Interpreting and unlocking modern forecasting settings with SageMaker and Streamlit

Motivated by stories of customers thriving to solve/unlock the challenges of modelling a complex, modern forecasting setting, while exhausting the imperfect existing data, I found the need of a custom-built solution which can support Scientists and Engineers with these unique scenarios.
The solution approaches the problem from two different spaces, one would explain/explore the inner workings, while the other would find ways to exploit/enhance the characteristics of the data in the form of self-adaptive mechanisms embedded within the problem formulation itself.
The user will benefit from a purpose-built visualisation with a modular and customisable machine learning pipeline and not a single line of code written. It could serve as a blue print for similar repurposed ML use cases as well.

Using SageMaker components for preprocessing, engineering, training, tuning and deploying repurposed algorithms, all orchestrated with a Step Function and seemingly integrated with an ECS hosted dashboard (which includes a Load Balancer and a Cognito User Authentication).

The first task is to generate a pipeline to explain feature importance and accumulated local effects, using SHAP. This generates a set of visualisations abstracted from classical Jupyter Notebooks, using instead a hosted dashboard which retrieves and displays the information in real time.

With a need for using a surrogate model to approximate the predictions of the underlying neural model as accurately as possible and to be interpretable at the same time, the tree-based modules with the popular variant of gradient boosted trees (XGBoost) are often the right choice to compute the full SHAP values. This would enable a comprehensive set of analysis, including global interpretation methods like feature importance, feature dependence, interactions, clustering and summary plots or local interpretability where each feature value is a ‘force’ that either increases or decreases the prediction.

This is achieved using SageMaker Processing with a custom docker image for the surrogate model and the subsequent tasks of specialised SHAP computations.

By capturing tensors while training the core neural model (incl. gradients and weights distributions), a complete understanding is achieved on how the network performs with further insights into how to tune it accordingly. This is built with the use of SageMaker Debugger and a repurposed MXNet-based DeepAR image.

The second task would propose a design of embedded self-attention signals which learn accumulated local effects, group level effects, as well as significantly improve the predictor’s best realisation. This component is relevant for scenarios where: accumulated local effects (which grow over time) shall not be disregarded or attributed to noise; non-stationarity is a known factor (thus explicitly required to be modelled as part of the null hypothesis); group-level effects shall not be lost within the overall predictive distribution; missing data (large gaps) shall not compromise the learning of a progressive underlying process; or the predictor’s best realisation shall not be compromised by the overall predictive distribution.

The self-adaptive component mentioned above is fully contained within a custom SageMaker ScriptProcessor and managed via a number of dedicated parameters, all configurable via the dashboard app and orchestrated with a Step Function.

Next I will discuss three of the modelling approaches used to build the self-adaptive covariates. Firstly, explicit modelling is required to overcome the challenges of missing data, while preserving the integrity of the signal, and to maintain the predictor's best realisation (a univariately built multi-step projection). The proposal is using either spline interpolation, followed by a triple exponential smoothing (where the smoothing parameters are optimised for each series independently) or using Neural Prophet (A Neural Network based Time-Series model), both to generate a synthetic pair for each time-series. Additionally, the selected after-mentioned univariately built model is used to predict n steps ahead (n - prediction length) and thus prepare the series best realisation replica, which is used down the line as a covariate for each target. Secondly, accumulated local effects which are deviant from the general data distribution could uncover a temporary or permanent change within the underlying processes. Therefore, covariates-aided modelling is used to learn where the deviation has occurred as well as what has caused it, as a integrative part of the core forecasting setting. Outliers are detected using a constructed empirical copula to predict tail probabilities for each data point. Quantifying the abnormality contribution of each dimension within a designated covariate for each predictor, will allow the model to focus on relevant subspaces interactions. Additionally, a global “extremeness” score is computed with an Auto-Encoder setting, followed by a heuristic contamination threshold and a user-defined rolling window. This is translated into a covariate variable shared between the predictors with the main objective of conveying data drift. Thirdly, allowing the model to learn item-specific behaviour or group-level effects will significantly improve the overall estimates distribution. This is achieved by grouping the time-series both by a user-specified category, as well as by the cluster it belongs to, after running a K-means time series clustering with dynamic time warping.

In the end, the self-adaptive part serves as a integrative part of the core neural forecasting setting, implemented with DeepAR, a state-of-the-art Autoregressive Recurrent Network for multi-step forecasts of multiple-related time series.The algorithm is a good selection for scenarios where interested in the full predictive distribution, not just the single best realisation, or there is significant uncertainty growth over time from the data, while the data presents widely-varying scales which requires rescaling and velocity-based sampling. DeepAR estimates the probability distribution of a time series’ future given its past, and this is achieved in the form of Monte Carlo samples, by computing quantile estimates for all sub-ranges in the prediction horizon.
In addition, the pipeline offers the option to run hyperparameters optimisation for the DeepAR model, towards the preferred objective metric.

The proposed setting, with the diligence of optimal parameters selection, could significantly improve the overall accuracy for a wide range of use cases, from sales or inventory forecasting to predictive maintenance, allowing for a self-adaptive end-to-end training-inference pipeline which gets better in time. There are no interactions required when the data characteristics change over time, in the form of concept or data drift. Additional predictors can be added or removed from the original setting, with no need for reformulating the problem.

In conclusion, the model interpretability will allow a thorough, in-depth analysis of the data, which helps formulate the problem and select the right set of parameters, while enabling a powerful self-adaptive forecasting setting. The customisation is driven by the specificity of the data and the business performance KPIs, all managed via a hosted dashboard app. The app is implemented using the Streamlit framework with Python scripting and custom widgets, hosted by ECS and packaged with a Cloud Formation.