**Methods and Data**

A typical supervised machine learning modelling process is divided into two phases: the training phase and the testing phase. The training phase is the “in sample” phase, where the model processes considerable amount of data which are, in our case, historical time series of different descriptors of the market and make probabilistic predictions on how likely market will crash in the near future in each round. Then these predictions are compared to the corresponding ground truth to provide feedback, enabling the model to “learn” and improve its predictions until the training session is complete. During the testing phase, each time the model is no longer “learning” but simply making “out of sample” predictions which are then recorded for us to evaluate the performance of the model, after all the testing data are processed and predictions are made.

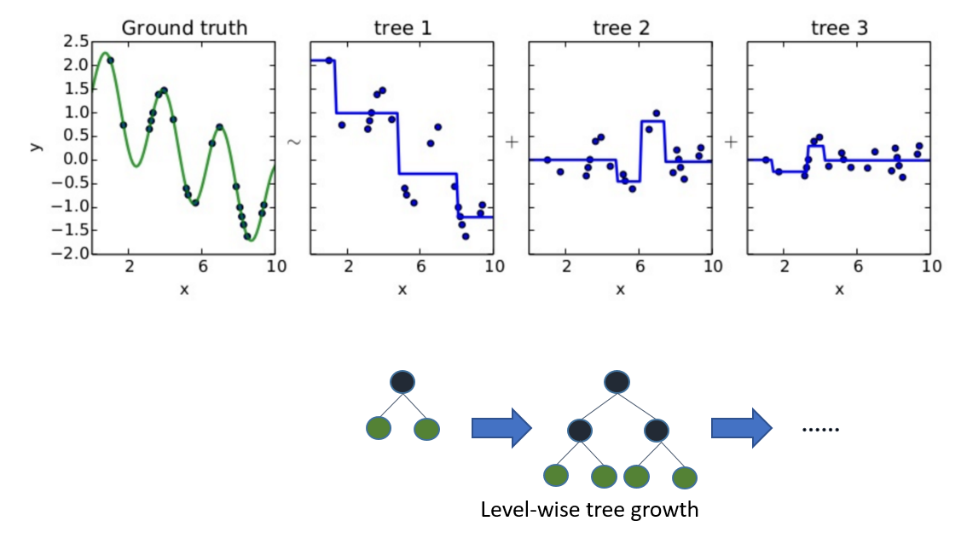
In such a modelling framework and pipeline, two high level preparations need to be made prior to the actual learning, namely, the data preparation and model specification. Our principle in approaching these tasks is to keep it (a) simple and (b) objective.

As outlined in the introduction section, as machine learning techniques have become more and more available in solving business problems, one option would have been to employ the most state-of-the-art modelling framework such as deep neural network or try as many models as possible then shopping around, which is similar to the approach in Chatizis et al (2018). This approach does not serve our purpose of examining and promoting TDA as a novel set of machine-learning-compatible toolkits. Instead, the intuitive exposition of how TDA can be applied to financial time series and profiling the crash landscape in Gidea & Katz (2018) inspires us to systematize its use, by taking an objective and measurable approach.

The rest of this section is organised as follow: we first describe the overall model and hyper parameter specification so as to link it to the findings exhibited in the result section; we then move on to the data preparation, in particular, to show that the feature engineering is objective and disciplined.

**Hyper Parameters**

The modelling framework we choose for our study is the decision-tree-based XGBoost model popular and effective among many data science competitions[[1]](#footnote-1) (Chen and Guestrin 2016). Meanwhile, its tree-based learning algorithm enjoys a simple logic and explicability as illustrated in the model-learning flow chart below:



There is a dozen or so hyper parameters to configure and regulate the model-learning process, for the sake of simplicity, we only focus on two key hyper parameters, namely,

1. *max\_depth*: the maximum depth the tree can grow, and
2. *nrounds*: the number of iterations performed during the training phase.

The first parameter determines the complexity of the final trained model for prediction; whereas the second parameter determines how fitting to the input dataset the model will be trained as. We exhaust a wide range of combination of the two parameter settings for the three market indices of interest (see the Appendix X for this fine-tuning exercise), and finally settle with max\_depth=3 or 5 and nrounds = 5 or 10, to minimise the complexity and risk of overfitting and at the same time consistently achieve stable performance (measured by AUC score).

Having determined the model hyper parameters, we next move to the hyper parameter specification for data preparation and feature engineering.

The raw data for our exercise covers 35 years of weekly price time series from major asset types, namely, Equity, Fixed Income, FX and Commodities, so that major market crashes since 1983 are included for the model to learn as widely as different crash scenarios can vary. Specifically, we want to pre-process these raw data into a tabular form, i.e. a matrix: each row is the weekly snapshot of a chosen market, quantitatively described by a number of market features or columns, so that in total, we will have over 1800 datapoints, multi-dimensional, to train and test the model. We will get to the feature engineering subsection to detail the list of indices and the features to construct.

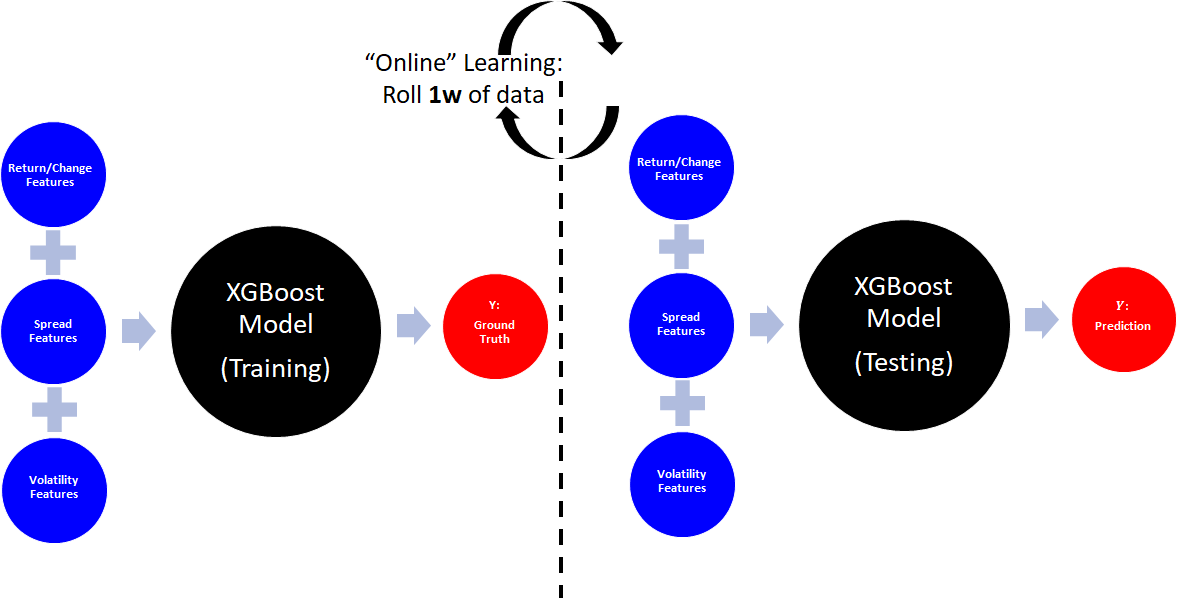
We next define the market crash as over 5% drop of the price of a chosen market index in a time window of 2 weeks. Admittedly, there is a subtle trade-off between data balance and learning the true crash patterns: on the one hand, if we have too few positive samples, i.e. data points of market crash, we will have rather imbalanced dataset to train the model; on the other hand, too low a threshold to define market crash will result in the model learning from noises instead of true crashes. Setting it to be 5% will yield a reasonable portion of positive labels in our learning dataset as shown in the table below, and at the same time, a reasonable magnitude that characterises a proper market crash. We have also considered a dynamic definition by number of standard deviations, however, that again would add to the complexity of the exercise and render the outcome and back testing less intuitive.



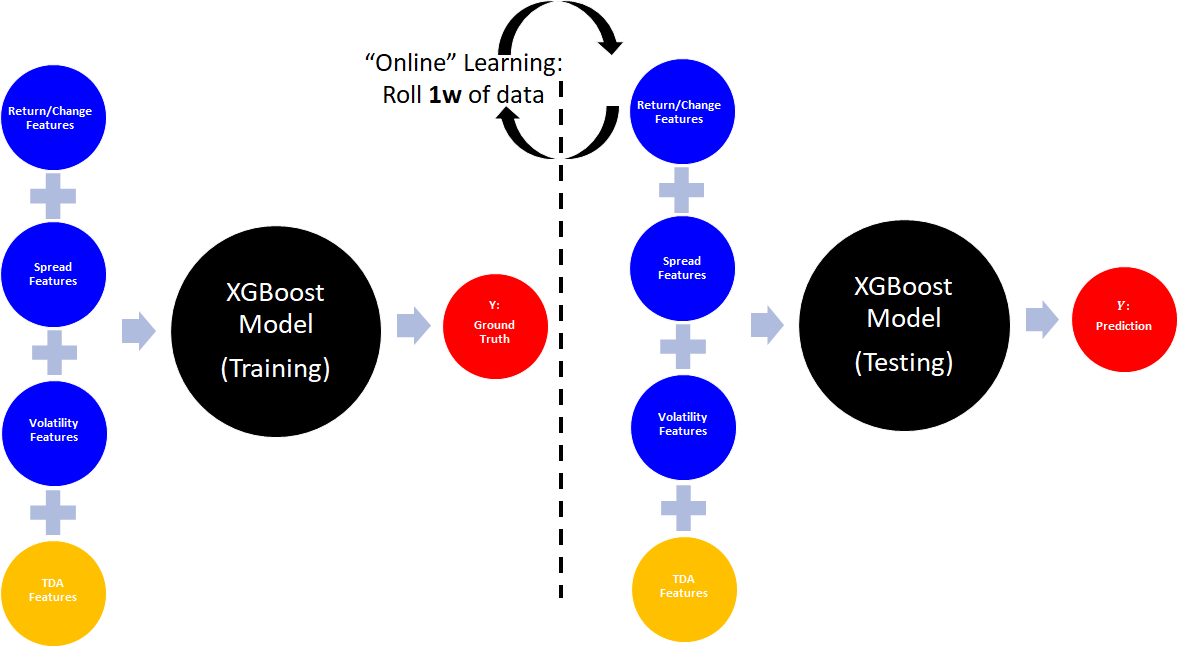
The next hyper parameter to consider is the width of the time window we would like to describe each weekly market snapshot. In principle, this should be proportional to the forecasting window of the crash indicator we build, and we choose 26 weeks, or 2 quarters as a typical market risk model would have a 26-weeks half-life which is similar to what is done in industry (Barbieri et al 2009). However, we do not do exponential-smoothing but only a simple equally-time-weighted approach when it comes to calculate volatility etc., again for the sake of simplicity and intuitiveness.

Having the above setup, we choose the first 10 years of our dataset as the initial training dataset, i.e. 520 datapoints, and make our first 2-week-window market crash prediction. We then expand the training dataset with one week timestep forward at a time. Every time, we re-train the model and make a new 2-week-window crash prediction, until the dataset is exhausted. We then collect the 1314 “out-of-sample” predictions to evaluate this “online-machine-learning” model against the corresponding ground truth, which yields us the results presented in the preceding section, for both the model scores (summarised in terms of Area under Curve and F1) and back testing results. The below workflow charts visually illustrate how we obtain our results for models with and without TDA, and at the same time, prelude the data preparation work which we detail next:

Model without TDA:



Model with TDA:



**Feature Engineering**

Governed by the principle of simplicity and objectivity, as visually illustrated above, the list of features we construct is parsimonious but wide enough to cover the following 3 types for the 4 asset classes the raw data was sourced from, in addition to the 4th “bolt-on” type from TDA:

1. Features of returns or changes
2. Features of spreads or differences
3. Features of volatilities
4. Features of TDA or shape of the time series of each time window

Below are the indices we use to represent the corresponding asset classes:

Equity: one from the three, S&P 500, FTSE 100 or Hang Seng

FI: 2 year, 5 year and 10 year treasuries

FX: DXY Dollar Index

Commodities: WTI Crude Oil and Gold

In particular, we take extra caution to only select one equity index to train the model in each market. For instance, for the US market, we only include S&P 500 in our feature engineering process, FTSE 100 and Hang Seng will be disregarded. Due to the nature of supervised learning, using more than one equity indices to train a model and predict one of them is likely to lead to overfitting, even the time segregation from the prediction/testing phase is strictly enforced, due to synchronicities between indices, information can leak: e.g. the crash of S&P on a Friday afternoon might only be felt and translated in HSI the week after. Therefore, to ensure the exercise is optimised to test the value of TDA techniques in enhancing the performance of crash detection, we only allow the relevant index from the same market in which we are detecting crashes.

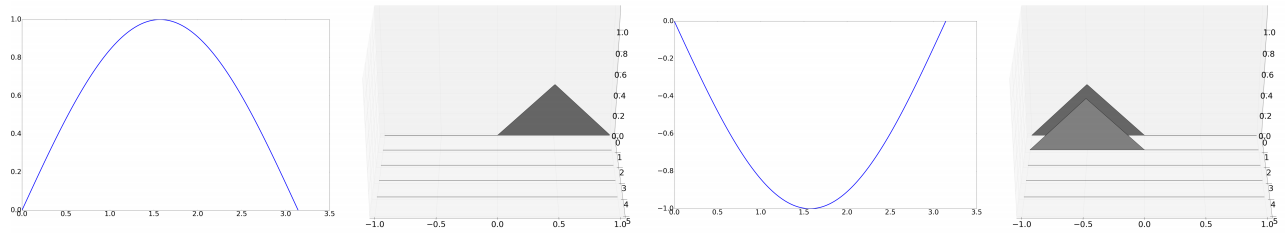
Overall, we construct 15 features to describe the market on a weekly basis covering the past 35 years from 8 April 1983 to 30 November 2018 inclusive. We now discuss how we customise TDA for our purpose. For detailed exposition of concepts such as persistent homology and persistence landscape, there are several excellent overviews (see for example Carlsson 2009, Edelsbrunner and Harer 2010 for a general introduction and Perea and Harer 2015 for applications to time series) , and we refer to Gidea and Katz (2018) for motivation in applying it to financial markets.

Essentially, we compute 3 TDA features, from the 3 asset classes, namely, the chosen equity market index which we also use to detect crashes, the dollar index (DXY) and the oil price.

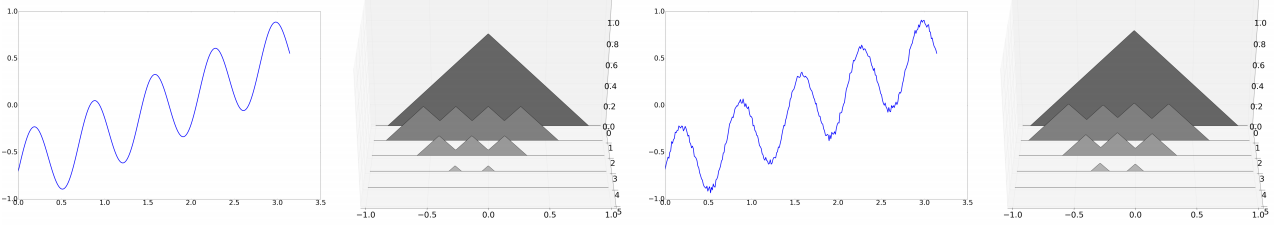
We use the same TDA descriptor, persistence landscape to quantitatively summarise the shape of each time series dataset. However, the key customisation are two folds:

1. To compute the persistence landscape, a filtration function is required. We simply use the top-down reverse height function, instead of the ball-radius used in Gidea and Katz;
2. We then further compute the change of the 1-norms of persistence landscape (PL) of the current time period against the one computed from the previous time period. Effectively, this version of the PL norm summarises the evolution of market landscape and regime shift.

Visually, (1) will return a persistence landscape matrix as below:



And the properties of TDA and Persistent Homology ensure the result is robust against noises:



By employing an easy to use decision-tree-based supervised learning model, and preparing the data most economically as above, we hope to achieve the purpose that such methodology not only make our findings understandable but also adaptable for other problem solving where TDA can become an effective performance booster.

For Appendix:

Hang Seng Index



S&P 500 Index



FTSE 100 Index



1. 17 out of 29 Kaggle competitions were championed by XGBoost based models. [↑](#footnote-ref-1)