CS 229 Project: Rossmann Time Series

Allen Huang
Stanford University
allenh@stanford.edu

Jesiska Tandy
Stanford University
jtandy@stanford.edu

1. Introduction

Datasets that vary with time are becoming increasingly important and prevalent in business and industry. Therefore, we felt that it would be interesting to learn more about the differences and nuances of interpreting and predicting time series data. To that end, we decided to use the dataset provided in a Kaggle's competition titled "Rossmann Store Sales"[1] to build and verify models that could be used to forecast sales of stores over multiple months given historical sales.

2. Related Work

As the creator of the R forecast package Rob J. Hyndman writes in his online textbook, there are many different types of techniques when dealing with time series data. There are decomposition-based methods that include moving-average techniques and other robust techniques such as STL ("Seasonal and Trend decomposition using Loess"). [1] Some of the techniques explored include clustering and random forests [2, 3]. There are even more advanced techniques such as exponential smoothing, ARIMA models, and even autoregressive neural networks models. [4] Classification and regression trees (CART) have been compared to many of these techniques for predicting time series, and while it may not be considered the best when it comes to time series [5], it is worth noting that all these complex models are fixated on forecasting based solely on a single time series. That is, these techniques were not made to handle the case where you have many other covariates and multiple possibly correlated time series data.

3. Dataset and Features

Table 1: Summary of the given features we used during the course of the project.

Column / Feature	train.csv	test.csv	store.csv	Comments
Store	x	x	x	Factor: (1-1115); store ID
DayOfWeek	x	x		1-7 (1- Monday, 7- Sunday)
Date	x	x		year, month, day
Sales	x			
Promo	x	x		Binary: 1 for active promo
StateHoliday	x	x		Binary: 1
SchoolHoliday	x	x		Factor: 0, a, b, c
StoreType			x	Factor: a, b, c, d
Assortment			x	Factor: a, b, c

4. Methods

We separated the given training data into three datasets: Training (1/1/2014 - 5/31/2015), Validation (6/1/2015 - 6/30/2015), and Test (7/1/2015 - 7/31/2015).

We also used a log transform on the sales outcome variable because sales revenue is inherently a positive outcome and the logarithmic transform mitigates the effect of the large range of sales across different stores.

[1] https://www.kaggle.com/c/rossmann-store-sales

The metrics we ultimately decided to use was the root mean square percent error and the mean absolute percent error. These metrics were computed in aggregate across all the daily sales figures for all the stores and for the entire month of the validation set, as shown below:

$$RMSPE = \sqrt{\frac{1}{D \cdot S}\sum_{i=1}^{S}\sum_{j=1}^{D}\left(\frac{y_{ij}-a_{ij}}{a_{ij}}\right)^2} \quad (1) \qquad MAPE = \frac{1}{D \cdot S}\sum_{i=1}^{S}\sum_{j=1}^{D}\left|\frac{y_{ij}-a_{ij}}{a_{ij}}\right| \quad (2)$$

where y_{ij} is the predicted sales for a given store and a given day, a_{ij} is the actual sales, S is the total number of stores, and D is the total number of days in the dataset.

Our intermediate and final model use recursively partitioned trees (rpart R package). The model is as follows [6]:

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m) \quad (3)$$

where C_m is a constant in each region. Minimizing the sum of squares $\sum(y_i - f(x_i))^2$, we would obtain the average of y_i as the best $\hat{c}_m$ [6], depicted in the equation below:

$$\hat{c}_m = ave(y_i \mid x_i \in R_m) \quad (4)$$

In words, the idea is that given n features, we partition the n-dimensional space into subregions. Given a new data point, we find the subregion in which it resides and guess the average outcome of all the training data points in the subregion. Recursive partitioning is a greedy algorithm that finds the optimal splits that define these regions.

The complexity parameter (cp) is the threshold above which a split in the tree has to improve the relative error for a split to be accepted. That is, a smaller cp value yields trees that have more branches.

5. Experiments

5.1 Baseline Models

Seasonal Naive Method

First, there was the seasonal naive method of simply predicting the median sales based the Store, DayOfWeek, and Promo covariates. Note that we also tried predicting the mean, though the median seemed to give better and more consistent results. This is shown in Figure 1 below.

Bagged Decision Tree Model

We began with an unbagged rpart tree model with the following R formula: Sales ~ Store + DayOfWeek + Promo + StateHoliday + SchoolHoliday + StoreType + Assortment.

We performed sensitivity analysis of the complexity parameter and the results are shown in Table 2. The results indicated that the optimal cp value was 0.00001 as decreasing the cp value started to yield only marginal improvements but increased the computational time significantly.

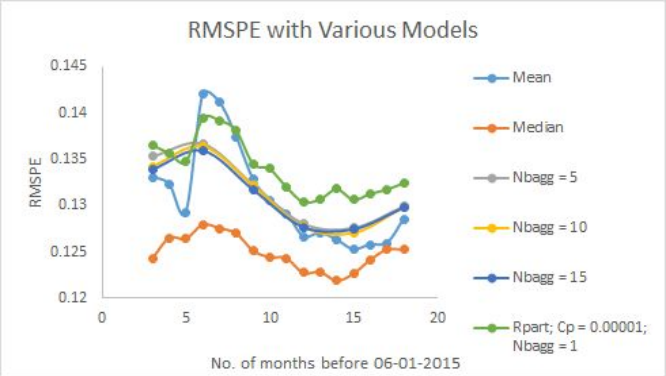
Table 2: RMSPE value with varying Cp.

Cp	0.01	0.001	0.0001	0.00001	0.000001	0.0000001
RMSPE	0.273	0.217	0.154	0.139	0.137	0.136

We then extended the rpart to a bagged regime whereby we bootstrapped the training dataset B times, fit B decision trees, and averaged the results of those B decision trees. This gives us a more robust result becauses it reduces the variance component of our generalization error since we're averaging over bootstrapped datasets. We do not expect that this process will reduce the bias component of our generalization error as we are not adding features.

5.2 Understanding the Effects of Time Series

In order to see one effect that time series had in our predictive power, we trained the models above on a subset of our training data. The idea was that data that was too long ago might be less relevant at predicting the sales of today.



As seen from Figure 1, all the models uniformly got better as you increased the amount of historical data it was allowed to train on up until around 15 months. When you add more, the predictive model starts to get worse. As a results, all of our models only rely on 15 months worth of historical data. The data after that seems to be noise!

Figure 1: Performance of various models trained on various lengths of historical data.

5.3 Finding Time Series Covariates

In order to find what sort of time series features would be the most relevant to predicting the sales of today, we did a series of autocorrelation analyses on the Sales of individual stores.

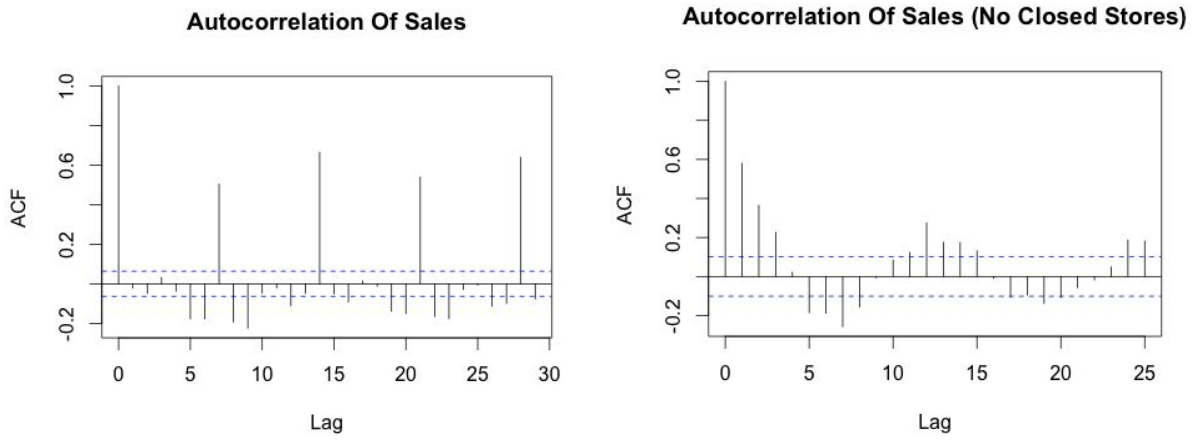


Figure 2. a) Autocorrelation of Sales including closed stores. b) Autocorrelation of Sales excluding closed stores.

Our first pass with autocorrelation (Figure 2a) showed a significant correlation between the sales of today and the sales 7, 14, 21, and 28 days prior. However, it was surprising to us that the sales of previous day did not correlate much at all, and upon further thought we realized that this was due to the inclusion of sales of closed stores. Since a majority of the stores were closed on Sundays, the zero sales on those days dragged down the correlations significant and accentuated the day of week effect. After removing those days when the sales were zero, the autocorrelation showed a significant correlation with previous day's sales, confirming our intuition.

From this analysis, we decided to add the following time series covariates to each row of our dataset. In our model, tm(i) represents the sales i days before the current sale date, where i = {1, 2, 3, 4, 5, 6, 7, 14, 21, 28}. MA7 represents the moving average sales over the last week, and MA28 represents the average sales over the last 4 weeks.

5.4 Significance of Time Series Covariates and Interaction Terms

From our autocorrelation graphs, we knew that the time series covariates that we added were important. However, in order to identify the relative importance between those features and their interactions, we decided to use an ordinary least squares (OLS) approach.

The advantages of the OLS approach were that A) we could directly inspect the significance of the time series coefficients and B) its computational efficiency (i.e. we could train ~1000 linear models whereas our bagged decision trees took approximately an hour each to train). The disadvantage was that we could not use Store as a factor in the analysis. (With 1115 stores, the lm method in R creates 1114 (i.e. 1115 - 1) dummy variables, which drastically exploded the number of columns of the design matrix it was trying to manipulate/invert).

Thus, we trained a separate OLS model for *each individual store* with all the columns of the time series features described above as well as with every single interaction term. R formula: Sales ~ tm1 + tm2 + tm3 + tm4 + tm5 + tm6 + tm7 + tm14 + tm21 + tm28 + MA7 + MA28 + **.*.**. We subsequently looked at the significance of the coefficients.

Table 3: **Abridged** output of the summary of the resultant linear model fit with time series data for store 229.

Covariate	**Coefficient**	**Standard Error**	**p-value**	**Importance**
Intercept	6.712	9.66E-01	2.36E-11	***
tm1	2.26E-04	7.62E-05	0.003283	**
tm2	2.17E-04	7.24E-05	0.002986	**
tm4	1.88E-04	7.77E-05	0.01608	*
tm21:MA7	-3.32E-08	8.03E-09	4.63E-05	***
MA7:MA28	2.05E-07	2.87E-08	7.63E-12	***

(Note: More stars means the coefficient is more likely to be significant. *** indicates 0.001 false positive rate for that coefficient.)

For us, we wanted a 0.01 false positive rate, but in order to account for multiple hypothesis bias, we used the Bonferroni correction in order to ensure that the coefficients we chose actually had a 0.01 false positive rate. Thus, our p-value threshold was 0.01 / m where m is the number of covariates that we had. After analyzing the significance coefficients from some randomly selected stores, we decided to include the following coefficients:

tm1 + tm2 + tm4 + tm7 + MA7 + tm1:tm2 + tm1:tm4 + tm1:tm6 + tm1:MA28 + tm2:tm3 + tm2:tm7 + tm2:MA28 + tm3:tm4 + tm3:tm7 + tm4:MA28 + tm6:MA7 + tm7:MA28 + tm21:MA28 + MA7:MA28

5.5 Final Model: Bagged Trees with Time Covariates

Armed with our time covariates, we trained our final bagged tree model with the following R formula[2]:

Sales ~ Store + DayOfWeek + Promo + StateHoliday + SchoolHoliday + StoreType + Assortment + tm1 + tm2 + tm4 + tm7 + MA7 + tm1:tm2 + tm1:tm4 + tm1:tm6 + tm1:MA28 + tm2:tm3 + tm2:tm7 + tm2:MA28 + tm3:tm4 + tm3:tm7 + tm4:MA28 + tm6:MA7 + tm7:MA28 + tm21:MA28 + MA7:MA28

[2] We used a cp parameter of 0.00001 and 20 bootstrapped datasets. (It took approximately an hour to train.)

6. Results

We compared the RMSPE and MAPE values of our baseline models and our final, best model. For our validation set, We found that the error values of the naive seasonal baseline model and our bagged tree without time covariates model produced very similar error values; however, our bagged tree with time covariates model gave a RMSPE and MAPE values that are 1% our baseline models. This 1% error will actually translate to a large number of sales, thus it is a huge improvement over our baseline models. When we ran our models on our test set, we actually saw that our baseline models got worse but our final model stayed roughly the same in terms of performance. This suggests that our final model with the proper time covariates is actually reasonably robust against overfitting.

Table 4. RMSPE and MAPE values of our models for validation set (val) and test set (test).

Model	**RMSPE (val)**	**MAPE (val)**	**RMSPE (test)**	**MAPE (test)**
Baseline (Median)	0.125	0.093	0.1373	0.102
Bagged Trees without Time Covariates	0.128	0.095	0.1457	0.108
Bagged Trees with Time Covariates	0.117	0.085	0.118	0.084

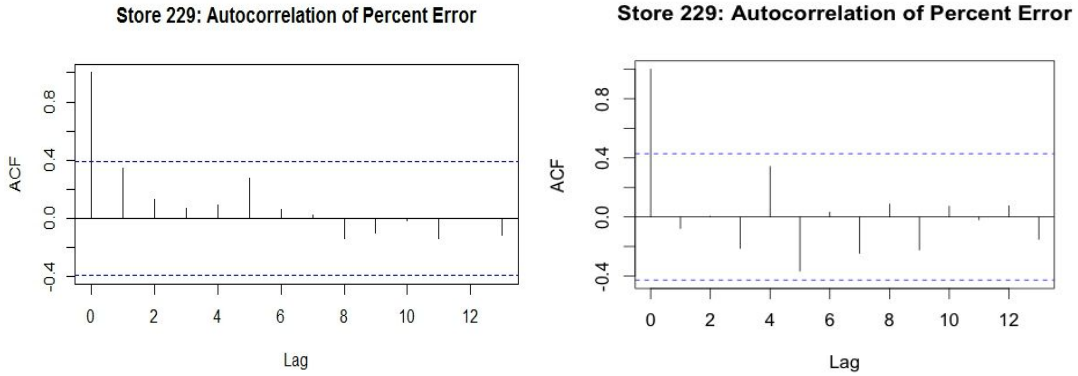


Figure 3. a) Autocorrelation of percent error using our bagged tree without time covariate model. b) Autocorrelation of percent error using our bagged tree with time covariate model.

Autocorrelation of the residuals of both the baseline and "bagged trees without time covariates" model yielded graphs that were similar to Figure 3a. The linear pattern in the residuals suggests that there was still some temporal structure of the models that was not captured by the model. Figure 3b suggests that our final model takes advantage of the temporal structure, accounting for the improvement in prediction.

7. Conclusion / Future Work

Our bagged decision tree model with the most important time covariates yielded the best and most consistent results on both the validation and test set with 0.117 and 0.118 RMSPE respectively. Autocorrelation analysis of the residuals showed that our best model did a better job of capturing the temporal structure of the data. If we had more time and computational power, we would explore gradient boosted trees as well as gradient boosted regression since we saw during the CS229 poster session on 12/8/2015 that those techniques seemed to work well for other project teams tackling the same dataset. At an even higher level, we would also like to explore ways in which to incorporate some of the advanced time series forecasting techniques (e.g. autoregressive neural nets or ARIMA models) into a type of ensemble structure. That is, how can we make some of the more advanced time series methods work together on a dataset like this with many related parallel time series?

References

[1] Hyndman, Rob J. and George Athanasopoulos. “Forecasting: principles and practice.” 2013. Web: http://otexts.org/fpp/. Accessed on Dec 9, 2015.

[2] Wang, Xiaozhe, Kate Smith, and Rob Hyndman. "Characteristic-based clustering for time series data." *Data mining and knowledge Discovery* 13.3 (2006): 335-364.

[3] Oliveira, Mariana, and Luıs Torgo. "Ensembles for Time Series Forecasting."*Proceedings of the Sixth Asian Conference on Machine Learning*. 2014

[4] Bontempi, Gianluca, Souhaib Ben Taieb, and Yann-Aël Le Borgne. "Machine learning strategies for time series forecasting." *Business Intelligence*. Springer Berlin Heidelberg, 2013. 62-77.

[5] Ahmed, Nesreen K., Amir F. Atiya, Neamat El Gayar, and Hisham El-Shishiny. "An empirical comparison of machine learning models for time series forecasting." *Econometric Reviews* 29.5-6 (2010): 594-621.

[6] Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. "Tree-Based Methods." *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. Springer, 2013. 307. Print.

[7] Therneau, Terry, Beth Atkinson, and Brian Ripley. “Recursive Partitioning and Regression Trees”. R package version 4.1-10. (2015) http://CRAN.R-project.org/package=rpart

[8] Peters, Andrea and Torsten Hothorn. “ipred: Improved Predictors”. R package version 0.9-5. (2015) http://CRAN.R-project.org/package=ipred