Project 2 Report: Walmart Store Sales Forecasting

Section 1: Technical Details

For data preprocessing, I first implemented SVD to reduce the noise in the training dataset for better error scores. We did this noise reduction per department. First, I computed the entire pivoted/spread X matrix, with the rows being combinations of store & department and the columns are individual dates with their corresponding Iekly sales for all the dates within the fold. Next, I filled any missing values with zero, these missing values occurred when a store department combination did not have a Iekly sale number for that date. Next for each department, I filtered the rows from this X matrix to just include sales from that department, then using the numpy linear algebra library I ran SVD decomposition on the Iekly sale values only. Then I kept only the top 8 diagonal components as this was the number recommended by the CampusWire post. Finally, I melt/gather this matrix back into the original format where the dates are made to be one column, finally after running SVD for each department, I combine the denoised data to get the final new training dataset.

Following this, I filtered the training dataset so that only department-store pairs that appear in both the training and testing data are included in the modeling process. In the preprocessing stage, I extracted *Wk* and *Yr* from the *Date* column in both the training and testing data. We set *Wk* as a categorical variable with 52 levels and added *Yr* as a separate column. After that, I created a design matrix that included *Store*, *Dept*, *Wk*, *Yr*, and *Yr^2* variables. Then, I grouped the data by each unique *Store* and *Dept* combination, storing each pair as a key in a dictionary along with its related data. To handle missing values and non-full rank matrices, I adjusted the design matrices to include all 52 weeks. When the training data did not cover all 52 weeks, the design matrix could become a non-full rank, which could cause issues in estimating coefficients. We identified and removed redundant columns, starting from the last column and moving backward, to ensure that each column was independent and couldn't be recreated from others.

We further fitted an Ordinary Least Squares (OLS) model for each unique department-store pair using denoised and pre-processed training data. In the end, I filled in missing predictions with zeros to provide a complete set of forecasts. Since I did not reach the desired average WAE, I have decided to implement post-adjustment to fold 5 because it had the highest individual WAE. For that, I adjusted the Iekly sales predictions for each department during the holiday season to better capture seasonal surges. First, I calculated a baseline sales level by averaging the predictions for Ieks 48 and 52, and a surge level by averaging the predictions for Ieks 49, 50, and 51. If the surge was at least 10% higher than the baseline, I applied a circular shift of 1/7 to redistribute a portion of the Iekly sales within this holiday period.

Section 2: Performance Metrics

The models were trained & tested on the Google Colab Python3 runtime using the CPU Hardware accelerator/default setting. Based on a lb search it is an Intel Xeon CPU @ 2.3 Ghz with 13 GB of RAM. For the first iteration, fitting a linear regression model, I got an average WAE of around 1658, then adding SVD to this got the error to 1613, by adding the Yr^2 variable, our error got to 1585. Finally, adding the post-processing correction to fold 5 gave us a final Average WAE of 1559.998. Here are the results of our final model with runtimes:

```
Starting Fold 1 ...
Fold 1 Runtime: 106.244 seconds
------------------------------------------------------------
Starting Fold 2 ...
Fold 2 Runtime: 92.857 seconds
------------------------------------------------------------
Starting Fold 3 ...
Fold 3 Runtime: 98.595 seconds
------------------------------------------------------------
Starting Fold 4 ...
Fold 4 Runtime: 100.491 seconds
------------------------------------------------------------
Starting Fold 5 ...
Fold 5 Runtime: 101.230 seconds
------------------------------------------------------------
Starting Fold 6 ...
Fold 6 Runtime: 103.099 seconds
------------------------------------------------------------
Starting Fold 7 ...
Fold 7 Runtime: 104.504 seconds
------------------------------------------------------------
Starting Fold 8 ...
Fold 8 Runtime: 108.948 seconds
------------------------------------------------------------
Starting Fold 9 ...
Fold 9 Runtime: 110.057 seconds
------------------------------------------------------------
Starting Fold 10 ...
Fold 10 Runtime: 107.373 seconds
------------------------------------------------------------
```

```
Fold 1 WAE: 1943.344
Fold 2 WAE: 1390.886
Fold 3 WAE: 1392.232
Fold 4 WAE: 1523.190
Fold 5 WAE: 2053.004
Fold 6 WAE: 1636.824
Fold 7 WAE: 1615.023
Fold 8 WAE: 1362.546
Fold 9 WAE: 1350.826
Fold 10 WAE: 1332.109
Average WAE: 1559.998
```