Forecasting Future Walmart Sales using KNN/Decision Tree Regression

Muhammad Salman
Department of Electrical and Computer Engineering
Carleton University
Ottawa, ON Canada
muhammadbsalman@cmail.carleton.ca

Marziyeh Zamiri
Department of Electrical and Computer Engineering
Carleton University
Ottawa, ON Canada
marziyehzamiri3@cmail.carleton.ca

ABSTRACT

In this research paper, we use a provided Walmart dataset from Kaggle, to predict future sales that the megastore will generate. The information in these datasets comprise a wide range of information regarding 45 stores and their localities. These include departments per store, weekly sales, sales during weeks of major holidays, physical size of the store, unemployment rate for the local population, etc. This is quite significant as it would allow the store to predict which departments within certain stores would yield greater sales depending on the locality and time of year. In a realistic sense, this would allow Walmart to calculate how much and what kind of inventory they would require, leading to a growth of sales and minimizing costs. All data analysis for this project past this point is done using Python3.9. Within Python, the Pandas library is used to attain quick access to CSV files and build a virtual data frame using the given data. The various data frames from each CSV file are first cleaned, as is done in every data science project. In simple terms, data that is N/A can be filled with 0 or any other value using Pandas' commands. Once that is complete, the CSV files are merged into one large data frame for combined access. The next step is exploring the data, and observing how certain features relate to others. For each comparison, a certain type of plot was generated using the Matplotlib and Seaborn libraries. An example of this was a bar graph that was generated to see the difference in sales between holiday weekly sales and standard weeks. The last step will include modelling the data, training the model, and verifying the model. A regression model is used, since the predictable value is a dependent variable, weekly sales, based on a series of independent variables, such as store size, holidays, locality statistics, etc.

KEYWORDS

Data Science, Walmart sale forecasting, Python, Sale prediction, Kaggle dataset

ACM Reference format:

Muhammad Salman, and Marziyeh Zamiri. 2021. Forecasting Future Walmart Sales using KNN/Decision Tree Regression. Final report for DATA5000 offered at Carleton University, Ottawa, Canada.

1 Introduction

Inventory has cost a lot of money to produce or purchase. Hence it is either a source of revenue or an expense item. A better understanding leads to a better decision of what inventory to buy, when to buy it, and how much to purchase. The goal of inventory management is to minimize the amount of inventory in stock, while ensuring the items are available when the customers want to purchase them[1-3]. One of the biggest challenges with inventory management for a

retail company is trying to predict customer demand of the items they sell. It is crucial to know what information helps to determine the amount of inventory to carry[4]. First step is to understand the shape of the demand curve for inventory items. It is important to consider if the selling product has high seasonal demand or not. As an example, winter clothing is a seasonal product with a few months popularity, but, black socks are not a seasonal product. Products that are not seasonal sell all year long with fluctuating sales week by week, but they don't show a time frame where sales have a substantial increase or decrease. The challenge for seasonal products is how many weeks or months the high-volume season will last. Besides, it is crucial to estimate how fast the demand will increase at the start of the season and how fast it will drop off at the end of the season. If retailers don't obtain inventory fast enough at the start, they will be out of stock for customers' orders and lose sails. If they don't decrease the inventory to match the end of the curve which has a downward slope, they risk being stuck with inventory that does not sell. Without careful planning, inventory does not sell during the peak season and continues to tie up cash [5,6]. The retailers may have to wait until next year for another opportunity to sell that item to release that cash or if the item is obsolete by then, they may have to write-off that inventory. Writing-off inventory is a loss to their business, and affects their profitability. Note that even for non-seasonal products there might not be a seasonal increase but the demand for that item may still vary weak-by-weak, and we see different demands across product properties such as size, color, etc. and geographical locations. There might be a long history of data on one item, but it still varies year by year as consumer brand preferences change and as economic indicators such as inflation rates, the consumer price index, and unemployment rate change, these economic factors impact consumer purchasing level.

Forecasting and planning, both are particularly important, and allow retail companies to better understand their future needs, and better support of their customers. In addition, help them to better predict a more accurate financial picture of the future. Till now, forecasting has not been easy. Sometimes, there are too many items to plan. It is not an effortless process to get all the information consolidated from everyone involved. Besides, getting a consensus view is often even more difficult. On top of that, there is a human fact of actually doing the forecasting and planning.

Aforementioned, forecasting is very crucial for retail companies, more importantly during the special occasions of the year. They use historical data to create models about expected sales during various times of the year. Walmart which is a giant retailer particularly in the United States, faces the challenge of making marketing decisions on limited historical data. Holidays such as Super Bowl, Labor Day, Thanksgiving, and Black Friday occur once a year. Therefore, the effectiveness of strategic policies can only be tested during that window. Walmart's business goal is to predict sales, sell more products, and prepare more inventory during busy holiday seasons. They initially designed a forecast to assist them in planning their inventory, managing its budget, setting targets, etc. going forward.

2 Dataset

Walmart provided datasets that comprise a wide range of information regarding 45 stores and their localities.The dataset was pulled from Kaggle [7] as a collection of CSVs. These datasets include departments per store, weekly sales, sales during weeks of major holidays, physical size of the store, unemployment rate for the local population, etc. In these datasets, features vary from numerical values, such as number of sales, to boolean True/False for if a week contains a holiday or not. Relevant CSVs within the datasets are listed below, along with the features within each one.

features.csv:
This file contains additional data related to the store, department, and regional activity for the given dates. It contains the following fields;

- *Store:* The store number of the particular data point that is being analyzed
- *Date:* The data of the current data point
- *Temperature:* The outdoor temperature within the locality of the current store

- *Fuel_Price:* The average price of gasoline in the locality in $/gallon
- *MarkDown1-5:* The amount in dollars that is reduced from store products for the current markdown, based on type of statutory holiday
- *CPI:* The Consumer Price Index of the current locality, which is the average price of goods for a standard shopping trip
- *Unemployment:* Rate of unemployment in the locality
- *IsHoliday:* If the week of the data point contains a holiday (binary True or False value)

stores.csv:
This file contains anonymized information about the 45 stores, indicating;

- *Store:* The store number of the particular data point that is being analyzed
- *Type:* Types can range from A, B, C, and do not represent any other data value from the surface
- *Size:* The amount of products within the particular store

train.csv:
This is the historical training data, which covers from 2010-02-05 to 2012-11-01. Within this file you will find the following fields;

- *Store:* The store number of the particular data point that is being analyzed
- *Department:* The department number within the particular store, upto to 99 departments
- *Date:* The data of the current data point
- *Weekly_Sales:* The weekly sales generated for the particular department and store
- *IsHoliday:* If the week of the data point contains a holiday (binary True or False value)

The dataset contains many values that are numerical, with IsHoliday being a boolean true or false. Another problem with this dataset is that it is not one unified CSV file, rather it draws data from three different CSVs. This caused data to be repeated and harder to analyze.

3 Methodology

Data analysis was performed using Python, with additional libraries such as pandas for data manipulation, scikit for training and machine learning algorithms, and matplotlib/seaborn for graphical observations. The methodology for this project was more or less the standard Machine Learning lifecycle.

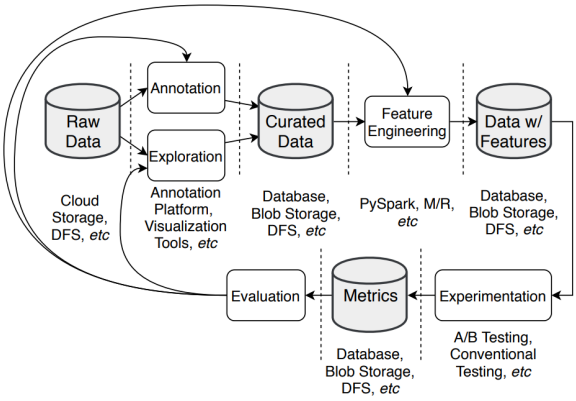

Figure 1: Machine Learning Lifecycle

3.1 Data Cleaning

The collection of data stage was ignored for the project since the dataset was simply collected from Kaggle. From there, the data had to be cleaned and optimized to best suit the needs of the project. This is first done through loading the CSVs as dataframes using pandas.

```
#Load all excel files in through pandas
train  = pandas.read_csv("train.csv")
features = pandas.read_csv("features.csv")
stores  = pandas.read_csv("stores.csv")
```

Figure 2: Load CSVs as Dataframes

An example of this could be seen as many values being missing, or N/A. This is seen for many values under the Markdown features in features.csv. The fillna() command under pandas takes care of this by substituting a value for any N/A value in the given column.

```
#For unavailable markdowns, fill with 0
features['Markdown1'] = features["MarkDown1"].fillna(0)
features['Markdown2'] = features['MarkDown2'].fillna(0)
features['Markdown3'] = features['MarkDown3'].fillna(0)
features['Markdown4'] = features['MarkDown4'].fillna(0)
features['Markdown5'] = features['MarkDown5'].fillna(0)
```

Figure 3: Dropping N/A Values from Columns

N/A values were subsequently filled with appropriate values for each feature. Then, the three data frames (train, stores, and features) from their respective CSVs were merged into one dataframe: main. This allowed parsing and analysis over one singular dataframe, instead of constantly switching between them. Any redundancies were removed.

In order to gain finer results, the date feature, which was originally in the notation Month-Date-Year, was broken down into individual components: Week, Year, and Day. Analysis can now be performed on each one of these features individually instead of one large date feature. Once complete, .drop() function can drop an entire feature/column from the dataframe, which is what was used to drop the initial Date feature.

```
#Sort date into finer detail
main['Date'] = pandas.to_datetime(main['Date'])
main['Week'] = main['Date'].dt.isocalendar().week
main['Year'] = main['Date'].dt.isocalendar().year
main['Day'] = main['Date'].dt.isocalendar().day
main=main.drop(["Date"], axis=1)
```

Figure 4: Break Date into Individual Features

Finally, the type of store feature was also simplified to numerical values, 1 for A, 2 for B, and 3 for C. The main data frame was then saved as its own CSV, to allow for easy data loading in future scripts.

3.2 Data Exploration

Moving forward, it was important to analyze which features correlated strongly with other features, and which one of them played the strongest role in predicting sales.

The first and most straightforward one to observe was which week of the year generated the most sales (Figure 5). As it can be seen, weeks 47 and 51 yielded the most amount of sales, which corresponds to Thanksgiving/Black Friday, and Christmas weeks respectively. This correlation proves to be quite strong as it allows us to derive that weeks with holidays will generally yield more sales.

Another correlation that was performed compared the Store#, against the Average Weekly Sales. (Figure 6). This visually showed us which stores will likely generate more sales, but they do not give us a general over-picture correlation for the dataset.

We correlated between Store Type with Store Size, with Store Type ranging between Type A-B-C (Figure 7). This showed us a strong correlation, with Type A stores having the greatest store size (in terms of number of products), followed by Type B, then finally Type C. This also tells us that there is a strong chance that A will yield higher sales than B, and B will yield higher sales than C.

However, this was to be disapproved by the next correlation between Store Type and Weekly Sales (Figure 8). According to this graph, almost no correlation is seen between the type of store and the impact it will have on sales revenue, however it does seem as Type C stores will yield the least amount of sales (Type A/B cannot be differentiated graphically).

Similar to before in Store# vs. Average Sales, we later graphed Department# vs. Average Sales simply to graphically observe which department would yield the highest revenue (Figure 9). Although this will not show a direct correlation, it will allow us to understand which departments are more common to generate higher sales and which ones could possibly lead to a loss in profit.

The final correlation done as part of the data exploration stage was between IsHoliday? and Weekly Sales. This correlation was expected to be quite strong, with sales on holiday weeks projected to be much greater than weeks with no holidays. However, only a slight rise in sales was seen for weeks with holidays (Figure 10).

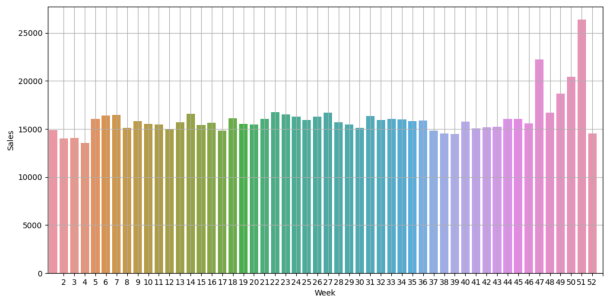

Figure 5: Week of the Year vs. Average Sales

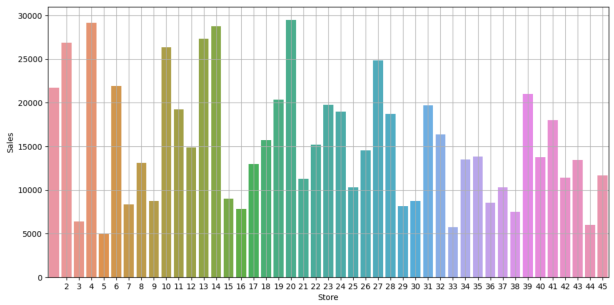

Figure 6: Store# vs. Average Weekly Sales

Figure 7: Store Type vs. Store Size

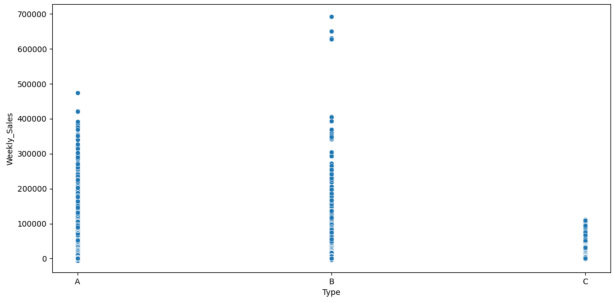

Figure 8: Store Type vs. Average Weekly Sales

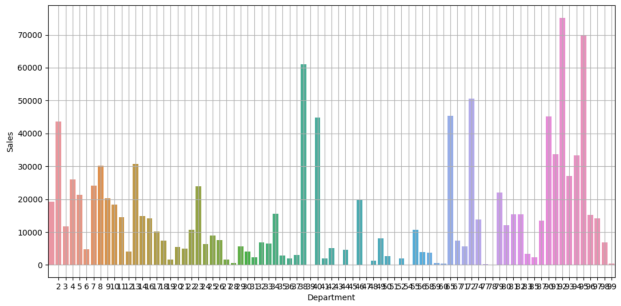

Figure 9: Department# vs. Average Weekly Sales

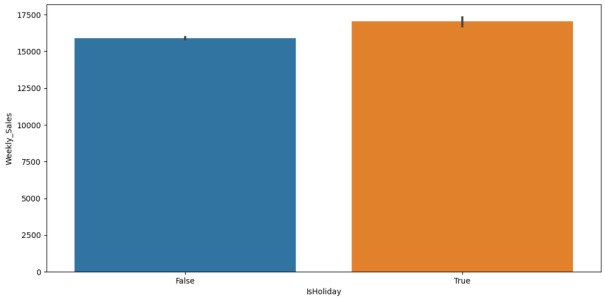

Figure 10: IsHoliday? vs. Average Weekly Sales

Using these key points and correlation information yielded from our observations, we can continue with data modelling in the next steps, as well as the various Machine Learning Algorithms and techniques that were implemented to yield optimal results.

4 Results

4.1 Machine Learning Algorithms

4.1.1 Decision Tree model

Decision tree is a type of supervised learning algorithm, that is mostly used in classification problems. A tree has many analogies in life and turns out it has influence in a wide area of machine learning covering both classification and regression. Trees are also known as carts (classification and regression tree). A decision tree is a flowchart like structure where each internal node denotes a test on an attribute, each branch represents an outcome of a test and each leaf or terminal node holds a class label. The top most node in the tree is the root node.

Figure 11 shows a schematic of a decision tree with all the branches. In decision analysis a decision tree can be used to visually and explicitly represent decision and decision-making as the name goes it uses a tree like model of decisions. So, the advantage of cart is, it is easy to understand, interpret, and visualize. Decision trees implicitly perform variable screening or feature selection. It can handle numerical as well as categorical data. It can also handle multi output problems. Decision tree requires relatively little effort from the user for data preparation and nonlinear relationships between parameters do not effect the clip performance. The disadvantages of cart however is that decision tree. The disadvantage of cost however is the decision tree learners can create over complex trees that do not generalize the data well. This is also known as overfitting decision trees can become unstable because small variation in the data might result in a completely different reading generated this is called variance which needs to be lowered by methods of bagging and posting. Greedy algorithm can not guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees. Decision trees also learn to create bias trees if some classes dominate. Therefore, it is recommended to balance the dataset.

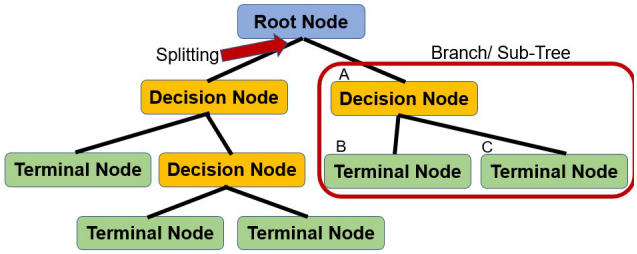

Figure 11: Decision Tree Schematic [8]

Decision tree is interpretable and the output that makes sense to a human. Further, it handels attributes nicely and can handle missing data. Additionally, it is compact, fast, and the cost for running a decision tree is in the order of the depth of the tree. The downside of the decision tree is that it is not guaranteed to be optimal. This algorithm is greedy which means at each level of the tree it is making the optimal decision for the current split not globally [8].

4.1.2 KNN model

K-nearest neighbor (KNN) is a very simple classification and regression algorithm. In case of regression, new data get labeled based on the average value of the k nearest neighbor. We provide K value to the algorithm based on experiment, or past knowledge, or etc., which identifies how many neighbors we want in a particular class. KNN method is a lazy learner since it does not learn much from the training data. The learning mainly is coming from the live data. Since we have information about the data, and target variable. Further, the method which is applied by default to create or to know the neighbors is based on the Euclidean distance. There are also a few more techniques, for example Manhattan which we can apply, experiment, and see which one gives us the best results. Generally K gets decided based on the square root of data points. We can also experiment with the K values close the the square root of the data. Then by analyzing the results we will learn what is the optimized value for K. Further, we need to normalize the data before applying it into the algorithm.

There is almost no assumption about the data in KNN modeling. The only assumption is proximity, meaning similar instances should have similar class values for regression. It is a nonparametric approach that lets the data speak for itself. We do not fit any distribution to the data in KNN method. The downside of this method is that we need to handle values to make the algorithm work. KNN method is sensitive to class-outliers and lots of irrelevant attributes unlike decision trees. Moreover, it is computationally expensive [9].

4.2 Data Modelling

Machine learning algorithms were imported from the scikit package in Python to perform analysis. The x_train (independent) variable was assigned with all features in the main data frame except for Weekly Sales, while the weekly_sales_train (dependant) variable was assigned with the Weekly Sales feature. The split chosen between training and testing data was 80/20, meaning 80% of x_train and weeklu_sales_train will be used to train the model (since the data was already present). The remainder of the 20% data is used to test the model and provide a score as to how accurate the model was.

Refer to Figure 12 for the train/test split command. Note, this splitting methodology was used for all algorithms and scripts, to provide consistency and allow the algorithm/model to be the only variable when judging which model was more accurate.

```
weekly_sales_train = x_train['Weekly_Sales']
x_train = x_train.drop(['Weekly_Sales'], axis=1)

#Split data into train/test
x_train,x_test,weekly_sales_train,weekly_sales_test=\
train_test_split( x_train, weekly_sales_train,\
test_size=0.20, random_state=0)
```

Figure 12: Splitting Train/Test Data

4.2.1 KNN Model Analysis

The KNN approach was initially chosen due to simplicity, with ten nearest neighbours chosen. However, due to the poor results initially, 15 nearest neighbors were adopted.

In order to create the model, the independent/dependent training variables (x_train/weekly_sales_train) were used to fit a KNN model. Once the model was available, the independent testing variable (x_test) was used to predict upcoming sales for the corresponding data, using the predict() command. Then the score, or accuracy of the point can be calculated (Figure 13), whilst also graphing predicted sales values against actual sales value from weekly_sales_test (Figure 14). The first model training was done as is with given features, without removing or tweaking any features.

```
#Using KNN Regression Model
knn_test = KNeighborsRegressor(n_neighbors=15, n_jobs=4)

#Fit training values to generate model
knn_test.fit(x_train,weekly_sales_train)

#Predict weekly_sales based off sample random test data
#Ideally should be a straight line
predicted_sales = knn_test.predict(x_test)
print(knn_test.score(x_test, weekly_sales_test))
```

Figure 13: KNN Model Training

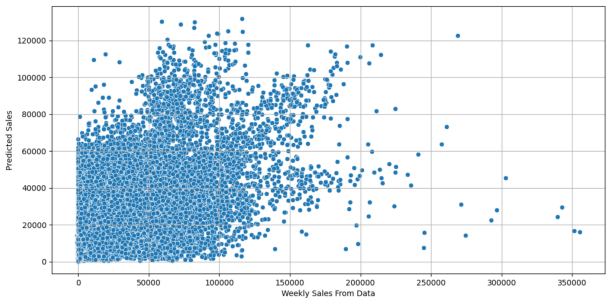

Figure 14: KNN Model (Predicted Sales vs. Actual Sales)

The score for the first KNN fit 35.8%, where 100% would yield a completely linear slope, meaning there was weak correlation between the predicted and actual weekly sales values. This where some edits were made to the features to obtain stronger correlation.

4.2.1.1 KNN Feature Engineering

In order to improve the results, the features were observed for their represented value in the dataset. It is quite possible that some features would simply throw off the correlation due to them being irrelevant. As such, in order to edit the KNN model, the Day, Year, data before 2011, and any redundant columns created as a result of previous data analysis, were dropped from the dataset, or now referred to as the x_train values. Please refer to Figure 15 for the list of commands to do this.

```
x_train = main
x_train = x_train.drop(['Unnamed: 0'], axis=1)
x_train = x_train.drop(['Day'], axis=1)
x_train = x_train.drop(['Year'], axis=1)
#x_train = x_train.drop(['CPI'], axis=1)
x_train.drop(x_train.loc[x_train['Year']<2011].index,\
inplace=True)
```

Figure 15: List of Dropped Features for KNN Edit

The first column that was dropped, as mentioned before, was simply done to reduce any latencies or redundancies in data analysis or within the dataset itself. The Day feature was dropped because the day number of the year will not show any direct correlation to weekly sales. The same logic applies to the Year

feature. The CPI feature was initially dropped, however due to a worsening of final results, it was kept. Finally, all data before 2011 was dropped since it was either missing or incorrect. Once again, this would throw off training the model.

4.2.1.2 KNN Final Results

Once the model was fitted, the KNN prediction was once again generated, with a score of 37.8% (Figure 16).

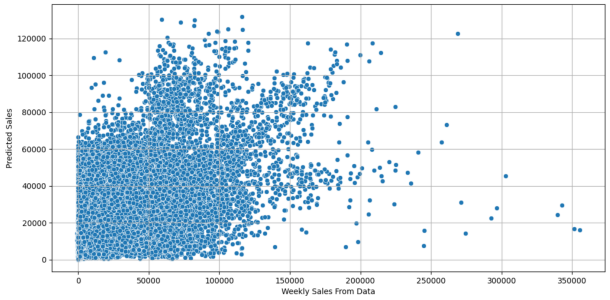

Figure 15: KNN Model Edit (Predicted Sales vs. Actual Sales)

The score however is not much better than before the edit. The three criteria that were used to judge the model were MAE (Mean Absolute Error), MSE (Mean Squared Error), and RMSE (Root Mean Squared Error).

The MAE represents the absolute error value between predicted and actual values, with this current model having a MAE of 11035.06. That translates to the average difference in dollars for predicted weekly sales against actual weekly sales was that amount.

The MSE allows us to understand that if there were a line of regression passing through this data set, how distant it would be from the data points. The MSE for this dataset is also quite large at 320087846.66. The RMSE is similar however the value is square rooted, with this model returning a RMSE of 17890.99 [14]. These commands can be shown in Figure 16 below.

```
print("MAE: " + str(mean_absolute_error\
(weekly_sales_test,\predicted_sales)))
print("MSE: " +str(mean_squared_error\
(weekly_sales_test, predicted_sales)))
print("RMSE: " + str(numpy.sqrt(mean_squared_error\
(weekly_sales_test, predicted_sales))))
```

Figure 16: KNN Error Calculations (MAE, MSE, RMSE)

In short, the KNN regression model was not a good model to use our dataset to train on. This can be seen through the weak score, and the very high magnitude of errors returned from the analysis.

4.2.2 Decision Tree Model Analysis

The next algorithm we turned our attention to was the Decision Tree. The same methodology as before was adopted, but the commands this time reflect plotting a decision tree. Firstly, simply the decision tree model, without any changes to the features, was generated (Figure 17).

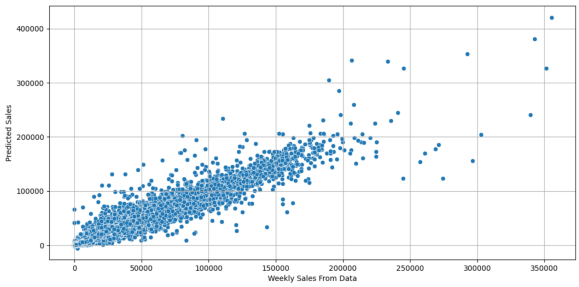

Figure 17: Decision Tree Model (Predicted Sales vs. Actual Sales)

Visually observing the graph, it can be seen to be much more linear and correlated than both KNN models, returning a score of 95.78%. The errors for this model were MAE: 1861.32, MSE: 21682219.34, and RMSE: 4656.42.

4.2.2.1 Decision Tree Feature Engineering

To harness improved results, we also edited the Decision Tree model, the same way we edited the KNN model, by dropping features, Day, Year, and any redundant columns. The results returned much stronger, with a very linear fit (Figure 18). The score

for this model was 99.99%, with the errors being MAE: 1.486, MSE: 3547.50, and RMSE: 59.56. The MAE especially is very promising, as that reflects that the magnitude of dollar amount that was different between the average predicted weekly sale and actually weekly sale, was only $1.47 approximately.

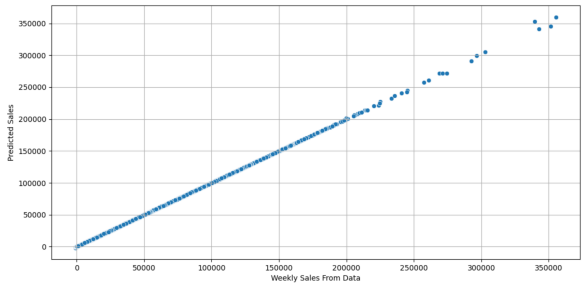

Figure 18: Decision Tree Model Edit (Predicted Sales vs. Actual Sales)

4 Conclusion

In conclusion, it was observed that the Decision Tree algorithm seemed to be a better fit for the data than the KNN model. The Decision Tree is also advantageous due to its quicker execution and model fitting, when compared to the KNN model, as it is less demanding on the memory and processor.

The KNN Regression model trains based on similarities of input features when predicting values, however in a dataset where many values are N/A or subsequently set to 0, the KNN model can be deemed inaccurate.

Using our model, predicting future sales, not only for Walmart, but any retailer could prove to be quite advantageous. Using datasets to predict future sales will be key for future Data Scientists working in the retail industry to provide large superstores with information to raise profits and minimize loss.

ACKNOWLEDGMENTS

We would like to thank the organizers of Data Day for allowing us to present our Walmart Forecaster Project, as a part of the DATA 5000 course. We would also like to thank Professor Michael Genkin for his dedication and support during the semester, as well as Professor Komeili and Professor Velazquez for teaching the course.

REFERENCES

[1] Fildes, Robert, Shaohui Ma, and Stephan Kolassa. "Retail forecasting: Research and practice." *International Journal of Forecasting* (2019).

[2] Aburto, Luis, and Richard Weber. "Improved supply chain management based on hybrid demand forecasts." *Applied Soft Computing* 7, no. 1 (2007): 136-144.

[3] Seaman, Brian. "Considerations of a retail forecasting practitioner." *International Journal of Forecasting* 34, no. 4 (2018): 822-829.

[4] Chu, Ching-Wu, and Guoqiang Peter Zhang. "A comparative study of linear and nonlinear models for aggregate retail sales forecasting." *International Journal of production economics* 86, no. 3 (2003): 217-231.

[5] Harsoor, Anita S., and Anushree Patil. "Forecast of sales of Walmart store using big data applications." *International Journal of Research in Engineering and Technology* 4, no. 6 (2015): 51-59.

[6] Sharma, Suresh Kumar, and Vinod Sharma. "Comparative analysis of machine learning techniques in sale forecasting." *International Journal of Computer Applications* 53, no. 6 (2012).

[7] Kaggle.com. 2014. Walmart Recruiting - Store Sales Forecasting | Kaggle. [online] Available at: https://www.kaggle.com/c/walmart-recruiting-store-sales-forecasting/data [Accessed 31 January 2021].

[8] raam.raam, a., 2021. *Decision Trees for Machine Learning*. [online] Devopedia. Available at: <https://devopedia.org/decision-trees-for-machine-learning> [Accessed 19 April 2021].

[9] Medium. 2021. *A Quick Introduction to K-Nearest Neighbors Algorithm*. [online] Available at: <https://blog.usejournal.com/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7> [Accessed 19 April 2021].

[10] Patricia S. Abril and Robert Plant, 2007. The patent holder's dilemma: Buy, sell, or troll? *Commun. ACM* 50, 1 (Jan, 2007), 36-44. DOI: https://doi.org/10.1145/1188913.1188915.

[11] Sten Andler. 1979. Predicate path expressions. In *Proceedings of the 6th. ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL '79)*. ACM Press, New York, NY, 226-236. DOI:https://doi.org/10.1145/567752.567774

[12] Ian Editor (Ed.). 2007. *The title of book one* (1st. ed.). The name of the series one, Vol. 9. University of Chicago Press, Chicago. DOI:https://doi.org/10.1007/3-540-09237-4.

[13] David Kosiur. 2001. *Understanding Policy-Based Networking* (2nd. ed.). Wiley, New York, NY.

[14]Statistics How To. 2021. *Mean Squared Error: Definition and Example - Statistics How To*. [online] Available at: <https://www.statisticshowto.com/probability-and-statistics/statistics-definitions/mean-squared-error/> [Accessed 18 April 2021].