Applying advanced techniques to the estimation of Multipurpose Digital Device Price Indices

Kevin Roche (kevin.roche@statcan.gc.ca), Lance Taylor (lance.taylor@statcan.gc.ca), and Roobina Keshishbanoosy (roobina.keshishbanoosy@statcan.gc.ca)

Prepared for the 17th Ottawa Group meeting in Rome, June 7th – 10th 2022

Abstract

Authors: ***Kevin Roche, Lance Taylor, Roobina Keshishbanoosy***

As part of the efforts to continuously improve the quality of the Consumer Price Index (CPI) at Statistics Canada, new methodologies and data sources are being researched for measuring price change in multipurpose digital devices. Web-scraped data is increasingly being used in CPI's because it has the benefit of increasing the monthly sample size and reducing manual price collection. The research demonstrated here integrates web-scraped data with supervised modelling approaches to produce different experimental quality adjusted price indices.

The results of using different algorithms in modelling prices for quality adjustment are compared, as well as different methods for constructing the corresponding quality adjusted price indices. Different specifications of ordinary least squares, random forest, and extreme gradient boosting (XGBoost) modelling algorithms are tested, with the XGBoost models delivering the best out-of-sample performance. Comparisons of the corresponding hedonic imputation and hedonic time dummy indices are made, including experimentation with the use of double machine learning (DML). DML is a recently developed technique that allows for causal inference when using non-parametric methods (i.e. XGBoost), which in this case is used in an attempt to create a time dummy index. Additionally, the effects of using different approaches to the indices are compared.

JEL Classification System: E31, C59, D12, L63
Keywords: Consumer Price Index, hedonics, web-scraped data, consumer electronics

1 - Introduction

The Consumer Price Index (CPI) measures the change in prices of consumer goods and services over time. In an effort to ensure that trends in consumer behaviour and the market are accurately captured, Statistics Canada continuously works to develop and test new methods of measuring price change in the various components of the CPI.

The Multipurpose Digital Devices Index (MDDI) represents 0.20% of the 2020 basket, and 2.1% of the Recreation, Education, and Reading Index, a major component of the CPI. The MDDI is made up of two elementary aggregates (EAs), the Smartphones EA and the Tablet Personal Computers EA – the latter of which is the aggregation of the price movements for smartwatches and tablets, where tablets make up roughly 80% of the EA. This paper summarises the research that's been undertaken on the price movements of these products, including:

- The incorporation of web-scraped data sources covering more brands and retailers;
- Construction of weights to ensure representativeness of brands and retailers;
- The development and comparison of different statistical models used to measure monthly price change by imputing missing prices;
- The comparison of different methods used in constructing the corresponding quality adjusted price indices.

This paper will largely focus on the latter two points, and proceeds as follows: the nature of the data is discussed in *Section 2*. The way the brand and retailer weights were constructed is discussed in *Section 3*. The variety of modelling techniques tested are discussed in *Section 4*. A comparison of the different methods used to construct price indices is discussed in *Section 5*. A discussion of the results of the research takes place in *Section 6*, with a brief conclusion in *Section 7*.

2 - Data

Web-scraping, the process of automatically extracting data from websites using bots, is a data source increasingly being used at Statistics Canada and other NSO's. Web-scraping is attractive because it greatly reduces the need for the manual collection of data and significantly increases the number of observations collected, often in the form of massive files with vast amounts of data in them for each retailer. Typically, we are only interested in a very small subset of that data – that which pertains to the "in-scope" products in the MDDI. In order to identify that subset, hard boundaries were set to determine whether a product was "in scope". These boundaries are defined as follows:

- **Smartphones:** New, unused, and unlocked.
- **Tablets:** A portable personal computer where the primary input is touch. May or may not have a physical keyboard, mouse, touchpad or stylus, and only adult models and size were considered.
- **Smartwatches:** Functionality that exceeds that of a basic fitness tracker, where connectivity features, activity monitoring, and phone control features are a must. Only adult models and size were considered.

In addition to the boundaries implemented to ensure a given product was "in scope", a few other boundaries were implemented to ensure the data was of acceptable quality:

- Upper and lower price bounds were established for major product type-brand combinations by subject matter experts. This serves to prune outliers and remove erroneous pricing in lieu of manual intervention.
- For retailers that have a marketplace, observations from marketplace sellers were filtered out.
- In the case that there were multiple identical products within a retailer in a period, the mean of their prices was taken. This is done to avoid potential composition effects that may arise from over-representing observations from a given retailer or brand.

After the identification of in-scope items took place, the data cleaning process involved using regular expressions and a rules-based approach, resulting in monthly, tabular, cleaned data.

3 – Weight Construction

This section details the methodology used to construct the retailer and brand expenditure weights, which are then used in the data modelling process detailed in *Section 4* and the index aggregation process detailed in *Section 5*.

3.1 – Retailer Weight Construction

Where possible, retailer weights were constructed from Statistics Canada's official Retail Commodity Survey (RCS). For some retailers (say, *Retailer 1* and *Retailer 2*), it was not possible to construct retailer weights from the RCS data. This was due to the business practices of these retailers not completely fitting into the structure of the survey (ie. weak brick-and-mortar presence). In this scenario, creative approaches were required to construct the weights.

According to an external source, *Retailer 1* has some multiple of the market share for consumer electronics that another retailer in the sample (say, *Retailer 3*) had. So, *Retailer 1*'s aggregate sales were simply calculated as that multiple of *Retailer* 3's aggregate sales.

It was a more involved process to construct weights for *Retailer 2*. Using a source, it was possible to determine *Retailer 2*'s global consumer revenue for the hardware sub-segment of its business (SS Revenue).

To determine *Retailer 2*'s aggregate sales for each product, the typical share of tablets, smartwatches and phones in the North American Industry Classification System (NAICS) data were calculated (Relevant Share). These proportions were then applied to *Retailer 2*'s revenue (SS Revenue), giving an approximation of *Retailer 2*'s aggregate sales from the three products. That result was then multiplied by the amount of *Retailer 2*'s revenue that comes from Canada (Canadian Share).

Lastly, *Retailer* 2's aggregate sales were multiplied by the revenue *proportion* of *Retailer 2*'s hardware sub-segment revenue (SS Proportion), so as not to include sales from other aspects of their business (ie. B2B sales).

Formally, *Retailer 2*'s aggregate sales in year *y* are computed as:

$$Aggregate\ Sales_y = SS\ Revenue_y * Relevant\ Share_y * Canadian\ Share_y * SS\ Proportion_y$$

3.2 – Brand Weight Construction

The construction of brand weights required data on both the annual quantities and the annual prices for each product brand. An external data source was used to determine annual quantities of the number of units sold by each brand for each product, and average annual prices were constructed from the prices in the web-scraped data. To start, the average monthly price by brand and product was calculated:

$$Average\ Monthly\ Price_t^{p,b} = \frac{\sum_{i=1}^{n} Price_i^{p,b}}{n}$$

From there, the average yearly price by brand and product was calculated as the mean price of all of the months within a given year:

$$Average\ Annual\ Price_y^{p,b} = \frac{\sum_{t=1}^{12} Average\ Monthly\ Price_t^{p,b}}{n}$$

Some months may have had varying level of data procurement, so having all observations in a year averaged without first taking the monthly average could result in some months being implicitly weighted more than others. Taking the monthly average for each product-brand combination and then averaging those across the year served to combat this effect.

Lastly, brand weights were calculated as follows:

$$Brand\ Weights_y^{p,b} = Units\ Sold\ (External\ Data)_y^{p,b} * Average\ Annual\ Price_y^{p.b}$$

4 - Data Modelling

This section details the variety of modelling techniques used in building the indices. Most of the models attempt to model monthly price change by imputing missing prices, with an attempt to model the period of the observation within a DML framework being made in Section 4.4.2.1. OLS, Random Forest, and XGBoost models were all considered.

4.1 – Data Preparation

The dataset contains information on a vast number of characteristics for each product from the beginning of 2019 onwards, but the degree of how well represented those qualities are for each product varies. For example, nearly every collected observation has a value for *operating system*, but less than 50% of them have a reported a value for *ram size*. In turn, this would mean that if *ram size* was used as a characteristic in the index calculation process, we would lose over half of the sample size.

In order to include as many observations in the modelling process as possible, subject matter experts were consulted to determine which characteristics were most important in determining the price of each product. Of these characteristics, the ones used were the characteristics that had at least 50% representation in the dataset[1]. The end result was that the covariates in the table below were used to model prices, where an asterisk denotes that a covariate is continuous. Note that observations that contain information on all the covariates within a product type are referred to as “complete” or “fully represented” throughout the paper.

[1] Low representation of a characteristic suggests that it may not be as relevant as the product’s other characteristics in its relationship to price.

Product Type	Covariates
Phone	Brand, Display Size*, Horizontal Screen Resolution*, Operating System, Camera Resolution*, Storage*, Vertical Screen Resolution*
Watch	Accelerometer, Brand, Case Size*, Calorie Tracking, Connectivity, GPS, Heart Rate, Incoming Calls, Microphone, Operating System, Sleep Tracking, Steps Tracking, SMS, Touchscreen, Voice Control, Waterproof
Tablet	Brand, Connectivity, Display Size*, Horizontal Screen Resolution*, Operating System, Rear Camera Resolution*, Storage*, Vertical Screen Resolution*

On a monthly basis, the number of complete (ie. fully represented) observations can fluctuate[2]. As more retailers have been added to the sample, the number of complete observations have grown. Since the most recent retailer was added, there has been an average of around 125 complete phone observations, 500 complete tablet observations, and 4000+ complete watch observations.

Once the relevant covariates were determined, the natural logarithm of the price and all the continuous (ie. non-dummy) variables were taken. This transformation has a variety of desirable properties, including:

- In the case that the residuals are not normally distributed, taking the natural logarithm may improve the fit by making the variables more closely approximate a normal distribution.
- In the case of an OLS regression, taking the log allows a 1% increase in a given covariate to be interpreted as a β% increase or decrease in price.

Once this covariate selection and transformation process was complete, model testing began. In this process, the relevant monthly data was randomly partitioned into a 70% training, 30% testing split. Randomly partitioning within each month ensures that there are a proportionate number of observations in each period of each window, which in turn helps ensure representativeness in both the training and evaluation of the algorithms.

4.2 – Hedonic Imputation Methodology

In machine learning, a hyper-parameter is a value external to the model that is used to control the learning process. In order to optimize the way that prices are hedonically imputed for each of the three product types, a variety of different model and hyper-parameter combinations were tested. The strategy here was to first grid search across a broad set of window lengths and hyper-parameters and then run a second, narrower grid search on the model that performed the best.

4.2.1 – High-level Grid Search

For the broad, initial grid search, three models were tested, each with a variety of hyper-parameter combinations. The OLS model used in hedonic modelling was set up as follows, where $\ln(X)$ refers to the matrix of continuous covariates that have been log-transformed and D refers to the matrix of dummy covariates:

[2] For example, data can be missing, new retailers can be added to the sample, and the number of observations can vary in any given month. These fluctuations are due to the data collection process (ie. they are out of our control). This is precisely why retailer weights are used to weight for representativeness.

$$\ln(Price)_i^p = \alpha + \beta \ln(X_i) + \theta D_i + \varepsilon_i$$

Random Forest and XGBoost models were also constructed using the same sets of covariates. Due to the non-parametric nature of these models, they cannot be generalized into an exact closed-form equation as seen above. Instead, they are generalized as follows:

$$\ln(Price)_i^p = f(\ln(X_i), D_i) + \varepsilon_i$$

Random Forest is an ensemble technique that uses bagging and feature randomness to combine the output of multiple decision trees into a single result. It reduces the risk of overfitting the data, but is time-consuming to calculate and computationally expensive.

XGBoost[3], which stands for extreme gradient boosting, is an optimized implementation of gradient boosting trees. Boosting is an ensemble technique where new statistical models are added iteratively to correct the errors of the existing statistical models. Its desirable properties[4] include:

- ***Auto Pruning:*** ensures trees do not grow beyond particular limit, making the statistical model more robust and decreasing computation time
- ***Regularization:*** prevents overfitting by shrinking estimates
- ***Iteration:*** each iteration of the statistical model uses the error residuals of the previous statistical model to fit the next model, which minimizes loss

At this point, it would be useful to define the hyper-parameters[5] the model grid searches over. Note that if a hyper-parameter is not defined below, the default value for that hyper-parameter is used in the model.

- ***Max Depth*** is the maximum depth of a tree. A lower *Max Depth* is less likely to over-fit, but does not model complex relationships as well.
- ***Number of Rounds*** defines the maximum number of boosting iterations. If model performance does not improve for 25 rounds, the model will stop iterating before it reaches the maximum number of iterations in order to combat overfitting. The lower this number, the less computationally expensive – but it is also important to ensure that as much information as possible is extracted.
- ***ETA*** controls the learning rate, scaling the contribution of each tree by a factor of $0 < ETA < 1$ when it is added to the current approximation. A lower *ETA* makes the model more robust to overfitting, but is more computationally expensive. Note that ETA is only tuned in the intensive grid search.
- ***Node Size*** is the minimum size of terminal nodes. Allowing for a larger node size leads to less complex trees, which makes the model less computationally expensive.
- ***mtry*** is the number of predictors that are randomly sampled at each split when creating the tree models. It helps to balance low tree correlation with reasonable predictive strength.

[3] A more nuanced description of how XGBoost works is available here.
[4] Other benefits include computational efficiencies such as *Parallelization* (uses all CPU cores in training), and *Cache Optimization* (stores intermediate calculations in cache), both of which enhance the speed of the algorithm.
[5] A more in-depth discussion of hyper-parameters is available here.

Some may ask why different window lengths (ie. the number of time periods used in a given model) are experimented with. By extending window length, and thus gaining more observations, the model is potentially better able to model the complex relationship between the observed characteristics and prices, which should aid in predicting price. If a model is effective at predicting the price on a window, the hope is to also establish that the model is effective at predicting the price on the period at the *end* of the window.

Notably, using window lengths involves some trade-off – some observations in the past may be less representative in the current period, but it is assumed that the level of bias this trade-off presents is acceptable given the improved prediction performance.

There is also a practical reason to experiment with window lengths – when working with messy web-scraped data, it is often the case that an observation exists in a previous period, exists in some future period, and does not appear in the current period. In this case, including the past observation due to a longer window length could aid modelling.

Models were constructed for each month in our sample, and a grid search was run over the following hyper-parameters for each model:

Model	Window Length	Use of weight	Max Depth	Node Size	mtry
OLS	1, 3, 6, 8, 12	Weighted, Unweighted	N/A	N/A	N/A
Random Forest	1, 3, 6, 8, 12	Weighted, Unweighted	N/A	3, 6	4, 6
XGBoost	1, 3, 6, 8, 12	Unweighted	6, 8	N/A	N/A

For each model resulting from each possible grid search combination, the *Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE),* and R^2 were calculated, as well as their corresponding weighted equivalents. This is done to ensure that the models perform well on representative data. These metrics were chosen mainly for their interpretability. Note that in both the high-level grid search and the intensive grid search detailed in *Section 4.2.2*, metrics were calculated based on the models out-of-sample performance on the test set. Applying cross-validation techniques is likely a superior way to obtain these metrics, but given the extent of the grid search, it was simply too computationally expensive to use it at this stage. Once optimal models were determined, cross-validation was implemented to ensure that the fit using cross-validation resembled the out-of-sample fit of the model on the test set. The aforementioned metrics are defined below:

- The ***Mean Absolute Error*** takes the average absolute magnitude of the errors in a set of predictions. In words, an MAE of $50 means that the average prediction was within $50 of the actual price. MAE and Weighted MAE (WMAE) are calculated as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

$$WMAE = \frac{1}{\sum w_i}\sum_{i=1}^{n} w_i|y_i - \hat{y}_i|$$

- The ***Mean Absolute Percentage Error*** is the percent equivalent of the MAE. In words, a MAPE of 10% means that the average prediction was within 10% of the actual price. MAPE and Weighted MAPE (WMAPE) are calculated as follows:

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{y_i}$$

$$WMAPE = \frac{100\%}{\sum w_i}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i| * w_i}{y_i}$$

- $\boldsymbol{R^2}$ is a widely used measure for goodness of fit. Values range from zero to one, with values closer to one indicating a better fit. R^2 and Weighted R^2 (WR^2) are calculated as follows:

$$R^2 = 1 - \frac{\sum_{=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{=1}^{n}(y_i - \overline{y_i})^2}$$

$$WR^2 = 1 - \frac{\sum_{=1}^{n} w_i(y_i - \hat{y}_i)^2}{\sum_{=1}^{n} w_i(y_i - \overline{y_i})^2}$$

To be explicit, using a weighted or unweighted metric doesn't change the *prediction* of a model, just how it is *evaluated*. Weighting the metrics shows how much better or worse the model performs on observations with higher weight. Also of note is the way these metrics are presented – because both the dependent variable and the continuous covariates were log-transformed, *e* was raised to the power of the inputs in the *MAE* and *MAPE* calculations to more closely approximate a dollar value when the metrics are reported. This methodology doesn't change the manner in which the models are evaluated, but allows less technical audiences to better understand the implications that these metrics have.

Note that XGBoost does not use weights in the same manner as other algorithms[6]. In the context of XGBoost, weighting means increasing the contribution of an observation to the loss function. So, instead of multiplying the target values by the weights, it instead multiplies the gradient and the hessian. This does not "weight" the values in the classical sense of the word (ie. for representativeness), so unweighted models are used throughout.

To evaluate the models, the aforementioned metrics for each product type and model grouping were averaged over all of the months in the sample to give us a very high level idea of which models were performing best. There weren't substantial differences between the weighted and unweighted OLS and random forest models, so only the unweighted models are reported for the simplicity of the table below. This does not have an effect on the conclusions drawn. The results were as follows:

[6] XGBoost Documentation — xgboost 1.6.0 documentation

Product Type	Model	Average MAE	Average WMAE	Average MAPE	Average WMAPE	Average R^2	Average WR^2
Phone	OLS	$167.65	$164.51	21.70%	24.44%	0.83	0.82
	Random Forest	$100.07	$96.01	13.61%	14.30%	0.92	0.92
	XGBoost	***$62.96***	***$62.05***	***9.18%***	***9.89%***	***0.94***	***0.94***
Watch	OLS	$110.29	$91.81	21.09%	19.46%	0.60	0.57
	Random Forest	$93.21	$70.96	17.16%	15.35%	0.70	0.72
	XGBoost	***$58.05***	***$43.15***	***10.77%***	***9.32%***	***0.79***	***0.80***
Tablet	OLS	$116.96	$134.43	12.83%	15.21%	0.94	0.93
	Random Forest	$103.65	$106.69	10.01%	10.73%	0.96	0.96
	XGBoost	***$37.28***	***$48.10***	***4.46%***	***5.54%***	***0.98***	***0.98***

And below the standard deviation of each of the monthly metrics is reported:

Product Type	Model	MAE SD	WMAE SD	MAPE SD	WMAPE SD	R^2 SD	WR^2 SD
Phone	OLS	$47.90	$47.48	8.25%	11.49%	0.110	0.108
	Random Forest	$32.48	$35.07	7.10%	7.65%	0.077	0.084
	XGBoost	$23.89	$26.59	6.54%	7.08%	0.082	0.088
Watch	OLS	$25.99	$24.97	4.33%	4.86%	0.143	0.248
	Random Forest	$21.10	$18.77	2.84%	3.56%	0.134	0.143
	XGBoost	$24.21	$24.76	3.95%	4.69%	0.131	0.223
Tablet	OLS	$19.74	$21.87	1.85%	1.47%	0.021	0.021
	Random Forest	$28.11	$31.06	2.82%	2.96%	0.024	0.026
	XGBoost	$19.89	$22.96	2.35%	2.36%	0.020	0.021

For all three product types, the XGBoost models outperformed the Random Forest and OLS models, and the models performed equally well using both weighted and unweighted metrics. Looking into the lower level groupings showed that medium-sized window lengths (ie. those in the 3-8 range) performed better than their extreme counterparts (ie. window lengths of 1 or 12). Furthermore, a *Max Depth* of 6 performed better on average than 8 did. With this information, XGBoost was selected as the model of choice, and a more intensive grid search was implemented.

4.2.2 – Intensive Grid Search

Now that XGBoost has been selected to model the data, a more intensive grid search was run over its hyper-parameters using what we had learned from the high-level grid search to optimize the model.

The grid search was run over the following combinations of hyper-parameters to create monthly models for each combination and product type:

Model	Window Length	Max Depth	No. Rounds	Eta
XGBoost	3, 4, 5, 6, 7, 8	2, 4, 5, 6, 7	300, 600	0.15, 0.2, 0.3

To evaluate the models, the aforementioned metrics for each product type and model grouping were again averaged over all of the months in the sample in order to determine which models were performing best.

Phone and watch models performed better with shorter window lengths[7], and a window length of four months was selected in both cases. The tablet models, on the other hand, performed better with longer window lengths – although the difference between short and long windows was somewhat negligible (difference in *MAE* of $0.25). Shorter window lengths are more desirable from a theoretical perspective because constant relationship between characteristics and price is assumed for a product within a window, and this assumption holds more easily under a shorter window. Thus tablets were modeled with a window length of five months, at the (nearly negligible) sacrifice of optimal performance.

Hyper-parameters were selected based on which combination was most optimal for each product in the window length of choice. In all cases, the metrics were the same for the optimal models whether the *number of rounds* completed was 300 or 600, indicating that 300 iterations was sufficient. The optimal XGBoost model[8] for each product type was as follows:

Product Type	Window Length	Max Depth	No. Rounds	Eta
Phone	4	6	300	0.15
Watch	4	6	300	0.3
Tablet	5	6	300	0.3

The corresponding average weighted and unweighted out of sample (test set) metrics for the optimal models were as follows:

Product Type	Average MAE	Average WMAE	Average MAPE	Average WMAPE	Average R^2	Average WR^2
Phone	$54.89	$54.91	6.21%	6.46%	0.955	0.957
Watch	$56.37	$41.95	10.3%	9.42%	0.795	0.838
Tablet	$34.46	$43.62	3.88%	4.91%	0.985	0.98

[7] This could be due to the higher relevance of annual flagship model releases for phones and watches compared to tablets.

[8] The code for this section was run multiple times, with each iteration producing ever-so-slightly different results. Listed here are the hyper-parameters that *tended* to perform the best, but that does not guarantee that they would be the best performing hyper-parameters on any given iteration. In all iterations, the difference between the hyper-parameters listed and the next best set of hyper-parameters was negligible.

And below the standard deviation of the monthly metrics are reported:

Product Type	MAE SD	WMAE SD	MAPE SD	WMAPE SD	R^2 SD	WR^2 SD
Phone	$14.30	$14.84	1.34%	1.65%	0.065	0.063
Watch	$18.78	$17.50	3.28%	5.28%	0.230	0.180
Tablet	$13.47	$11.78	1.48%	1.33%	0.010	0.008

The main takeaway here is that the models still perform similarly with both unweighted and weighted metrics, which indicates the algorithm is able to handle relevant observations as effectively as less relevant observations.

It is also valuable to provide some intuition as to why each model performs as it does. The tablet and phone models both perform extremely well, while the watch model's performance is relatively lacklustre. There are two potential reasons why this is so:

- The technical characteristics of tablets and phones are better represented than they are for watches, so the actual functionality of the product is better described in the data.
- The available characteristics of tablets and phones have greater relevance, so the provided characteristics are the set of characteristics that are relevant to the consumer (more so for tablets). In other words, there is less of a difficult-to-define form and fashion component compared to watches.

Also of interest is how these models perform on observations in the current period (ie. most recent period) of their window only, given their intended use in hedonic imputation. Below the metrics obtained from using the models to make predictions only on observations *from the current period* in the test set are reported:

Product Type	Average MAE	Average WMAE	Average MAPE	Average WMAPE	Average R^2	Average WR^2
Phone	$58.40	$53.91	6.20%	6.51%	0.960	0.950
Watch	$67.76	$44.60	11.80%	11.69%	0.634	0.671
Tablet	$41.51	$52.73	4.66%	5.82%	0.983	0.980

And below the standard deviation of the monthly metrics are reported:

Product Type	MAE SD	WMAE SD	MAPE SD	WMAPE SD	R^2 SD	WR^2 SD
Phone	$20.59	$24.57	2.19%	3.22%	0.047	0.089
Watch	$21.77	$21.43	3.56%	7.44%	0.359	0.354
Tablet	$16.14	$28.64	1.90%	2.28%	0.012	0.018

Some noise is to be expected due to the significantly smaller sample size for which the metrics are being calculated, but overall these results indicate that the phone and tablet models are roughly as good at predicting observations in the current period as they are across the sample as a whole.

Lastly, the cross-validation results are compared to the out-of-sample results on the test set. Due to the nature of the program used in this comparison, both *MAE* and *MAPE* results are reported below without raising *e* to the power of both the prediction and the test set. However, the important result here is that the cross-validation model's performance is similar to the out-of-sample test sets performance.

Product Type	Average OOS CV MAE	Average OOS Test MAE	Average OOS CV MAPE	Average OOS Test MAPE
Phone	0.0721	0.0675	1.13%	1.05%
Watch	0.122	0.118	1.97%	1.91%
Tablet	0.0460	0.0448	0.716%	0.695%

For each product type, monthly statistical models were calculated and stored using the optimal hyper-parameters. During the aggregation process detailed in *Section 5.1*, the corresponding monthly models were used to impute missing prices where appropriate.

4.3 – Hedonic Time Dummy

The hedonic time dummy modelling process followed the same data preparation process outlined in *Section 3.1*. Paramaterization is a necessity when employing a time dummy approach to price measurement because the measure of price change relies on an interpretable coefficient, which means that historically, only Generalized Linear Models (GLMs) such as OLS and Weighted Least Square (WLS) have been appropriate. In a time dummy hedonic model, the time dummy parameter $\hat{\delta}$ measures the effect of "time" on the log of price after controlling for the relevant quality-determining characteristics.

4.3.1 – GLM Testing

Two approaches to modelling hedonic time dummies were tested, an (unweighted) OLS model and a WLS model. The OLS model for a given period and product is set up as follows, where $\ln(X)$ refers to the aforementioned continuous covariates that have been log-transformed, D refers to the dummy covariates, and *Z* refers to the matrix of time dummy variables:

$$\ln(Price)_i = \alpha + \beta \ln(X_i) + \theta D_i + \sum_{t=1} \delta_t Z_i(t) + \varepsilon_i$$

The Weighted Least Squares (WLS) approach follows a similar methodology, except that the observations are weighted to be representative based on their corresponding brand and retailer weights – as detailed in *Section 3*.

In all cases, the WLS model outperformed the OLS model from a goodness of fit (R^2) perspective, but the performance of both models was still far inferior to the XGBoost models. As a result, the output from the WLS models was used when aggregating the indices in this comparative study.

4.4 – Double Machine Learning Time Dummy

The previous section notes that, historically, only GLMs have been used to calculate time dummies thanks to the interpretability of their time dummy coefficients. This is largely because parameterized machine learning techniques yield biased parameters – in other words, the effect of individual covariates cannot be interpreted.

In 2018, Chernozhukov et al pioneered a new framework referred to as Double Machine Learning (DML) that has the potential to address this limitation. DML allows us to control for covariates using machine learning techniques while still producing an interpretable coefficient on a variable of interest.

4.4.1 – A Brief Overview of DML

Say there is a dataset with a response variable, price, a treatment variable, time, and a host of other control variables whose interpretability is irrelevant. Double ML works as follows:

1. The data is split into two sets. (Typically 50/50, so not akin to a training/test split)
2. The response variable, price, is modeled as a function of the control variables in both datasets. The treatment variable, time, is not included in this model.
3. The treatment variable, time, is modeled as a function of the control variables in both datasets.
4. The response residuals are determined.
5. The treatment residuals are determined.
6. The response residuals are regressed (OLS) on the treatment residuals for both datasets, and the treatment effect is then defined as the average of the coefficients on the treatment residuals from each of the respective regressions. In each of the periods in the sample, the treatment effect for the corresponding period serves as the time dummy.

The framework is called "double" machine learning because two machine learning models are built, one for the response variable and one for the treatment variable. There is no specific machine learning model that needs to be used to model the response and treatment variables – for example, random forest, XGBoost and support vector machines models could all be appropriate. Given that the data has to be split into two, having a large number of observations is a luxury.

DML allows an interpretable coefficient to be produced because it addresses two sources of bias: regularization bias and overfitting bias. For a more nuanced discussion on these biases, see the footnote here[9].

4.4.2 – DML Model Construction

Following the outline laid out above, the data for each optimal product type – window length combination detailed in *Section 4.2* was randomly partitioned into two separate sets, each making up 50% of the overall observations. From there, the response variable, price, was modeled as a function of

---

[9] The goal of DML is to get a root-N consistent estimator and the confidence intervals of a parameter of interest (in this case, time). There are two sources of bias that DML addresses, which is what allows for an interpretable coefficient to be produced.

1. ***Regularization bias*** is addressed by imposing the Neyman Orthogonality Condition on the score function – which yields an unbiased, root-n consistent estimator.
2. ***Overfitting bias*** is addressed by sample-splitting and cross-fitting. By splitting the data into two, obtaining a time dummy for both sets, and taking the average of the two results, the risk of overfitting is reduced.

For a more in-depth discussion on these biases and how they're addressed, see this presentation from Chernozhukov et al. where they discuss their aforementioned original paper.

the control variables in both datasets. Time was not included as a control. The results of this modelling were similar to those in *Section 4.2.2*.

Next, the treatment variable, time, was modeled as a function of the control variables in both datasets. A number of different methodologies were employed in an effort to give a reliable estimate – none of which led to reliable results.

4.4.2.1 – DML Time Model Experimentation

In an effort to successfully predict which window in the sample an observation came from (ie. to satisfy step 3 of the DML procedure), the classification problem was structured in a variety of different ways:

1. ***Binary structure:*** In this structure, the data was read in two periods at a time, with the current time period being coded as 1 and the previous time period being coded as 0. This is the most theoretically sound way to approach the problem, and the one used to create the experimental indices.
2. ***W-class structure:*** In this structure, the time period was coded from 0…W, where 0 was the oldest window in the sample and W was the most recent window.
3. ***Three-class structure:*** In this structure, time was coded as either 0, 1, or 2, where 2 was a dummy for the most recent period, 1 was a dummy for the second-most recent period, and 0 was a dummy for all other periods in the window.

In classification problems, there are four possible outcomes[10] for each predicted class, and metric scores are calculated for each individual class. The four possible outcomes of a prediction in a given class (in this example, class one in a W-class structure) are as follows:

- ***True Positive:*** the model correctly predicts class one as class one.
- ***True Negative:*** the model correctly predicts class two, three, or four as class two, three, or four.
- ***False Positive:*** the model incorrectly predicts class two, three, or four as class one.
- ***False Negative:*** the model incorrectly predicts class one as class two, three, or four.

The process is then repeated for classes two, three, and four. The same logic applies to the simplified cases of the binary class structure and the three-class structure.

The above metric scores were calculated for each class, and the models were then evaluated on two commonly used metrics, *precision* and *recall,* which are detailed below:

- ***Precision*** is a measure of what proportion of predicted positives are truly positive. Optimizing for precision ensures that the number of *false positives* will be minimized. Precision is calculated as:

$$\frac{True\ Positives}{True\ Positives\ +\ False\ Positives}$$

- ***Recall*** is a measure of what proportion of actual positives are correctly classified. Optimizing for recall ensures that the number *of false negatives* will be minimized. Recall is calculated as:

[10] A short write-up further explaining confusion matrix, precision, and recall is available here.

$$\frac{True\ Positives}{True\ Positives\ +\ False\ Negatives}$$

The score of each metric averaged across all classes in the *binary* structure is reported below. The binary structure is the most theoretically sound way to approach the problem because when there are only two periods, the coefficient on the treatment effect can be interpreted as the price movement between the two periods – which is the classical interpretation of the adjacent period time dummy approach.

Product Type	**Average Precision**	**Average Recall**
Phone	0.389	0.388
Watch	0.441	0.437
Tablet	0.384	0.389

The average *precision* value for phones in these results can be interpreted to mean that when the model predicts which period the observation is from, the predicted class is the correct class only 38.9% of the time. The average *recall* value for phones can be interpreted to mean that only 38.8% of the true positives in a given class are correctly classified. Given there are only two classes in this structure, these results effectively mean that you'd be better off predicting which period an observation is from via random draw.

There was also an attempt to structure the problem in the aforementioned *W-class* and *three-class* structures. These structures produce less interpretable treatment effects due to the way the raw residuals are calculated, but they were tested in an effort to exhaust all avenues of modelling.

Below, the score of each metric averaged across all classes in the *W-class* structure are reported, as well as the score of each metric in period *W,* the most recent period in the window.

Product Type	**Average Precision**	**Average Recall**	**Average Period *W* Precision**	**Average Period *W* Recall**
Phone	0.181	0.184	0.222	0.219
Watch	0.236	0.235	0.323	0.314
Tablet	0.125	0.131	0.167	0.171

Context is important when interpreting the results of models with different numbers of classes. At a quick glance, it might appear that the *binary* structure outperformed the *W-class* structure. However, the results need to be interpreted keeping in mind the number of classes. A precision score of 25% in a four-class model is akin to a precision score of 50% in a binary model, because in both cases, the predicted class is the correct class *1/W* of the time. In other words, the results of the *W-class* structure are effectively the same as the *binary* structure – the model cannot predict which period in the window the observation is from with any accuracy at all.

Lastly, the scores of each metric averaged across all classes in the *three-class* structure are reported below. Given that the only movement of interest is the movement between the previous period and the current period, there was some theoretical backing to break the problem into only three classes.

Product Type	Average Precision	Average Recall	Average Period *W* Precision	Average Period *W* Recall
Phone	0.304	0.319	0.265	0.175
Watch	0.361	0.367	0.362	0.238
Tablet	0.309	0.329	0.221	0.099

The conclusion here is the same – based on the above approach, the time period that an observation falls in cannot be predicted with any certainty.

4.4.3 – Discussion Regarding Results

For all three of phones, watches, and tablets, a given item has constant characteristics for its time on market. Furthermore, churn rates are not high enough and the characteristics of entering products are not significantly different enough to affect the composition of product characteristics in the sample in a relevant manner. As a result, it is extremely challenging to accurately predict which period the product was sold in in the MDDI based on its characteristics.

Given the weakness of the time component of the DML models and the excellent performance of the price models[11], there would be no justification to implement the DML approach in the MDDI. Nonetheless, this research is an important contribution to the research and development of CPI's for two main reasons:

- Elementary aggregates that experience large amounts of product churn or seasonality, such as the price indices for clothing, are potentially well suited for a DML framework.
- More generally, any price indices that are calculated on longer intervals – such as quarterly[12], semi-annually, or annually – and experience notable product churn over that interval have the potential to benefit from a DML framework.

While the DML approach could not be used in the estimation of the MDDI, the indices were still calculated using the results of the *binary-class structure* in an effort to:

- See how they compare to the results from the hedonic imputation and hedonic time dummy approaches.
- Lay the ground work for future research on the implementation of DML for price measurement.

The result of these index calculations was that the price movements of the DML indices were seemingly random, likely due to the inability of the model to accurately predict time. The indices were not very comparable to the indices constructed using other methodologies, and as a result they are not shown in Section 6.[13]

[11] This result is actually implied by the fact that the phone and watch models performed just as well in the most-recent period of the window. If there were any discernible characteristics across periods, it likely would've led to differences in performance across periods.
[12] For example, Australia reports their CPI on a quarterly basis.
[13] If you'd like to see the DML indices or chat further about DML, please contact the authors.

5 – Aggregation Structures

This section details the aggregation structures tested in determining the best way to construct the indices.

5.1 – Hedonic Imputation Aggregation Structures

The following process describes the general framework used to aggregate the hedonic imputation indices. First, within each brand *b* of each product *p*, price relatives for each observation *i* in a time period *t* were calculated by dividing *P1* by *P0*:

$$price\ relative_{i,t}^{p,b} = \frac{P1_{i,t}}{P0_{i,t}}$$

Next, within each product, the retailer weights were divvied up by the number of observations within each retailer. For example, if *Retailer 1* had a retailer weight of $10, and there were ten *phone* observations from that retailer, each observation would be assigned a retailer weight of $1. This is done so that the retailer weights in each period are constant for each retailer. This is desirable because it avoids the potential pitfall of implicitly weighting one retailer more than another in scenarios where the amounts of observations collected by retailer vary across periods, which is common with this type of data.

From here, the price relatives are aggregated up to the product-brand level by calculating the *geometric mean of price relatives* for each group. Each observation in this calculation is weighted by its corresponding divvied up retailer weight value:

$$price\ relative_{t}^{p,b} = \prod_{i=1}^{N} price\ relative_{i,t}^{p,b}{}^{\,w_{i,t,r}^{p} / \sum_{i=1}^{N} w_{i,t,r}^{p}}$$

In the case that a brand appears in the current period but did not appear in the previous period, it is imputed using *overall mean imputation*.

In each period, weights are *price-updated* by multiplying the previous period's weight by the current period's price movement. In other words, price-updating involves the initial set of expenditures being multiplied by the estimated pure price change:

$$w_{t}^{p,b} = w_{t-1}^{p,b} \times price\ relative_{t}^{p,b}$$

These *price-updated weights* are then used to aggregate up to the product level:

$$price\ relative_{t}^{p} = \frac{\sum w_{t}^{p,b}}{\sum w_{t-1}^{p,b}}$$

This produces a single number – the relative (weighted) price change for product *p* from *P0* to *P1.*

Note that the arithmetic mean is taken instead of the geomean at this stage. Taking the geomean tends to yield a smaller price movement, and because the price movements at the brand level are mostly

independent (ie. people do not substitute between brands[14]), a weighted arithmetic mean is used *when aggregating to the product level* to better capture the effects of each of the brands individual price movements.

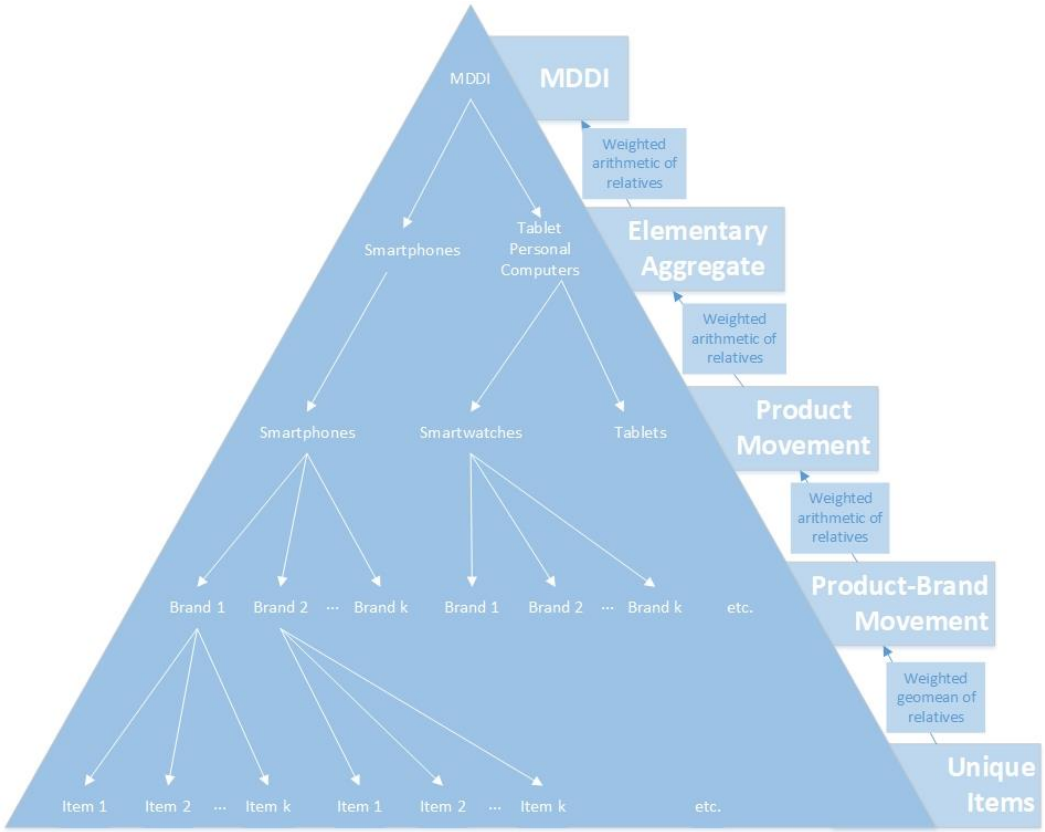
Some of the products in the sample fall under the same EP. When this is the case, one further aggregation is required. To do this, the weights are once again price-updated – this time at the EA level.

$$w_t^p = w_{t-1}^p \times price\ relative_t^p$$

There is not much substitution at the product level (ie. consumers do not substitute a tablet for a smartwatch), so a weighted arithmetic mean is again used to aggregate up.

$$price\ relative_t^{EA} = \frac{\sum w_t^p}{\sum w_{t-1}^p}$$

The general framework used in the MDDI's hedonic index construction described above is summarized in the following diagram:



In the next section, a number of variants are tested in an effort to determine the robustness of the methodology.

[14] There is some argument to be made that people are more likely to substitute between brands *within* an operating system. This could potentially be an interesting aggregation structure to look at in the future, but it is not pursued in this paper.

5.1.1 – Robustness Testing

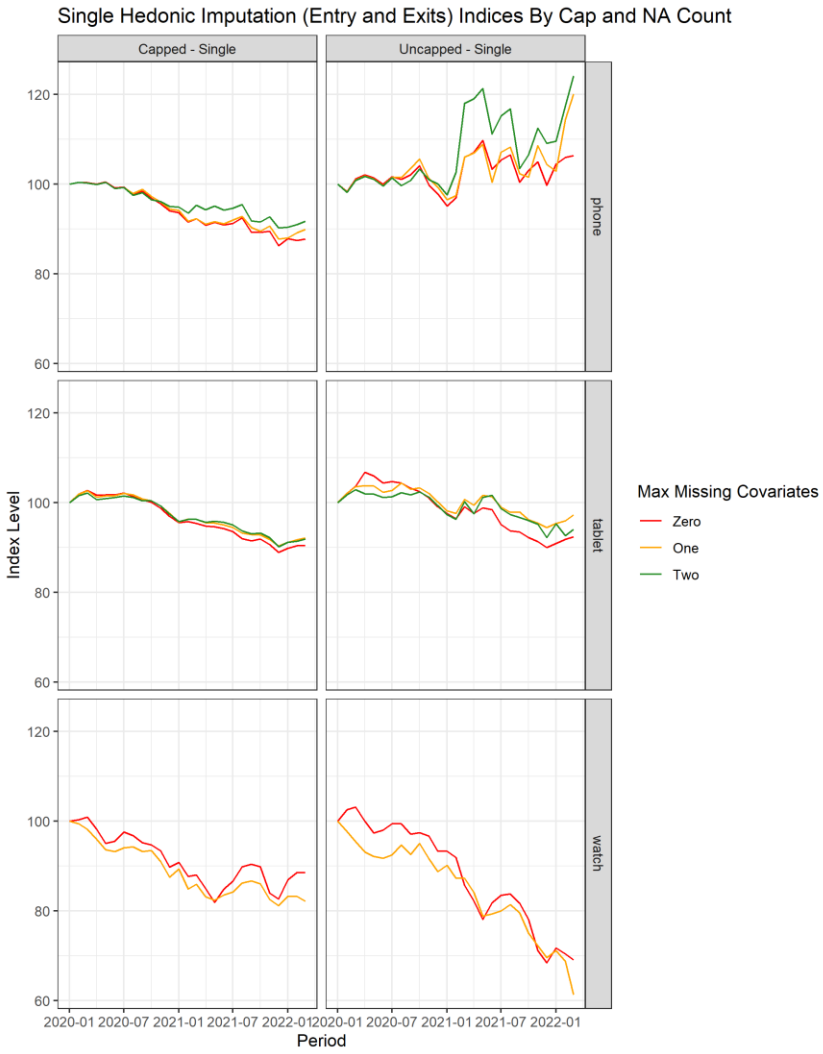
This section details the different approaches taken in using the XGBoost model to quality adjust price movements in the indices, namely:

- Capped vs uncapped predictions
- Single imputation (imputing missing prices of entries and exits) vs full imputation (imputing prices of all observations)
- Limits on the number of missing covariates used in prediction

From an operations standpoint, XGBoost potentially has an extremely desirable property – its ability to handle missing values. The algorithm uses a technique called ***sparsity-aware split finding*** when making its predictions – in easy to digest terms, once a model is trained it simply attaches a default direction[15] to each decision node. In the context of hedonic imputation, this means that predictions can still be made with XGBoost, *even when some characteristics in the data are missing.*

To test how effective XGBoost was at making predictions when characteristics were missing, a variety of single hedonic imputation indices were constructed – capped and uncapped, with varying numbers of missing covariates allowed – to see if there were drastic changes across indices when the maximum number of missing characteristics allowed was changed.

[15] For a more in-depth discussion of sparsity-aware split finding, see section 3.4 of Chen and Guestrin's 2016 paper on XGBoost.

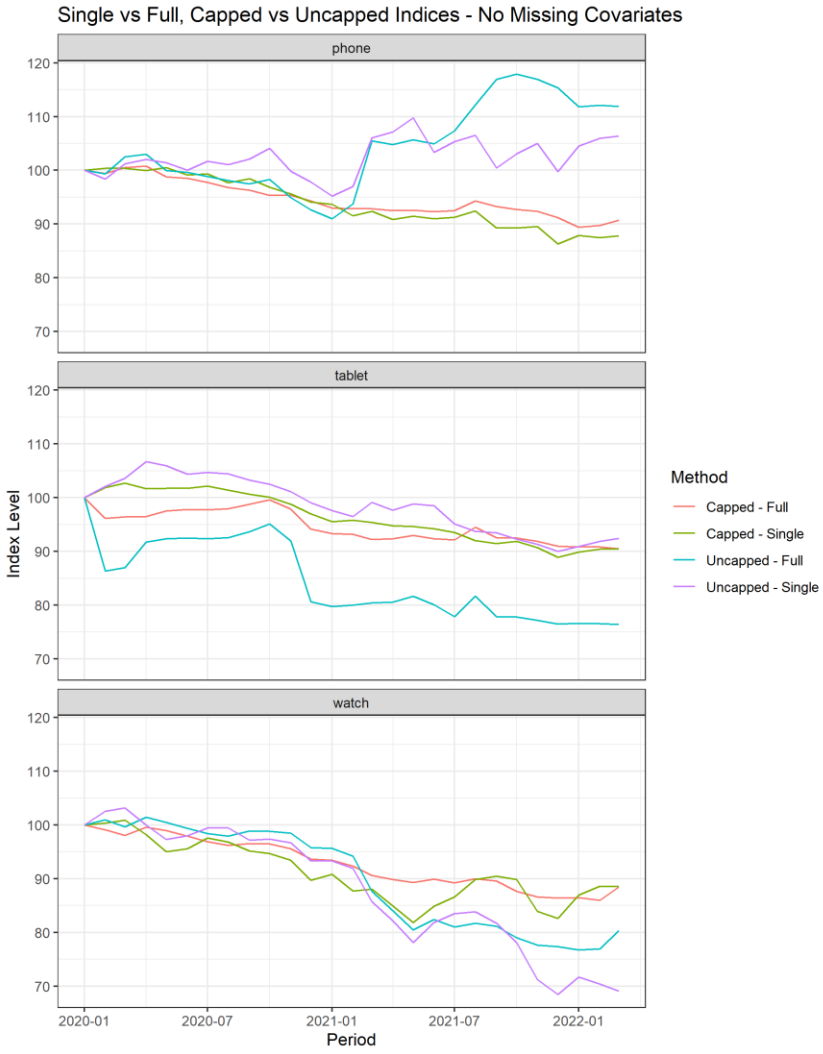


Recall that the covariates used in the watch models are all binary except for the case size, which is why the maximum number of missing covariates stops at one.

For the uncapped indices of each product, there were noticeable differences between the series involving only observations without missing characteristics compared to those using any amount of sparsity-aware split finding. This suggests that while XGBoost has the potentially desirable feature to still make predictions in the presence of missing covariates, its use would require further investigation into how to optimize it such that indices are not arbitrarily affected by its use. Additionally, it doesn't make much in the way of sense to model with missing covariates just for the sake of more observations. To justify using observations with missing covariates, it should be the case that sparsity-aware split finding increases sample size in areas that need better representation. For instance, if sparsity-aware split finding led to more observations for retailers or brands that had very high weight per observation but a low number of observations, the trade-off could potentially be justified. However, until a metric is defined to determine if segments of the data would benefit from this, there is no justification to use it in the indices.

The difference in capped and uncapped single hedonic imputation indices also helped inform the argument on capping predictions. In practice, NSO's often flag observations with large imputed price changes or set a cap on imputed values at +/- some multiple of the observed value's price. This paper experiments with a cap of 10%. The logic behind this is that there's a better chance that the *direction* of a prediction is right than the exact *magnitude* of a prediction, so capping could be used to reduce the impact of noisy predictions in an effort to reduce the volatility of an index while still maintaining the trend.

To further test the effect of capping, full hedonic imputation indices using only observations with no missing covariates were constructed and contrasted with their single hedonic imputation counterparts. It is of note that full imputation indices require vastly more predictions, and as a result could be subject to larger amounts of noise. The full and single hedonic imputation indices are contrasted below:



Particularly of interest is the capped vs uncapped results for phones. At the beginning of 2021, a major upwards price movement is observed. This coincides with the introduction of several new large-scale retailers that make up over half the sample, and lasts for approximately the length of the window. This means that the model was making predictions on observations that potentially had both different

relationships between characteristics and price than the incumbent sample, as well as a different composition of characteristics in general. Before and after the introduction of these retailers, the price movements reflect the general trend seen in the capped models – particularly in the single imputation case, which is potentially less prone to noise than full imputation is. This effect underscores the need for further research on how best to accommodate changes in the data when introducing new sources to the web-scraped data. In the case of watches, there were issues with the data generation process of the new retailers, resulting in those retailers being introduced a few months later, at which time a similar magnitude large price movement is observed. Two potential solutions to this issue could be:

- Conditional caps being implemented in scenarios where new characteristics or data sources are introduced.
- A subject matter expert making manual changes to determine price movements when price movements stray across some threshold.

Also surprising is that the uncapped full imputation index for tablets experience such volatility. With a cap of 10%, and an average MAPE of only ~5%, one would expect the cap to have little effect on the results. Conversely, with a cap of 10% and an average MAPE of over 10%, one would expect the cap to have had a larger effect on watches. Further investigation on the relationship between MAPE and the effects that capping predictions has on the indices is warranted.

To summarise the results of this robustness testing:

- Uncapped indices are preferred to 10% capped indices because the introduction of the cap led to alterations in the trend without appearing to lower volatility. Further research will look at scenarios where capping could be beneficial, such as when new data sources are introduced.
- Indices constructed with no missing covariates are preferred to indices that make use of XGBoost's sparsity-aware split finding feature because no research has yet been undertaken to determine the extent to which segments of the data would benefit from it.
- Single imputation is preferred to full imputation as the majority of the prices it uses are observed instead of imputed, and as a result it is less prone to volatile predictions.

5.2 – Time Dummy Aggregation Structures

Using the results of the calculations described in Section 4.3.1, the time dummy is then calculated as the coefficient on the current period less the coefficient on the previous period, raised to the power of *e*. This is referred to as a *movement splice* structure.

$$\text{Time Dummy Relative}_t = e^{\delta_t - \delta_{t-1}}$$

Note that in a two-period window, the time dummy movement simplifies to e^{δ_t}., as there is no δ_{t-1}.

The hedonic time dummy models were generally not able to model the data nearly as well as the XGBoost models. For products where the goodness of fit was comparable to XGBoost (ie. tablets), the time dummy methodologies produced somewhat comparable indices, and for products where the goodness of fit was far inferior (ie. watches), the time dummy methodologies produced drastically different indices. This result suggests that the time dummy indices produced inferior results to the hedonic imputation indices. It was also for this reason that *window splice* and *mean splice* structures

were not tested – given the superior modelling power of XGBoost, it was difficult to justify spending time developing robustness checks for the OLS-dependent methodologies.

The same formula applies to constructing the DML time dummies. As expected, the DML methodology produced different and more volatile indices than the other methodologies.

6 - Discussion of Indices

The plot below compares the aforementioned indices, as well as matched model indices[16] for each of the products and monthly average prices, which act solely as a reference point. Also of note is that in all cases, the weighted hedonic time dummy indices do not resemble either the matched model or the hedonic imputation indices, which potentially speaks to fact that the OLS models were outperformed by the XGBoost models.

[16] It is assumed that the reader is familiar with matched models. Note that the matched models constructed below use only observations that had no missing characteristics. This was done in an effort to make them more comparable with the hedonic imputation indices.

The comparison of indices by methodology across products highlights different results for each product. In the case of the single imputation index for phones, removing the window featuring the introduction of new retailers yields an index with little price change over the two year window, though it is quite volatile. This does not well approximate any of the other methodologies. The results for tablets are the most consistent, and the hedonic imputation index well approximates the matched model with no specifically volatile months. Lastly, in the case of watches, similar price movements across the matched model and hedonic imputation indices are observed until the new retailers are introduced, at which time the two indices diverge. In all cases, the hedonic imputation indices were less volatile than average prices – which is what one would expect given the goal of hedonic imputation, like all quality adjustment methods, is to remove the effect of quality changes on measured price changes.

These results highlight potential areas for further research. It is encouraging that the hedonic imputation indices for tablets appear to be largely unaffected by the structural break caused by the introduction of the new retailers, and research will be undertaken to determine how best to account for structural breaks of this nature moving forward. In the case of the watch indices, which experience higher volatility and worse modelling fits than the other products, work could be undertaken to either investigate whether an alternative item has similar price movements or to further refine the data and methods used.

Lastly, it is of value to discuss how these results will evolve as time passes. Due to the technical nature of the products in question, as time passes it is likely that:

- New technical characteristics that the model does not currently capture will become an important input.
- Current technical characteristics will lose relevance and become a less important input.
- Interactions between different characteristics and how those interactions relate to the price of a given product may change.

The first two points underscore the need for a regular and systematic review of the product landscape to ensure that the models are able to operate in periods where the composition of products and data are rapidly changing. The last point is addressed by the un-parameterized nature of XGBoost – which allows for the model to capture the changing effects of these interactions over time.

7 – Conclusion

This research is part of Statistics Canada's overarching goal of modernizing the methods and data used in the Canadian CPI.

This preliminary research suggests that DML is not applicable to the MDDI. However, by allowing for causal inference in an otherwise non-parametric framework, DML has the potential for increased flexibility of the application for machine learning in quality adjustment processes. For this potential to be realized, it must be possible to establish a useable relationship between characteristics and period, which was not found in the case of the MDDI.

Furthermore, our work on hedonic imputation underscores both the predictive power of XGBoost for NSO's performing hedonic imputation and makes note of both the potential and risks for the algorithm to be used in reducing the operational burden in a production environment with its ability to make predictions when characteristics are missing.