BMAN73701 Report

Group 20

List of Abbreviations

RMSE - Root Mean Squared Error

RFR - Random Forest Regressor

HGBR - Histogram **-** Based Gradient Boosting Regressor

MSTL - Multi-seasonal Trend Decomposition using Loess method

ACF - Autocorrelation Function

ML - Machine Learning

ARIMA - Autoregressive Integrated Moving Average

GBDT - Gradient Boosted Decision Tree

EDA - Exploratory Data Analysis

MoE - Margin of Error

List of Tables and Figures

1. Introduction

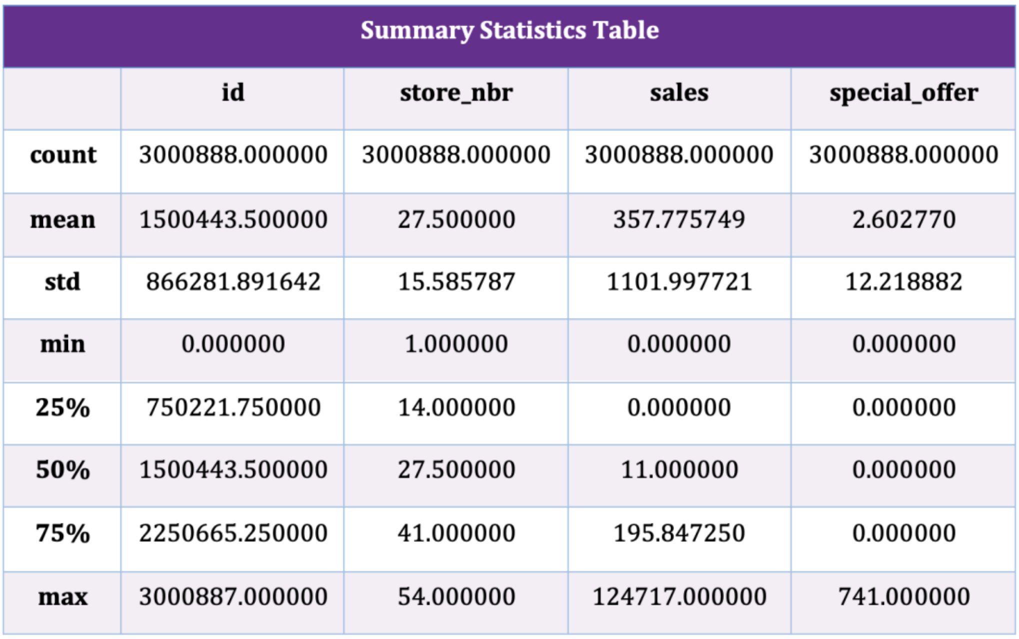
The South American grocery retailer ABC needs a method to improve their ability to predict product stocks during different times of the year. This report attempts to solve this challenge by applying a Histogram Based Gradient Boosting Regressor and Random Forest Regressor to ABC's sales data from 2016 to 2017. To achieve this, we engaged in data cleaning, feature engineering, and model fitting, which led to the selection of the HGBR approach. We developed two distinct HGBR models by applying hyperparameter tuning and cross validation and compared the models using the RMSE. Our results indicate that our second HistGradientBoostRegressor model is better suited to predict sales for ABC.

2. Data Preparation

2.1. EDA

The dataset given to us - "Products_Information.csv" - contains information about ABC's sales from 01/01/2013 to 15/08/2017 with variables including date, store identification number, product type, sales, and special offers.

Table 1. Summary statistics of numeric variables



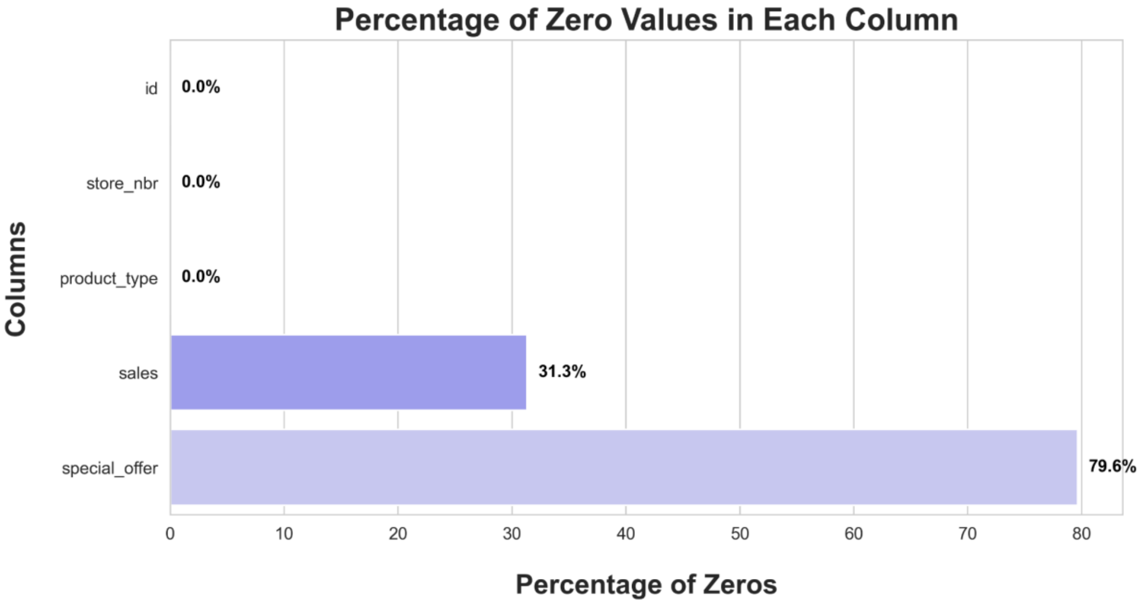
Summary Statistics Table

	id	store_nbr	sales	special_offer
count	3000888.000000	3000888.000000	3000888.000000	3000888.000000
mean	1500443.500000	27.500000	357.775749	2.602770
std	866281.891642	15.585787	1101.997721	12.218882
min	0.000000	1.000000	0.000000	0.000000
25%	750221.750000	14.000000	0.000000	0.000000
50%	1500443.500000	27.500000	11.000000	0.000000
75%	2250665.250000	41.000000	195.847250	0.000000
max	3000887.000000	54.000000	124717.000000	741.000000

Table 1 shows the summary statistics for all numeric variables in the dataset. There are more than 3M observations, raising a concern about the computational power required for our analysis. Additionally, the column ‘id’ has no meaningful value for our analysis and was therefore dropped.

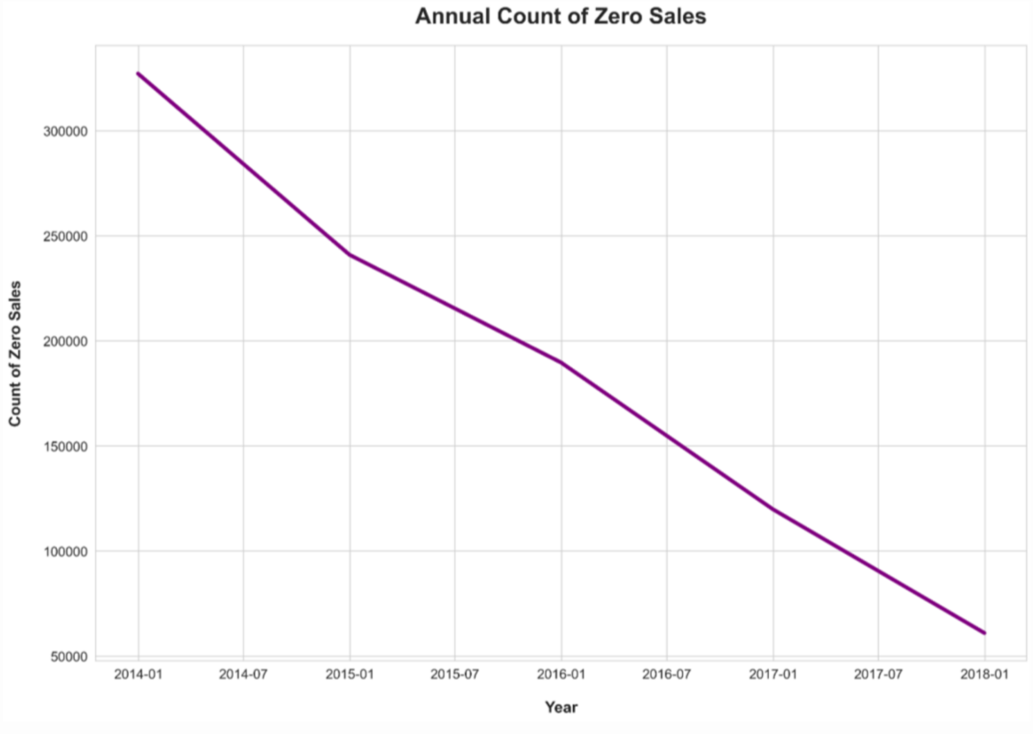
The wide range and skewed distribution of sales values could reflect varying customer responses to promotions, differing popularity of product types, and possible closures of shops for certain time periods.

Figure 1. Percentage of zero values in each column 2013-2017



There are no missing values coded explicitly as NaNs (Python notation) in the dataset. However, Figure 1 shows there are many null values in the sales and special offer variables. The large number of missing in ‘special_offer’, is particularly relevant to decision tree models whose performance is more adversely affected by sparse features.

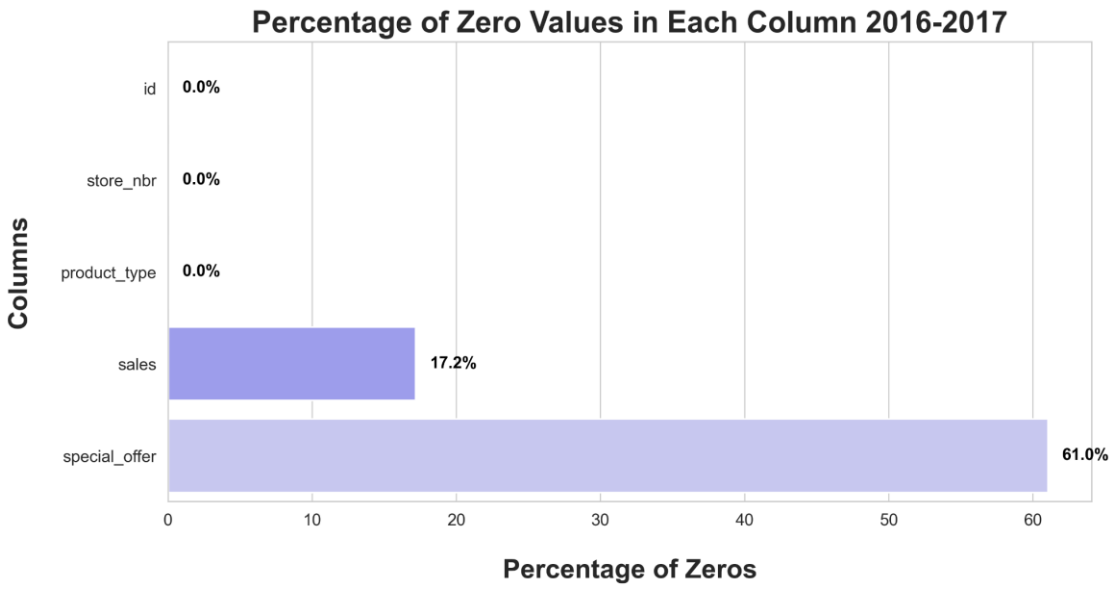
Figure 2. Count of zero values for sales by year 2013-2017



The sharp decrease in the count of zeros in sales between 2013 and 2017 (Figure 2) indicates that certain stores were potentially closed and/or opened during different points of the data collection period. Furthermore, it may suggest that certain products were introduced at a later stage.

To increase computational efficiency and utilise the most up to date information for our report, only data from 2016 onwards were used in our forecasting model. After subsetting the data, the proportion of 0 values in sales drops to 17.15%, and 61.04% in special offers (Figure 3).

Figure 3. Percentage of Zero Values by Variable 2016-2017



The null values in sales in the data subset of 2016 onwards were kept for the model training process for two key reasons:

- Firstly, many 0 values are valid: such as those for certain holidays on which stores would be closed, or when a specific product has not been sold in a day. Imputing or removing these values would decrease the accuracy of the model because it would be fed inaccurate data. Accidentally removing valid 0 values would cause the data to be missing not at random (MNAR) and introduce bias into our model.
- Secondly, there is no differentiation between valid zero values and missing values due to a lack of metadata. We do not know if stores can close on regional holidays, as we have no information on the location of ABC.

Figure 4. Average Sales By Store 2016-2017

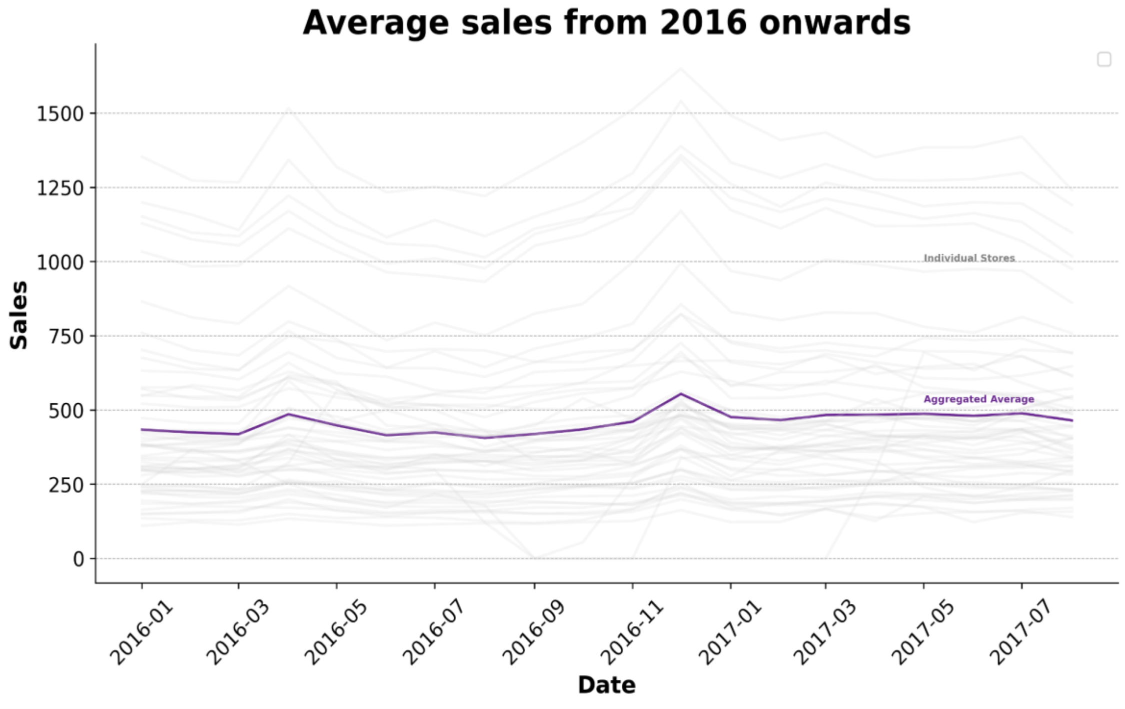


Figure 5. Aggregated Sales per Store 2016-2017

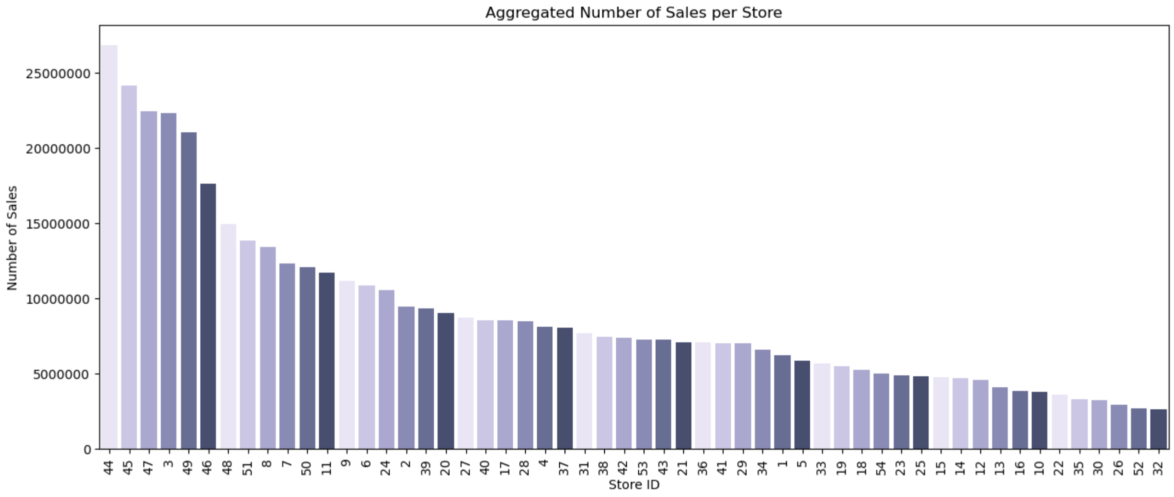
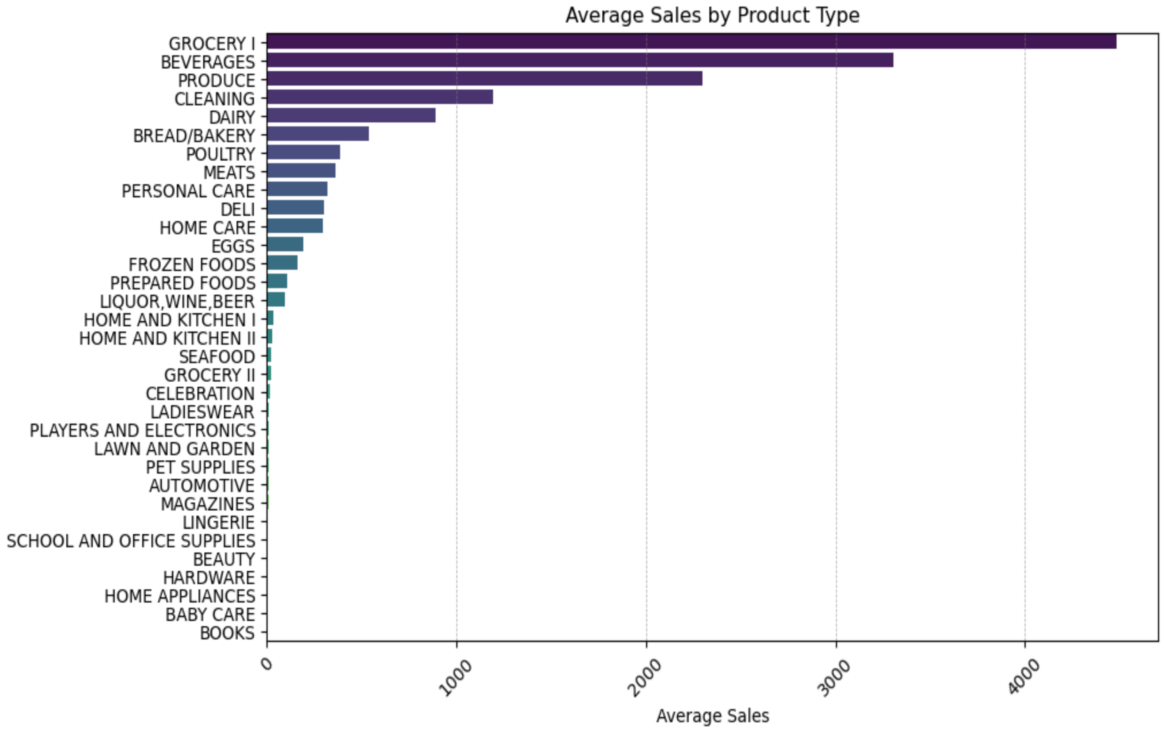


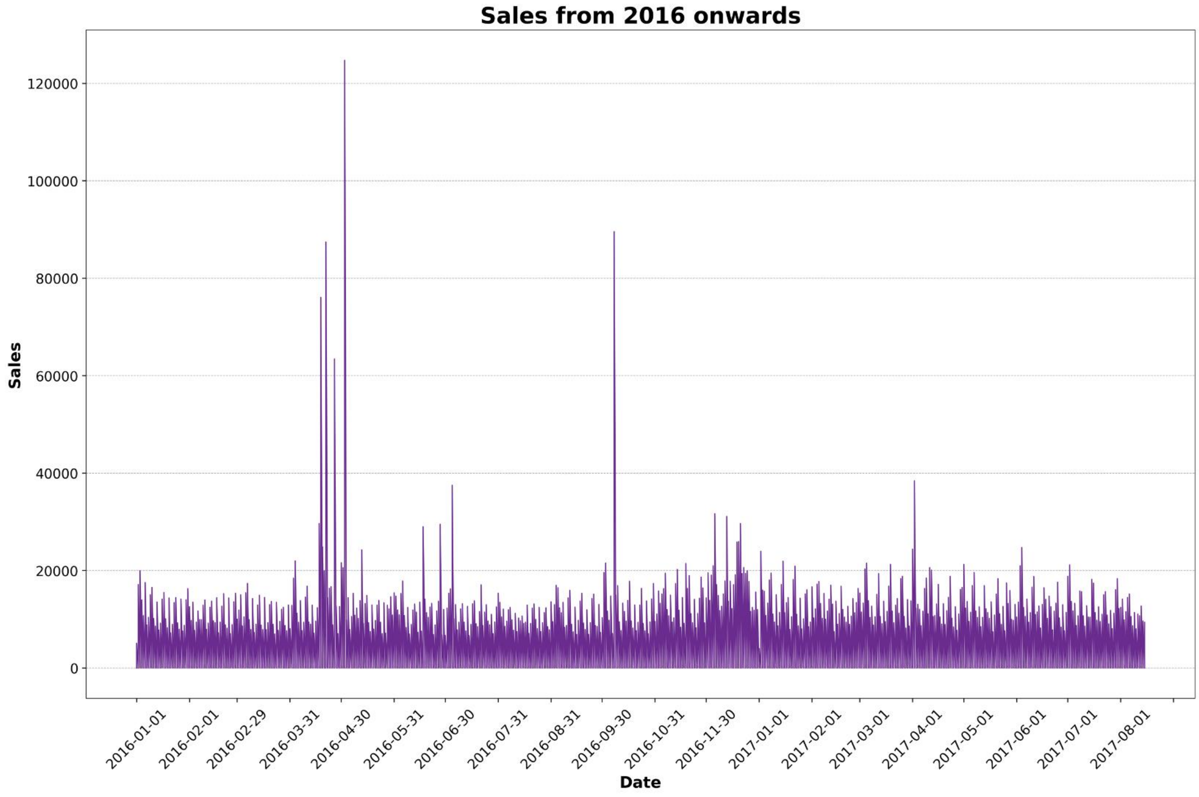
Figure 4 illustrates average sales and Figure 5 provides aggregated sales for each store between 01/2016 and 08/2017. These visualisations demonstrate significant discrepancies between the performance of individual stores, which is relevant for our feature selection process. Furthermore, Figure 4 indicates the presence of prolonged zero sales periods for some stores, which could pose limitations on our model accuracy. Figure 5 provides an overview of sales across different locations and product categories, possibly indicating regional and product specific trends.

Figure 6. Average sales by product time from 2016 onwards



Product type categories also show significant differences in sales across product categories - “Grocery I” had the highest average sales, followed by “Beverages” and “Produce” (Figure 6).

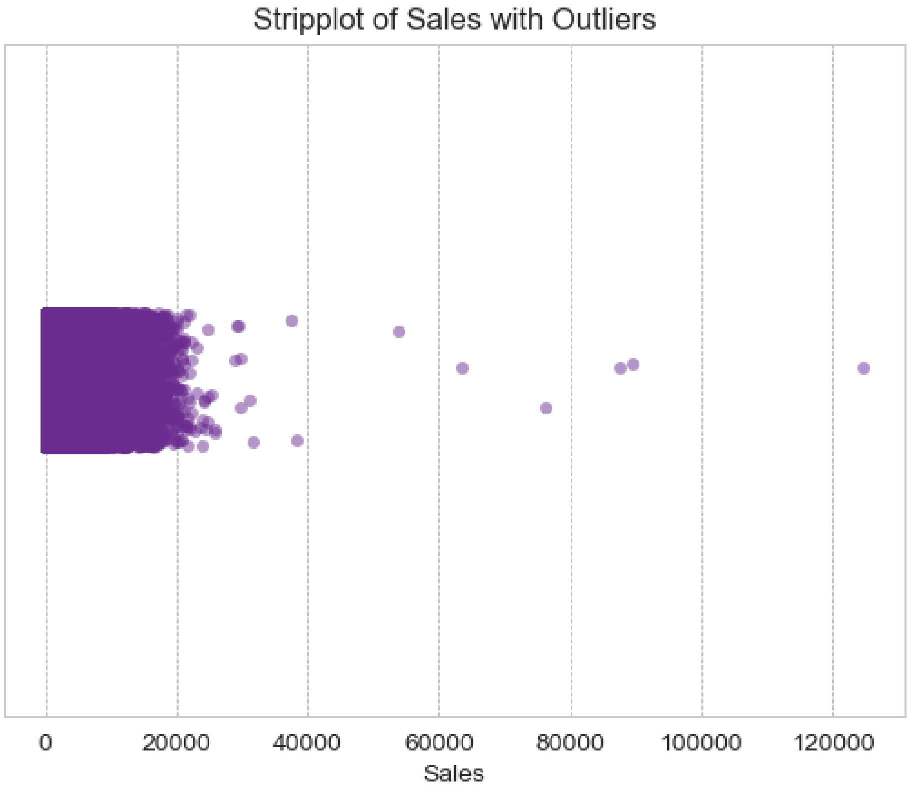
Figure 7. Total Sales 2016-2017



The outliers in the data have the potential to negatively affect model performance by distorting error metrics that are susceptible to them, thus giving a misleading account of accuracy. Furthermore, models find optimal parameters by minimising the loss function. Since outliers can generate disproportionately large errors, a model may be biased towards choosing parameters that minimise errors associated with those outliers, rather than those that optimise overall model performance.

Using 1.5 times the interquartile range to identify outliers has proved infeasible due to the high variance of the data. Hence, removing outliers based on this metric would exclude almost 400k observations from the full dataset. Therefore, outliers were identified by plotting the time series data in Figure 7, and confirmed with a strip plot (Figure 8). Values over 40k were removed, which comprises 6 observations (Penn State, 2023). This allowed us to keep variability of our data.

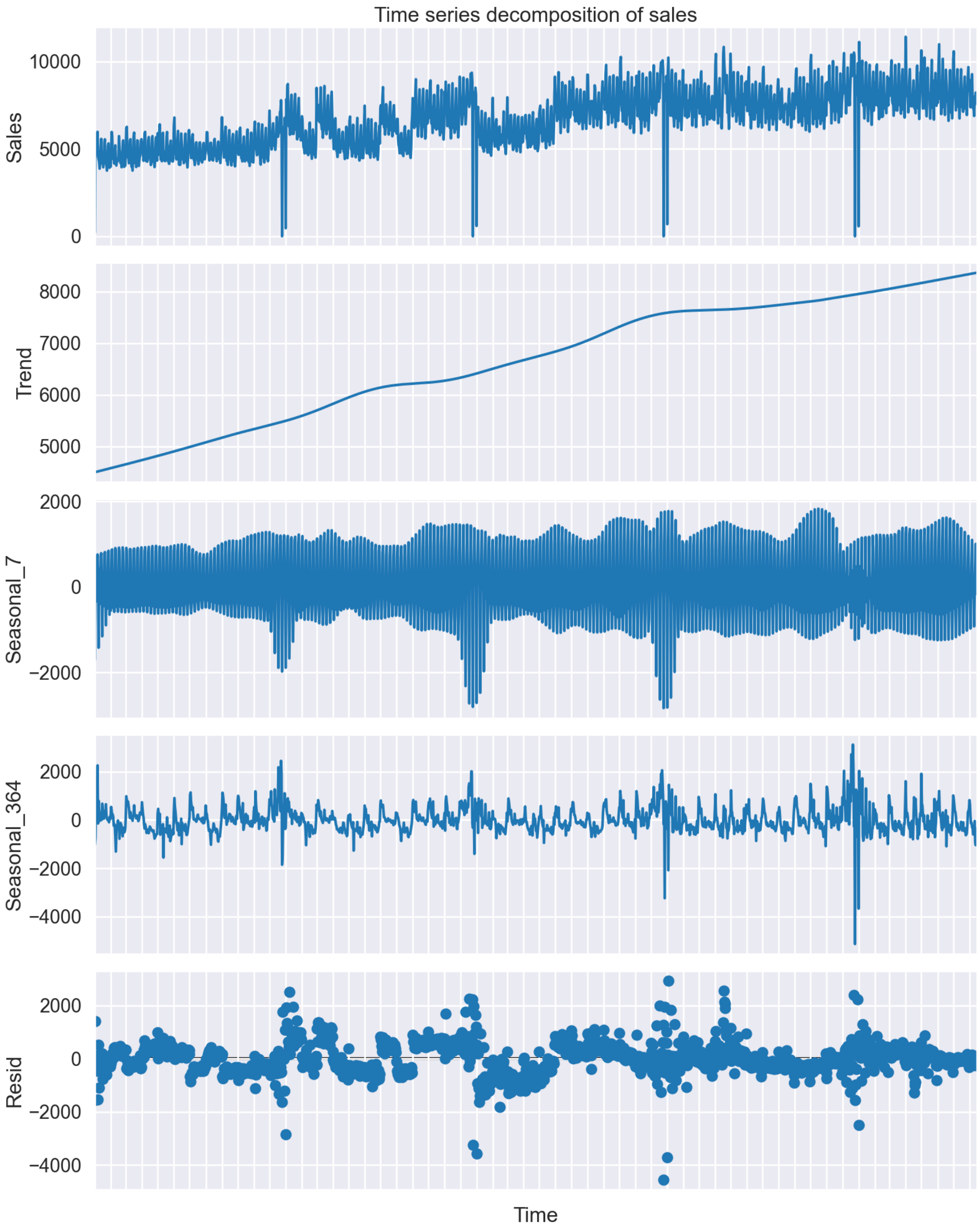
Figure 8. Strip plot illustrating outliers in sales



2.2. Seasonal EDA

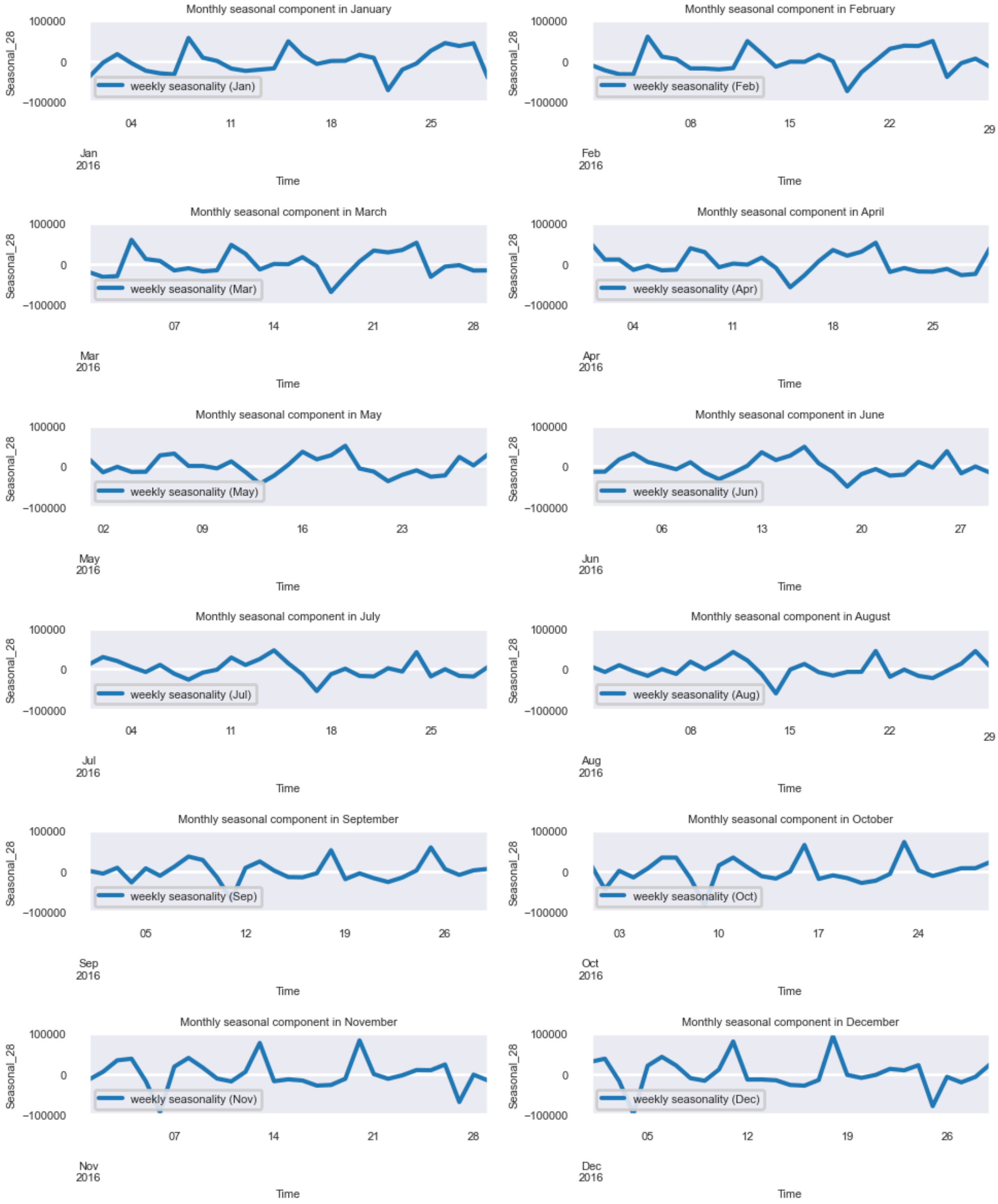
Identifying patterns in sales is essential to understanding and modelling temporal data. The sales dataset captures daily observations. Given the high frequency of the data, we wanted to explore the possibility of multiple seasonalities (weekly, monthly, annual). Time series decomposition methods enable time series to be deconstructed into three components: trend, seasonality, and residual. We selected the Multi-Seasonal Trend Decomposition using Loess (MSTL) method. MSTL can help capture multiple seasonal patterns in time series more efficiently than other decomposition methods.

Figure 9. MSTL For ABC Sales From 2013-2017



In our analysis in Figure 9, we observed a positive, mostly linear long term sales trend. Weekly seasonality, as shown in Seasonal_7, reveals increasing sales towards weekends. Seasonal_364 highlights some monthly patterns but not uniformly across all months. In 2016, as detailed in Figure 10, we noted nearly identical seasonal trends in corresponding months (e.g: January and February), suggesting multi-seasonal characteristics in our data. The residuals plot, representing unexplained variations, showed no discernible pattern, indicating that the trend and seasonal components captured most of the variability in the data.

Figure 10. Monthly Seasonal Components for ABC Sales For 2016



Overall, our EDA demonstrated that there are several patterns within the data, such as weekly trends. These findings raise implications for feature engineering and the selection process.

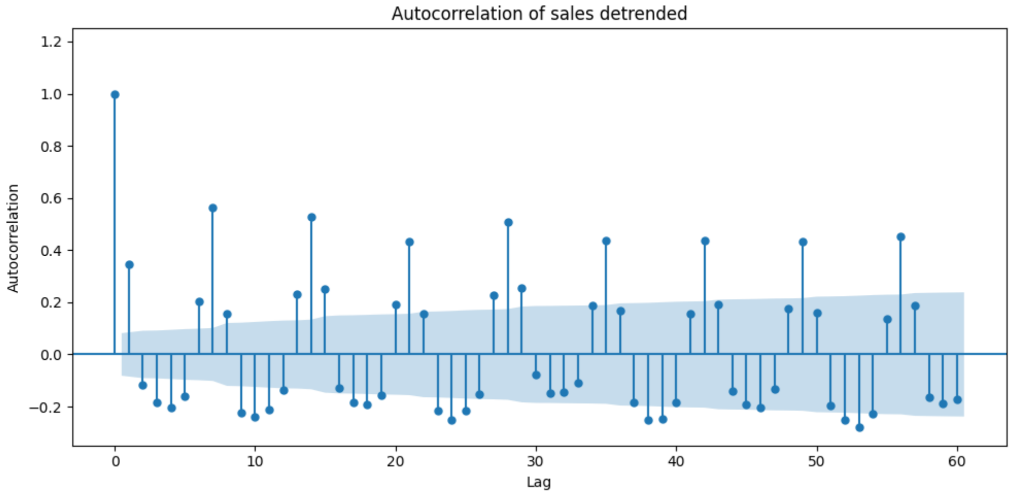
3. Feature Selection

This third section goes through the feature engineering and selection process. Three categories of additional features were created: lagged, rolling window, time-based and existing.

3.1. Lagged Features

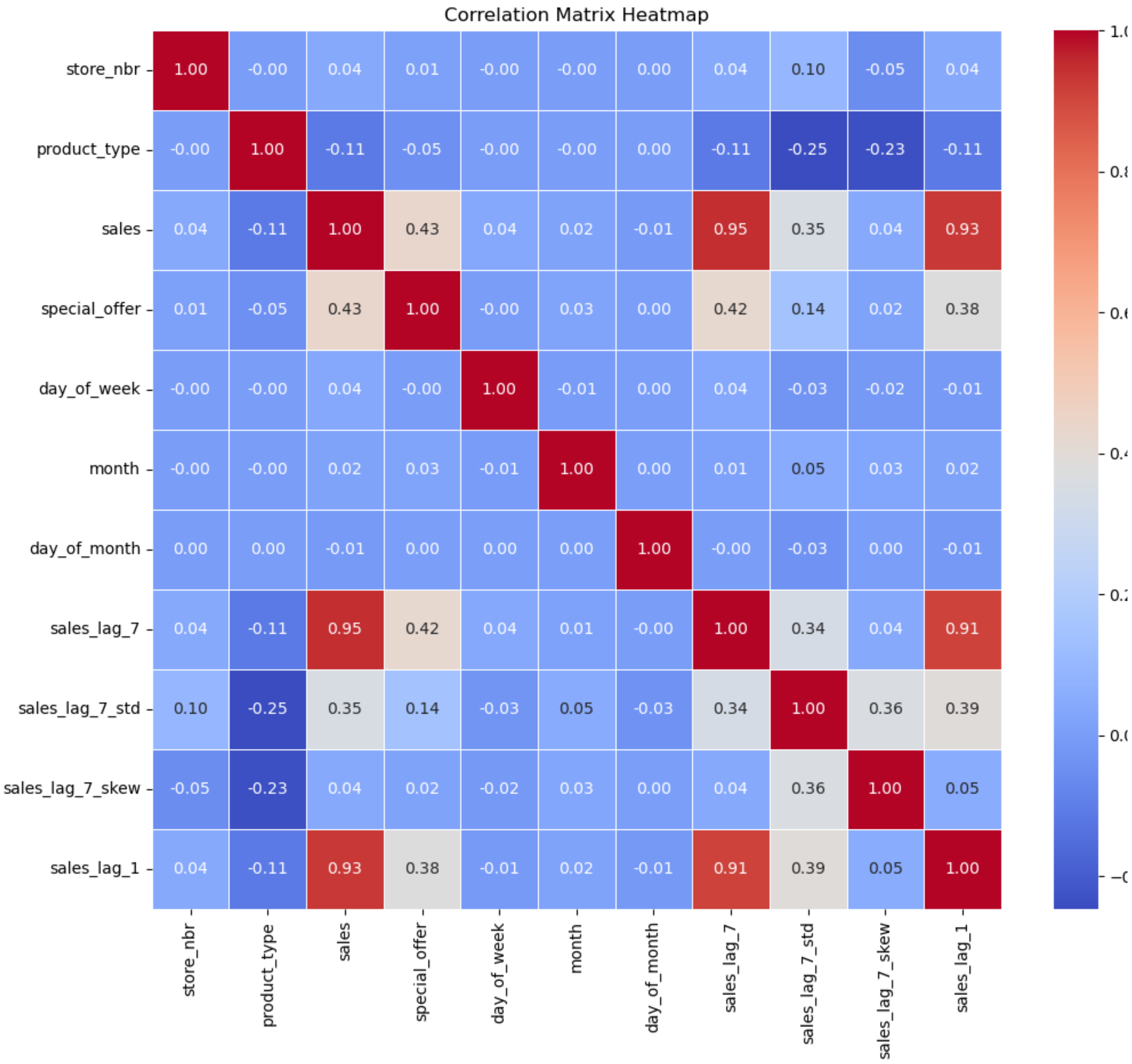
A lagged feature relates each target variable value, in this case “sales”, to a previous value of itself based on a fixed period of time. We added a lagged feature of 7 days, grouped by store and product type, to account for temporal patterns in our time series dataset, as well as the effect of past information on future predictive power.

Figure 11. Autocorrelation Function Plot For Detrended Sales By Day



Lag captures the historical values of the target variable. Figure 11 shows the presence of autocorrelation in our data. The ACF was developed after detrending the sales data to better show the inherent seasonality within it. Figure 9 demonstrates peaks at a lag of 1 and multiples of 7, and the highest correlation exists at a lag of 1.

Figure 12. Correlation Matrix of Selected Features Plus ‘sales lag_1’



While we initially considered incorporating a lag of day 1 in the model, we observed high multicollinearity (0.91) between lagged sales with a day of 1 and 7 (Figure 12). A high level of multicollinearity would inflate the feature importance of either, or both, lag sales features. This would negatively affect the performance of a model like HGBR, as it will distort its predictions based on multicollinear features rather than individual feature importance. Therefore, we decided to incorporate only lag sales for day 7 on the basis of strong weekly seasonality (Figure 11). Moreover, the 7 day lag had higher Pearson correlation coefficient with our target variable “sales” compared to the 1 day lag, which indicates stronger predictive power (Figure 12). Therefore, we decided to incorporate lag sales for day 7, since the ACF indicates strong weekly seasonality.

3.2. Window Features

Rolling windows function by aggregating a subset of the data within a fixed window, which is moved along sequential data points. By including rolling windows for the standard deviation and skewness of sales, grouped by store, and product type, our model can capture the variability and direction of extreme values for past sales over the previous week. The lack of pairwise correlations between these features (Figure 12), suggests they can provide unique information on historical sales dynamics. To ensure we did not introduce look-ahead bias, which occurs when current or future information is fed to the model (giving it distorted predictive advantage) (Baquero, ter Horst and Verbeek, 2005), each rolling window is started from a 1 day lag. In addition, this also prevents data leakage, since it restricts the model from accessing the same information it is trying to predict.

3.3. Date Features

Retail sales have weekly seasonality related to the day of the week. Extracting features from the date can help capture multiple seasonalities, especially for tree-based models. By extracting features for the day of the week, month, and day of the month, we can capture weekly and month-based seasonality, respectively. We decided against dummy variables for time, as tree-based models do not work well with a large number of sparse features (Ravi, 2019). Tree-based models can handle non-linear relationships between target and time feature variables, unlike linear models (Müller and Guido, 2017, p. 84)

3.4. Existing Features

The existing features from the "Product_Information.csv" dataset, which we selected for our modelling framework, were product type and store number. We used label encoding to replace the categories of product type with values of 0 to n_classes-1 (Scikit-learn.org, 2019), because it was initially in a string format, which would have given an error when running a regression model. Label encoding does not create dummy variables and is compatible with ML models (Scikit-learn.org, 2019). Since there are 33 different products in our dataset, dummy variable encoding would result in 33 additional features, thus adding complexity and more computational time to the model.

4. Model Selection

This section will compare linear models and ensemble methods, arguing in favour of the latter based on our data's characteristics. Subsequently, we will provide more detail on the ensemble methods Random Forest Regressor and Histogram-Based Gradient Boosting Regressor. By evaluating their performance, we will demonstrate that HGBR is the best choice for our modelling needs.

4.1. Linear Models versus Ensemble Methods

There are important differences between linear models and ensemble methods that make them suited to different types of tasks. Consequently, it is crucial to consider the task and nature of our data to select an appropriate model.

Our dataset "Products_Information.csv" is of time series format, which tends to have nonlinear characteristics and a high amount of noise, as it is generated by human activity (Liu et al, 2022). ARIMA (Autoregressive Integrated Moving Average) is often the go-to approach for time series forecasting (Newbold, 1983), however, it is not available in the scikit-learn library. Furthermore, it relies on linear assumptions about the data, which are not satisfied in our dataset.

Simpler linear models available in scikit-learn, such as linear regression, also struggle to fully capture nuances in seasonal patterns, which are typically non-linear, and are more susceptible to multicollinearity (Harvey, 1977). Moreover, linear regression models are the least robust to autocorrelation, which is the degree of correlation between observations at different time points within a series (Penn State, 2023). Hence, linear regression would not be a suitable choice for this project due to the presence of autocorrelation between current and past values of our sales data (Harvey, 1977).

Therefore, our model selection process was influenced by the ability of distinct model types to handle non-linear relationships. Ensemble models perform better on that criterion due to their structure consisting of multiple decision trees, each built on a different subset of the data (Müller and Guido, 2017, p. 83). This allows them to capture complex patterns in data without being overly sensitive to the noise that may lead a single model, like linear regression, to overfit. Moreover, they are less sensitive to multicollinearity, which makes them more stable than linear models.

Considering these arguments, we decided to implement Random Forest Regression and Histogram-Based Gradient Boosting Regression on our data and compare their performance. Both of the models have been shown to display very strong predictive accuracy on time series (Papadopoulos and Karakatsanis, 2015), which makes them particularly suited to our sales forecasting task. The corresponding ML models from sci-kit learn library are RandomForestRegressor and HistGradientBoostingRegressor.

4.2. Random Forest Regressor

Random Forest Regressor is an ensemble method that combines multiple randomly selected decision trees (Behesti, 2022). The random aspect in building the model’s trees ensures diversity, thus reducing the risk of overfitting, the latter being a common challenge in single decision trees (Müller and Guido, 2017, p. 83).

An additional advantage of Random Forest models when compared to linear models, is that they are less affected by multicollinearity due to bootstrap and feature sampling. In other words, row and column sampling enables different models to capture distinct features (Raj, 2020). RFR has also been shown to provide more accurate predictions on time series data than ARIMA (Kane et al., 2014).

However, the main challenge with using a random forest regressor for our dataset is that the complexity of the model would become substantial and require significant computational resources for optimisation (Müller and Guido, 2017, pp.87-88). This shortcoming led us to seek an alternative model that would address them.

4.3. Histogram-Based Gradient Boosting Regressor

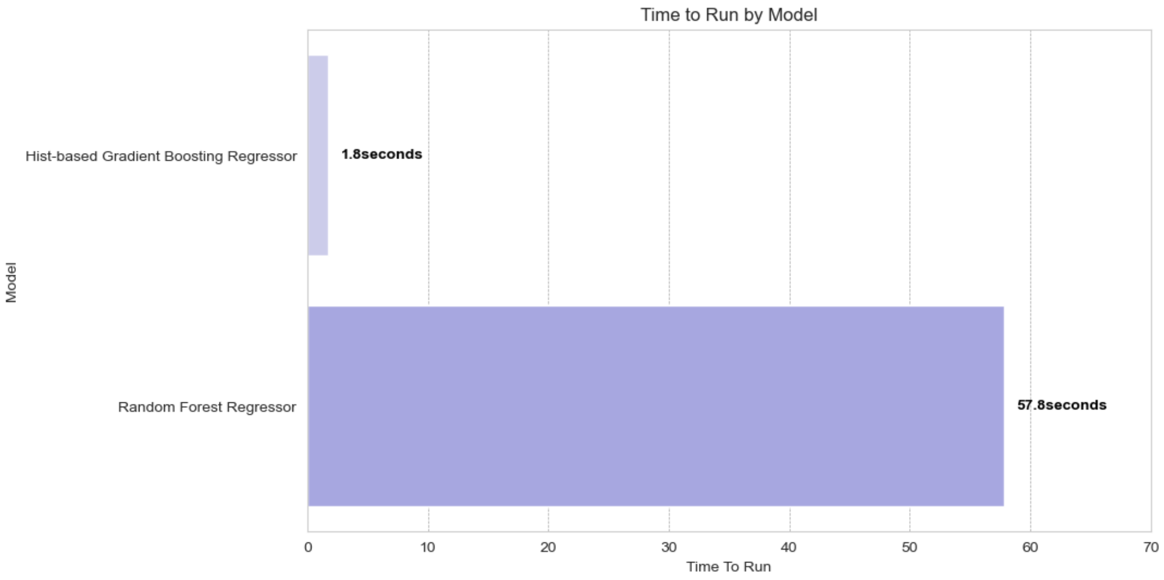
The second, and final model being considered is a Histogram Based Gradient Boosting Regressor. HGBR is available from the sci-kit library, and it is a form of Gradient Boosted Decision Tree (GBDT). GBDTs share similarities with Random Forest algorithms, however, the difference lies in the process of building each separate decision tree (Rady et al., 2021). In GBDTs, trees are constructed successively by fitting the residual error of the previous tree, resulting in improved fit of the data (Rady et al., 2021).

Furthermore, histogram-based variations of GBDTs are faster, which significantly improves computational efficiency, allowing us to train and test our data more easily than a RFR model (Scikit-learn.org, n.d.). These models improve performance by sorting continuous features into bins, similarly to a histogram, reducing the computational resources required (Scikit-learn.org, n.d.). These computational advantages make HGBR very suitable for modelling our “Products_Information.csv” dataset.

4.4 Random Forest Regressor vs HGBR

In order to assess the performance of RFR when compared to the HGBR, we fitted them on our training set which was split in 3 folds with a time series split. Importantly, this approach maintains the temporal dimension of the data, unlike a standard train-test-split (Hasanov et al., 2022). We ran both models without specifying any hyperparameters, and the same random state in order to assess their accuracy and efficiency objectively.

Figure 13. Execution Times of HGBR and RF Models



5. Evaluation and Tuning

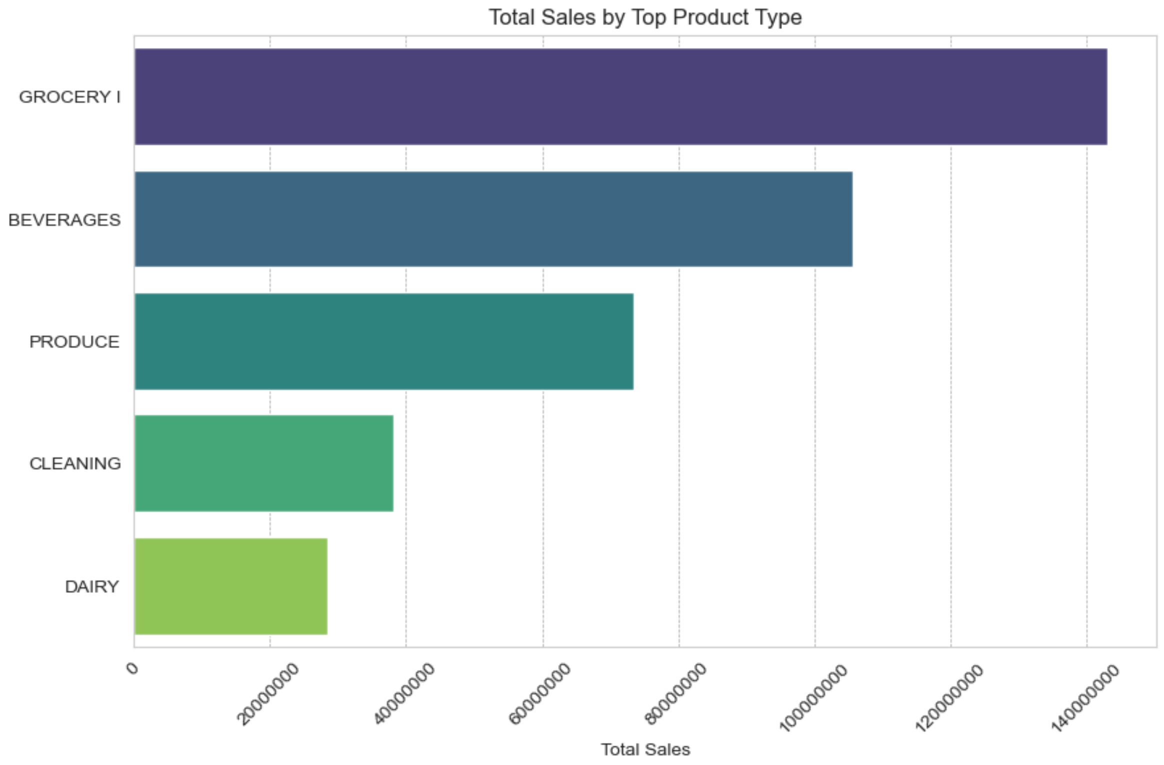
5.1. Tuning and Optimisation

We implemented k-fold cross-validation using a Time Series Split to ensure the reliability and generalisability of our model. This approach allowed us to maintain the temporal order of the data (Hasanov et al., 2022), which is crucial for accurate sales prediction of time series data to avoid look-ahead bias. The time series split, unlike traditional forms of cross validation, does not split data randomly. This avoids disruptions in time-dependent patterns. This ensures that our cross validation simulates a real-world scenario where the model is only trained on historical data to predict future values and thus avoid look ahead bias. However, TimeSeriesSplit does not utilise the entire dataset in each split, which is a necessary trade-off to maintain the temporal sequence of the data.

In order to find the optimal values for our hyperparameters, we used RMSE as our scoring metric and applied RandomizedSearchCV, which uses a random subset of hyperparameters values from a pre-defined hyperparameter grid. This leads to better computational efficiency at the cost of a minor decrease in a model's performance in comparison to GridSearchCV, since less hyperparameter combinations are used.

To assess the performance of our model during the optimisation process, we used the Root Mean Squared Error (RMSE), which calculates the root average squared difference between the actual values and the predicted values for the 'sales' feature. The advantage of RMSE is that it is expressed in the same unit as the target variable, which aids interpretability. A key aspect of RMSE compared to other scale-dependent metrics, like the Mean Absolute Error, is that it disproportionately emphasises larger errors, which is particularly important for a grocery store like ABC. Larger errors in sales forecasting are disproportionately more harmful, as they are more likely to result in excessive overstocking. This is significant for ABC, as many of their highest selling products are perishable goods (Figure 14) which will become unsellable if they remain within storage for too long, resulting in revenue loss and wastage. Furthermore, grocery stores account for a margin of error (MoE) when forecasting sales, so small errors will be more likely to occur within the MoE making them less disruptive, whilst large errors will not. Measuring RMSE, therefore, suits the business logic of forecasting grocery sales better than other performance evaluation metrics.

Figure 14. Total Sales for The Top 5 Product Types



To aid in optimising our model, we have measured feature importance using permutation importance. This measures the decrease in a model's performance after a single feature’s values have been randomly shuffled. The larger the decrease after the shuffle, the higher importance the feature displays. Permutation importance is a superior method for evaluating feature importance when compared to the impurity-based methods conventionally used in ensemble decision tree models (Altman et al., 2010). This is because impurity-based feature importance favours features with high-cardinality (Altman et al., 2010), which deflates the importance of categorical features with fewer classes, like “day_of_week” and “day_of_month”. Measuring feature importance allows us to check whether our model aligns with the expectations from our EDA, namely that features that we anticipate to strongly influence sales, like product type and stores, are identified by our model as having high feature importance. This increases the interpretability of our model for stakeholders.

5.2. Accuracy and Efficiency

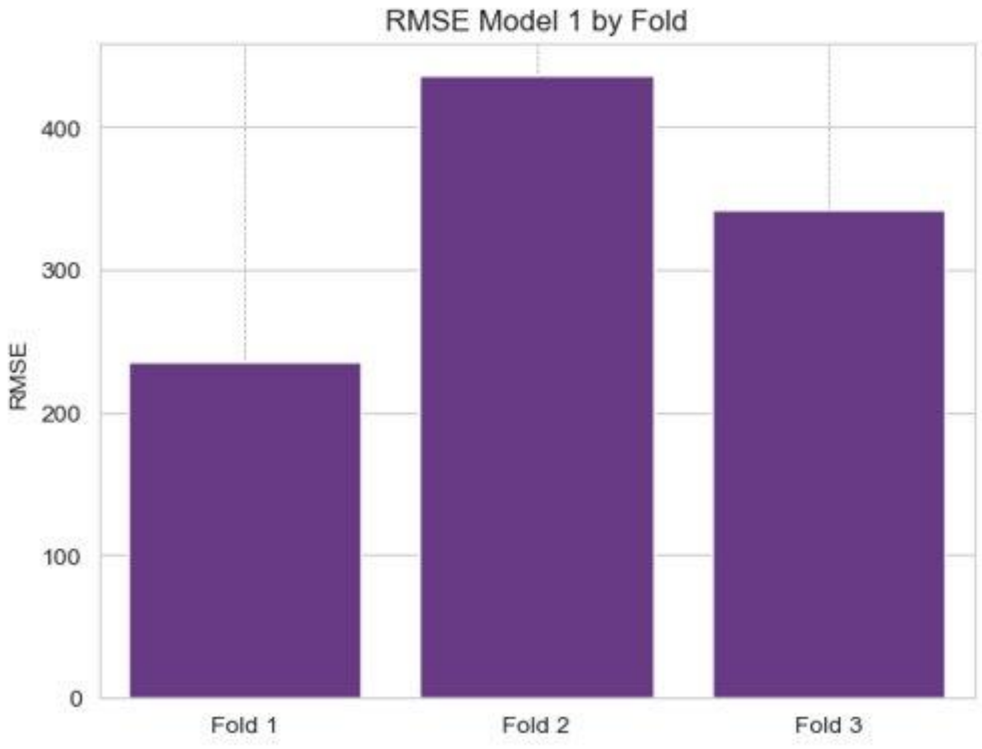
This section comprises an evaluation of the 2 variations of our HGBR model. Model 1 is the initial version of the model, while model 2 is the refined version after additional hyperparameter tuning.

After splitting the dataset into the training and test sets, we performed Randomised Search Cross-Validation using Time Series Split in order to find the optimal hyperparameter values. A random state of 20 was used to ensure reliable comparisons between models. Choosing the optimal number of hyperparameters for a ML model is often a very difficult task that relies on trial and error (Claesen and De Moor, 2015). Based on the findings from our EDA we selected 4 hyperparameters for tuning. The two most important were:

- Max_depth: this hyperparameter controls the depth of each decision tree in the model. A greater depth increases the risk that a model could overfit by capturing the noise within the data. Since we are using time series data, which tends to be inherently more noisy, as we observed in our own dataset (Figure 7, p12.), we chose to use a lower range of values (1 to 3) to counter this.

- Learning_rate: The learning rate controls the speed of convergence. The lower the learning rate, the greater the number of iterations the model can use to find the parameters that minimise the cost function the most, resulting in overall better performance at the expense of longer training times. By leveraging the high computational efficiency of our HGBR model (Figure 13, p25.), we can offset this drawback. Because of this, we chose a very low range of learning rate values (0.01 to 0.05).

Figure 15. Model 1 RMSE by Fold

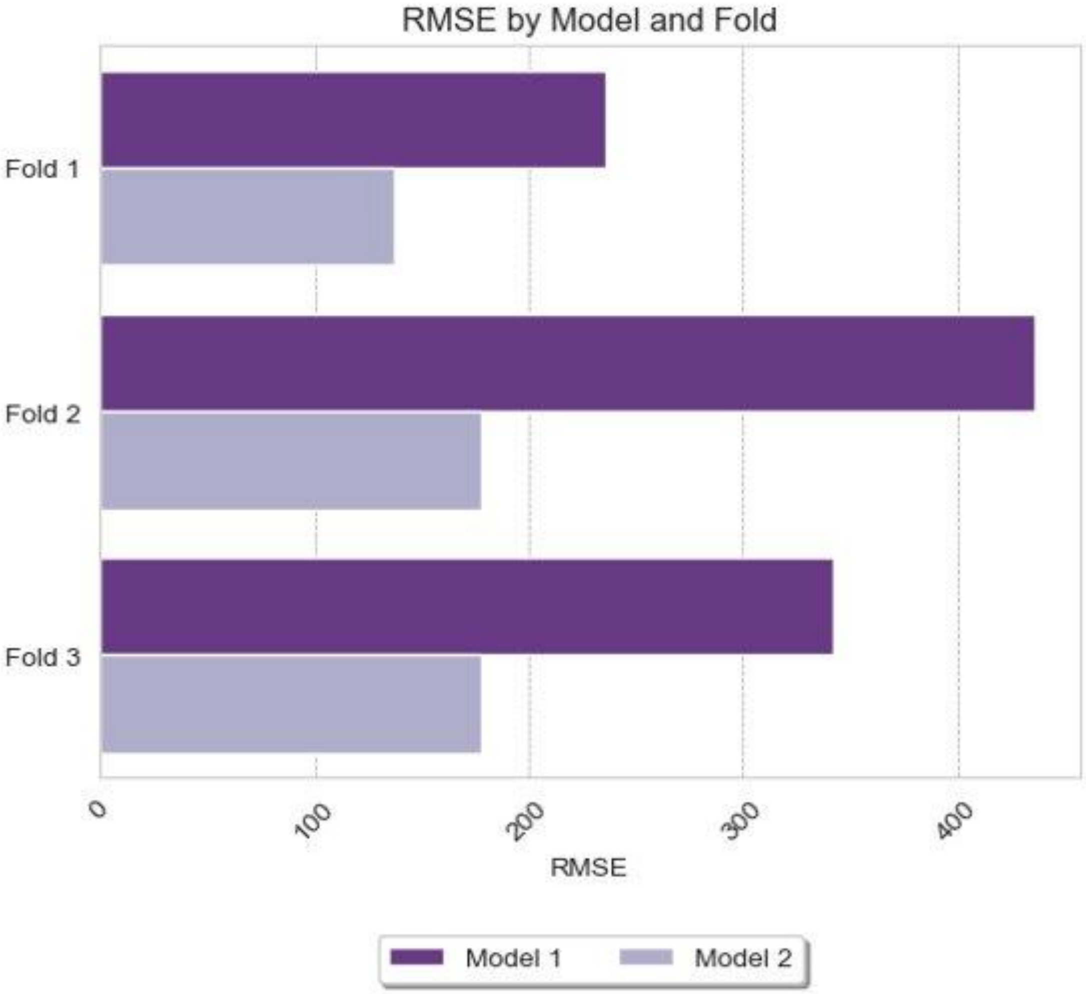


Model 1 had highly variable performance across the 3 time series folds (Figure 16), indicating it would not generalise well to a wide range of time periods. Moreover, Model 1's performance, measured as the average RMSE across the validation set folds, was 347. The RMSE score is poor, but due to the model's low run time of around 10 seconds (evidenced by the timer in our code), we could afford to perform more computationally intensive tuning. Given these factors, we decided to include two additional hyperparameters in our second model:

- max_iter: controls the maximum number of trees within the model. When specifying a higher range for max_iter, and thus a greater number of trees, we can increase the stability in the performance across folds, because the model becomes better at successfully distinguishing between genuine patterns and random fluctuations (Scikit-learn, n. d.)

- max_leaf_nodes: controls the complexity of tree nodes. A greater number of leaf nodes allows the model to better segment time series data, which reduces the chances of our model fitting to the noise of the data. This further increases stability and performance across folds during cross-validation.

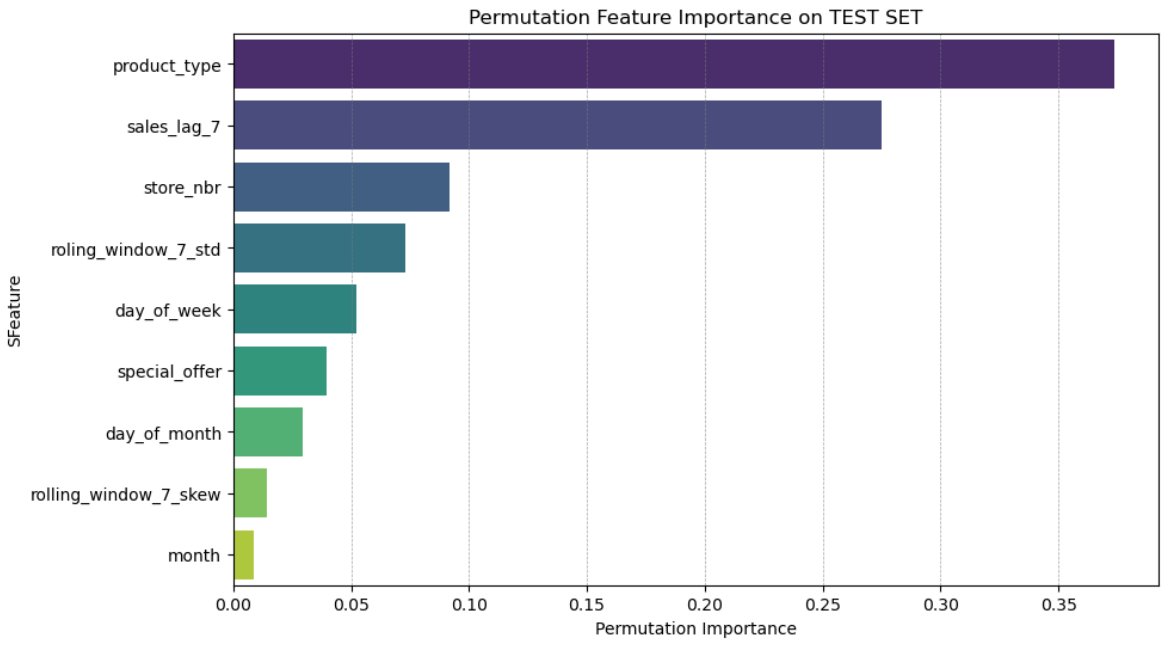
Figure 16. RMSE By Model and Fold



As demonstrated by Figure 16, Model 2 showed a consistent performance across all folds during cross-validation, indicating a higher level of generalisability. Thus, the model will perform well on a wider range of time periods. Additionally, our model still has a very low run time (under one minute), demonstrating high computational efficiency. Furthermore, our model has an average RMSE of 163 across the folds, indicating that it is drastically more accurate than Model 1.

After calling Model 2 on the test set, we observed a result of 221 RMSE, which indicates a pronounced level of discrepancy between the scores on the test set and the cross-validation set. This reduced performance on the test set is a sign that our final model is overfitting to the training data. We will perform a deeper analysis of Model 2's performance across stores and product types in Section 6.

Figure 17: Permutation Feature Importance on Test Set



After performing hyperparameter tuning on our model, we measured its feature importance on the test set using permutation importance. The results shown in Figure 17 align with most of the expectations set out in our EDA, which aids the interpretability of our model. Firstly, that ‘product type’ and ‘store number’ hold significant predictive power since they have a large amount of variability between the classes within those categorical variables (Figures 5 and 6). Secondly, that the 7 day lagged feature holds strong predictive power because of significant weekly autocorrelation (Figure 9). The fact that the observed feature importances align with our expectations improves the reliability of our model. This is because stakeholders are more likely to have confidence in our model’s forecasts when the features that are identified as important align with their domain-knowledge.

Surprisingly, the ‘month’ feature had a low level of feature importance, despite monthly patterns often aligning with traditional factors that influence grocery store sales like produce availability and holidays. A possible explanation for this is that the vast majority of the test data was from a single month, which diminishes the models ability to identify traditional monthly seasonal variations, and thus diminishes the importance of the feature.

5.3. Overfitting

Addressing the challenge of overfitting was crucial to ensure the reliability of our sales forecasting. This was done by using multiple hyperparameters with different configurations, and progressively finding the best combination through an iterative process to achieve a balance between model complexity and generalisability. The increased speed of HGBR allowed for more flexibility when testing hyperparameter configurations.

- **Max_iter:** Unlike Model 1, Model 2’s max_iter of 100 to 600 allowed us to finely tune the number of iterations in the boosting process. This range was critical to ensure sufficient learning without causing overfitting. A lower iteration might underfit, and too many can lead the model to learn noise from the training data. The optimal iteration count in Model 2 was chosen to get a balance, with better stability across folds when compared to Model 1. Model 1, with scores of 264, 436 and 342, performed worse when compared to Model 2’s scores of 137, 177, 177. The higher speed of HGBR allowed for additional headroom when tuning the number of iterations. However, this did not significantly change the results.
- **Max_leaf_nodes:** In Model 2, the max_leaf_nodes parameter, with a setting of 15 to 55, allowed for more control over the complexity of the different trees when compared to those in Model 1. Adjusting the number of leaf nodes, Model 2 was better able to segment the data enabling it to better generalise without overfitting.
- **Max_depth:** Both Model 1 and 2 used max depth, but Model 2’s broader range of 1 to 10, compared to 1 to 3 for Model 1, allowed for more nuances to be captured by trees due to greater depth. The greater depth setting in Model 2 allowed it to capture the necessary details in the dataset, without becoming too specific to the training set, which Model 1 struggled with, due to its more limited

depth range of 1 to 3. Preventing models from being too complex and learning too much from the training data helps reduce overfitting.

- **Learning_rate:** Model 2's learning rate was based on a shorter range (0.1, 0.01, 0.001) to enhance the learning speed and accuracy in convergence. We experimented with lower rates with minimal variance, but the final model incorporated a more concise range, leading to quicker learning. This approach balanced the benefits of a slower rate - gradual convergence and avoiding overlooking optimal solutions - with the speed of training and avoiding the pitfalls of higher rates such as instability.
- **Min_samples_leaf:** The wider range of values in Model 2 (20, 60, 10) allowed for more refined control over how many samples the leaf should contain - a lower leaf value leads the model to capture noise and a higher one can lead it to make overly generalised predictions. This allowed us to create more comprehensive nodes that were less vulnerable to fluctuations in the data, mitigating overfitting. Model 1's narrower range limited its ability to fine tune this balance, therefore leading to probable noise in the leaves.
- **L2 Regularisation:** This hyperparameter penalises large coefficients for features in a model, by adding a penalty term to the loss function that is equal to the sum of the square of each coefficient, multiplied by the regularisation strength factor. This balances the training process by preventing any single feature from being too influential. This causes the model to be less specific to the training data, and perform better on new data. Due to the poor performance of Model 1, which was indicative of underfitting, we chose a lower range of values for regularisation to enable a greater degree of complexity in the model so that it could better fit the underlying information in our dataset.

This specific combination of hyperparameters in Model 2 led to better performance across training and test sets, while mitigating overfitting. Its more consistent cross-validation RMSE scores across the splits and its ability to generalise better to unseen data, as demonstrated by the superior performance on the test set, makes it much more effective at preventing overfitting. This balance in parameter tuning allowed the model to capture essential patterns, while keeping it generalisable.

5.4. Challenges and improvements

Throughout data preprocessing and model building, several challenges emerged. The first notable challenge involved computational constraints due to the size of ABC's dataset. Optimally, we would have liked to include more than 2 years worth of data. Furthermore, the computational constraints meant we had to be more restrictive in feature engineering, the number of iterations in hyperparameter tuning, and other computationally expensive processes. A potential solution to this would be to leverage a cloud environment with more computing power.

The last significant limitation we encountered regarded the necessary expertise to evaluate complex improvements more comprehensively. For example, we are aware that tree-based models struggle to extrapolate data, and given our data has a strong positive trend, we could detrend our data in preprocessing and add the trend back to predictions. Determining the right method to remove the trend proved to be challenging, as it would have entailed using the trend component from seasonal decomposition. Therefore, we were unable to remove the trend in our underlying dataset. Similarly, we are equally aware of the strong seasonality between month pairs (see Feature Selection), but identifying a feature that could incorporate this required additional expertise.

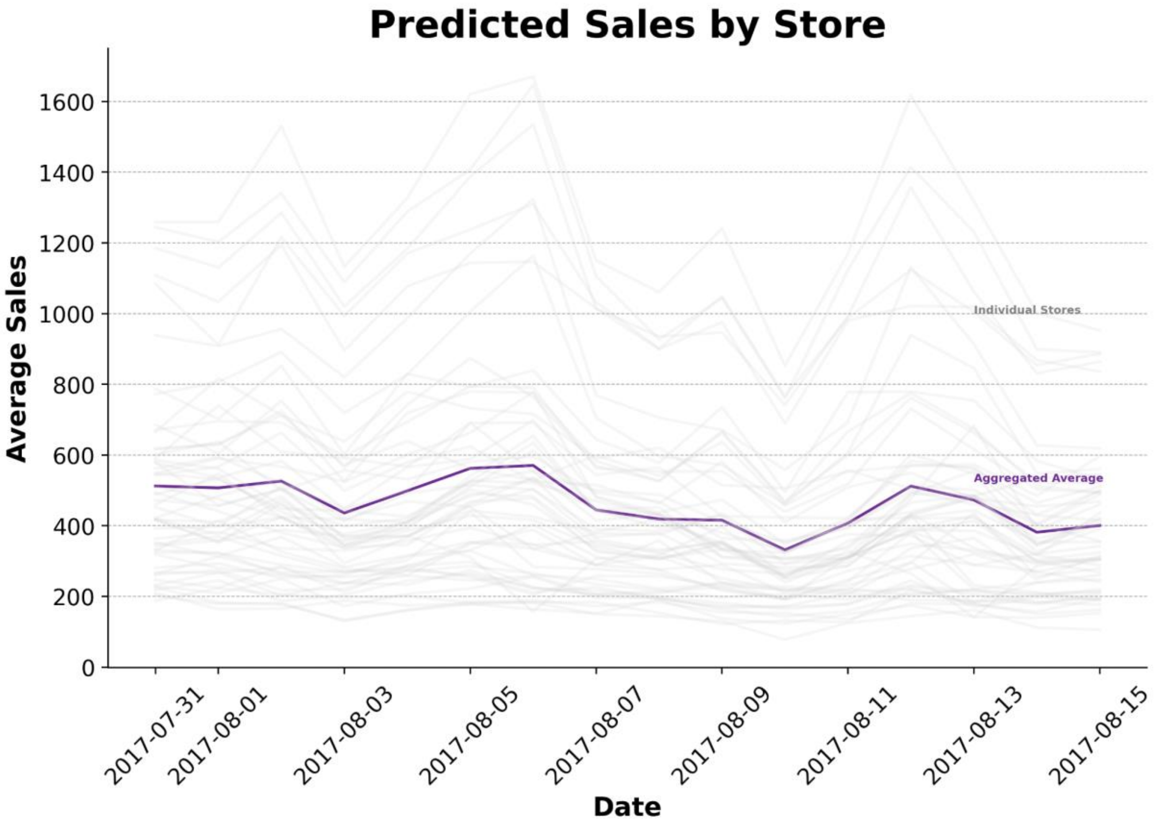
The lack of metadata left us with a relatively limited scope to enhance our model in ways that are relevant to ABC’s circumstances. Although we are aware that ABC is expanding and new stores are being created, a stronger history of the data would have aided in building a more precise model. Furthermore, with additional data, such as the location of stores and country data, we could incorporate external data points such as holidays, economic growth indicators, and other potentially casual data.

Lastly, assuming the data measures real sales and not ideal sales, the model cannot prevent understocking. If a product was understocked and sold out this would not be explicitly recorded in data. Only the sales which resulted in a product being sold out would be available. These are the sales our model aims at predicting. Therefore, our model would predict the sales which resulted in understocking previously and the store would understock once again. Further analysis should aim at identifying which products were sold out and what would be the potential sales had the supplies not ran out.

However, for the purpose of this assignment, this analysis assumes that the sales are ‘ideal sales’ which our model aims to predict.

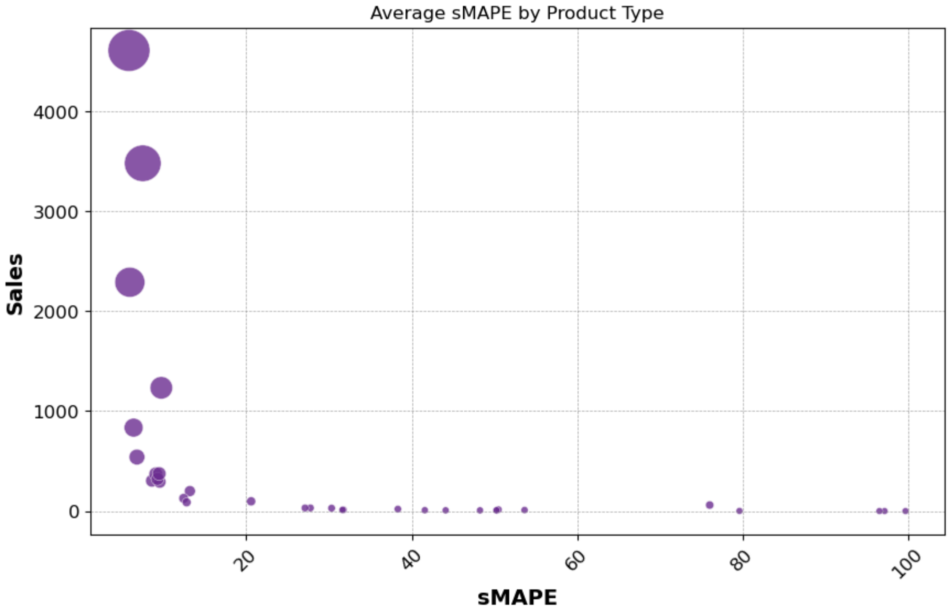
6. Results

Figure 18: Predicted Sales by Store



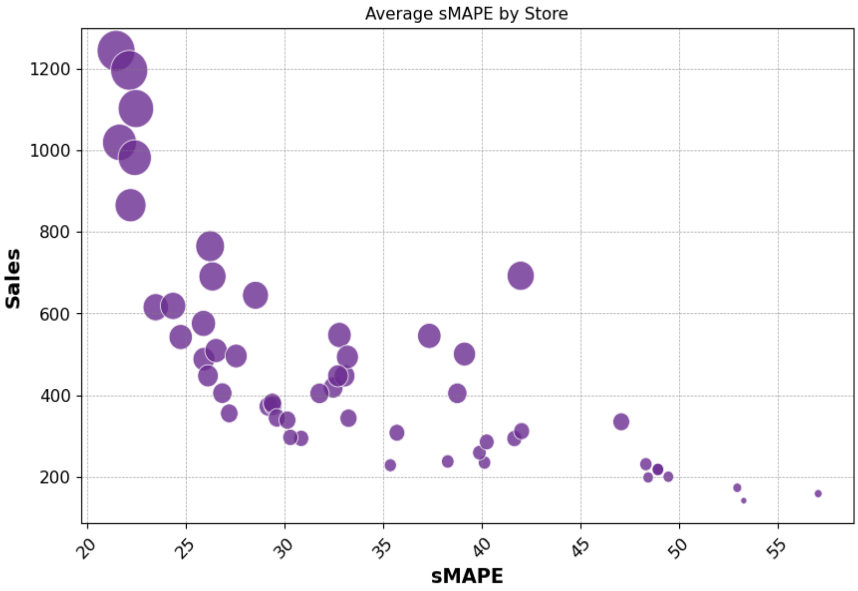
Whilst having a high overall accuracy is important, it is essential that our model is good at predicting sales in areas that are the most financially profitable to ABC. In this section, we will explain an additional benefit of our model, namely that it is good at forecasting sales for the highest revenue-generating products and stores. Symmetric Mean Absolute Percentage Error is a mean absolute error that is adjusted to the scale of the prediction (Sarra, 2022). sMAPE allows for a more interpretable error when it is broken down by product or store. The metric was calculated for each product type and store individually. Figure 18 below shows that the sMAPE is much lower for products with the highest sales such as groceries, beverages and produce, where accurate prediction matters the most. The model was less accurate at predicting a small number of product types with negligible sales.

Figure 19: Average sMAPE by Product Type



Similarly, the sMAPE shows that the model is more accurate in predicting sales for large retailers than smaller ones. The persistent focus on categories with large sales should minimise lost revenue effectively following Hans Rosling’s 80/20 rule (Gapminder, 2023).

Figure 20: Average sMAPE by store



7. Conclusion

This report presents our approach to resolving ABC's need to improve the precision of its sales predictions, using the "Products_Information.csv" dataset. To achieve this, we employed a framework consisting of data preparation, features selection, model selection, model evaluation, and results.

EDA allowed us to understand the key themes of the data and the implications for building a model. Subsequently, we created features from the data that could best capture the objective of the task. The findings of the model selection section led us to utilise Histogram Based Gradient Boosting Regressor as our preferred approach due to its versatility, speed, and flexibility. Model evaluation then assesses the model on different hyperparameter configurations, using metrics like RMSE. Lastly, by employing SMAPE, the results section evidences that our model performs effectively when predicting sales in crucial areas that maximise revenue, such as groceries.

Bibliography

Altmann, A., Toloşi, L., Sander, O. and Lengauer, T. (2010). Permutation importance: a corrected feature importance measure. Bioinformatics, 26(10), pp.1340–1347. doi:https://doi.org/10.1093/bioinformatics/btq134.

Beheshti, N. (2022). Random Forest Classification. [online] Medium. Available at: https://towardsdatascience.com/random-forest-classification-678e551462f5.

Baquero, G., ter Horst, J. and Verbeek, M. (2005). Survival, Look-Ahead Bias, and Persistence in Hedge Fund Performance. Journal of Financial and Quantitative Analysis, 40(3), pp.493–517. doi:https://doi.org/10.1017/s0022109000001848.

Claesen, M. and De Moor, B. (2015). Hyperparameter Search in Machine Learning. Available at: https://www.researchgate.net/publication/272195620_Hyperparameter_Search_in_Machine_Learning

Gapminder (2023) Factfulness:Size Available at:
https://www.gapminder.org/factfulness/size/

Harvey, A.C. (1977). Some Comments on Multicollinearity in Regression. Applied Statistics, 26(2), p.188. doi:https://doi.org/10.2307/2347027.

Hasanov, M., Wolter M., and Glende E. (2022). ‘Time Series Data Splitting for Short-Term Load Forecasting’, PESS + PELSS 2022; Power and Energy Student Summit, Kassel, Germany, 2022, pp. 1-6.

Liu, F., Chen, L., Zheng, Y. and Feng, Y. (2022). A Prediction Method with Data Leakage Suppression for Time Series. Electronics, 11(22), pp.3701–3701. doi:https://doi.org/10.3390/electronics11223701.

Müller, A.C. and Guido, S. (2017). Introduction to machine learning with Python : a guide for data scientists. Beijing: O'reilly.

Newbold, P. (1983). ARIMA model building and the time series analysis approach to forecasting. Journal of Forecasting, 2(1), pp.23–35. doi:https://doi.org/10.1002/for.3980020104.

Papadopoulos, S. and Karakatsanis, I. (2015). Short-term electricity load forecasting using time series and ensemble learning methods. 2015 IEEE Power and Energy Conference at Illinois (PECI). doi:https://doi.org/10.1109/peci.2015.7064913.

Penn State (2023). T.2.3 - Testing and Remedial Measures for Autocorrelation | STAT 501. [online] online.stat.psu.edu. Available at: https://online.stat.psu.edu/stat501/lesson/t/t.2/t.2.3-testing-and-remedial-measures-autocorrelation.

Rady E. H. A., Fawzy H., Fattah, A. M. A. (2021). 'Time Series Forecasting Using Tree Based Methods', Journal of Statistics Applications & Probability, 10(1), pp. 229-244.

Raj, S. (2020). 'Effects of Multi-collinearity in Logistic Regression, SVM, Random Forest (RF)', [online] Medium. Available at: https://medium.com/@raj5287/effects-of-multi-collinearity-in-logistic-regression-svm-rf-af6766d91f1b.

Ravi, R. (2019). One-Hot Encoding is making your Tree-Based Ensembles worse, here’s why? [online] Medium. Available at: https://towardsdatascience.com/one-hot-encoding-is-making-your-tree-based-ensembles-worse-heres-why-d64b282b5769.

Sarra, D. (2022). How to Interpret SMAPE Just Like MAPE. Medium. Available at: https://medium.com/@davide.sarra/how-to-interpret-smape-just-like-mape-bf799ba03bdc

scikit-learn. (n.d.). 1.11. Ensembles: Gradient boosting, random forests, bagging, voting, stacking. [online] Available at: https://scikit-learn.org/stable/modules/ensemble.html#categorical-support-gbdt.

Scikit-learn.org. (2019). sklearn.preprocessing.LabelEncoder — scikit-learn 0.22.1 documentation. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html.

Kane, M.J., Price, N., Scotch, M. and Rabinowitz, P. (2014). Comparison of ARIMA and Random Forest time series models for prediction of avian influenza H5N1 outbreaks. BMC Bioinformatics, 15(1). doi:https://doi.org/10.1186/1471-2105-15-276.