**Introduction/problem background, problem definition**

Within finance, a prominent problem is the pricing of options. Empirically validated models such as Black-Scholes tell us that if you know the volatility of a stock, then the model allows you to correctly price an option on that stock. This makes volatility prediction a significant question in finance, so the core question our project seeks to answer is that of volatility prediction. In contrast to stock prices, stock volatility is more predictable (Brownlees 2011), making this a feasible project.

Volatility of a stock at time t is defined as:

A common approach to volatility forecasting is to use time-series models like GARCH (Engle 2007). The goal of this project is to explore the use of machine learning methods as an alternative. The Kaggle’s Optiver Realized Volatility Competition dataset contains roughly 6 months of historical time-series data at 1-second increments, consisting of 126 different stocks with 13 features for each stock at each time step (full data description in appendix). We propose a machine learning approach combining unsupervised and supervised models to predict volatility 10 minutes out, based on the preceding 10 minute window of time-series data.

**Data Description/Processing:**

This dataset offers detailed stock market data, including order book snapshots and executed trades, with a high resolution of one second. The hidden test set contains data to predict around 150,000 target values. This is a temporal dataset, so it will be split in terms of time.

Training: We will use the earliest 80% of the dataset for training.

Validation: We will withhold the final 20% of our dataset to fine-tune our model and parameters.

Testing: We will use the provided test dataset.

Evaluation: We will use a subset of real-world market data after the period of the training data.

**Methods**

The most notable components of this project are data cleaning/transformation, feature engineering, and actual machine learning model selection and tuning. Feature engineering is the creation of useful inputs to be used in a machine learning model, and can be done using unsupervised methods. To find the useful patterns for volatility prediction, we will apply Gaussian Mixture Model to generate features from the market microstructure data. Next, random forests will be used in feature processing and volatility forecasting by carefully considering the time-series nature of the data.

**Potential results and Discussion**

To evaluate the performance of our model, it is crucial to establish a baseline volatility model, which we will then compare to demonstrate the effectiveness of our model. The GARCH model is popular in the literature (Brownlees 2011), so we select this as the baseline. Outperforming the GARCH model will indicate that we have been able to leverage machine learning to find useful relationships that could be used to better price options and other derivatives.

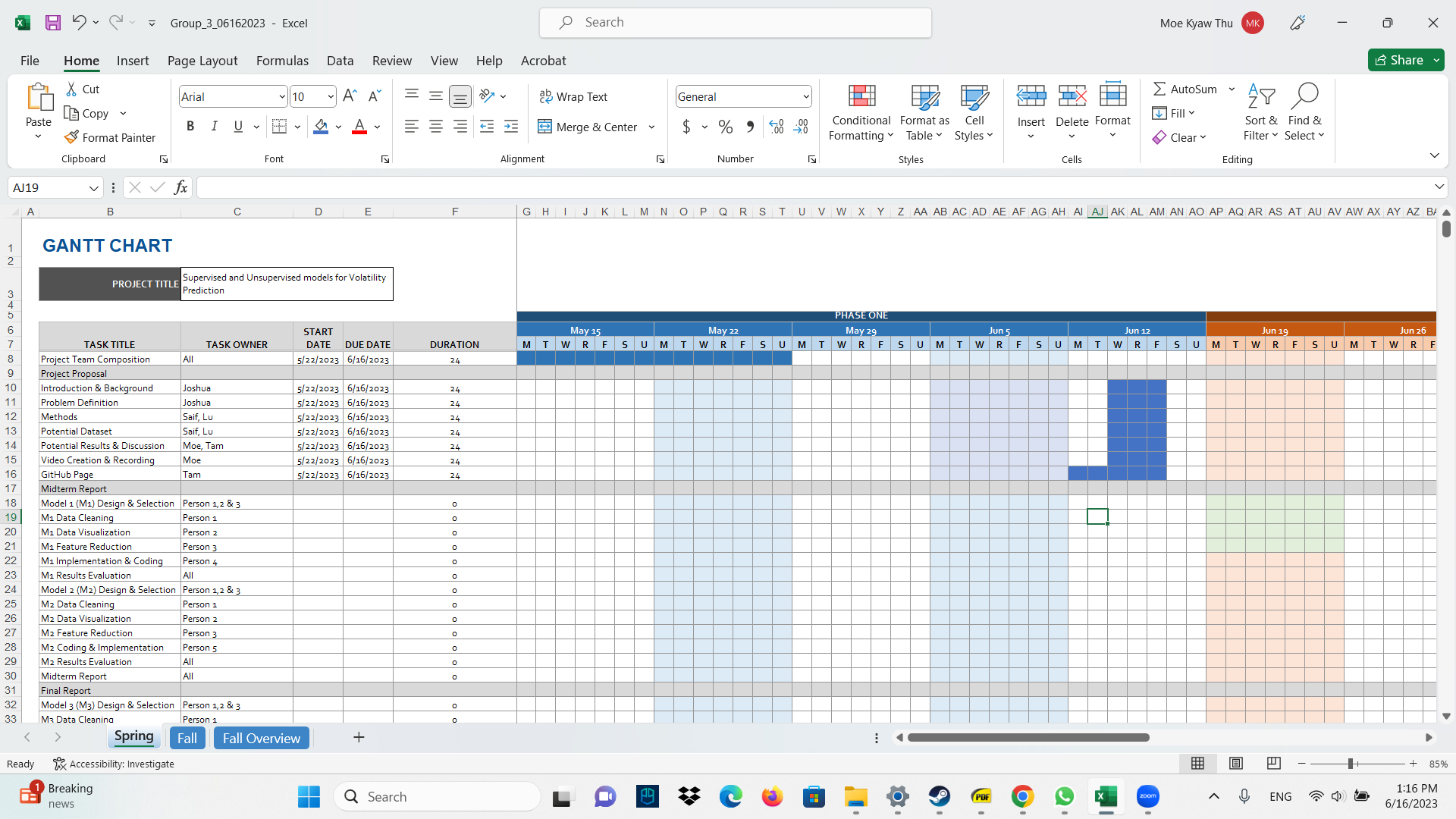
**References**

Brownlees, C., Engle, R., & Kelly, B. 2011. A practical guide to volatility forecasting through calm and storm. In The Journal of Risk (Vol. 14, Issue 2, pp. 3–22). Infopro Digital Services Limited.

Engle, Robert F, 1982. "Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation," Econometrica, Econometric Society, vol. 50(4), pages 987-1007.

Hull, J. 2017. Options, Futures and Other Derivatives. 10th Edition, New York: Pearson Education.

**Contribution table:**

****

**Appendix**

For more information, please see here: <https://www.kaggle.com/competitions/optiver-realized-volatility-prediction/data>

**book\_[train/test].parquet: Provides order book data on the most competitive buy and sell orders entered into the market.**

* stock\_id - ID code for the stock.
* time\_id - ID code for the time bucket.
* seconds\_in\_bucket - Number of seconds from the start of the bucket, always starting from 0.
* bid\_price[1/2] - Normalized prices of the most/second most competitive buy level.
* ask\_price[1/2] - Normalized prices of the most/second most competitive sell level.
* bid\_size[1/2] - The number of shares on the most/second most competitive buy level.
* ask\_size[1/2] - The number of shares on the most/second most competitive sell level.

**trade\_[train/test].parquet Contains data on trades that actually executed.**

* stock\_id - ID code for the stock.
* time\_id - Same as above.
* seconds\_in\_bucket - Same as above.
* price - The normalized average price of executed transactions happening in one second.
* size - The sum number of shares traded.
* order\_count - The number of unique trade orders taking place