Prediction of Weekly Sales based on Economic Conditions

Chulhee Kim, Tingting Lin, Vatim Vanak, Yan Zhang

December 2022

1 Introduction

Foot traffic, conversion rate, and total sales during a period of time may be considered to be important indicators of store performance. Forecasting them may allow for business managers plan stores operation in the near future in an efficient way [2]. Data science offers wide range of methods for forecasting based on historical data, but each method can show different performance based on specific features of source data. Therefore, it is critical to find and apply appropriate method based on data. In this work we will analyze historical data that covers sales data for 45 stores of Walmart from 2010-02-05 to 2012-11-01. The dataset is public dataset available in [3].

We will try to apply various methods for data analysis and forecast, compare their performance and find the best method.

2 Basic description of the data

This section describes the dataset used for the analysis.

- Name of the dataset: Walmart_Store_sales.csv
- Shape: 6,435 rows / 8 columns
- Detail column information can be found in table 1 below.
- The following dates are marked as holiday weeks:
  - Super Bowl: 12-Feb-10, 11-Feb-11, 10-Feb-12
  - Labor Day: 10-Sep-10, 9-Sep-11, 7-Sep-12
  - Thanksgiving: 26-Nov-10, 25-Nov-11
  - Christmas: 31-Dec-10, 30-Dec-11
- No missing values in the whole dataset
- Weekly_Sales is the response variable, and the rest of the columns are the features
- Numerical features have different scales, and CPI is the largest.

3 Data exploration

- Since the dataset does not contain missing data and there is no apparent outliers there is no need for dropping or imputing data.
- Date is in dd-mm-yyyy format, so converting it to datetime object would be better for our analysis.
- With grouping by Store and aggregating on each column, we can gain some insights:

Column	Description	Data type	Min	Max
Store	store number	Nominal (int)	1	45
Date	week of sales	Date (Object)	02-05-2010	10-26-2012
Weekly_Sales	sales for the given store	Continuous (float)	209,986.25	3,818,686.45
Holiday_Flag	1: holiday week 0: non-holiday week	Nominal (int)	0	1
Temperature	temperature on the day of sale	Continuous (float)	-2.06	100.14
Fuel_Price	cost of fuel in the region	Continuous (float)	2.47	4.47
CPI	CPI in the region	Continuous (float)	126.06	227.23
Unemployment	unemployment rate in the region	Continuous (float)	3.88	14.31

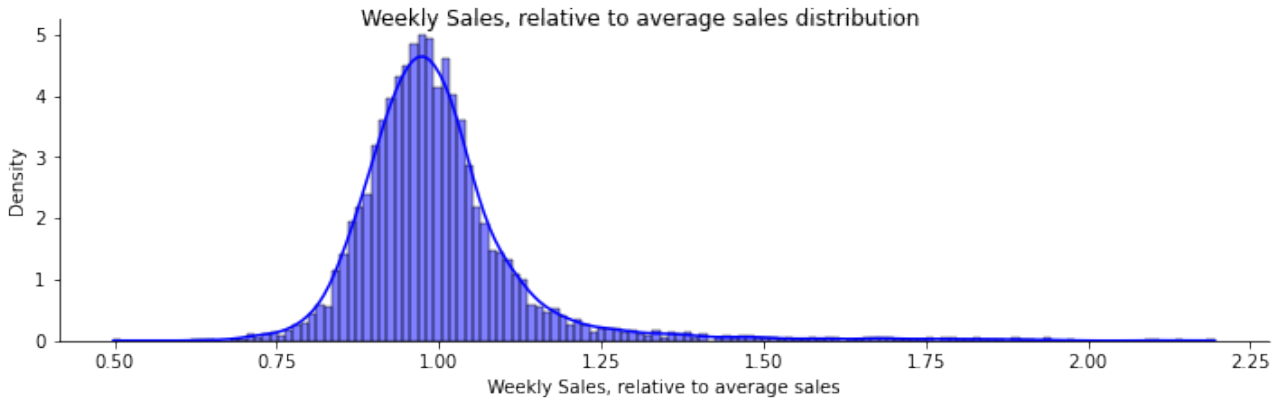
Table 1: Columns of the dataset

  - For every store, Date is from 2010-02-05 to 2012-10-26. We may need to be aware of these dates for our analysis (e.g. comparing yearly sales or monthly sales).
  - The mean of Weekly_Sales for each store varies a lot.
  - Date, Holiday_Flag are the same for all stores, and Fuel_Price is also quite similar.
  - Weekly_Sales, Temperature, CPI, Unemployment vary a lot across stores.
- We will sum Weekly_Sales by Date and examine any notable trend.
- The relationship between the response variable, Weekly_Sales, and each of the features will be explored by showing scatterplots and correlation matrix.
- We will explore whether certain dates, days, or months have any influence on sales. If there is significant impact, we may have to create separate columns for these.
- There may be general trends over time for each of the feature columns, so we can plot each of them over the whole dates.

4 Visualizations and captions

4.1 Descriptive statistics and data distribution

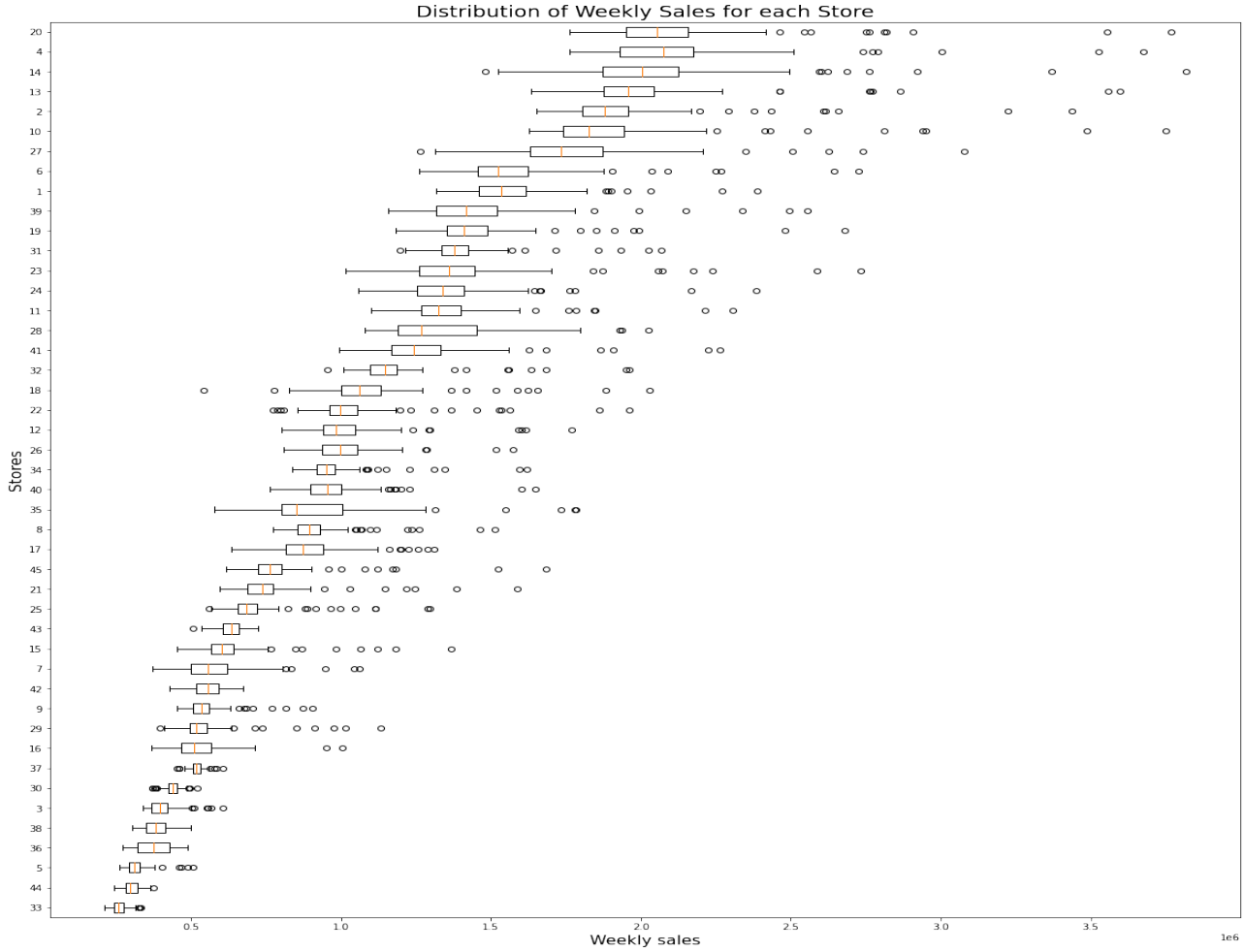
Weekly sales, as % of mean sales value shown on the chart below:



On average weekly sales have distribution which resembles a normal, but it has fat tails.

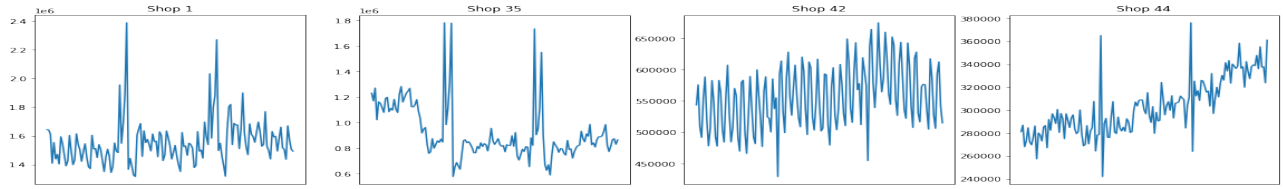
Distribution of weekly sales per shop is given on the chart on the next page.

Average weekly sales varies from about 250 thousand dollars to about 2 millions. There are outliers in weekly sales almost for all shops in the dataset.



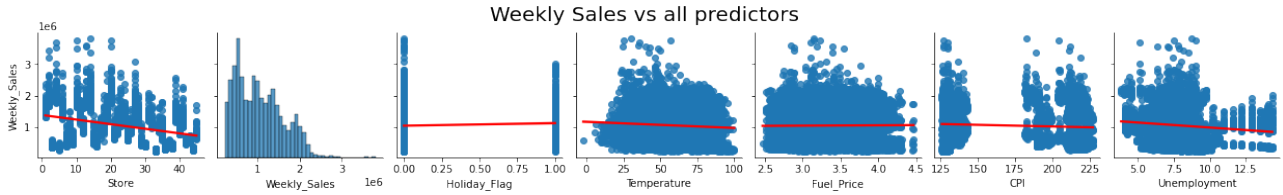
4.2 Temporal trends in data

While majority of stores show similar trends, some stores show completely different trends. Such examples are given below:



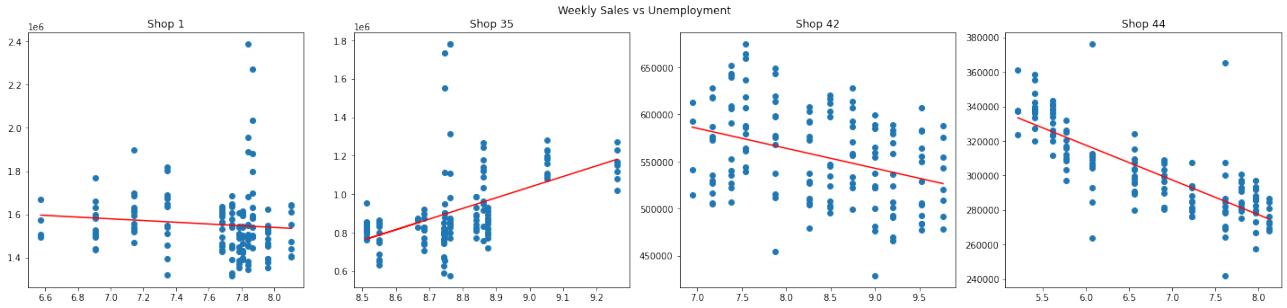
4.3 Correlation analysis

Scatterplots, Weekly_Sales vs all features are shown below:



There is no strong correlation between Weekly_Sales and any other feature, but some trend, for example between Weekly_Sales and Unemployment can be noticed.

If we plot correlation plot for individual shops, we can find that trend varies a lot across different stores.



5 Project question

In this project we will investigate the following question: "Which algorithm is the best for prediction of weekly sales based on economic conditions including CPI, Unemployment Index, etc."

To answer this question we will start with linear regression as our baseline estimation method. Since linear regression has number of assumptions under which it performs the best, we will analyze if these assumptions are met. If not, we will apply alternative estimation techniques, (for example Time series Analysis, Random forest, Boosting) to see if we can achieve more accurate results. We will evaluate accuracy of each method and select the best approach for predicting the sales.

6 Baseline Model/Linear Regression

As a baseline we will use simple Linear Regression Model including all the features with first degree, without scaling. Since store number is a categorical variable, we will use one-hot encoding for this variable. It is reasonable to expect that independent variables affect dependent variable in a multiplicative way. For example, sales on holidays will increase not by fixed amount in comparison of sales on workday, but multiplied by certain factor. Therefore, we will make log-transformation of weekly sales value. As a result, our baseline model will be in the form:

$$\log(S_t) = \sum_{i=1}^{N} \beta_i e_i + \alpha_1 H_t + \alpha_2 F_t + \alpha_3 T_i + \alpha_4 C_i + \alpha_5 U_i$$

where S_t-weekly sales at date t, e_i – one-hot encoding of shop number, H_t, F_t, T_i, C_i, U_i – are holiday flag, fuel price, temperature, CPI and Unemployment at date t correspondingly and β_i, α_i – model coefficients.

6.1 One-hot encoding

To implement one-hot encoding we are using get_dummies method of pandas.

Note that one-hot encoding with 45 stores creates problems with validation. Since flag for certain store presents only in $1/45 \approx 2,22\%$ of data, and the dataset is sorted by store number, there are high chances that cross validation leaves one store out completely in training set. Such set has problem with multicollinearity and generates extremely large MSE, with absolute values over 10^{11}.

To avoid this problem we used 10-fold cross validation with shuffling. Fitting and cross-validating baseline model with 10-fold cross-validation gives the following results:

Mean cross-validation MSE	Standard deviation for cross-validation MSE
0.014862	0.001485

Table 2: Performance of baseline model

For comparison with other models we calculated RMSE for original values (i.e. without log-transformation) from training and test datasets:

Train RMSE	Test RMSE
157660.393005	163569.59395

Table 3: Performance of the baseline model

6.2 Alternative models

Charts above clearly shows that sales very much depends on week number, having big spike before Christmas and significant decline after it. To keep information about date, we have considered alternative model with added week number using one-hot encoding to the regressors list. In addition to baseline model we have evaluated the following sets of models:

1. Models with eliminated regressor (one model for each regressor).
2. Model with only one-hot encoded Store and Holiday_Flag used as regressors
3. Models with squares of the following regressors – Temperature, Fuel_Price, CPI, Unemployment
4. Model with interaction term between all regressors mentioned above

Models from groups 1 and 2 helps us to understand if we have insignificant regressors. Models from groups 3 and 4 can help us to find if there is non-linear dependency between regressors and weekly sales.

The results are shown in the table 4.

Model	MSE	Standard deviation
Baseline model	0.014862	0.001485
Added week	0.007426	0.000749
Eliminated Holiday_Flag	0.007426	0.000749
Eliminated Temperature	0.007511	0.000716
Eliminated Fuel_Price	0.007440	0.000757
Eliminated CPI	0.007427	0.000748
Eliminated Unemployment	0.007666	0.000748
Eliminated Store	0.339037	0.015000
Added square Temperature	0.007315	0.000769
Added square Fuel_Price	0.007426	0.000752
Added square CPI	0.007428	0.000751
Added square Unemployment	0.007422	0.000740
Added interaction Holiday_Flag,Temperature	0.007437	0.000750
Added interaction Holiday_Flag,Fuel_Price	0.007422	0.000743
Added interaction Holiday_Flag,CPI	0.007437	0.000747
Added interaction Holiday_Flag,Unemployment	0.007432	0.000746
Added interaction Temperature,Fuel_Price	0.007427	0.000747
Added interaction Temperature,CPI	0.007410	0.000757
Added interaction Temperature,Unemployment	0.007407	0.000750
Added interaction Fuel_Price,CPI	0.007423	0.000753
Added interaction Fuel_Price,Unemployment	0.007410	0.000737

Model	MSE	Standard deviation
Added interaction CPI,Unemployment	0.007427	0.000748
Store only	0.007719	0.000745

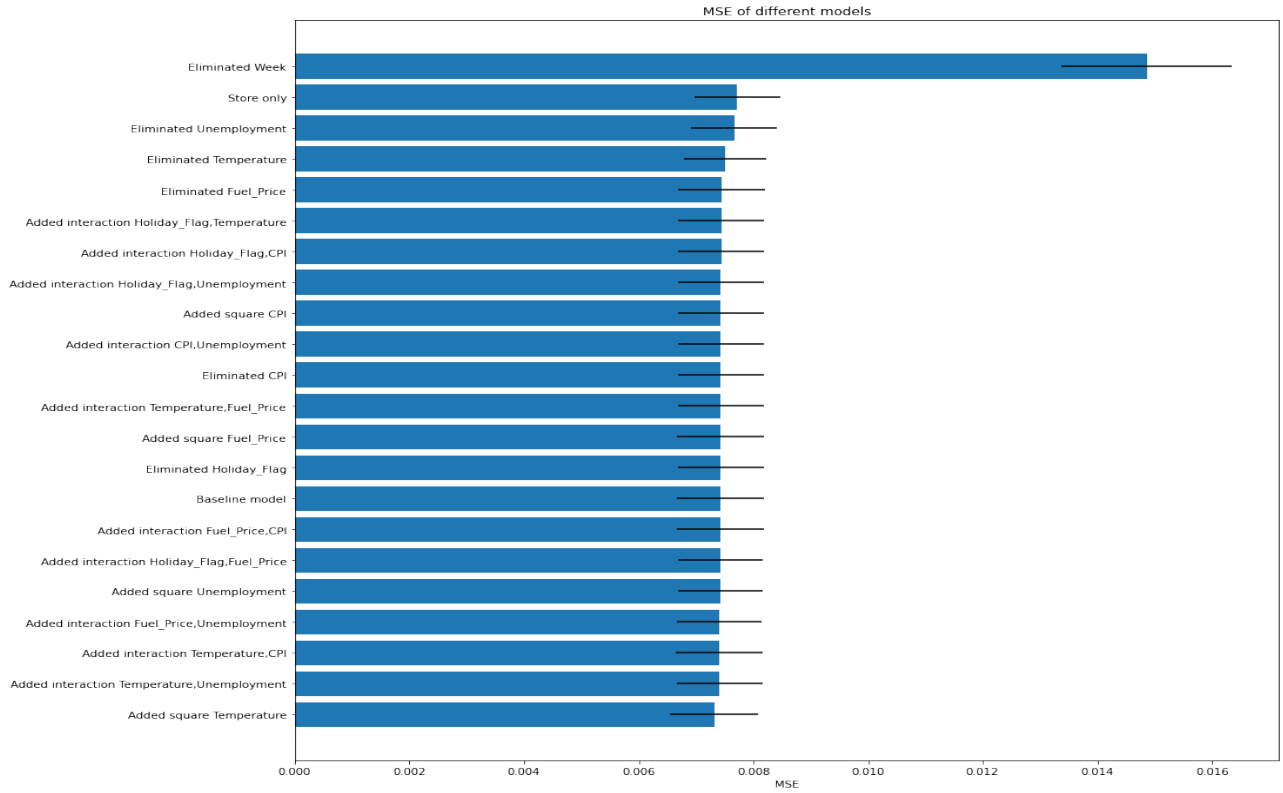
Table 4: MSE of different linear models

Elimination of Store significantly decrease model precision. We have excluded this model from further consideration. Visualization of the results available in the chart below.

Elimination of Week also decrease model precision significantly.

There is very little differences between other models. Based on these results we can say that:

1. Very simple model with Store and Holiday_Flag gives results which only marginally gets worse than results by models with more regressors
2. Adding second degrees of regressors and interaction terms does not make any significant improvement in the model.



Based on the results above we selected baseline model with one-hot encoded week as model which provides the best results. For comparison with other models we calculated RMSE for original values (i.e. without log-transformation) from training and test datasets:

Train RMSE	Test RMSE
86048.049244	86641.836049

Table 5: Performance of best linear regression model

According to Gauss-Markov theorem ordinary least squares estimators are BLUE (best linear unbiased estimators) which means that they cannot be improved if all linear regression assumptions are met.

We will check linear regression assumptions to understand if further improvements possible.

6.3 Regression assumptions check

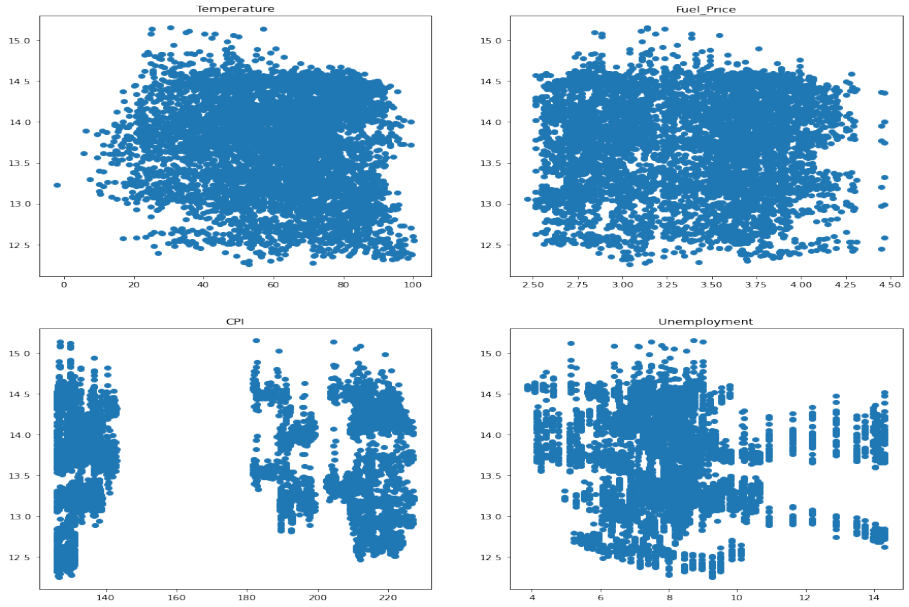
6.3.1 Linear relationship between expected value and predictors

One of the most important assumptions of linear regression is linearity.

All estimations for linear model based on this assumption. To validate it we plot logarithm of weekly sales versus each of regressors. Chart is available below.

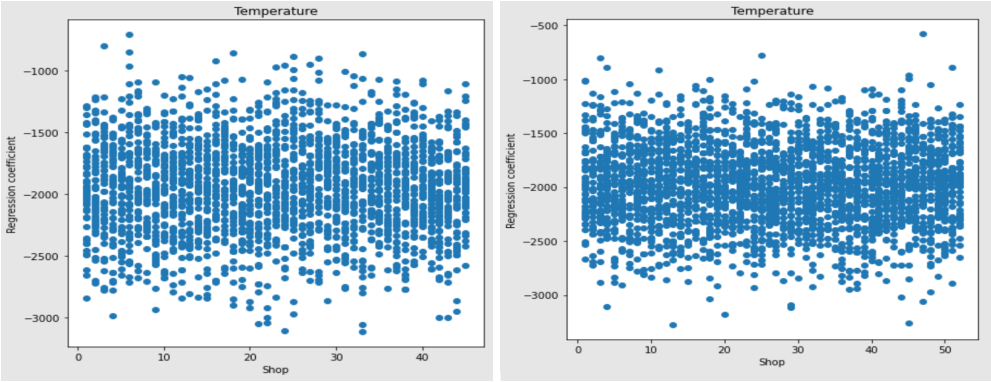
There is no apparent deviation from linearity assumption. Besides, according to table 3 adding square and interaction terms didn't change model results significantly, which also supports hypothesis that dependency is linear.

In case of dummy variables (Store, Week) there could be no non-linear dependency, since they take only value 0 and 1. However, there could be interaction between these variables and continuous regressors. If it is the case, regression coefficients for regressors will be different for different shops.



We have estimated regression coefficients for each combination of (shop,regressor) using bootstraping. In case of interaction the result is dependent on shop number.

The chart below plots regression coefficient for Weekly_Sales versus temperature for each shop and for each weak correspondingly obtained with bootstraping procedure.



Same picture is observed for all other regressors.

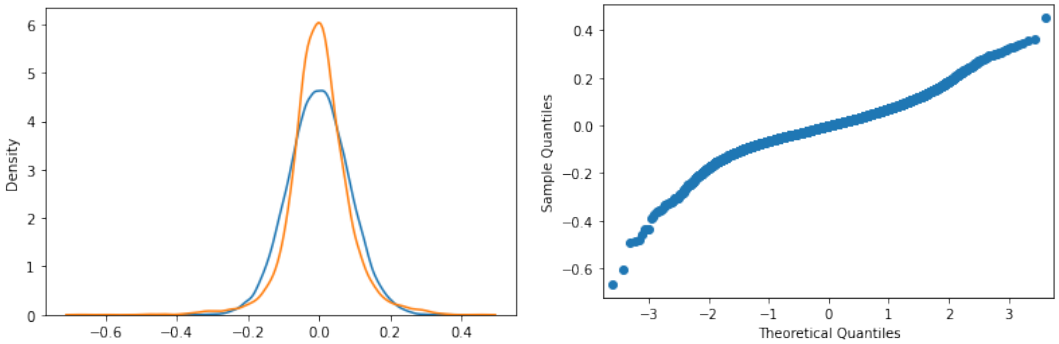
There is no visible dependency in regression coefficients for different shops.

We have not found deviations from linearity in our model.

6.3.2 Normality: Residuals are normally distributed about expected value

Normality is not required by Gauss-Markov theorem and estimation of ordinary least square method remain BLUE even if residuals are not distributed normally. However, if normality assumption is violated, confidence interval estimation can become biased or inconsistent.

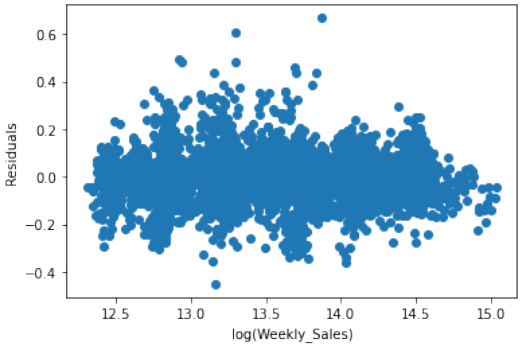
To check normality we used density plot and Q-Q plot:



There is very clear deviation from normal distribution in residuals.

6.3.3 Constant variance (Homoscedasticity)

Homoscedasticity variance of residuals is constant for all observations. To check it we plotted residuals versus logarithm of weekly sales:

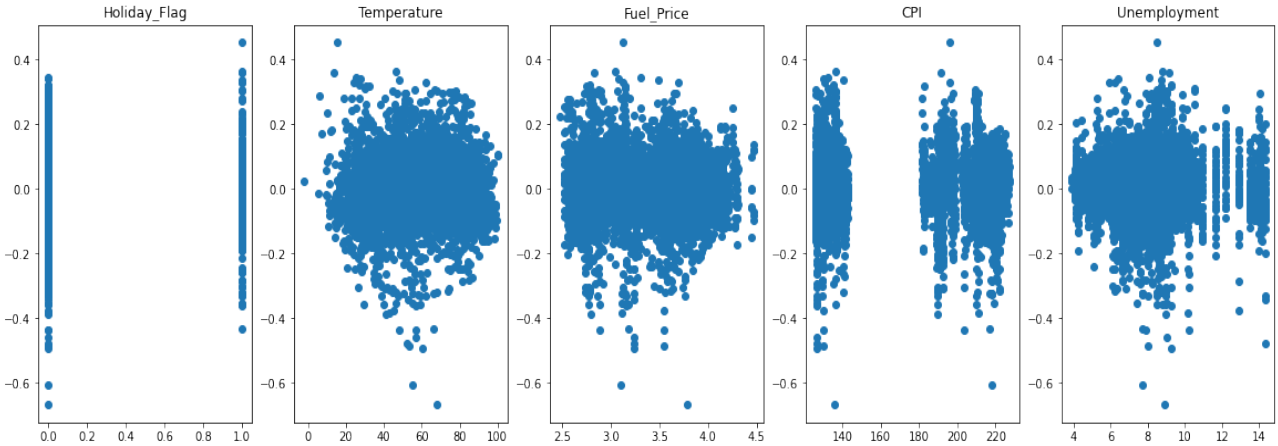


The chart above suggest that homoscedasticity is fulfilled in our case.

6.3.4 Independence: Observations are independent of one another

This part consists of Independence of the error term and variables and autocorrelation between consequent variables.

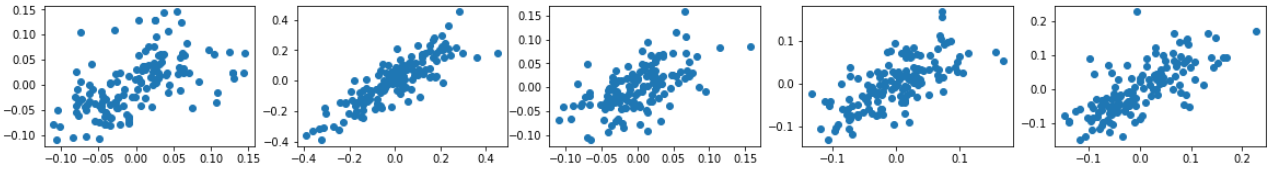
Independence of the error term and variables



Error terms and regressors seems to be independent

Autocorrelation

The plots below shows scatterplots of consequent residuals for shops 6-10.



Picture for other shops is similar.

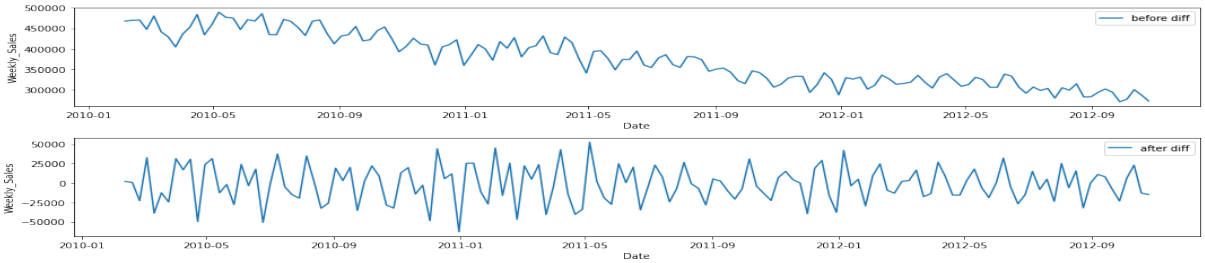
There is very clear autocorrelation between regressors. It suggests that application of time-series analysis techniques can improve our model.

7 Time Series Analysis

To take into account time dependency we will use SARIMA for sales forecasting. The model notation is SARIMA$(p,d,q)\cdot(P,D,Q)_m$. The p, d, q are the same as the ARIMA model. There are four seasonal elements that are not part of ARIMA model are: P– Seasonal autoregressive order, D – Seasonal difference order, Q – Seasonal moving average order, m – The number of time steps for a single seasonal period.

As presented before, we found that the majority of stores have similar trends. The dataset has an obvious seasonal component and no obvious trend. Weekly sales tend to peak for the Thanksgiving and Christmas season and then decline after the holidays.

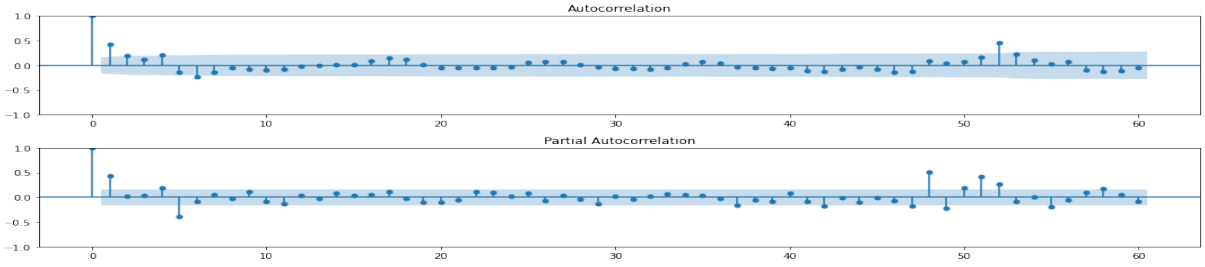
However, five stores [36, 38, 42, 43, 44] have different trend. The dataset contains an obvious trend but no obvious seasonal component. To make it stationary, we will calculate the difference between sales in each week and create a new dataframe to store the difference. Below shows how our data looked before and after the differencing transformation.



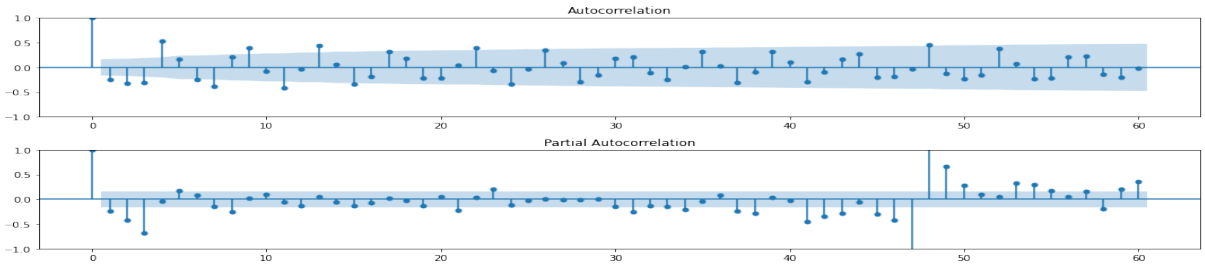
7.1 Using ACF and PACF charts to find the optimal parameters.

To find the optimal parameters for SARIMA model, we first review Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots. ACF can be used to help determine the optimal number of terms (q) to use in an Moving Average (MA). PACF can be used to help determine the optimal number of terms (p) to use in an AutoRegressive (AR) model.

Here we take store 21 to represent the non-trend group a and store 36 to represent the trend group b.



Judging from the plot above, there is a strong positive autocorrelation in lag-52 as well as lag-1 as we expected when observing the time series plot. The significant spike at lag-1 in the ACF suggests a non-seasonal MA(1) component, and the significant spike at lag-52 in the ACF suggests a seasonal MA(1) component. This implies a cyclical annual pattern at the end of the year.



Group b plots show that both the ACF and PACF have significant spikes at lag-4. Consequently, we begin with an ARIMA(4,1,4) model for the non-trend group.

7.2 Fit the model

For the dataset has an obvious seasonal component, we use SARIMA to fit the model. SARIMA stands for Seasonal-ARIMA and it includes seasonality contribution to the forecast. AIC (Akaike information criterion) is an estimator of the relative quality of statistical models for a given set of data. Given a collection of models for the data, AIC estimates the quality of each model, relative to each of the other models. The low AIC value the better. Our output suggests that SARIMAX(1, 1, 1)x(1, 1, 1, 52) is the best combination, so we consider this to be optimal option for group a. ARIMA(4,1,4) is the best optimal solution for group b.

The residuals from these two models can be used to observe the uncorrelated error. The ACF and PACF of residuals show all the spikes are now within the significance limits, so the residuals appear to be white noise.

We use the chosen model to predict. The comparison of the true values with the forecast predictions are shown below.

Our forecasts catch up with some of the peaks and fit with the true values moderately well. However, the forecasting model has a lot of room to improve further by tuning the hyperparameters. One cons of SARIMA model is that SARIMA has 7 hyperparameters which can be tuned affecting significantly the speed of the forecasting process.

8 Random Forest

8.1 Selection of Models

Random Forest is one of the most popular algorithms for regression problems. It is a modified form of bagging that creates ensembles of independent decision tress. It's one of the ensemble method that combines several base models in order to reduce the variance in the final prediction and produce on optimal predictive model.

The key advantages:

- Randomly choose a subset of predictors which can de-correlate the trees in the bagging
- Can handle categorical variables
- It does not make any assumptions about the data or its distribution
- Parallel processing, which makes it faster than regular boosting methods.
- Increasing the number of trees in the ensemble generally does not increase the risk of overfitting.
- Can use OOB instead of cross validation for the hyperparameter selection, which avoid the data leaking.

In the dataset, variables Date and Store are categorical variables. Although we know tree can handle categorical variables, and the performance of with OHE or without OHE will be close, we will try both and see the results.

8.2 Random Forest – with OHE

Convert variable 'Date' to 'Week', and one-hot-encoding 'Week' and 'Store'. 'Weekly_Sales' is the response variable. For the simplicity and fairness, we set number of estimators as 100, max_features as 0.5 (to sample half number of original variables for each split/subset) for both with and without OHE. The reason of choosing 0.5 instead of n/3 in each subset is that there are only 7 predictors in the dataset, and n/3 number of predictors as subset will be too small.

Tune hyperparameter, the max depth of tree, by finding the best OOB score. From tree depth range 1 to 30, the best tree depth is 30.

Final train score is 0.9883, and test score is 0.9588.

8.3 Random Forest – without OHE

Treat variable 'Week' and 'Store' as ordinal variables without one-hot-encoding.

Set hyperparameter, number of estimators and max_features, the same as with OHE model. Tune hyperparameter max tree depth from range 1 to 30. The best max tree depth is 19, which is shallower than with OHE.

Fit the entire train dataset. The final train score is 0.9930, and test score is 0.9560.

With and without OHE conclusion: The performance is pretty much the same, and the max tree depth of without OHE is smaller. So based on the similar performance, we can see that tree can handle the ordinal variables well. We would prefer train the random forest without OHE because of its good performance and less complexity.

Below Feature Importance is without OHE.

8.4 Feature Importance

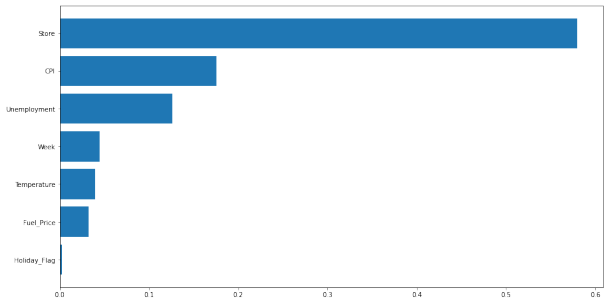
Total 100 estimators in random forest, and here is the count of each variable appears as the top node and second node in each estimator.

	Store	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	Week
top node counts	38	0	5	0	5	36	16
second node counts	38	0	4	2	28	16	12

Table 6: The count of each variable appears as the top node and second node

We can see the Store has the most counts of top node and second node.

Here is the feature importance:



The Store is the most important variable, which aligns with the counts of top node and second node, and other models' results as well. The Holiday_Flag is the least relevant variable, and one of the reasons is that Holiday_Flag is a binary variable, and the model may underestimate its importance here.

Overall, the random forest performs well, which achieves above 0.95 accuracy score. However, it's not as good as the baseline model- Linear regression. The potential reason is that the limited number of predictors causing there is not enough relevant variables in each split.

9 Boosting

9.1 Selection of Models

Boosting is an algorithm that helps us to reduce bias. It is easy to implement and has many options of hyper-parameter tuning to improve fitting. Here, we will use XGBoost, which is fast and performs great on large tabular data. Its key advantages are:

- Built-in regularization which prevents the model from overfitting.
- Parallel processing, which makes it faster than other boosting methods.
- Cross-validation at each iteration of the boosting process

9.2 Data Preparation

Although we know that the boosting methods can handle the label-encoded categorical variables, we will try to implement both dataset, one with one-hot-encoding and one without it, for training models and see which one actually works better.

9.3 Model Implementation and Results

9.3.1 XGBoost (*xgboost* in XGBoost Python Package)

We chose the XGBoost (Extreme Gradient Boosting) over GBM (Gradient boosting Machine) because compared to GBM, XGBoost uses a more regularized model formalization to control overfitting, which gives it better performance in general.

Even with just the default parameters, XGBoost worked quite well both with and without one-hot-encoding of categorical variables. The one without OHE actually performed slightly better although the difference is not significantly large. Based on this finding, we decided to work with the dataset without OHE since working on hyper-parameter tuning for both cases seemed unnecessary and time-consuming. It is also consistent with our initial thought that the boosting methods can deal with label-encoded categorical variables.

	R^2**(train)**	R^2**(Test)**
With OHE	0.9939	0.9683
Without OHE	0.9967	0.9763

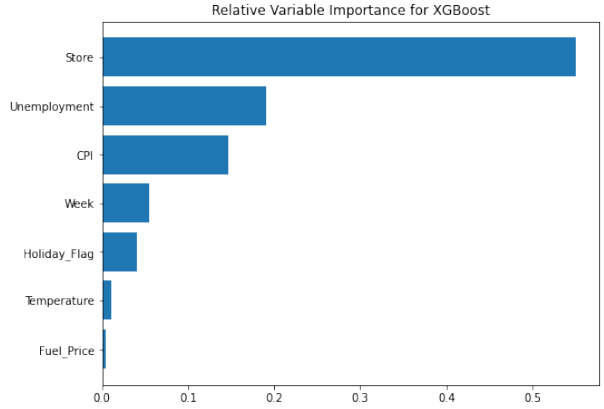
Table 7: XGBoost performance with and without OHE

Using *GridSearchCV*, we could arrive at the optimal hyper-parameters, which turned out to be very similar to the default values. Therefore, our final XGBoost model's performance made a very slight improvement.

R^2**(train)**	R^2**(Test)**
0.9970	0.9786

Table 8: Performance of optimal XGBoost model

Below plot demonstrates the relative importance of features for the XGBoost model.



It is worth noting that we did not optimize every hyper-parameter available in XGBRegressor. For the efficient GridSearchCV process, we selected a subset of important hyper-parameters for tuning and used a subsample (90%) for each iteration of boosting. Store is by far the most important feature in this model, and Fuel_Price is the least relevant. The final model scored test R^2 of 97.86%, which is a remarkable performance.

10 Discussion

Comparison by RMSE for linear models and SARIMA given in the tables 9.

model	Train Score	Test Score
Baseline Model	157 660	163 569
Linear regression	86 048	86 641
SARIMA	NA	56 730 / 13 146

Table 9: RMSE scores of different models

For SARIMA we have calculated test scores only for 2 shops, for these shops there is clear improvement over results from linear model. Based on these results we can say that SARIMA shows the best results. However, this model has one serious disadvantage. Because of model complexity and limited project timeline this model makes predictions based only on store number and previous sales. This approach works very well to predict nearest future, but does not help to understand how our stores will perform in different economical conditions.

For this reason we tried to improve linear model using tree-based approaches. These results are presented in the table 10.

model	Train Score	Test Score
Baseline Model	0.993029	0.956001
Random Forest	0.988340	0.958831
Linear regression	0.975517	0.975733
XGBoost	0.997001	0.978635

Table 10: R^2 scores of different models

We can draw few conclusions regarding these results:

1. Linear regression provides very good results. The linear regression assumptions mostly satisfied this is expected results, but there are few assumptions violations which leaves room for improvements. One clear disadvantage of this approach is that it required additional work including checking regression assumptions, doing data transformation (for example one-hot encoding etc).
2. Random forest provides comparable results but didn't manage to overperform.
3. XGBoost is the best for making predictions.
4. Clear advantage of tree-based models in this case is that you don't need very deep analysis regarding preconditions, and they can work with non-transformed data.

11 Future Work

Although we have reached very good performance and tested four various approaches there is still room to improve predictions quality. Further improvements can include

- In our work we used only approaches taught in CSCI E-109A courses. Other methods can be used to improved the results – Gaussian Process, SVM, Neural Network, etc.
- More regressors can be integrated into SARIMA model. Based on results shown by our limited SARIMA model this approach looks promising.
- Our dataset covers only limited timeframe. With more data, our model might improve even more. More recent data will be helpful as the customer and market behavior changed a lot since 2012.
- Studying other retail stores data can help to find patterns and approaches which can be potentially applied to Walmart dataset analysis.

References

[1] *A Machine Learning Approach for Retail Sales Forecasting. SAS Customer Support Site.* Mar. 2020. URL: https://support.sas.com/en/support-home.html.

[2] C. Fuenzalida. *Forecasting key retail performance indicators using interpretable regression. Sensors (Basel, Switzerland). U.S. National Library of Medicine.* Mar. 2021. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7962459/.

[3] *How to forecast demand for your retail store (and why you should). Kaggele.com*. May 2020. URL: https://www.kaggle.com/datasets/aditya6196/retail-analysis-with-walmart-data.

[4] M. Kay. *How to forecast demand for your retail store (and why you should). Shopify*. Aug. 2021. URL: https://www.shopify.com/retail/demand-forecasting#4.

[5] M. Liebeskind. *5 machine learning techniques for sales forecasting. Medium. Towards Data Science.* May 2020. URL: https://towardsdatascience.com/5-machine-learning-techniques-for-sales-forecasting-598e4984b109.

[6] A Molodoria. *How to apply machine learning to demand & sales forecasting in retail. MobiDev.* Oct. 2022. URL: https://mobidev.biz/blog/machine-learning-methods-demand-forecasting-retail.

[7] K. Prawtama. *Predicting store sales - random forest regression. Medium. MLearning.ai.* Feb. 2022. URL: https://medium.com/mlearning-ai/predicting-store-sales-random-forest-regression-b77abec64c17.

[8] Harvard Business Review. *How to choose the right forecasting technique. Harvard Business Review.* Nov. 2021. URL: https://hbr.org/1971/07/how-to-choose-the-right-forecasting-technique.