Walmart Retail Data Analytical Report

Group 5

Yining He
Li Jiang
Yutong Liu
Liangliang Sun
Jingru Zhang
Xiaopeng Zheng

1) Introduction

Provided by Kaggle, the dataset is "Walmart Recruiting – Store Sales Forecasting." The challenge that we face is how to make business decisions based on limited history. Specifically, our final goal is to predict weekly sales of each of the stores based on other variables. Predicting sales is important because managers can do inventory management better and plan for promotions when there is expected to be a dip in sales. Sales forecasting can also largely help a retail company to boost its profit.

The dataset includes information of 45 Walmart stores in different regions. Each store has different departments and each store has different sizes. The variables include weekly sales, date, temperature, fuel price, CPI, unemployment, markdowns, and whether it is a holiday. We hope to find out as many patterns as possible among those variables.

First, we described the findings from preliminary analysis based on the graphs plotted as well as the correlations between variables. Then, we used other more advanced methods to predict the sales -- random forest, neural network, gradient boosting machine, and time series analysis. By using these models, we are able to predict the weekly sales analytically.

2) Data cleaning

There are three original datasets -- "Features", "Sales", and "Stores". All of these datasets are csv files, and we combine the three csv files into one.

The "Features" and "Sales" datasets are combined through an outer join, on three keys -- stores, date, and IsHoliday. For the combined new table, we sum weekly sales of all departments for the same store, so that we can predict sales on a store level instead of a department level. Then, we merge the newly-created dataset with the "Stores" dataset. We use the outer join function again here. Now we have columns of "Store", "Type", "Size", "Date", "IsHoliday", "Promotion", "Temperature", "Fuel_Price", "CPI", "Unemployment", "Weekly_Sales". Since the "IsHoliday" column is of Boolean values, we use one-hot encoding to convert the column to three columns of dummy variables. We also parse from the "Date" column to create two addtional columns -- "Year" and "month". so that we can use the specific year or month to do the analysis later.

After merging all three csv files into a single dataset, we remove the outliers (those samples that have abnormally large or small weekly sales values. We use the Z value to filter the outliers. We choose to keep 99% area which is also the probability under the normal distribution as our normal sales data and eliminate the 1% of the two tails. We find the 2.58 as the threshold z value. We do this to keep the sales data that are within the 2.58 standard deviations of the mean.

3) Descriptive Statistics and Preliminary Results

We first plot a set of histograms (Figure 1) to analyze the distributions of six main variables. There are three concentrations of store sizes, which may be related to the store types. Promotions are extremely centralized around $0 to $500. Most unemployment rates are distributed around 6.4% to 8.7%.

The box plot of store types and sizes (Figure 2) shows significant size differences among the three types of stores. Type A stores have the largest size and are likely superstores; Type B stores are medium-sized stores, and type C stores are the smallest kind. Both type A and B stores are negatively skewed with the median store sizes around 202k and 115k respectively; the size of type C stores is concentrated around 40k.

Figure 3 shows the relationship between weekly sales and store types after removing outliers of sales. The distributions of all three box plots are close to normal distribution. Similar to Figure 2, three types of stores can easily be classified according to the different sale ranges. We can see that Type A stores have the most weekly sales, followed by Type B and then by Type C stores.

Figure 4 provides the impact holidays have on sales. During holidays, more stores have higher sales volume. Furthermore, when considering store types, we found out that type B stores were relatively more sensitive to the influence of holiday. We also plot the weekly sales volume against time. We do not see an overall trend through the years, but the spikes during holiday seasons are obvious from the graph.

Figure 8 is the pairwise correlation matrix. Except for size and weekly sales, other variables do not have a specially high correlation with any other individual variable.

Figures 9 and 10 are linear regressions of each type of store's weekly sales against fuel prices and promotions. We find that fuel price only has a statistically significant effect on type B stores' weekly sales. With the same threshold, all types of stores are correlated with promotions.

4) Machine Learning Models

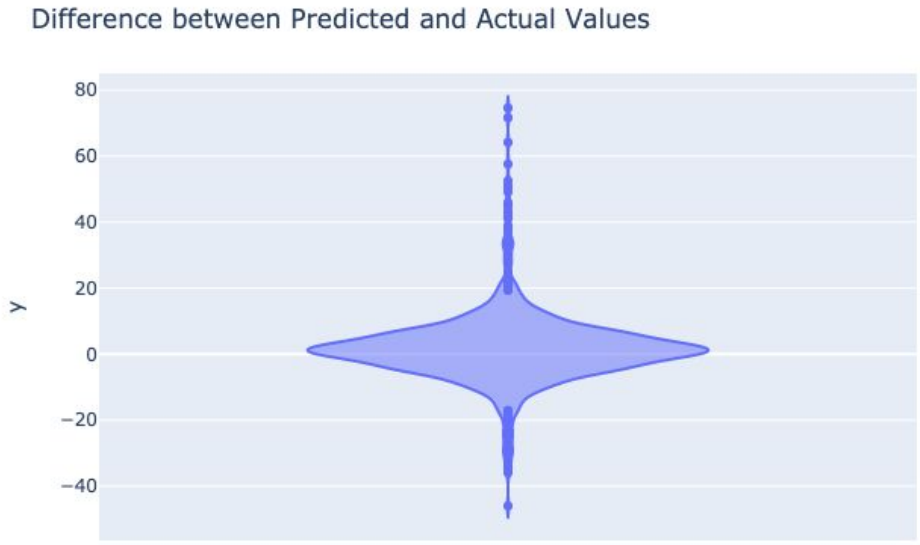
Before we build models, we turn the categorical features into numbers through one-hot encoding. The "Type" column is now split into three separate columns, corresponding with the values of the one-hot encoding. We also split the entire dataset into a training set (72%), a validation set (8%), and a testing set (20%). The purpose of the validation set is for hyper-parameter tuning, which we will discuss later. Because of the nature of our models, the relative scale of each feature does not play a role in determining their importance in training, we do not need to normalize the data.

1. **Random Forest:**
   Random forest is an ensemble model composed of a plethora of decision trees. Unlike decision trees, however, random forest only takes into account a random subset of features rather than all of the available features. This characteristic, along with bootstrapping, provides more versatility given the limited features and a relatively small dataset.

- *Hyper-parameter tuning*: Two of the most important hyper-parameters in random forest are the number of estimators and the max depth of each tree. We use Scikit-learn's GridSearchCV to perform an exhaustive search over different values for these hyper-parameters. "n_estimators" can be chosen from 100,200,300,400, and 500. "Max_depth" can be chosen from 5,10,15,20,25. GridSearchCV also performs cross-validation while picking the best parameters. We set the cv to be 5, meaning there

is 5-fold cross-validation for each combination of the parameters, and the average score across the 5 cross-validations is used as the metric for the best parameters. Therefore, there are 125 iterations in total. Due to our limited computational time and power, the grid search cross-validation is performed on the validation set rather than the entire training set. The set of hyper-parameters selected by the cross-validation is 20 for the max depth and 400 for the number of estimators.

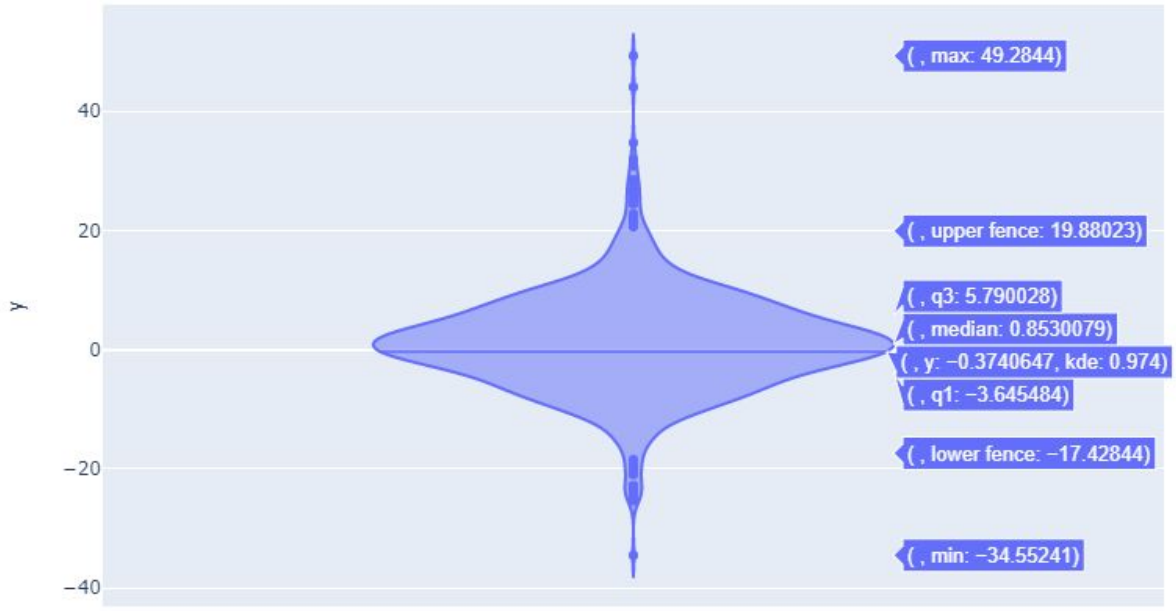
- *Cross-validation*: We then perform a 10-fold cross-validation on the entire dataset using the parameters chosen in the last step. This means that the entire dataset is split into 10 equal-sized folds. First, fold 1 is used as the testing set, and the rest 9 folds are used as the training set. Then, fold 2 is used as the testing set, and the other 9 folds are used as the training set. We repeat this ten times. In the end, the average R-Squared for the 10 Cross-validations is 0.937.
- *Prediction*: The plot below shows the distribution of the difference between the predicted values and actual values as a percentage of the actual values. Most prediction errors are with 20 percent of the actual values.



2. Gradient Boosting Decision Tree Regressor (GBDT)

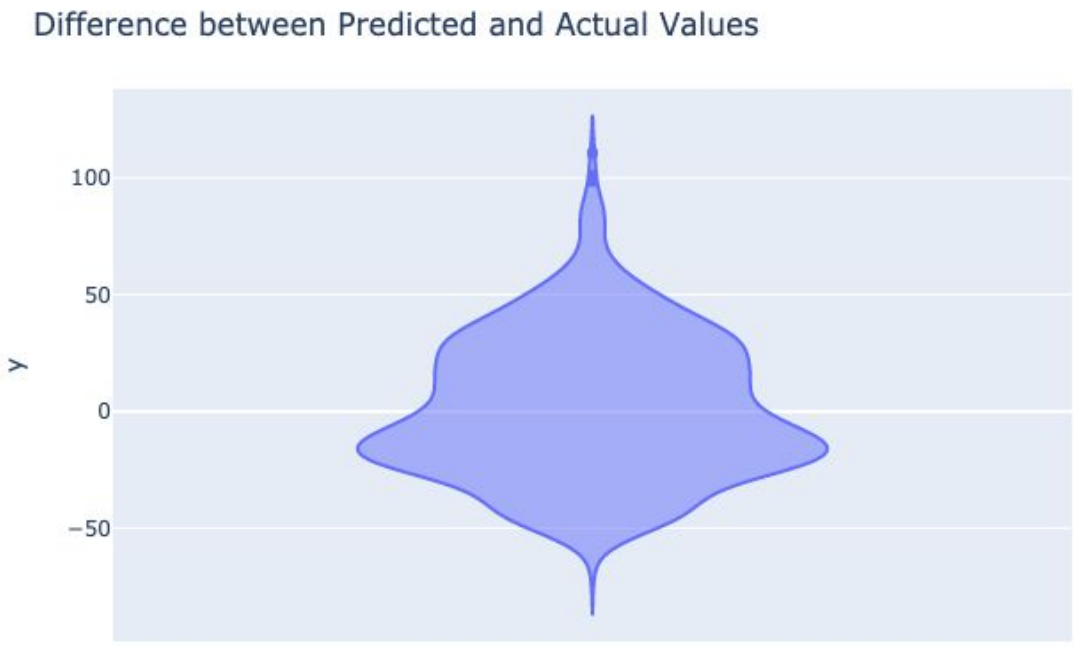
- *Methodology*: According to our descriptive analysis, the sales of different types of stores have different levels of sensitivity to the parameters such as fuel price and promotion. We hope a tree-based model can help differentiate such features.
- *Hyper-parameter tuning*: We employed the same methods as in the random forest. The best hyperparameters obtained by the GridSearchCV are 0.05 for learning_rate, 4 for the max_depth and 900 for n_estimators.
- *Prediction Results*: After generating the best parameters, we performed a 10-fold cross-validation on the entire dataset (training, validation, and testing sets) and obtained scores for each cross-validation. In addition, we also fit our training data set using the parameters, and then used the fitted model to predict the test values. The average score (R-Squared) for the 10-fold cross-validations is 0.969. The plot below shows the

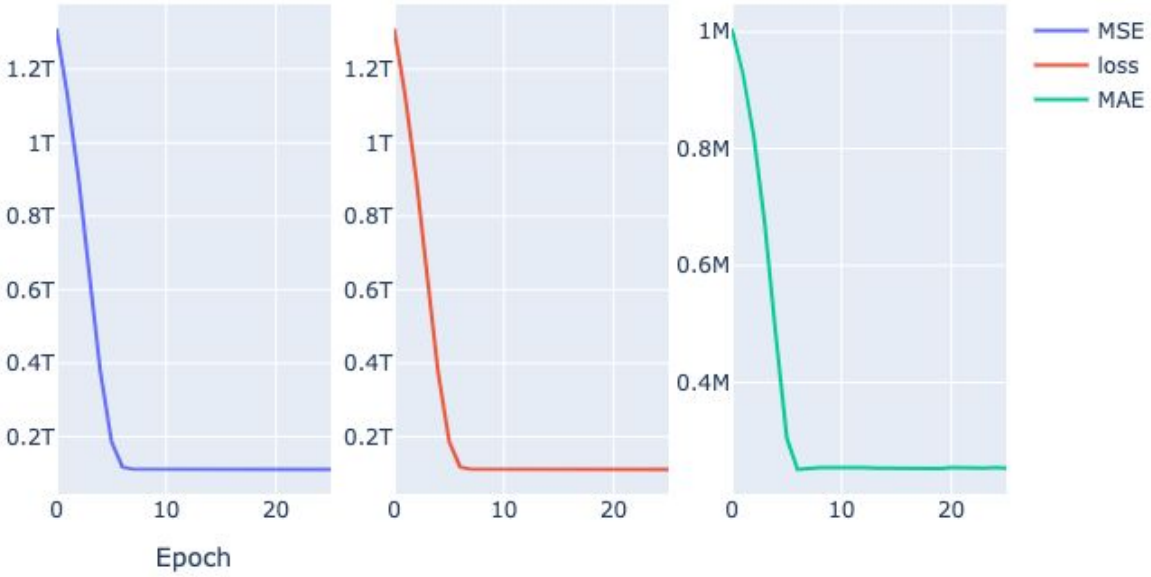
distribution of the difference between the predicted and actual values (as a percentage of the actual values). We can see that most deviations are within 20% of the actual values.



3. **Neural Network:**
   Neural networks also seem to be a good choice. Given a large number of data, neural networks usually have very good prediction power. We use the high-level API of TensorFlow, called Keras, to build the model.

- *Network Structure*: The neural network has four dense layers, composed of one input layer, one output layer, and two hidden layers. We used "Relu" as the activation function for the first three layers. "Relu" is a linear transformation, the output f(z) is zero when z is negative and f(z) is equal to z when z is above or equal to zero. The activation function for the output layer is also linear, which is suitable for regression.
- *Training*: We set the number of epochs to be 50, also permitting early stopping to prevent overfitting. If the validation loss is smaller than 0.001 for five epochs, the training will stop. In the end, there were only 26 epochs before the model converged. (See first graph blow)
- *Result*: The results of the neural network are not optimistic. (See second graph below) Its root mean squared error is almost twice as large as that of random forest. This might have resulted from a few factors. First, neural networks need a large number of samples. The dataset is relatively small for sufficient training. Second, the number of features is sparse. Finally, neural networks do not have a special advantage over other models when the data are structured.

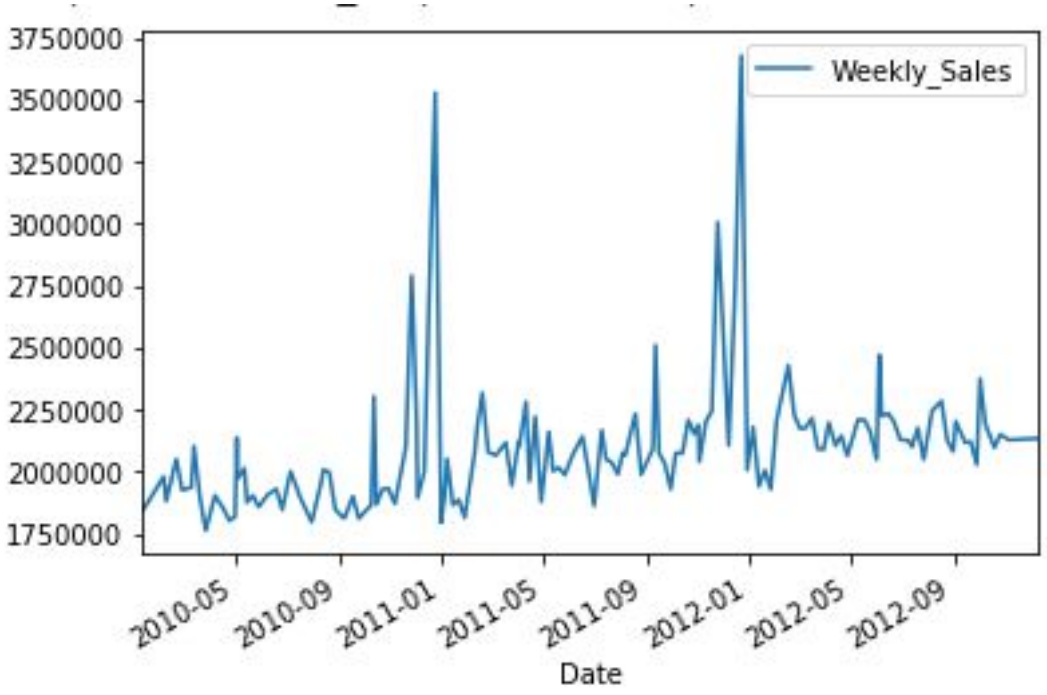


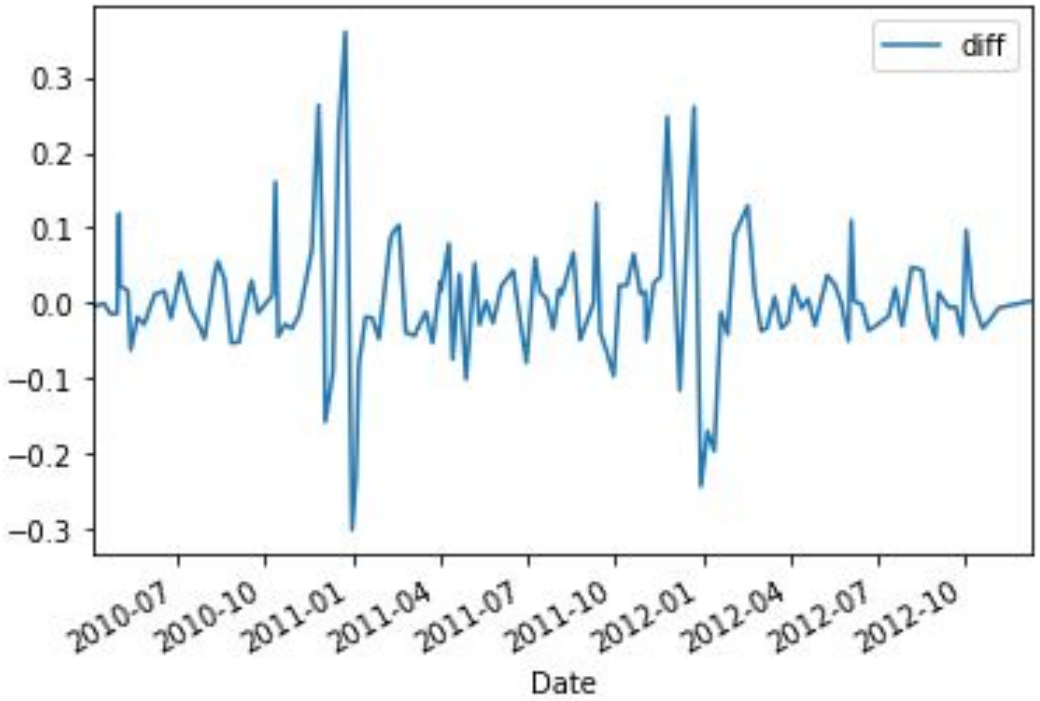
- 

4. **Time Series Analysis (ARIMA)**
   It is known that a store's sales could have seasonal fluctuation and may increase when the time goes by. When we plot out a specific store's weekly sales and date from our data set, this trend is obvious. The line chart shows Weekly Sales of Store No.4, from which we can say the sales soared during December and January, and there is a small annual increase between 2010 and 2012.

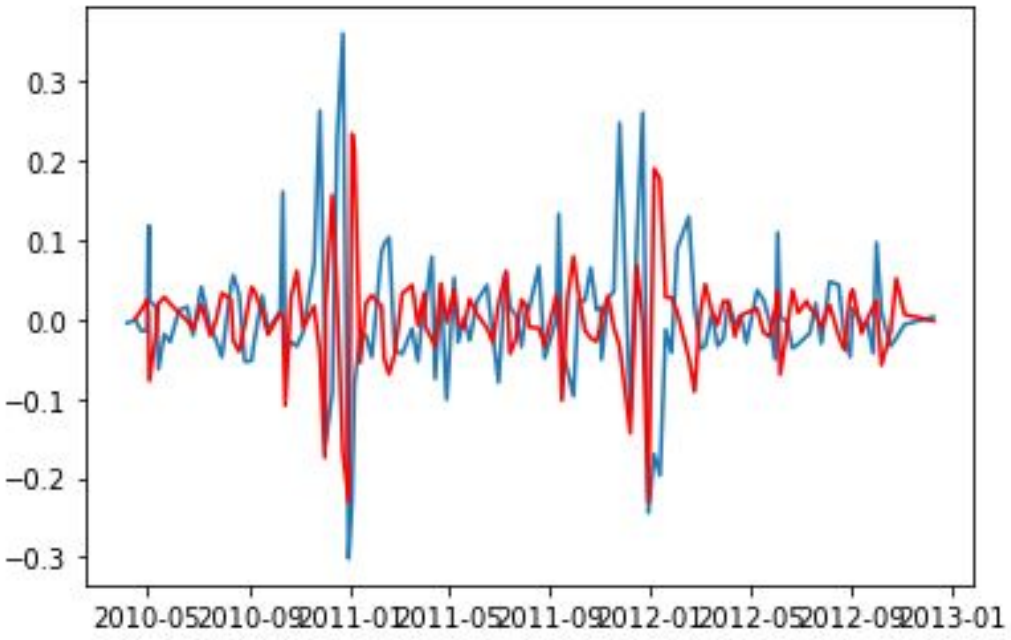
- *Find Seasonal Trend:* Before we compute the seasonal peak and low points, we need to eliminate the annual growth rate. To realize this move, we first log all the data and then

calculate moving averages based on logged data. By simply subtracting the moving average from the sales, we can calculate the effect of seasons.





- *Get Parameters:* Here are the parameters for ARIMA model
    a. Number of AR (Auto-Regressive) terms (p): AR terms are just lags of the dependent variable. For instance, if p is 5, the predictors for x(t) will be x(t-1)….x(t-5).
    b. Number of MA (Moving Average) terms (q): MA terms are lagged forecast errors in prediction equation. For instance if q is 5, the predictors for x(t) will be e(t-1)….e(t-5) where e(i) is the difference between the moving average at ith instant and actual value.
    c. Number of Differences (d): These are the number of nonseasonal differences, i.e. in this case we took the first order difference. So we can either pass that variable and put d=0 or pass the original variable and put d=1. Both will generate the same results. After calculated, we found p = 2, q = 1 and d = 0.
- *Fit the Model*: The results were desirable. Even without other variables, Walmart's sales can be predicted by date. So, if we put more weight on the date and seasonal features, our machine learning could be more accurate



5) Conclusions

From the above analysis, we see that tree-based models perform well when predicting store sales. However, because the data size and number of features are too small for neural networks to generate significant learning, neural networks underperformed compared to the tree-based models. The time series analysis was also successful in predicting weekly sales. We only did an analysis for one store. Because all the stores had predictable seasonality and trend patterns, which were slightly different among different stores. Therefore, we could further improve our model by combining the time series analysis with other prediction models like random forest and produce even better results.

Appendix

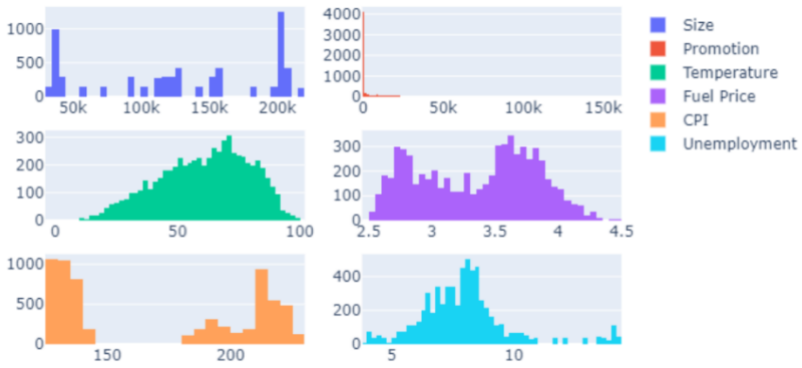


Figure 1 Distribution of select variables

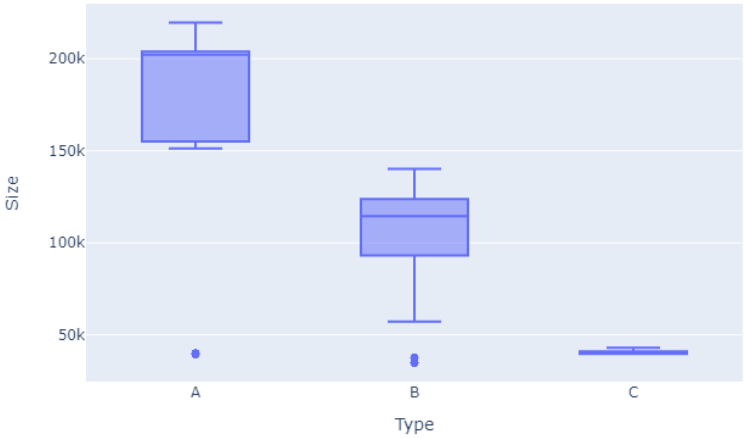


Figure 2 Relationship between store types and sizes

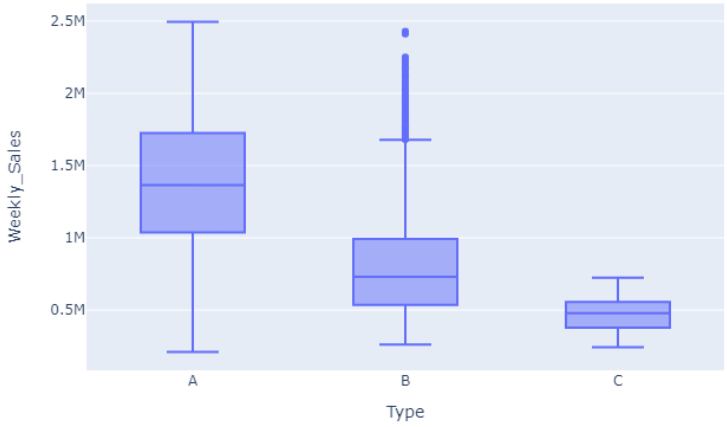


Figure 3 Relationship between store types and weekly sales

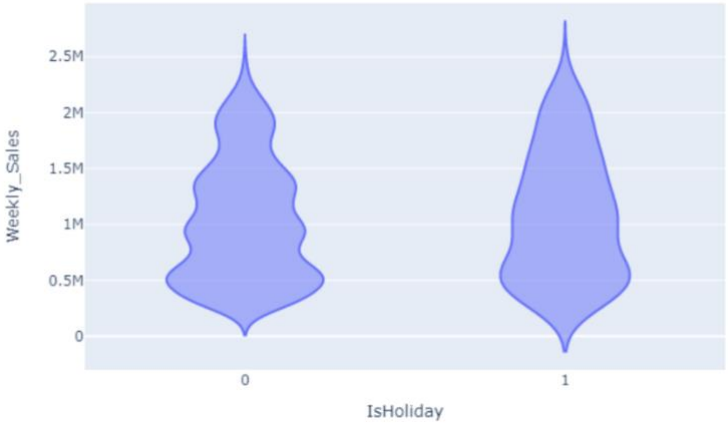


Figure 4 Relationship between holiday and weekly sales

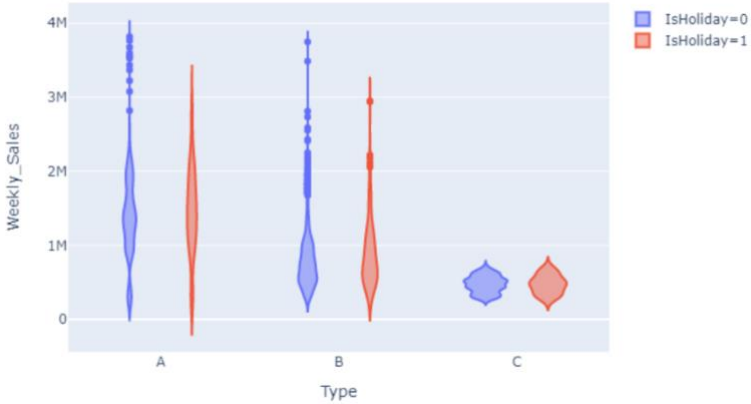


Figure 5 Relationship between store types, holiday and weekly sales

Figure 6 Relationship between date and weekly sales

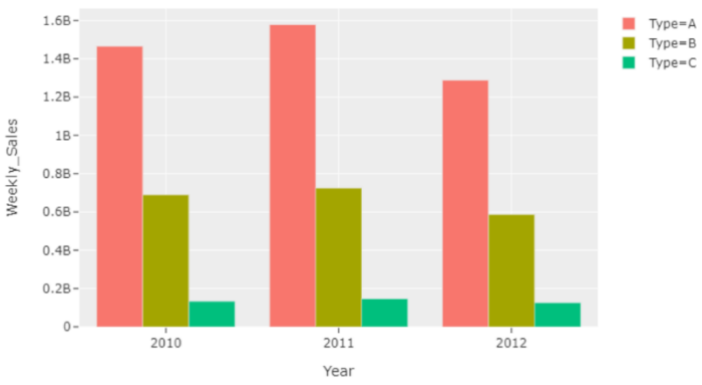


Figure 7 Relationship between Year and weekly sales

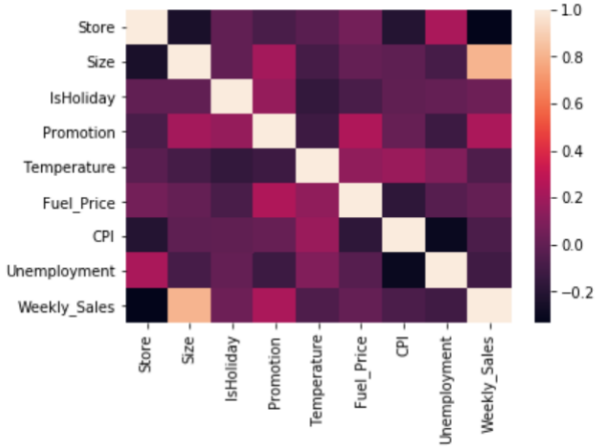


Figure 8 Correlation between every two variables

Figure 9 Regression of weekly sales and fuel price

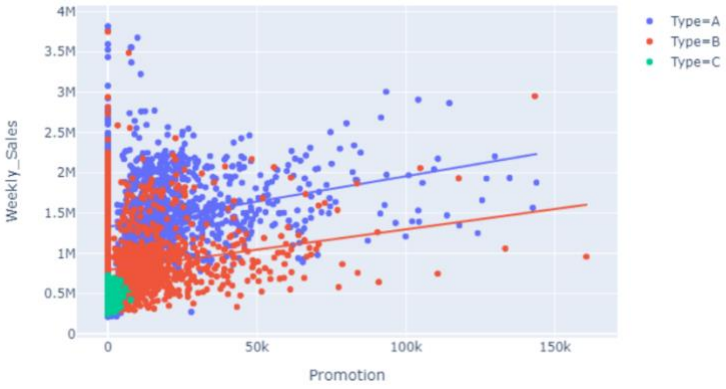


Figure 10 Regression of weekly sales and promotion