Project 2: Walmart Store Sales Forecasting

1 Overview

We are interested in predicting the future department-wide weekly sales of Walmart stores based on historical data. A Kaggle competition [1] has provided the historical departmentalized-sales data for 45 Walmart stores located in different regions. Using only the *train.csv* in this data set, we build a prediction model and report on its prediction performance. We leverage a few findings from many data analyses available in Kaggle, especially from the winner code [2]. This report documents the workflow of data preparation, model development, and prediction performance.

2 Initial Assessment

We start with an overview of the data set. The "train.csv", provided by class staff on the Piazza Resources page, has 421,570 rows and 5 columns. Each row represents a weekly sales record for a department in a Walmart store. The columns are:

- *Store* – the store number
- *Dept* – the department number
- *Date* – the week
- *Weekly_Sales* – the sales for the given department in the given store
- *IsHoliday* – whether the week is a special holiday

Using an R script provided by class staff, we generate the following files:

- train_ini.csv [5 columns]:
  provides training data from 2010-02 to 2011-02.
- test.csv [4 columns]:
  provides testing data (*Weekly_Sales* column being removed) from 2011-03 to 2012-10.
- fold_x.csv where x = 1, 2, …, 10 [5 columns]:
  provides testing data with true *Weekly_Sales*, one for every two months starting from 2011-03 to 2012-10.

"train_ini.csv" contains 45 distinct store numbers ranging from 1 to 45, and 81 distinct department numbers that run from 1 to 99. Interestingly, there are 457 records with negative sales, which may be due to returned items.

Figure 1 shows the time histories of *Weekly_Sales* for a few departments. We notice that there are some consistent patterns in the department-specific data among different stores, and the pattern differs for each department. For example, department #1 shows some seasonal patterns whereas department #13 does not show any particular trend for the considered time range. Therefore, it is critically important to capture the proper pattern for each department.

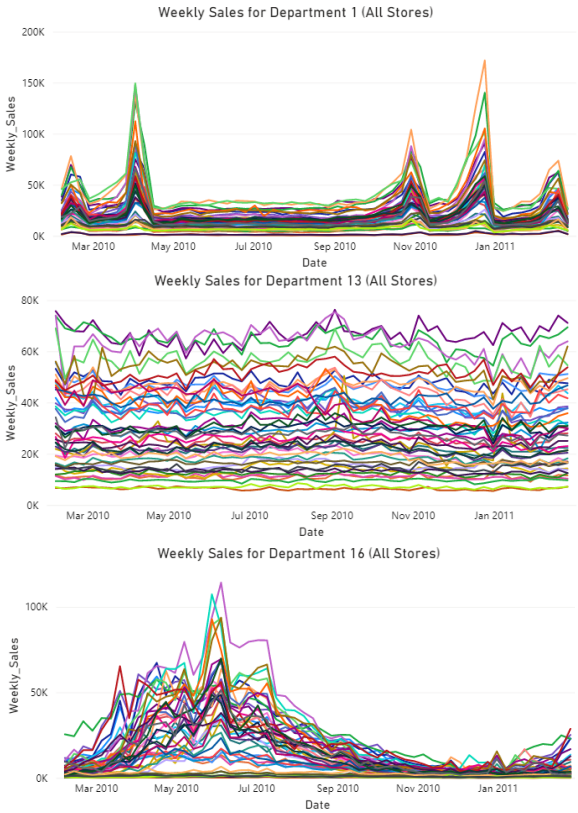

Figure 1 Time history of weekly sales for all 45 stores: department #1 (top), #13 (center) and #16 (bottom)

This suggests that instead of trying to predict the sales pattern among departments in a store, we should focus on predicting the sales pattern for each department among all stores. If a seasonal event is ongoing, we expect to see an increase in sales among all stores. Depending on the event type, some departments may not experience any change in their sales pattern. We also need to properly handle store-specific hikes in the data by applying some form of smoothing or dimensionality reduction. Removing

noise from the data can help to identify trends and seasonality.

3 Data Pre-processing

Time-series forecasting requires a slightly different data structure than a typical train/test k-fold cross-validation scheme. As time evolves, the true values of forecasted data become available and we can improve our current model by appending them as new training data. This process usually consists of several filtering and merging operations on the test and train data.

Guided by our findings in the previous section, we want to create a model for each department using data from all stores. This means that we need to reshape the sales data so that it suits our needs better. We perform the following pre-processing steps (in order) on the data:

1. Read the training data from "train_ini.csv".
2. Read the test data from "test.csv" and filter for the two months of interest - our window of prediction.
3. Remove *IsHoliday* column, add *Weekly_Pred* column and initialize to zero.
4. Find the unique set of values for *Dept*, *Store*, *Date* in the test data.
5. Create generic "train_frame" and "test_frame" that store information about a specific department.
6. For each department:
   a. filter training data for that department, store it in the "train_frame", and reshape the "train_frame" to get sales data for stores as columns.
   b. similarly, reshape "test_frame" to get sales data (initially zero) for stores as columns.
   c. if needed, reduce noise by using singular value decomposition (SVD) to obtain a rank-reduced approximation of reshaped "train_frame".
   d. fit model and forecast sales.
   e. add forecast for this department to the current window of prediction.
   f. repeat steps a-d for all departments.
7. Perform scoring based on weighted-mean-absolute-error (WMAE).
8. Load the true sales data from "fold_x.csv", append to the training data, and go to step 2. Repeat for x = 1, 2, …, 10.

Note that at step 6.c, we use the SVD method to remove some noise from the training data. Based on screening analysis, we found that using 8 to 10 components for SVD usually results in better average WMAE. This coincides nicely with the number of rows in the two-month prediction window (8 - 9), and we used this as the number of components for SVD of the training data to obtain a rank-reduced approximation of itself.

4 Model Development

We build three models to predict the weekly sales of all departments in 45 Walmart stores. The first model uses a naïve approach. The second model adds seasonality to the naïve model. The third model is a linear regression model. We have also investigated the effect of noise reduction by replacing the training data with a rank-reduced approximation of itself.

4.1 Naïve Models

Our first model uses a naïve approach and sets the last value as the sales forecast for a specified horizon (two months). We use the "naive" function from the "forecast" library in R to generate this model. Despite its simplicity, a naïve model provides a benchmark-like score that can be used to understand how well other improved methods are performing.

Our second model is based on a seasonal naïve approach that uses the sales data from the previous year for the specified prediction window. We use the "snaive" function from the "forecast" library in R to generate this model. This model is an improvement over the naïve model and makes better use of the training data.

Both models are very easy to build and run. No parameters are required to run these models. Using a rank-reduced approximation of the reshaped training data reduces the average WMAE for naive and snaive models by about 1.6% and 3.5%, respectively.

4.2 Regression Model

Our third model computes a forecast using linear regression and seasonal dummy variables. We use "tslm" function to fit a model that captures *trend* and *seasonality* in the training data. Then, we use the "forecast" function to predict *Weekly_Sales* of the test data for a specified horizon (two months). Using a rank-reduced approximation of the reshaped training data reduces the average WMAE for tslm model by about 2.5%.

5 Model Performance

We use the test data to predict the weekly sales for a two-month window for all departments of each store. These predictions from our model, described in the previous section, are compared against the true sales data provided in files called "fold_x.csv". We calculate weighted-mean-absolute error (WMAE) [3] for predicted sales of each fold and report our results.

TABLE 1 summarizes WMAE values for each model and each two-month window (fold). We can see that the naïve and snaïve methods produce larger WMAE compared to tslm method. Performing SVD and reconstructing the training data using a smaller number of components, results in a ~2.5% reduction in WMAE of tslm method. The last column (tslm with SVD) in the table produces a WMAE of *1611.366*, which is less than the benchmark value of **1630**.

TABLE 1 also provides the execution time of *each fold* in R. The run time of a fold is about 7.5 seconds for naïve and snaïve methods, whereas it jumps to about 32 seconds for tslm method. The cost of SVD is negligible. The platform and machine where these runs are performed can be seen at the bottom of the same table.

TABLE 1 Performance Summary on the Test Data

Fold ID	Weighted Mean Absolute Error (WMAE)			
	naive	snaive	tslm	tslm (SVD)
1	2078.726	2262.422	2042.401	1945.517
2	2589.338	1787.081	1440.083	1365.797
3	2253.936	1779.052	1434.716	1384.045
4	2823.098	1716.117	1596.988	1535.278
5	5156.012	2400.395	2327.638	2313.043
6	4218.348	1696.900	1674.185	1632.615
7	2269.904	2086.967	1718.577	1686.125
8	2143.839	1750.283	1420.817	1399.842
9	2221.145	1719.887	1430.801	1417.946
10	2372.425	1680.956	1447.034	1433.457
average	***2812.677***	***1888.006***	***1653.324***	***1611.366***
runtime (s)	*75.97*	*75.49*	*324.85*	*333.71*

Platform
- RStudio Version 1.3.1073 with R version 4.0.2 (2020-06-22)
- Windows 10 x64 build 18363, Intel® Core™ i7-2600 CPU @ 3.40GHz, 16.0 GB RAM

6 Conclusion

We documented an end-to-end workflow of predicting the weekly sales of departments in 45 Walmart stores located in different regions. Starting with an initial assessment of the data set, we pre-processed the data and made forecasts using three different models: naïve, snaïve, and tslm. Then, we performed predictions on the test data and reported WMAE values of *Weekly_Sales* – response variable.

We note that using the *forecast* library in R makes it very easy to work with time-series data. It houses many advanced methods to model time-series data. We explored a few basic methods that provided good scores and were adequate for our purposes. Additional adjustment of the training data (e.g., shifting days to compensate for changing holiday weekends) were not required to hit the benchmark for this project. However, a simple noise reduction approach using SVD improved the accuracy of predictions without adding significant computational cost.

Throughout this project, we had some challenges and converted them to learnings as we managed to resolve some. Working with time-series data was very interesting and fun.

7 References

[1] "Kaggle - Walmart Recruiting - Store Sales Forecasting," [Online]. Available: https://www.kaggle.com/c/walmart-recruiting-store-sales-forecasting/data. [Accessed 10 2020].

[2] "Walmart_competition_code: The Winner," [Online]. Available: https://github.com/davidthaler/Walmart_competition_code. [Accessed 10 2020].

[3] "Kaggle Competition: Evaluation Metric," [Online]. Available: https://www.kaggle.com/c/walmart-recruiting-store-sales-forecasting/overview/evaluation. [Accessed 10 2020].

8 Deliverables

This project has the following deliverables:

- **mymain.R**
  R code that contains the function *mypredict* and returns prediction for the next two months stored in a column named *Weekly_Pred*.
- **Project_2_Report_xxx.pdf**
  Report that provides technical details on data pre-processing, model development, and prediction accuracy with additional technical details.