Prediction of Monthly Sales in the US Census Data

Irene Siemen

6/10/2020

Introduction

The US Census gathers monthly data of the country's total sales and inventory for manufacturers, merchant wholesalers, and retailers. This project focuses on analyzing the dataset and then creating predictions for the total business monthly sales. The dataset contains records dating back to January 1992. Each month and year contains values for monthly sales, inventory, percent change of sales, percent change of inventory, and the inventory to sales ratio. Throughout the data all of the dollar values are reported in millions. The total business monthly sales predictions were generated by creating and training a machine learning algorithm using a subset of the data. Then, the model calculated predictions on the validation dataset. The success of the machine learning algorithm was evaluated using the root mean square error (RMSE).

The first step was to complete an exploratory data analysis (EDA) on the training data set to review the data. The EDA included reviewing the characteristics of the variables, checking for NaNs, performing filters and counts, and graphing the relationships between variables. The EDA revealed trends in the data, such as increasing sales values over time, and these trends were used as factors in the machine learning models. Once the data was put into a tidy format, then a linear model was created to determine predict the monthly sales based on the month and year. Next, a K-Nearest Neighbors (KNN) model was created on the train dataset to calculate more accurate predictions. The fitted KNN model was used to create predictions on the test data and returned a desirable RMSE value. Finally, the KNN model created predictions for the validation dataset and calculated the final RMSE value.

Methods and Analysis

Downloading the data The US Census Manufacturing and Trade Inventory and Sales data can be downloaded directly from their website at: https://www.census.gov/econ/currentdata/datasets/MTIS-mf.zip

```
#Download the dataset
dl <- tempfile()
download.file("https://www.census.gov/econ/currentdata/datasets/MTIS-mf.zip", dl)

#Access the data files:
original <- read.csv(unzip(dl, "MTIS-mf.csv"),
                     sep = ",",
                     col.names = c('idx','code','desc','col4','col5','col6','col7'),
                     fill = TRUE,
                     blank.lines.skip = TRUE,
                     nrows =  19400,
                     header = FALSE,
                     stringsAsFactors = FALSE
                   )
```

Preparing the data The original data was stored in a .csv file. The file had 8 tables listed on one sheet. The first 7 tables contained codes and descriptions. The last table contained the actual data and referenced codes from the first 7 tables. The data needed to be transformed into tidy data. The first step was to store each table as a separate data frames.

```
## Warning in stri_extract_first_regex(string, pattern, opts_regex =
## opts(pattern)): longer object length is not a multiple of shorter object length
```

The next step was to check for NaN values and to assign the correct data type for each columns. Then, the data frames containing the codes and description were connected to the primary data using a left join.

```
#Assign the correct data types to each column
data<-data%>%mutate(per_idx = as.numeric(per_idx),
                    cat_idx = as.numeric(cat_idx),
                    dt_idx = as.numeric(dt_idx),
                    et_idx = as.numeric(et_idx),
                    geo_idx = as.numeric(geo_idx),
                    is_adj = as.numeric(is_adj),
                    value = as.numeric(value))

#Remove NaN values
data<-data[complete.cases(data), ]

#Use a left join to combine the codes tables with the actual data table.
data<-left_join(data,timeperiods, by = "per_idx")
data<-left_join(data,categories, by = "cat_idx")
data<-left_join(data, datatypes, by = "dt_idx")
data<-left_join(data, errortypes, by = "et_idx")
```

Every time period originally had multiple rows because there was a separate row for each data type (sales, inventory, percent changes) and category (total business, manufacturing, merchant wholesalers, and retail). Tidy data should have one row for each time period with separate columns for the different data and category values. Additional columns were added for the pervious period's monthly sales and percent changes.

```
#Convert to tidy data so that each time period displays the all of the data types and
#category values in one row
data<-data%>%
  unite(dt_desc,cat_desc,col = "dt_cat", sep = ": ")%>%
  unite(dt_code,cat_code, col = "dt_cat_code" , sep = "_")%>%
  select(per_idx, period, month, year, dt_cat_code,value, is_adj, et_idx)%>%
  spread(dt_cat_code,value)%>%
  filter(is_adj == 0, et_idx == 0)%>%
  select(-c('NA_RETAIL','NA_TOTBUS','NA_WHLSLR'))

#Add columns for the previous period's Total Business monthly sales and percent change
data<- data%>%
  mutate(PREV_SM_TOTBUS = lag(SM_TOTBUS),
         PREV_SM_MNFCTR = lag(SM_MNFCTR),
         PREV_SM_RETAIL = lag(SM_RETAIL),
         PREV_SM_WHLSLR = lag(SM_WHLSLR),
         PREV_MPCSM_TOTBUS = lag(MPCSM_TOTBUS),
         PREV_MPCSM_MNFCTR = lag(MPCSM_MNFCTR),
         PREV_MPCSM_RETAIL = lag(MPCSM_RETAIL),
```

```
        PREV_MPCSM_WHLSLR = lag(MPCSM_WHLSLR))

head(data)
```

```
##   per_idx  period month year is_adj et_idx IM_MNFCTR IM_RETAIL IM_TOTBUS
## 1       1 Jan1992   Jan 1992      0      0    375389    235509    802948
## 2       2 Feb1992   Feb 1992      0      0    378146    239270    809329
## 3       3 Mar1992   Mar 1992      0      0    377223    244966    813301
## 4       4 Apr1992   Apr 1992      0      0    379047    249466    819247
## 5       5 May1992   May 1992      0      0    380525    247282    815688
## 6       6 Jun1992   Jun 1992      0      0    375933    246247    812610
##   IM_WHLSLR IR_MNFCTR IR_RETAIL IR_TOTBUS IR_WHLSLR MPCIM_MNFCTR MPCIM_RETAIL
## 1    192050      1.79      1.80      1.68      1.38           NA           NA
## 2    191913      1.63      1.82      1.63      1.44          0.7          1.6
## 3    191112      1.51      1.72      1.50      1.27         -0.2          2.4
## 4    190734      1.58      1.70      1.53      1.30          0.5          1.8
## 5    187881      1.56      1.62      1.52      1.33          0.4         -0.9
## 6    190430      1.43      1.62      1.44      1.26         -1.2         -0.4
##   MPCIM_TOTBUS MPCIM_WHLSLR MPCSM_MNFCTR MPCSM_RETAIL MPCSM_TOTBUS MPCSM_WHLSLR
## 1           NA           NA           NA           NA           NA           NA
## 2          0.8         -0.1         11.1          0.4          3.7         -4.3
## 3          0.5         -0.4          7.3          8.6          9.3         13.4
## 4          0.7         -0.2         -4.0          3.3         -1.7         -2.5
## 5         -0.4         -1.5          1.5          3.6          0.7         -3.5
## 6         -0.4          1.4          8.1         -0.4          5.2          6.4
##   SM_MNFCTR SM_RETAIL SM_TOTBUS SM_WHLSLR PREV_SM_TOTBUS PREV_SM_MNFCTR
## 1    209438    130683    478951    138830            NA            NA
## 2    232679    131244    496844    132921        478951        209438
## 3    249673    142488    542833    150672        496844        232679
## 4    239666    147175    533768    146927        542833        249673
## 5    243231    152420    537400    141749        533768        239666
## 6    262854    151849    565572    150869        537400        243231
##   PREV_SM_RETAIL PREV_SM_WHLSLR PREV_MPCSM_TOTBUS PREV_MPCSM_MNFCTR
## 1             NA            NA                NA                NA
## 2         130683        138830                NA                NA
## 3         131244        132921               3.7              11.1
## 4         142488        150672               9.3               7.3
## 5         147175        146927              -1.7              -4.0
## 6         152420        141749               0.7               1.5
##   PREV_MPCSM_RETAIL PREV_MPCSM_WHLSLR
## 1                NA                NA
## 2                NA                NA
## 3               0.4              -4.3
## 4               8.6              13.4
## 5               3.3              -2.5
## 6               3.6              -3.5
```

Partitioning the data This was a relatively small dataset because there was only one row per month dating back to January 1992. The data was partitioned into 3 sets: validation, test and train. This allowed the machine learning algorithm to be developed on the training data and then evaluated on the test and validation data. If the data was not partitioned, then the results would have been over-confident. The validation data set contained 10% of the data. The remaining 90% of the data was then partitioned again, so that test data contained 10% and the training data contained 90% of the remaining data. A 90-10 split

was selected instead of an 80-20 or 50-50 because the dataset was relatively small so it was important to train on as much data as possible.

```
#Partition out the train and validation data set.
set.seed(1)
temp_index <- createDataPartition(y = data$per_idx, times = 1, p = 0.1, list = FALSE)
train <- data[-temp_index,]
validation <- data[temp_index,]
dim(validation)
```

```
## [1] 36 34
```

```
dim(train)
```

```
## [1] 303  34
```

```
#Partition out the test data set from the training data set
temp_index <- createDataPartition(y = train$per_idx, times = 1, p = 0.1, list = FALSE)
train <- train[-temp_index,]
test <- train[temp_index,]
dim(test)
```

```
## [1] 32 34
```

```
dim(train)
```

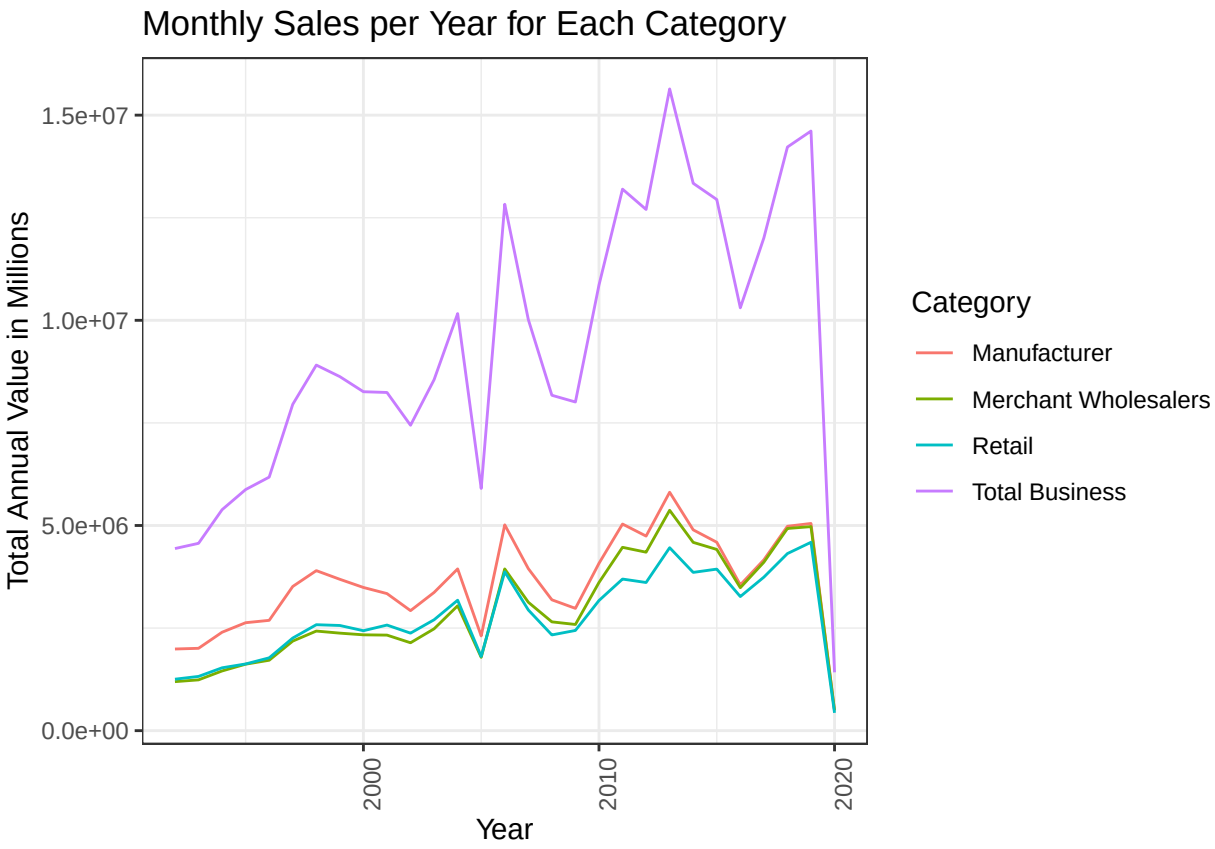
```
## [1] 271  34
```

```
#Remove NaNs
train<-train[complete.cases(train), ]
test<-test[complete.cases(test), ]
validation<-validation[complete.cases(validation), ]
```

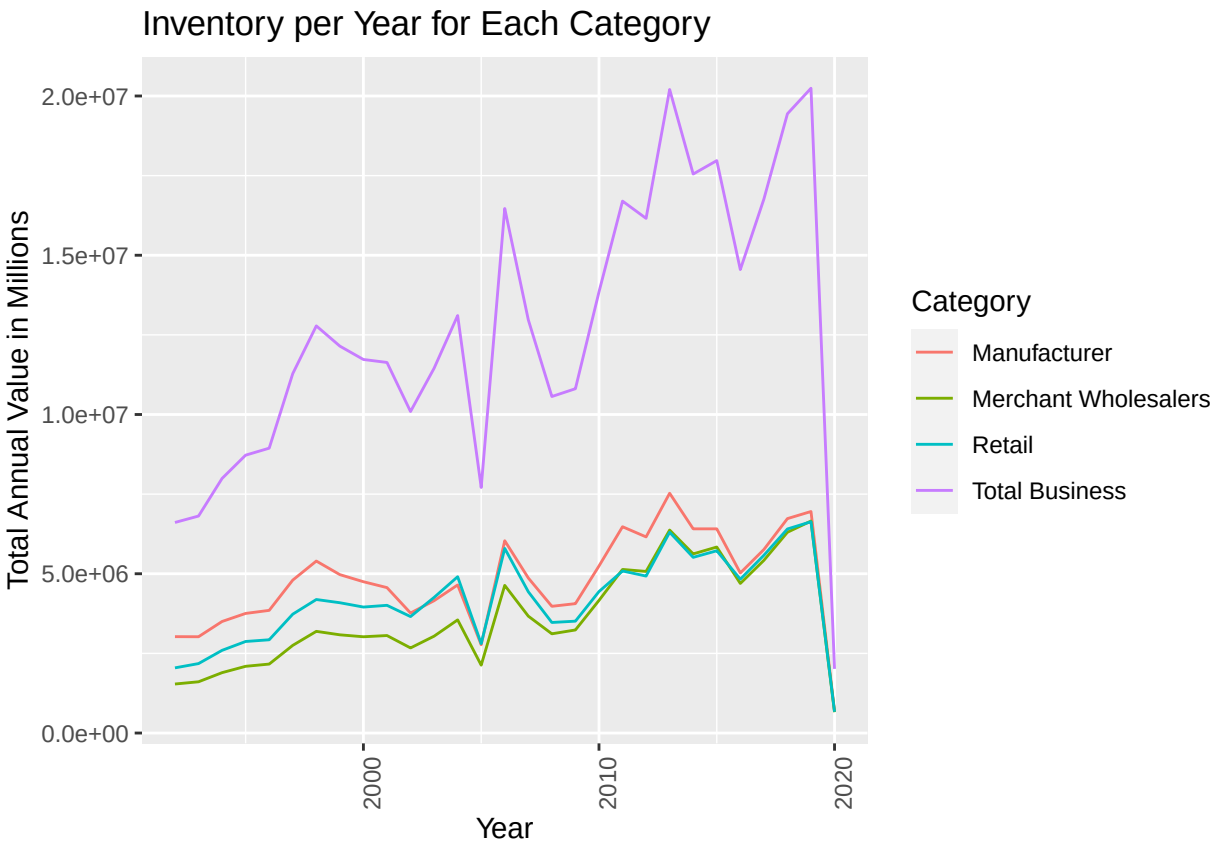
Exploratory Data Analysis The first step of the EDA was to review summaries of the training data.

```
str(train)
summary(train)

min_sales<-train%>%summarize(min = min(SM_TOTBUS))%>%pull(min)
max_sales<-train%>%summarize(max = max(SM_TOTBUS))%>%pull(max)
avg_sales<-train%>%summarize(avg = mean(SM_TOTBUS))%>%pull(avg)
```

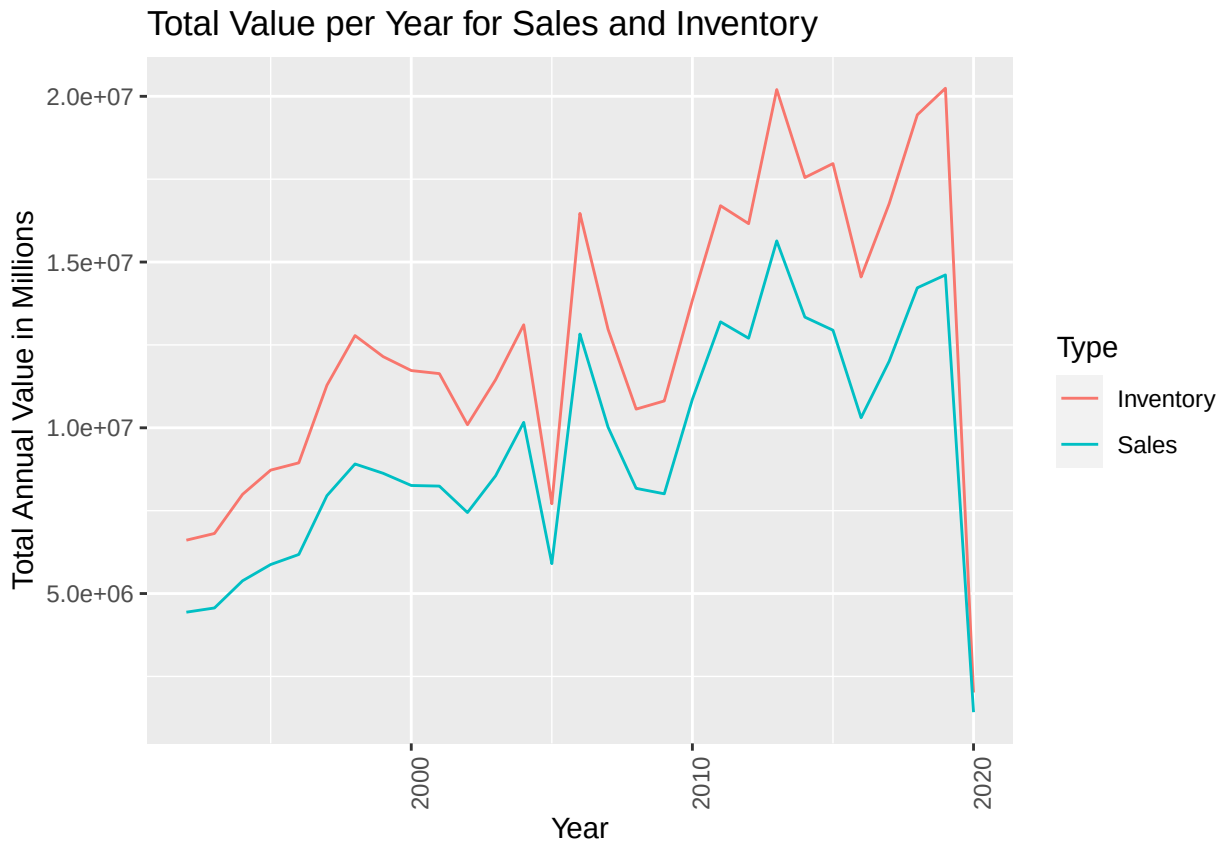
The next step was to create various plots and look for any patterns or trends in the data. A timeseries plot of monthly sales was created first. The plot has separate line for each category. Throughout the data all of the dollar values are reported in millions.


Monthly Sales per Year for Each Category
Total Annual Value in Millions
1.5e+07
1.0e+07
5.0e+06
0.0e+00
2000
2010
2020
Year
Category
Manufacturer
Merchant Wholesalers
Retail
Total Business

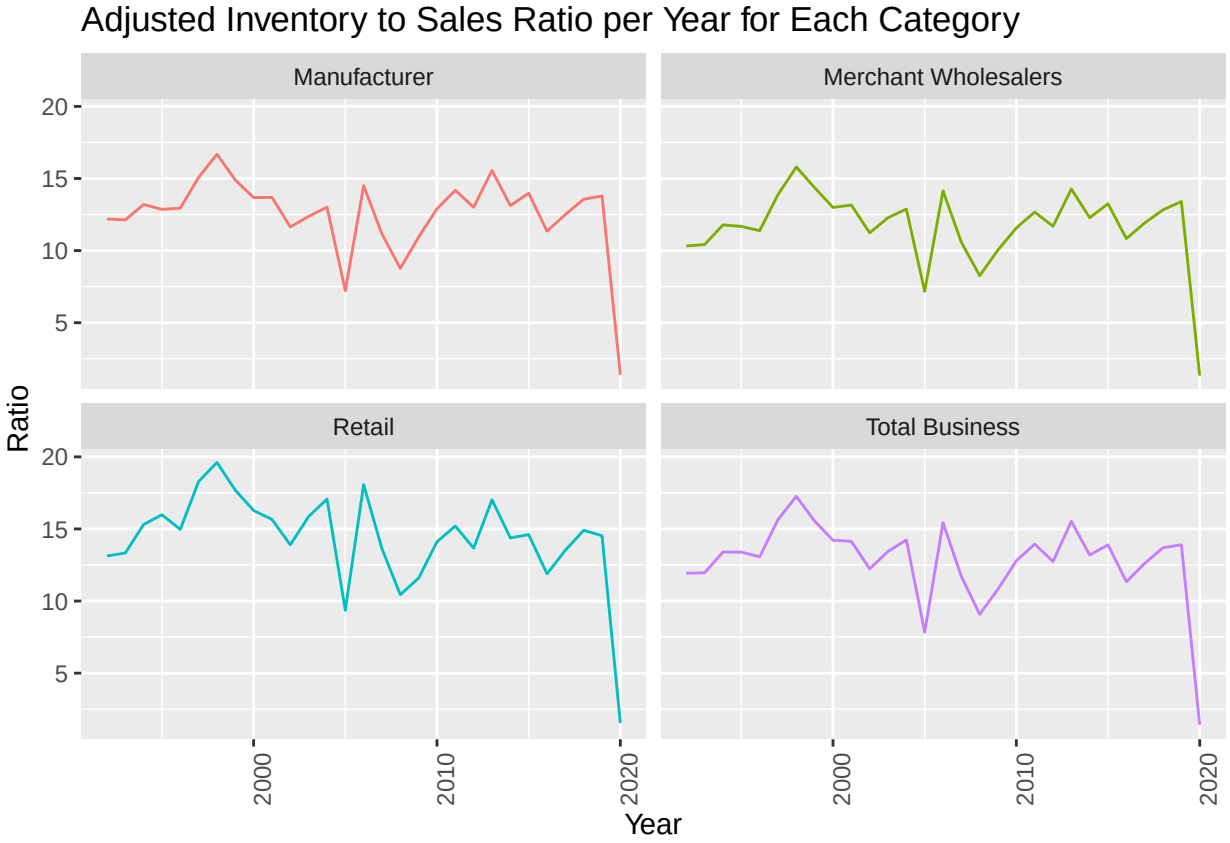
A similar timeseries plot was created displaying inventory values instead of sales.



A third timeseries was created to compare the inventory and sales values for the total business category.

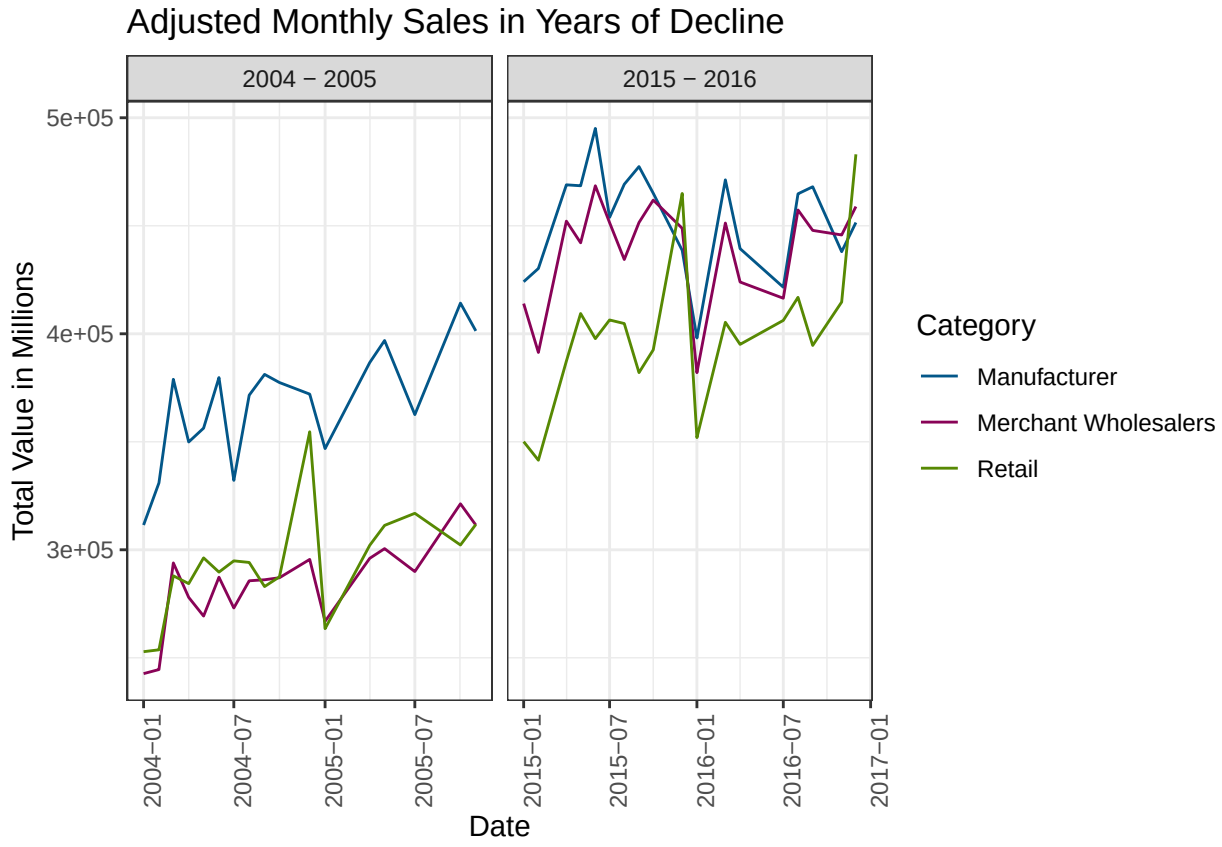


The data also contains the inventory to sales ratio. This was included in a timeseries and faceted by category because the chart was too messy when all of the categories were on the same plot.

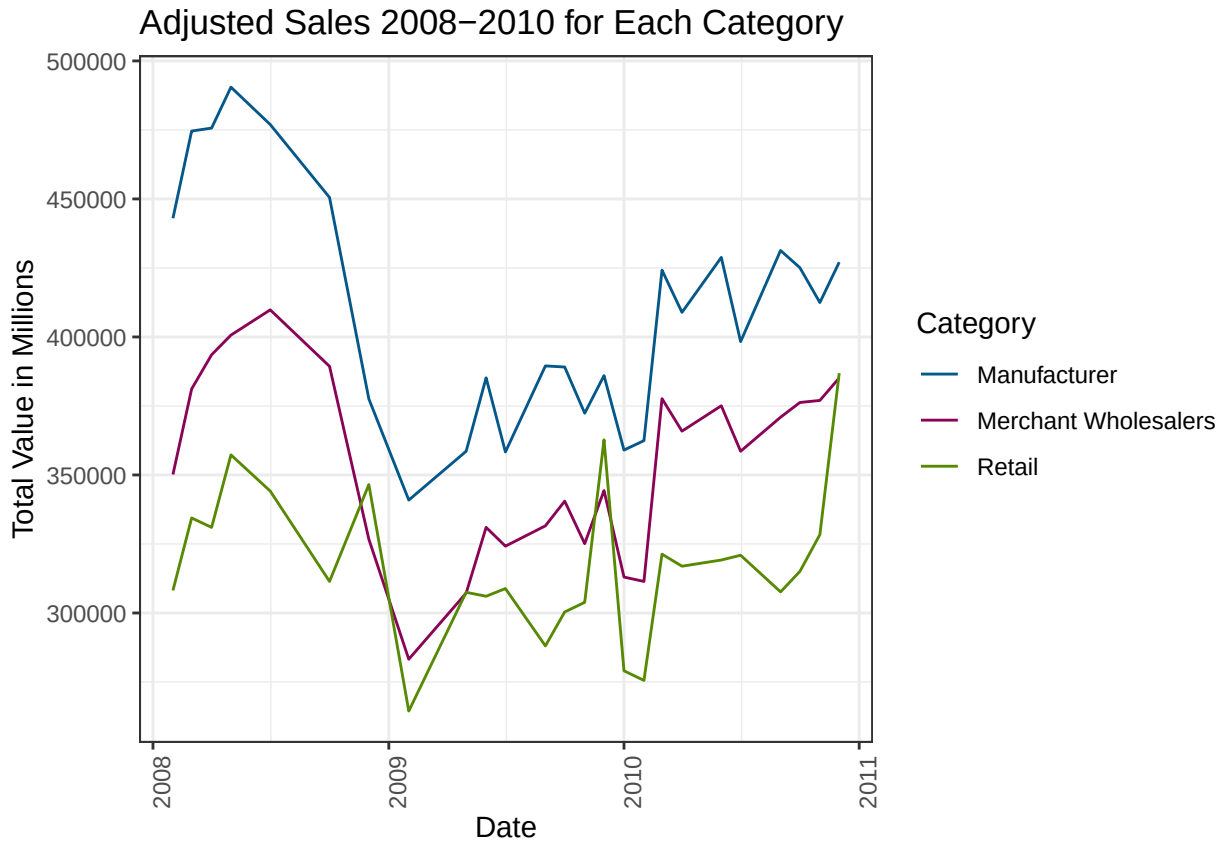


The timeseries plots revealed an overall upward trend in both sales and inventory as time progressed. It also showed how drastically low the sales and inventory have dropped in 2020. However, the drop was a bit misleading because it only contains data for the first 3 months in 2020 and the previous years contain data for all 12 months.

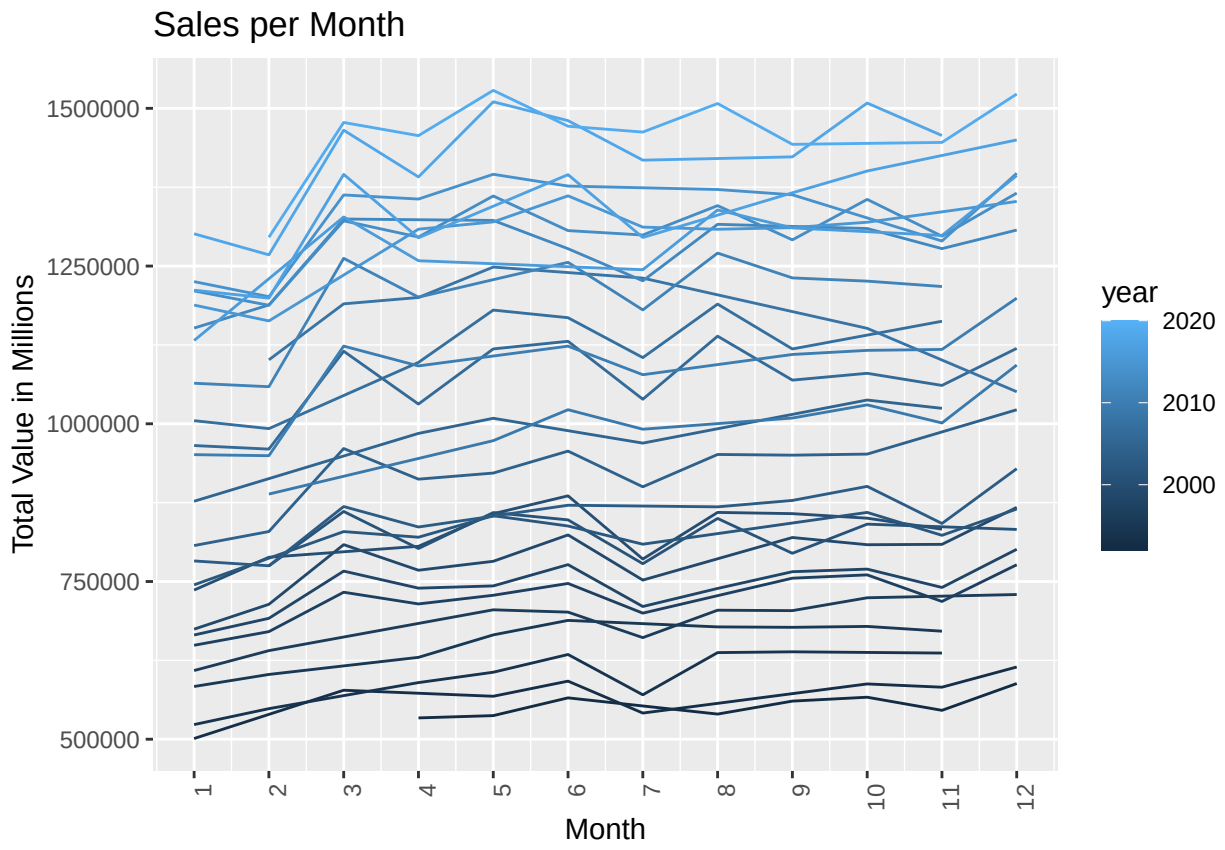
The next step in the EDA explored trends during previous declines. The sales data showed temporary declines in 2004 and 2015, and a prolonged decline between 2008-2010. A timeseries was created to the compared the 2004 and 2015 data.



A separate timeseries was created just for the 2008 - 2010 data. In both plots, it appeared that retail sales increased after a period of decline quicker that manufacturers and merchant wholesalers.

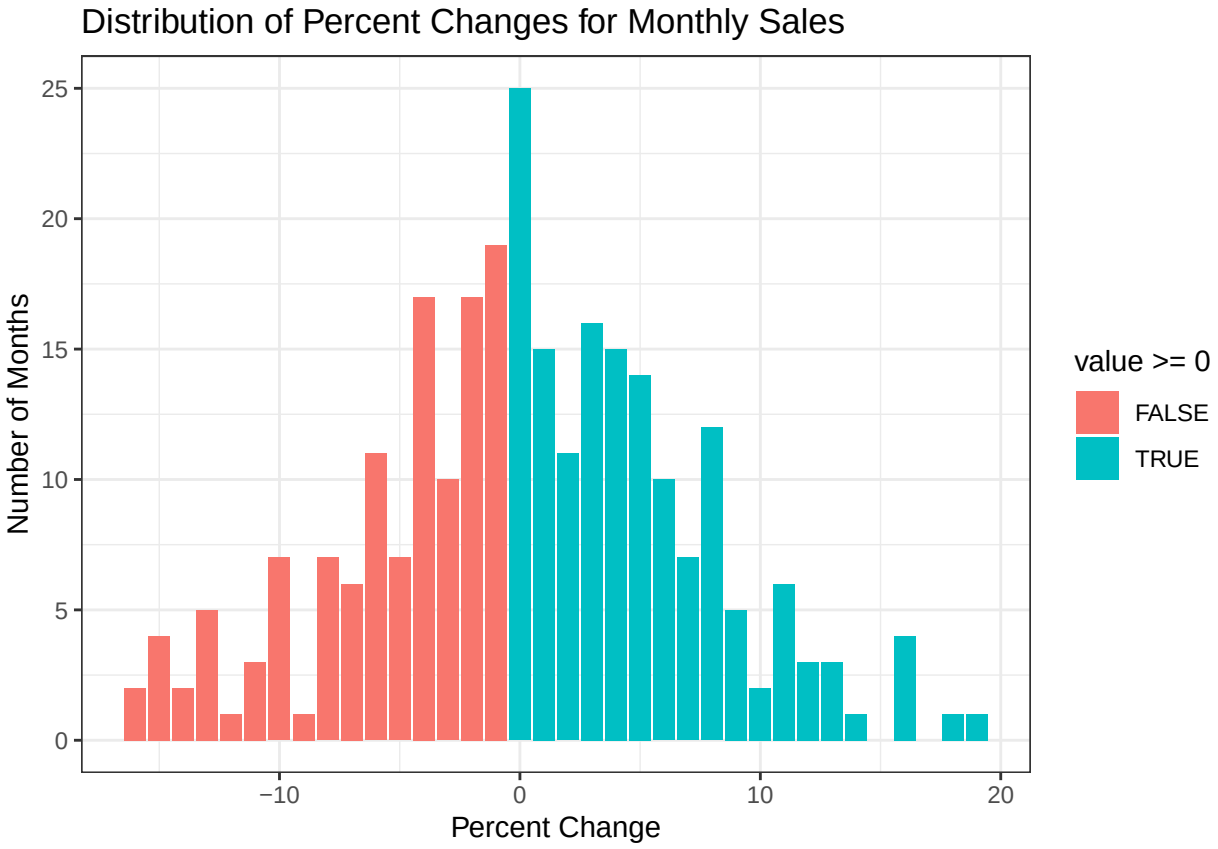


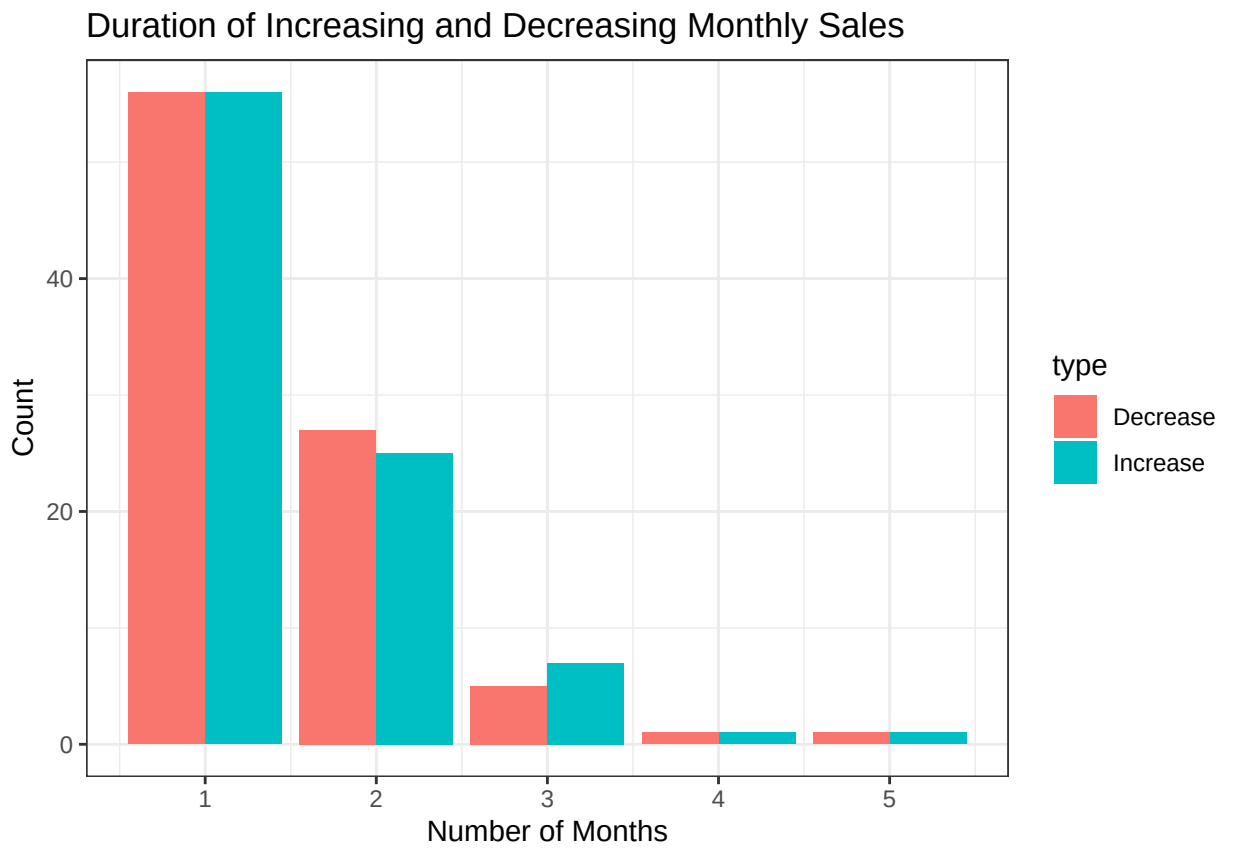
Next, a plot was created to see if seasonality was a factor of the monthly sales. The plot showed that there tended to be an increase in March, decrease in July, and increase in August.



A scatterplot was created to see the correlation in monthly sales between the categories of manufacturer, retail, and merchant wholesalers. The retail data has the weakest correlation with the total business sales.

The final plots in the EDA were created to explore the percent change values for monthly sales. The percent change values represent the percentage that the current month's sales have increased or decreased from the previous month. First, a histogram was created to see the distribution the of the percent change values. Then, a second histogram was created to explore the duration of increasing and decreasing percentage changes.





Machine Learning Models After completing the EDA, the next step was to apply machine learning models to the training data. The EDA revealed an overall upward trend in sales over time. This indicated that month and year were strong factors for predicting the total business monthly sales. In the real world, it is only possible to look at data from previous months when predicting sales for the current month. Therefore, all factors were removed from the dataframe with the exception on the period (month and year) and the previous month's values for monthly sales and monthly percent changes.

First, a linear model was fit to the training data and predicted the total business monthly sales based on the period. The model was then applied to the test dataset to created predictions.

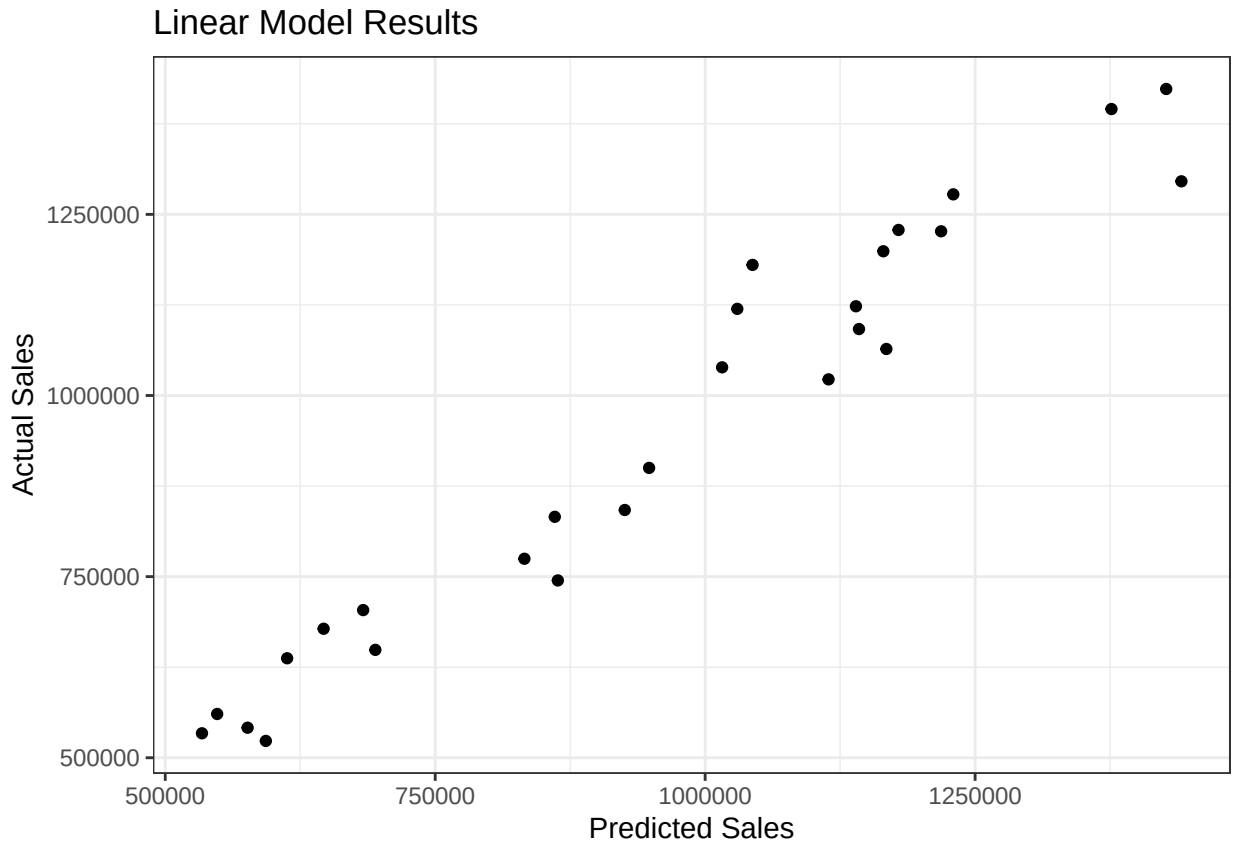
```
#fit a linear model to determine the affect of time on the total monthly sale
fit_lm <- lm(SM_TOTBUS ~ per_idx, data = train)
fit_lm$coefficients
```

```
## (Intercept)     per_idx
##      522724        2817
```

```
#predict the total monthly sales based on the time period
y_hat_lm <- predict(fit_lm, test)
```

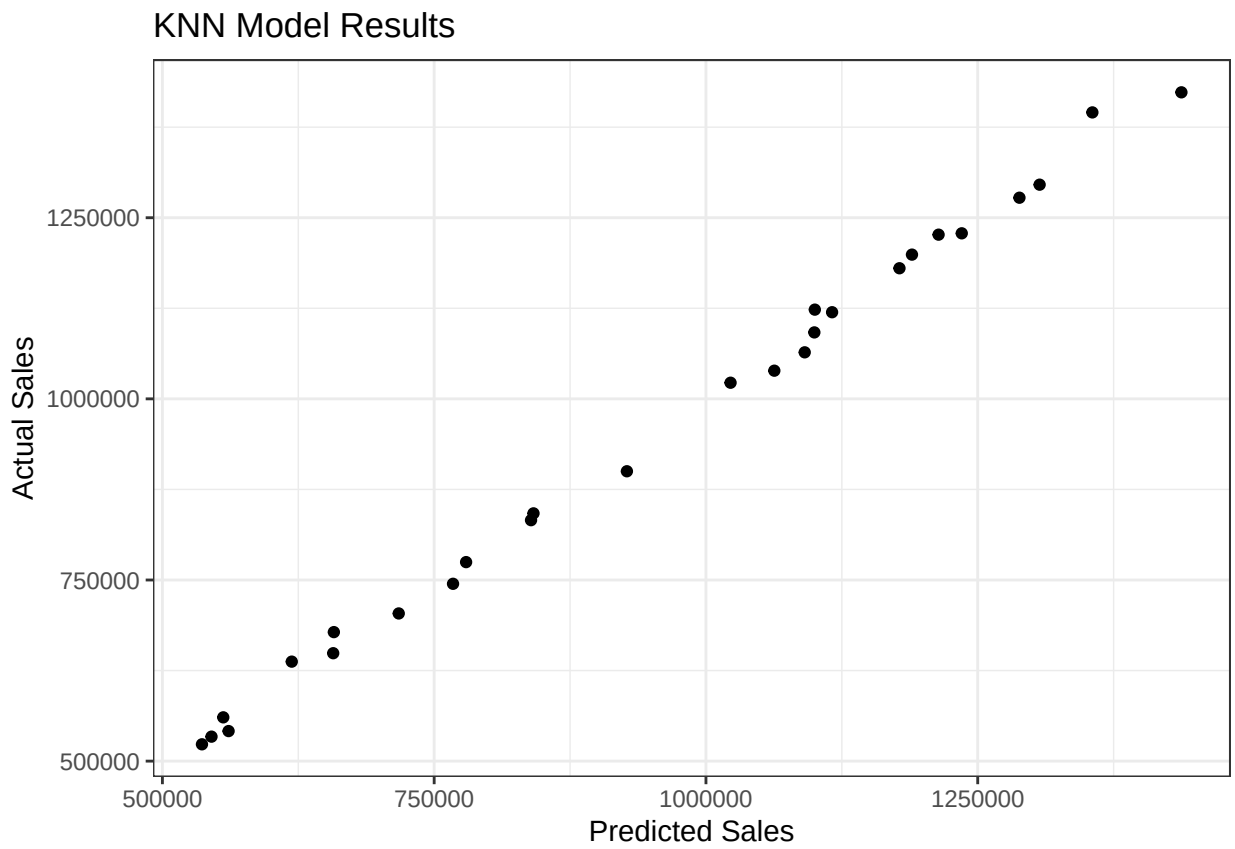
The predictions were evaluated by calculating the Root Mean Square Error (RMSE) between the predicted and actual values for the total monthly sales. A scatterplot was also created to visualize the relationship between the predictions and actual values.

```r
#RMSE calculation to evaluate predictions
RMSE <- function(true_ratings, predicted_ratings,n){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}

time_RMSE<-RMSE(test$SM_TOTBUS,y_hat_lm)
```



A second model was created using the K-Nearest Neighbors method in order to get more accurate predictions. The K-Nearest Neighbors method compares 'k' number of records with similar values (i.e. closest distance) to create the predictions. This results in a more flexible prediction than a linear model. Initially, 'k' was set to 5.

```r
#Fit the knn model on the train data and evaluate on the test data
fit_knn<-knn.reg(train = train, test = test, y = train$SM_TOTBUS, k = 5)

#Evaluate the RMSE
knn_RMSE<-RMSE(test$SM_TOTBUS,fit_knn$pred)
```



A range of 'k' values from 3 - 100 was evaluated to make sure the optimal value of k is being used. This showed that 3 had the lowest RMSE value, however 'k' was set to 5 in the final model to prevent over-fitting.

```
#Evaluate a range of number of neighbors (K)
ks <- seq(3, 100, 2) #k sequence to test: odd numbers between 3-100
RMSE_results <- map_df(ks, function(k){
  fit <- knn.reg(train = train, test = test, y = train$SM_TOTBUS, k = k) #fit the knn value
  test_RMSE<-RMSE(test$SM_TOTBUS,fit$pred)
  tibble(k = k, RMSE = test_RMSE) #create a tibble with all of the train and test accuracy values
})

as.data.frame(RMSE_results)%>%filter(RMSE == min(RMSE))
```

```
##   k  RMSE
## 1 3 12036
```

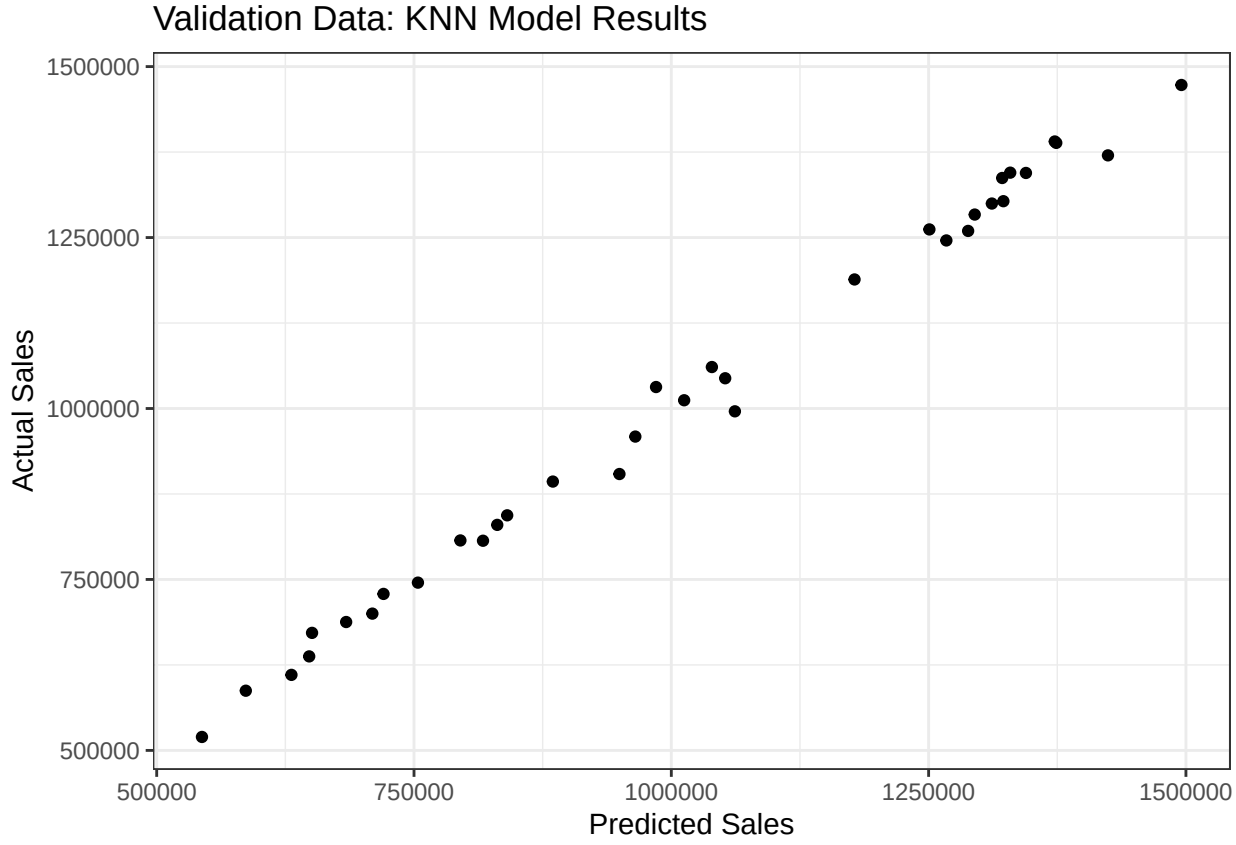
Finally, the KNN model with k = 5 was used to create predictions on the validation dataset and was evaluated by calculating the RMSE.

```
#Apply knn model to the validation set
fit_knn_val<-knn.reg(train = train, test = validation, y = train$SM_TOTBUS, k = 5)

#Evaluate the RMSE
knn_RMSE_val<-RMSE(validation$SM_TOTBUS,fit_knn_val$pred)
```

Results

The linear model predicted the total business monthly sales based solely on the time period and was able to create predictions with a RMSE value of 6.516×10^4 million USD. This is a decently small RMSE given that the average monthly sales value is 1.003×10^6 million USD. However, a more accurate model was achieved using the KNN method which considered the previous period's values in addition to the date.

The final KNN model had a RMSE equal to 2.258×10^4 million USD. The below scatterplot shows the correlation between the predictions and the actual values.



Conclusion

The KNN model returned an reasonable RMSE value of 2.258×10^4. The solution was created from the exploratory data analysis. The factors for time period, previous period's monthly sales and previous period's percent changes were major contributors to the low RMSE score. The largest limitation to this project was the relatively small dataset. Additionally, the model was creating predictions solely based on monetary factors and did not consider any political or global events. In reality, predicting sales one month in advance is much simplier than a quarter or year in advance. Future work could be done to predict monthly sales for different categories and for longer date ranges. Other machine learning models could be applied and potentially provide a lower RMSE.