Store Sales Prediction

DSI Yuteng Chen

December 2023

1 Introduction

This report delves into the "Store Sales Time Series Forecasting" competition hosted on Kaggle. Our goal is to forecast sales for a variety of product families in Favorita stores throughout Ecuador. The original dataset is vast, containing over 3 million data points. For practicality in model training, we've narrowed our focus to a subset—specifically, sales data from a single store. This reduces our dataset to around 56,700 entries, enabling a detailed yet manageable analysis without losing sight of the overall sales trends and patterns.

The dataset is a time-series, necessitating specific methods for data handling, including splitting and cross-validation. Time-series data poses distinct challenges, particularly the need to preserve the data's chronological order for accurate sales forecasting. Our methodology takes these unique aspects into account to ensure robust and reliable predictions.

Additionally, the dataset contains missing values, posing another layer of complexity. Handling these missing values is crucial for maintaining the quality and accuracy of the predictions. Techniques are employed to address these gaps in the data, ensuring a robust and reliable analysis.

The implications of this project extend beyond the academic exercise, providing valuable insights for retail operations. Accurate sales forecasting is vital for inventory control, marketing, and financial planning. The insights derived from this analysis are intended to aid retailers in making informed decisions, enhancing efficiency, and improving customer satisfaction in a competitive market. In the following sections, the report will delve into a detailed exploratory data analysis, discussing the unique characteristics of the dataset, followed by the methodology for data preprocessing, machine learning model selection and training, and performance evaluation. It ends with an outlook section about what could be done better.

2 Exploratory Data Analysis

Periodic Patterns: The dataset exhibits strong periodic trends typical of time-series data as shown in Fig. 1. This is evident from the accompanying graph, which shows regular fluctuations corresponding to specific days of the

week. Additionally, there's a noticeable overall increase over the years. Understanding these patterns is crucial for accurate forecasting.

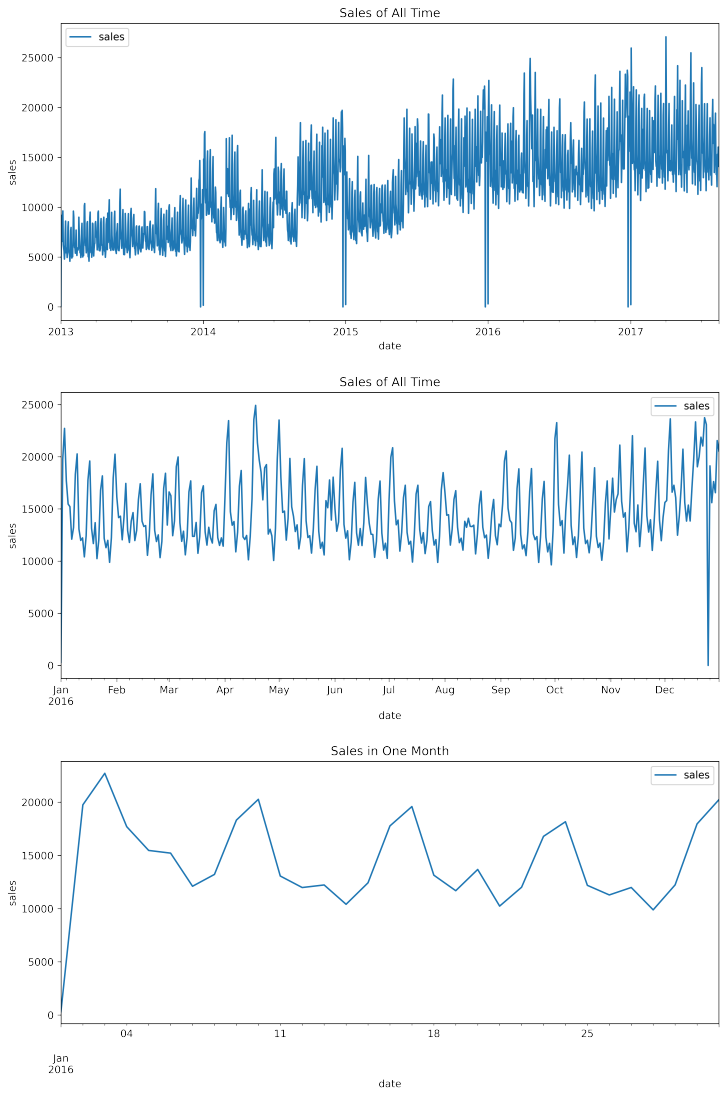
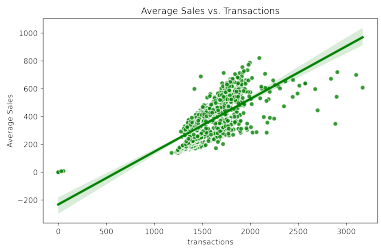


Figure 1: Sales in different time windows

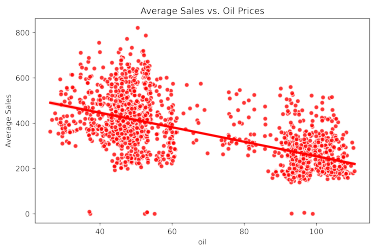
Correlation with Supplementary Datasets: Alongside the main dataset, we analyzed three supplementary datasets:

- *Holiday Dataset:* This dataset identifies holidays, classifying them as local, regional, or national.

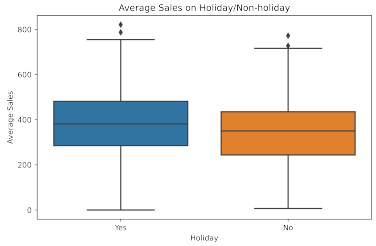
- *Oil Price Dataset:* Given Ecuador's reliance on oil exports, fluctuations in oil prices can influence market dynamics. This dataset provides daily oil price details.
- *Transaction Dataset:* This includes transactional data of different stores. The analysis how in Fig 2 reveals a moderate to high correlation between these additional features and the main dataset. This finding justifies the integration of these supplementary datasets into our primary analysis for a more comprehensive understanding of the sales trends.



(a) Transaction



(b) Oil Price



(c) Holiday

Figure 2: Correlation between supplementary features and target feature

3 Methods

3.1 Data Splitting Approach

For this time-series dataset, we adopted a date-based splitting strategy. Specifically, we reserved data points post-2016 for testing, accounting for approximately 14 percent of the total samples. Despite this being a lower percentage than usual, the large size of our dataset compensates for it. For the remaining data, we implemented time-series splitting for training and cross-validation. This method resembles k-fold cross-validation in that it continually uses a new set of data for validation. However, it differs by only training on data points preceding the validation set. This approach avoids the problem of training on

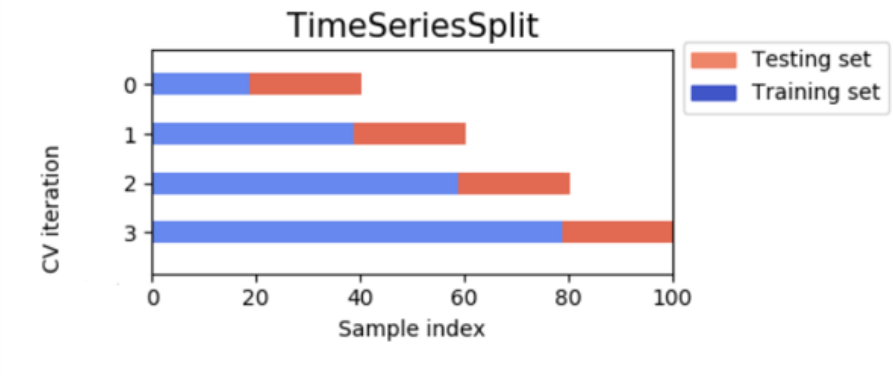


Figure 3: Illustration of time series split

future data while predicting past events. Additionally, the amount of training data increases with each iteration, allowing for more comprehensive training in later stages.

3.2 Pre-processing

Our preprocessing approach is straightforward yet effective. We use done hot encoder for all categorical features, ordinal encoder for all ordinal features, and standard scalar for numerical features. Before applying standard scalar, we used iterative imputer to impute two numerical features with 0.006 and 0.314 fraction of points missing respectively. This preprocessing ensures that our model receives data in a format that enhances its ability to learn and make accurate predictions.

3.3 Evaluation metrics

In this project, we utilize two key metrics: R squared (R^2) score and Root Mean Square Error (RMSE). The R^2 score is akin to an accuracy measure for regression tasks, with the highest possible score being 1. This metric offers a clear indication of a model's performance. However, its limitation lies in its unsuitability for baseline models. This is because the R^2 score reaches a maximum of 0 for constant predictions. To address this, we also employ RMSE, providing a more nuanced comparison of machine learning models against baseline predictions, which are based on mean or median values from the training dataset.

3.4 Models and hyper-parameters

For this analysis, we've trained a variety of models, choosing those typically covered in our class for regression tasks. We excluded Support Vector Regression

(SVR) due to its inefficiency with large datasets, as it requires significantly more training time. To optimize the balance between training duration and model performance, we have limited the number of hyper-parameters for each model to two or fewer.

Model	**Lasso**	**Elastic Net**	**Random Forest**	**KNN**	**XGBoost**
Hyper-parameters	Alpha	Alpha, L1 ratio	Max depth, N estimators	N neighbors, Weights	Learning rate, Max depth

Figure 4: Models and corresponding hyper-parameters

3.5 Machine learning pipeline

Our approach involves training each model individually, repeating the process five times. For each run, the data is initially split into training and test sets based on dates before and after 2016. The training set is then processed using the predefined preprocessor, followed by the test set. We use a time-series split with four splits for cross-validation. Grid search is applied to determine the best hyper-parameters based on the defined parameter grid. Once the optimal model is identified, it is saved along with its test scores.

4 Result

4.1 Test scores

The table below presents the test scores for all models. A notable point is the absence of standard deviation in these scores due to the consistent data splitting by the time-series method, which results in no variation in scores. However, for the random forest model, which is non-deterministic, calculating the standard deviation is feasible. The key insight from these scores is that the two decision tree models, random forest and XGBoost, perform the best, with XGBoost slightly edging out in terms of accuracy and consistency. While KNN also shows commendable performance, the linear models are notably outperformed by their counterparts.

	Lasso	Elastic Net	Random Forest	KNN	XGBoost	Baseline (mean)	Baseline (Median)
Root Mean Square Error	300.63	289.96	96.12	169.19	78.63	733.96	791.22
Standard deviation of RMSE			10.14				
R-square score	0.83	0.84	0.98	0.94	0.99	-0.01	-0.17
Standard deviation of R-square			0.01				

Figure 5: Test score table

4.2 Distribution of predicted values

Analyzing the distribution of predicted values sheds light on model performance. We've used boxplots to represent the distribution of predictions for different models alongside the actual values. A clear observation is that the decision tree models and KNN have lower errors because their predicted distributions closely mirror the actual values. In contrast, the linear models not only have a higher median and longer tails but also predict numerous negative values, which are not reasonable when predicting sales and should be avoided.

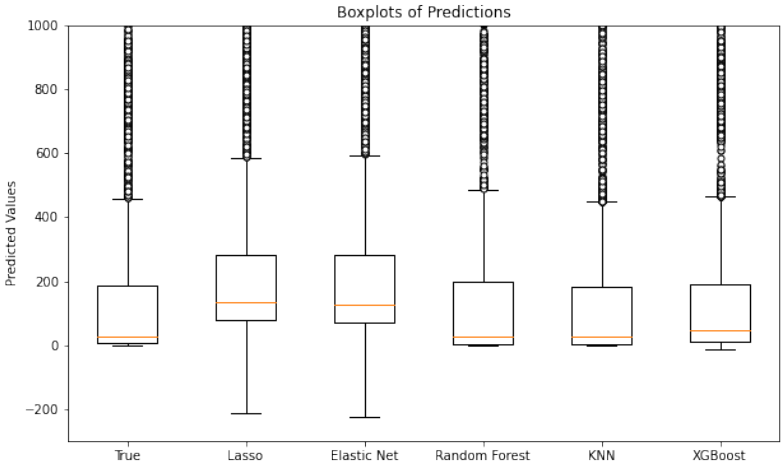


Figure 6: Distribution of predictions of different models in boxplot

This analysis is further elaborated in a graph plotting true values (x-axis) against predicted values (y-axis). In an ideal scenario, a perfect model's predictions would align with a straight red line on this graph. The closer a model's predictions are to this line, the better its performance. Here, KNN, Random Forest, and XGBoost cluster near the red line, with KNN's predictions being somewhat more dispersed. This indicates that these models are not only accurate but also consistently close to the true values across a range of target values. This consistency is crucial for a reliable and trustworthy machine learning model. On the other hand, the linear models display many points deviating significantly from this line, particularly in the target value range of 2000 to 3000, indicating less accurate predictions.

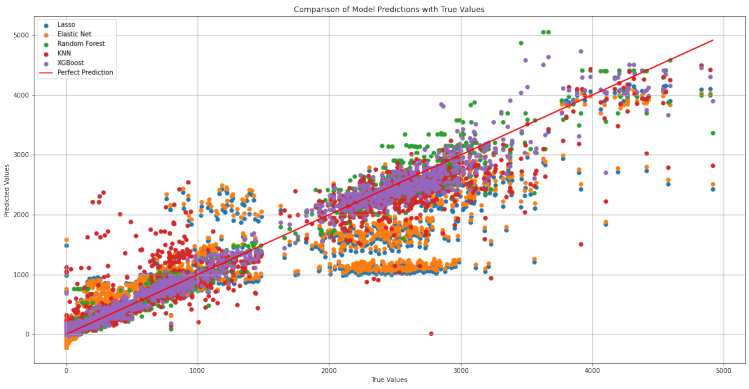
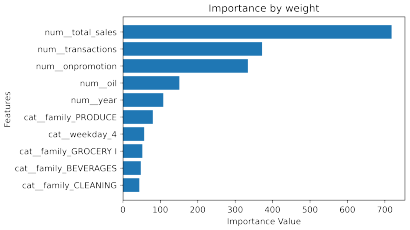


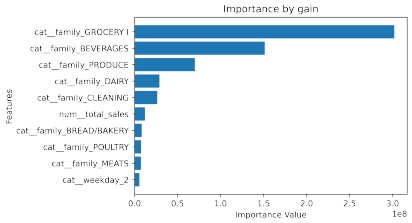
Figure 7: Predictions of models compared with the perfect model

4.3 Feature importance

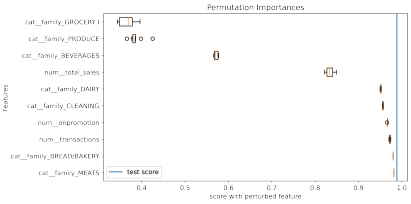
In assessing the global feature importance for our models, particularly XGBoost, we employed different metrics: XGBoost's built-in feature importance, permutation importance, and SHAP value. The resulting importance scores are depicted in the figures 8.



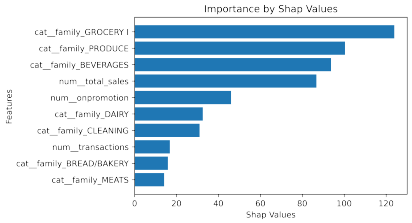
(a) XGBoost importance by weight



(b) XGBoost importance by gain



(c) Permutation importance



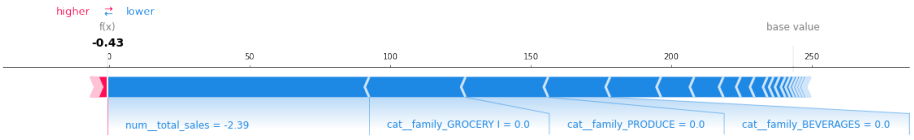
(d) Shap value

Figure 8: Various importance scores

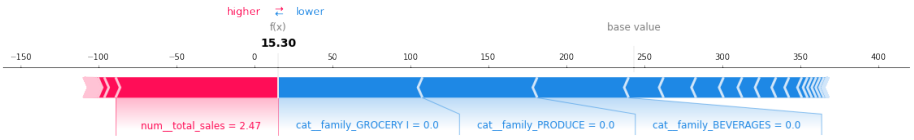
A key observation from these analyses is that the feature importance varies

across different methods and models. Nonetheless, there are consistent patterns in the significance of certain features. Notably, the categorical feature 'family', which denotes the type of goods sold, frequently emerges as a significant predictor in various models. This is accompanied by several numerical features, such as 'promotion' and 'total sales', which also consistently appear as important factors.

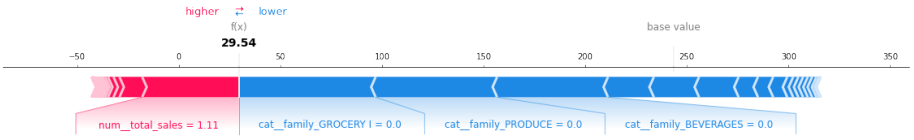
The local feature importance using Shap value also validates this point as the major contributors to the predictions are family and total sales.



(a) Shap value of data point 0



(b) Shap value of data point 100



(c) Shap value of data point 200

Figure 9: Three plots in a row

5 Outlook

Looking ahead, there are a few ways we could make our project even better:

1. **Trying More Settings:** If I had more time and a stronger computer, I'd test a lot more settings in my models. Trying different settings usually makes models work better, but it takes more time. Right now, I haven't used many settings, so there's a good chance the models aren't as good as they could be. While I don't think the simpler models (linear models)

would improve much, the more complex models might do better with more settings.

2. **Early Stopping:** Another idea is to stop XGBoost from learning too much, which can make it too specific to our current data and not good at predicting new data. I didn't use this method before because it's a bit tricky to do with the way I was testing settings, but it could make XGBoost even better.

3. **Finding Better Features:** I would also spend more time picking and creating features to use in the models. Choosing the right features is really important for making good predictions, so working more on this could help the models perform better.

These changes could help us get even better at predicting sales, making our project more useful.

6 Reference

1. Kaggle Competition - Store Sales Time Series Forecasting: https://www.kaggle.com/competitions/store-sales-time-series-forecasting