Predictive Analytics on Walmart's Weekly Sales Data

CS 584 Machine Learning Project Report

Kyung Jin Kwak
Ritu Tushar Bagul

Introduction

This report presents a detailed analysis of Walmart's sales data with the aim of predicting weekly sales. The significance of accurate sales forecasting for retail operations is well-established, with direct impacts on inventory management, staffing, and financial planning. By exploring a comprehensive dataset that includes a range of features from various Walmart stores and departments, this project seeks to not only forecast sales but also identify the key factors influencing sales outcomes. The potential business implications of these insights could contribute to more informed decision-making and strategic planning within the retail sector.

The project can be found in the GitHub repository:

https://github.com/Ritu2807-creator/Walmart-Sales-Prediction

Data Exploratory Analysis (EDA)

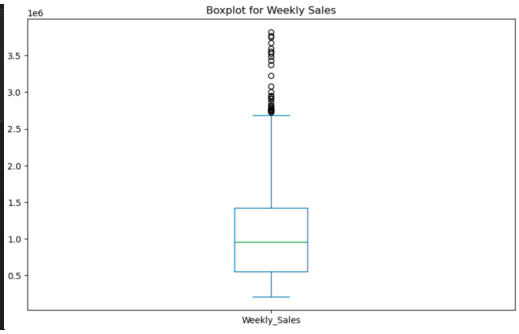
The dataset comprises records from multiple Walmart stores, encompassing features such as department numbers, weekly sales, store size, and type. Initial analysis revealed certain features that exhibited strong correlations with weekly sales, which prompted further investigation. Through the application of preprocessing techniques, data integrity was ensured, enabling a more accurate analysis.

- Data preprocessing (change data type, find missing values, outliers, etc.)

```
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Store         6435 non-null   int64
 1   Date          6435 non-null   object
 2   Weekly_Sales  6435 non-null   float64
 3   Holiday_Flag  6435 non-null   int64
 4   Temperature   6435 non-null   float64
 5   Fuel_Price    6435 non-null   float64
 6   CPI           6435 non-null   float64
 7   Unemployment  6435 non-null   float64
dtypes: float64(5), int64(2), object(1)
```

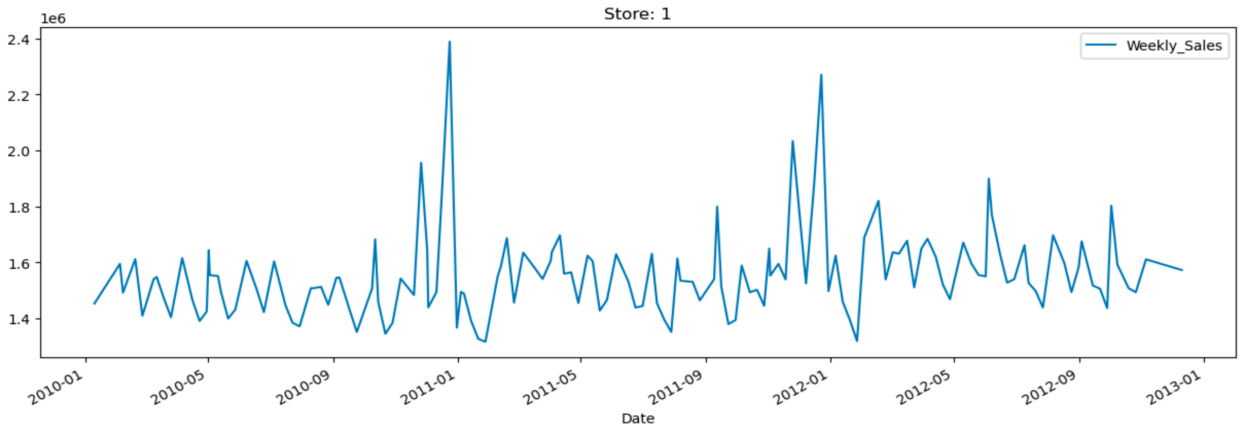
```
# Check for missing values
missing_values = data.isnull().sum()
print(missing_values)
```

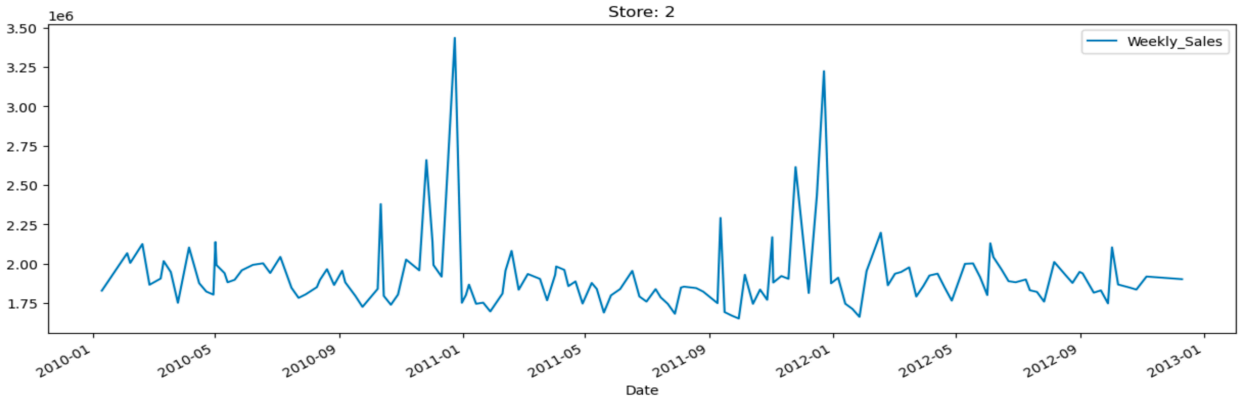
```
Store           0
Date            0
Weekly_Sales    0
Holiday_Flag    0
Temperature     0
Fuel_Price      0
CPI             0
Unemployment    0
dtype: int64
```



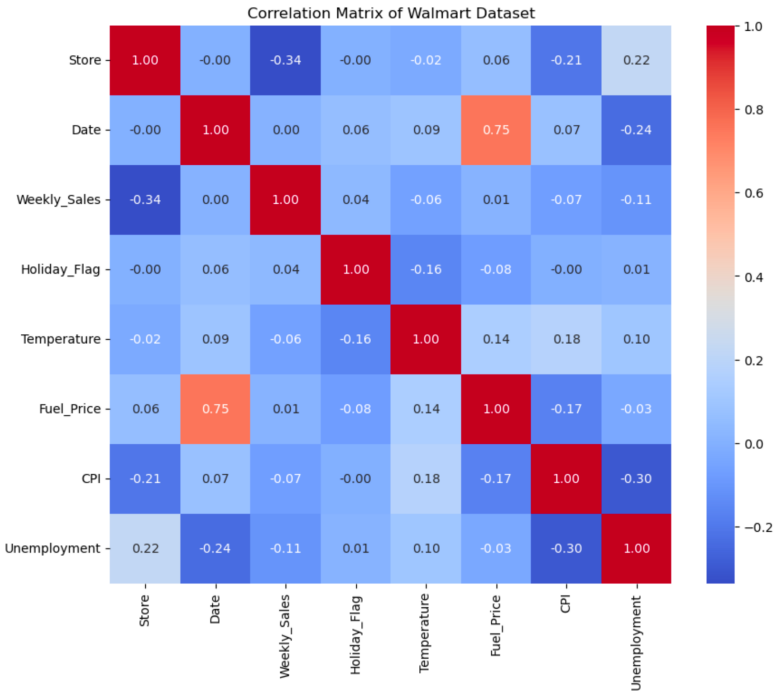
- As seen in the above figure, data type of column name 'Date' is object, so we changed its data type to datetime64.
- There was no missing values in the dataset.
- There are outliers of weekly sales value because it shows significant growth when it's holiday season in November through December. We did not delete any outliers because our target variable is sale which shows seasonal trend, so that we can have better predicting model out of this data.

- Sales trend by stores (45 stores)
- Below is the weekly sales trend by each stores. As seen in the plot, sales go up rapidly when it's holiday season. Every 45 stores show almost the same trend. There are trend plots of two stores shown as samples below.





- Correlation matrix

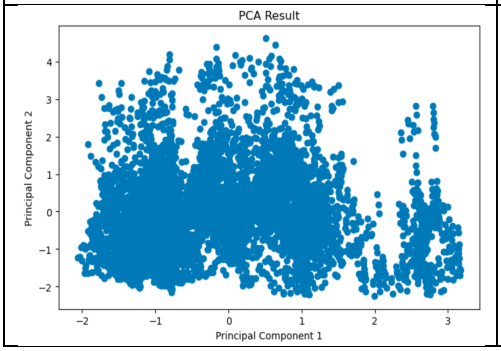
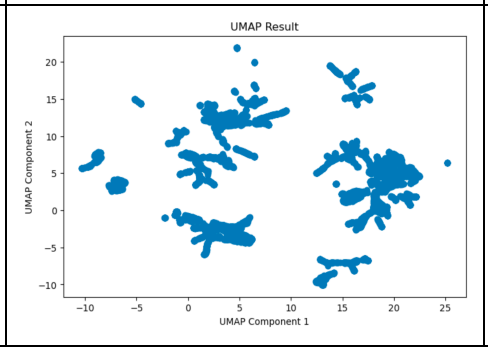
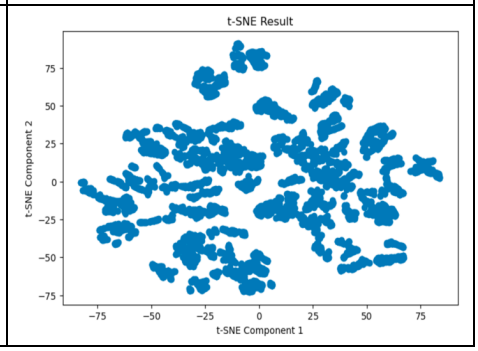


- Some Important observations are:
  - Weekly_Sales and Store: There is a moderate negative correlation, suggesting that sales may differ across stores, and certain stores may consistently have higher or lower sales.
  - Fuel_Price and Date: There is a strong positive correlation, which may indicate that over time, fuel prices have been increasing.

- CPI and Unemployment: There is a moderate negative correlation, indicating that as CPI increases, unemployment tends to decrease, or vice versa. This could be due to economic growth leading to increased consumer prices and reduced unemployment.
- Holiday_Flag doesn't seem to have a strong correlation with Weekly_Sales: This suggests that holidays might not have a significant impact on sales, which could be counterintuitive and may warrant further investigation.

Visualization and Dimensionality Reduction

Dimensionality reduction techniques, including PCA, UMAP, and t-SNE, were utilized to simplify the high-dimensional data into a more manageable form, revealing the most impactful features on sales. The visualizations from PCA, t-SNE, and UMAP each provide a different perspective on the structure of your high-dimensional data by projecting it onto two dimensions.

PCA	UMAP	t-SNE
		

- The PCA result shows the data spread out mostly along the first principal component with some spread along the second, this might suggest that there is one main feature driving the variance in the dataset with a secondary feature providing some additional structure.
- In terms of clustering, looking at these three results, the clustering is not very clear. UMAP is showing less distinct than t-SNE, but since we have relatively small number of predictive variable, we decided not to delete any variables to proceed to select predictive model.

Identifying appropriate cross-validation strategy

There are several CV techniques, like K-fold cross validation, Stratified K-fold cross validation, Time series cross validation, Leave one out Cross validation. Here, since we are predicting sales with weekly data, we thought that it would be the best to consider Time series cross validation, but to do double check, we also ran random forest.

- Model1 : Random Forest
- We tried to run a very simple random forest model and you can see the results are bad. So, we also tried to do a little hyper parameter tuning to check if we can improve the results.

1st attempt

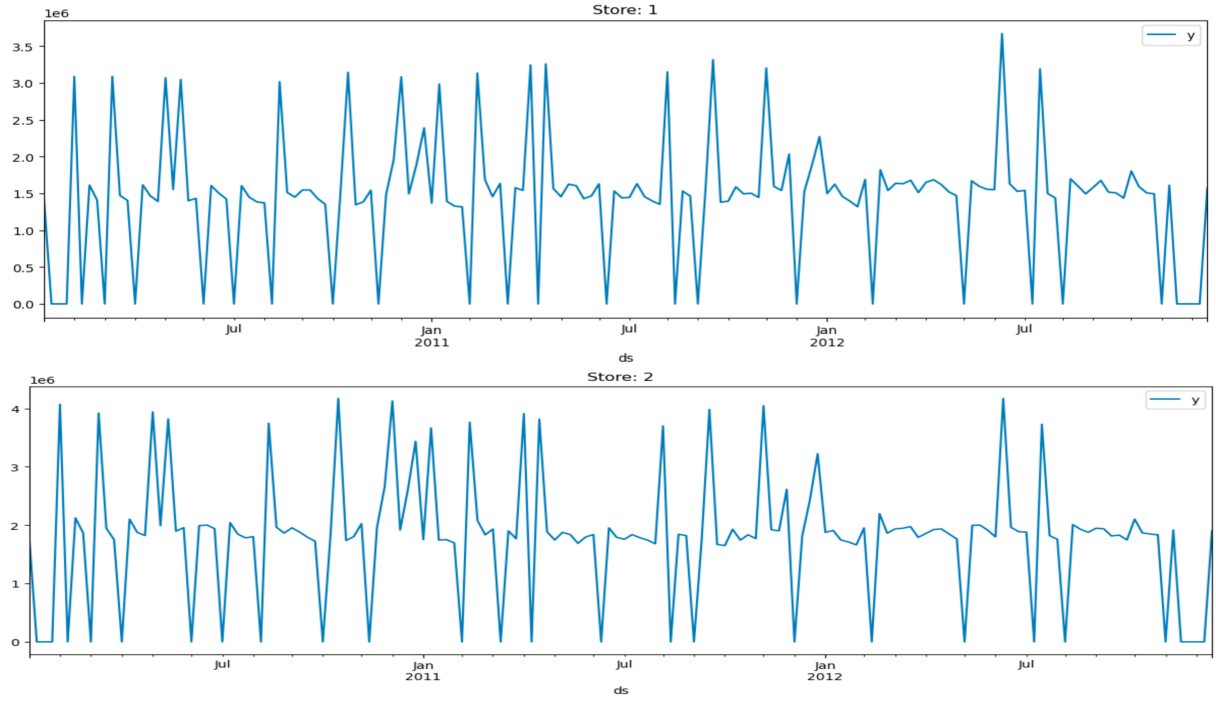
```
MSE for the split: 22439462758.18324
MSE for the split: 76280376.18990317
MSE for the split: 28201014.679586668
MSE for the split: 13179667.982770158
MSE for the split: 71518690.91337529
Average MSE across all splits: 4525728501.589775
CPU times: user 11.6 s, sys: 19.9 ms, total: 11.6 s
Wall time: 11.7 s
```

Hyper parameter tuning

```
Final MSE: 313232.95075125364
Final RMSE: 559.6721815056146
Final MAE: 284.69392479955144
```

- Model 2 : NeuralProphet

- Since we have seen some seasonality and trends in the plots in the beginning, we decided to go with NeuralProphet. For neural prophet to work, we need to resample the time series data because, the NP model only works with single time frequency. In our case, we resampled to weekly basis.
- After resampling the data, we can see on the below plots from 2 stores as sample, there are no holiday spikes, and the data is more consistent.



- Model training
  1) 1st Tuning condition :
     - params = {"seasonality_mode": "multiplicative", "learning_rate": 0.01}
     - NeuralProphet(**params).crossvalidation_split_df(df_weekly, freq="W", k=5, fold_pct=0.20, fold_overlap_pct=0.5)
     - Result :

```
# First model
metrics_train
```

	MAE	RMSE
0	0.199621	0.273613
1	0.194459	0.273073
2	0.183624	0.270298
3	0.182612	0.257026
4	0.174397	0.252152

```
metrics_test
```

	MAE_val	RMSE_val
0	0.385272	0.444531
1	0.168188	0.252842
2	0.116386	0.195330
3	0.190908	0.274038
4	0.287731	0.324750

  2) 2nd Tuning condition :
     - params = {"seasonality_mode": "additive", "learning_rate": 0.001, 'yearly_seasonality': True}
     - NeuralProphet(**params).crossvalidation_split_df(df_weekly, freq="W", k=5, fold_pct=0.20, fold_overlap_pct=0.5)
     - Result :

```
#second model
metrics_train2
```

	MAE	RMSE
0	0.198868	0.264801
1	0.195934	0.267269
2	0.189210	0.264499
3	0.185695	0.263347
4	0.175883	0.255940

```
metrics_test2
```

	MAE_val	RMSE_val
0	0.694734	0.784560
1	0.230999	0.299187
2	0.171206	0.256511
3	0.208117	0.296356
4	0.254742	0.299092

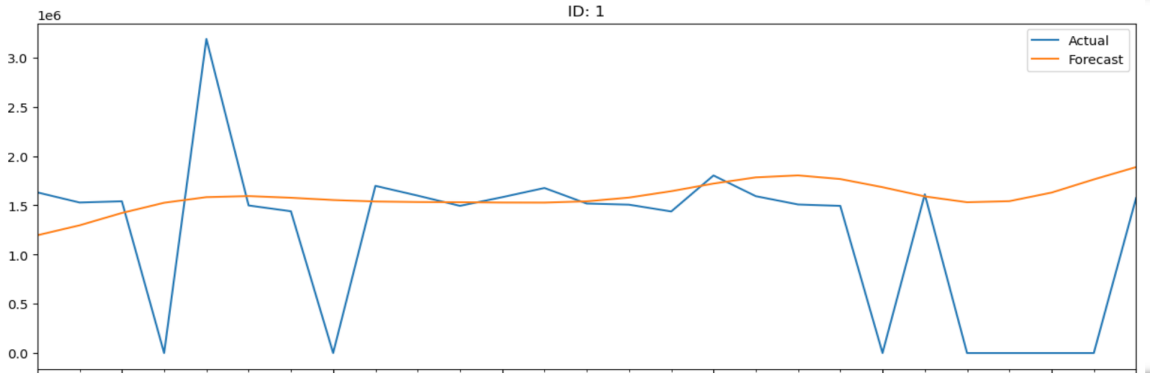
- Looking at the result of NeuralProphet and conducting two different tuning models, second tuning model performed well compared to the first tuning model. It might be because since we gave lower learning rate and also, we added yearly_seasonality component to the model which might have helped in understanding the seasonality of the data well, because from the plots above where we saw for sales trend per stores, we can see that there is high peak during December compared to other months.

Prediction

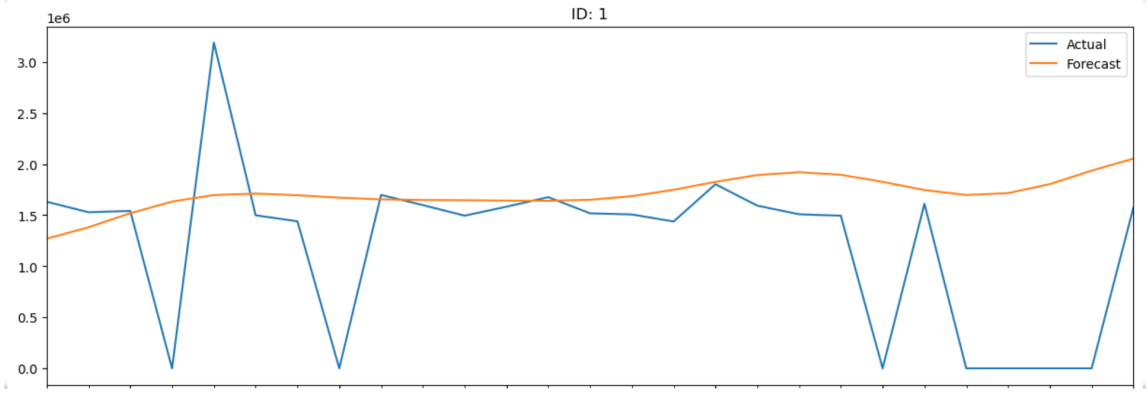
When we did the prediction in the model that we selected, we can see from the below results, the forecast line is pretty good but it did not grasp the trend points accurately. So we first gave trend components parameter and train the model, secondly calculated for the auto regressor value, and considered there are 10 lags because it's a time series dataset, we did not improve efficiency, it instead decreased the efficiency or first 10 points were missing as you can see in the plot below.

- As you can observe below plots, despite our efforts to increase model efficiency, 1st attempt has given best result.

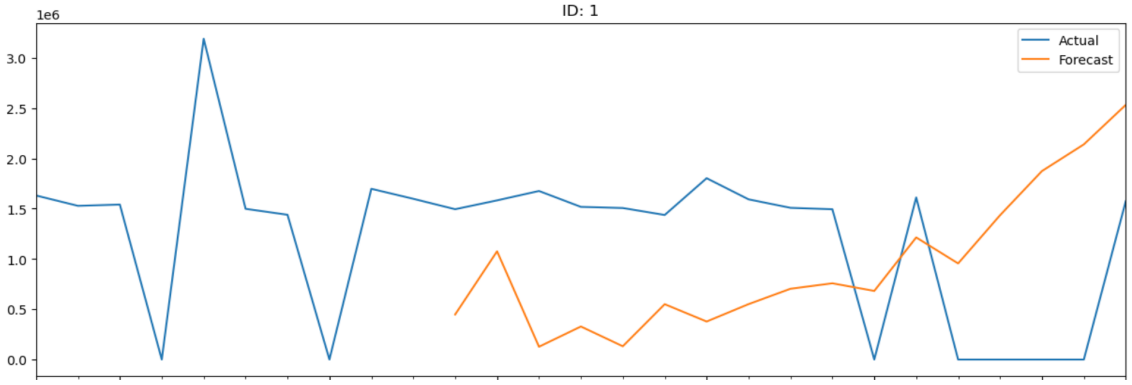
1st attempt (one store as sample)



2nd attempt (one store as sample)



3rd attempt (one store as sample)



Topic Review

Until now, we have done :

- Try to visualize data and do dimensionality reduction techniques such as PCA, UMAP, t-SNE.
- Use Time series Cross validation (CV) strategy.
- Use Random Forest (RF) model as a simple basic starting step of our project.
- Use CV for RF and also Fine tune RF.
- Use Neural Prophet (NP) model as our advanced model in this project.
- Use CV for NP and also Fine tune NP.
- Propose ways to improve NP model efficiency.
- Conducted further 2 more experiments to test results.

Future enhancements

To further fine tune the model, we can make use of HyperOPT. Using Hyperopt would be a more efficient and sophisticated method for hyperparameter tuning compared to a manual search. Hyperopt uses Bayesian optimization for hyperparameter tuning, which means it tries to find the best hyperparameters based on the results of previous iterations, rather than exhaustively searching the entire space. This usually results in finding better hyperparameters with fewer iterations.

But because of computer limitations, we will not be doing that for now and leave it for further studies.