FIN580 Midterm

Utsav Popat, Srishti Singla, Chris Sun

April 22, 2020

Abstract

With the unprecedented severity of Covid-19, we are facing a major structural change with global spillovers and unpredictable economic and social consequences. The goal of this mid-term project is to forecast daily volatilities of major stock indexes after the COVID-19 outbreak and provide risk measures than can help portfolio managers to make investment decisions during this turbulent period. We use data from the Realized Library of the Oxford-Man Institute of Quantitative Finance for 31 different global indices, and also assess some alternative datasets such as Google word trends to model this structural break. Due to potential cross-correlation and lead/lag effects among different markets, we decide to cluster indices that depend on similar co-variates together. The reports discusses in detail a range of linear and non-linear models for predicting daily Realized Variances, which are validated over the time period starting February 1st, 2020. We propose using a Simple Moving Average Model as it significantly outperforms the rest.

Keywords: Covid-19, Realized Variance, Value-at-Risk, Random Walk, Forecasting

Contents

1 Introduction

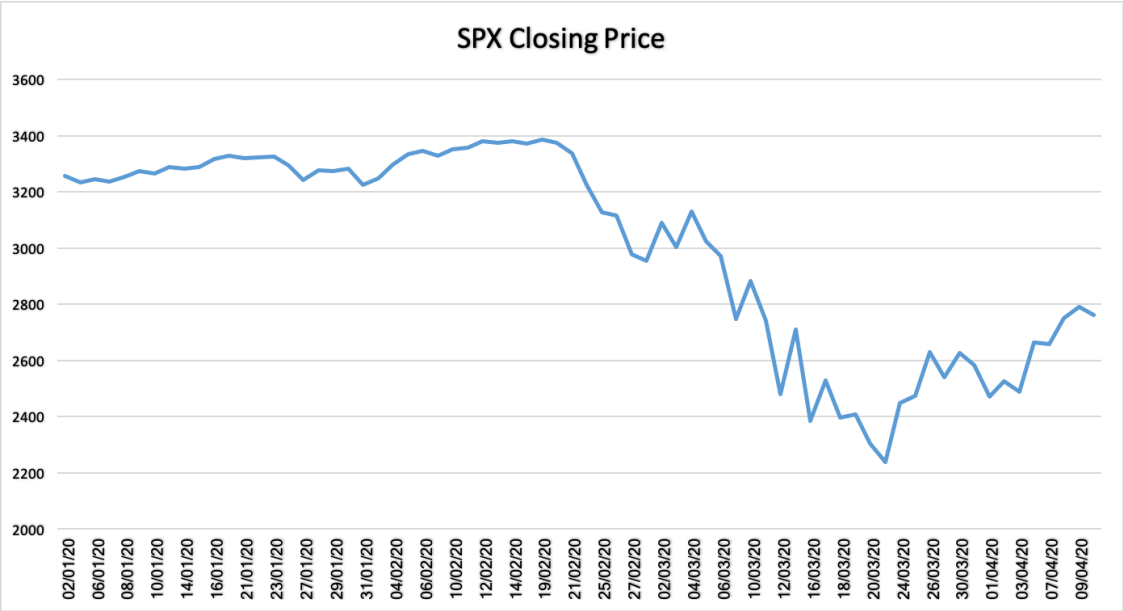


Figure 1.1: SPX Closing Price starting Jan, 2020

This is a graph of the S&P index over just the last 4 months. The relative calm and slow upward trend made way for madness and a period of constant highs and lows as financial markets bore the brunt of Covid-19. The S&P500 entered a bear market on March 12, 2020 (for the first time in 11 years), only to exit it in 15 days - marking the shortest bear market in history. In such times where it seems that the market fluctuations each day are tail events, portfolio managers would depend heavily on risk metrics such as Value-at-Risk to check their potential losses in the worst-case scenarios. However, computing VaR requires an underlying distribution for the returns. A reasonable assumption to make is that $r_t|\text{past information} = h_t\epsilon_t$, with $\epsilon_t \sim N(0,1)$, and $h_t = \sqrt{RV_t}$, the realized variance for day t. Then, $Var_{\alpha\%,t+1} = c_\alpha\sqrt{\widehat{RV}_{t+1}}$, where c_α is the $\alpha\%$-percentile of a standard normal distribution. This imposes a small restriction in that for $\alpha < 50\%$, $VaR_{\alpha,t+1}$ is always negative as $c_\alpha < 0$ and $\widehat{RV}_{t+1} > 0$. Similarly, $VaR_{\alpha,t+1}$ is always positive when $\alpha > 50\%$. In practice, VaR can be positive or negative (though one prefers that it be negative). This might be an issue later on when we try to set VaR values for extremely turbulent times, but are restricted to always predicting a negative VaR, i.e. there would be no loss.

Therefore, in this project, we will focus on predicting realized variances from historical data. We source the historical data from the Realized Library of the Oxford-Man Institute of Quantitative Finance for 31 different global indices. Note that the dataset from the Realized Library is incomplete for the S&P 500 and IBEX 35 indices. Hence, we will consider daily Realized Ranges (RR_t) instead of daily Realized Variances for these stocks, where

$$RR_t = \frac{(\ln(\text{Max Price}) - \ln(\text{Min Price}))^2}{4\ln 2}$$

For the other 29 indices, we will consider the five-minute realized variances.

Rest of the report is split into following sections - Data Preparation and Visualization, Forecasting, Results, and Conclusion.

2 Data Preparation and Visualization

The Realized Variances Oxford-Man dataset starts from January 3, 2000 and is updated daily. For each of the 31 indices - marked under the column "Symbol", we have three daily entries - open price, close price, and five-minute Realized Variance.

2.1 Dates

We first focus on the dates in the dataset. As a master record, we take the union of all the dates across which each index traded. We observe that a date is of two formats: "yyyy-mm-dd 00:00:00" and "yyyy-mm-dd 00:01:00". However, no date has both the formats present; hence, it is safe to ignore the timestamps altogether. Next, we confirm that for each index, there is at most one record per date. Therefore, we can create a grid of all the available index data with respect to the master record of dates. Now this will naturally create some missing values as there is no guarantee that the trading days were the same across all the indices. Nevertheless, we create the grid and fill in the missing data using imputation, as it will enable us to train models that will capture the global nature of the crisis and the correlation between some financial indices.

Here is a snapshot of first few rows and columns of the our grid:

```
> head(sample[, 1:7])
           Date open.AEX close.AEX       rv.AEX open.AORD close.AORD      rv.AORD
2625 2010-01-04   336.96    343.05 4.338067e-05    4877.7     4885.0 1.563637e-05
2626 2010-01-05   343.52    341.96 3.812023e-05    4899.8     4934.9 3.634501e-05
2627 2010-01-06   341.58    341.23 5.965597e-05    4943.7     4947.2 8.183277e-06
2628 2010-01-07   340.56    340.76 5.471112e-05    4948.2     4930.7 1.887266e-05
2629 2010-01-08   341.98    341.75 1.386734e-04    4931.4     4940.6 1.822273e-05
2630 2010-01-11   344.20    340.10 4.736969e-05    4950.7     4984.0 1.607547e-05
```

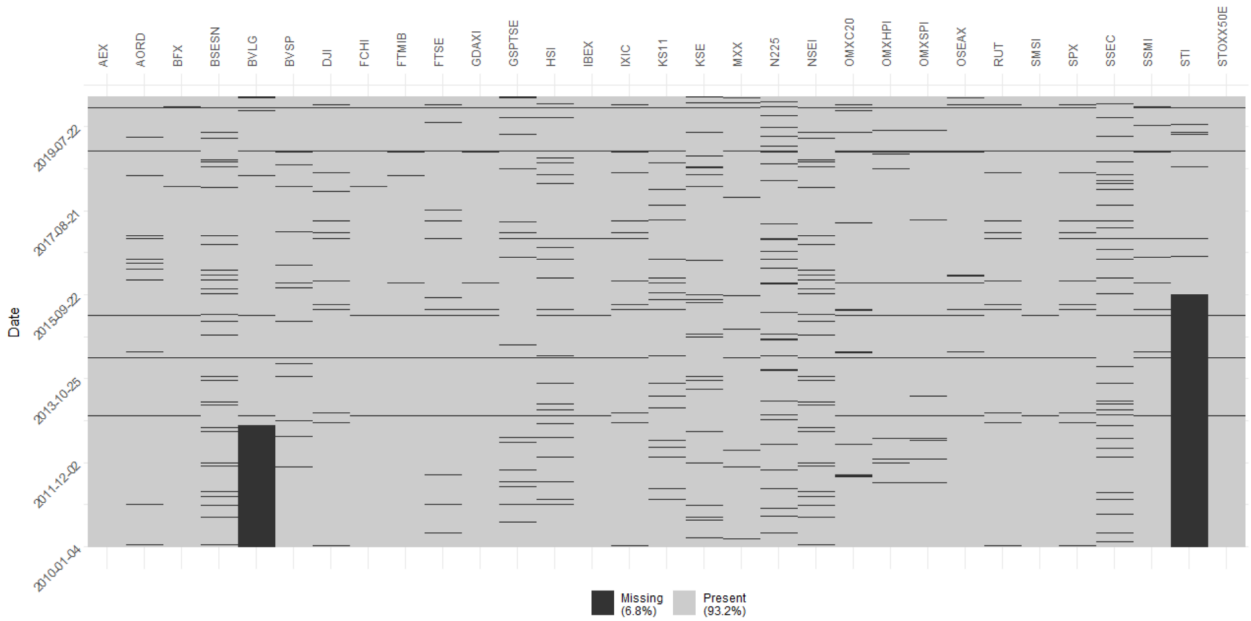
Figure 2.1: Head of our datatable starting Jan, 2010

2.2 Missing Data

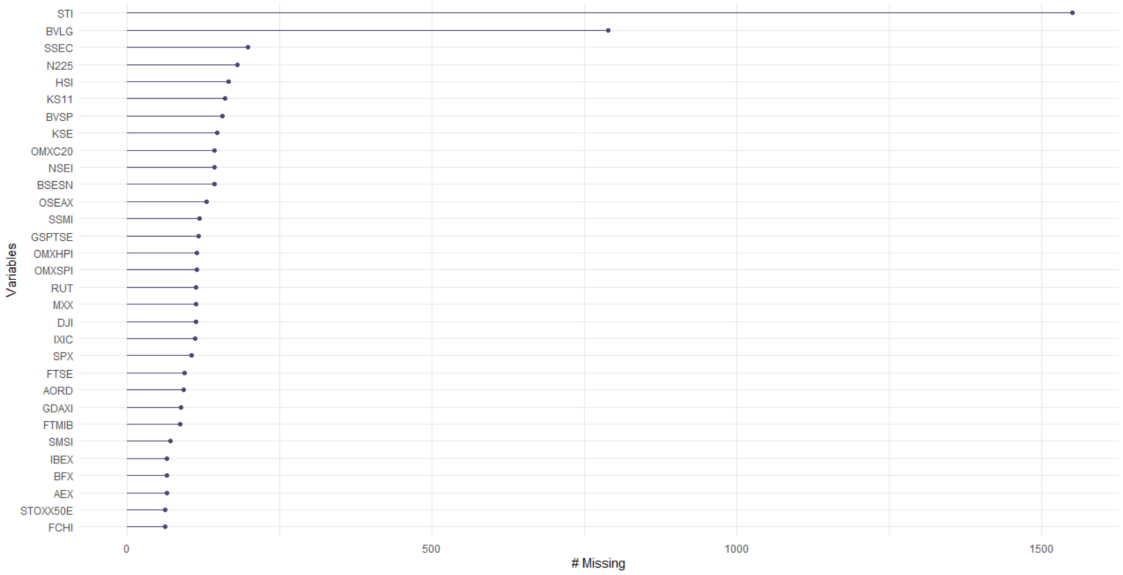
We plot a heat map all of the missing data in Figure 2.2. For simplicity, we present data only since the preceding bull run, starting on January 01, 2010. Over 6.7% of the data is missing. The presence of long horizontal lines reaffirms our statement above that some indices traded on days when most other indices were shut.

A major portion of the missing values appear right at the start of the graph for the indices BVLG and STI. While BVLG only started operations October 19, 2012 onwards, STI has been active since 1988. Perplexingly, the data from the Oxford-Man Institute of Finance simply does not have any data points for STI for the period January 2008 - September 2015. This conveys to us that for any analysis we do, we need to keep an eye on how this large gap of missing data is treated. We show the result of some imputation techniques below, and resolve this issue in Section 2.3. However, apart from these huge chunk of missing data for BVLG and STI, we find that the maximum length of continuous missing data is five days.

We explore three different ways of data imputation: constant, linear, and spline. We present visualizations for two different scenarios here - the imputed values for timeseries AORD (Figure 2.3) that has most of its data present, and the timeseries STI (Figure 2.4), which is missing a major portion of initial data. We zoom into the period from May 2015 to November 2016 (approximately 400 values) to better see the differences in imputations.



(a) Missing RV data for each index starting 2010

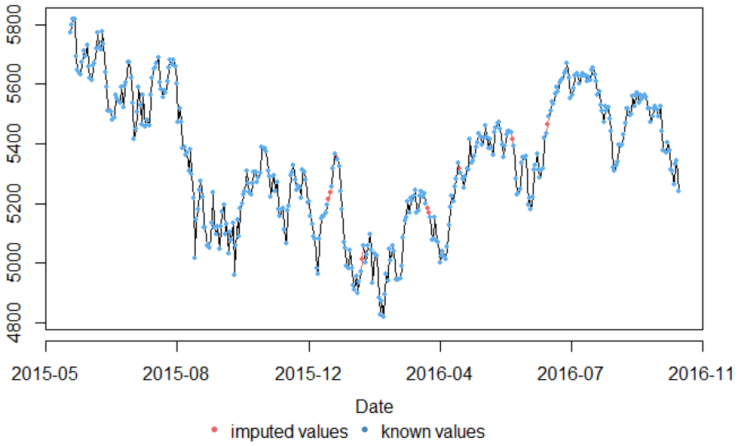


(b) Count of missing RV data for each index starting 2010

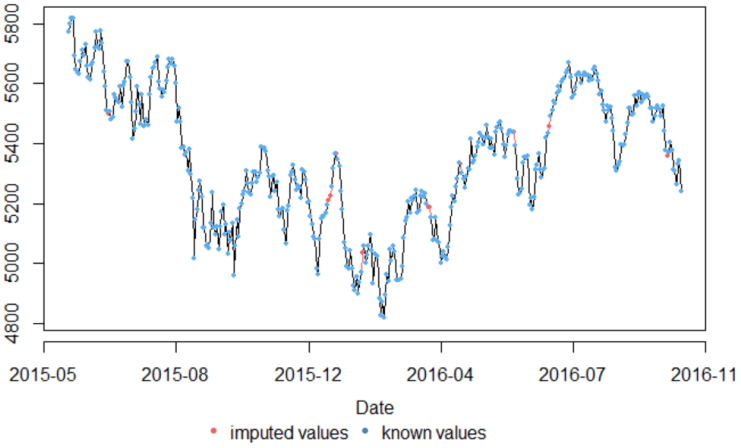
Figure 2.2: Summary of missing data in the dataset

As can be seen in the figures, both the linear and spline interpolation methods introduce an element of look-ahead bias - be it between two data points or right at the start of the data (as in Figure 2.4). In fact, spline interpolation performs extremely poorly for STI - probably as it tries to fit to the volatile data with a high degree polynomial (a lot of turning points are needed to capture the pattern in the data), which leads to highly erroneous results. Constant interpolation (forward-filling) seems to be our best choice to fill in the missing data values. The logic also holds empirically - we claim that most of the missing data is one-off (and not a contiguous block) due to differences in trading days; hence, it makes sense to assume that the index did not move at all on

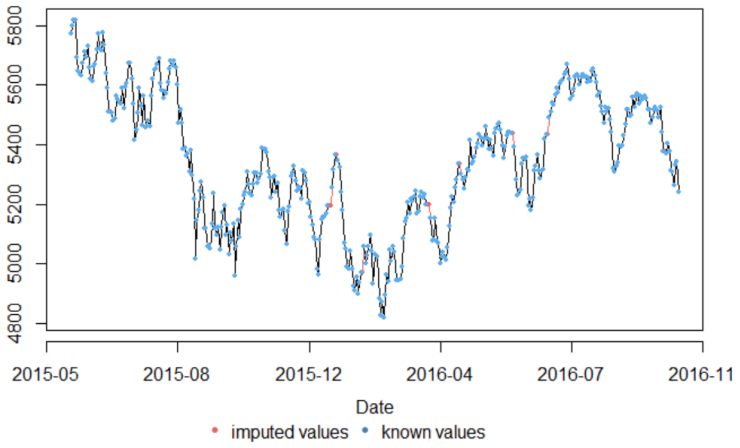
such a day.



(a) Interpolation using linear method

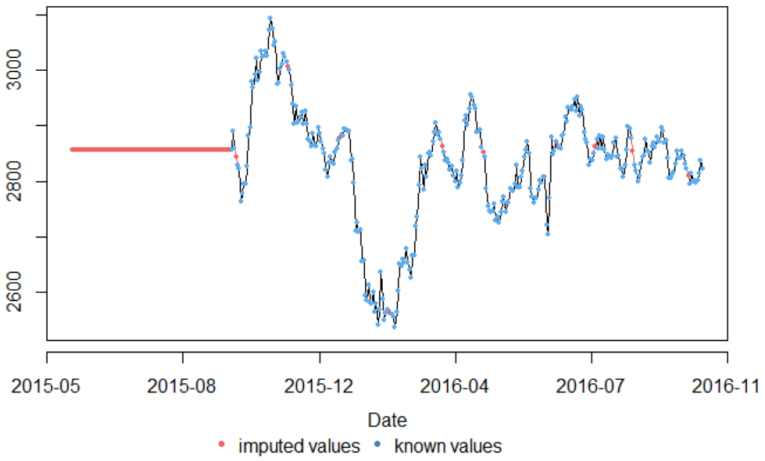


(b) Interpolation using spline method

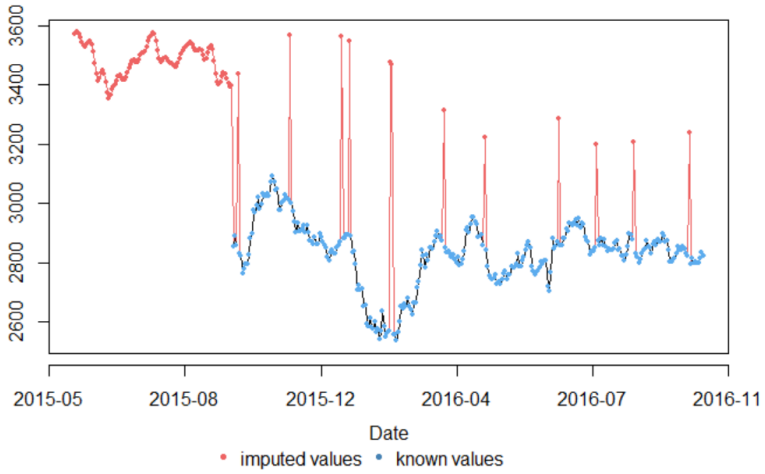


(c) Imputation using constant method

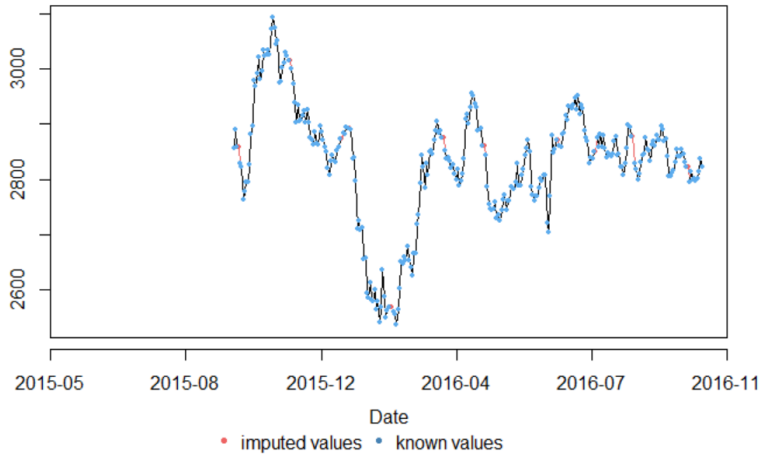
Figure 2.3: Visualisations of different imputation techniques for AORD



(a) Interpolation using linear method



(b) Interpolation using spline method



(c) Imputation using constant method

Figure 2.4: Visualisations of different imputation techniques for STI

2.3 Structural Break

As mentioned earlier, the past month has been unlike the previous months - or any crisis seen so far. There is a clear indication of a structural break, and we quantify this trend by looking at average correlations among indices. We look at the data starting 2006 and calculate the average rolling correlation across windows of length 500. Figure 2.5 shows the average correlation time series. The idea is that market correlations increase during a global downturn - due to intertwined nature of the global markets, the indices tend to spiral down together; however during good times, indices tend to reflect the national economic health and aren't as correlated. By using a long window of 500 trading days, we are able to capture significant structural breaks and ignore random noisy fluctuations. This belief is corroborated in Figure 2.5, where we see peaks in the average rolling correlation around the times of global recession (2008) and also for the flash crash (2013). Interestingly, average correlation shot up past 0.65 over the last three months - similar to the peak level around the 2008 financial crisis. This tells us that the indices are highly correlated over the period of interest (which might dictate what methods we can use), and that the current period can be read as a structural break from the normal trend. Hence, we need to be careful in choosing our dataset - we neither want to take too long a period and introduce noise, nor do we want to consider too short a period and lose out on some important historical context. To this effect, we choose to focus only on the timeseries starting July 02, 2019 onwards.

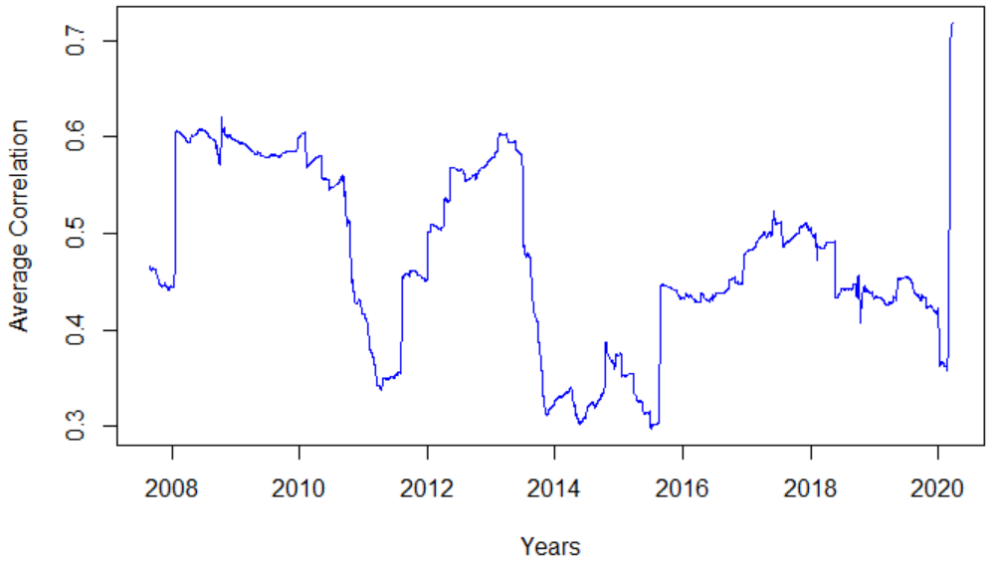


Figure 2.5: Rolling 500 days Average Correlation for constant imputed data

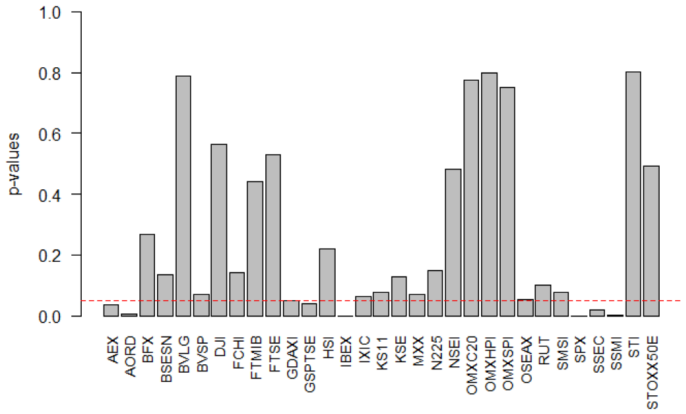
2.4 Normality Assumptions

In Section 1, we assumed that daily returns of a given index given the past information to have the following mode: $r_t|\text{past information} = h_t\epsilon_t$, with $\epsilon_t \sim N(0,1)$, and $h_t = \sqrt{RV_t}$, the realized variance for day t. It is important to test whether ϵ_t actually follows a standard normal distribution over the time period chosen by us.

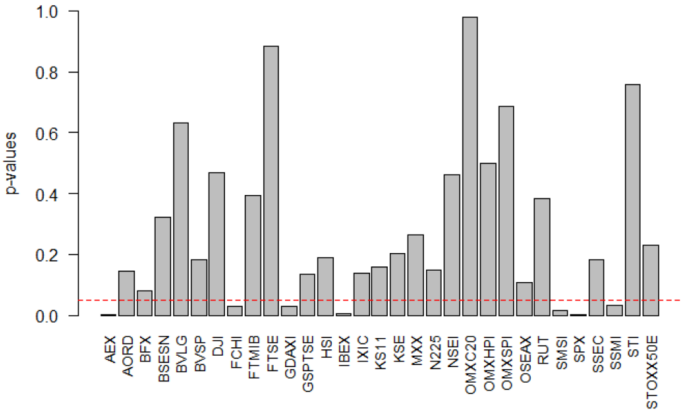
To check for normality for each index, we first compute daily standardized returns $\frac{\ln(\text{Close})-\ln(\text{Open})}{\sqrt{RV_t}}$, and run the Shapiro-Wilk Normality Test and the Jarque-Bera Test. Both the tests check for null hypothesis that the series is standard normal and return a p-value for the test. If p-value given by

the test is greater than 0.05 then we can not reject the normality assumption at 5% significance level.

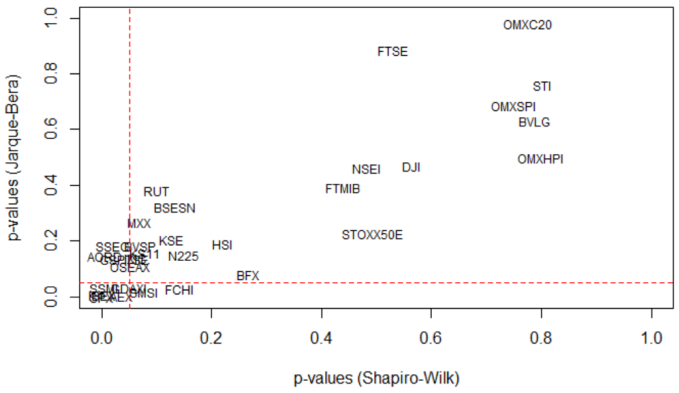
We find that for most of the indices normality assumption holds well. The null hypothesis is rejected for only 7 of the 31 indices by either test and only for 4 indices by both the tests, as seen in Table 2.1.



(a) Shapiro-Wilks Test



(b) Jarque-Bera Test



(c) A comparison of both tests

Figure 2.6: P-values for Normality Tests

Test Metric	Indices
Shapiro-Wilk Test	AEX, AORD, GSPTSE, IBEX, SPX, SSEC, SSMI
Jarque-Bera Test	AEX, FCHI, GDAXI, IBEX, SMSI, SPX, SSMI
Both	AEX, IBEX, SPX, SSMI

Table 2.1: Indices for which the null hypothesis was rejected

2.5 Variable Lag Choice

As mentioned in a paper, "Empirical Example: Forecasting Large Dimensional Realized Covariance Matrices" (discussed in class), we know that Realized Variances are highly persistent over time. So, it might be important to consider and test the presence of auto-correlation for each RV series. Figure 2.7 reaffirms our belief, with lags even upto 10-15 looking significant for some indices.

However, we simply cannot use as many lags. Given the initial date chosen, we have only 213 dates per index, from July 02, 2019 to April 21, 2020. As we would be looking at performing walk-through validation over windows of length 90 days, the number of covariates (31 at each lag) would greatly outnumber the number of data points in each window. However, not every covariate is going to be as important at every lag for a particular index - Nordic indices would be well linked with and have similar dependencies as other Nordic indices, than say with respect to MXX, the Mexican index. Hence, it would be more effective to cherry-pick which covariates and lags would be used to predict a particular index. However, we clearly cannot do this for each of the 31 indices - it would be better to group them up into clusters that depend on the same covariates, and create specific datasets per cluster.

To this extent, we consider all 31 realized variances as predictors and run 31 separate Adaptive LASSO models (in a rolling window fashion) to predict the next day's realized variance for each index. For each beta obtained from the models, we calculate the proportion of β assigned to each covariate, $\frac{|\beta|}{\|\beta\|_1}$, and average this quantity over all windows. Then, we use k-means clustering to find clusters of betas that depend upon the covariates in similar proportions. We consider this derived property instead of the correlation between the various realized variance series because we want to find clusters of indices that have similar dependencies - not indices that are correlated. This would allow us to fine tune the covariates without sacrificing convenience.

In Figure 2.8, we define score to be `(Total within-cluster sum of square distances) / (Average square distance between cluster means)`. In seeking the optimal number of centers, we want to lower the within-cluster sum of square distances and/or increase the average square distance between cluster means, which means we want to look for a lower score. Based on the elbow plot, 3, 4, and 5 all appear to reasonable center values. In combination with qualitative understanding of the breadth of indices involved and their geographical relations, we decided that having four clusters would be optimal in the tradeoff between specificity and convenience. We get the four clusters as mentioned in Table 2.2.

While most clusters fit our intuition well - multiple indices from group together, European countries split into two clusters, Southeast Asian countries group together - there are some interesting connections, such as Mexico and China forming a separate cluster on their own.

Unfortunately, it is extremely difficult to visualize the reasons behind the formation of these clusters, as the centroids are 31-dimensional vectors. Instead, we analyse our clusters in light of coronavirus related news by looking at Google trends for the word "Coronavirus" across different countries represented in our data. We plotted a correlation heat-map for the difference in the country-wise daily frequency starting January, 2020. The idea is that countries that are being affected by Covid-19 on a similar day-by-day basis might reflect similar uncertainties in the stock

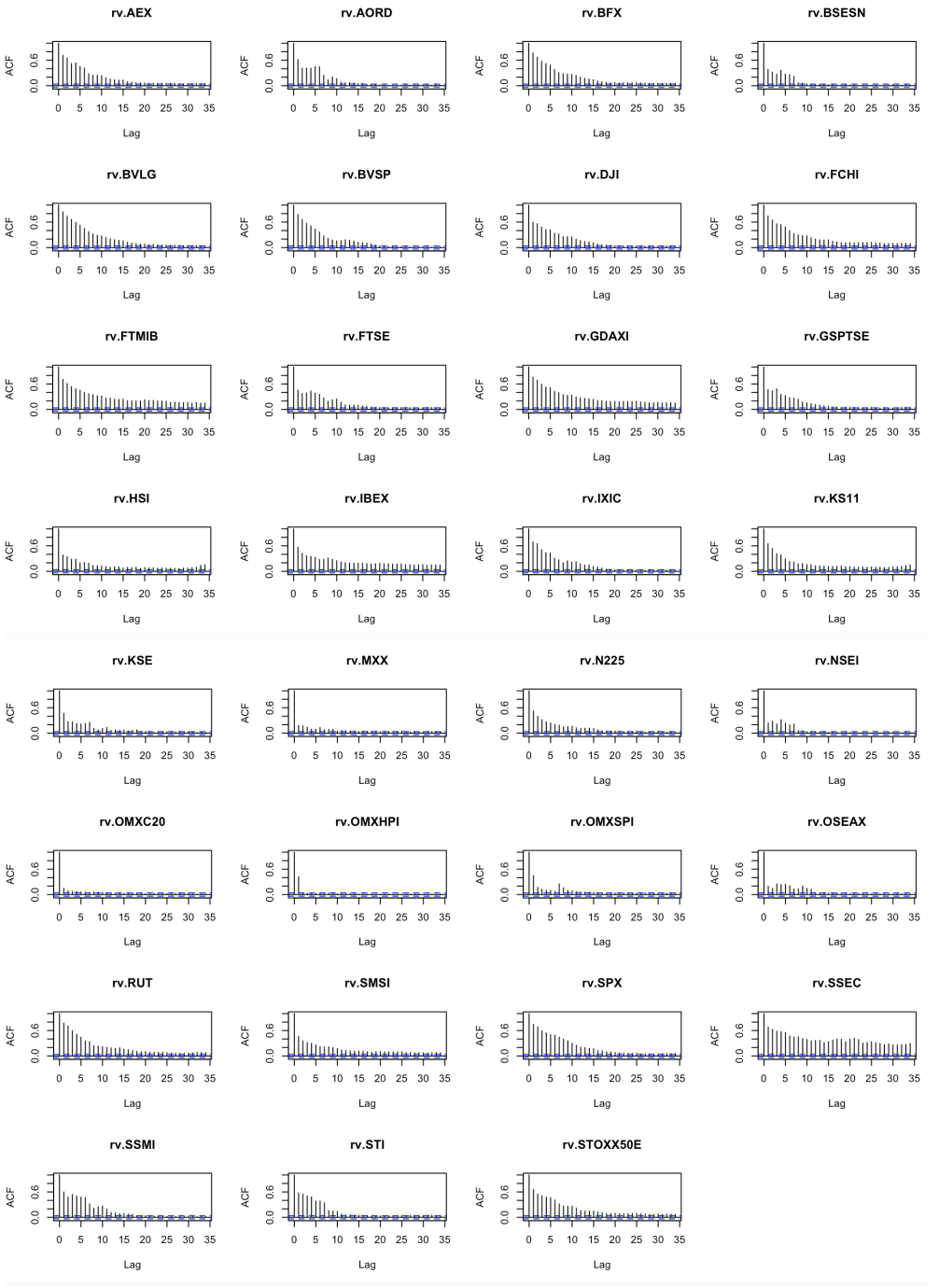


Figure 2.7: ACF plots for all the indices

market.

Figure 2.9 can be a good way to test the validity of our four clusters. The black lines separate the countries into the four clusters we obtained, and a deeper red color points out to higher correlation between the pair. We look at the four highlighted triangles along the diagonal, and there seems to be a significant inter-country correlation for almost all the clusters - suggesting that these clusters are a reasonable choice.

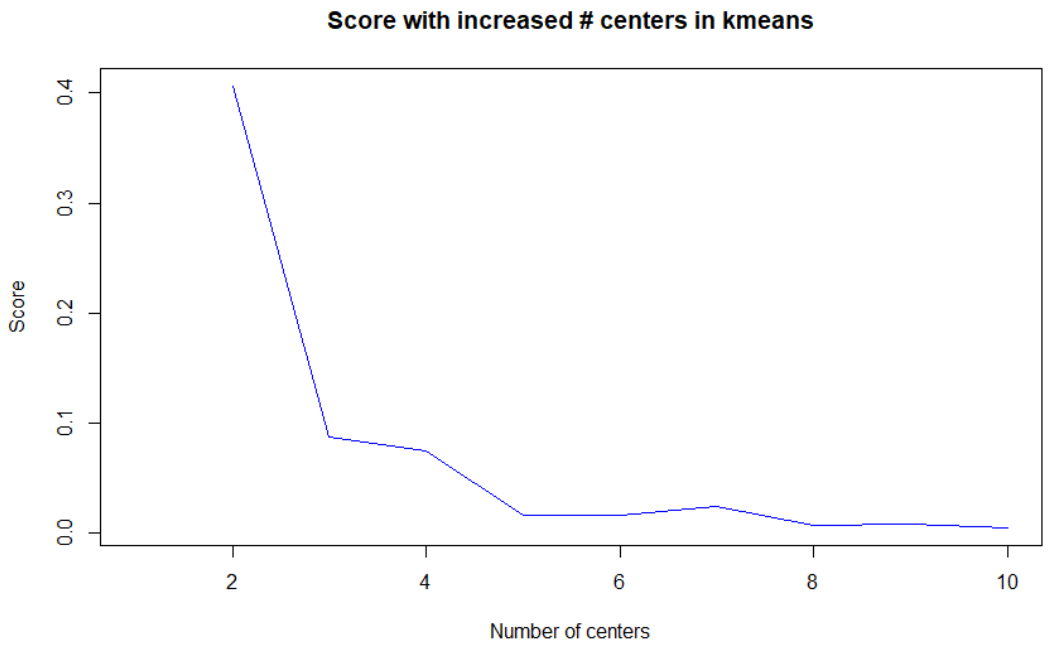


Figure 2.8: Elbow Plot for choosing number of clusters

Cluster 1	BFX (Belgium), BSESN (India), BVLG (Portugal), FCHI (France), FTMIB (Italy), NSEI (India), OMXC20 (Denmark), OMXHPI (Finland), OMXSPI (Sweden), OSEAX (Norway), SMSI (Spain)
Cluster 2	AEX (Amsterdam), AORD (Australia), BVSP (Brazil), DJI (USA), GDAXI (Germany), GSPTSE (Canada), IXIC (USA), RUT (UK), SPX (USA), SSMI (Switzerland), STOXX50E (EU)
Cluster 3	MXX (Mexico), SSEC (China)
Cluster 4	FTSE (UK), HSI (Hong Kong), IBEX (Spain), KS11 (Korea), KSE (Pakistan), N225 (Japan), STI (Singapore)

Table 2.2: Clusters of 31 Indices using K-means Clustering

Now, we can take these clusters and try various combinations of derived covariates: lagged, second-order terms, cross terms, and Google trend values. Note that for each cluster, we will continue considering all 31 realized variances at lag 1 - the additional covariates being considered will be derived from within the cluster itself. As we cannot use LASSO for variable selection when the number of covariates and the number of observations are of a similar scale, we will use Adaptive LASSO. We use AIC to choose the optimal shrinkage parameter, as our datasets are shallow and wide; BIC over-penalizes in such contexts. OUr results are summarized in Table 2.3.

Looking at Table 2.3, we see that the optimal choice of additional covariates for Clusters 1, 2, and 4 is 3. This seems to be a balance between overfitting and the persistence of realized variances - the more the lags the better it is in general, but given that the window length is just 90, we would have a lot of covariates given that these clusters are of size 11, 11, and 7 respectively. Hence, adding more lags would overfit the data and lead to poor generalization. This seems to be the issue with considering cross terms as well - given 11 covariates, we end up with $\binom{11}{2} = 55$ cross product terms and 11 squared terms for a total 92 terms along with the original 31 covariates (in comparison, the

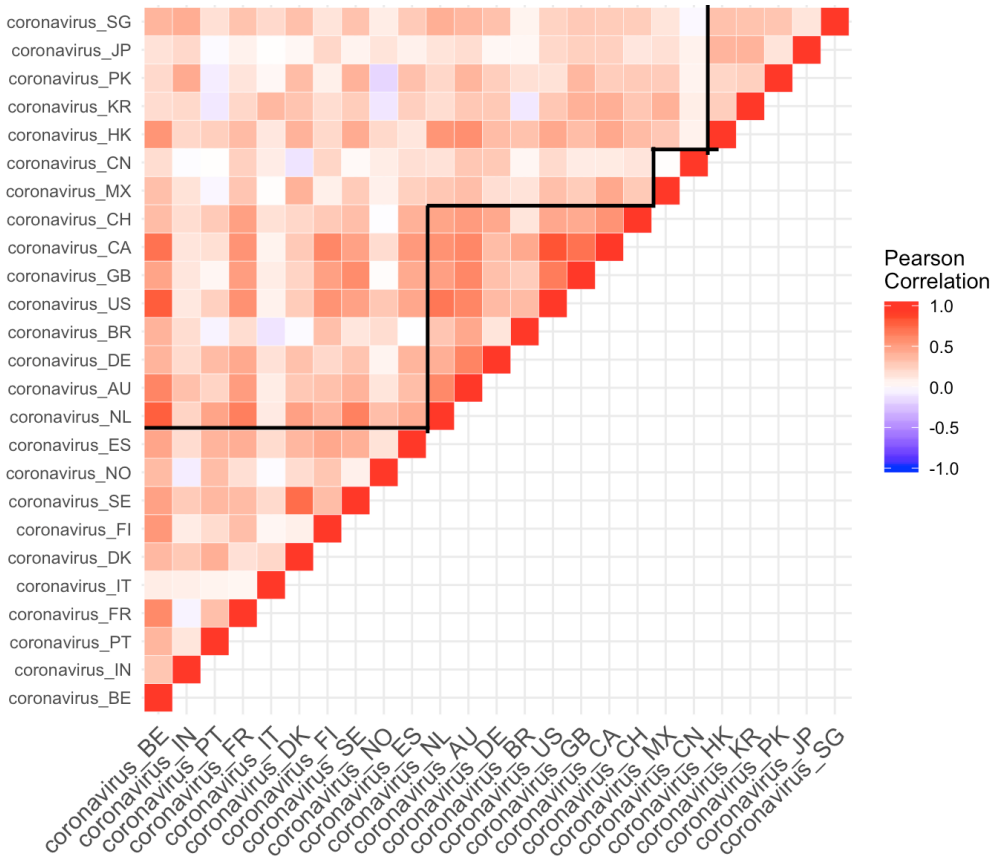


Figure 2.9: Correlation chart for Google trend for word- Coronavirus

Additional Covariates	Cluster 1	Cluster 2	Cluster 3	Cluster 4
Lag 1	5.040682×10^{-5}	1.295538×10^{-6}	9.065896×10^{-8}	3.540022×10^{-6}
Lag 2	6.453548×10^{-5}	2.211641×10^{-6}	1.267511×10^{-7}	5.398966×10^{-6}
Lag 3	5.218178×10^{-6}	8.909640×10^{-7}	1.298884×10^{-7}	7.397949×10^{-7}
Lag 4	1.589625×10^{-5}	2.091150×10^{-6}	1.529310×10^{-7}	3.012391×10^{-6}
Lag 5	1.692226×10^{-5}	1.734750×10^{-6}	1.866605×10^{-7}	2.745279×10^{-6}
Squared terms	8.287323×10^{-4}	2.882148×10^{-4}	1.991727×10^{-7}	4.578092×10^{-4}
Cross terms	3.289938×10^{-5}	2.683560×10^{-6}	7.657772×10^{-8}	1.344668×10^{-4}
Google trends	3.378902×10^{-5}	1.167724×10^{-6}	8.218027×10^{-8}	3.633296×10^{-4}

Table 2.3: Mean Squared Errors for different models in each cluster

total number of covariates at lag 3 would be 53). This problem eludes Cluster 3 which has just two elements, and hence, much fewer additional terms. Cross terms seem to work best for this cluster. Discussion of the Google trends covariate set follows in the next subsection.

Running the Adaptive LASSO model above also has an additional benefit. We take the datasets as selected above, and count the number of relevant variables per window for each of the 31 Adaptive LASSO models run. We present the change in number of relevant variables (represented as percentages of total number of covariates in order to adjust for the different number of covariates in each fine-tuned dataset) over time for the first four models and the average across all models in Figure 2.10.

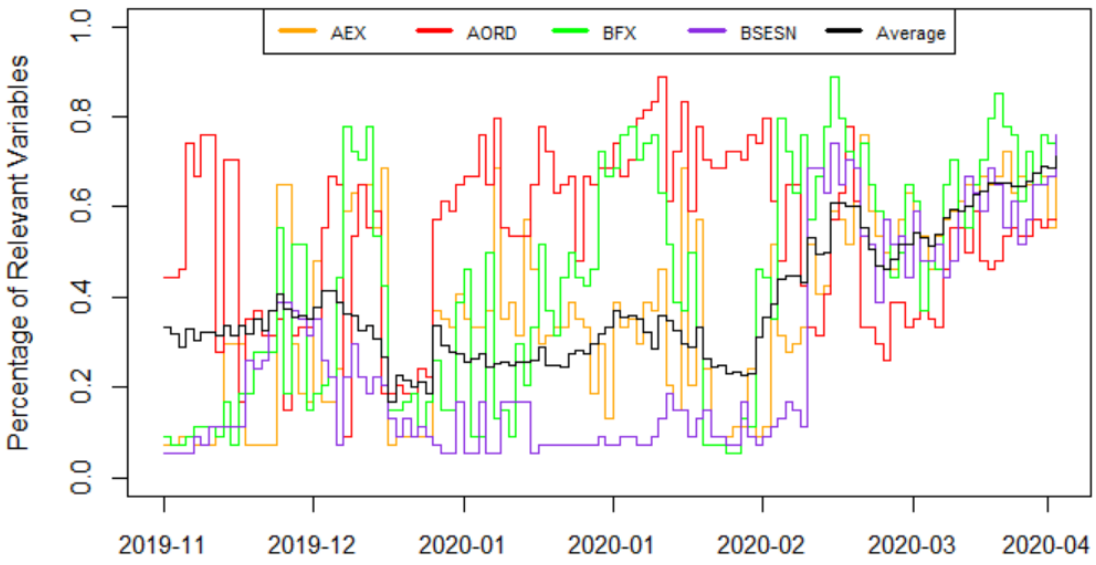


Figure 2.10: Change in percentage of relevant variables over time

We can see that after some time in February, there is a clear uptick in the number of relevant variables being chosen by the Adaptive LASSO model. While we cannot exactly pinpoint the date for this uptick, we have from the data that it occurs around the last week of February - we fix it to be February 24, 2020. The average percentage of relevant variables changed from 30.08% in the period before February 24 to 55.37% in the period after. This clearly indicates a structural break in the data, where methods such as LASSO and Adaptive LASSO fail to introduce any sparsity. Hence, we will focus on two types of methods - methods such as moving averages and HAR that solely depend on the data from just one index, and methods such as Gradient Boosted Trees and Random Forests that are adapted to multi-target regression and can exploit the correlations within the data.

2.6 Additional Datasets

From our analysis of various covariate sets, we note that the inclusion of the external Google trends data source did not improve model performance. The Google trends data added as a covariate set for each cluster in this case consisted of the daily Google trend values for web searches on the terms 'coronavirus' and 'COVID-19' for each of countries included in that cluster. Thus, for a cluster of n indices, there would be an additional $2n$ covariates by including this dataset. The resulting values were then adjusted to fit numerical format, so 'NA' was replaced with 0 and '<1' was replaced with 0.5. Given the prevalence of NA values, we did not find it fitting to only include the data set when it had non-NA values. The countries included in the Google trends data were not a one-to-one mapping of the countries included in each cluster because of some indices being overlapped (multiple indices tied to the United States, for example) and some indices not having a clear association with an available country code (such as the EU). Finally, data on days that did not appear in the original dataset (such as non-trading days) were discarded.

We hypothesize that the reason why including the additional dataset did not improve model performances is due to the structure of the Google trends dataset.

- Firstly, data points are normalized in accordance with the total volume of searches in the specified geographical location over the specified period of time. This means that the pro-

vided values are relative to other search topics, which could lead to artificial appreciated or depreciated values based on idiosyncratic events that have no predictive power on index values.

- Secondly, trends are cleaned so that duplicate searches by the same individual over a search period of time are removed. In this particular case, however, it is easy to imagine scenarios in which citizens are rapidly searching for coronavirus-related news over a short period of time, perhaps after a recent announcement by their government. Thus, the dataset values may in fact be overly flattened in a way that would diminish the predictive capabilities of this data source.

- Finally, this dataset is highly susceptible to keyword selection. The number of additional covariates increases linearly with the number of search terms included, and we felt that any further specificity in keyword selection would introduce other data points that we did not want to include. For example, words like "symptoms" or "testing" may show up in many contexts outside of those we wanted for the sake of our models.

Beyond Google trends, some of the additional datasets we considered were:

- **News sources:** We could use a web crawler then perform sentiment analysis on top news publications from each country. However, this approach also yields itself to significant bias given that we do not have the domain expertise necessary to shortlist a subset of news publications that would make this a feasible approach for the scale of this project. Furthermore, given the performance of this Google trends dataset, we felt that the web crawler approach would not likely yield improvements without much more significant analysis of a more refined text analysis approach rather than general sentiment.

- **Death rates:** We could add death rate data as presented by government authorities as a covariate. However, this approach would likely be too unpredictable to use as a reliable indicator. Firstly, the death rates attributed to the virus themselves at such a volatile time were potentially incorrect. Especially in the early stages, it's unclear how much visibility was given to deaths that were not initially linked to the coronavirus. As a result, we would potentially see arbitrary spikes that did not actually convey information about an uptick in the disease but instead an uptick in visibility. While this could be argued as a negative indicator for markets regardless of if the increased death rate reported was because of actual increased health concerns or because of lagged data, these two explanations have very different directional impact on index values going forward, and there's not good cause to believe that the explanation is consistent over time.

- **Predicted infection numbers:** With the progression of this virus, many scientists have built models about predicted infection numbers, which could be a potential data source that would have predictive power. However, we were concerned about the consistency of such data. Predicted infection numbers are likely to be dependent on government actions, which can be announced at irregular times, and there is no requirement on how quickly the numbers need to be updated following any particular announcement. As such, we were concerned that there would be instances when predicted infection numbers lagged significantly behind news but also other cases when predicted infection numbers adjusted quickly following public announcements. This type of inconsistency would likely detract from how informative this additional dataset would be.

As such, we rejected the use of additional datasets for the models we considered going forward.

3 Forecasting

The aim of our project is to build a forecasting model for each major country index to predict daily Realized Variances starting February 2, 2020. We would like to use these predictions to give VaR estimates, which is an important metric for analysing a portfolio.

3.1 Baseline Models

We first discuss the results from the two baseline models - Random Walk and Heterogeneous Autoregression (HAR), as these will be the benchmarks for us to beat.

3.1.1 Random Walk

If we believe that Realized Variance follows a Random Walk, then the best estimate of the value at time t+1 is the value at time t. This can be modelled as

$$\widehat{RV}_{t+1} = RV_t$$

The following Figure 3.1 shows MSEs of predictions for every index starting February 2020.

3.1.2 Heterogeneous Autoregressive (HAR) model

The heterogeneous autoregressive (HAR) models of high-frequency realized volatility are inspired by the Heterogeneous Market Hypothesis, and incorporate daily, weekly ,and monthly realized volatilities in the volatility dynamics with a (1,5,22) time horizon structure.

Mathematically, HAR model can be written as:

$$\widehat{RV}_{t+1} = \beta_0 + \beta_1 RV_t + \beta_2 RV_{w,t} + \beta_3 RV_{m,t}$$

where $RV_{w,t} = \frac{1}{5}\sum_{i=0}^{4} RV_{t-i}$ and $RV_{m,t} = \frac{1}{22}\sum_{i=0}^{21} RV_{t-i}$

We run HAR model by using 90-days rolling windows for estimating the β coefficients and using those coefficients for making predictions for the next day starting February 2020.

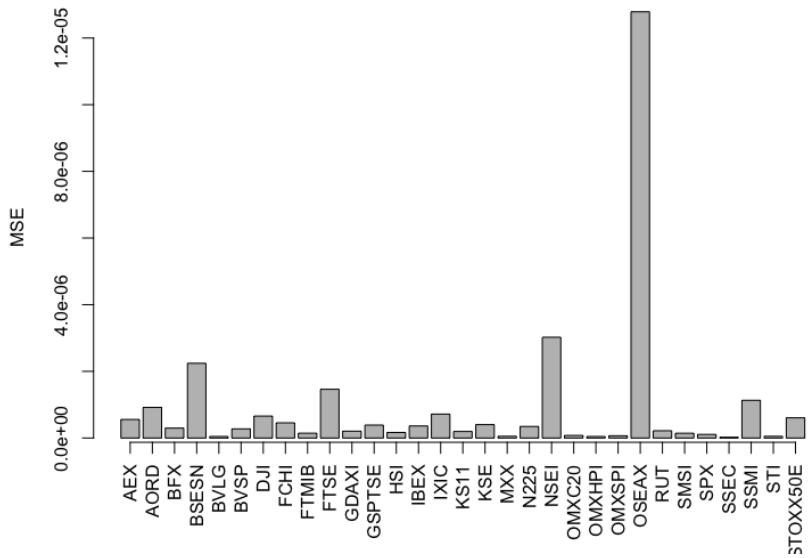
On comparing the MSEs from two models as seen in Figure 3.1, the tall bars for OSEAX and NSEI are interesting to note - both models seem to perform poorly, hinting that the relationship may not be linear for these indices. Focusing on NSEI and BSESN (both indices from India), we can see that the HAR model performs exceedingly poor in the scenarios where the Random Walk fails. While it may seem that the HAR model performs better than Random Walk on all other indices, this is solely due to the humongous size of errors for OSEAX, NSEI, and BSESN. In fact, the simple Random Walk model beats HAR for all but one index, SSEC.

3.2 Our Proposed Models

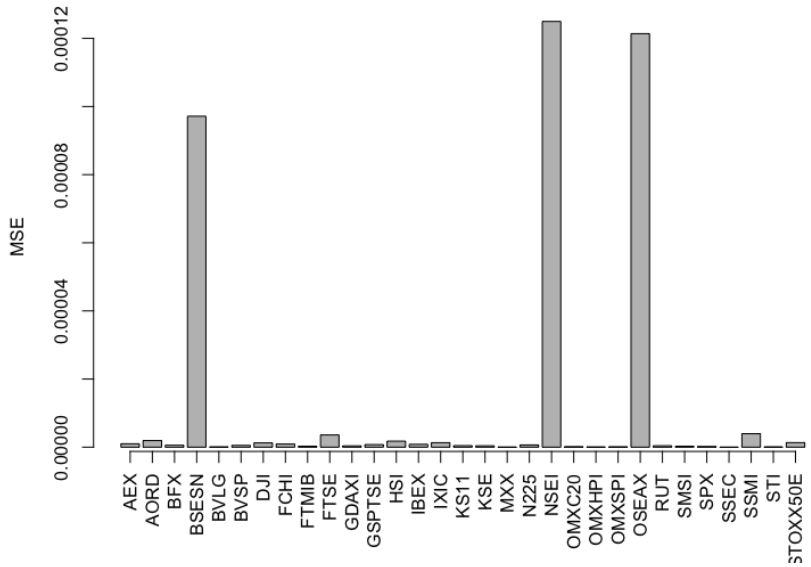
In this section, we talk about our model ideas and compare their performance with the benchmarks as well.

3.2.1 Simple Moving Average Model

Inspired by the (1,5,22) time horizon structure for volatility dynamics from the HAR model, we propose a simpler version of it where are coefficients are fixed throughout time for daily, weekly and monthly realized volatilities.



(a) Random Walk MSEs



(b) HAR Model MSEs

Figure 3.1: MSEs from Baseline models

One of the intuitive ways to assign the weights would be to give the highest weight to the observation at time t, then decreasing order to weights from weekly to monthly average realized variances. However we look at a general set of weights and impose a few conditions on these weights - we want these weights to be non-negative and we also want the sum of these weights to add up to one.

Mathematically, our model can be written as:

$$\widehat{RV}_{t+1} = a_1 RV_t + a_2 RV_{w,t} + a_3 RV_{m,t}$$

where $RV_{w,t} = \frac{1}{5}\sum_{i=0}^{4} RV_{t-i}$ and $RV_{m,t} = \frac{1}{22}\sum_{i=0}^{21} RV_{t-i}$
and $a_1 + a_2 + a_3 = 1$ and $a_1 \geq 0$, $a_2 \geq 0$, $a_3 \geq 0$.

We start with an initial set of weights as $a_1 = \frac{1}{2}$, $a_2 = \frac{1}{3}$, and $a_3 = \frac{1}{6}$ and we see that this Simple Average model beats the Random Walk MSEs for most of the indices, as shown in Figure 3.2. The ones that are still better for Random Walk are - BVLG, BVSP, OMXC20, OMXHPI, OMXSPI, RUT, SMSI. Interestingly, the SMA model performs much better on OSEAX, NSEI, and BSESN where the Random Walk had made uncharacteristically large errors.

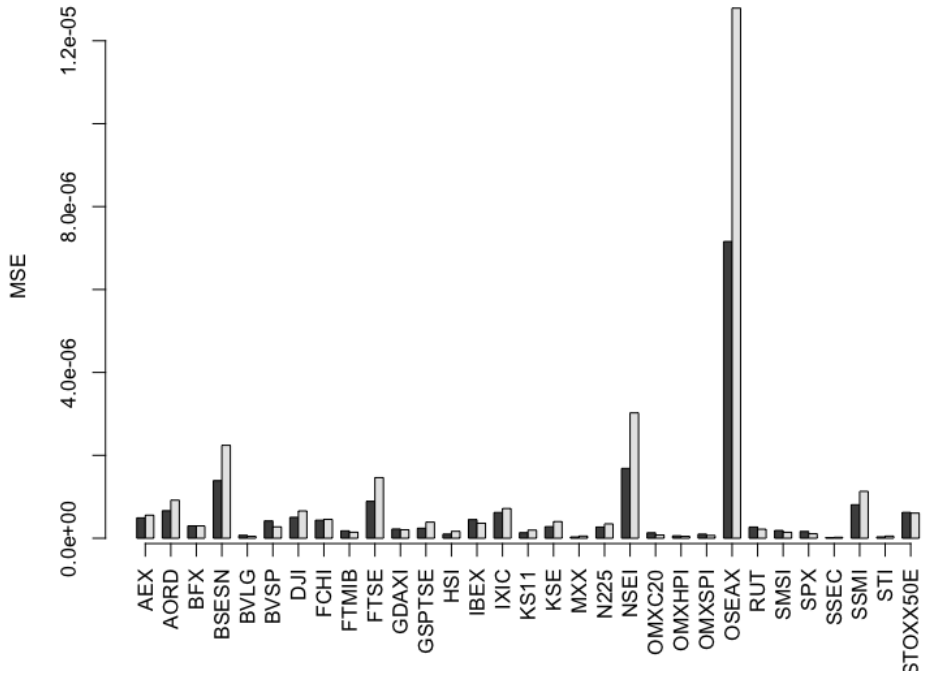


Figure 3.2: MSEs for Simple Moving Average Model (dark) vs Random Walk (light)

We then further look at different sets of weights and do a cluster-wise analysis to see which weights work the best for a specific cluster. To come up with ideas for weights, we run a regression of post-February data (like we do for the HAR models) and average the beta across the indices in a cluster. The following table summarises our MSE results for predictions starting February 1st, 2020 for different sets of chosen weights.

SMA Weights	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Overall
Random Walk	1.7575×10^{-6}	5.2578×10^{-7}	3.7738×10^{-8}	4.2674×10^{-7}	9.0899×10^{-7}
$(0.5, 0.33, 0.17)$	1.2145×10^{-6}	4.3245×10^{-7}	2.6050×10^{-8}	3.1887×10^{-7}	6.5809×10^{-7}
$(0.5, 0.5, 0)$	1.2356×10^{-6}	4.2609×10^{-7}	2.6600×10^{-8}	3.2100×10^{-7}	6.6383×10^{-7}
$(0.2, 0.8, 0)$	1.0938×10^{-6}	4.3691×10^{-7}	2.3450×10^{-8}	3.0669×10^{-7}	6.1393×10^{-7}
$(0, 0.8, 0.2)$	1.0556×10^{-6}	4.9391×10^{-7}	2.2300×10^{-8}	3.1774×10^{-7}	6.2300×10^{-7}
$(0.6, 0.4, 0)$	1.3129×10^{-6}	4.3409×10^{-7}	2.8250×10^{-8}	3.3473×10^{-7}	6.9730×10^{-7}
$(0.1, 0.8, 0.1)$	1.0627×10^{-6}	4.5736×10^{-7}	2.2600×10^{-8}	3.0793×10^{-7}	6.1036×10^{-7}

Table 3.1: Mean Squared Errors for each cluster under SMA with different weights

As our objective is to minimize the overall average MSE, choosing weights to be $(0.1, 0.8, 0.1)$ seems to be the best choice among all the weights. It leads to about 32.85% decrease over the Random Walk average MSE error, with much better performance cluster-wise as well. The performance

of weights $(0.1, 0.8, 0.1)$ is also very close to the best performance for each cluster, conveying good generalization by the model. To further test that these weights are not biased for the time-frame we are looking at, we use these weights to make Realised Variance predictions for the period July 2019 - January 2020, and still find about 29.82% improvement over Random Walk.

However, when we look at index level, we find that there are still indices for which Random Walk does better starting February. This brings us to the next proposed model, where we split the time frame to run different models.

3.2.2 Simple Moving Average Model and Random Walk Hybrid

From the Simple Moving Average Model $(0.1, 0.8, 0.1)$ above, we know there are some indices for which Random Walk still does better in the period starting February 2020. So in order to understand which of the two method - SMA or RW - works better in a certain market condition, we look at their daily MSEs. As we decided to work on data starting July 2019 and we need first month's data for calculating the 22-day average, our predictions start from August 2019. We see the market dynamics change starting around February 24, 2020 (this coincides with the point in our average correlation chart when it jumps from 0.3 to 0.7 as seen in Figure 3.3).

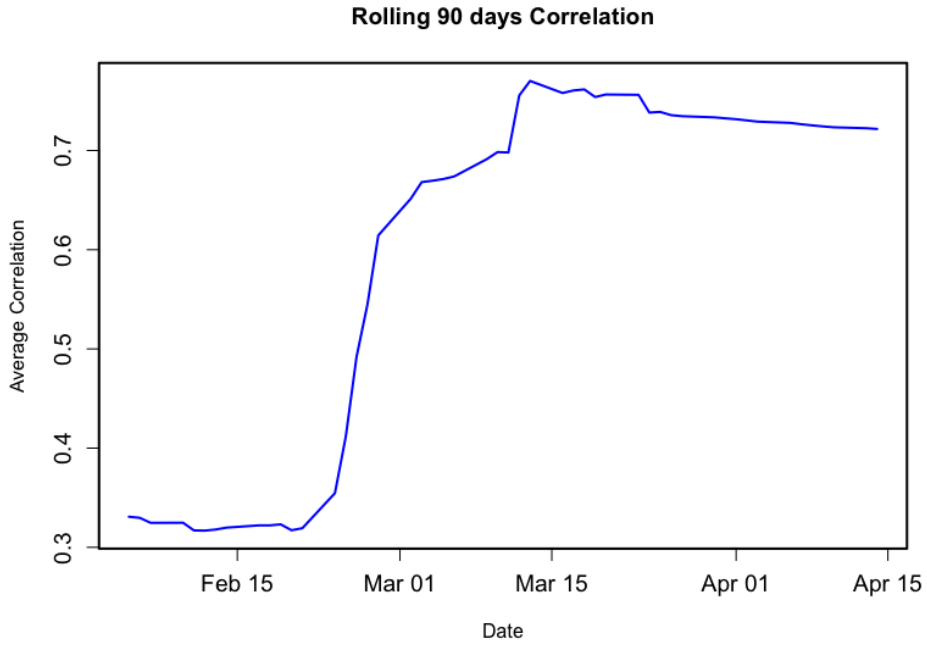


Figure 3.3: Correlation Chart from February - April 2020

In order to test which model of the two is better for the time before and similarly the time after the change in market dynamics, we look at daily MSEs and compare the performances on a daily basis. We also do this analysis separately for the pre-February 24 and post-February 24 period. We find that for the pre-February 24 period, the two methods outperform each other on considerably same amount of days. However post-February 24, we find that Random Walk does better an average of 60% days over all indices.

So we make a hybrid model - using Simple Moving Average $(0.1, 0.8, 0.1)$ model to make predictions till February 24th and Random Walk model to make predictions after that. However, when we compare the overall average MSE, the decrease is a meagre 0.01% over the Random Walk model.

This is understandable, as the hybrid model is pretty much a random walk, except for first 17 days in February. The MSE results are tabulated in Table 3.2.

Model	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Overall
Random Walk	1.7575×10^{-6}	5.2578×10^{-7}	3.7738×10^{-8}	4.2674×10^{-7}	9.0899×10^{-7}
$(0.1, 0.8, 0.1)$	1.0627×10^{-6}	4.5736×10^{-7}	2.2600×10^{-8}	3.0793×10^{-7}	6.1036×10^{-7}
Hybrid	1.7582×10^{-6}	5.2564×10^{-7}	3.6650×10^{-8}	4.2651×10^{-7}	9.0888×10^{-7}

Table 3.2: Mean Squared Errors for different models for each cluster

3.2.3 Modified HAR Model (Time and log-scale)

The original rationale behind the HAR model's choice of the (1, 5, 22) time horizon structures is that it thus incorporates daily, weekly, and monthly (short-term, medium-term, long-term, respectively) realized volatilities. However, in light of the rapid progression of virus-related news, we hypothesized that shorter and/or more frequent time horizon structures could prove beneficial. We thus decided test time horizon structures (1, 3, 5), (1, 5, 10, 22), (1, 3, 5, 10, 22). The choice of 3 was informed by previous lag analysis, and the choice of 10 was to represent a two-week horizon as part of the additional time horizons consideration.

In addition to testing different horizons, prior research surrounding HAR models had suggested that perhaps using the log of realized variances would show improved performance. We also saw that as-defined, the HAR model would potentially yield negative RV values, given that it is an OLS estimator for the betas (unlike, for example, SMA, where the predetermined weights would necessarily mean predictions come out positive). Using log-scaled values would restrict predictions to be strictly positive, which is desired behavior in this case. Thus, we also paired every tested time horizon with a log-scaled version.

The results of our modified HAR models are shown in Table 3.3. Random Walk results are also shown for each of the clusters and overall across all indices as a benchmark for comparison. All values shown in the table below are exclusively for the post-Feburary 1, 2020 period.

Model		MSEs				
time	log	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Overall
Random Walk		1.7575×10^{-6}	5.2578×10^{-7}	3.7738×10^{-8}	4.2674×10^{-7}	9.0899×10^{-7}
(1, 5, 22)	no	3.1452×10^{-5}	1.1874×10^{-6}	2.9171×10^{-8}	1.1121×10^{-6}	1.1835×10^{-5}
(1, 5, 22)	yes	1.3019×10^{-6}	9.2340×10^{-7}	2.4829×10^{-8}	4.2134×10^{-7}	8.8636×10^{-7}
(1, 3, 5)	no	4.1727×10^{-5}	1.0560×10^{-6}	2.7083×10^{-8}	1.3787×10^{-6}	1.5494×10^{-5}
(1, 3, 5)	yes	1.2077×10^{-6}	4.9114×10^{-7}	2.4017×10^{-8}	3.2308×10^{-7}	6.7732×10^{-7}
(1, 5, 10, 22)	no	4.3990×10^{-5}	1.4684×10^{-6}	3.0529×10^{-8}	1.4009×10^{-6}	1.6449×10^{-5}
(1, 5, 10, 22)	yes	1.4551×10^{-6}	1.2236×10^{-6}	2.5925×10^{-8}	4.3582×10^{-7}	1.0506×10^{-6}
(1, 3, 5, 10, 22)	no	5.5455×10^{-5}	1.7587×10^{-6}	3.1591×10^{-8}	1.7022×10^{-6}	2.0688×10^{-5}
(1, 3, 5, 10, 22)	yes	1.4881×10^{-6}	1.2175×10^{-6}	2.5784×10^{-8}	4.3249×10^{-7}	1.0594×10^{-6}

Table 3.3: Mean Squared Errors for different modified HAR models for each cluster

Overall, we observe that our hypothesis to shorten the overall HAR time horizon was correct, as the (1, 3, 5) horizon performed best in terms of average MSE in all clusters and also showed an approximately 25% improvement over the Random Walk model. Furthermore, we see that the overall trend suggests log-scale predictions to indeed be more accurate than predicting realized variances directly.

In Figure 3.4, we show a bar plot comparing the average MSE performance for the modified HAR(1, 3, 5) log-scale model and the average MSE performance using random walk on a per-index basis. We observe that, while not all indices necessarily saw a decrease in MSE from using the modified HAR model, the modified HAR was able to significantly improve on some of the worst-performing indices for Random Walk (most notably OSEAX).

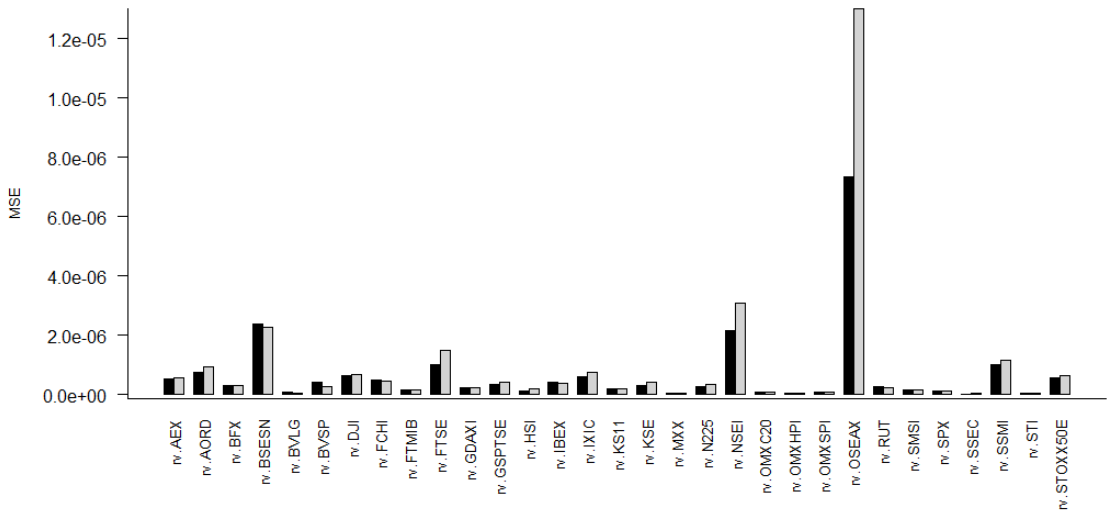


Figure 3.4: MSEs for Modified HAR Model (HAR(1, 3, 5) with log-scale; dark) vs Random Walk (light)

Given that the HAR model is essentially a linear regression of different realized variances, we were interested in potentially fitting the residuals from these linear regressions, a standard way to improve such models. While a cursory glance at plots of the residuals did not immediately suggest a clear trend as to what the appropriate covariates for predicting the residuals would be, we nonetheless attempted further improvements on our current modified-HAR frontrunner, the HAR(1, 3, 5) with log-scale model.

3.2.4 Modified HAR Model (residuals)

We tested two sets of variables with linear regressions to fit the residuals: (1) the RVs and mean RVs over the chosen time horizons for all the other indices, and (2) the square of the RVs and mean RVs for the index being predicted. Essentially, the goal of these tests was to ascertain whether introducing other indices or higher powered terms would improve the model. The data shown in Table 3.4 is again focused on the post-February 1, 2020 period for as long as the model redictor structures would allow. Instead of providing random walk as the benchmark model in this case, we show the modified HAR(1, 3, 5) with log-scale model results to see how fitting the residuals would potentially impact the results.

Interestingly enough, we observe that these two methods of fitting the residuals did not provide any noticeable gains on the predictive power of the modified HAR model. This result suggests that an index's predictive capability of its own price path dominates the predictive capability of other indices and further supports why index-specific methods like random walk and simple moving average can perform so well in these instances.

Model	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Overall
Baseline	1.2077×10^{-6}	4.9114×10^{-7}	2.407×10^{-8}	3.2308×10^{-7}	6.7732×10^{-7}
Linear Regression on other indices	1.2077×10^{-6}	4.9114×10^{-7}	2.407×10^{-8}	3.2308×10^{-7}	6.7732×10^{-7}
Linear Regression on squared terms	1.2077×10^{-6}	4.9114×10^{-7}	2.407×10^{-8}	3.2308×10^{-7}	6.7732×10^{-7}

Table 3.4: Mean Squared Errors for different modified HAR models for each cluster, compared to baseline of HAR(1, 3, 5) with log scaling

3.2.5 Boosted Trees

Now, we turn our attention to tree based methods adapted for multi-target regression. We first use Gradient Boosted Trees with a least squares loss function and the Friedman criterion for judging quality of splits. We have three hyperparameters to train - learning rate, number of estimators, and depth of the trees. Since there is a tradeoff between learning rate and number of estimators, we will first fix depth to the default value of 3 and find the optimal combination of the two, before tuning the depth hyperparameter. Note that we do not consider a hybrid model in this case - we assume that the trees would be able to decipher the structural break in the data through their splitting mechanism, and hence, train the same model over all of our data using windows of length 90 days. Since gradient boosted trees depend upon the randomization of their initial state, we run each model thrice and average out the MSEs over the validation set. The results are showin in Table 3.5.

Model		MSEs				
LR	n_est	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Overall
0.1	50	1.4612×10^{-6}	6.8821×10^{-7}	3.2969×10^{-8}	3.9407×10^{-7}	8.5382×10^{-7}
0.1	100	1.4406×10^{-6}	6.9140×10^{-7}	3.2969×10^{-8}	3.9497×10^{-7}	8.4783×10^{-7}
0.1	200	1.4493×10^{-6}	6.9426×10^{-7}	3.2969×10^{-8}	3.9362×10^{-7}	8.5161×10^{-7}
0.2	50	1.4341×10^{-6}	6.7777×10^{-7}	3.2969×10^{-8}	4.0911×10^{-7}	8.4174×10^{-7}
0.2	100	1.3842×10^{-6}	6.8899×10^{-7}	3.2969×10^{-8}	4.0989×10^{-7}	8.2822×10^{-7}
0.2	200	1.4351×10^{-6}	6.8895×10^{-7}	3.2969×10^{-8}	3.9868×10^{-7}	8.4373×10^{-7}
0.5	50	1.7111×10^{-6}	6.9496×10^{-7}	3.2969×10^{-8}	4.5847×10^{-7}	9.5727×10^{-7}
0.5	100	1.7590×10^{-6}	6.7741×10^{-7}	3.2969×10^{-8}	4.2340×10^{-7}	9.6014×10^{-7}
0.5	200	1.7351×10^{-6}	6.9072×10^{-7}	3.2969×10^{-8}	4.3504×10^{-7}	9.5903×10^{-7}

Table 3.5: Mean Squared Errors for different Boosted Tree models for each cluster

We have some very interesting results. Cluster 3 seems to be impervious to any tuning of hyperparameters, while Clusters 1 and 4 were the most sensitive to changes in the hyperparameters. The performance over Cluster 3 could be because of the covariates chosen - given that $x \to x^2$ is a monotonic function over non-negative reals, squaring the lag 1 realized variances does not introduce new information for constructing the splits. Further analysis reveals that this is due to target variables - the next day realized variances for MXX and SSEC. We try a variety of different predictors such as Cluster 3 at multiple lags and Cluster 4 with cross terms while keeping the target values fixed - but the MSEs do not change for any combination of the hyperparameters. In fact, the feature importances are always 0 for any model that we train - as per Gradient Boosted Trees, our datasets have no predictive power when it comes to these two indices. Given the lack

of interpretability of a boosted trees model, we cannot say why it believes so. However, this could shine light on why the k-means clustering made such an unintuitive cluster - compared to the other β vectors from Adaptive LASSO, these two indices must have shown minimal dependencies on the covariates, and hence been grouped together.

Now we look at tuning the depth hyperparameter for the optimal models chosen here. For Cluster 3, we fix this to be the model with learning rate 0.2 and number of estimators 50. As per Table 3.6, leaving the depth to be 3 seems to be the optimal choice. Cluster 4 shows a preference for depth 5; however, there is a very slight difference between the MSEs at depth 3 and 5.

Depth	Cluster 1 (LR=0.2, n_est=100)	Cluster 2 (LR=0.2, n_est=50)	Cluster 3 (LR=0.2, n_est=50)	Cluster 4 (LR=0.2, n_est=200)
3	1.3842×10^{-6}	6.7777×10^{-7}	3.2969×10^{-8}	3.9407×10^{-7}
5	1.4879×10^{-6}	6.7913×10^{-7}	3.2969×10^{-8}	3.9324×10^{-7}
10	1.4232×10^{-6}	6.9454×10^{-7}	3.2969×10^{-8}	3.9716×10^{-7}

Table 3.6: Tuning depth for different Boosted Tree models

Summarizing all our results and comparing against the baseline Random Walk model, we get from Table 3.7 that on the whole, using Gradient Boosted Trees gives us better results than simply using a Random Walk. However, these results should be taken with a pinch of salt - Cluster 2 performs quite poorly under this method, and even though Cluster 3 shows a reduction in MSEs compared to random walk, the values seem suspect. The poor performance of these two clusters was foreshadowed by the relative lack of sensitivity of changes in hyperparameters in Table 3.5, compared to Clusters 1 and 4.

	Random Walk	Gradient Boosted Trees	Percentage Change
Cluster 1	1.7575×10^{-6}	1.3842×10^{-6}	-21.24%
Cluster 2	5.2578×10^{-7}	6.7777×10^{-7}	28.91%
Cluster 3	3.7738×10^{-8}	3.2969×10^{-8}	-12.64%
Cluster 4	4.2674×10^{-7}	3.9324×10^{-7}	-7.85%
Overall	9.0899×10^{-7}	8.2260×10^{-7}	-9.50%

Table 3.7: Summarizing the results from Gradient Boosted Trees

3.2.6 Random Forest

For Random Forest, we implement a similar strategy as we did for Gradient Boosted Trees. We use a Random Forest with mean squared error as the splitting criterion that chooses the bootstrapping sample with replacement, and tune the number of trees and maximum depth hyperparameters. We fit the same model over the entire dataset, assuming that the structural break will be captured during the splitting process, and use walk-through validation with the next day's realized variances to analyze the performance of the model. Lastly, since the bootstrapping and the sampling of features at each node are randomized in this model, we run each model thrice over the dataset and average out the performance. The results can be seen in Table 3.8.

We consider `n_est = 10, 50, 100, 200` and `max_depth = None, 3, 5, 10`, (where `max_depth=None` implies that the trees are fully expanded).

We cannot discern any trend in the data - different clusters choose different combinations. However, one facet that is predominantly visible for Clusters 2 and 4 is that having only 10 trees

Model		MSEs			
n_est	Max Depth	Cluster 1	Cluster 2	Cluster 3	Cluster 4
10	None	1.3867×10^{-6}	6.1668×10^{-7}	3.2951×10^{-8}	4.3179×10^{-7}
10	3	1.3704×10^{-6}	6.2651×10^{-7}	3.2893×10^{-8}	4.1638×10^{-7}
10	5	1.3307×10^{-6}	6.0229×10^{-7}	3.2973×10^{-8}	4.1158×10^{-7}
10	10	1.2767×10^{-6}	6.1249×10^{-7}	3.3064×10^{-8}	4.1823×10^{-7}
50	None	1.3266×10^{-6}	5.8367×10^{-7}	3.3006×10^{-8}	3.9190×10^{-7}
50	3	1.3395×10^{-6}	5.8670×10^{-7}	3.2976×10^{-8}	3.9727×10^{-7}
50	5	1.3197×10^{-6}	5.8823×10^{-7}	3.2982×10^{-8}	3.9410×10^{-7}
50	10	1.3448×10^{-6}	5.9447×10^{-7}	3.2966×10^{-8}	3.9962×10^{-7}
100	None	1.3018×10^{-6}	5.8642×10^{-7}	3.3009×10^{-8}	3.9530×10^{-7}
100	3	1.3302×10^{-6}	5.8626×10^{-7}	3.2963×10^{-8}	3.9794×10^{-7}
100	5	1.3309×10^{-6}	5.8467×10^{-7}	3.2980×10^{-8}	3.9031×10^{-7}
100	10	1.3283×10^{-6}	5.8463×10^{-7}	3.2960×10^{-8}	3.9247×10^{-7}
200	None	1.3023×10^{-6}	5.8452×10^{-7}	3.2953×10^{-8}	3.9383×10^{-7}
200	3	1.3230×10^{-6}	5.8375×10^{-7}	3.2984×10^{-8}	3.9523×10^{-7}
200	5	1.3080×10^{-6}	5.8300×10^{-7}	3.2957×10^{-8}	3.9079×10^{-7}
200	10	1.3158×10^{-6}	5.8400×10^{-7}	3.2947×10^{-8}	3.9174×10^{-7}

Table 3.8: Mean Squared Errors for different Random Forest models for each cluster

leads to considerably higher validation MSEs, hinting towards underfitting of data. On the other hand, Clusters 1 and 3 end up performing best with just 10 trees (even though the variation in MSEs across various depths is largest for 10 trees compared to `n_est=50, 100, 200`). It is interesting to note that Cluster 3 does show some minor variations when the hyperparameters take on different values. However, the model surprisingly does not assign any importance of all of the features. It too believes than the covariates have no predictive power when it comes to Cluster 3. We assume that the minor variations seen in the results are simply to the randomization in bootstrapping and feature selection that Random Forests perform.

We compare the results to the Random Walk benchmark in Table 3.9. We see that there is an overall decrease in MSEs of 17.36%, even with Cluster 2 not performing as well. Interestingly, the random forest performs better on every cluster than the gradient boosted trees.

	Random Walk	Random Forests	Percentage Change
Cluster 1	1.7575×10^{-6}	1.2767×10^{-6}	-27.36%
Cluster 2	5.2578×10^{-7}	5.8300×10^{-7}	10.99%
Cluster 3	3.7738×10^{-8}	3.2893×10^{-8}	-12.84%
Cluster 4	4.2674×10^{-7}	3.9031×10^{-7}	-8.54%
Overall	9.0899×10^{-7}	7.5120×10^{-7}	-17.36%

Table 3.9: Summarizing the results for Random Forests

4 Analysis and Results

4.1 MSE Comparison

We now take the top results from each section and compare them to make our final decision.

Model	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Overall
Random Walk	1.7575×10^{-6}	5.2578×10^{-7}	3.7738×10^{-8}	4.2674×10^{-7}	9.0899×10^{-7}
HAR	3.1452×10^{-5}	1.1874×10^{-6}	2.9171×10^{-8}	1.1121×10^{-6}	1.1835×10^{-5}
Simple Moving Average	1.0627×10^{-6}	4.5736×10^{-7}	2.2600×10^{-8}	3.0793×10^{-7}	6.1036×10^{-7}
Hybrid Model	1.7582×10^{-6}	5.2564×10^{-7}	3.6650×10^{-8}	4.2651×10^{-7}	9.0888×10^{-7}
Modified HAR	1.2077×10^{-6}	4.9114×10^{-7}	2.4017×10^{-8}	3.2308×10^{-7}	6.7732×10^{-7}
Gradient Boosted Trees	1.3842×10^{-6}	6.7777×10^{-7}	3.2969×10^{-8}	3.9324×10^{-7}	8.2260×10^{-7}
Random Forests	1.2767×10^{-6}	5.8300×10^{-7}	3.2893×10^{-8}	3.9031×10^{-7}	7.5120×10^{-7}

Table 4.1: Summarizing all the results

The Simple Moving Average model $(0.1, 0.8, 0.1)$ performs the best across all clusters. We compare percentage improvement in MSEs for each cluster over the baseline, Random Walk, in Table 4.2. Simple Moving Average model leads to an overall improvement of **32.85%** in MSEs.

	Random Walk	Simple Moving Average	Percentage Change
Cluster 1	1.7575×10^{-6}	1.0627×10^{-6}	-39.53%
Cluster 2	5.2578×10^{-7}	4.5736×10^{-7}	-13.01%
Cluster 3	3.7738×10^{-8}	2.2600×10^{-8}	-40.11%
Cluster 4	4.2674×10^{-7}	3.0793×10^{-7}	-27.84%
Overall	9.0899×10^{-7}	6.1036×10^{-7}	-32.85%

Table 4.2: Summarizing the results from Simple Moving Average

4.2 Coverage Probabilities

Value-at-risk (VaR) is one of the main measures of financial risk. VaR is an estimate of how much value a portfolio can lose in a given time period at a given confidence level. One of the main uses for having a good volatility forecast is to compute Value-at-Risk measures. In Section 1, we showed that under our assumptions, VaR can be calculated as $Var_{\alpha\%,t+1} = c_\alpha \sqrt{\widehat{RV}_{t+1}}$, where c_α is the $\alpha\%$-percentile of a standard normal distribution.

For each of our indices, we compute daily VaR levels and compare them with their actual daily returns. We use daily data to assess the performance of VaR models, which is the goal of VaR backtesting. Figure 4.1 shows the daily log returns along with VaR estimates from our top models for AORD index.

In the following sections, we run two tests for each of our models - Kupiec's POF Test and Traffic Light Test.

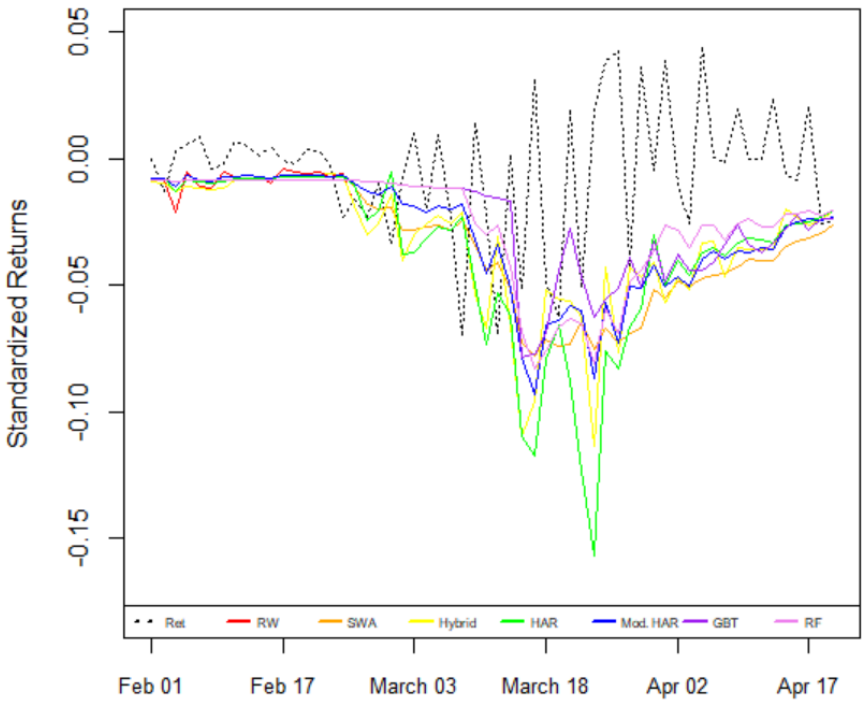


Figure 4.1: Daily Returns for AORD along with VAR Estimates

4.2.1 Kupiec's POF Test

Kupiec (1995) introduced a variation on the binomial test called the proportion of failures (POF) test. The POF test works with the binomial distribution approach. In addition, it uses a likelihood ratio to test whether the probability of exceptions is synchronized with the probability p implied by the VaR confidence level. If the data suggests that the probability of exceptions is different than p, the VaR model is rejected. The POF test statistic is:

$$LR_{PoF} = -2log\left(\frac{(1-p)^{N-x}p^x}{(1-\frac{x}{N})^{N-x}(\frac{x}{N})^x}\right)$$

where x is the number of failures, N the number of observations and $p = 1–VaR$ level.

This statistic is asymptotically distributed as a chi-square variable with 1 degree of freedom. The VaR model fails the test if this likelihood ratio exceeds a critical value which depends on the test confidence level.

4.2.2 Running Kupiec's POF Test on our models

For data starting February 1, 2020, we look at the daily returns and predicted VaR estimates at 5% level and evaluate our VaR estimates using Kupiec's test. Table 4.3 shows percentage acceptances by the test for each of the models at a 5% significance level. It also mentions a list of indices which are accepted for each model.

The SMA model performs the best on this test as well, with significant test statistics for over 58.06% of the indices, compared to 45.16% of the Random Walk. In fact, it is significant for all but one indices that the Random Walk is significant for. The Hybrid method is the next best method -

understandably due to its construction. The HAR and Modified HAR have the same percentage of acceptances, but accept different indices. The tree based models perform extremely poorly - even though they have better MSEs than the Random Walk. Interestingly, both the tree models are significant for SSEC, given that they performed poorly on Cluster 3 (which contained MXX and SSEC).

Model	Percentage of Acceptances	Accepted Indices
Random Walk	45.16%	BVSP, FCHI, FTMIB, GDAXI, GSPTSE, HSI, IXIC, KS11, N225, OMXC20, OMXSPI, OSEAX, SPX, SSEC
HAR	35.48%	BVLG, GSPTSE, HSI, IXIC, MXX, N225,OMXC20, OMXSPI, OSEAX, SMSI, SSEC
Simple Moving Average	58.06%	AEX, BFX, BVLG, BVSP, DJI, FCHI, FTMIB, GDAXI, GSPTSE, HSI, IXIC, KS11, MXX, N225, OMXC20, OMXSPI, OSEAX, SSEC
Hybrid	48.39%	AEX, BVSP, DJI, FCHI, FTMIB, GDAXI, GSPTSE, HSI, IXIC, KS11, N225, OMXC20, OMXSPI, OSEAX, SSEC
Modified HAR	35.48%	FCCHI, FTMIB, GDAXI, GSPTSE, HSI, IXIC, KS11, OMXC20, OMXSPI, OSEAX, SSEC
Gradient Boosted Trees	6.45%	HSI, SSEC
Random Forests	9.68%	HSI, OMXC20, SSEC

Table 4.3: Summarizing all the results for Kupiec's Test

4.2.3 Traffic Light Test

A variation on the binomial test proposed by the Basel Committee is the traffic light test or three zones test. For a given number of exceptions x, one can compute the probability of observing up to x exceptions. The probability is computed using a binomial distribution.

The three zones are defined as follows:

- The "red" zone starts at the number of exceptions where this probability equals or exceeds 99.99%. It is unlikely that too many exceptions come from a correct VaR model.
- The "yellow" zone covers the number of exceptions where the probability equals or exceeds 95% but is smaller than 99.99%. Even though there is a high number of violations, the violation count is not exceedingly high.
- Everything below the yellow zone is "green." If you have too few failures, they fall in the green zone. Only too many failures lead to model rejections.

4.2.4 Running Traffic Light Test on our models

We run the Traffic Test for all indices starting February 1, 2020 and summarize the results in Table 4.4.

Model	Red	Yellow	Green
Random Walk	9.68%	58.06%	32.26%
HAR	9.68%	64.52%	25.81%
Simple Moving Average	3.23%	51.61%	45.16%
Hybrid	6.45%	61.29%	32.26%
Modified HAR	9.68%	67.74%	22.58%
Gradient Boosted Trees	58.06%	35.48%	6.45%
Random Forests	48.39%	41.94%	9.68%

Table 4.4: Summarizing all the results for Traffic Light Test

The results obtained from the Traffic Light Test are quite similar comparatively to those obtained from Kupiec Test. SMA model performs the best, and the tree models perform the worst. This test also lets us differentiate between models that perform similarly on the Kupiec Test. For example, both HAR and Modified HAR are significant for 35.48% of the indices on the Kupiec Test, albeit for different sets of indices. Unless one has a preference for some indices, it is impossible to choose between the two. However, we can see in Table 4.4 that HAR performs slightly better than Modified HAR.

We further display the results of the traffic light test in Table 4.5, where the color of the cell displays the result from the test. We also add checkmarks within a cell if the Kupiec test was signifcant for an index under a particular model, to enable easier comparison of the two tests.

Indices	Kupiec's and Traffic Light Test Results						
	RW	HAR	SMA	Hybrid	mod HAR	GBT	RF
AEX			✓	✓			
AORD							
BFX			✓				
BSESN							
BVLG		✓	✓				
BVSP	✓		✓	✓			
DJI			✓	✓			
FCHI	✓		✓	✓	✓		
FTMIB	✓		✓	✓	✓		
FTSE							
GDAXI	✓		✓	✓	✓		
GSPTSE	✓	✓	✓	✓	✓		
HSI	✓	✓	✓	✓	✓	✓	✓
IBEX							
IXIC	✓	✓	✓	✓	✓		
KS11	✓		✓	✓	✓		
KSE							
MXX		✓	✓				
N225	✓	✓	✓	✓			
NSEI							
OMXC20	✓	✓	✓	✓	✓		✓
OMXHPI							
OMXSPI	✓	✓	✓	✓	✓		
OSEAX	✓	✓	✓	✓	✓		
RUT							
SMSI		✓					
SPX	✓						
SSEC	✓	✓	✓	✓	✓	✓	✓
SSMI							
STI							
STOXX50E							

Table 4.5: Coverage Probability Tests Results for all models

We can notice that the two tests are almost synonymous and give similar results. Every cell that is green has a checkmark in it, and every cell that is red does not have a checkmark in it.

Qualitatively, the SMA model looks the best as it has the maximum number of green cells and checkmarks. We cannot exactly differentiate between the other non-tree methods, but there seems to be no strong trend as to which indices a model falls in the red zone for. However, the green zones definitely occur in blocks and groups. This tells us that all models perform well on similar groups - or that it was relatively easy to capture the trends on most indices.

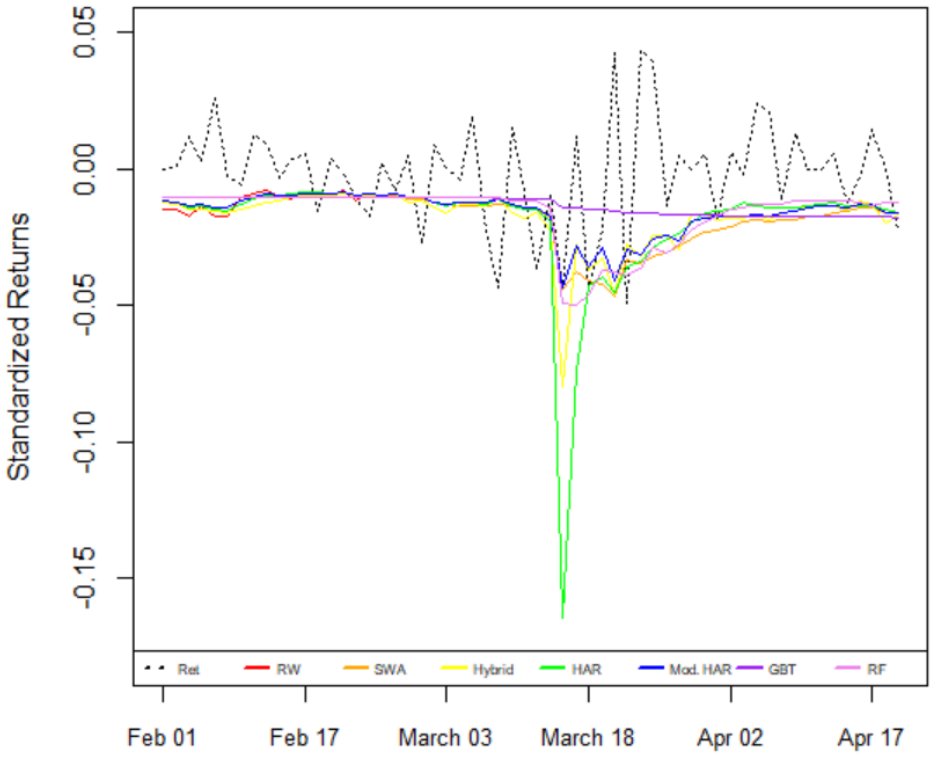
The tree based methods seem to be closely related in their performances. Even though Random Forest had a marked improvement in MSEs compared to Gradient Boosted Trees, this improvement does not translate to VaR levels. The color map is nearly the same for both the methods.

Table 4.5 also lets us compare indices along with the models. NSEI and BSESN that were

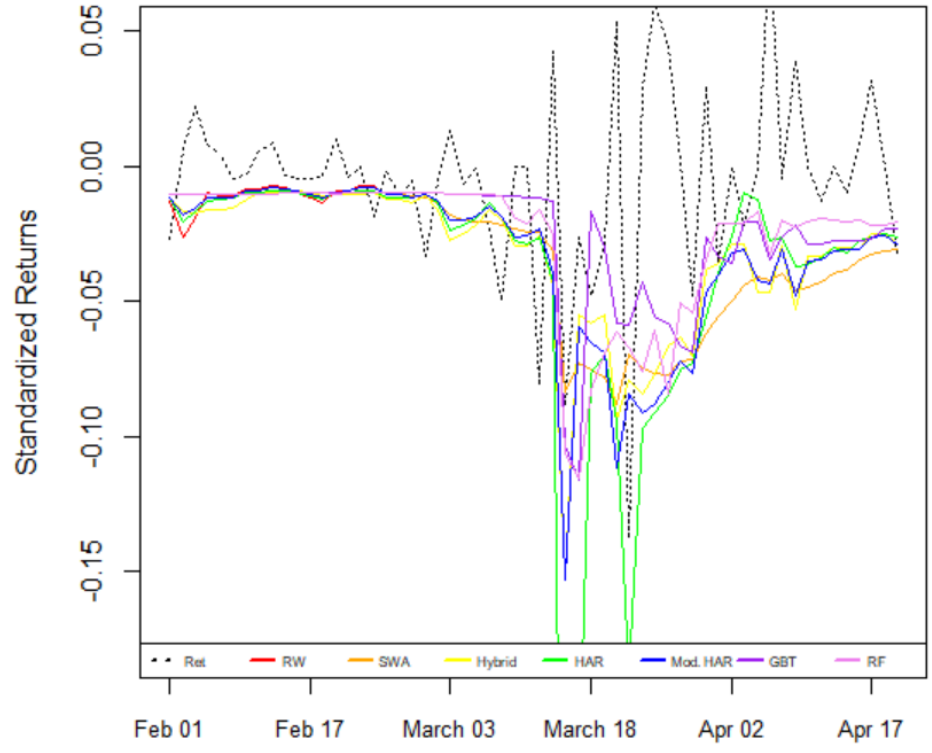
particularly hard for all of the models (as seen in Figures 3.2 and 3.4) often breached the VaR levels predicted by the models. Given that both these indices are from India and the results for Karachi Stock Exchange (KSE) aren't very promising, we believe that our models were unable to capture some of the local effects in South East Asia. On the other hand, our models performed quite well on the Nordic Indices (apart from OMXHPI).

To better understand the results, we look at the daily log returns overlayed with the VaR predictions for two indices: HSI (which had all greens) and BSESN (which had only yellows and reds) in Figures 4.2a and 4.2b respectively. The graphs make the reasons for this stark difference immediately clear. While HSI has fluctuations, it fluctuates within a fixed band and does not have sharp peaks - in fact, the predicted VaR level from the Gradient Boosted Trees is almost flat. Hence, most models are able predict well, as seen especially in the portion after March 18 when the VaR levels neatly trace out the troughs of the HSI index. On the other hadn, the BSESN index has wild fluctuations and extremely sharp movements. In the period surrounding March 18, we can see both record highs and record lows, introducing a lot of uncertainty. A small caveat with respect to BSESN is that there are some days when the index reports an open and close price, but has realized variance 0 (such as August 05, 2019). We leave in such data points due to their infrequency of occurrence, but it is obvious that no method would be able to predict 0 for such days, and that this might have a local (but vanishing) impact some of the estimates from that day on.

These images also provide us with a glimpse into the inner workings of the methods. They seem to agree in periods of relative calm (such as the start of the period, when the Covid-19 crisis was yet to make a global impact), but diverge in turbulent times (such as around March 18). The sharp dips in HAR tell us that we should not look at VaR as the only risk-management metric. While the daily returns are unlikely to breach the sharp dip, it holds low informational value from a predictive perspective. An interesting test to conduct would be to estimate the degree of overshooting by each model.



(a) Daily Returns for HSI along with VAR Estimates



(b) Daily Returns for BSESN along with VAR Estimates

Figure 4.2: Daily Returns and VaR Levels

5 Conclusion

After comparing best models for each of the method, we conclude that a Simple Moving Average Model is the best choice for making Value-at-Risk predictions. The model outperforms the rest on both a cluster-wise and the overall level. Furthermore, it passes the VaR backtesting for the most number of indices, making the predictions more reliable. Therefore, we have used SMA $(0.1, 0.8, 0.1)$ for submitting predictions for the next fifteen days. In our dataset, we had data until time t to predict the values at time $t + 1$. However, as we need to make predictions as far as $t + 15$, we use the previously predicted values as a proxy for true values. Note that we predict Realized Variance for all indices but IBEX and SPX, for which we predict Realized Ranges.