Predictive Analysis for Grocery Store Sales Using Time Series Data

Group #6

Cory LeRoy
University of California - San Diego
Halicioğlu Data Science Institute
San Diego, California
cleroy@ucsd.edu

Alec Bothwell
University of California - San Diego
Halicioğlu Data Science Institute
San Diego, California
abothwell@ucsd.edu

Will Earley
University of California - San Diego
Halicioğlu Data Science Institute
San Diego, California
wearley@ucsd.edu

Abstract:

Throughout the course of this analysis, we sought to minimize Root Mean Squared Logarithmic Error (RMSLE) on time series data from an Ecuadorian grocery store chain. We have done extensive research to explore and understand the most effective models in predictive analysis with time series datasets. This research has led us to the following models: Ordinary Least Squares (OLS), Seasonal Auto Regressive Integrated Moving Average (SARIMA), eXtreme Gradient Boosting (XGBoost) and FBProphet models. While working through the data science pipeline, exploratory data analysis, and feature engineering we were able to reveal key relationships and trends which we incorporated in our models. Each step in the pipeline and set of hyperparameters we tuned helped lead us to the extremely accurate and predictive model seen in this paper. Our research can lead to significant impact including reducing food waste throughout the world and improving food security in developing countries.

I. Introduction

Up to 30% of food in an average grocery store is thrown away every year, leading to up to 16 billion pounds of food waste in the US alone[6]. Through somewhat simple, yet elegant machine learning models, any grocer, especially those in developing countries, can more accurately predict the amount of food they should have on their shelves to meet the needs of the customers, but not waste inventory. This is where we come in, we seek to create a low cost, highly predictive, and easy to implement machine learning model that can help the Ecuadorian Grocery store Corporación Favorita, better serve their customers and increase food security throughout the nation.

The input to our algorithm is a row of features, including a date, store number, sales promotion, information about the grocery item family, the price of oil that day, and any holidays that took place on that day. We then use a few different models including SARIMAX, XGBoost, and FBProphet to output a predicted sales amount for that item on that day. These inputs have produced accurate results through our modeling techniques. This is a regression problem using supervised learning. Our main evaluation metrics were Root Mean Squared Error (equation 1) and Root Mean Squared Logarithmic Error (equation 2), the equations for which can be found below.

$$\text{RMSE} = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}(yi - \hat{y}i)^2} \quad (1)$$

$$\text{RMSLE} = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}(log(xi + 1) - log(yi + 1))^2} \quad (2)$$

II. Related Work

In today's grocery market, it's hard to find grocers that still crunch sales numbers manually and predict sales without the use of machine learning. Larger grocers opt to use large SAAS (software-as-a-service) companies that claim to have the best performing forecasting models for the job. One such company is Algonomy[14], who offer a demand forecasting model, and provide a solution that fits a model specific to the customer's needs. This model seems to be the gold standard in the market, as many large restaurants and grocers use this solution, but there's little detail to what the underlying models look like. Other approaches to large-scale forecasting include using high gradient boosting regressors that imitate gradient boosting regressors[15] but are more suited for large data. These models perform great for large grocers that have no shortage of funds and a somewhat predictable population to serve. These models may suffer when it comes to smaller grocery stores, markets, and stores in third world countries.

The second grouping of models are tuned to a specific grocer, often small, and looks to local trends. This is most like the problem discussed in this project. One paper explored the impact of many different models, like ARIMA, Decision trees, XGBoost, KNN, etc. on the final RMSLE[16]. SARIMAX showed the best predictions for this dataset, and the feature engineering had a large impact on the accuracy in this model. These are powerful models, but this study could have benefited from using some of the plug and play models like Prophet and OLS. Another research paper used KNN, Random Forests, and gradient boosting to predict on a dataset that had quite a bit more features than this project did[17]. Random forests proved to be the best predictor for this model, but this model was fitted on a very different dataset than the one in this project. Another paper that looked at XGBoost[18] specifically also had a similar feature set to the one worked on for this project. Lots of feature engineering was done on this set and the XGBoost model was trained to come up with a set of predictions. This study lacked a clear loss metric but gave a good foundation to build off. Lastly, was a paper[7] centered around deep learning and time series forecasting to reduce grocery waste. CNN seemed to be a good predictor for sales over time, and offered ideas that were used in this project.

III. Dataset and Features

The train dataset is of size 118,946 KB. There are 3,000,888 observations with 6 features which are id, date, store_nbr, family, sales, and onPromotion. The dates range from 2012-12-31 to 2017-08-15 Train was linked to the following tables:

- Holidays: contains the dates of holidays at the city, state, and national level and when they are observed
- Stores: contains store locations and cluster the store belongs to
- Oil: contains the daily oil price. There are some null oil prices and many dates that are missing from the dataset

A data diagram of the table structure is shown below (figure 1).

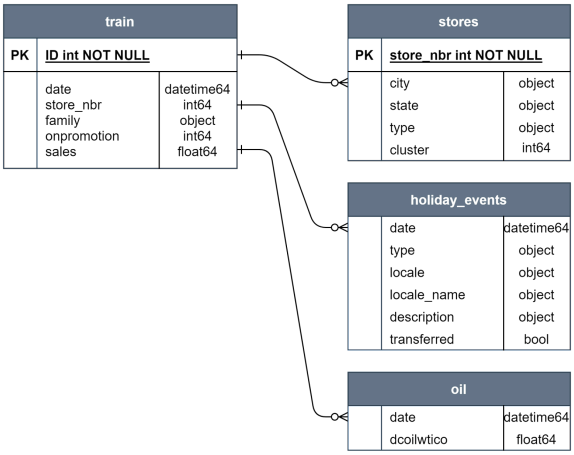

Figure 1: Data diagram

With the given dataset, each row is aggregated for the day per family and store. The predictive task was chosen to be done at this level. The prediction could have been done at a higher level of aggregation such as total sales across all families and stores but a more granular approach would better inform stores what products need to be stocked on what days and at what locations.

Feature engineering was performed to get the final dataset the models would work with, see table 1. Below is a summary of the feature extraction steps:

- Joined train, oil, holiday events, and stores
- Split date into day_of_week, month, year
- Stores have rows with 0 sales for all features when they are closed. Some stores close for months, others only open midway through the collection of data. Rows when stores are closed are removed from the training set to prevent the closed sales from impacting predictions
- Oil has 43 null values in price and is missing 481 dates completely. Oil prices were interpolated for the missing values
- 33 distinct families reduced to 6 clusters using k-means
- There are holidays at the city, state, and national level. Some holidays have a huge impact on sales, others have no recognizable impact. OLS was used to determine which holidays are not statistically significant so they can be removed. 18 holidays remain out of the original 103
- Store_cluster, holiday, family_cluster are categorical features which are one hot encoded
- All features aside from date and sales are scaled to be between 0 and 1

Finally, data is split 70%-10%-20% between train, validation, and test ordered by date.

Field Name	Field Value	Description
date	2013-01-11	Datetime64 of sales date
store_nbr	1	Int64 of id for the store
family	BEVERAGE	Object value of the category of item sold
familycluster	1	Int64 values01 through 5
onPromotion	0	Int64 value of number of items on promotion
storeCluster	13	Int64 value which represents the ID of the cluster a store belongs to
holiday	Carnival	Object value of the name of the holiday on a given day
oil_price	97.01	Float64 value of the price of oil on a given day
sales	172.0	Float64 value of total sales for the product category in a store on the given date

Table 1: Dataset after feature engineering.

IV. Methods

A. Ordinary Least Squares (OLS):

OLS[1] is a simple linear regression model which minimizes the sum of residuals between the observed and predicted values. It is an econometrics model that is used in virtually every industry. OLS is easy to interpret which is why it will be used as a baseline model. OLS only accepts numerical inputs which means that any categorical fields need to first have dummy variables created for them before being fed into the model. The model outputs coefficients of the features which translate to 'for each increase in the given independent variable, how much will the dependent variable increase'. This makes it easy to see the impact and relationship between each feature compared to the value being predicted. The relationships between variables in OLS is given below (3).

$$Y = \beta_0 X_1 + \ldots + \beta_n X_n + \epsilon \quad (3)$$

- Y is dependent variable being predicted which is sales
- $X_1, \ldots, X_n$ are the features
- $\beta_0, \ldots, \beta_n$ are the coefficients of the features which quantify the impact of each feature
- ϵ is the error term

Equation (3) is optimized by minimizing the sum of squared residuals (4). Statsmodels.api[1] was used which uses Moore-Penrose pseudoinverse to optimize the least squares. This is a linear algebra method.

$$\Sigma_{i=1}^{n}(Y_i - \widehat{Y_i})^2 \quad (4)$$

- Y_i is the observed actual value of an observation
- $\widehat{Y_i}$ is the predicted value

OLS is evaluated using R^2. R^2 is a value 0-1 that is a measure of how closely the model fits the data. 1 indicates a perfect fit while 0 represents no relationship between the dependent and independent variable.

The features of the model are evaluated using an t-test[2] which tells us if the coefficients are statistically significant in predicting the target variable. This is done through a hypothesis test where the null hypothesis is that the given coefficient is equal to 0 which means it does not have an impact on predicting the target variable. The alternative is the feature impacts the target variable. The t- statistic is calculated as the predicted features coefficient, β, over the standard error of β (5).

$$t = \frac{\beta}{SE} \quad (5)$$

From the t-statistic, a t-distribution is inferred and a p-value can be found. The p-value is compared to a predetermined significance value which is usually .05. The p-value is the probability of observing a t-stat as extreme as what was calculated from the data under the null hypothesis. If the p-value is less than the significance level, the null hypothesis is rejected and the feature is statistically significant. Otherwise, the feature can be dropped. This is useful because it tells how much of an impact each feature has on the prediction and allows features to be dropped that are proven to not have an impact on the model.

While OLS is very expressive, it is not as powerful a predictive model as others used. OLS relies on a linear relationship between the independent and dependent variables which is rarely true. It also assumes homoscedasticity which means that the errors are constant. The Ecuadorian store sales data does not perfectly meet these assumptions which means that the predictions will not be the most optimal. Finally, OLS is not explicitly built for time series data. The elements of time instead are fed in as day of week, month, and year all as integer values. Other models are built for time series and even contain parameters that help predict seasonality.

B. Seasonal Autoregressive Integrated Moving Average Exogenous (SARIMAX):

SARIMAX[3][4][11] is a time series model that looks at the trends and patterns of data over time. SARIMAX relies on the data being stationary. Stationary means that the statistics describing the time series data do not change over time. These are:

- Constant mean over time, which means that the trend is not increasing or decreasing
- Constant variance over time
- Constant autocorrelation which is how a value correlates to earlier values, lags, in the time series

Most datasets, including the Ecuadorian stores sales do not meet all the conditions to be stationary. This is rectified by making transformations on the data. Fortunately, SARIMAX handles this. SARIMAX is made up of the following pieces which each have their own parameter to tune::

- Autoregressive (AR): looking at the past values to predict the current value with regression. The value of AR, p, decides how many of the past values are used to predict the current. The Autocorrelation and Partial Autocorrelation charts are tools to help choose the value of p. α is the coefficient each lagged value is multiplied by which is found by the linear regression model.

$$y_t = c + \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + ... + \alpha_{t-p} y_{t-p} \quad (6)$$

- Moving Average (MA): Represent the lags in the error, ϵ. This represents a linear regression done on q error terms. This is represented in the equation below

$$y_t = c + \alpha_1 \epsilon_{t-1} + \alpha_2 \epsilon_{t-2} + ... + \alpha_{t-q} \epsilon_{t-p} \quad (7)$$

- Integrated (I): Represents differencing which is subtracting the i previous sales values from the current value. This makes the data stationary. When i =1 the regression equation becomes transformed as shown below.

$$y_t - y_{t-1} = c + \alpha_1 (y_{t-1} - y_{t-2}) + \epsilon \quad (8)$$

- Seasonality (S): Seasonality accounts for patterns with fixed time intervals. It has 4 parameters (P,D,Q,s). P, D,Q correspond to the seasonal autoregressive, moving average, integrated aspects. S the time period the cycle happens over. Seasonality adds 2 more terms equivalent to equations (6) and (7) lagged by s shown in (9) and (10) below.

$$y_t = c + \alpha_1 y_{t-s} + \alpha_2 y_{t-2s} + ... + \alpha_{t-p} y_{t-sp} \quad (9)$$

$$y_t = c + \alpha_1 \epsilon_{t-s} + \alpha_2 \epsilon_{t-2s} + ... + \alpha_{t-p} \epsilon_{t-sp} \quad (10)$$

- Exogenous (X): Exogenous variables are variables other than what we are predicting, y and the date values. For the Ecuadorian stores sales this will be variables like store_nbr and family

C. Prophet FB

Prophet[13] is a forecasting model developed by Facebook's Core Data Science team, designed specifically for time series forecasting tasks. It's directed for use on time series data, and can easily analyze seasonality and regression trends. Prophet is suitable for a wide range of forecasting applications, including business demand forecasting, financial analysis, and trend prediction. We tried prophet because of its simplicity, flexibility, and ability to provide reliable forecasts with minimal effort, making it accessible to both experts and non-experts in time series analysis.

There's three main components in the prophet model as described below:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t. \quad (11)$$

- g(t) is the trend function, that models non-periodic changes in the sales of the forecasting
- s(t) is the periodic changes with respect to sales changes based on weekly and yearly trends
- h(t) is the periodic changes over a day or several days due to a holiday
- e(t) is the one-off changes that are not accommodated by the above three functions

Some of the techniques that didn't prove to perform as well included one Prophet model trained on the entire train set, and fitted with no additional parameters. This served as a good baseline moving forward. Then additional regressors were tested with the one prophet model including the store_nbr, family_cluster, family, and oil_price. All of these variations made the model marginally better. As the developers began to understand the underlying functionality of the Prophet model better, it was decided to train 33 different Prophet models on their own respective product families. This improved the RMSLE, but it still felt like there was improvement to be had. A deeper dive on the significance of store_nbr, in our feature exploration, showed a difference in sales patterns across each. It was then decided to split the modeling into one model per store as well for a total of 33 families x 54 stores Prophet models. The results were then aggregated.

The optimal model for this use case was a set of prophet models that each represented an individual product category (family) and store_nbr combination. A model was trained on individual product categories for a store, taking into account local Ecuadorian holidays, and predicted on the test set corresponding to the product category and store. Results were aggregated from each of the models, and metrics were calculated.

D. eXtreme Gradient Boosting (XGBoost)

XGBoost[12] is known for its efficiency and effectiveness in predictive modeling tasks which made it a suitable candidate for this scenario. XGBoost belongs to the ensemble learning family and is based on the gradient boosting framework. This model works in a sequential manner, adding decision trees to an ensemble, each tree is able to learn to correct the errors made by the previous trees. The final prediction is then made by summing up the predictions from all the trees in the ensemble.

XGBoost uses a collection of decision trees which function as base learners. These trees are shallow, which helps prevent overfitting and improves our ability to generalize to unseen data. Gradient descent is then used to optimize the objective function (12). It computes the gradient of the loss function with respect to the model predictions and adjusts the model parameters in the direction that minimizes loss. These parameters include the tree structure and leaf scores.

Let $\hat{y}_i$ represent the predicted output of the i-th tree in the ensemble for a given input x_i, and y_i represent the actual output (true sales in our case). The objective formula can be represented as:

$$Obj(\theta) = \Sigma_{i=1}^{n} L(y_i, \hat{y}_i) + \Sigma_{i=1}^{k} \Omega(f_k) \quad (12)$$

Where:

- $L(y_i, \hat{y}_i)$ represents the loss function measuring the difference between the true label and the predicted label.
- $\Omega(f_k)$ is the regularization term penalizing the complexity of the model
- θ represents the parameters of the model, including the structure of the decision trees and the leaf scores. These are parameters such as learning rate, number of estimators, max depth of the trees, the number of iterations involved as well as cross validation.

L1 and L2 regularization techniques are implemented to prevent overfitting, penalizing large parameter values and encouraging simpler models that can generalize better, which is what we want. This will help us spread our model to different grocery stores in Latin America. XGBoost also incorporates feature importance, calculating feature importance scores based on how frequently a feature is used in the construction of decision trees and how much each feature contributes to reducing the loss function. This helps us greatly in realizing which features provide the greatest impact on our results. In conclusion, the versatility of this powerful algorithm makes it an ideal candidate for our time series prediction task.

V. Experiments and Results

The data was split in a 70%-10%-20% Train, Validation, Test split. It was split based on an ordered date range. Train goes from 2013-01-01 to

2016-04-26. The validation set goes from 2016-04-26 to 2016-10-03. The test set goes from 2016-10-03 to 2017-08-15. It is important to order the data first before splitting because some models are better at predicting data closer to the known data points. So data further in the future has a less certain prediction. This is real world accurate as we will not always know sales right up until the prediction date and may be forecasting months out into the future.

We chose to use RMSLE as our main accuracy metric for this regression problem (2). One of the reasons for this is that it's considered a scale metric meaning that the units given the metric correspond to sales. This makes it easily interpretable, especially in the context of time series forecasting. We use RMSLE instead of MSE because of the large deviations in predictions.

A. OLS:

OLS was used as a baseline model because it is simple and expressive. The input features were a constant, onpromotion, day_of_week, month, year, oil_price, holiday, family, and storecluster. However, OLS only takes numerical values so the categorical values of holiday, family, and store cluster were one hot encoded. This led to 73 features being input for model fitting. The fit model has an R^2 value of .6. This means 60% of the model's variance is explained by the chosen variables. For a simple model, this is decent.

OLS has a validation RMSE of 789.9 and RMSLE of 2.27. It has a test RMSE of 863.7 and RMSLE of 2.47. Of the models used, this is the lowest score. This makes sense as the time features of the date were modeled as integers of day of week, month, and year. This implies that the sales move in a linear direction throughout the year. Other models are able to account for the time data more robustly. When plotting the actual test values vs prediction (figure 2), the predictions are consistently higher than actuals. The team thinks this is because December always sees a large spike in sales. Because the months are input as integers 1-12 the December spike artificially raises the predictions for the other months.

Figure 2: Testing OLS predictions vs Actual values

B. Prophet FB

The prophet model is limited in hyperparameter tuning, so for this model we decided to split into several individual prophet models, one per product category and store number combination. A function was added to take into account local Ecuadorian holidays. Prophet allows for dataset columns to be added as individual regressors, so we tested the following columns: store_nbr, family_cluster, and oil_price. All of these had no benefit to the final RMSLE as a regressor, and this was a driver for separating into several models dependent on store_nbr as it should have a positive impact on the metrics.

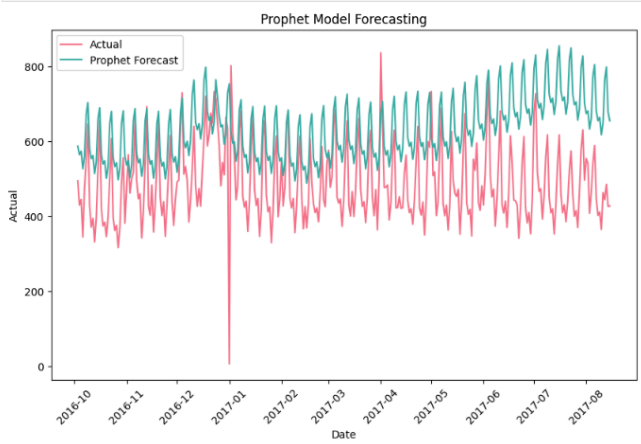

Figure 3: Testing Prophet predictions vs Actual values

The prophet model proved to perform best on the test set, given the RMSLE value of 0.729. In an effort to minimize overfitting, the model made predictions for the validation set as well, and had an RMSLE of .511. This number shows the flexibility that this model offers. The Prophet forecast was always making sales predictions on the higher side than the actual (Figure 3), and may indicate the effect that higher values had on the predictions. The team believes that with more developer notes on how FBProphet works behind the scenes, a model could have better fitted the large swings that the model seemed to have.

C. XGBoost

In order to select the best hyperparameters for XGBoost, we utilized the validation set (10% of original data). This helped prevent overfitting, show a more accurate evaluation of unseen data, and take advantage of cross-validation. Through this tuning, we found that a learning rate of 0.1 was best for our data. Learning rate is essentially the contribution of each tree to the final prediction, 0.1 represents a good balance of speed and accuracy while avoiding overfitting. For the number of estimators, or number of trees in the ensemble, 200 was selected. This was a middle ground of the 50, 100, 200, and 400 estimators we tested. This tempers the model's complexity without too high of a computational cost. We controlled the depth of each tree using max_depth. We tried many different depth levels but found that going down 8 levels worked best to avoid underfitting and strike a balance between bias and variance.

We wanted to make sure to implement cross validation in our hyperparameter tuning. Our validation set represents 10% of the data, so we chose 10 fold cross-validation to make sure to ensure robustness in estimating the model's performance. 10 fold also provides a good balance between bias and variance as well as prioritizing accuracy without overloading computational efficiency.

	Before Hyperparameter Tuning	**After Hyperparameter Tuning**
RMSLE	1.45	1.276

Table 2: Evaluation metrics for XGBoost with and without hyperparameter tuning.

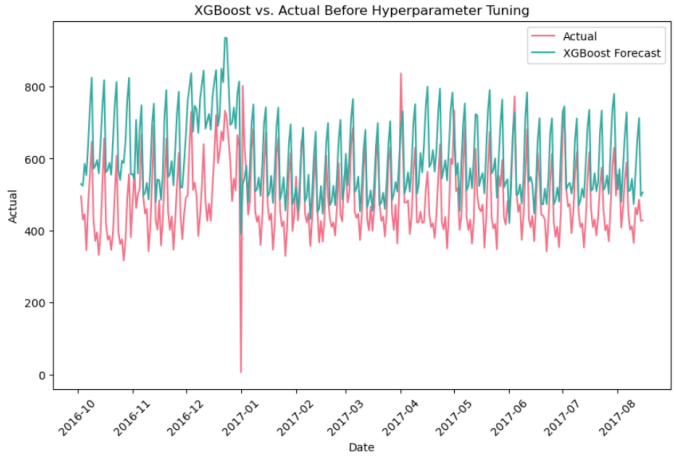

Figure 4: XGBoost vs. Test before Hypertuning

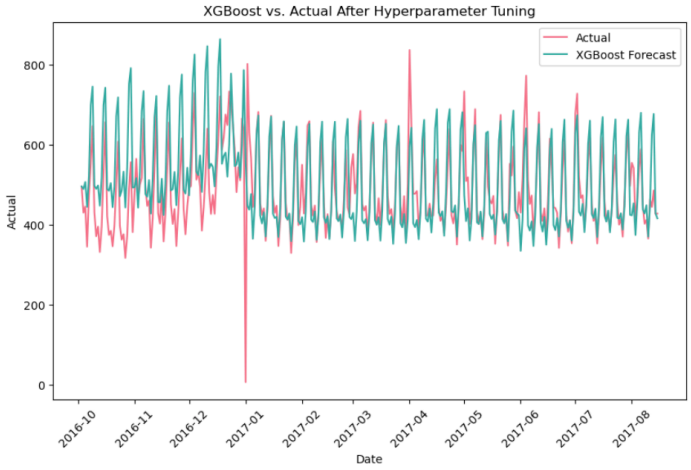

Figure 5: XGBoost vs. Test after Hypertuning

As seen in the tables and charts above, tuning the hyperparameters was very impactful in increasing the accuracy of our results. It decreased the RMSLE, our key metric, by 16%, more accurately representing the actual sales of the grocery stores. It is clear from the charts that XGBoost is consistently over predicting sales by just a bit. This is likely because of imbalance in the dataset, perhaps some outliers, and shows room for growth with feature engineering.

D. SARIMAX:

The team attempted to train SARIMAX using only the date, store_nbr, family, and sales features. This is the bare minimum number of features that can be used and still have every row from the train set be unique. However, the model not was not able to produce results. It demanded too much resources and either produced a memory error or would run for hours without ending. This was true even when the date range in the train set was greatly reduced. Instead of training on this set, the team trained on the summed daily sales, ignoring the other features. This output daily sales predictions as a sum across all stores and families. Therefore, the results can not be compared to the other models' results. This is a big area of improvement for the future.

To get an estimate for the q parameter, which is the moving average term, an autocorrelations function, ACF plot[5] was created (figure 6). The graph shows the correlation between the current value in the time series data and its previous, lagged values. Values outside of the shaded region are thought to have a significant impact on the current value while the values of lags inside the shaded region can not be distinguished from noise. This shows which lagged values have a big impact on the current value and can show the effects of seasonality. From this, a value of 7 was chosen for s as we can see weekly cyclical autocorrelation. For now 1 was chosen for q, the moving average parameter due to the large spike at 1 in the ACF chart. This can be further tuned but because the model is not usable this will be left for another project.

The partial autocorrelation function, PACF, plot (figure 6) shows the partial autocorrelation between a time series and its lagged values controlling for the effects of the other lagged values.This can be used to estimate p which is the autoregressive hyperparameter. From this chart p was assumed to be 1 due to the large spike at the 1 lag value.

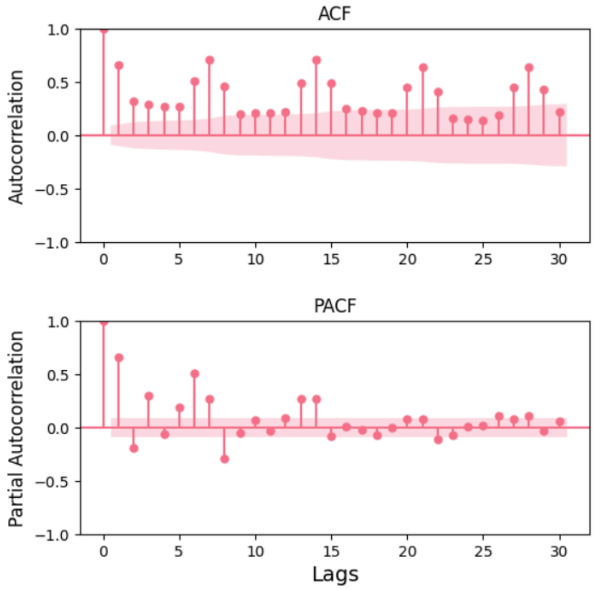

Figure 6 : ACF and PACF plots

This produced a fairly strong RMSE score of 208,252 and RMSLE of .354 but again these cannot be compared to the other models as it is not predicting the same sales values. The chart of actual sales vs predicted is shown below, figure 7. The model misses at the big dips and spikes as those relate to holidays. This is information not being fed into the model.

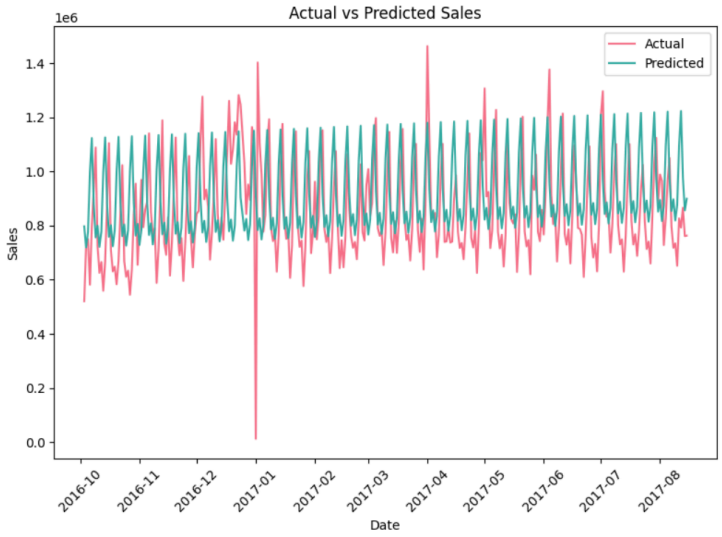

Figure 7: SARIMAX Actual vs Predicted Sales

VI. Conclusion

In summary, the exploratory data analysis performed, feature

engineering administered, modeling put in place, hyperparameter tuning efforts, and visualization of results all have led to a very successful project. The RMSLE can be seen for each below, and no doubt, the result of this project will be able to positively impact Favorita Grocery Stores by allowing them to more accurately forecast how much product they should purchase in the future. FB Prophet ended up providing the most accurate results using models trained to the specific stores and product categories, and offered a highly tuned aggregated set of models. Models that were able to capture the seasonality and were inherently for time-series forecasting seemed to perform best with the dataset, as expected.

Our goals of exploring various models and coming up with an accurate regression forecast predictor that would fit the Ecuadorian grocery store were completed. Specific to the models, we hoped for better results from the Sarima and in the future would like to explore Random Forest models. On the other hand, XGBoost and Prophet modeling produced successful results. A correlation we had noticed with heatmaps was the relationship different product families had. K-means was used here to cluster the families, but this feature only harmed our accuracy. This may be a worthwhile tradeoff depending on the use case, but we decided to leave it out because of the goal of a more accurate model. We were able to make progress on tuning these models with multiple developers on the project and a clear understanding of the dataset and features after our exploration.

Model	**RMSLE**
OLS	2.27
SARIMAX	NA
XGBoost	1.26
FB Prophet	0.729

Table 3: Final results for all models.

Contributions

Over the course of this group project, we did a good job working together, communicating effectively, and evenly dividing and conquering the tasks. We meet weekly to discuss updates, brief each other on progress, and formulate a plan for moving forward. At the very beginning of the project we collaborated to select an appropriate dataset and define the problem we were looking to solve. During the data exploration phase, each of us took a different set of features and explored how they could impact future models, moving into feature engineering where we made appropriate tweaks or manipulations to the features we were focusing on.

From there, Cory focused on developing models for SARIMAX and OLS, Will for XGBoost, and Alec for FBProphet. Each of us did deep dives into our respective models, learning the inner workings of their algorithms so as to optimize them for our problem. We trained, evaluated, and deployed the models and kept each other in the loop about how they were performing and asking for insight as to how to improve our shortcomings. We have continued to monitor and maintain our models, modifying as we see fit and continually collaborating on any issue that may arise.

As we approached different project milestones, implemented documentation, and gathered feedback from our peers, TA, professor, and other outside sources, we maintained an open line of communication to make our project the best it could possibly be. This has been an incredible team that worked mostly seamlessly to achieve an outstanding result. We are thankful for this opportunity to work together and look forward to being data science collaborators for years to come.

Reflections

SARIMAX was not a successful model. It took far too long to train to the point where the team never got a trained model. Part of the reason it was so intensive was that the features being fed in as input needed to be numeric. So the family cluster and store cluster were represented with dummy variables which greatly increased the total number of features being fed into the model. SARIMAX would do better if it is predicting sales off only a date. But the focus of the project was to predict based on at least the date, store number and product family. Additionally, the team's model was trained on only one set of parameters for SARIMAX. There are 7 hyperparamters to tune. Autoarima is a gridsearch method for ARIMA which tests a set of hyperparameters that the user chooses. The team did not get to this as even training on one set of parameters was not feasible. If given more time and resources, the team would have paid for space on a virtual machine to run this model. SARIMAX should have been competitive with FBProphet for the best performing model.

XGBoost performed very well but if we were doing this again we would pursue other boosting and gradient descent based models. Models such as LightGBM, CatBoost, and Gradient Boosted Decision Trees could have potentially offered better performance than the models we used. Additionally, in much of our research Long Short-Term Memory (LSTM) was a successful model on similar data, this could have helped us with increased accuracy as well.

The Prophet model performed best on the validation and test set. If it was possible to get a training on some of the finer grain details on how to use the Prophet library effectively, that would be helpful in order to tune this model even more. It served as a great model, and could have been improved upon even more. With more time it also would have been worth trying other pre-build time-series forecasting models, as they all have their unique spin on what datasets they perform well on and not.

Citations:

1. Seabold, S., & Perktold, J. (2010). statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*. Available at: https://www.statsmodels.org/stable/regression.html (Accessed: 07 March 2024).
2. A. Hayes, “T-Test: What It Is With Multiple Formulas and When To Use Them,” Investopedia, Jul. 20, 2022. https://www.investopedia.com/terms/t/t-test.asp
3. Matt Sosna (2021) A Deep Dive on ARIMA Models https://towardsdatascience.com/a-deep-dive-on-arima-models-8900c199ccf
4. Bajaj, A. (2021, May 11). ARIMA & SARIMA: Real-World Time Series Forecasting. Neptune.ai. https://neptune.ai/blog/arima-sarima-real-world-time-series-forecasting-guide
5. LEONIE, “Time Series: Interpreting ACF and PACF,” kaggle.com. https://www.kaggle.com/code/jamleonie/time-series-interpreting-acf-and-pacf
6. Recycle Track Systems. "Food Waste in America in 2021: Statistics & Facts." Accessed March 12, 2024. https://www.rts.com/resources/guides/food-waste-america/#:~:text=Grocery%20Store%20Food%20Waste.
7. Shi, Weijing. "Food Waste Prediction in Grocery Stores: Time Series Forecasting by Deep Learning." 2022.
8. Falatouri, Taha, et al. "Predictive Analytics for Demand Forecasting–A Comparison of SARIMA and LSTM in Retail SCM." Procedia Computer Science 200 (2022): 993-1003.
9. Turgut, Yakup, and Mustafa Erdem. "Forecasting of Retail Produce Sales Based on XGBoost Algorithm." In Industrial Engineering in the Internet-of-Things World: Selected Papers from the Virtual Global Joint Conference on Industrial Engineering and Its Application Areas, GJCIE 2020, August 14–15, 2020. Springer International Publishing, 2022.
10. Jha, Bineet Kumar, and Shilpa Pande. "Time Series Forecasting Model for Supermarket Sales Using FB-Prophet." In 2021 5th International Conference on Computing Methodologies and Communication (ICCMC). IEEE, 2021.
11. Statsmodels Development Team. "Statsmodels: Econometric and Statistical Modeling with Python." 2022. https://www.statsmodels.org/stable/index.html.
12. XGBoost Development Team. "XGBoost Documentation." Last modified 2022. https://xgboost.readthedocs.io/en/latest/index.html.
13. Taylor SJ, Letham B. 2017. Forecasting at scale. PeerJ Preprints 5:e3190v2 https://doi.org/10.7287/peerj.preprints.3190v2
14. "Retail Demand Forecasting in Grocery." Algonomy. Accessed March 14, 2024. https://algonomy.com/resources/guides/retail-demand-forecasting-in-grocery/.
15. Luis, CJ. "Data Science Forecasting Food Retailer." GitHub, https://github.com/luis-cj/data-science-forecasting-food-retailer.
16. Justin, Jabo. "Sales Analysis and Forecasting of the Grocery Stores Using the Gradio-Streamlit Application." Medium. Accessed March 14, 2024. https://medium.com/@jabojustin250/sales-analysis-and-forecasting-of-the-grocery-stores-using-the-gradio-streamlit-application-and-34cb1b1fcd0a.
17. Odegua, Rising. "Applied Machine Learning for Supermarket Sales Prediction." ResearchGate. Accessed March 14, 2024. https://www.researchgate.net/publication/338681895_Applied_Machine_Learning_for_Supermarket_Sales_Prediction.
18. Saci, Samir. "Machine Learning for Retail Demand Forecasting" Accessed March 14, 2024. https://towardsdatascience.com/machine-learning-for-store-demand-forecasting-and-inventory-optimization-part-1-xgboost-vs-9952d8303b48.

Reply to Review

Both the TA and our peers commented on the correlation heat map for the sales of families. During data exploration, the team found that the sales between product families were highly correlated. However, this information was not used in the original models. After the peer reviews, there were efforts in clustering the families together to reduce the number of dimensions. This could help with overfitting. Different numbers of clusters were tested using TimeSeriesKMeans. The inertia per cluster as well as the Silhouette score per cluster were charted (figure 8). The optimal cluster is supposed to be where the inertia elbows and where there is a plateau or high point in the Silhouette scores. Clusters of size 6, 14, 21 were tested on the OLS baseline model. The larger cluster sizes produced lower RMSLE scores but none were better than the families being input with no clustering. We think valuable predictive information is lost when clustering families together and that due to the sample size of a little under 3 million rows that the models are not prone to overfitting.

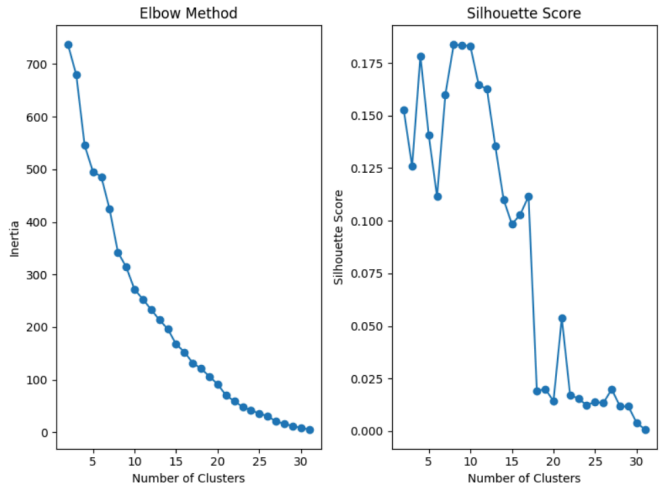

Figure 8: Inertia and Silhouette Scores

Additionally, we received a critique that stated we should more clearly explain what makes a metric "good". This is feedback we found helpful and have addressed at length in our final report. Root mean squared logarithmic error was the ideal metric to measure our accuracy given the scale of our data, this is not something we fully elaborated on in the presentation so we were glad to get the chance to do that here.

Another piece of helpful feedback we were given by our peers was that we didn't do the best job of comparing our models to one another. It was clear that we showed and explained each model individually but could have explained how they compared to other models more explicitly. This was a blind spot for us and one that was clear to see once pointed out by our peers. We responded to this in our final paper by clearly comparing the results and explaining why some models performed better than others. One of these charts is shown here, clearly identifying the success metric of each model.

Model	**RMSLE**
OLS	2.27
SARIMAX	NA
XGBoost	1.26
FB Prophet	0.729

Table 4: Final results for all models.

Furthermore, we were informed that we could use a few more charts for our predictions, meaning we weren't showcasing our results quite enough. This is something we really wanted to focus on for our final report. Thus, instead of spending more time developing different or novel models, we really honed in on the current models we had and worked to display the results they were achieving. By showing where the models are excelling and where they are falling short, we not only understand them more deeply ourselves, but can visualize model performance more effectively for the viewer. We went with a unified and consistent charting format for our predictions charts, as shown below.

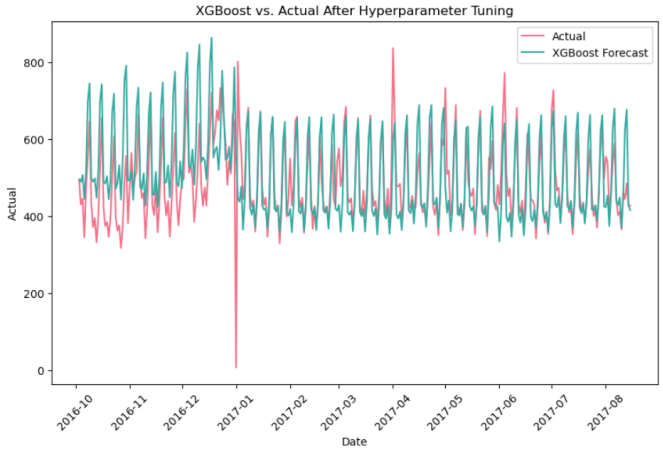

Figure 9: How XGBoost was affected by hyperparameter tuning

Finally, setting the stage for our problem statement and explaining our approach was one place we fell short in earlier iterations of the project. Our wonderful TA, Zheng Zhou gave very specific and helpful feedback to define the problem more clearly. By including the input/output of our model and expanding more about the regression we hope to achieve through supervised learning, our reader has a much clearer sense of what we are working towards in the report. This is feedback we found actionable and are grateful to have received.