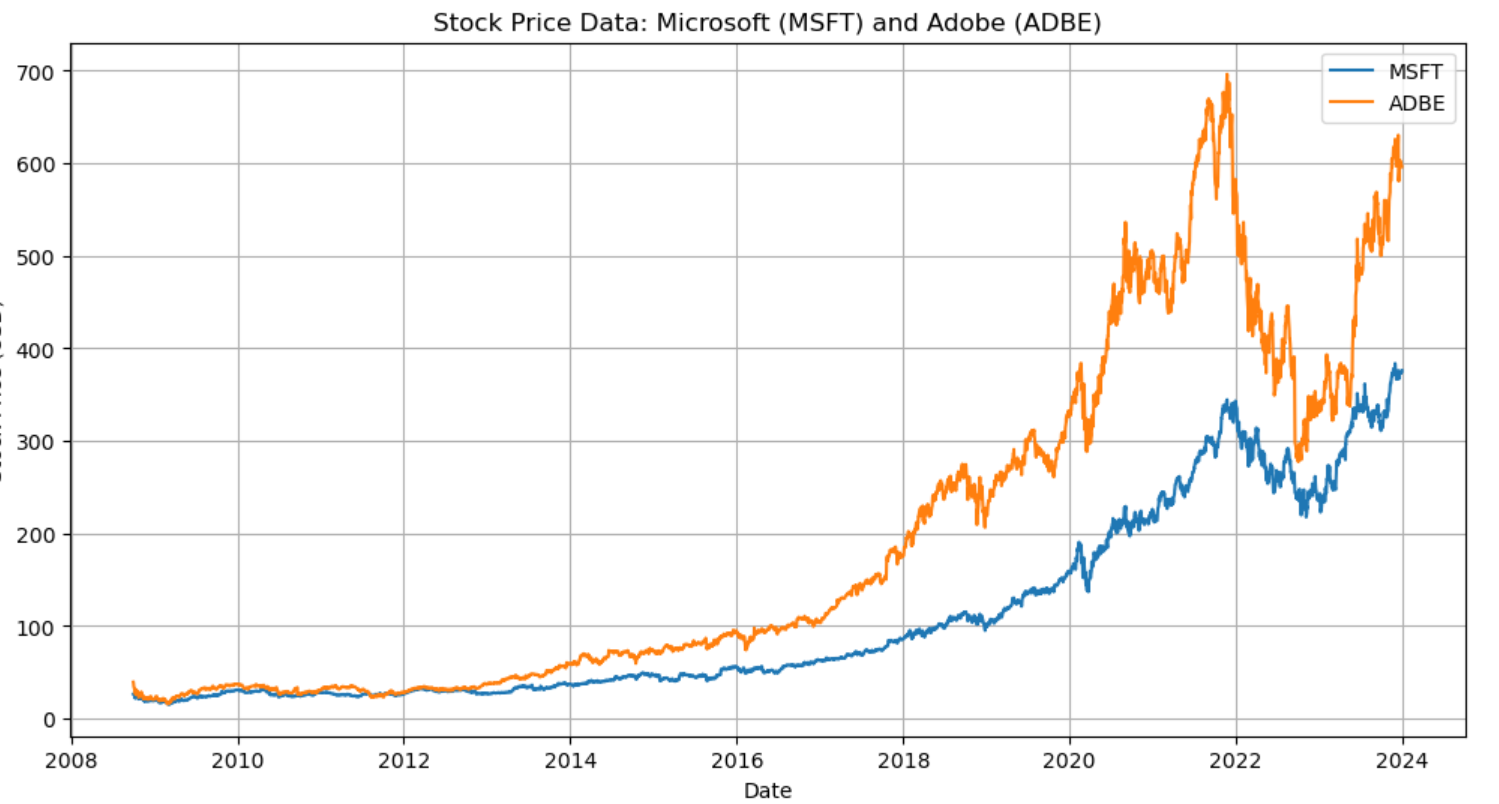
**About:** The following document includes all of my major learnings from the material that was provided in the first week of this project. It includes both, things I learnt, along with some queries that I believe will be addressed in the coming weeks as more material will be provided.

So, firstly about stationarity. Whenever we graph an observation (any variable) over time as a “time-series” (which is nothing but a collection of data points that are recorded at regular intervals over time), we consider it to be “stationary” or possess the characteristic of “stationarity” if the mean of the data and standard deviation remains constant over a period of time, and there is no seasonality (periodicity) in the data.

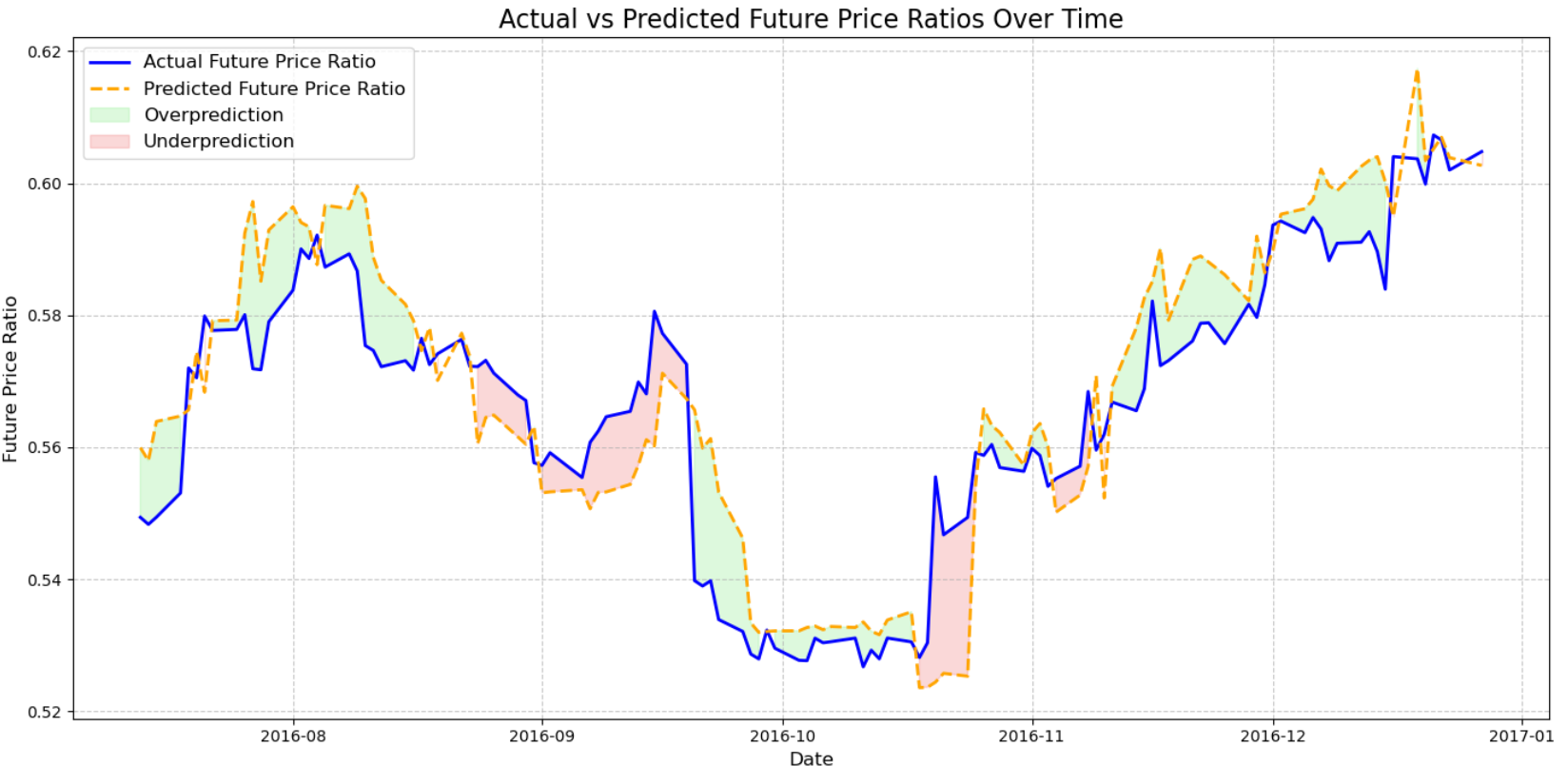
So, even though visually it is very easy to estimate a time series as stationary or not, there are statistical ways through which we can back our observations with mathematical results and prove stationarity. One of those tests is the augmented Dickey Fuller test (ADF).

Before that, there is something called a null Hypothesis (H0), which basically is that “two financial elements are not corelated/ cointegrated”, and the alternative hypothesis (H1), which states that they are indeed correlated/cointegrated. We use different statistical measurements like z-scores, t-statistics and the p-value from ADF test to reject the null hypothesis. Basically, if the null hypothesis gets rejected, we can say that the two elements under study share some correlation. Mostly, (atleast what we did here) to measure correlation between two stocks, we take either their spread or their price ratio, and check stationarity of its time series. If p-value of the ADF test comes less than 0.05, we can say that the time series (price ratio as taken ahead) is stationary and the two stocks aree somewhat correlated. We can exploit this to our benefit.

Now, coming to the stocks MSFT (Microsoft) and ADBE (Adobe) that we were supposed to study, firstly the stock prices, and consequent price ratio curves look somewhat like this.

The p-value for the price ratio time series comes out to be around 0.3, so we can only take a segment of this and test that for stationarity, as we desire a p-value < 0.05. So, we choose an interval of 2013-2017 accordingly, with p-value of 0.03, which is satisfactory enough to proceed (maybe).

Next, I used the mentioned features, MAVG5, MAVG20, MAVG60, STD20, zscore20\_5, ROC to predict the future price ratio, based upon the training data. Now, we get pretty decent results upon using train\_test\_split, and setting the test data as 20% of our input dataframe. Random Forest Classifier gives a 89% prediction accuracy (between the price ratio predicted and actual Price ratios). It gives us a return of 4756.07$ which is pretty decent, however the issue lies in the function ‘train\_test\_split’, which uses both past and future values for prediction, and proceeds to randomly pick the 20% test data points, from past/ future (which should only be future values), to apply the model on. This is not viable, as in reality we can have access to only the past values, and using them predict the future ratios. Upon using this, the obtained returns were bizarre and terrible for the least.



This is the curve, which shows the actual v/s predicted price ratios, estimated using RandomForestRegressor. The model’s accuracy jumps to an extremely low 56.41%, than the previous 89% because the factor which removes uncertainity and input feature dependency from randomforestregressor gets removed.

So, what I planned to implement (and plan to try again in subsequent weeks) was, predict the future price ratios, by using some supervised training model (using the above mentioned features to train the model), and check its accuracy with the actual price ratio. Whichever algorithm provides a good accuracy measure, we proceed to use it to calculate the zscore ratios, to determine the “buy” and “sell” call (using the zscr>-1 and zscr<1, mentioned in the reference) and apply these calls on the actual price ratios, to see if I make a profit or not. Not that I am fully satisfied with the code I have used, but I can surely say the results I have got, can be improved upon by a better, well constructed code, but some things need to be rethought and applied properly. Will explore further on the new content shared, and work on the same in next week’s tasks.