REALIZED VOLATILITY PREDICTION

PROJECT REPORT

OCTOBER 19, 2022

Authored By: Amit Kukreja

1.0 Executive Summary

1.1 Background

In financial markets, volatility captures the amount of fluctuation in prices. It is an essential input for determining option prices of stocks. Optiver, a global electronic market maker firm that trades in stock options, wants to evolve its industry leading options pricing algorithm to the next level by improving prediction of short-term volatility.

The goal of the project was to devise an algorithm to accurately predict the future 10-minute realized volatility in the weighted average price (wap) of each stock. **WAP** is the weighted average of the top bid & ask price in the book. Realized volatility (RV) is calculated by taking the square root of the sum of the squared log returns of wap.

where r is the log return of wap.

$$\mathbf{RV} = \sqrt{\sum_{t} r^2_{t-1,\,t}}$$

As part of a Kaggle contest, Optiver has provided the following training data for 112 stocks across approx. 3800 10-min time periods:

1. *Book* and *Trade* data for each stock for the prior 10-minutes i.e. the 10 minute time period before the time period for which the prediction is to be made.
2. Target: The realized volatility in the future 10-min period (to train the model)
3. Link: [Data for realized volatility prediction]

No data other than the prior 10-min book and trade data can be used for making the volatility predictions.

1.2 Evaluation metric and performance benchmark

Optiver's preferred evaluation metric is the **RMSPE** (*Root Mean Squared Percentage Error*) that needs to be minimized.

Realized volatility is known to be auto correlated. A naïve (baseline) model that predict future 10-min volatility based on prior 10-min volatility has a RMSPE score of **0.341** for the training dataset provided.

1.3 Insights from data

The exploratory data analysis of the book and trade data provided some useful insights:

1. WAP is a random walk, it's log returns are stationary
2. WAP & Trade Price closely track one another, but it is difficult to say which one drives the other
3. Features derived from the data, such as realized volatility of current 10-min period, trade size, high-low ratio of wap etc. are not normally distributed
4. 'target' – the realized volatility of future 10-min period is strongly correlated to features like
    a. 'realized_volatility_wap' with r = 0.84
    b. 'high_low_ratio_wap' with r = 0.79
    c. 'realized_volatility_trade_price' with r = 0.83
5. Many stocks show strong correlation with each other on realized volatility. The 112 stocks can be clustered into 3-4 groups based on realized volatility feature.

1.4 Key outcomes

1. Features based on following knowledge areas are useful for models. This was corroborated by linear and light GBM models, that did not score as well, but helped understand feature importance.
    a. Technical analysis of stocks
    b. Concept of motion dynamics applied to stock price
    c. Stock clusters derived using K-Means clustering
    d. Splitting the current 10-min time period into smaller time bands.
2. The following features figured on the top in feature importance analysis:

    i) realized_volatility_wap1
    ii) realized_volatility_wap1_cluster_mean
    iii) realized_volatility_wap1_H2
    iv) price premium
    v) acceleration_trade_price
    vi) momentum_trade_price
    vii) high_low_ratio_wap
    viii) momentum_wap

3. A deep learning model based on numeric & categorical features delivered a strong performance in predicting future realized volatility, with RMSPE score of **0.215365** on the test set. This score is among **Top 30** scores present on the Kaggle contest leaderboard.

1.5 Business Applications

The deep learning model developed has strong predictive power for determining future 10-min realized volatility across more than 100 stocks. It has a simple architecture and good computational performance:

a. 3 hidden layers and 1 embedding layer (for categorical features)
b. 100 – 200 neurons per hidden layer
c. Avg. computation time of **0.018** seconds for all 112 stocks

As such, it suited for real time applications like stock options trading and can be deployed by electronic market maker firms such as Optiver.

2.0 Introduction

In financial markets, volatility captures the amount of fluctuation in prices. High volatility is associated to periods of market turbulence and to large price swings, while low volatility describes more calm and quiet markets. For global electronic market maker firms like Optiver, accurately predicting volatility is essential for the trading of options, whose price is directly related to the volatility of the underlying product. To evolve its industry-leading pricing algorithm to the next level, Optiver has sponsored a Kaggle competition to predict realized volatility in stock prices for more than 100 stocks.

2.1 Problem Statement

Predict the future 10-min realized volatility in the weight average bid/ask price (wap) of the given stocks.

2.2 Context

Optiver wants the prediction of future 10-min volatility to be done only using the prior 10-min book and trade data. No data other than the prior 10-min book and trade data can be used for making the predictions.

2.3 Criteria for success

The goal is to minimize the **RMSPE** (*Root Mean Squared Percentage Error*) of the predictions made. This metric is calculated using the predicted and true future values provided in the training dataset.

2.4 Dataset

As part of a Kaggle contest, Optiver has provided the following training data for 112 stocks across approx. 3800 10-min time periods:

1. *Book* and *Trade* data for each stock for the prior 10-minutes i.e. the 10 minute time period before the time period for which the prediction is to be made.
2. Target: The realized volatility in the future 10-min period (to train the model)
3. Link: Data for realized volatility prediction

2.5 Constraints within solution space

1. The non-availability of data beyond the current 10-minute period limits the ability to learn patterns from a longer time series analysis.

2. There could also be external factors such as news, earnings call, Fed meetings that influence a particular stock's volatility or the market as a whole, However, these are not available for model building.

2.6 Key Definitions

Before diving into the project details, let us understand some key definitions.

1) **WAP**: WAP is the weighted average of the top bid & ask price in the book. It is calculated as follows:

$$\text{wap1} = \frac{\text{bid1\_price} * \text{ask1\_size} + \text{ask1\_price} * \text{bid1\_size}}{\text{ask1\_size} + \text{bid1\_size}}$$

2) **Realized Volatility**: Realized volatility (RV) is calculated by taking the square root of the sum of the squared log returns of wap.

$$RV = \sqrt{\sum_{t} r^2_{t-1,\,t}}$$

where r is the log return of wap between successive time stamps t-1 & t.

3.0 Data Understanding & Cleaning

3.1 Inspecting the datasets available

There are 3 table provided:

1) **Book**: This table contains the details of the top two bid and asks for a particular stock.

	time_id	seconds_in_bucket	bid_price1	ask_price1	bid_price2	ask_price2	bid_size1	ask_size1	bid_size2	ask_size2
0	5	0	1.001422	1.002301	1.00137	1.002353	3	226	2	100
1	5	1	1.001422	1.002301	1.00137	1.002353	3	100	2	100
2	5	5	1.001422	1.002301	1.00137	1.002405	3	100	2	100
3	5	6	1.001422	1.002301	1.00137	1.002405	3	126	2	100
4	5	7	1.001422	1.002301	1.00137	1.002405	3	126	2	100

Table 3.1: Book table

A quick definition of some of the important variables

- time_id : Id given to a 10-min time window
- seconds_in_bucket: seconds within the 10-min time window, takes values 0 to 599
- bid/ask prices: Prices are normalized

One can see that there are *missing* 'seconds_in_bucket'. This means that between those time stamps there was no change in the top 2 bid & ask prices or sizes. For the purpose of competition, Optiver recommends that data can be forward filled.

The table also doesn't contain a column for the stock name or stock-id. The stock-id is part of the table name and there is a separate table for each stock.

Other than above, the table is clean and has no missing values.

2) **Trade**: This table contains the details of buying/selling activity for a particular share:

	time_id	seconds_in_bucket	price	size	order_count
0	5	21	1.002301	326	12
1	5	46	1.002778	128	4
2	5	50	1.002818	55	1
3	5	57	1.003155	121	5
4	5	68	1.003646	4	1

Table 3.2: Trade table

- Price: This is the price at which a trade occurred
- Size: No. of shares traded in that particular 'second_in_bucket'
- order_count: No. of orders executed

The trade table is sparser than the book table and understandably so. Buying/selling happens at a lower frequency as compared to placing of bids and asks. However, in this table, data in the intervening 'seconds_in_bucket' cannot be filled as there were no trades at those time stamps.

There were no missing values found in the trade table. Most stocks had data for 3830 time_ids, except about 10 stocks that had data for approx. 3815 time_ids.

3) **Training**: This table contains the 'target', which is the realized volatility of the 10-min time window proceeding a particular time_id. For e.g. 0.004136 is the realized volatility in the 10-min that followed time_id 5. So, while book & trade data is available for time_id 5, it is not available for the 10-min following that. The only variable available is the 'target'. This is provided to help train a predictive model.

	stock_id	time_id	target
0	0	5	0.004136
1	0	11	0.001445
2	0	16	0.002168
3	0	31	0.002195
4	0	62	0.001747

Table 3.3: Training table

There were no missing values in the training table.

Important points to be noted

While data tables are quite clean, some key points to be noted basis inputs provided by Optiver

1) **Time_ids do not have a sequential logic**. This means that time_id 6 does not occur after time_id 5. Optiver has coded time_ids in this way, so that the predictive model works only with the prior 10-min time window.
2) **Time_ids are consistent across all stocks**. This implies that time_id 5 is the same actual time period across all stocks. This could be useful in understanding if behavior of different stocks has anything in common.
3) **Prices are normalized**: This is done to avoid correlating stock data with that of the actual company.

3.2 Key Outcomes

1. To have a wap price generated at a constant frequency of one second, I added the missing 'seconds_in_bucket' to the book file. I also added the stock-id in a separate column, so that this feature could be picked up directly while doing further processing. The updated table looks like below:

	seconds_in_bucket	time_id	bid_price1	ask_price1	bid_price2	ask_price2	bid_size1	ask_size1	bid_size2	ask_size2	stock_id
0	0	5.0	1.001422	1.002301	1.00137	1.002353	3.0	226.0	2.0	100.0	0
1	1	5.0	1.001422	1.002301	1.00137	1.002353	3.0	100.0	2.0	100.0	0
2	2	5.0	1.001422	1.002301	1.00137	1.002353	3.0	100.0	2.0	100.0	0
3	3	5.0	1.001422	1.002301	1.00137	1.002353	3.0	100.0	2.0	100.0	0
4	4	5.0	1.001422	1.002301	1.00137	1.002353	3.0	100.0	2.0	100.0	0

Table 3.4: Updated book table

2. As no test data is available, I split all the 3 files with an 80:20 train-test ratio, keeping the time_ids consistent across the 3 files.

4.0 Exploratory Data Analysis

Some of the main questions in my mind that I explored during the EDA stage were:

1. How do the 'wap' and trade prices move during a time bucket? Are they a random walk or stationary?
2. Does 'wap' impact the 'trade price' or vice-versa?
3. Build some basic features from the data in the two files and see how these features are related to the 'realized_volatility'
4. What is the distribution of realized volatility? Is it normal? What about other features like size, order count, turnover?
5. How does 'realized_volatility' moves with time. Is it auto-correlated?
6. Is realized_volatility of different stocks connected? Are there any clusters of stocks with similar volatility across time buckets?

4.1 WAP

WAP is perhaps the most important parameter to understand as the ‘realized_volatility’ is based on its movement over a 10-min period.

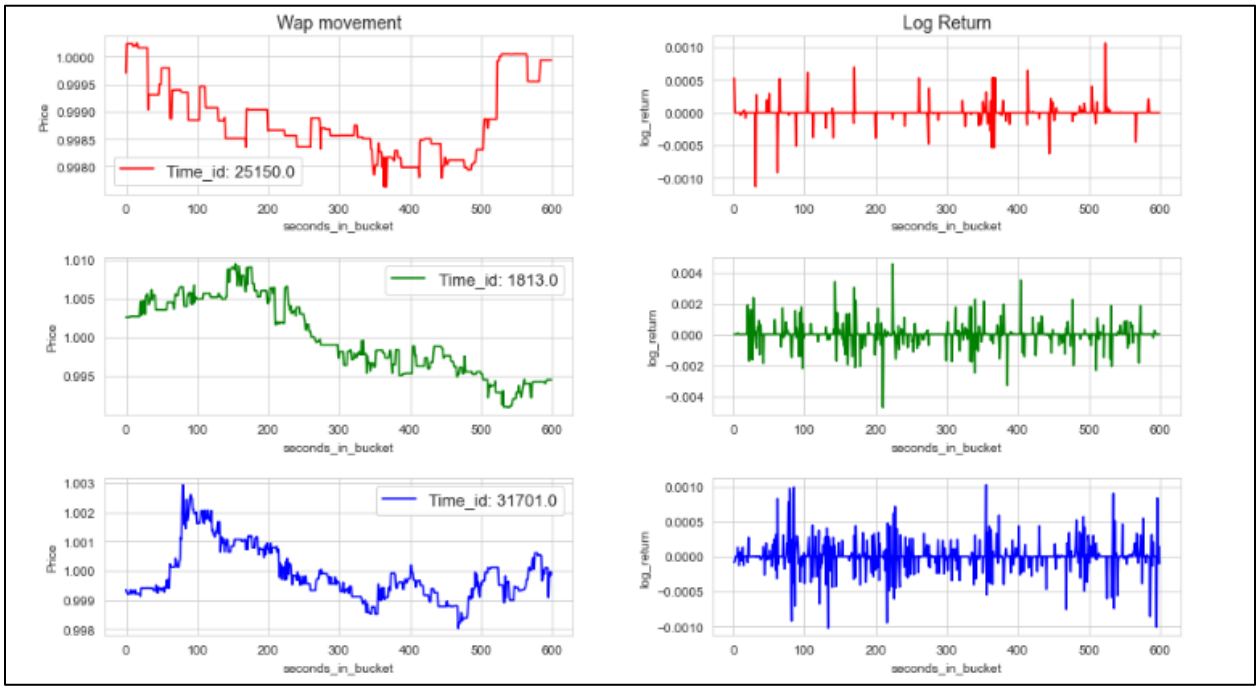


Fig 4.1: Wap movement and its log returns

On visual inspection in figure 2.1, wap seem like a random walk. There are time_ids where is moves a lot (green & blue), while in others (red), it is quieter.

The log returns look stationary. The augmented Dickey Fuller test confirms that WAP is non-stationary, while log returns are stationary.

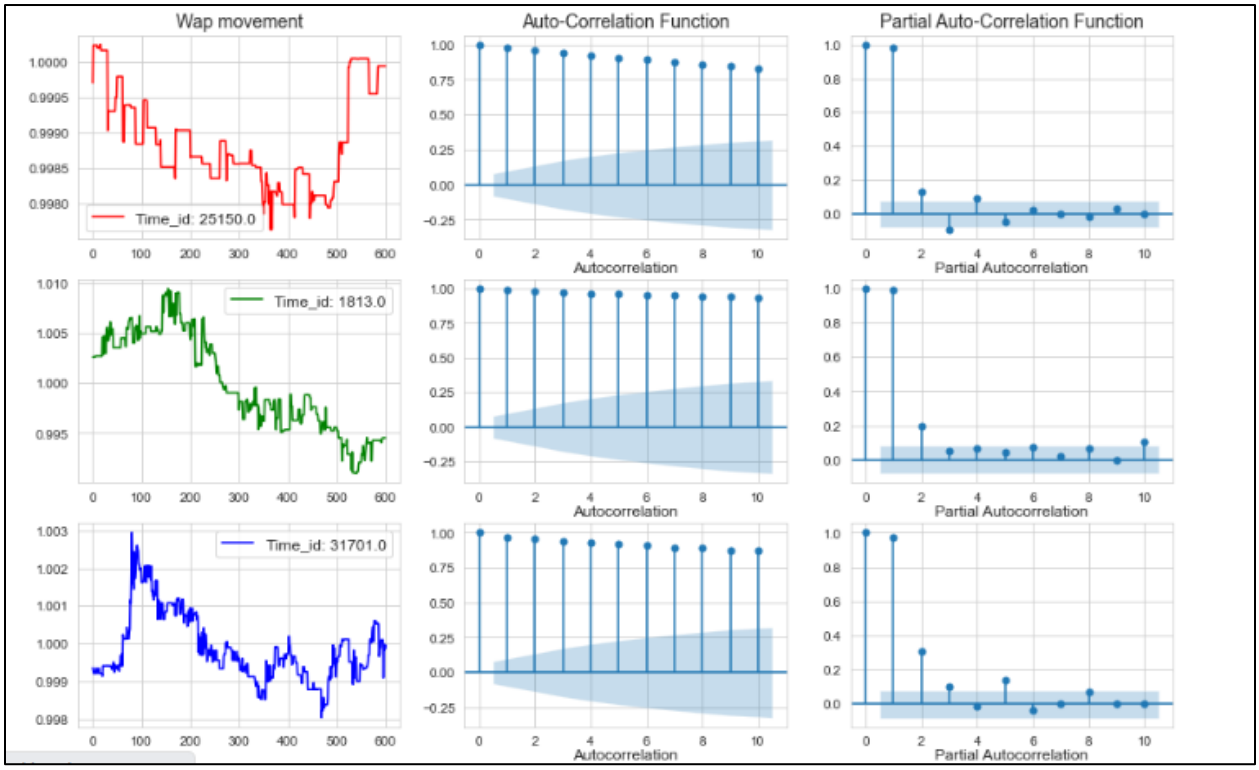


Fig 4.2: The ACF and PACF plots for WAP

The plots in figure 4.2 show that the ACF for WAP trails off very gradually, while the PACF cuts off after varying lags. So, the WAP movement may not be so easy to model using AR or MA models.

4.2 WAP and Trade Price Interplay

Next, let's plot the WAP and Trade Price together. As the charts in figure 4.3 show, WAP bounces around a lot more than the Trade Price. It should be noted that the trades are not happening at successive seconds, so the blue line only connects the trade prices that could be apart in time. No trade price exists in the intervening seconds.

From the chart one can see that sometimes, the trade price is above the WAP, other times below and yet other times they are very close to each other. It is difficult to gauge which one drives the other. But clearly, there is a lot more activity in WAP as compared to the trade price.

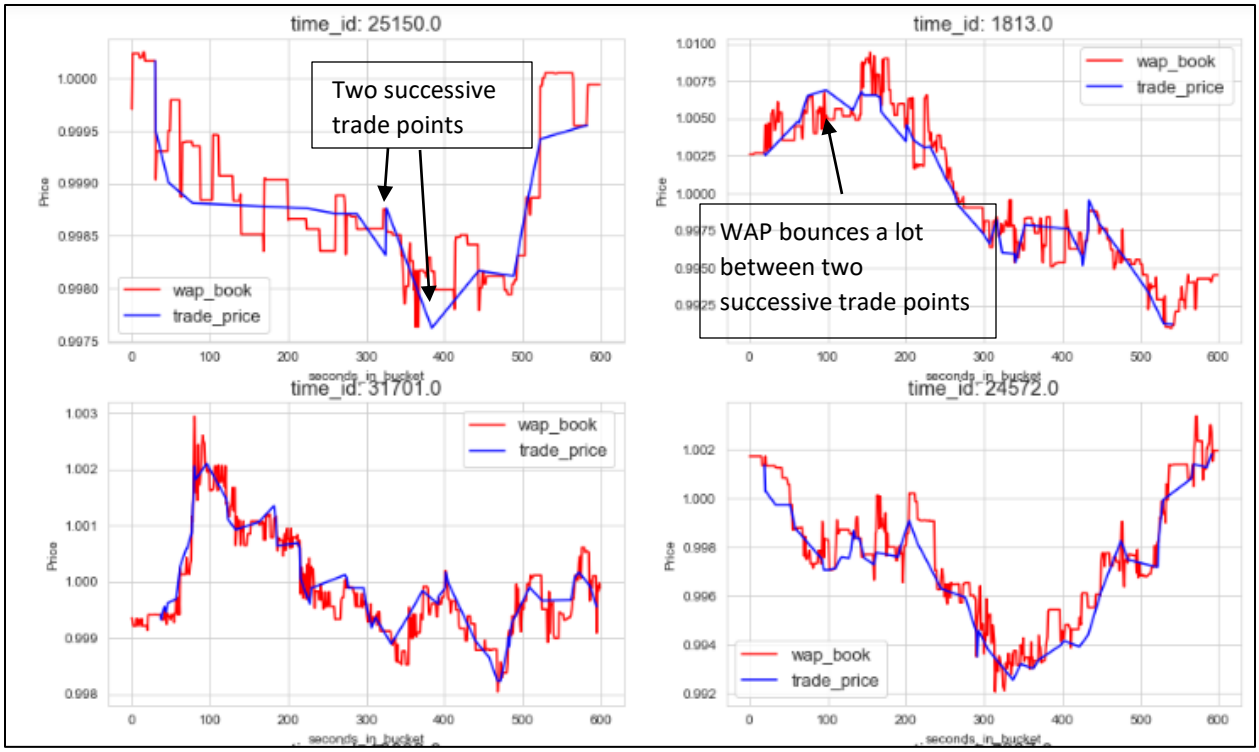


Fig 4.3: WAP and Trade Price movement

4.3 Understanding distribution of basic features

Another important question to answer is the distribution of different variables.

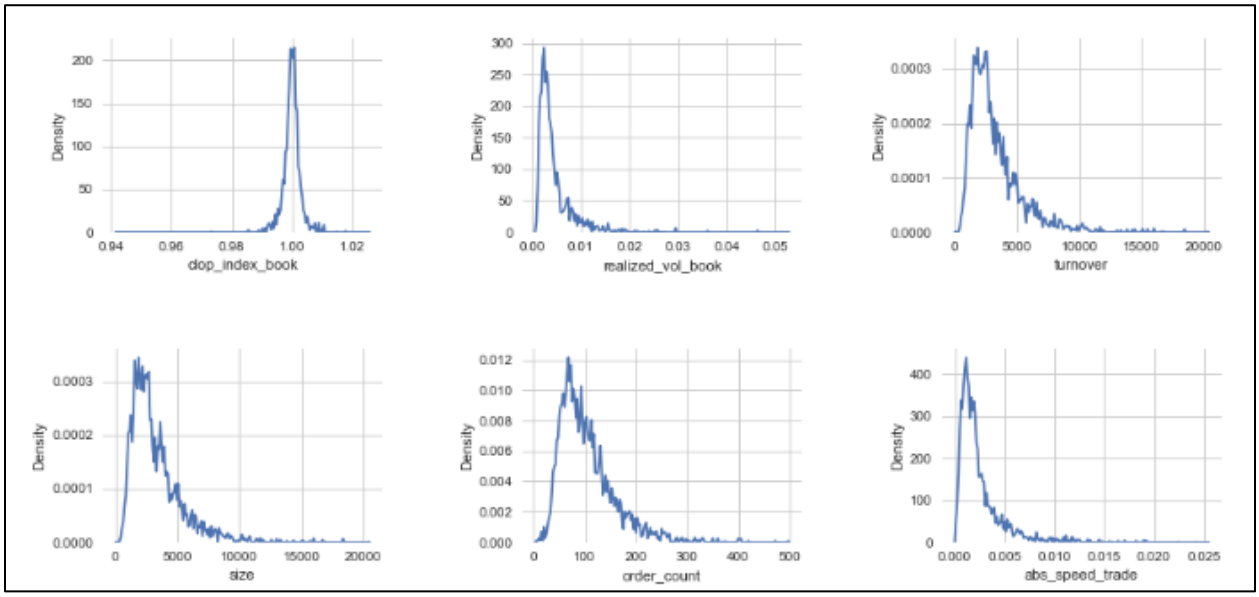


Fig 4.4: Density plots of a few variables

Some observations from the density plots in figure 4.4 are:

1. All the distributions other than clop_index_book are **right tailed**. Clop_index_book is a ratio of the closing wap price to the opening wap price. The right tail in other is understandable as a the lower end, these variables cannot go below 0, but there is no upper limit.
2. 'realized_vol_book' (realized volatility of the wap) has a very high density of approx. 300 at the central value. The same goes for 'abs_speed_trade' which is the speed of trade price movement. The concept of speed will be explained in a later section on feature engineering.
3. Some distributions look normal, so let's use some statistical tests to check this.

One technique is to generate the qqplots and another one is the Kolmogorov Smirnov (KS) test.

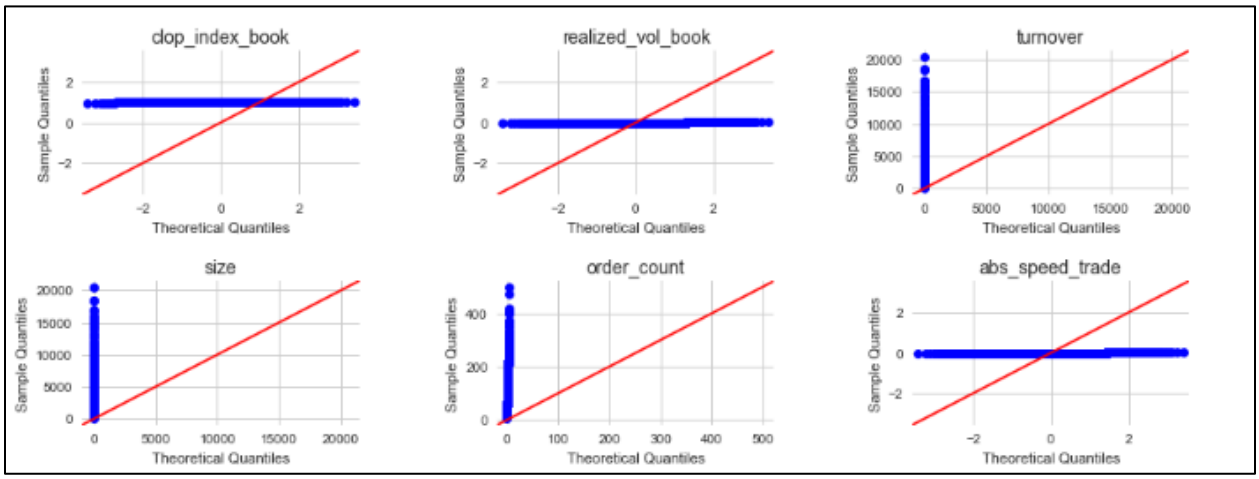


Fig 4.5: qqplots of some features

For normal distribution, the sample quantiles should track or be close to the theoretical quantiles of the distribution (red line). We see that for all the variables, this is not the case. The KS confirms the same answer, with p-values for the test being 0.00 for all variables. So we do not assume normal distribution for any of the variables.

4.4 Interplay of features and target

Next, we focus our attention on understanding how connected the different features with the 'target' are. The scatterplots between single feature and target give a good visual insight into the interplay of the feature and target. One can clearly see from figure 4.6 that realized_volatility_book and abs_speed_trade track more closely with the target as compared with the other features. The Pearson correlation coefficients, as shown on the figure confirm that. The target's correlation is weaker with features like size and order_count.

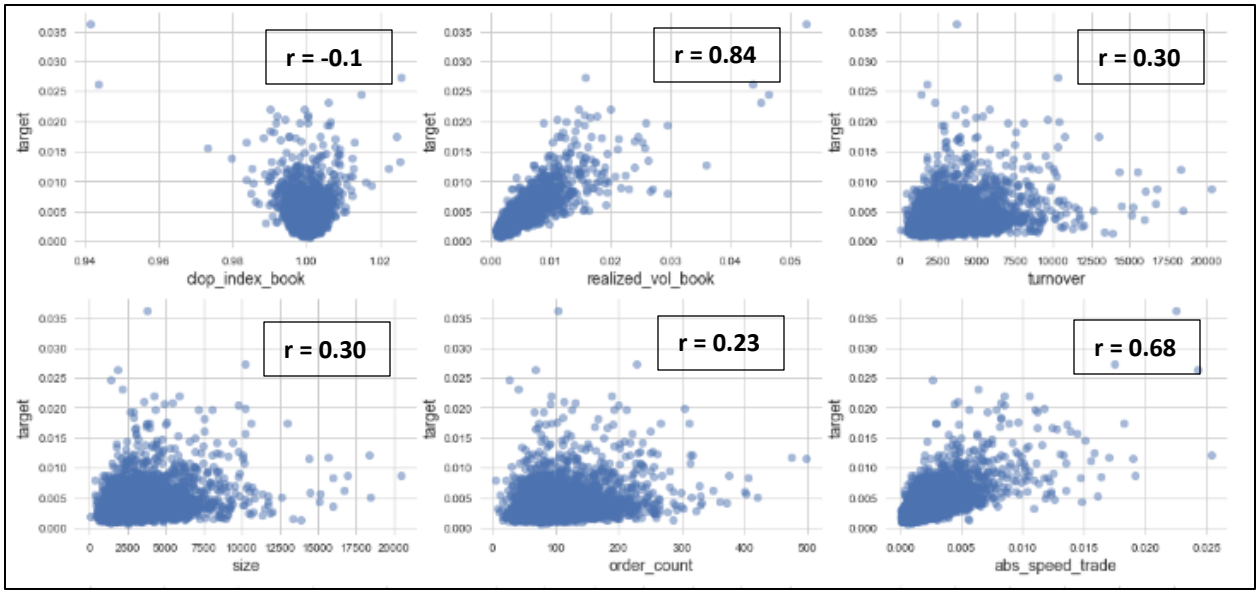


Fig 4.6: Scatterplots between single feature and target

Interesting observation: The '**conical**' shape of the chart between realized_volatility_book and target. It suggests that as the realized volatility of current 10-min period goes up, the range in which target could fluctuate also increases.

4.5 Is the volatility of different stocks correlated?

Another aspect to gather understanding on is the behaviour of different stocks on volatility. Are there any groups or clusters of stocks that show similar behaviour?

Figure 4.7 shows the heatmap with correlation coefficients in the boxes for each pair of stocks. Some observations from this chart are:

1. There many stocks that have a strong correlation (**r** > 0.7) on volatility.
2. The correlation of stocks to the market average (taken here for 30 stocks) is even higher, with **r** greater than 0.85 for many stocks.
3. One can see signs that there could be clusters. For e.g.
    a. Stock_1 has extremely strong (**r** >= 0.85) correlation with 6 stocks namely, stock 26, 95, 101, 102, 116 and 126.
    b. It has moderate correlation (**r** < 0.7) with 4 stocks, namely stock 58, 80, 81, 110.
    c. And with the other stocks it has somewhat strong correlation (0.7 < **r** < 0.85)

These observations suggest that it might be worthwhile to identify clusters of stocks that show strong affinity on an important variable like realized volatility. So next we will do a clustering analysis of all 112 stocks using KMeans clustering technique.

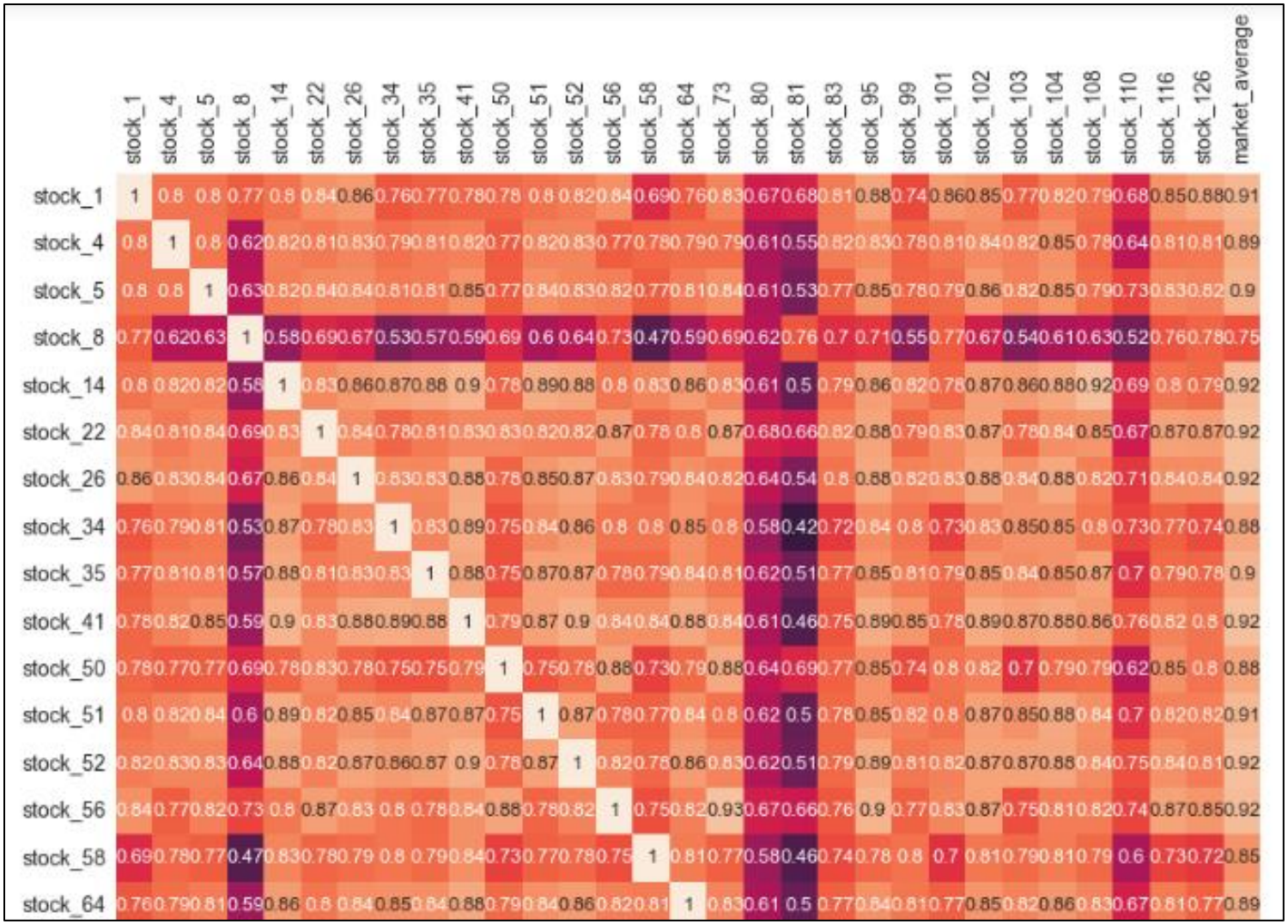


Fig 4.7: Heatmap showing correlation in volatility between some stocks

4.6 Stock Clusters

The sum of squared error chart (Fig 4.8) made by clustering stocks on realized volatility shows an elbow on 3 and 4 clusters.

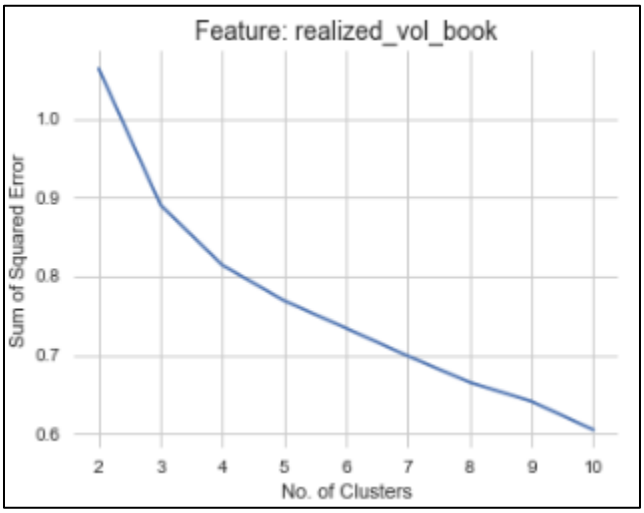


Fig 4.8: Sum of squared error for stocks clustered on realized volatility

The next figure (Fig 4.9) plots these clusters (3 and 4) by extracting the two principal components axis. Visually, it seems like one large cluster in the center with many outliers, but I guess we can live with 3 or 4 clusters as the central cluster is being neatly sub-divided into 2 or 3 and it doesn't seem like over fitting too much.

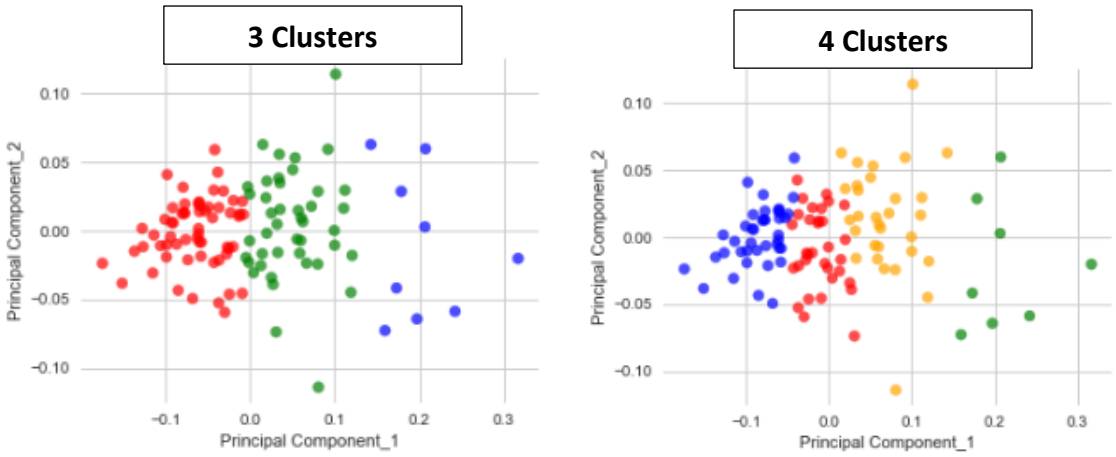


Fig 4.9: Stock clustered into 3 or 4 clusters on realized volatility

4.7 EDA – Key Learnings

The key learnings in the EDA exercise are summarized below:

1. WAP is a random walk, it's log returns are stationary
2. WAP & Trade Price closely track one another, but it is difficult to say which one drives the other
3. Features derived from the data, such as realized volatility of current 10-min period, trade size, high-low ratio of wap etc. are not normally distributed
4. 'target' – the realized volatility of future 10-min period is strongly correlated to features like
    a. 'realized_volatility_wap' with r = 0.84
    b. 'abs_speed_trade' with r = 0.68
5. 'target' is has weaker correlation with to features like
    a. 'size' with r = 0.30
    b. 'order_count' with r = 0.23
6. 'abs_speed_trade' with r = 0.68
7. Many stocks show strong correlation with each other on realized volatility. The 112 stocks can be clustered into 3-4 groups based on realized volatility feature.

5.0 Preprocessing & Modelling

The first step in the preprocessing was to generate a rich collection of features based on the data in the book and trade tables. Towards this objective, I first studied various indicators used in the technical analysis of stocks. There is a very large no. of such indicators present but most are used for analysing stock prices and trends, not volatility. A very often used measure of volatility is the VIX index and it is calculated using options prices. These are not available in our case.

However, based on the learnings gathered from the technical analysis study and the insights from the EDA, I created a whole host of features that can be broadly grouped as follows:

1. **Basic Features**: simple features from book & trade files
    a. wap1, wap2, spread, price premium, realized volatility, turnover
2. **Features based on concepts of motion**: Capture movement dynamics of price
    a. speed, acceleration
3. **Based on technical analysis**: Capture magnitude & inertia of price activity
    a. high-low index, momentum (size * speed) , force (size * acceleration)
4. **Cluster Features**:
    a. Cluster (categorical feature), avg/std of realized volatility of a cluster
5. **Some of above features across time-bands**: Divided the 10-minutes into
    a. Two 5-min halves (H1 & H2)
    b. Four 2.5-min quarters (Q1 – Q4)

Once the features were generated, I scaled the numeric features using Robust Scaler, as there were significant outliers in many features.

5.1 – Modeling

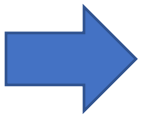
Linear Model

Given that the target is a continuous variable, I first modeled it with a Linear Regression model.

However, before creating the model, one must remove collinearity & multicollinearity.

1. To remove collinearity, I generated the correlation heatmaps for the numeric variables and removed close to 90 features that had Pearson correlation coefficient r > 0.95.
2. Multicollinearity was tackled using the variance inflation factors. I iteratively removed features that had vif > 10, while ensuring that features that were important such as realized_volatility_wap1 were not removed.

VIF Factor	features
inf	vol_diff_from_cluster
inf	acc_sum_mean_minus_H2
inf	log_momentum_book_sum_H1
inf	realized_vol_wap1_H2
inf	log_momentum_book_sum_H2
inf	rv_mean_minus_H1
inf	acceleration_trade_price_mean
inf	rv_mean_minus_H2
inf	log_momentum_book_sum_Q1
inf	realized_vol_wap1
inf	log_momentum_book_sum_Q2



VIF Factor	features
9.512287	momentum_trade_mean
9.472200	bid_size1_mean
8.983602	ask_size1_mean
8.822655	realized_vol_wap1
8.491537	value_premium_1_mean
8.203072	price_premium_1_sum
6.794630	seconds_active
6.640426	acceleration_trade_price_mean
5.810010	value_premium_1_std
5.505904	realized_vol_wap1_cluster_mean
5.010561	force_trade_std
4.974507	order_count
4.726221	wap1_hl_index

Fig 5.1: Removing multicollinearity using variance inflation factors

Post removal of multicollinearity, there were 25 features left. These were used to train a Linear Regression model. The model had an RMSPE score of 0.**2833** on the validation set. The model helped in understanding feature importance by analysing and ranking the magnitude of the coefficients of various features. The top 10 features were as shown in Fig 5.2

Coef	weights
realized_vol_wap1	2.069792e-03
realized_vol_wap1_cluster_mean	5.924311e-04
price_premium_1_sum	2.559763e-04
trade_rv_mean_minus_H1	1.260712e-04
wap1_hl_index	8.742610e-05
momentum_book_sum	8.278138e-05
force_trade_std	3.483076e-05
value_premium_1_mean	1.258734e-05
ask_size1_std	9.962202e-06
wap1_mean	8.058074e-06
bid_size1_std	4.972991e-06
wap_diff_mean	7.673287e-07
const	3.997996e-19

Coef	weights
log_momentum_book_sum	-2.415072e-06
bid_size1_mean	-2.839625e-06
seconds_active	-1.409950e-05
order_count	-1.932193e-05
ask_size1_mean	-2.384914e-05
seconds_active_book	-2.836506e-05
momentum_trade_sum	-3.737814e-05
log_force_book_std	-4.506134e-05
value_premium_1_std	-4.617142e-05
momentum_trade_mean	-9.674820e-05
acceleration_trade_price_mean	-1.226120e-04
rv_mean_minus_H2	-3.354489e-04

Fig 5.2: Feature importances by ranking the magnitude of coefficients

Interestingly, the top 10 features represented most of the concepts used for feature engineering such as stock clusters, technical analysis and motion dynamics.

Neural Network Model

As the total no. of features generated was quite large, next I built deep learning models. I built two kinds of deep learning models

1) **Type 1**: Models based on numeric features only
2) **Type 2**: Models based on numeric & categorical features: Stock cluster ids were used as the categorical features here.

The following hyperparameters of the models were iteratively tuned to improve model performance:

1) No. of dense and embedding layers
2) No. of neurons in each layer:
3) Activation Function
4) Learning rate optimization using "ReduceLROnPlateau" callback

Post tuning, the best type 1 model had a RMSPE score of 0.2125, while the best type 2 model had a RMSPE score of 0.2104 on the validation set. The training and validation loss for type 2 model are shown in Fig 5.3.

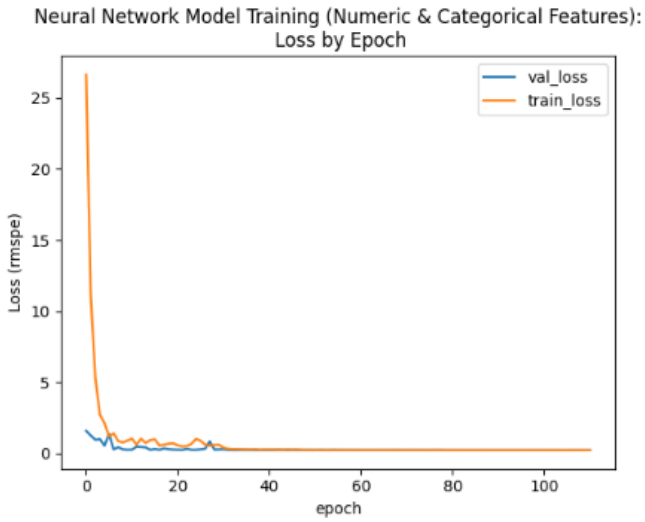


Fig 5.3: Training and validation loss by Epoch for Neural Network Model 2

I then used Model 2 to predict target for the test set, this was 20% of the total data not seen by the model during training. Model 2 had a RMSPE score of **0.2154** on the test set and this score ranks among the **Top 30** score for the Kaggle competition.

Model 2 has a fairly simple architecture as below and provides a fast compute performance of 0.018 seconds to predict the target for all 112 stocks.

1. No. of dense layers : 3
2. No. of embedding layers: 1
3. No. of neurons in dense layers: [layer 1- 200, layer 2 - 200, layer 3- 100]
4. Activation Function: LeakyReLU
5. Batch size: 128
6. Learning rate: Used “ReduceLROnPlateau” callback to iteratively reduce LR to 1e-06.

Light GBM Model

Next I built a tree-based model with Light GBM, as it is known to provide fast computation speeds. To improve performance, I tuned hyperparameters such as learning rate, feature_fraction, max_depth, min_data_in_leaf. The Light GBM had an RMSPE score of 0.2335 on the validation set, this was not as good as the neural network model 2. This modeling approach also helped in understanding feature importance. As shown in Fig 5.4, the top 20 features again had good representation of all concepts used in feature engineering.

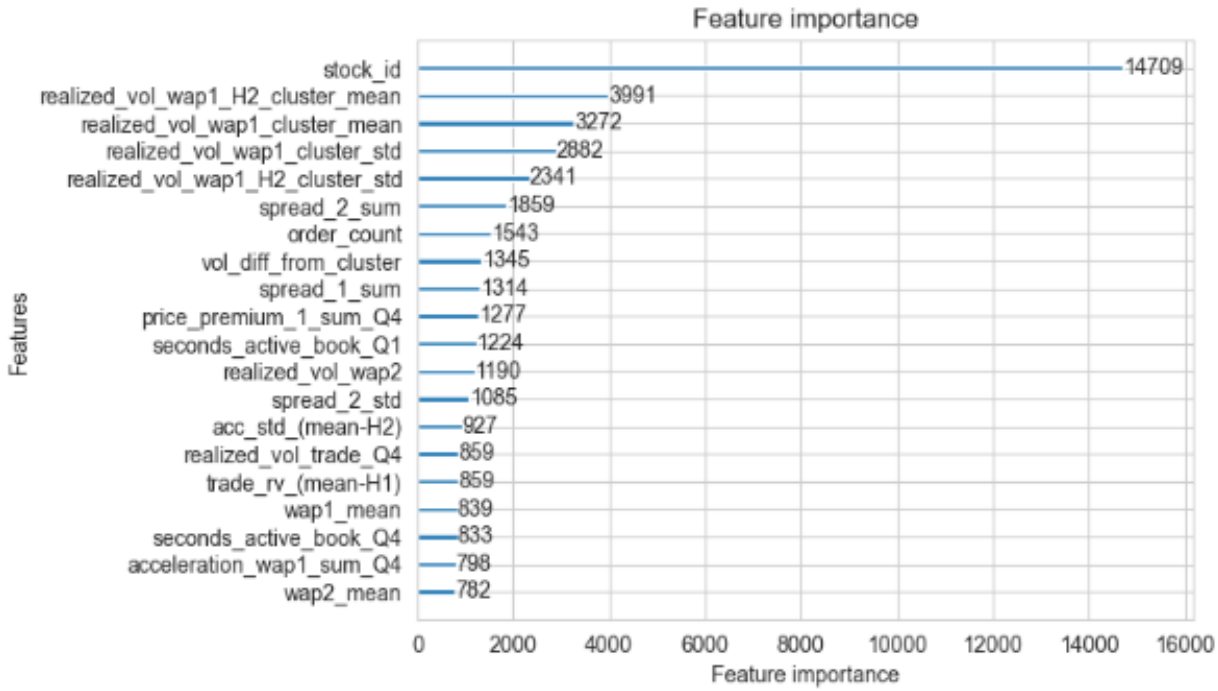


Fig 5.4: Top 20 features for Light GBM Model

5.2 Summary – Modeling

The key learnings and outcomes from the preprocessing & modeling step were:

1. Features based on technical analysis, price motion dynamics, clustering and time-bands are useful for modelling future 10-min volatility
2. Features based on both book and trade data are important
3. A deep learning model based on numeric and categorical features delivers a strong performance with an RMSPE score of **0.2154** on the test set
4. In this deep learning model, 2 categorical features were used in addition to approx. 170 numeric features

6.0 Business Applications

The deep learning model has a simple architecture and strong computational performance

- 3 hidden layers and 1 embedding layer
- 100-200 neurons per hidden layer
- Takes just 0.018 seconds to make predictions for all 112 stocks

As such, it suited for real time applications like stock options trading and can be deployed by electronic market maker firms such as Optiver.