* **Write a short outline of your approach to the task and any assumptions you made. Explain why you chose your approach with justification.**
* The overall aim of this exercise is to run a regression analysis of the price of a security against the volume and the time horizons present in a sample dataset.
* Since all the data is contained within the ABC\_Level\_One\_Tick\_Data.csv, I have opted just to use this dataset
* First, the dataset is loaded into python and inspected to see which columns are relevant.
* The Volume column is directly present and this will be used. The Time\_Horizon is calculated by using the difference between the MaxTimeHour and Time\_Hour. The price is obtained by using the VWAP column.
* The Volume and Time\_Horizon columns are stored in a training variable and the Price is stored in a results variable.
* For this model, a random forest algorithm is used to analyse how Price is related to Volume and Time\_Horizon.
* The first step in the training requires a split of the data into a training dataset and a testing dataset.
* A fixed number of estimators needs to be stated when building the model and this value is slowly increased from 2 to 100 within the model.
* Once the model has been trained on the training dataset, it is tested on the testing dataset. A root mean squared value is obtained as final results to show how much the trained model deviates from the actual results. The aim is to minimise this value as much as possible.
* With the range of estimators modified, it is found that 40 estimators can provide a sufficiently low root mean squared error such that it does not need to be decreased further. We can reduce the root mean squared error further by increasing the number of estimators, however, it will use more resources without giving a significantly better results. Hence, 40 estimators is acceptable for the final model.
* **Explain how you might expand your approach to have a single model to predict the liquidity costs for any security in the market, rather than one model per security.**

There are three variables considered in this model:

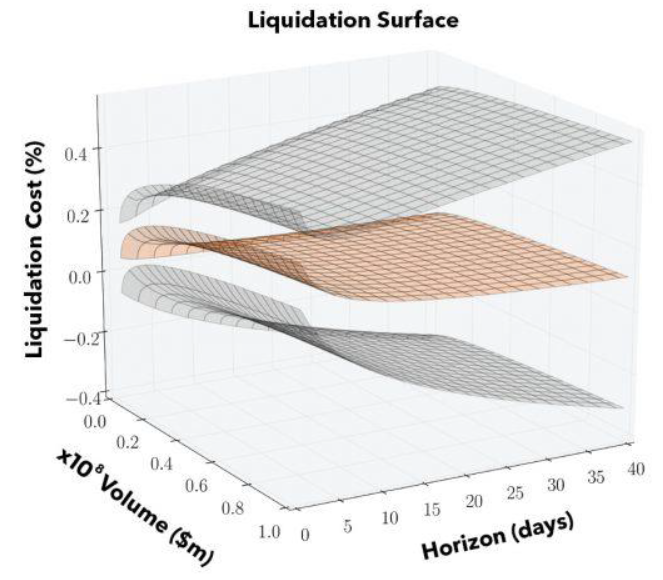
* The price of the security
* The Volume of the security sold within a time slot
* The Time Horizon of the security that is being sold

The volume and time horizon are already independent of a security and can be considered as already standardised variables in the analysis. It is only the price of the security that needs to be standardised. In order to do so, one fixed price of a particular security at a very specific time may be considered as a reference value and all other prices to be considered in the model would be expressed as a percentage of this value. This would ensure that independent of the actual price of the security, it will always be represented as a percentage in the model.

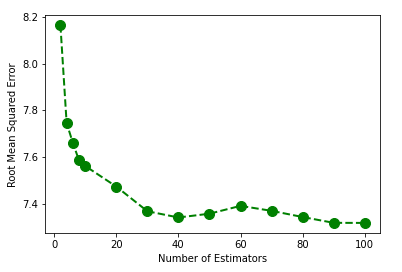
Hence, with all variables present as standardised values, the model can be used universally for all securities. However, this may miss subtle patterns present in certain securities. The universal model will only show high level patterns.

* **Explain how you will present your model to the liquidity business team? Would you create any charts of model results? Would you utilise any visualisation tools in your presentation?**

The presentation will be run in different stages. The first stage will be the rationale. This would encompass a brief introduction of why the whole analysis is relevant and have an overview of the various data that we have available for securities. The second stage would be to introduce the dependencies of the price on volume and time horizon. This stage will be accompanied by the generic graph provided in the introductory pack and shown below:



This will lead to the third stage explaining that we can reproduce such a model for particular securities provided that we have the volume, time\_horizon and price variables. This would lead on to a brief explanation of the random forest algorithm and how it depends on the number of estimators to provide a good model. The final graph that can be displayed is the root mean squared graph variation with the number of estimators and the explanation that we can get a good enough estimate with only 40 estimators. The graph is shown below:



With only one sample dataset, no intricate visualisation is necessary. However, if data from more securities were made available, it would be possible to overlap all the graphs to show a more interesting overall idea or even make an interactive dashboard in tableau where the liquidity business team could themselves select which securities are relevant for them and see only filtered information.

* **Explain how you would run your model using any cloud provider (e.g. Google Cloud) and which cloud components you would use. Give reasons for your choices.**

I would use AWS cloud because it is natively supported by AWS Lambda. Hence, code compatibility is not an issue. The code can be run by fixed time triggers in the AWS Lambda environment. This would also ensure that the cost of running the algorithm is kept to a minimum since resources are not permanently hogged by the code. It is only at the time of these triggers that any resources are used. The code can also directly access live information from other websites as the initial data source and produce an export of the final decision regularly.

* **Briefly describe how you would structure and scale the code of your model if a larger dataset was provided? *(optional)***

Considering that the training is only considering two variables and that the number of estimators for the random forest algorithm is only around 40, the model could be scaled for quite a large dataset without any modifications required.

However, if for some reason the runtime or computation became unmanageable, there are two possibilities that could be explored:

* Forcing the model to only consider a random subset of the data rather than the whole dataset – However, this has the disadvantage of potentially missing part of the patterns in the data.
* Trying to get numpy, sklearn and pandas to run with parallel programming. This can be done by splitting the code into smaller chunks using modular programming concepts. Each job can then work on a parallel thread rather than all of them working on a single core.
* **What changes would you make to your code to make it suitable to run on a production environment? *(optional)***

Currently, the dataset is in the format of a csv which has been provided. In order to make sure that the code is relevant for a production environment, a constantly changing and up-to-date data source may be required. This could be sourced via an API which could take this up-to-date information directly from live data. The results of the algorithm could also be saved with a timestamp so that people could refer back to the results for auditing purposes.