Learning Optimazation Report: Store Sales - Time Series Forecasting

Conor Fallon; Tassilo Henninger

1/21/23

Table of contents

1 Project Goal and Outline

The task at hand is to build a time-series forecasting model which will make predictions for 15 days after the last date in the train data. The dataset is based on a real-life retailer 'Corporación Favorita' from Ecuador and contains over 1,000,000 entries in the training data, featuring both categorical and continuous features. Additionally, there are other datasets which contain information which could impact sales, such as oil prices over this period, or holidays, which can be incorporated if desired to improve the model performance.

The primary train dataset is to be split into a new train and a validation set.

Our tactic for model selection is as follows: simply compare the below models with the aforementioned baseline models, predicting solely on the date and ignoring the family and store categories (i.e. we will group by date). The assumption here is that the model that performs best under this simplified process will also perform better when the family and store categories are taken into account.

The baseline methods are Exponential Smoothing, Double Exponential Smoothing, and a SARIMA model. These are then compared with two other models: Prophet, an algorithm by Facebook; and decision trees (regression tree and a Random Forest). For each of these, where relevant, the final parameters chosen were decided upon through a mixture of EDA, trial-and-error, and hyperparameter optimization.

This is a 'top-down' approach to this type of forecasting. We will rely on the distribution of the sales amongst the stores and families in the train dataset as our basis for how to divide up the sales that occur on a given day to the various stores and families. This will be how we make our final prediction for each id in the test set.

Our metric for selection, as per the instructions in the kaggle competition, is the Root Mean Squared Logarithmic Error (RMSLE). However, other metrics will be used in order to better make comparisons between the methods. This comparison of error will be done on the validation set from our earlier splitting of the initial train set. The best model will be decided based on which performs best under this comparison, and then the outlined 'top-down' approach will make the final predictions.

2 Data and Information

The dataset is from a kaggle "getting started" competition on time-series forecasting. The data originates from a large grocery retaile named "Corporación Favorita" from Ecuador. It contains information of 54 stores and 33 product families for the timeperiod of 2013-01-01 to 2017-08-31. The goal is to build a model that predicts the unit sales for each product family and store combination for the next 15 days after the last given timepoint. The given data consists of the following 6 files:

- Train: contains time series of the stores and the product families combination. The sales column gives the total sales for a product family at a particular store at a given date. Additionally we have the onpromotion column, which gives the total number of items in a product family that were being promoted at a store at a given date.
- Test: same structure as train, except without the 'sales' value - this is what we have to predict with our model. A prediciton will be made for the 15 days after the end of the train set.
- Store: gives some information about stores such as city, state, type, cluster.
- Transactions: contains the number of total transactions for each store at a given date.
- Holidays and Events: contains information of holidays during the timeperiod. The columns are date,type,locale,locale_name,description and transferred
- Daily Oil Price: contains the oil price at a given date. This is important for predictions as Ecuador is an oil-dependent country and it's economical health is highly vulnerable to shocks in oil prices and thereby also the sales of the grocery retailer.

Additional notes for the challenge are:

- Wages in the public sector are paid every two weeks on the 15th and on the last day of the month. Supermarket sales could be affected by this.
- A magnitude 7.8 earthquake struck Ecuador on April 16, 2016. People rallied in relief efforts donating water and other needed products which greatly affected supermarket sales for several weeks after the earthquake.

2.1 Preprocessing

The preprocessing we done is mainly restricted to the train, test and the holidays and events data. The first thing we did is splitting the train data into a taining and validation set. The Training data will be used for training, the validation data will be used for model comparison. We then aggregated the train, validation and test set by date over all stores and product families to get the overall sales at a given date. We used that datastructure for all of the follwing models: Exponential Smoothing, Double Exponential Smoothing, SARIMA and the decision trees.

For the Prophet model we additonally used the holidays and events data as well as the on-promotion feature. The holidays and events data needed quite some preprocessing as some national holidays have been transferred and it needs to be accounted for that. Additonally there are regional and local events and holidays which make it even more complicated. As we havent used the regional feature we decided to only include the national events and holdays as they apply for the whole contry and therefore all stores.

3 EDA

To start off we did a lot of exploratory data analysis. We will cover only the most interesting findings in the following. Additional plots are referenced in the text and can be found in the appendix.

3.1 Transactions

We found a stable pattern in the boxplot of the transactions (Figure 1). All months are similar except of the month December. In Figure 13 we the same behavior across every store. The store sales always increased at the end of the year.

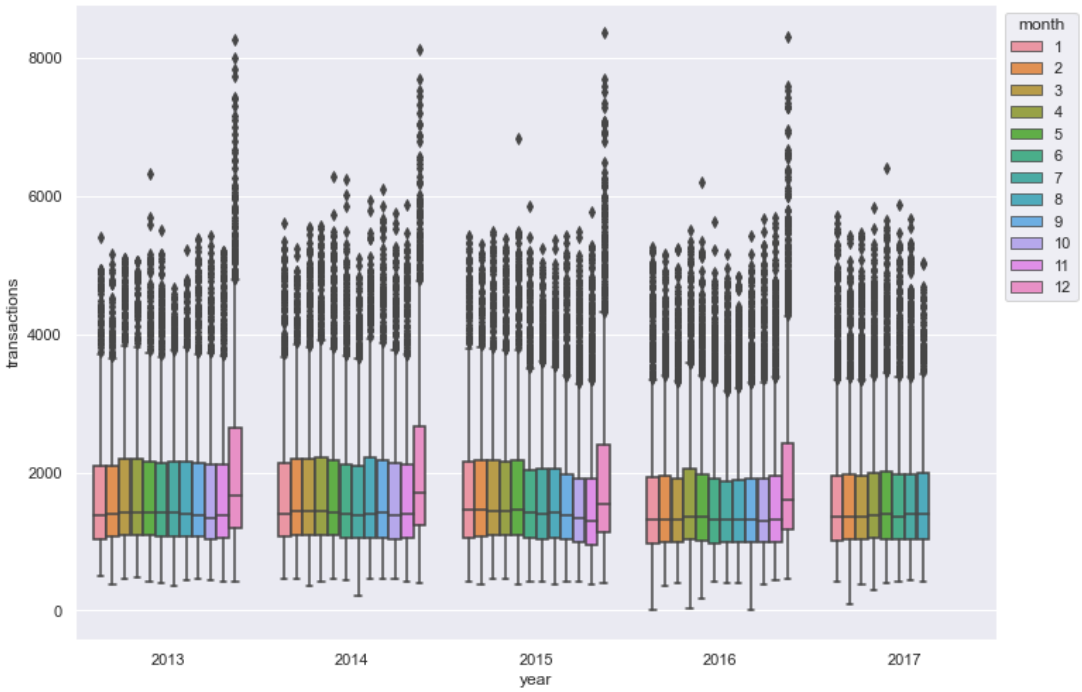

Figure 1: transactions_boxplot

Additional, the days of week seem to be very important for shopping. In Figure 2 we can identify a strong pattern. More transactions are made at the weekend. This pattern is consistent for all the years (2013-2017). Saturday is clearly the most important day for shopping.

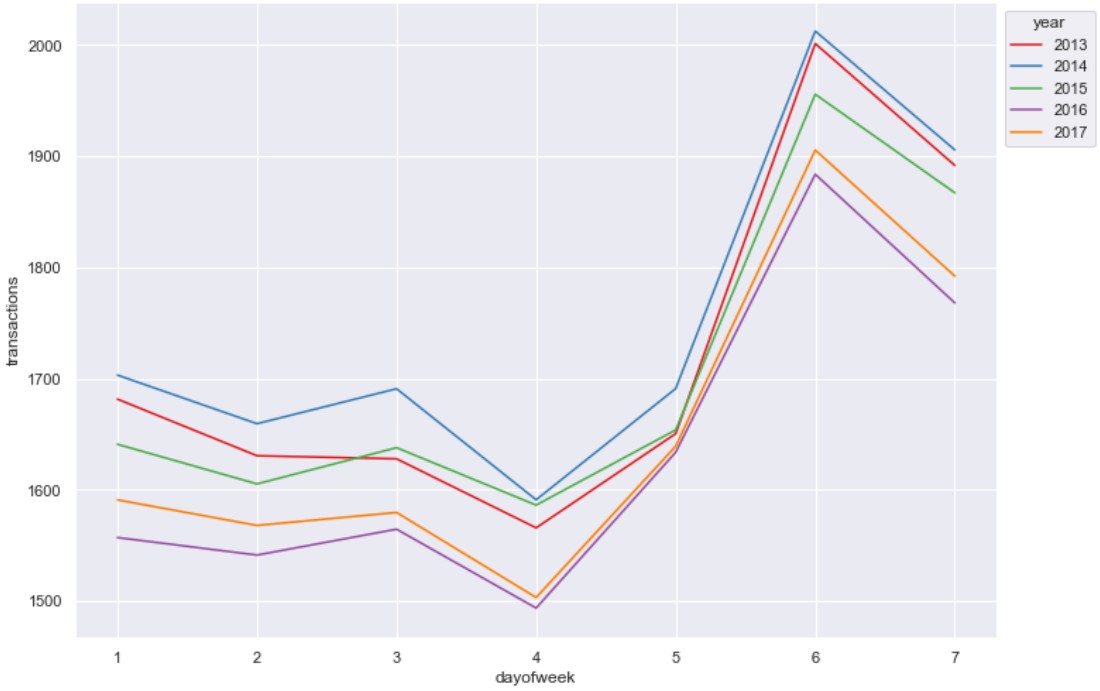

Figure 2: transactions_dayofweek

3.2 Sales

Our main objective is to predict store sales for each product family and at each store. For this reason, the sales column should be examined more closely. We started of with EDA, to get some general understanding of the data, followed up with investigating the autocorrelation, which can tell us how timeseries are correlated to their lags, and ended with STL decomposition.

We started EDA by plotting the correlation among stores for overall-sales at each timepoint. By examining Figure 14, we can see that most of the stores are similar to each other. Some stores, however, such as 20, 21, 22, and 52, may be a little different.

Next we took a look at the daily total sales over all stores and products. We can see clear drops in Figure 3. By doing further analysis we found all the sales < 100.000 which are listed in the following table. Interestingly, this is always the day after New Year's Eve. This is perhaps unsurprising as many of the shops will be closed on New Year's Day.

date	sales
2013-01-01	2511.62
2014-01-01	8602.07
2015-01-01	12773.62
2016-01-01	16433.39
2017-01-01	12082.50

Next up we created a seasonal plot. A Seasonal plot is similar to a time plot except that the data are plotted against the individual "seasons" in which the data were observed. We created it over month (Figure 15). We can see the same behaviour like in the transaction data. At the end of the year we have an increase in sales over all years, which is reflective of an increasing trend overall.

Additionally, we found that the feature onpromotion column is positive with the sales at a given day (Figure 16). A bigger onpromotion value (mean per day over all products and stores) leads to a higher sales value.

3.2.1 Autocorrelation - How are timeseries correlated to their lags?

Just as correlation measures the extent of a linear relationship between two variables, autocorrelation measures the linear relationship between lagged values of a time series. The lag features means shifting a time series forward one step or more than one. This autocorrelation for an observation and an observation at a prior time step is comprised of both the direct correlation and indirect correlations. These indirect correlations are a linear function of the correlation of the observation, with observations at intervening time steps. The partial autocorrelation function removes those indirect correlations.

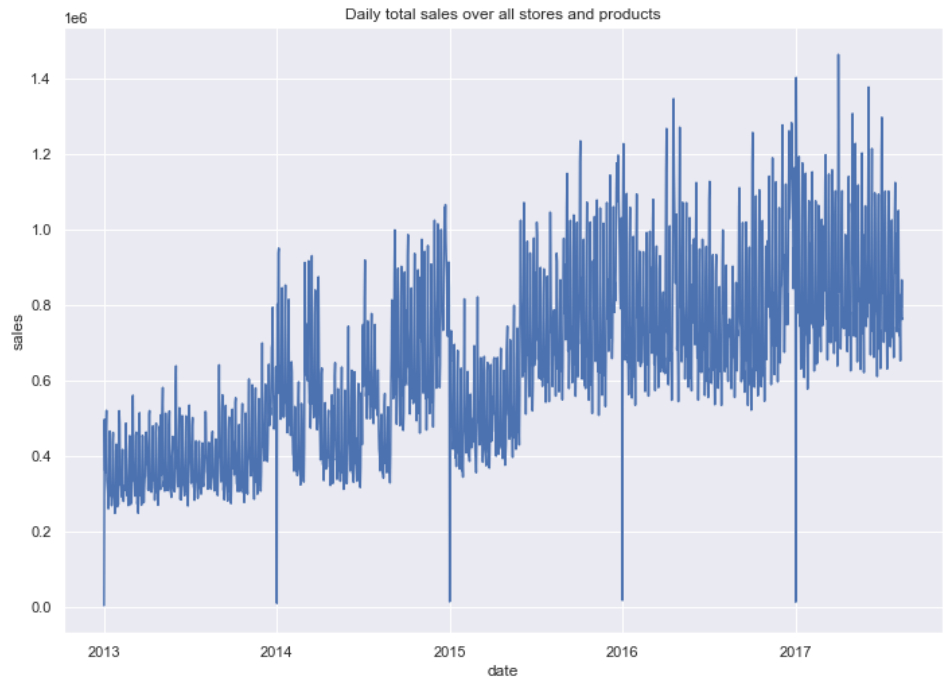

Figure 3: Transactions DayofWeek

To better understand the data we can use the autocorrelation function (ACF) and partial autocorrelation function (PACF) and identify the siginificant lag features, that we should include into the model. We see can clearly see (Figure 4) a 7 lag period in the ACF, which corresponds to a weekly periodicity. That was already expected, but it is good to have some statistical backing. The PACF shows a bigger partial autocorrelation for the 1,6 and 7 lags. Those lags could be interesting to include in the model.

Additionnally we also check the timeseries for stationarity. A time series is said to be stationary if its statistical properties do not change over time. In other words, it has constant mean and variance, and covariance is independent of time. We run the Dickey-Fuller statistical test to determine if a time series is stationary or not. The Dickey-Fuller-Test tests if a unit root is present. As the p value is at 0.0897 and bigger than 5%, we can not reject the nullhypothesis, and therefore have to assume that the process is not stationary.

3.2.2 STL decomposition

STL is a robust method for decomposing time series data. The acronym stands for "Seasonal and Trend decomposition using Loess". It decomposes the data into a trend, season and the remaining part. The remainder should ideally be white noise, as a trend or pattern in the noise hints to an underlying feature which influences our data. We set the period parameter

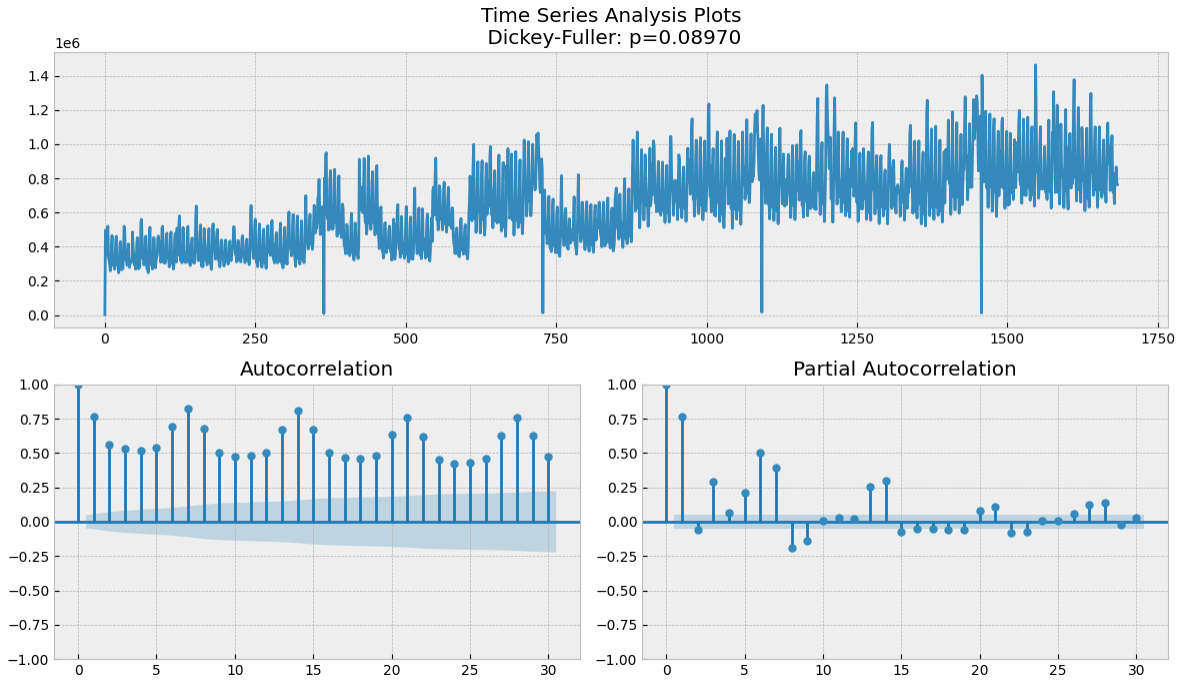

Figure 4: Autocorrelation

to 365 days for the following plot (Figure 5). The trend component is quite clear. We can see an upwards trend for the sales. The seasonal component reveals a yearly pattern. We also got a clear weekly pattern by setting the period parameter to 7 days. The remainder component mostly shows white noise. Except of some outliers above and below. The outliers below are at the same time as our findings during EDA in section "Time Plots". They occure all from the day after New Year's Eve. Good to make the same findings across different methods.

Figure 5: STL Decomposition

4 Forecasting

In the following chapter we applied the models introduced in the outline. We trained each model on the training data and evaluated it based on the evaluation set. For each forcasting method we give a detailed explanation how it works and share the optained metric results.

4.1 Baseline model - Exponential Smoothing

We start of with a baseline model. The naive forecast would be using the last value or a moving average. What we choose as a basline is the simple univariate model exponential smoothing.

Exponential smoothing was proposed in the late 1950s and is a method of weighted averages of past observations, with the weights decaying exponentially as the observations get older. It is very simple, but therefore is not capable of modelling a seasonal or trend component. The formula is the following:

Forecast Equation: $\hat{Y}_{t+1} = \alpha y_t + \alpha(1-\alpha)y_{t-1} + \alpha(1-\alpha)^2 y_{t-2} + ...$

Therefore you only have to set the parameter α and decide how many previous values you want to use for the prediction. We based out prediction only on the last value as a baseline and tried different numbers for alpha out. Our manual tries were worst then setting the optimization option = True.(Figure 6) Therefore, we choose the optimized version for calculating the validation error. We got the following metric scores:

Model	RMSLE	MSE	MAE
Exponential Smoothing	0.13876294591305696	1.316025e10	99941.04

4.2 Double Exponential Smoothing

The exponential smoothing method got extended by Holt to add a trend component. As we have a clear trend in out data, that is a good idea. The Double Exponential Smoothing involves a forecast equation and two smoothing equations (one for the level and one for the trend) and can be seen in the following:

Forecast Equation: $\hat{y}_{t+h} = l_t + hb_t$

Level Equation: $l_t = \alpha y_t + (1-\alpha)(l_{t-1} + b_{t-1})$

Trend Equation: $b_t = \beta(l_t - l_{t-1}) + (1+\beta)b_{t-1}$

We have now the new parameter beta and h to set. Alpha is again the smoothing for the level and beta now the smoothing for the trend. The parameter h is the number of steps you want to forecast ahead. As we basing out prediction only on the previous value, we set it to 1. Our

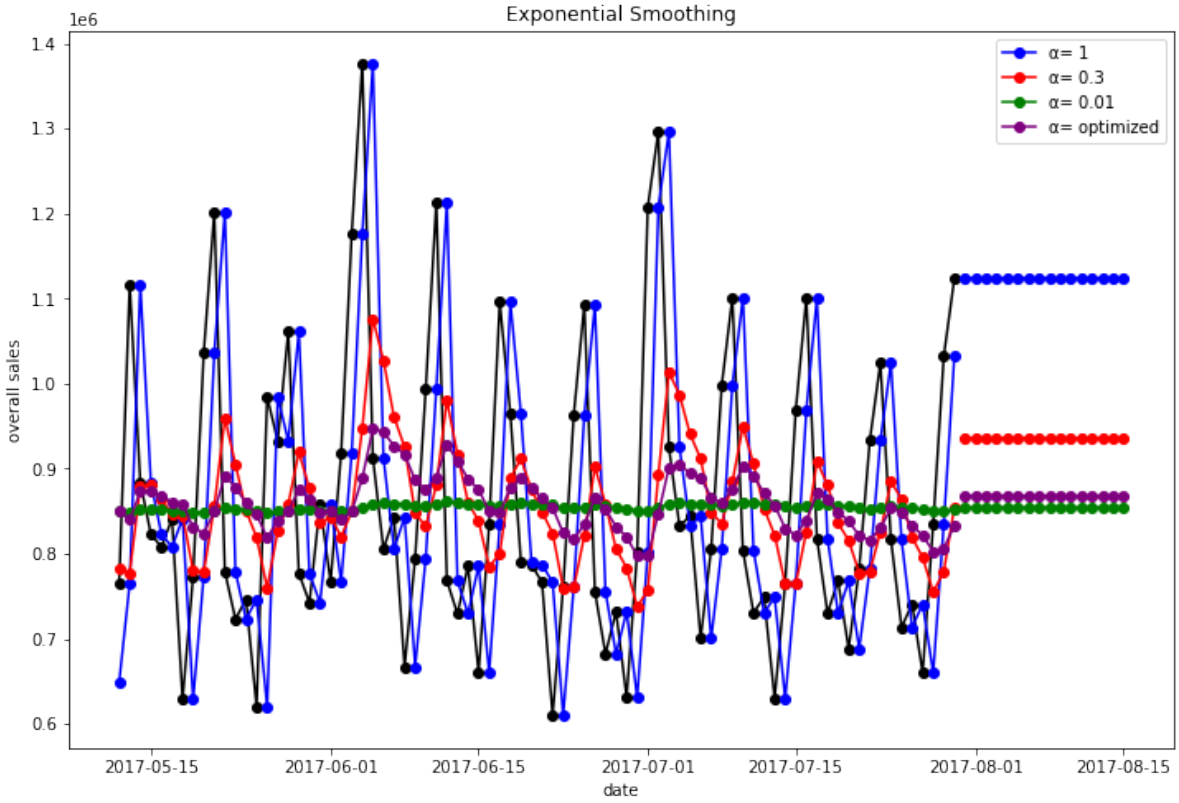

Figure 6: Exponential Smoothing

manual tries were worst then setting the optimization option = True (Figure 7). Therefore, we choose again the optimized version for calculating the validation error. Of course alpha and beta do not have to be the same. We got the following metric scores for different alphas and betas:

Model	RMSLE	MSE	MAE
Double Exponential Smoothing	0.1393858049730297	1.327847e10	100600.89

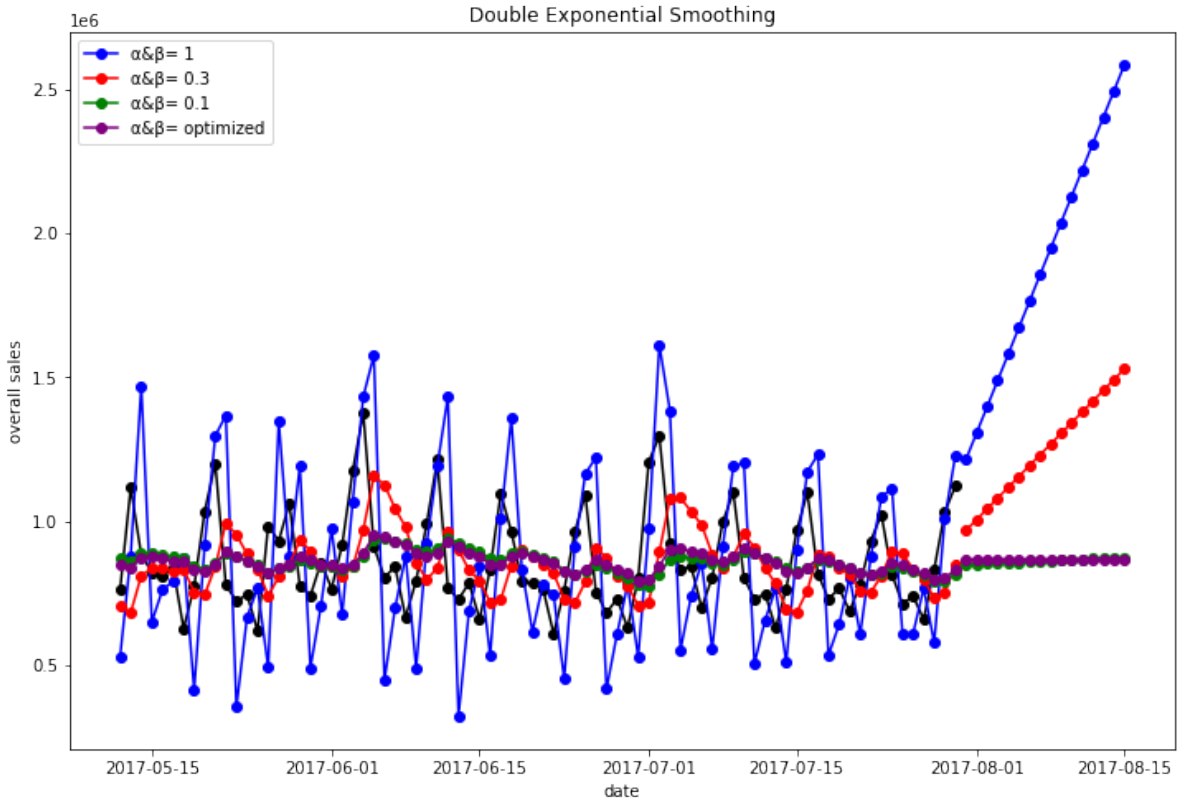

Figure 7: Double Exponential Smoothing

There would also be a further addition called Holt-Winters method, whichs adds an additional seasonal component. For that there are two variations. There is the additive method, which is preferred when the seasonal variations are roughly constant through the series, while the multiplicative method is preferred when the seasonal variations are changing proportional to the level of the series. With our series we would have chosen the multiplicative version.

4.3 SARIMA model

The SARIMA model acronym stands for "Seasonal Auto-Regressive Integrated Moving Average".

The "autoregressive" (AR) component of the model is represented by AR(p), with the p parameter determining the number of lagged series that we use. We found out in the ACF and PACF (chapter "Autocorrelation"), the first 7 lags, particular lag 1,6 and 7 are interesting for that. After the first 7 lags it repeats itself and the lags are not significant anymore. The "integrated" (I) is the difference order, which is the number of transformations (d) needed to make the data stationary. The parameter d represents the number of differences required to make the series stationary. As we have found out in chapter "Autocorrelation" that we have to assume non-stationary for our series, which comes in handy. The "moving average" (MA) is the moving average model with its parameter q as the number of included error lags. In an MA(1) model, our forecast is a constant term plus the previous error times a multiplier, added with the current error. The "Seasonal" (S) component extends the ARIMA model with an additional set of autoregressive and moving average components with the parameters S(P, D, Q, s). The parameters P and Q are the same as p and q, but now for the seasonal component. D is again the order of seasonal integration representing the number of differences required to make the series stationary. S is simply the season's length. This was found to be 7, i.e. the seasonality in the model we selected is considered to be best represented by weekly seasonality. Combining all, we get the SARIMA(p, d, q)(P, D, Q, s) model.

In order to choose the other parameters for this SARIMA model, only a single number of differencings for both the seasonal and non-seasonal parts were needed to make the series stationary. Hence, both D and d are 1. Next, trial-and-error was used to select the order of the autoregressive (AR) and moving average (MA) terms. For the moving average, the fact that there the autocorrelation plots showed peaks at lag 1 made this a good starting point for our P and p terms. Our choice of Q and q equaling 1 for the Moving Average parts implies that only the error term from one time point prior is being taken into account when making our next prediction. For all of these Ps and Qs, perhaps higher values could have produced better model performance, but at the risk of model overfitting. Finally, the residuals of the model (shown in the top left of the diagram below) Figure 17 seem to be distributed like noise, which implies that the model was a good fit for the data and that there are no major patterns left to be explained (with the exception of the four peaks, which are presumably representative of the stores being closed on New Year's Day; this can be subsequently taken into account).

The performance on the vaildation data is given in the following table, graphically it can be seen in Figure 8:

Model	RMSLE	MSE	MAE
SARIMA	0.134025810826792	1.391188e10	83125.15

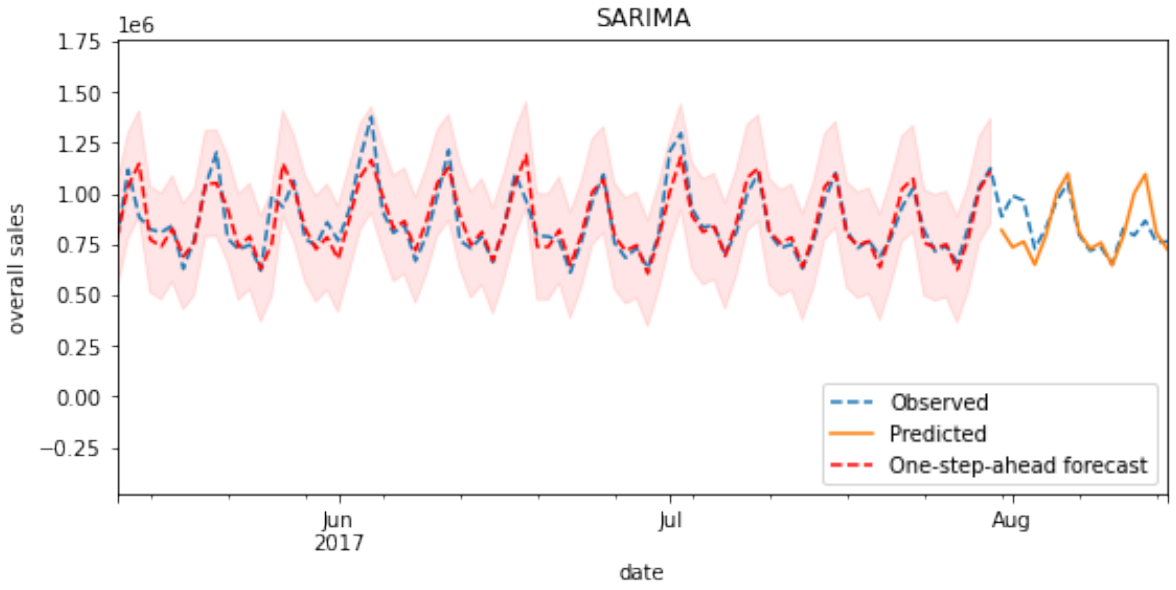

Figure 8: SARIMA model

4.4 Prophet

The Prophet algorithm was introduced by Sean J. Taylor and Ben Letham from Facebook in 2017. It is designed to be easy and completely automatic. > "It implements a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects."

Based on the additive model, we are also able to add regressors which coefficents can be interpreted as in the SARIMAX model.

We trained three different versions of the Prophet model. First we fit only on the training data. Next we added the preprocessed national holidays and events information as two different holiday types. For the third model we added the "on-promotion" feature as an additional regressor. The performance on the validation data of the three models is given in the following table:

Model	RMSLE	MSE	MAE
Prophet Baseline	0.10672917726787304	0.866192e10	67894.59
Prophet +Holidays	0.10471472504386517	0.840120e10	65890.04
Prophet +Holidays_OnPromotion	0.10471472504386517	0.840120e10	65890.04

As we can see adding the holidays and events information to the baseline was decreasing every

error metric. Adding the additional on-promotion feature on the other hand was not useful for the model. Therefore we stick with the "simpler" Prophet +Holidays model, which is graphically represented in Figure 9. On the components plot (Figure 18) of the final prophet model we can clearly see, that it picked up the trend component, the weekly and yearly component of the sales data.

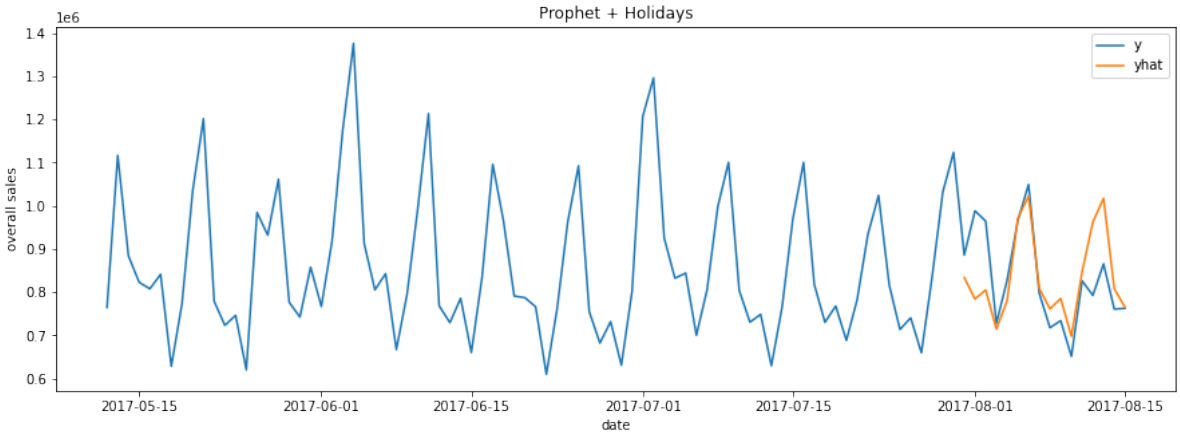

Figure 9: Prophet & Holidays

4.5 Regression Tree and Random Forest

The code for this is visible in the 'Decision Tree and Random Forest' notebook.

A regression tree is a specific class of tree which will predict a numerical dependent variable based on a number of explanatory variables. This can be difficult to model through simpler models like linear regression; however, by partitioning the data into smaller regions, the interactions between the explanatory variables, potentially very complex, can become more manageable. In a tree model, this partitioning and sub-partitioning (i.e. recursively partitioning each partition) is done by the tree.

The splitting is done in a top-down, greedy manner, where the splitting starts at the top of the tree (i.e. all data-points belong to the same region), then this is split in 2, and this process is repeated for each split in the tree. It is greedy in the sense that it does not look ahead steps into the future; it simply predicts the best split for the split in question.

In preparing to train our models, we performed some feature engineering on the dataset such as the day of the week, the month, quarter and so on. This is to help the tree make better decisions as we have found things like the time of the year and the day of the week to have an important effect on the sales on a given day.

Something we were aware of in the creating of these models is that trees are not necessarily the best choice of model for time series where there is a clear trend, as there is in ours. As such, we

train our models to predict the *differences* between sales values, rather than the sales values themselves, in the hope that this can in someway get our model to implicitly learn about the ongoing trend in our dataset.

The performance of the regression tree on our validation set is as follows:

model	RMSLE	MSE	MAE
Regression Tree cheated	0.128111	1.104584e+10	78066.83
Regression Tree	0.307477	4.854608e+10	198903.99

The very good performance of our first version of Regression Tree and Random Forest, led us to examine our code and we had, admittedly, made a mistake in our construction of both. Our model had been allowed 'look' at points in the past, at least implicitly, when it should not have. Hence these models were essentially cheating. In a second iteration of both models, we ammend these faults. After that both models were not performing outstandingly anymore.

A plot of its predictions over the actual data is shown below (Figure 10). It is of course not as good as the cheated version in Figure 19.

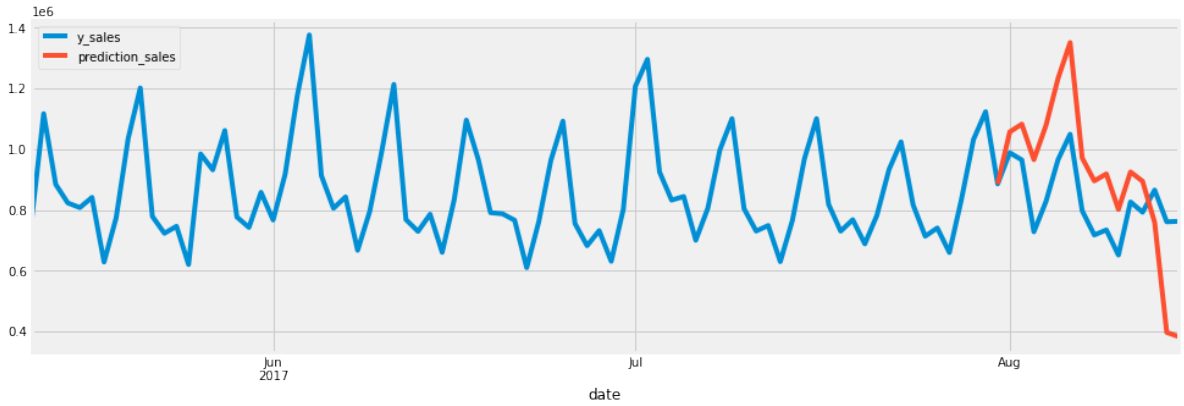

Figure 10: Decision Tree

Our other candidate is the random forest. A random forest is a type of ensemble method, in this case similar to bagging for a regression tree; however, fewer parameters are chosen for each step that the tree is grown. This allows for more subtle effects to be found in a random forest than in a bagged regression tree, because in bagging all the dominant variables are contained within each bag, which will mean that variables that produce more nuanced effects on the prediction become crowded out.

This model is optimised using a simple Grid Search with Cross Validation for hyperparameter optimisation in order to choose the the best number of predictor variables to feature in the

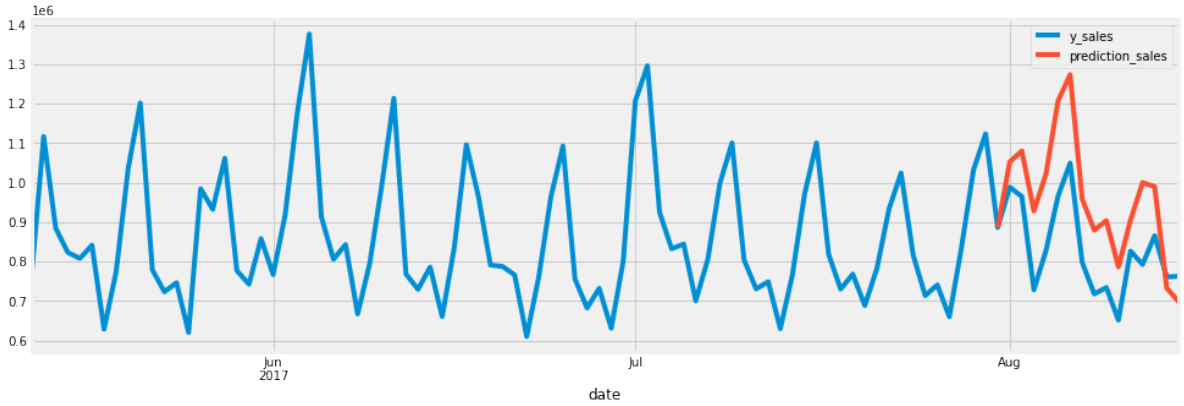

Figure 11: Random Forest

random forest. It produces the following prediction plot (Figure 11), again not as good as the cheated version in Figure 20.

The performance of the random forest is as follows:

model	RMSLE	MSE	MAE
Random Forest cheated	0.083058	0.4648461e+10	46158.14
Random Forest	0.173279	2.4810410e+10	144618.02

The random forest performs a better than the regression tree. If one looks at the graph at early August, it seems to have captured that particular 'n'-shaped pattern which we can see is a recurring type of shape in the data. The regression tree is however calculable hundreds of times quicker which may make it the more preferable choice of model as it allows us to iterate quickly where the random forest does not.

5 Model Comparisson

Model	RMSLE	MSE	MAE
Exponential Smoothing	0.138762	1.316025e10	99941.04
Double Exponential Smoothing	0.139385	1.327847e10	100600.89
SARIMA	0.134025	1.391188e10	83125.15
Prophet +Holidays	0.104714	0.840120e10	65890.04
Regression Tree	0.307477	4.854608e+10	198903.99
Random Forest	0.173279	2.481041e+10	144618.02

In the first iteration, we made a mistake in the implementation of the decision tree and the random forest. This resulted in outstanding results and led us to examine our code and finally correction our mistake. In the second iteration both tree based models have performed worst. The clear winner across all metrics is the Prophet model. Hence we will use the Prophet model for the final prediction tasks.

Finally, it is worth briefly noting that the SARIMA model outperforms the two exponential smoothing models. This is not unsuprising as SARIMA models have long been a powerful performer in time-series tasks, and its strong performance is further vindication of the arguments put forward for the selection of the SARIMA parameters as discussed earlier on.

5.1 Distribution for Top-Down Approach

There are 54 stores and 33 families which leads to 1782 possible combinations of the two being produced. The proportion of sales for each of these combinations has been found and this will be applied to the 'bottom' part of our approach to help us figure out how the sales on a given predicted day ought to be divided up. The distribution for each of the stores and familees can be inspected in the appendix (Figure 21 & Figure 22)

There is a somewhat naive assumption made here in proportioning the data: the proportions for all these various family product combinations were done over the entire period of the train set. This of course assumes that the proportion of sales is relatively static over time. An argument could be made that the most recent few weeks may be more suitable; however, some simple analysis showed that there did not seem to be a large difference in the proportion of products sold over time. A more robust statistical test could have been done here in order to be more rigorous. Qualitatively, we argue that taking the entire time-series into account is more robust as any extreme events (such as the earthquake) will be dampened by the largely normal patters that predominate. Also, it is possible that in taking the proportions of only the most recent weeks, we could be basing our top-down model on an anomolous time period unbeknownst to ourselves.

Note, we found that out that some stores don't sell some product families during EDA.This is also covered by the top-down approach.

5.2 Prediction of the Final Model

The code underlying how the final predictions were made is visible in the short notebook testing.ipynb. It produced 28512 predictions and these were submitted to Kaggle, giving us a user score of 0.65612, which placed our model in position 718 out of 892. A visual representation of the prediction of the unknown test data can be seen in Figure 12.

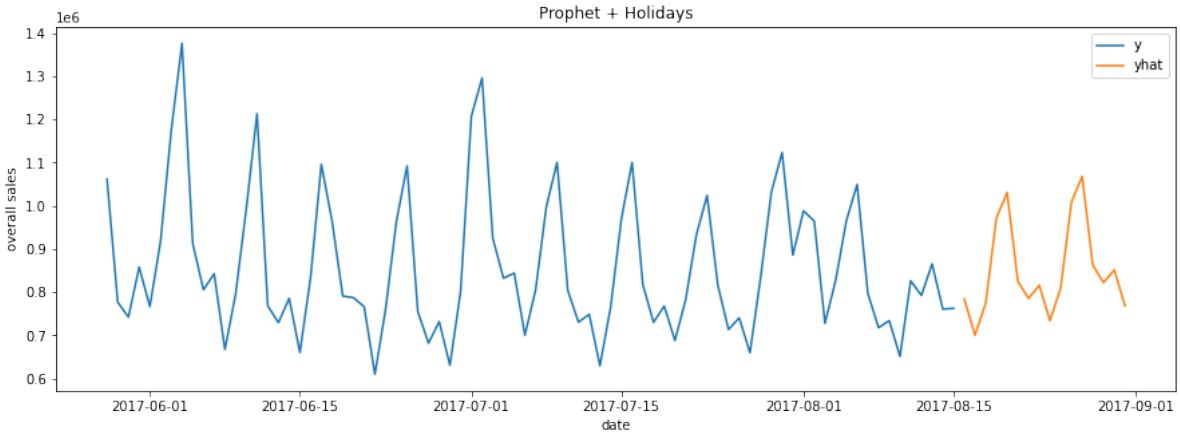

Figure 12: Prophet Prediction on Test Data

6 Future Work and Discussion

In this study, we evaluated a predictive model for retail sales using various techniques. Our best performing model was the Prophet model, which produced 28512 predictions and gave us a user score of 0.65612 at Kaggle. This placed our model in position 718 out of 892. It is hard to say if this is a good score or not, as we would have to know how much effort the other teams had put in to get the results that they did. We hope at the very least that the predictions are somewhat sensible. It is also worth mentioning that this is the only submission that we entered; other teams both above and below us in the rankings had multiple attempts.

There are several areas where the model could be improved in future iterations. Firstly, as demonstrated in the exploratory data analysis, stores 20, 21, 22, and 52 have relatively low correlation with the other stores. Further examination could involve an error analysis of how the predictions for these specific stores behave in terms of the difference between predicted and actual test values. However, it is possible that the top-down approach, which created pairwise store-family pairs, captures these anomalies.

Additionally, we excluded oil data, which could be a significant omission given that the country's economy is heavily dependent on the oil price. The reason for this is two-fold: Ecuador is a member of the Organization of the Petroleum Exporting Countries and its oil-production is state-owned, which impacts the strength of the economy, as well as for the government to provide services and social supports, which in turn will affect how much money people have to spend on things. Secondly, if oil prices increase, people will have to pay more for goods in general, not only for things like electricity but also for goods in shops, and the prices in shops will likely be higher owing to increased costs of business which will presumably negatively impact how much people can purchase.

Another omission in our model is the lack of transactions data. It is reasonable to expect that this would improve our predictions and would be included in future iterations of our model. Similarly, public wages were not included in the model. It would be easy to incorporate this as a sort of 'holiday' variable in the Prophet model. However, one could also argue that public sector workers are a small enough section of the population that it would not have a significant effect on sales.

Finally, the earthquake that occurred has not been explored in our EDA nor incorporated into our modeling. It is possible that the earthquake had a prolonged and large impact on the dataset. Given that earthquakes are relatively rare and unpredictable events, it would have been an option to interpolate more 'regular' values for this time period in this specific case. In our modeling, there is the implicit hope that the earthquake would be more akin to noise in the data given the rather prolonged time period in question.

In conclusion, while the model performed well in predicting retail sales, there are several areas where it could be improved in future iterations by incorporating omitted data and exploring the impact of rare events such as earthquakes.

7 Appendix

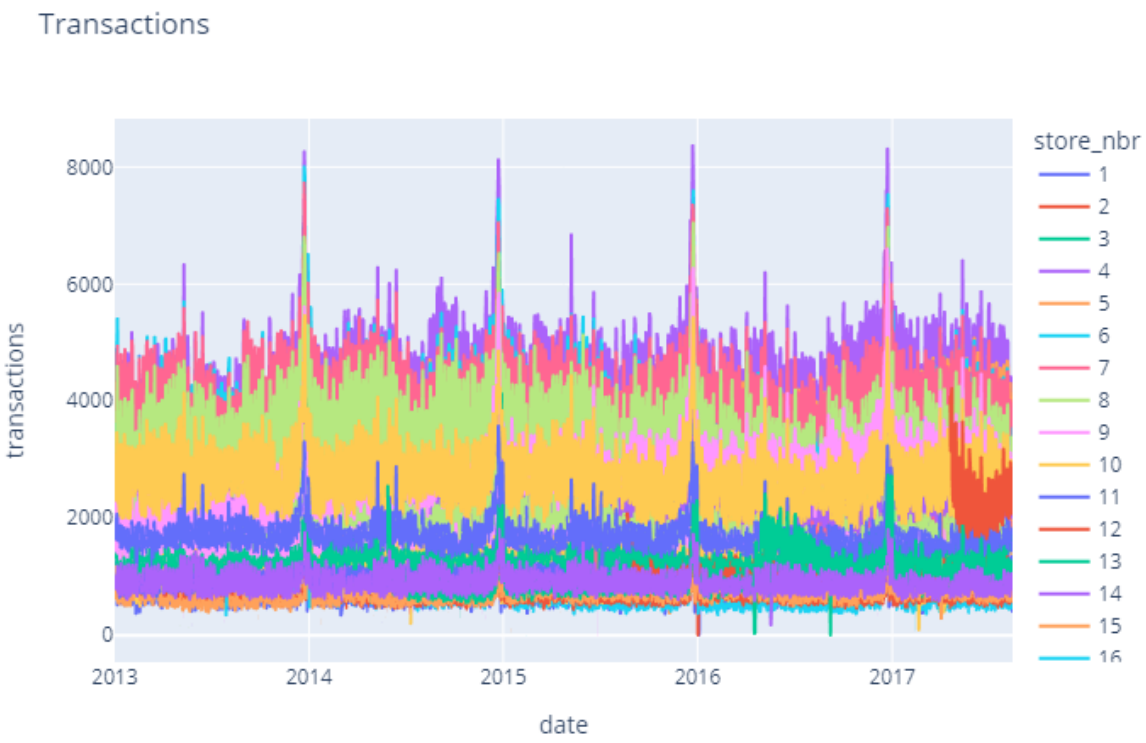

Figure 13: transactions_timeline_by_store

Figure 14: correlation matrix among stores

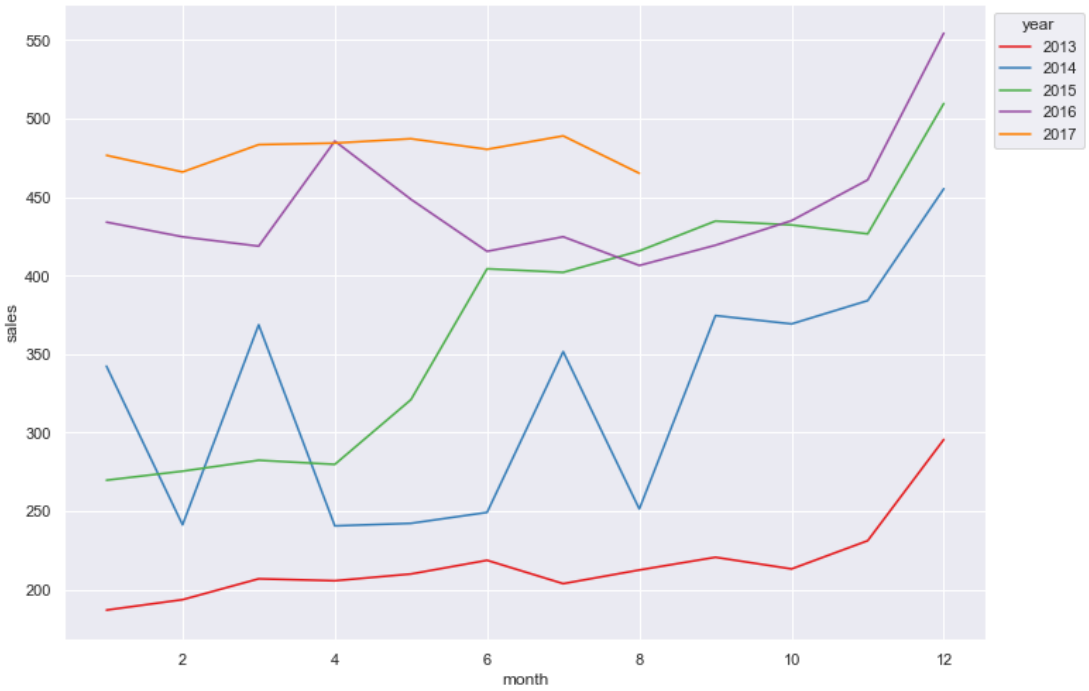

Figure 15: Seasonal Plot for Sales over Months

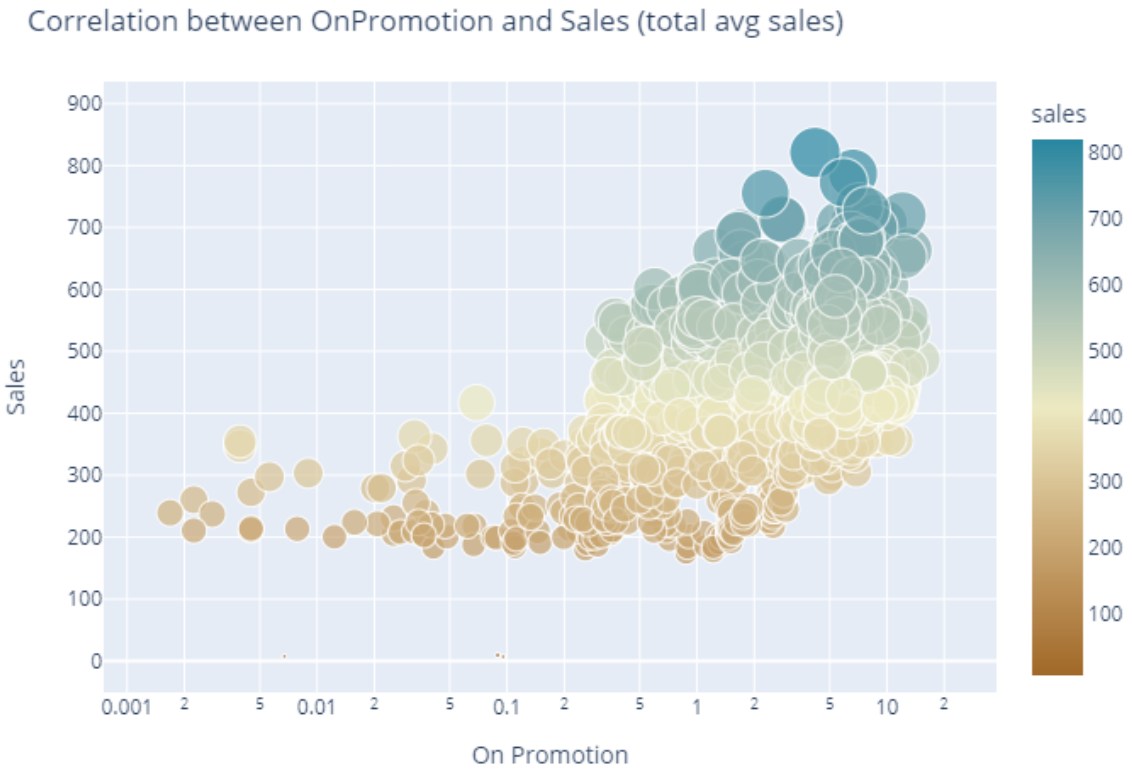

Figure 16: Correlation between Onpromotion and Sales

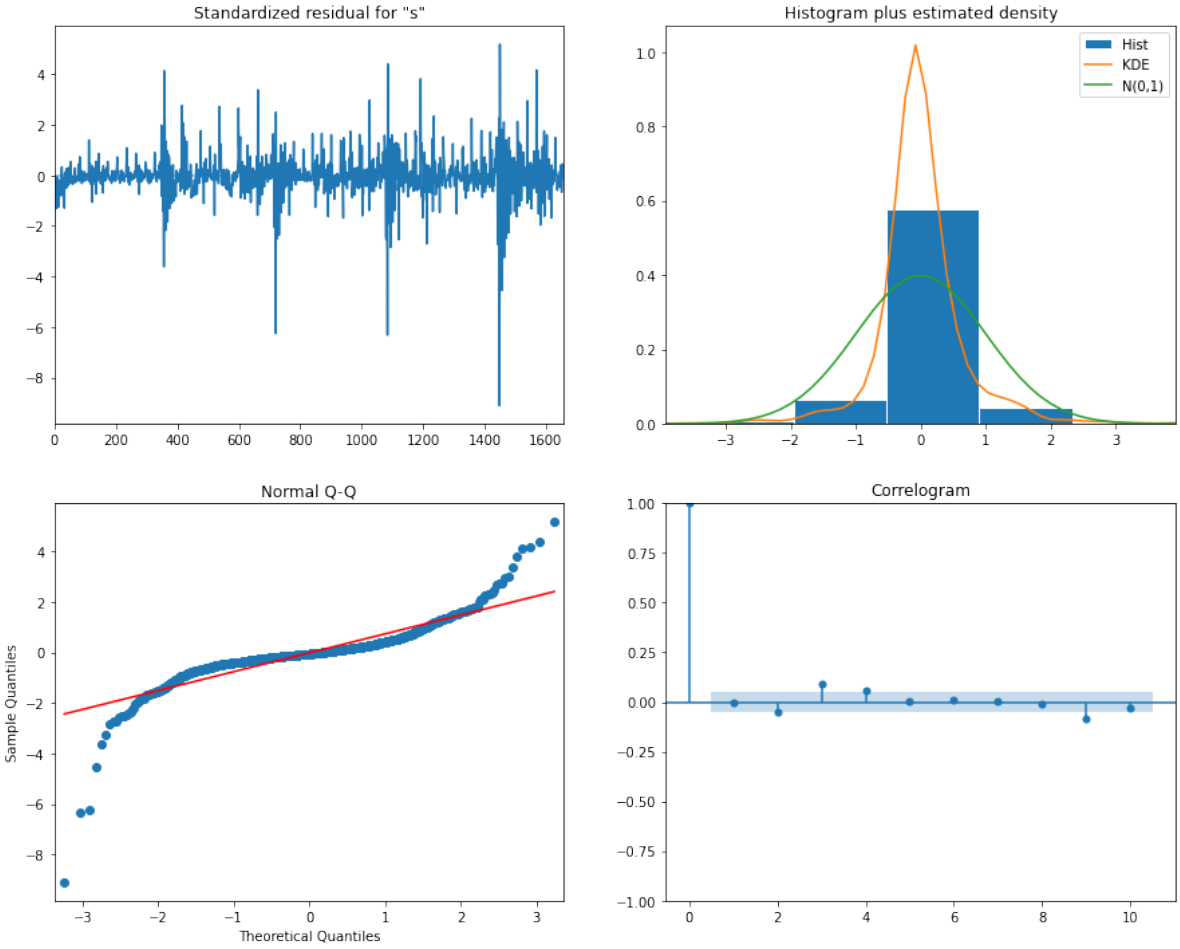

Figure 17: SARIMA Diagnostics

Figure 18: Prophet Components

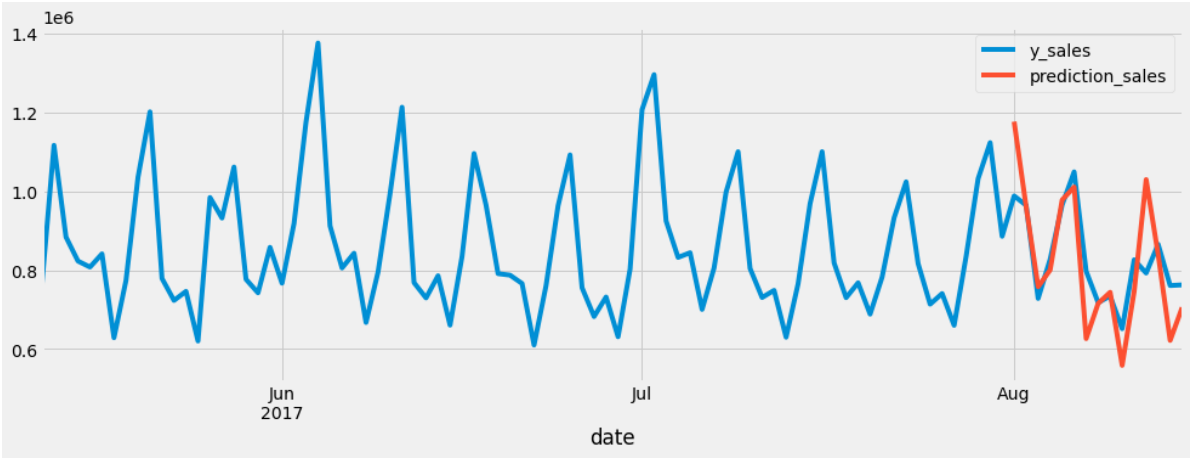

Figure 19: Decision Tree Cheated

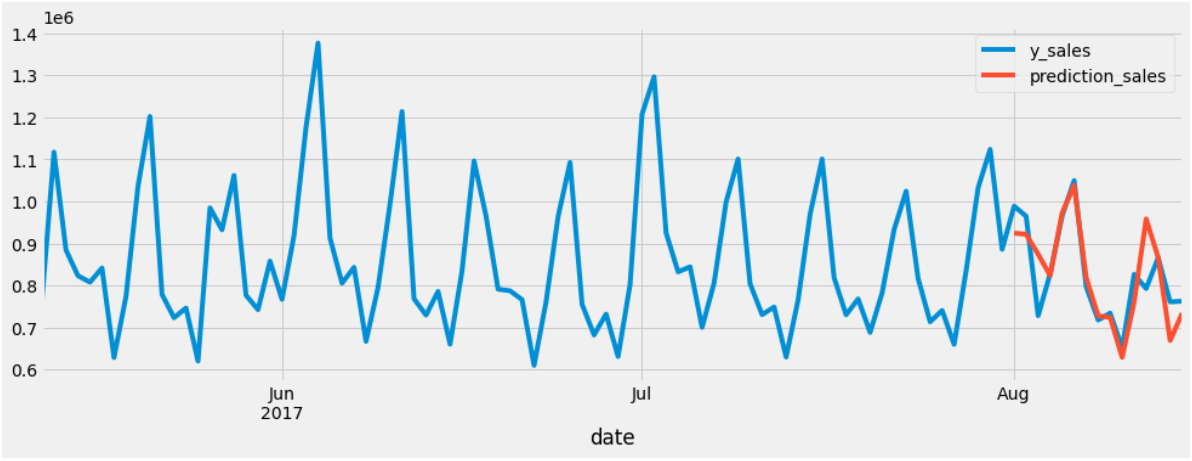

Figure 20: Random Forest Cheated

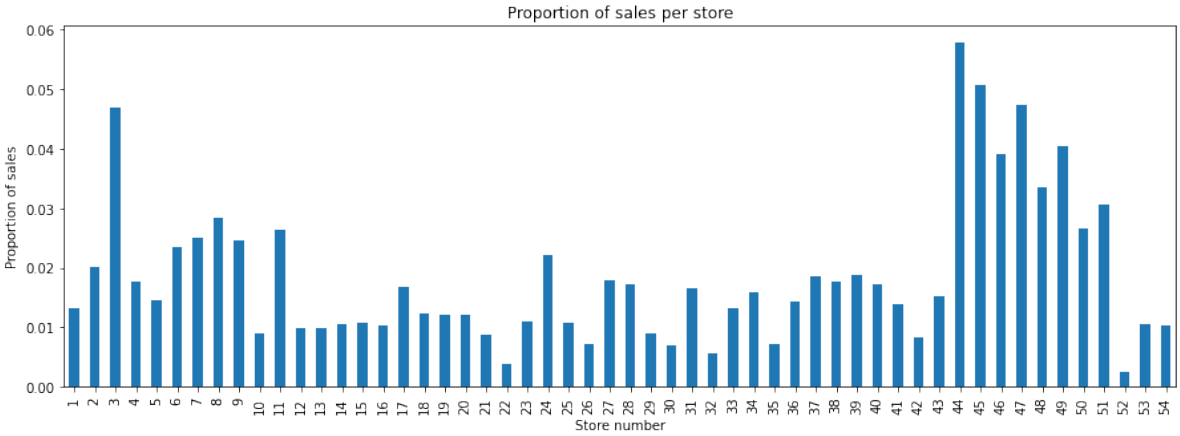

Figure 21: Distribution of Stores

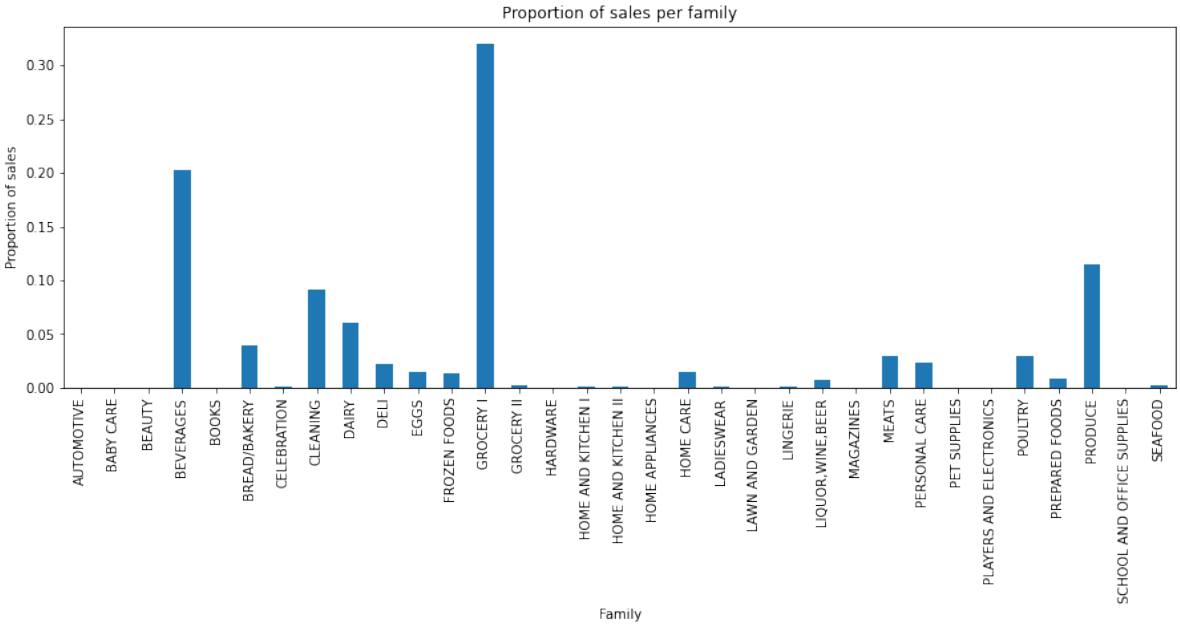

Figure 22: Distribution of Families