Corporacion Favorita

Modeling Forcast of Ecuadorian Grocery Sales

Matthew Barrett
Clarissa Franklin
Timothy Lai
Brett Scroggins
Meyappan Subbaiah

Advanced Predictive Models - Dr. Joydeep Ghosh

Abstract

Accurately forecasting product sales has always been a crucial component of profitability for the retail industry. An inaccurate forecast can result in missed sales opportunities, unhappy customers, and supply chain inefficiency. This is a problem that grocery store chains face daily, as many products have limited shelf lives and, if overstocked, will result in waste. Corporacion Favorita, a large Ecuadorian grocery chain, has recently prioritized this issue and has challenged the Kaggle community to build an accurate forecast that predicts product sales which will ultimately help to optimize product inventory and reduce loss. As this is an active Kaggle competition, the goal of this project is to not only build an accurate predictive forecast, but also to submit findings to Kaggle to be measured by their error metric.

To build this forecast, several tree-based predictive methods were implemented as well as more robust techniques, such as the neural network and a tree-based ensemble. The initial approach involved pre-processing the data to group items into families so that similar items could be grouped together for a more accurate prediction. While these initial tree-based methods ultimately proved moderately effective, a modified approach was later taken in order to take a different perspective to building predictive models. This involved generating predictions at a more granular, item-based level to make better use of all contributing features of the data. This modified approach ultimately produced more effective results that drastically lowered the calculated Kaggle error metric by which the forecasts are judged. With a low error and consistently accurate forecasts, this problem could be addressed and Corporacion Favorita should look into implementing similar models in order to improve their sales forecasting in order to ultimately maximize profits.

Contents

List of Figures

1.1 Introduction

Accurately forecasting product sales has always been a crucial component of profitability for the retail industry. If the forecast is low, the on-hand quantity of a product may deplete resulting in missed sales opportunities and unhappy customers. If the forecast is high, overstocking can lead to supply chain inefficiency and lost opportunity cost of shelf space for other products.

This phenomenon is further amplified for grocery stores, due to the high proportion of their products that have limited shelf lives. If the sales forecast is overshot, many perishable products could spoil before being sold. This results in a cost for the grocery store due to purchasing, shipping, handling, and shelving the items with no possibility for any return on their investment. The negative impact on profit margins can grow to serious problems if poor forecasting remains a continuous problem. These issues are further complicated if the grocery store includes multiple locations, especially if these include a variety of local spending habits.

Some economies have further uncertainty that can result in further unpredictability of future grocery sales. Ecuador has an economy that is largely driven by oil exportation; and as the international price of oil fluctuates this can significantly affect the spending habits of Ecuadors citizens. Building on this volatility, the country is an emerging economy; which compounds the issue of accurately forecasting product demand.

Considering this complex matter, the Ecuadorian grocery chain Corporacion Favorita has challenged the predictive modeling community to build an improved model for use in predicting sales forecasts. Prior modeling methods have been limited and based on intuition as opposed to data. By opening a Kaggle competition, they have enabled an unlimited number of analytics teams to attempt creating the best possible model for predicting the sales forecast.

Each of the models generated by the competing teams will be judged by their normalized weighted root mean squared logarithmic error (NWRMSLE). Normalizing the metric allows for a more standardized way of comparing each model, and using the log values serves to a similar purpose. The purpose of the weighting on products was to account for the differences in the shelf life of each product. This further aligns the measuring metric of the Kaggle competition with the business value Corporacion Favorita needs.

1.2 Data Description

The data was provided in the form of seven different csv files: items data, stores data, transactions, holiday events, oil prices, a training set, and testing set. The data provided is a time series and covers all retail transactions from 2013 to 2017. This amounts to approximately 125 million rows by the number of descriptive columns, corresponding to a size of 4.88 GB for the available data.

Furthermore, a majority of the descriptive data corresponding to each transaction included a large portion of categorical variables. This was a noteworthy feature of the data, as implementing models required factorizing the data, resulting in sparse matrices requiring significantly increased storage capacity. Each of Corporacion Favoritas multiple data sets described above provided necessary information to help more accurately forecast product sales. The datasets were mapped together using joins on primary keys as necessary from each dataset. The schematic description of how all the data connections is visualized below:

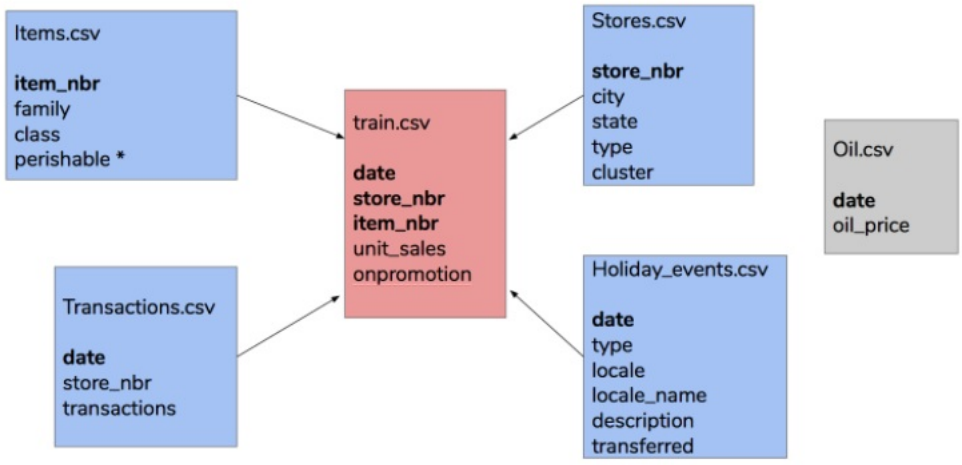

Figure 1.1: Relational diagram for the datasets provided

Each portion of the data provided was studied individually before further pre-processing to fully understand and relate the data structures. Some of the more notable features already provided in the datasets were as follows:

- All **dates** provided were in an interpretable programming format - more specifically, the day, month, and year were extracted. This would provide the capability to roll up the data into various time splits to potentially identify trends not seen at first glance.
- The **store number** and **item number** are unique identifiers for each store and item. Individual stores by location can be identified and sales could be grouped by regions

and thereby considered in the later predictions. For items, this would help identify each of the 4,096 items that are of interest in this competition.

- **Unit sales** were immediately slightly more difficult to differentiate quantity-wise for each item, as a value of 1 in the data set could represent different quantities for an individual product (i.e. this could represent one bag of chips, or one kilogram of cheese). To circumvent this issue, items were then classified as part of a similar family of items (i.e. beverages, deli items, etc.) to their respective groups. Lastly, items that were returned would also have to be accounted for in the model building process, as negative sales must be misinterpreted in this context.

- The **on_promotion** value identified whether the item was sold using a discount or sale. This could potentially be valuable information to identify whether promotions have an effect over time on the sales of any particular products.

- Holiday data was provided at a local, regional, state and national level. The type of holiday indicates whether it was transferred or not. A transferred holiday falls on a specific calendar date but was moved by the Ecuadorian government, which seems to be a common occurrence. For simplicity, **national holidays** were the only holidays considered in searching for exploratory trends.

- Oil information was also provided, as Ecuador is a member of the Organization of Petroleum Exporting Countries (OPEC). Since the Ecuadorian economy is historically quite dependent on the **price of oil**, effects on unit sales could potentially be correlated in both the training and test data sets.

- The **perishable** indicator was a key variable that needed to be strongly considered in building predictive models. Since perishable items were weighted more heavily in the NWRMSLE, these item sales would need to be predicted more accurately which would result in a decreased error. This effectively represents Corporacion Favoritas concern of decreased profits and food waste if perishable items are overstocked and not efficiently sold.

Outside of the features that were inherently available with the data, the cluster variable of the data set was a unique feature that was engineered by Corporacion Favorita. This variable would be indicative of how Corporacion Favorita felt their items should be grouped and could perhaps provide some insight on how products are stocked on shelves. With the above understanding and insights regarding the data, pre-processing and Exploratory Data Analysis were then performed.

1.3 Data Pre-Processing

Due to the datas large size and complexity, a combination of Spark (via Databricks) computing and Amazons S3 data storage services were utilized for pre-processing. As an additional tool, PySpark provided the ability to manipulate the data efficiently with its built-in libraries. This combination of available resources enabled more efficiently data pre-processing on the needed scale with parallel computing.

With these capabilities applied, the initial step taken was to first create a flat file that incorporated all related variables with the training data, excluding the oil price. A few of the binary columns were converted into proper Boolean types, for example holiday and on promotion for the purpose of manipulation. The data was also explored to potentially find ways to reduce dimensionality. Once investigated, it was determined that some columns either added no additional explanatory information when compared to similar features or simply were poorly engineered by the providing company. For instance, the city and state of a store location were redundant, since the store number and type both proved more valuable features capable of providing the same details.

Once the flat file was created, the training data remained intact with the initial 125 million rows; however the increased number of columns proved difficult to leverage even while using Sparks parallel computing capabilities. To remedy this, the flat file was randomly sampled down to 10% with a set seed to guarantee reproducibility. This new sample data set was comprised of roughly 12 million rows, which not only was able to capture majority of the datas necessary information for forecasting purposes but also allowed for the development of predictive models.

As a result, this data structure and component system was used for purposes of preparing effective models in trying to minimize the NWRMSLE in forecasting the grocery unit sales for Corporacion Favorita. A full data exploration was then completed in order to identify contributing factors to grocery sales and identify trends that would potentially assist in more effectively building models.

1.4 Explaratory Data Analysis

The data provided dates ranging from 2013 to 2017, during which time Corporacion Favorita grew from 46 to 54 stores. The below figure depicts the location of all stores in Ecuador and the corresponding size. In figure 1.2, presented below, the size of each store was represented by the average number of transactions.

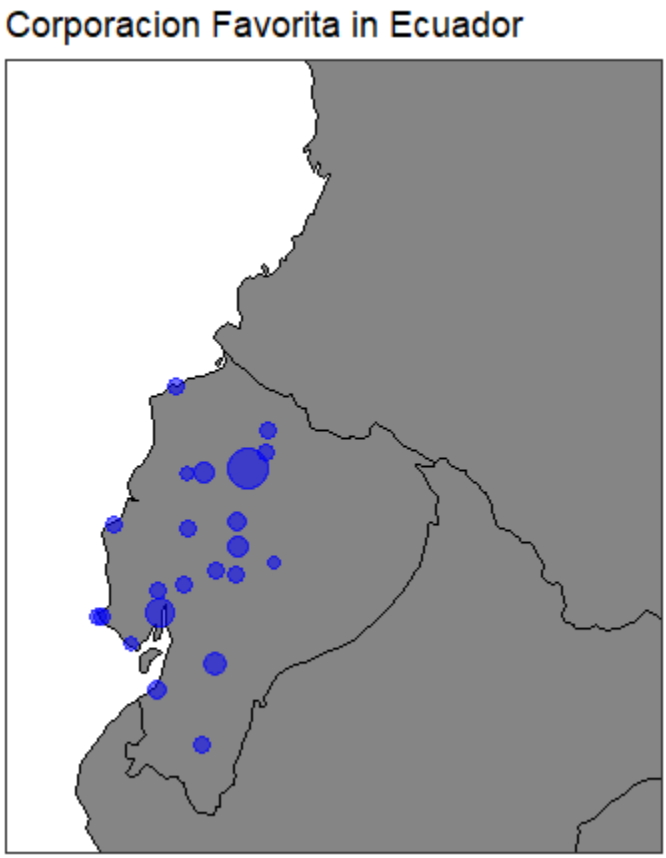

Figure 1.2: Location and Density of Corporacion Favorita Stores in Ecuador

Next, in order to verify the Ecuadorian economys dependence on oil, the cost of oil shares were compared to the average number of transactions to identify if there were any distinguishing correlations. From the stacked plots below presented in figure 1.3, it is evident that Corporacion Favoritas number of transactions over the course of 2013-2017 is not affected by slight fluctuations in oil prices, but is instead reactive to significant drops in the market. As an example, this is evident at the beginning of 2015 when the number of transactions sharply declined after a fall in the oil market over the last six months of 2014.

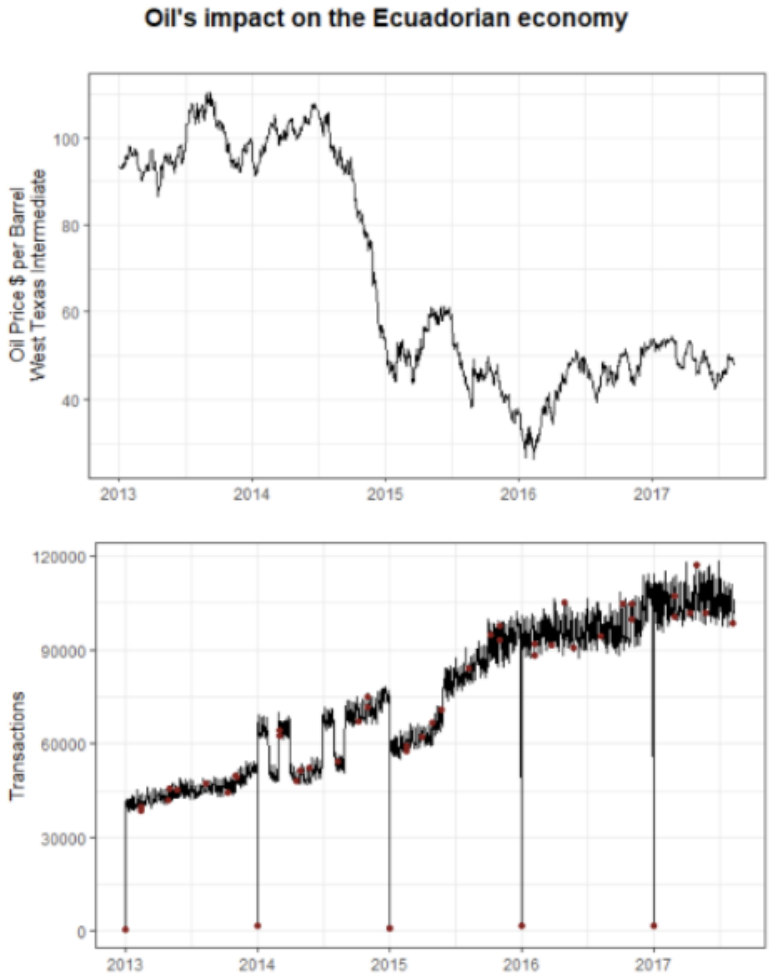

Figure 1.3: Oil price per barrel (top), Number of transactions in Corporacion Favorita stores (bottom).

Outside of the large market drop with a slight lagging effect into the start of 2015, there seem to be no significant events outside of a very few micro-trends. While this possible lack of trend can be manually observed in comparing the above plots, verification of this hypothesis is necessary. To do this, a regression was run to determine correlation and impact of the price of oil, and whether it should be included in forecasting models. The results confirmed our hypothesis as the regression showed in an insignificant correlation between sales and oil; thus moving forward, oil prices were not kept as a valuable feature for the modeling process.

Following this, the companys growth profile was created to see store development over time. What can be immediately seen is that the number of stores and average product variety of stock both grew between 2013 and 2017. This can be seen in the below graphs depicted in figure 1.4 showing the change in number of each over time. This suggests that

the company is both growing and becoming more diverse with its inventory. Both factors offer certain challenges that must be addressed; for example, one particular store location may offer certain products that are not seen at another location. Likewise, new products and new stores are introduced over time - therefore historical sales data for similar products and stores would need to be mapped to these new variables. This was a crucial aspect to generating and maintaining an accurate prediction that would reduce error.

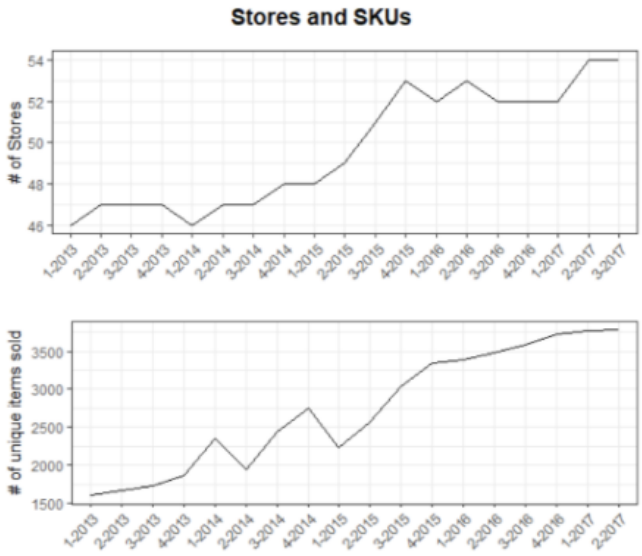

Figure 1.4: Total stores in operation (top) and total transactions (bottom) from 2013 to 2017.

To help remedy the concern of predicting sales for newer stores, Corporacion Favorita provided two different store metrics; store type and store cluster. Store type was generated as a simple grouping of a small number of stores which combined similar stores in terms of sales and location. Meanwhile, store cluster was an engineered feature which attempted to group similar types of stores together. Since the methodology behind Corporacion Favoritas feature engineering was not provided, we attempted to validate the importance of this variable.

The plot shown in figure 1.5 shows the frequency of log(transactions) by store cluster. With log(transactions) being shown on the plot, the range of transactions would be scaled for more effective visualization while also showing the differences in each cluster. From the plot, it is evident that Corporacion Favorita did not effectively engineer this feature. Specifically, clusters 2, 6, 7, 11, and 13 show bimodal distributions within each cluster - suggesting that the transactions do not correlate with each other effectively. Furthermore, both cluster 1 and 10 have outliers and a small sample of values which fall below the average for these clusters. For these reasons, this feature will not be used in our analysis and the focus of grouping stores was shifted to store type.

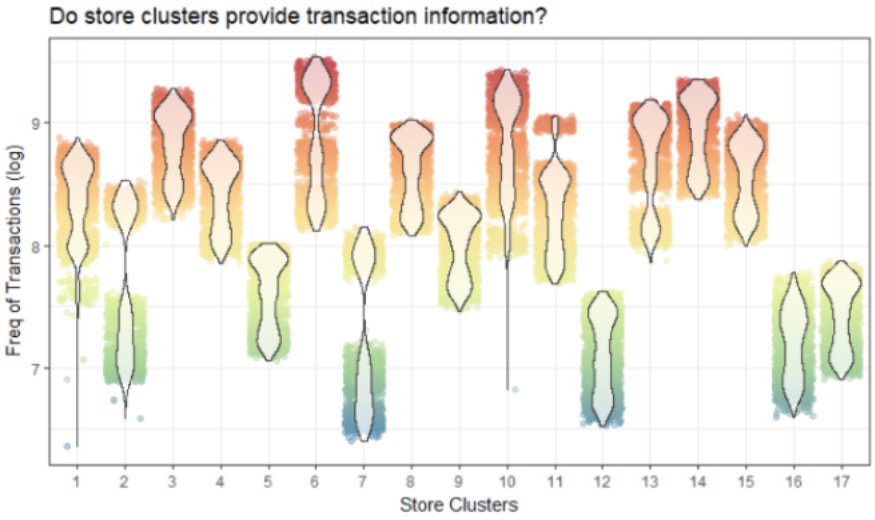

Figure 1.5: Log frequency of transactions by store cluster.

Next, transactions were organized by group and by item in order to visualize the proportion of sales that originated from each type of product sold. Figure 1.6 below details the number of transactions for different families of the items/units provided in the training dataset. The size of each box represents the number of transactions in which this family was involved with once the data was scaled to the log scale as done previously. The separations shown by the white lines represent a more granular approach that identifies how often an item within a family is involved in a transaction. Ultimately, the plot suggests that the grouping of items accurately reflects the types of product, and how each of these types relates to the proportion of sales.

Figure 1.6: Relative number of transactions for each product type.

Lastly, to better forecast unit sales, it is important to identify possible trends and seasonality effects in the data. As unit sales for different items are on different scales, the average number of transactions were used as a proxy to determine demand. Figure 1.7 below takes a more detailed look at day-to-day sales, and shows the average number of transactions through the days week for a given month. As seen by the red blocks, there is a higher demand during the holiday season in December as well as peaks during school holidays, in July. Therefore, a slight seasonality trend is observed with higher sales towards the end of the year as compared to the beginning of the year.

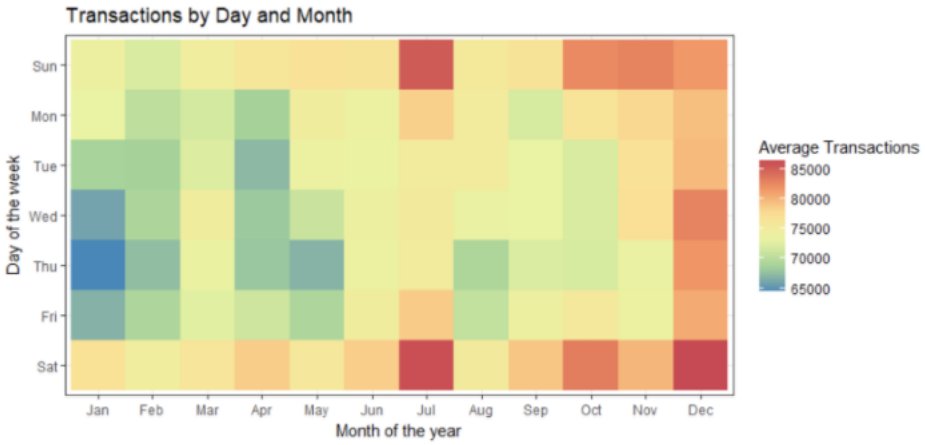

Figure 1.7: Average daily transactions by month and day of week.

With the above trends identified as part of the full EDA, feature selection and engineering was the next logical process towards building predictive models. Features would be considered based on the information learned as part of EDA that would more effectively explain the data in a concise manner.

1.5 Feature Selection and Dimension Reduction

Since the resulting aggregate dataset comprised of thousands of dimensions with millions of rows, dimensionality reduction was a key area of focus as part of feature selection and engineering. This firmly prompted for attempts at systematically finding ways to reduce dimensions less useful to the accuracy of the model. The first approach taken involved implementing a LASSO regression in R.

Since the penalizing system for coefficients in LASSO regression forces weak variable coefficients to zero, it is a great first attempt at feature selection. By running the model iteratively using slowly increasing values of the error penalty λ, the elimination of dimensions

can be monitored through the process. Additionally, upon completion of the final iteration, we can identify which value of λ resulted in the minimum error of the model.

When the LASSO model was constructed and run, the best value for lambda identified was zero. As seen in figure 1.8, this suggests that the best regression model was actually one with no error penalty term; which is equivalent to a standard linear regression. Due to the nature that the data was in the form of a large and incredibly sparse matrix, this output was reasonable to consider as accurate.

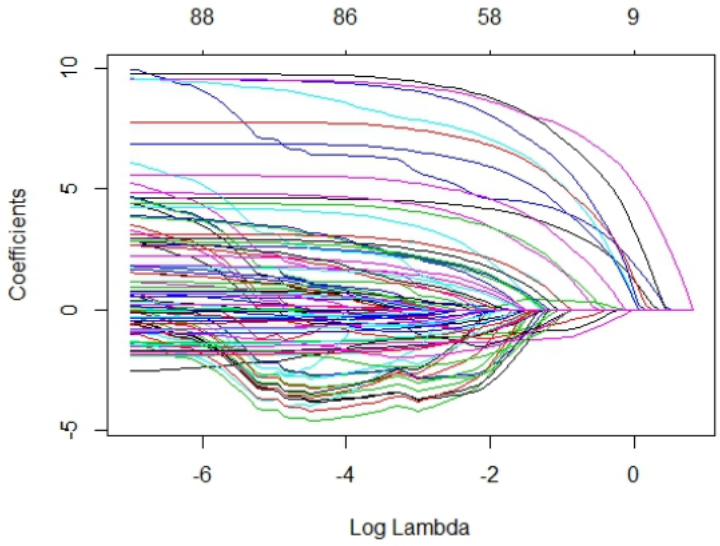

Figure 1.8: Lasso results for feature reduction.

From here, the next step in attempting to reduce dimensionality was to perform Principal Component Analysis (PCA). PCA was considered due to the understanding that it is based upon modeling the dimensions into factors in a way to minimize the number of factors required while maximizing the amount of variance explained. A Bayesian approach to PCA[1] was adopted as the basis of this model creation. The open source package PCAMethods was used in R to complete the PCA for this project. The function used from this package (bpca) allowed for an iterative PCA that modified inputs based upon the results of the previous iteration, and facilitated the continuance of this process until the error difference between iterations reached some predetermined threshold.

Despite the capabilities of the Bayesian PCA listed above, the model was unable to run on the dataset due to the gigantic requirement for memory to complete (256GB). This was due to the thousands of dimensions added by categorical variables from date, item number, and store number. The data had to be rolled up from a daily category to quarterly, store numbers were simplified to general areas, and items had to be reduced simply to types of

products, with an included variable to account for items that are perishable.

The PCA model was run until the error difference between iterations was below 10^{-4}. PCA was performed using both 2 dimensions and 15 dimensions, and the amount of variance explained was analyzed. Even with 15 dimensions of PCA, only around half of the variance was explained. Figure 8 below shows this cumulative variance of the first 15 principal components, which only accounts for about 50% of the variance.

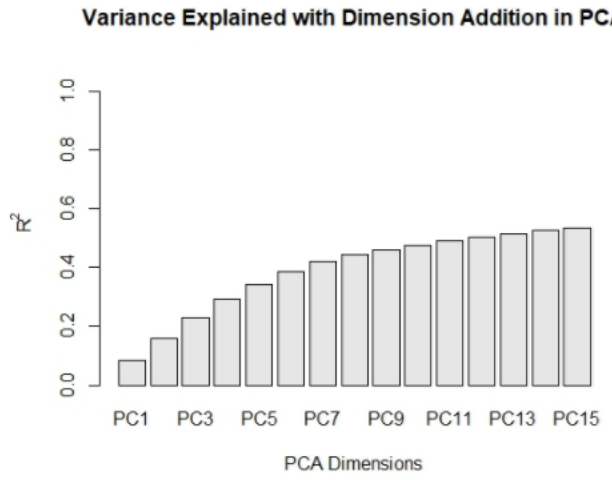

Figure 1.9: Cumulative variance explained as PCA dimensions increased.

Based upon the results of both LASSO and PCA, it was determined the best course of action would be to continue with model development without attempting a reduction of variables, as the sparse data did not lend well to the methods used to reduce dimensions.

1.6 Model Fitting

1.6.1 Random Forest Modeling Method

As the transition into model development was began, the decision to attempt using tree-based methods was the logical starting point due to their flexible and robust nature during a prediction process. Among the available tree modeling options, random forests appeared to be the best starting point since it is a simple approach, uses multiple trees, and then compiles to find the best prediction for our products at the store level. Surprisingly, this proved to be the most impressive model based on NWRMSLE that was run during the initial approach to model grocery sales forecasts.

Based upon the dataset of this project, computing power was the most limiting resource and biggest challenge faced while developing the random forest since each node in the tree

makes a decision over the dataset. This is then used to generate predictions, which would be quite intensive (10 million rows by all descriptive columns). The time to make a decision at each of these nodes was significant because of the large number of variables that the model would have to iterate through. In order to better leverage resources, parallelized computing through the H2O package[**?**] was applied to best utilize the maximum personal computing power available. Even with this implemented, the model fitting process was still quite arduous as running a full iteration of a random forest took at least one hour each, and frequently much longer.

In regards to parameter tuning, a grid search was implemented upon choosing variables of importance in order to specify the ideal parameter values. **Mtries**, or the sub-sample of variables that are included prior to splitting into a branch, was initialized to be either 6, 9, 12. Additionally, **n_trees**, or the number of trees to be build for each random forest, was specified to be any of the following values: 50, 100, 150, 200.

Finally, the **max_depth** was set to 10, which seemed to be a reasonable set point to prune the forest. In analyzing the results of the output using these parameters, mtries appeared to be the more important variable, while the number of trees specified proved a much lesser factor. From the grid search, it was determined that a specification of **mtries = 12, n_trees = 50, and max_depth = 10** would be the optimal parameters used to fit the data, while also not overfitting to the training data and maximizing computational efficiency.

Having identified the desired parameters for this simple tree model, the model was able to return the initial best scoring model of the approaches to be taken. This ultimately produced a NWRMSLE of 0.728. While this was better than the naive approach that Kaggle provided (0.911), it was still worse in comparison to the additional provided metric from Kaggle which simply took the mean item sales forecast for the year and used that as the prediction (0.726). From this, we believed that the approach taken may have some aspects previously unaccounted for. A variable importance plot or the optimal random forest, shown below in figure 1.10, was produced to better understand what was driving the sales predictions.

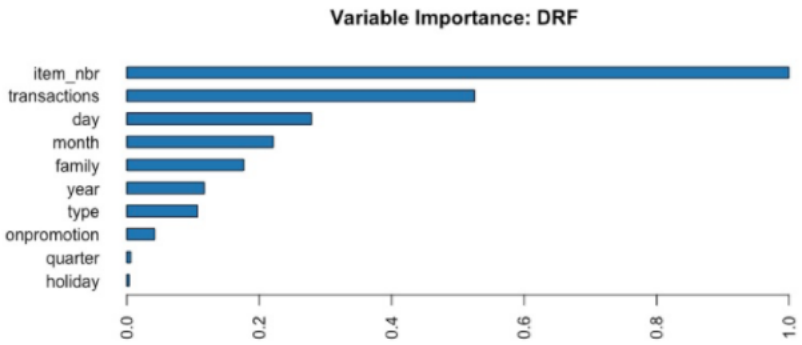

Figure 1.10: Observed variable importance from random forest model developed.

Immediately, it is evident that the item number returned as the most important variable. However, as sales forecasting should be based on the other aforementioned factors in Section 1.2, the variable importance plot identified a flaw in this approach. Moving forward, more complex types of models were applied in an attempt to alleviate this concern and create a more robust and accurate forecast.

1.6.2 Extreme Gradient Boosting Method

Extreme gradient boosting (XGBoost) followed a logical progression of predicting with tree based predictive models. While this boosted tree ensemble method can be significantly more computationally intensive, it can often result in a more adaptable and powerful model. By providing column subsampling in addition to row subsampling, and using higher order approximations for the minimization of loss terms, it was prudent to determine the effectiveness of this modeling method in the application of forecasting for Corporacion Favorita.

Again, in order to find the ideal parameters, a grid search was utilized to find the optimal parameters. For XGBoosts parameter tuning, the variables of interest to be checked were: **n_trees**, or number of trees, between a range of 50, 100, 150 and **learning rate** was inputted for a range of 0.1, 0.01, 0.001. Additionally, the default **max_depth** of 10 was again acceptable. It was notable that computational resources to run the grid search were very intensive (about 6 to 8 hours for fitting the grid).

Upon completion of this process, the ideal number of trees determined was 100 with an optimal learning rate of 0.1. Utilizing these parameters, XGBoost produced slightly worse results when compared to the random forest, with a returned NWRMSLE value of 0.793. for XGBoost. As a third and final method of tree-based method to be utilized, Gradient Boosted Machine (GBM) modeling was then implemented for comparison.

1.6.3 Gradient Boosted Machine Regressor Method

Gradient Boosting Machine Regressors (GBMs) are a type of ensemble model that uses decision trees as the weak learners in gradient boosting. The loss function for GBM can be any differentiable loss function, although GBM models, including the one implemented in this report, typically utilize mean squared error. Gradient boosting utilizes a loss function, a weak learner for predictions, and an additive model for weak learners that minimizes the loss function.

The trees are constructed in a greedy manner, where the criterion is quality of a split, as determined by mean squared error (MSE) or Friedman MSE. To avoid overfitting, the weak learner trees are often restricted in some way, such as the maximum number of splits. The more constrained tree creation is, the more weak learners will be necessary to build an accurate model. As the trees are added, the loss is calculated. Each subsequent tree is chosen so that its addition to the model reduces the overall loss and follows the gradient along the path to the minimum. This process repeats iteratively until the model reaches a predefined loss level or until the preset number of trees has been added.

When it came to implementation of this model, it was important to find the appropriate combination of the input variables yielding the best result. The parameters for the Gradient Boosted model were also determined using a Grid Search in order to iterate through different subsets of predictor variables, and the output of results is shown in table 1.1 below. Ultimately, the model was optimized by varying the number of trees (weak learners), maximum tree depth, learning rate (rate at which the model moves to minimize the loss function), and minimum number of rows in each leaf. Surprisingly, increasing the number of trees did not improve the model fit. This was an indication that the model was not a good fit for our data.

Table 1.1: Parameters and results of GBM models used.

Predictor Variables	Target	Tree #	Max Depth	Min Rows	NWRMSLE
10	Unit Sales	5	5	10	0.905687
4	Unit Sales	5	5	10	0.881413
2	Unit Sales	5	15	2	0.881955
2	Unit Sales	5	20	2	0.882507
1	$\ln[\text{Unit Sales} + 1]$	5	5	5	0.933275

Overall, the Gradient Boosting models were prone to overfitting and were not able to perform significantly better than the nave forecast, and were outperformed by the mean item

sales benchmark. Additionally, this process was very computationally taxing, and a rework was necessary to utilize this going forward.

1.6.4 Multi-Layered Perceptron Method

As an alternative option to tree-based methods, the Multi-Layered Perceptron (MLP) was implemented in hopes of minimizing the NWRMSLE. MLP was chosen to integrate stochastic gradient descent into the model building, which learns in an iterative process based on past data. With this feature, MLP would ideally provide an advantage versus the tree-based methods. As MLP was implemented for the project, the data was standardized in order to better prepare the data prior to iterating through predictions by removing the mean and scaling the data to unit variance. Once these preparations were completed, target parameters were chosen. A detailed descriptions of each parameter and a short reasoning of why the parameters were chosen are provided below:

- **Hidden Layer Sizes:** represents the number hidden layers to be used in the MLP. This provides additional steps where inputs can be transformed into more useful tools to provide an optimum output. In modifying this parameter, the value was varied between 2 and 4.

- **Activation Function:** is the activation function[3] specified for the hidden layer. For this parameter, two options were selected - the tanh and ReLU. From our experience with TensorFlow to illustrate the usage of neural networks in regression problems, tanh performed well and converged to a result quickly. Additionally, according to Hinton et. al, ReLU trains roughly six times quicker than tanh - hence it could be an optimal choice to reduce training time.

- **Learning Rate:** represents how quickly the MLP learns over time[4]. While too high of a learning rate would not be able to converge easily, too small of one would waste computation time without convergence. Intuitively, the learning rate should be set at a reasonable value to start, and then varied. To accomplish this, an adaptive learning rate was selected in order to provide a heuristic approach to tune this parameter, which will in turn minimize the NWRMSLE.

- **Batch Size:** determines the number of samples that are going to be propagated through the network. Precautions were taken here to optimize the batch size so that significant computation time and memory would not be wasted. For the project, the range of values chosen varied between 200 and 1000 samples for the various iterations

run with MLP. Again, to determine the best value, the lowest NWRMSLE would be considered.

Table 1.2: Results of various MLP parameters used.

Hidden Layers	Activation Function	Learning Rate	Batch Size	NWRMSLE
2	Tanh	0.005	500	0.916
4	ReLU	Adaptive	200	0.903
5	ReLU	Adaptive	1000	0.909

From the results shown above, the lowest error returned using MLP was 0.903, with the following parameters: Hidden Layers = 4; Activation Function = ReLU; Learning Rate = Adaptive; Batch Size = 200. This provides an interesting result - as a higher number of hidden layers and lower batch size seemingly provided a minor improvement versus the other iterations attempted. Nonetheless, these results did not return a better error metric than what was produced using the tree based methods. Therefore in moving forward, those methods were more heavily considered.

1.7 Modified Approach

As in the case with any predictive process, there must be constant evaluation of the progress and consideration of adjustments to create a better predictive model. Upon reviewing NWRMSLE scores from the models previously fit on all data, it was apparent the models run were struggling to forecast demand. With some scores only marginally better than a nave forecast which Kaggle set as one of their benchmarks, and none being able to beat a simple mean forecast, there was obvious room for improvement.

Furthermore, upon analysis of the variable importance plot shown in the Random Forest section, it became apparent why the models were not performing well item number was the most important variable utilized in the prediction. While this was an intuitive result since the goal of the forecasting was to to predict the sales of each product at an itemized level, with over 4,000 items in the dataset this also suggested that the models were being overwhelmed by the item number itself and could not focus on the truly important factors to predicting. Because sales were not correlated across items, there is no simple and effective way for a single model to develop an effecive tree ensemble that covered all items. This revealed that an adjustment to methodology was necessary to better improve the forecasting models.

What was best for the modeling approach going forward was to revise the approach and predict on a more granular level. Instead of utilizing the entire training dataset at one time, the strategy was devised to use each algorithm and predict on each product separately. It was hypothesized that in doing so, this would give the models the freedoms needed to create a better product forecast. As anticipated, this gave very promising results that will be described in depth.

Additionally, during this modified approach, as the scoring metric utilized a natural logarithm while rating models, it was determined that it would be better to predict with the log of sales. Again, it was hypothesized that this change would help to linearize the data, and thus allow the predictive models to become more accurate during forecasting. This as well gave a significant reduction in the scoring metric and the results are discussed below.

1.7.1 Modified Random Forest

Using the approach discussed previously for the random forest model, a similar approach was determined here. Using grid search suggested that a max tree depth of 10 and number of trees of 50 would be optimal parameters to minimize NWRMSLE. When running our model for each individual product, the Modified Random Forest showed much better results than previously discussed for this type of model and produced a score of 0.542 on the scoring metric. While this was a significant improvement from the previous fit, additional improvement could still be seen in scaling the data.

Upon doing this and predicting on $ln(\text{sales} + 1)$ yielded an improve forecast that reduced the NWRMSLE slightly further to 0.513. Going forward, this approach seemed to be a more logical approach in reducing the error, and would be implemented in all modified approach models.

1.7.2 Modified GBM

As similarly done for other models, once $ln(\text{sales} + 1)$ was implemented for GBM, results were significantly improved. LightGBM[5] was incorporated as a more robust version of GBM that was previously attempted, due to its quick training time and ease of implementation. This transition to Light GBM allowed the model to train 18 times faster.

This was selected over a modified XGBoost approach since the original XGBoost had a significant computational limit. LightGBMs NWRMSLE reduced to **0.507** using a combination of the following parameters: Learning Rate = 0.1; n_estimators = 40; max depth = 25. The LightGBM ultimately provided the best results out of the modified models. From

here, an ensemble method was built to try to build upon the success of both the GBM and random forest models.

1.7.3 Modified Tree Based Ensemble

The ensemble method as aforementioned was built with the intention to choose the best model type for each item number based on a subset of the training data. The ensemble would then fit that model to the entire data set.

We used the model parameters we had previously determined were best for each model, then attempted to refine the parameters in the context of the ensemble method. As this method utilized two tree-based models, it was important to only change the parameters for a single model at a time to better understand its effects while modeling to attribute the (positive or negative) change in NWRMSLE to. The ensemble algorithm was developed with the thought process as follows:

(a) If there are fewer than 5 rows of data for an item in the training set, choose the random forest model.
(b) If there are 5 or more rows of data in the training set, perform an 80/20 split to obtain a reduced training set and a validation set.
(c) Fit a random forest model to the reduced training set (80%) and score based on the validation set (20%); note that calculations do not need to include weight since the model is simply used for comparison purposes.
(d) Repeat (c) with a GBM model.
(e) Choose the model from (c) and (d) with the lowest score (NWRMSLE).
(f) Fit the model to the full training set and predict on the test set.

The ensemble algorithm was able to achieve the some of the best scores on our test set. The ideal parameters for the random forest model prior to the ensemble model were 200 trees with max depth of 5. Additionally, the GBMs best parameters were 25 trees and a max depth of 50. Ultimately, it was determined that the ensemble model used the same parameters for random forest, but performed better when utilizing modified parameters for GBM. The best model proved to be an ensemble of a Random Forest with 200 trees and max depth of 10; combined with a GBM of 100 trees and max depth of 3. This produced a NWRMSLE of 0.510.

Once these results were gathered, the ensemble method provides the most consistent approach due to the low NWRMSLE values produced, but ultimately LightGBM itself provides

a more robust approach due to saving on computation time and ease of application. The ensemble method was still a successful approach as it further verified the validity of utilizing $ln(\text{sales} + 1)$ in prediction as well as confirming the effectiveness of both Random Forests and GBM.

1.7.4 Modified MLP

The modified approach was also extended to MLP for completeness despite the initial approach yielding a preference for tree-based models. The intent was again to identify whether or not an itemized prediction would perform better than the initial MLP. For parameter tuning, a grid search was again performed, which returned 2 hidden layers, an activation function of tanh, and a learning rate of 0.01.

The results of using the modified MLP again did not produce results that compared well with tree-based methods. The NWRMSLE returned in predicting the sales without any scaling was 0.831. Similarly for log(sales), the NWRMSLE only slightly improved to 0.795, but this value did not even match up to Kaggles mean item sales forecast. Therefore, it was safe to say that moving forward, MLP would not be further utilized in hopes of minimizing the NWRMSLE and tree-based methods will be the only types of models to be considered.

1.8 Final Summarized Results

All model result values have been compiled into the following table:

Table 1.3: Summary of models generated.

Method	Initial Approach	Revised Approach	Log Sales Approach
Naive Forecast		0.911	
Mean Item Sales Forecast		0.726	
Random Forest	0.728	**0.542**	0.513
XGBoost	0.793	0.553	N/A
GBM	0.881	**0.542**	**0.507**
MLP Regressor	0.903	0.831	0.795
Tree-Based Ensemble	N/A	N/A	0.511

1.9 Conclusion

Ultimately, this data set provided a variety of significant issues along the path to completing a model with impactful business value. Numerous methods attempted and lessons learned along the way cumulatively resulted in the arrival at our best model: a LightGBM approach trained across all stores trained to predict the $ln(\text{sales} + 1)$ for each specific product.

The LightGBM model proved to be the most computationally efficient method and predicted product sales with a NWRMSLE of 0.507. This result was a significant improvement over the previously used less technical method of forecasting in which products were simply forecast using the previous mean item sales - a method which produced a NWRMSLE of 0.726. The generated method effectively reduces the forecasting error by slightly greater than 30%.

With this increase in accuracy, Corporacion Favorita can be more confident with their purchasing and stocking decision process. Having a more accurate forecast means needing a smaller surplus of product inventory to maintain the same confidence interval that their stores can meet demands without completely depleting inventory. This simultaneously reduces cost and waste that previously could not be recouped.

Bibliography

[1] Henning Redestig *https://www.rdocumentation.org/packages/pcaMethods* 2013.

[2] Amazon Web Services H2O *http://h2o-release.s3.amazonaws.com/h2o/master/3474/docs-website/h2o-py/docs/intro.html* 2015.

[3] A. Krzhevsky, I. Sutskever, G. Hinton *http://www.cs.toronto.edu/ fritz/absps/imagenet.pdf* 2012.

[4] Suki Lau *https://towardsdatascience.com/learning-rate-schedules-and-adaptive-learning-rate-methods-for-deep-learning-2c8f433990d1* 2017.

[5] Microsoft Corporation *https://github.com/Microsoft/LightGBM/tree/master/python-package* 2017.