Project 2: Walmart Store Sales Forecasting

CS598: Practical Statistical Learning

Naomi Bhagat - nbhagat3, Michael Miller - msmille3, Joe May - jemay3

11 November 2023

Assignment Data

Program: MCS-DS Assignment post: campuswire

Team contributions:

Person	Contribution
Naomi Bhagat	Report
Michael Miller	Algorithm
Joe May	Algorithm

Overview

Given historical sales data from 45 Walmart stores spread across different regions, our task was to predict the future weekly sales for every department in each store. The dataset is from https://www.kaggle.com/c/walmart-recruiting-store-sales-forecasting.

The measure we use for evaluation is the weighted mean absolute error (WMAE) between our predictions and the actual values of the weekly sales.

Section 1: Technical Details

Pre-Processing

Our function `process_fold` is the meat of the prediction algortihm. We first read the training and test data in the given file directory parameter `file_dir`.

We first clean the test data by introducing a new vairable `Wk` to numerically represent each week of the year, ranging from 1-52. We do the same thing with a new variable `Yr` to represent the year of the already existing `Date` in thetest dataset.

We then initialize an output matrix, a counter variable, and the a variable to keep track of the number of departments in the data. In order to speed up our processing speed, a variable called `dept_to_eval` is created to keep track of departments that exist in both the test data and the training data.

Prediction

From here, we perform the same steps for each department stored in the `dept_to_eval` variable. Our first call is to a custom function called `spread_df`, which takes the parameters of the training data and the current department being evaluated. In the `spread_df` function, we start with a variable `X` which takes the training data and filters out any departments that are not the current department, as well as keeps the relevant data from the training data, including `Store`, `Date`, and `Weekly_Sales`. Finally, we use the `spread` function from the `tidyr` package to expand the training data to include top-level columns for `Store` and `Weekly_Sales`. After replacing all the `NA` values with 0 in `X`, we drop the `Date` row and take the transpose of `X`, and return a list consisting of X, stores, and dates.

Now, we have acquired the relevant data to begin the prediction. We check if the dataset stored in `X` is large enough to perform SVD by checking the dimensions of `X` against 8, the minimum number of dimensions we use for SVD. The SVD process keeps dimensions the same but reduces noise and populates values that would otherwise be `NA`. SVD is implemented with standard methodology. If SVD is not required, we continue the rpediction algorithm using `X`.

We then use one more helper function called `gather_mat` that takes in parameters of `X`, the full returned list value of the `spread_df` funtion call, and the current department. This function simply pivots the `X` matrix back into the original format of the given test and train dataframes. Calling this function then gives us the final, smoothed training data for the current department.

Now, for each unique store associated with the current department, we filter the newly smoothed training data and the testing data by the current store, and convert this data to a design matrix to be used for model training. To train the model, we use the linear model `lm`, the `Weekly_Sales` from the filtered training data, and the design matrix created from the filtered trainign data. The coefficients are extracted from the model, and any `NA` values are zeroed out.

Model Evaluation

We then evaluate the model using the model's coefficients and the testing data.

Post-Processing

To handle the edge case in fold 5 - Christmas falling earlier in the week - we pivot some sales, removing the "too high" value from its current week and shifting it to be counted in the week after. We can then join the results for the current department-store combination into the output dataframe with all the `Weekly_Pred` values stored in the same place. Finally, we write the predictions to `mypred.csv`.

Evaluation

Our evaluation function `myeval` uses the created file `mypred.csv` from the `process_fold` function call to calculate WMAE per fold. For each fold, the prediction values are read, at which point the `Weekly_Sales` and `Weekly_Pred` are read. Then, weights are added to holiday datapoints, and finally, WMAE is calculated with the formula `sum(weights * abs(Weekly_Sales - Weekly_Pred)) / sum(weights)`.

Section 2: Performance Metrics

The computer system this was run on was a Dell Inspiron 3501, 1.00 GHz with 8.00 GB of installed RAM for all 10 training/test splits. Below is a summary of the fold #, the WMAE for each fold, and the time it took to train the models:

Fold	WMAE	Time (s)
1	1941.586	69.40
2	1363.507	67.85
3	1382.469	69.29
4	1527.293	80.14
5	2156.606	80.31
6	1634.892	78.32
7	1613.908	70.70
8	1355.016	68.13
9	1336.911	71.56
10	1334.010	73.67

The average WMAE over all folds is 1564.62.