Project Report: Custom Loss Function to Improve TFT Performance on Seasonal Data

Uday Sapra, Data Science Intern, Promo Analytics A&BC

I. Background:

The current Temporal Fusion Transformers model achieves a weighted SMAPE of 37% across all product categories and adversions. However, the biggest contribution to this error stems from poor performance on seasonal categories {Plot}. Seasonal categories are harder to model as they are highly imbalanced - they have very low or no sales for most of the year and sudden extreme spikes during seasonal weeks.

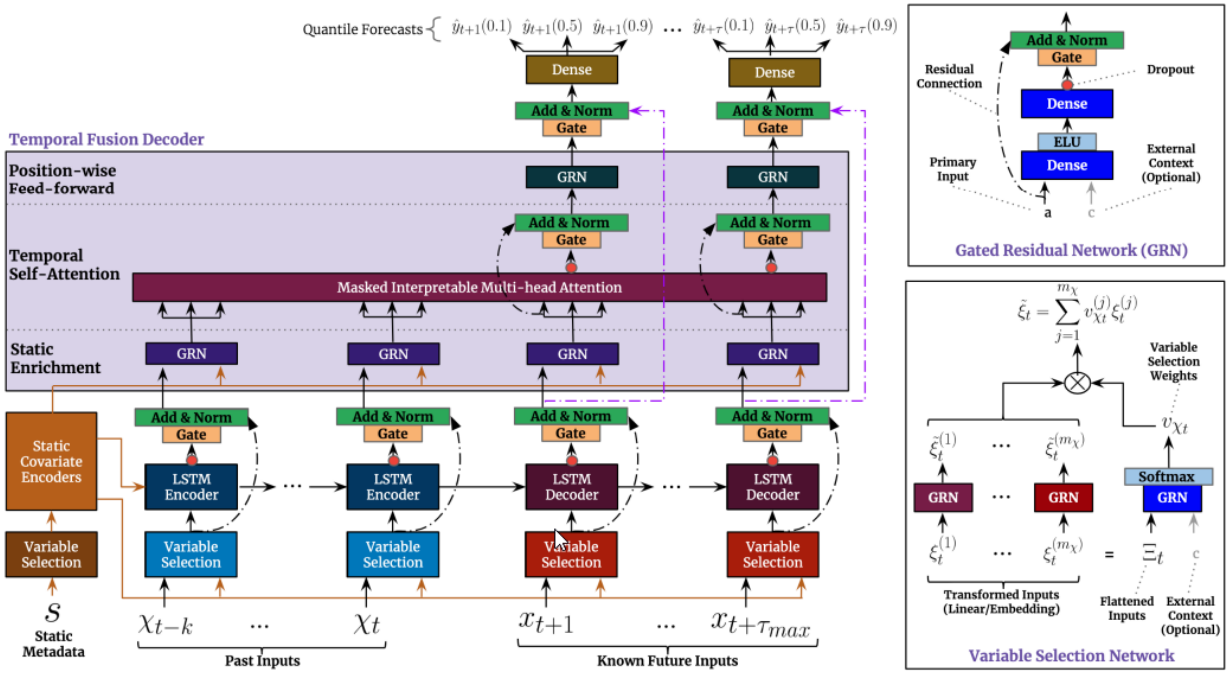
Traditional loss functions do not allow for the best modelling in such cases as they make the model overfit to the imbalance. Ultimately, the models end up predicting 0s or low quantities across - achieving high performance on low sales weeks but ignoring the rare peaks. The current models train on SMAPE, which gives an equal weight to all losses across all weeks and Joint-IDs, irrespective of sales. Intuitively, for better learning of seasonality, we want the model to face a greater loss for poor performance on weeks and Joint-IDs with sales peaks, i.e. seasonal highs. The best way to communicate this requirement to the model would be through a custom loss function. Thus, I was tasked to test and implement different loss functions to instruct the TFT to pay more attention to seasonal highs within seasonal datasets {insert seasonal data plot}.

II. Project Goals:

1. Understand the TFTs and our In-House Trained Model and Code
2. Improve TFT Performance on Seasonal Categories
3. Do this with Low Complexity and Insignificant Increase in Compute Needs

III. Temporal Fusion Transformer:

Temporal Fusion Transformers are attention-based models for multi-horizon forecasting tasks – predicting multiple time-steps in the future based on past context. Contrasting with PatchTST, my last model development, TFTs also allow for multiple input variables which may be static or time-series. The model also allows integration of known future inputs toward better modeling of the target. For its architecture, it is designed around the idea of harnessing all input variables to study inherent temporal relationships. To achieve this, static variables are assigned a separate set of encoders for processing, while for time-series inputs, the model employs both recurrent layers and attention layers. Such a design allows for the interpretation of both short and long-term temporal relationships between all input variables. Given the higher complexity of its architecture, it is able to incorporate more exogenous information into its predictions, leading to higher accuracy over models that rely solely on autocorrelation. Given these qualities of TFT coupled with our access to multiple variables that affect sales, our team employs a TFT based model for high-accuracy retail-data forecasting.



Via the TFT Paper

IV. Project Development

1. **Tools, Libraries, and Specs:** The TFT model is already implemented by the team. We follow PyTorch Forecasting and PyTorch Lightning libraries for implementation. I was given access to this TFT code to develop and test different loss function. For my development, I used TensorBoard to monitor model training and performance across different loss functions. All models were trained with 1 NVIDIA Tesla V100 16GB GPU.

2. **Dataset:** Category 30 Adversion A was used to benchmark.

3. **Hyperparameters:** Unchanged from the current final model.

4. **Model Outputs:** Unchanged from the current model. The forecasting horizon for SCAN_QTY, for context, is 26 future weeks.

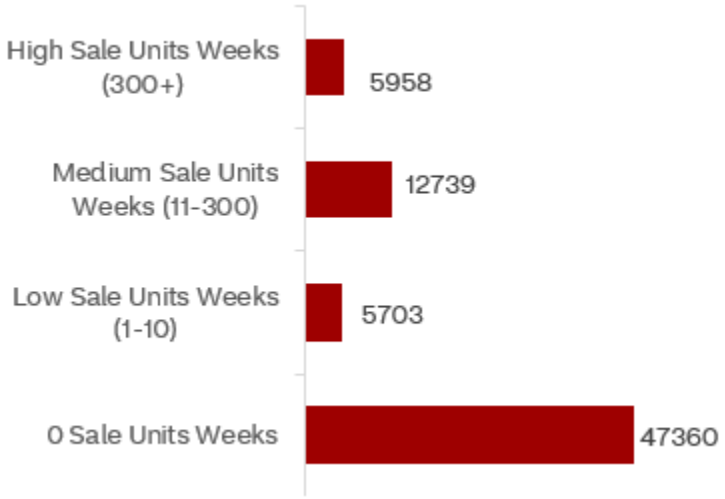
5. **Loss Functions Tested:**

    a. MSE
    b. RMSE
    c. SMAPE
    d. Log Transform on Data + SMAPE
    e. Custom Weighted Loss
    f. Log Transform on Data + Custom Weighted Loss
    g. Quadratically – Rooted and Squared – Custom Weighted Loss

Note: While all functions were tested, only the best performers were reported in this document.

V. Loss Function Requirements:

As mentioned, seasonal categories have a heavy imbalance in SCAN_QTY observations. Most observations across the year are low sales, or even 0s, followed by extreme highs.



With that in mind, the loss function had 3 major requirements:

1. Stable performance on targets and predictions at 0 or close to 0.

2. Higher weight to both –
    a. Weeks with high sales
    b. Product with high sales.

3. Easy integration into the current training process without the need to change preprocessing or training code. This necessitated that the loss function to be an external function, with the only arguments passed being y_pred and y_true.

VI. Loss Functions Discussion:

Below is a commentary on the observations for each loss function.

1. MAPE and SMAPE

$$MAPE = \frac{1}{n} \times \sum \left| \frac{actual\ value\ - forecast\ value}{actual\ value} \right|$$

$$SMAPE = \frac{1}{n} \times \sum \frac{|\ forecast\ value\ - actual\ value\ |}{(|\ actual\ value\ |\ +\ |\ forecast\ value\ |)/2}$$

The MAPE and SMAPE functions are the industry standard to train time-series forecasting models as they provide the error in an interpretable percentage point format. However, these functions are unstable as the targets or predicted values approach 0.

Function	Range	Target at 0 when Predicted not 0	Target Close to 0 when Predicted not 0	Predicted at 0 when Target not 0	Predicted Close to 0 when Target not 0
MAPE	**0 - ∞**	**Undefined**	**∞**	**100%**	**~100%**
SMAPE	**0–200%**	**200%**	**~200%**	**200%**	**~200%**

As can be observed, these functions max out or become undefined if either the target or the predictions are 0 or approaching 0. This means, at edge cases, even extremely small absolute errors lead to the SMAPE and MAPE functions reaching their maximum value. Training is unstable in these edge cases as the model may be heavily penalized for minor errors, signaling unideal modifications to model parameters. Our seasonal datasets contain a lot of these edge-case values, natively. In addition, since inputs are min-max scaled data, a lot of our inputs fall into the edge-case range. Conclusively, MAPE and SMAPE are not viable for our use-case.

2. MSE, RMSE, and MAE

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \qquad RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2} \qquad MAE = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

MSE, RMSE, and MAE all follow similar math. However, have a different scale of output and capture different influences of outliers. MSE yields the largest impact of outliers to the model during training as the errors are squared, leading to a quadratically higher error on extreme values – this aligns the best with our need of highlighting high sales outlier weeks. RMSE is followed by MSE in terms of outlier influence as it first squares the loss and then roots it back. MAE captures outlier impacts the least as no quadratic transformations are involved. The only intuitive drawback of these loss functions for our use case is that they are not interpretable. Given our model outputs a very high dimensional tensor with 26 future outputs for each input across hundreds of Joint-ID, the losses, when flattened, accumulate to extremely high values through the network. These high values are meaningless for interpretation.

However, given the intuition behind a higher influence of outliers, we tested both MSE and RMSE to train the model. The TensorBoard output showed quick training that plateaued learning after the first few epochs. The models however, did not train well as the models minimized loss by predicting 0s across. Conclusively, MSE and RMSE were unideal for our data too.

3. Log Transform of Inputs

a. Transform

```
Target = np.log(1 + Target)
```

b. Inverse Transform

```
Output = np.exp(Output) - 1
```

Vu recommended a log transform on the input data before min-max scaling as the logic had been employed in the TFT paper to improve training by reducing skew.

The model outputs were then inverse log transformed to get the original scale back for model validation. This method improved model training across both SMAPE and custom loss functions over not using the transform. **Hence, the final recommendations in this report use the Log Transform preprocessing step.**

VII. Development of Custom Weighted Loss

1. Goals in Mind

This was the custom weighted loss that was implemented to achieve all –

a) Higher weight on higher sales weeks
b) Higher weight on higher sales Joint-IDs
c) Edge Case Handling
d) External Function with only the y_true and y_pred arguments.

The weighted custom loss, point-wise, is calculated as below.

$$CUSTOM\ LOSS = \frac{1}{n}\sum \frac{(Point\ Wise\ SMAPE) * (Point\ Wise\ Target + \epsilon)}{\sum (Target\ across\ Prediction\ Horizon\ for\ JOINT\ ID)}$$

Note: Other versions of this loss were tested, to scale the output up or down quadratically, however, they did not perform well.

2. Custom Loss Constituents

a. SMAPE: It is used as the baseline loss as it performs the best out of all traditional loss functions. (1)

b. Local Weight: The loss in (1) is multiplied by the Scan_QTY of that time-step (week) to weigh high sales weeks for one particular Joint-ID more. An extremely small epsilon of e^{-10} is added to handle the edge case where the true value is 0 however the model does not predict 0. Thus, loss would still exist however will be very small, this aligns with our business case as we don't

focus accuracy of low-sales weeks. This modification is important as without the epsilon, the model would see no loss for non-zero predictions on 0 targets, sending the wrong signal. (2)

c. **Global Weight:** Result of (2) is divided by the sum of SCAN-QTYs for that particular Joint-ID for the forecast horizon, which is the next 26 weeks. This contextually scales (2) and enables the model to better recognize and model weeks that have an upcoming trend.

3. Edge Case Handling

As evident from the table below, we have handled the edge-case instability of SMAPE and MAPE with the custom function.

Function	Range	Target at 0 when Predicted Not 0	Target Close to 0 when Predicted Not 0	Predicted at 0 when Target not 0	Predicted Close to 0 when Target not 0
SMAPE	**0-200%**	**200%**	**~200%**	**200%**	**~200%**
Custom Loss	0-200% **(Approx Ignoring Epsilon)**	Very Small Loss	Very Small Loss	Scaled SMAPE based on Target/Future	Scaled SMAPE based on Target/Future

There are 3 cases that this function might encounter and here are associated values:

a. All 26 weeks for the Joint-ID have no sales: In this case if the model predicts 0, the loss will be 0. However, if the model outputs non-zero values, we will see a very small loss.
b. All or some of 26 weeks for a Joint-ID have sales: In this case each week's loss would be a scaled loss < 200%. Handling of 0 targets and predicts is universal across each of these cases.
c. Only one week has out of the 26 has sales. For this case, the model will witness he highest loss on the one highest sales week, which can be upto 200%.

4. Summary

This loss function, by being computed at a Joint-ID per week level adds a higher weight to both – Joint-IDs with higher sales and weeks with higher sales. Thus, intuitively, the function assigns a higher penalty to the model for poor performance on Joint-IDs and weeks with high sales.

It is also implemented as an external function and accesses the weights from y_true input tensors itself. The inputs are tensors of dimensions (Joint-IDs in Batches, Forecast Horizon) in batches of 350. Hence, point-wise multiplication, row-wise addition, and point-wise division were the only transformations needed.

VIII. Training Results

Below are the training results for the 4 loss functions alongside the best ensemble-based models – which are discussed later. The results are divided among overall SMAPE and SMAPE on high sales weeks. The threshold for high sales weeks is set at 300 sales due to alignment with business needs.

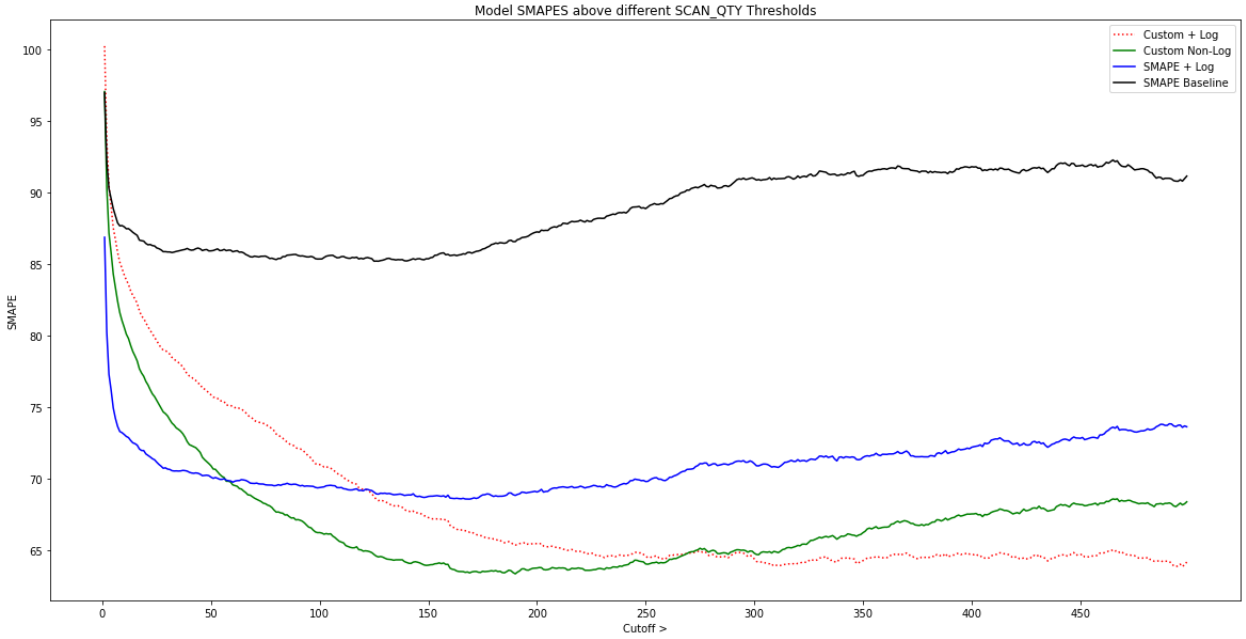
Model Trained on Loss Function	Overall SMAPE	High SCAN-QTY SMAPE	Overall % Improvement	High SCAN-QTY % Improvement
SMAPE Baseline	**39%**	**91%**	----	----
Log Transform + SMAPE	35%	71%	+4%	+20%
Custom Loss	44%	65%	-5%	+26%
Log Transform + Custom Loss	43%	64%	-4%	+27%
Balanced Ensemble	**34%**	64%	**+5%**	+27%
High SCAN-QTY Ensemble	41%	**61%**	+2%	**+30%**

Current Model in Red, Best in Bold, Best Standalone Model Underlined

IX. Analysis of Results

1. Performance Plots

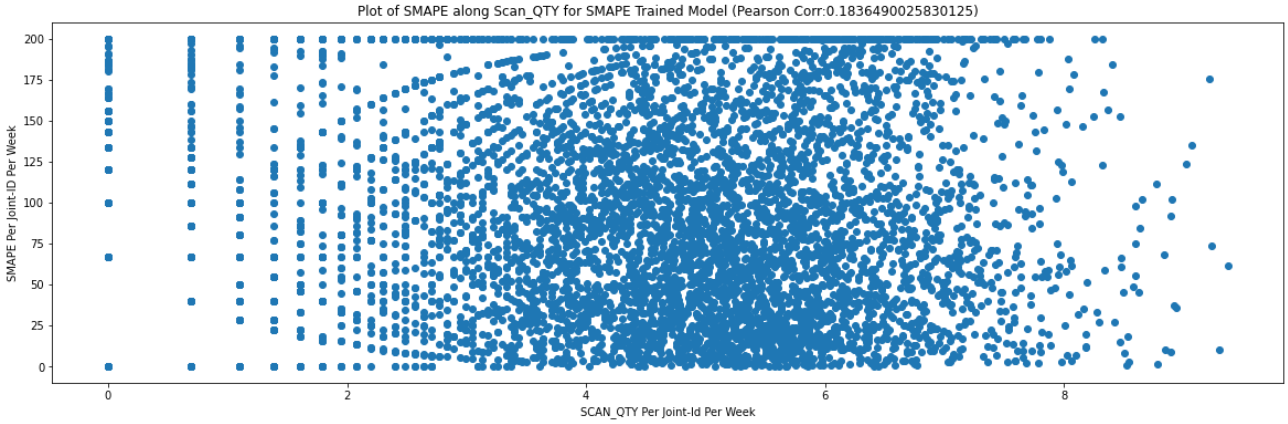
All tested-loss functions outperform the baseline SMAPE model on High SCAN_QTY observations by a minimum of 20%. Conversely, on overall SMAPE, the custom models see a 4-5% decline. This is expected due to two reasons. First, the custom functions tradeoff low-sales accuracy for high sales accuracy and second, the dataset is heavily imbalanced toward low sales observations.



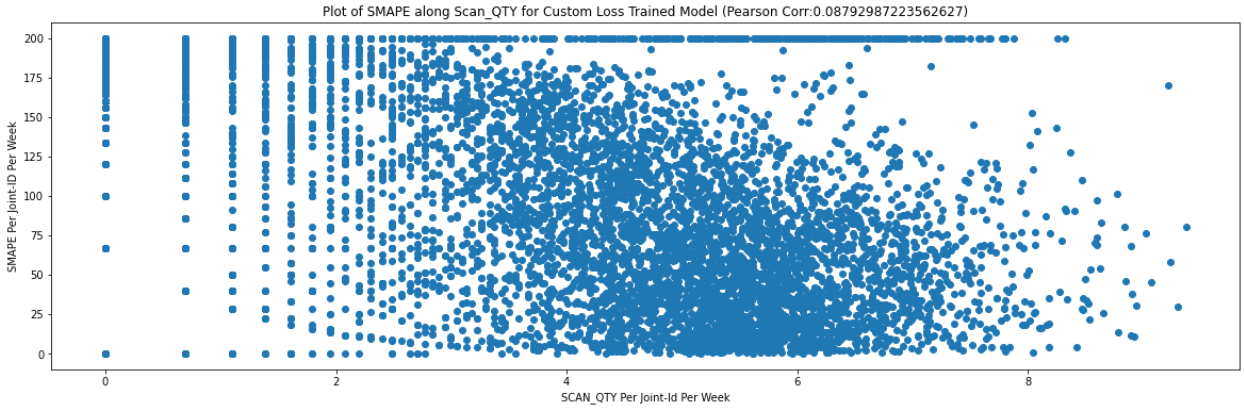
This plot exhibits the SMAPE of all models for different thresholds of SCAN_QTYs. For each threshold, the SMAPE is calculated on SCAN_QTY observations at and above that threshold. As can be seen, custom models significantly outperform the current SMAPE trained model on High SCAN-QTYs – witnessing improvements over baseline at about the 60 sales threshold for the non-log version and 120 for the log transform version. While the SMAPE increases with an increase SCAN-QTY thresholds for SMAPE trained models, the custom models exhibit and inverse relation – which is what we wanted.

For further analyzing model performance, I plotted scatterplots of all test SCAN-QTY observations against their corresponding SMAPE to examine how the relationships compared for different models.

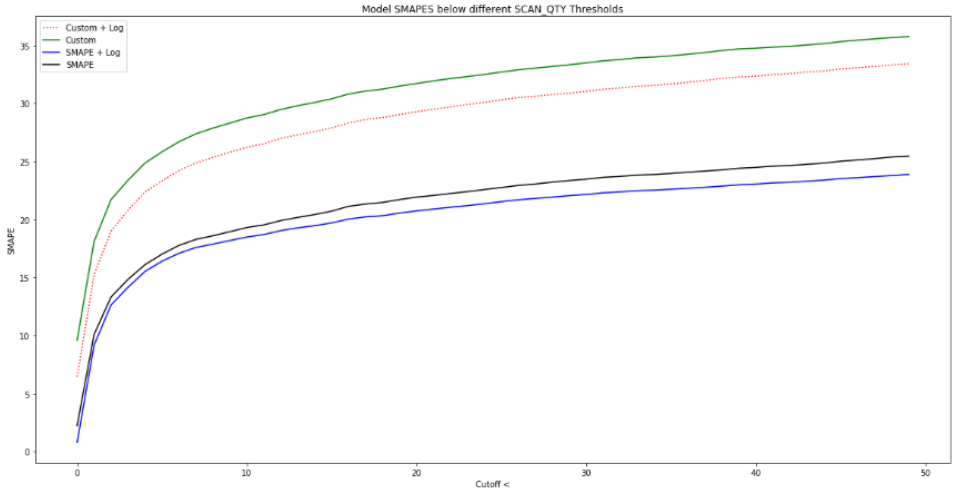
This is the plot of the baseline SMAPE trained model. It shows no visual relationship between SCAN_QTY and SMAPE. SMAPE values seem to high across all SCAN-QTYs.



The custom loss model plots, conversely, visually exhibit an inverse relation between SCAN_QTY and SMAPE. This shows the impact of the weighted loss function – the accuracy of our model improves with an increase in SCAN-QTYs, enabling us to model higher sales weeks better. The correlation co-efficient is still low, but this is due to dataset imbalance.

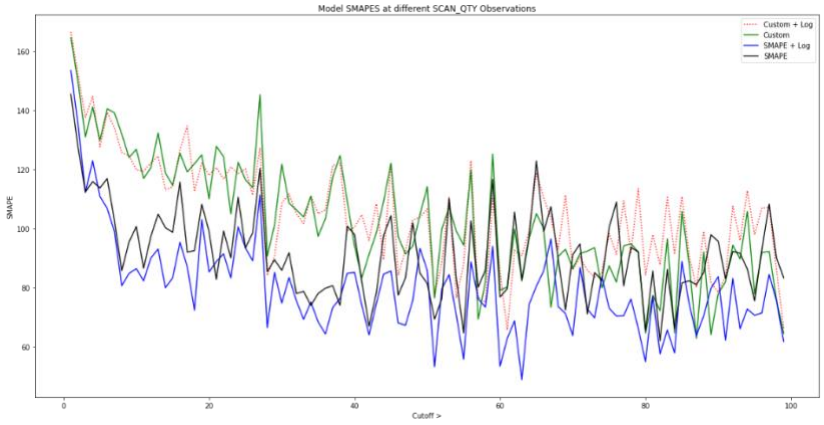


While the accuracy on high-sales weeks has increased, our model has become worse in predicting low to medium-low sales weeks.



This is a plot showing model SMAPE values below and at a given threshold. As can be seen, the SMAPE trained models are significantly better at predicting low sales observations than the custom models.

When we plot observation-wise SMAPEs, we see that especially at lower sales observations, the SMAPE and log-SMAPE functions perform better, however, the divergence keeps decreasing with an increase in SCAN_QTYS.



2. Model Attention Plots

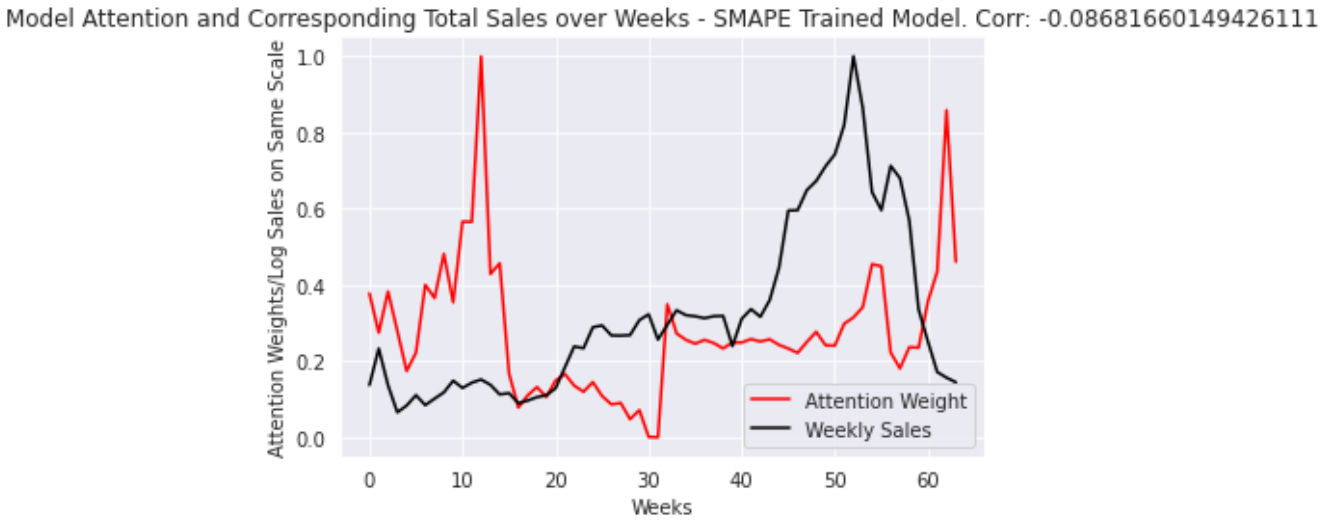
To access the true impact of the custom loss function on model parameters, I analyzed and compared the attention weights of the custom loss trained model with the SMAPE baseline.

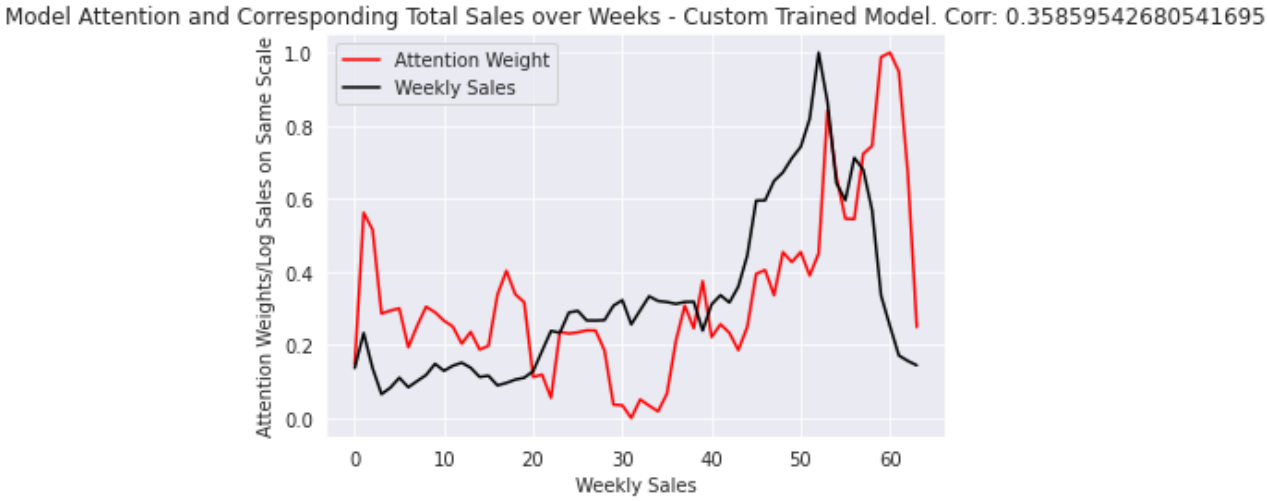
Note: Actual corresponding plots show a lag between weekly sales and attention weights, with attention weights lagging weekly sales by about 20-26 weeks. This is expected as the model computes loss and trains on the prediction horizon of the next 26 weeks at any given time-step. Therefore, the impact of weights is not felt at the actual high sales weeks, but at the 26 weeks preceding them. Consequently, it is not the high sales weeks that will be should higher attention, but the 26 weeks preceding them. The model, any given moment, has insight 26 weeks into the future, hence the increase in attention weights will begin 26 weeks before the actual peaks. For fair analysis, we can shift the weekly sales time-series up by 26 weeks. All plots shown below correct for this lag.

The shift in the model's focus towards high sales periods is strongly seen when we compare the attention weights of the baseline model with the custom model. These attention plots can also be analyzed across two dimensions – as attention distributed over training weeks or Joint-IDs.

a. Attention Weights Distributed Across Training Weeks

The plots below show the attention weights of the baseline and custom loss model across training weeks alongside the total weekly sales for those weeks.





<u>There are 3 points of comparison:</u>

i. The custom model's attention weights correspond closer to the weekly sales plot – with a correlation co-efficient of 0.35 and MAE of 019. This shows that the custom loss model gives more attention to the weeks with more sales. Contrastingly, for the baseline SMAPE model, the correlation coefficient with weekly sales is -0.086 and the MAE is 0.23.

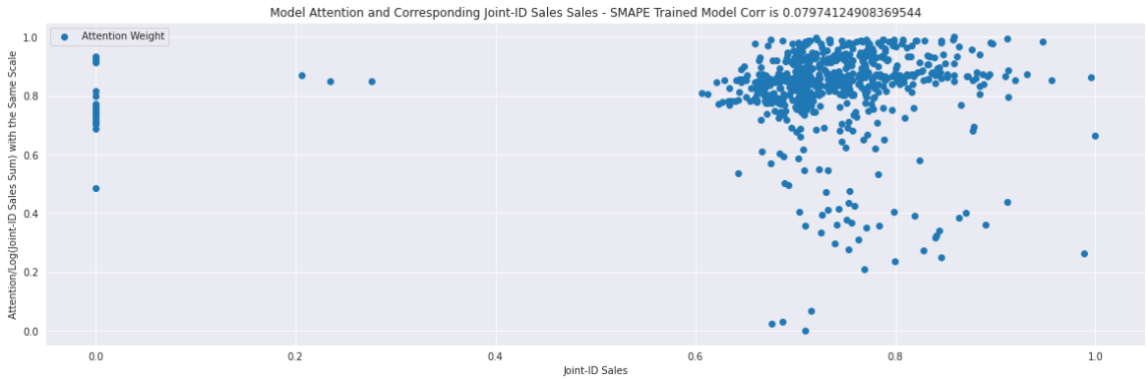
ii. The custom loss model exhibits more stability of attention weights across weeks. The low sales weeks correspond to low attention and the high sales weeks see gradual peaks in attention. However, for the baseline SMAPE

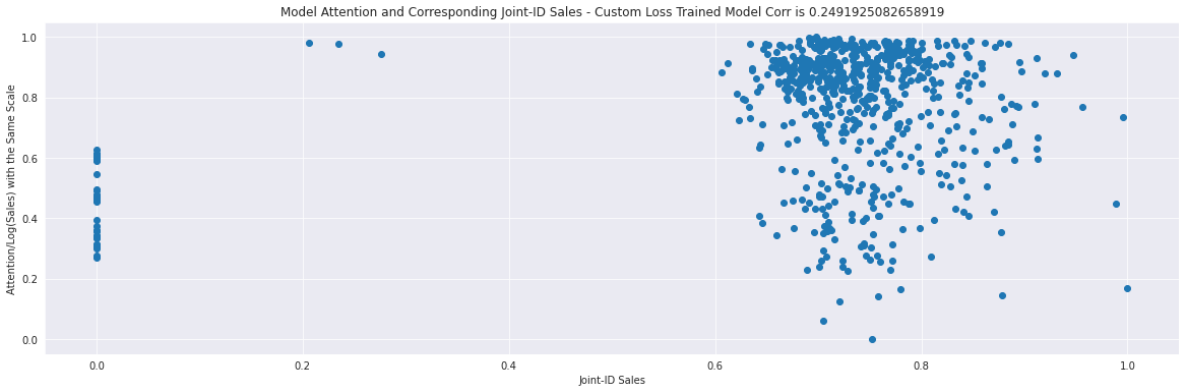
model, we see unexplained peaks even for low sales peaks. This furthers our analysis that the current SMAPE model does not exhibit a relation between total weekly sales and the attention it yields to those weeks.

iii. The custom model accumulates attention with increasing sales weeks while for the SMAPE based model, the attention weights plateau at peaks. This logically stems from the difference in the loss functions. While the SMAPE treats all weeks equal, the Custom loss assigns a higher weight to higher sales weeks within a series of inputs. Resultingly, in the context of upward trends, the custom loss function will scale the loss such that each successive week gets a higher weight until the peak starts to subside. Consequently, the attention plots also seem to follow the trend and increase attention for each successive week in a seasonal high.

b. Attention Distributed Across Joint-IDs

I analyzed the attention weights of the two models distributed across Joint-IDs with the total sales for each Joint-ID.





For the SMAPE model, we see from the concentration in the upper-right, that Joint-IDs irrespective of their total sales are assigned high weights. Resultingly, there is no correlation between total sales for a Joint-ID and the attention weight assigned to it. The Custom Loss model, however, visually improves the distribution of attention weights across Joint-IDs and makes them correspond better to total sales for the Joint-ID with a correlation co-efficient of 0.25. This bolsters the fact that the Custom Model yields higher attention to Joint-IDs with more sales.

Conclusively, having studied the attention weights, we can claim that the custom loss function enables the TFT to do both:

a. Give more attention to weeks with high sales
b. Give more attention to Joint-IDs with higher sales

Both advantages over the baseline enable better modelling of seasonal data, especially in terms of performance on seasonal peaks. The plots and numbers confirm the intuition that went into developing our loss function. The math enables the model to fit better to high sales weeks over the baseline. While the performance is worse on low-sales week, such was expected as the custom loss function instructs the model to trade-off accuracy on low sales weeks for better accuracy on high sales weeks through weights.

X. Ensembling

As discussed, custom models witness a 4-5% drop in overall performance over baseline models as the data is skewed towards low SCAN-QTYs and the custom models perform worse on lower sales observations. To counter this and harness the strengths of both models, we can employ ensembling. The idea is to retain performance on low-sales weeks using SMAPE trained models and enhance seasonal peaks using custom-loss trained functions. While computationally more intensive and operationally more complex, ensembling does enable us to further performance gains.

For ensembling, there are two approaches:

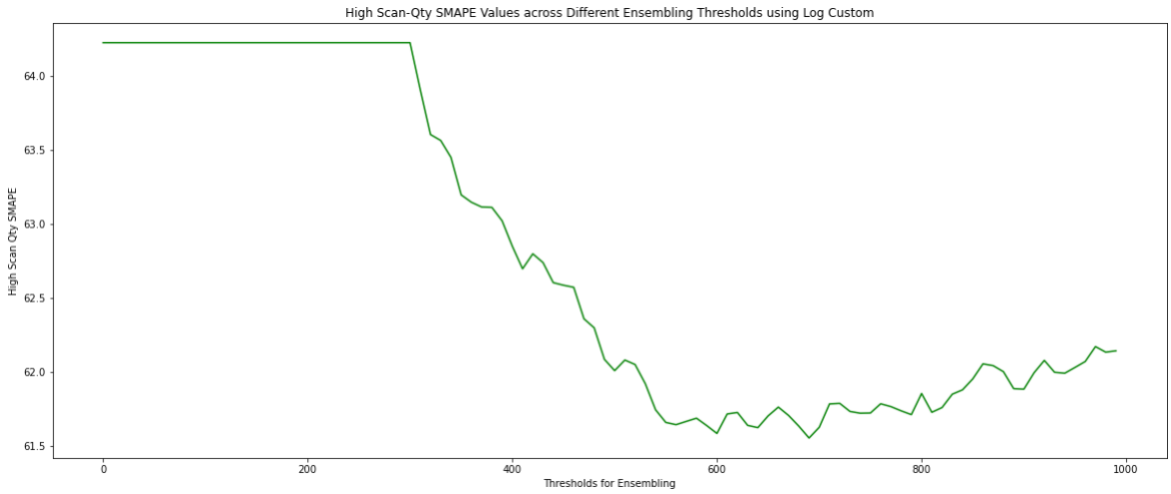
1. Take simple average of all model predictions.
2. Set a SCAN_QTY threshold before which to use the predictions of the SMAPE trained models and after which we use the custom models.

Two final ensembles were created for reporting, both are threshold-based models and use the Custom Log model for high sales weeks predictions:

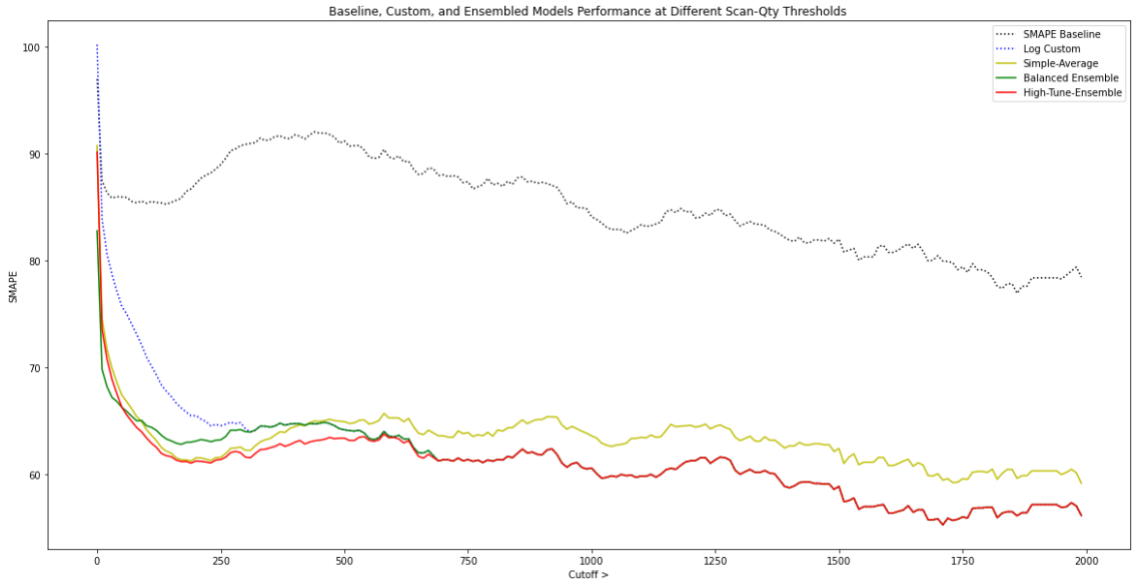
1. **Balanced Ensemble –** To balance the performance across low sales as well as high sales weeks. This uses the mean of the SMAPE and Log SMAPE model predictions for low sales weeks.

2. **High Sales Tune –** To further increase prediction accuracy on high sales weeks. This uses the mean of the Log SMAPE and Non-Log Custom for low to medium sales weeks.

The thresholds were set using the plot below. These allowed me to eyeball the optimal point where the strengths of each model work together to improve final predictions.



After the optimal threshold was set, both ensembles as well as a simple average model was plotted against custom and baseline models.



As can be observed, all ensembles offer improvements over baseline and custom functions, while all threshold-based ensembles outperform the simple average model. We can also observe that the ensembles overcome the standalone custom model's poor performance at SCAN_QTYs below 300 - that is where most of the SMAPE gains for the ensembled models come from. For difference among the high-tune and balanced ensembles, the balanced version seems to predict extremely low SCAN_QTYs better, while the high-tune predicts medium-low SCAN_QTYs better.

Model Trained on Loss Function	Overall SMAPE	High SCAN-QTY SMAPE	Overall % Improvement	High SCAN-QTY % Improvement
Custom Loss Baseline	**44%**	**65%**	**----**	**----**
Simple Average Ensemble	42%	62%	+2%	+3%
Balanced Ensemble	**34%**	64%	**+10%**	+1%
High Sales Tune Ensemble	41%	**61%**	+3%	**+4%**

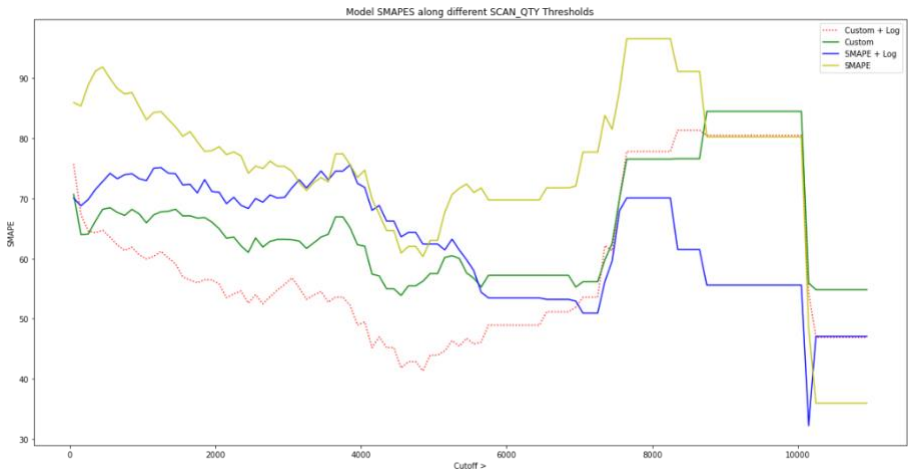
XI. Model Limitations

1. Performance on Low Sales Weeks

While the custom-loss function is easy to implement into the current pipelines, we will see a loss in predictive power on low-sales weeks. If low sales week do have tangible business impact, ensembled models can be used to achieve a balance, however, require more computational resources.

2. Unstable Performance on Extremely High Sales Weeks

All custom and ensemble models, following baseline models, see a sharp decline in performance at extreme outliers. As can be seen in the plot below, from the range of 7000 above all models perform poorly and the custom loss functions were not able to overcome this obstacle.



3. Hard to Integrate Ensembling

Ensembled models will be computationally and operationally more intensive to integrate into the current model pipelines and would lengthen deployment timelines.In addition, the effort and compute might not be worth the increase in accuracy we see in low-sales weeks as they hold low-business value.

XII. Final Recommendation

The final recommendation depends on use-case, business needs, and business impact. Below are some suggested models based on a corresponding use-case.

1. **Easy Integration and Accuracy on Seasonal Peaks:** The custom-loss model is very easy to integrate as it is written as a function that inherits from TSAI's Multi-Horizon Metric class and hence can be swapped in-place of SMAPE for current models. This model is inherently good at modelling incoming seasonal peaks, which holds business value.

2. **Inventory Cost and Management**: If inventory management is important and cost of holding inventory is high then low-sales weeks (0-10) must be accurate, for such cases, the balanced ensemble would be ideal. It offers accurate predictions on low-sales weeks ensuring additional inventory costs through dead-stock are not undertaken.

3. **Accuracy on Medium Sales Weeks:** If medium-low to medium sales week hold a lot of business value, then the high sales tune ensemble can be used. It offers better performance on both, medium sales weeks as well as high-sales weeks, for a bit of a decline in low sales performance.

XIII. Going Forward

While all goals of this project were met, additional improvements performance can be made using the ideas below.

1. **Hyperparameter Tuning:** All models reported in this project used the same hyperparameters that were tuned for the current SMAPE based models. Since we have a new loss function, tuning hyperparameters according to it might yield additional performance gains. Since the new loss function has changed the scale of the loss, the learning rate would be one major hyperparameter that could give significant gains. These tuning changes can be made using Optuna.

2. **2022 SCAN_QTYs as Weight:** Currently, the loss function only has access to the targets in the forecast horizon (26 weeks) as weights. This was done in order to keep the loss function external and not add any preprocessing steps. However, for better modelling, we could have the loss function take another argument for SCAN-QTYs across 2022, since that is the most representative year for sales. This would give the model better access to global seasonal

trends, enabling even better weighing of loss, helping us model seasonality even better.

3. **Extreme Outliers:** Since extremely high outliers hold high business value, it would be interesting to see if quadratically weighted loss functions could offer a solution for poor performance on those.

XIV. Project Summary

This project achieved all goals that we set out with. In terms of both - project goals and loss function development requirements. I was able to understand the logic behind the TFT and the team's implementation of it. I was able to improve performance on seasonal categories over baseline models with a swap-in custom loss function without requiring any significant changes in preprocessing steps or increase in compute needs. The loss function itself yielded the results that we were looking for – it overcomes edge-case instability of MAPE and SMAPE and enables the model to fit better to high sales weeks and products during training. There are also improvements that the team can make to further enhance performance in subsequent model updates.