Project Report: Sales in Stormy Weather

Group Members

Wanxing Dai

Ke Deng

Ruopu Fan

Ran Pei

Introduction

In this project, we are going to predict the unit sales of weather-sensitive products at Walmart stores. We will use four csv files retrieved from a Kaggle competition by Walmart. Data files include two pre-split training and testing sets, a key and a weather file. The train data set records sales of 111 different products sold in 45 Walmart stores across different areas. The key data indicates the number of weather stations shared by each store. The weather data records local weather information for each station at specific days, including variables like average temperature, dewpoint, snowfall, precipitation, etc. The test data set excludes the unit sales and will be used to make predictions.

In this competition, Walmart asked for sales prediction of 111 products that are potentially weather-sensitive. As Walmart is operating numerous stores around the world, it is important to conduct systematic data analysis that can effectively estimate the sales pattern around extreme weather events. An accurate estimation can help the stores replenish inventory of a certain good before any severe weather event that can increase the sales, so that the customers in need will not be disappointed when visiting the stores.

To predict the sales, we will first visualize some of the predictors and discuss their patterns. We will then construct a multiple linear regression model with chosen variables. By merging the train, key and weather data sets, we will be able to combine all the information we need to run the regression and make estimation. Plus, we will make improvements based on the initial model, methods including adding interaction, stepwise selection, Poisson regression, LASSO regression, KNN and TREE model, Random Forest, and adding new variables (Black Friday and weekends). In the final step, we will use our best model to predict the missing sales in the test set and present the Kaggle score to prove our estimation.

Exploratory Data Analysis

Data Pre-processing

When going through the weather data file, we find that there are “M” (missing value) and “T” (trace) in some variables. Meanwhile, the variable classes in the original weather data are all factors. Thus, in order to get a more precise model, we start with cleaning the data file.

Since “date” can be an important variable for us to use, we first change the date category by using *as.Date* function. In this way, we can extract year, month and day easier. In the cleaning process for variable *snowfall* and *preciptotal*, although both contain “T” and “M”, the data structure of “T” value is different since there are two blanks before “T”. Therefore, we use *gsub* function to delete the blanks and to turn this vector into the same length with other vectors. Also, the “0” elments in *snowfall* are represented as “0.0” and “0” vectors in *preciptotal* are represented as “0.00”. For the convenience for our future model processing, we change both “0.0” and “0.00” to “0”. We also convert “M” to “NA” and “T” to “0”. Furthermore, we choose *avgspeed* (average speed), *resultspeed* (resultant wind speed) and *resultdir* (resultant direction) as our cleaning objects, with “ M” converted to “NA”. For the variables *tavg*, *tmax*, *tmin*, *depart*, *dewpoint*, *wetbulb*, and *sealevel*, converting “ M” to “NA” is the only step we use for cleaning these three variables since there are only some missing values. Meanwhile, because the “heat” and “cool” variables in this data are not just the normal temperature but defined in a special basis: “the degree in a day based on 65F”, we believe it might be helpful for our model. Thus, we apply the same cleaning process for them too.

Before the modelling process, we plan to go through again with “train” data file to check if there are other variables in “train” can have connection with the variables in “weather” data file. We find that some stores are located with different stations in “train” and there is a variable called *stnpressure* (station pressure) in “weather”. Thus, we also clean this variable for predicting the sales units in our model.

Data Visualization

To look at the relationship among unit sales and all other variables, we merge the datasets of train and weather using key and create the following correlation graph:

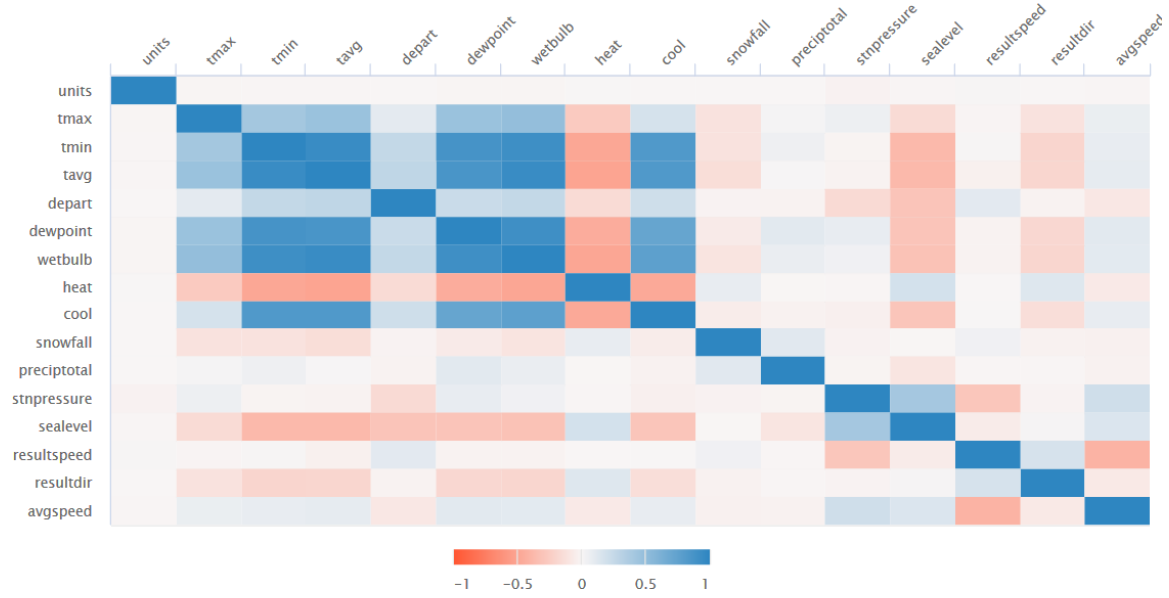

Figure 1: Correlation

The graph indicates high correlation among several groups of variables: *tmax*, *tmin* and *tavg* are highly correlated, and so do the variables of *dewpoint* and *wetbulb*. The *heat* and *cool* variables also show correlations with *tmin*, *tavg* and *wetbulb*, with an absolute value of 0.5. Therefore, it will be necessary to drop some of these variables when building models to avoid multicollinearity.

From the bar plot of precipitation (Figure 2), we can see that there is more rain during spring and summer (the rainy seasons). However, there are many missing values in *snowfall*, and we can barely observe a pattern through Figure 3. This variable might explain little of the unit sales due to the large amount of missing values.

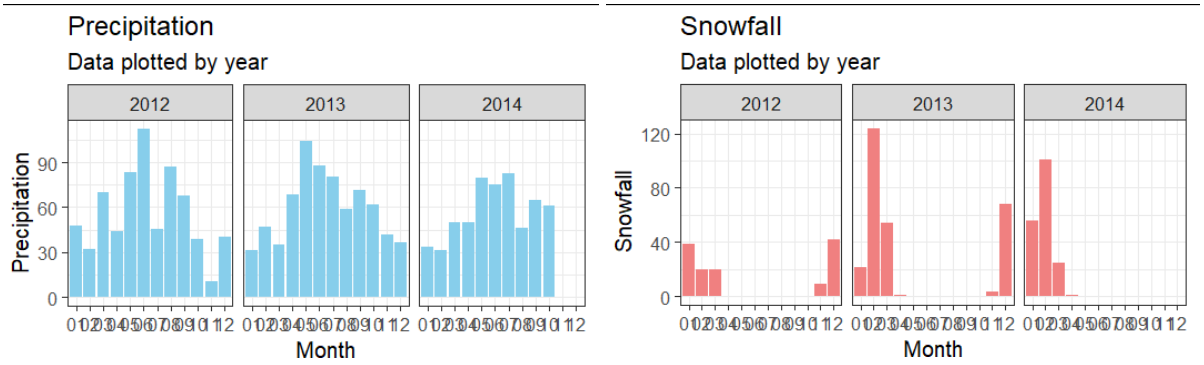

Figure 2: Precipitation / Figure 3: Snowfall

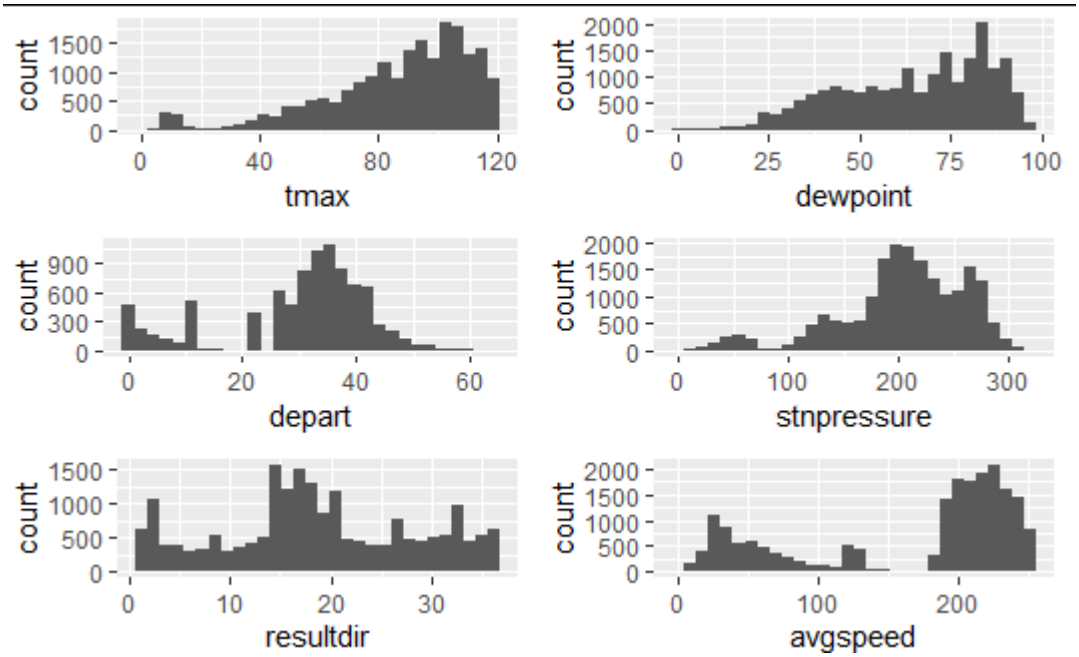

Figure 4: Histograms

Above histograms (in Figure 4) present the distributions of selected variables. We choose to display these variables because they are less correlated and may introduce more information. We can observe that the max temperature, dew point and station pressure are skewed to the left. The distribution of departure from normal assembles a normal one in the center, with several values on the left tail (most of them are zeros). The resultant direction of wind tends to cluster in the middle, while the average wind speed is distributed more randomly.

We also graph the sum of sales by month has been going down from 2012 to 2014 (Figure 5). From the graph, the sum of sale units reaches the highest in the first month, declines until the midyear, rises again by a little and continues to go down until the end of the year.

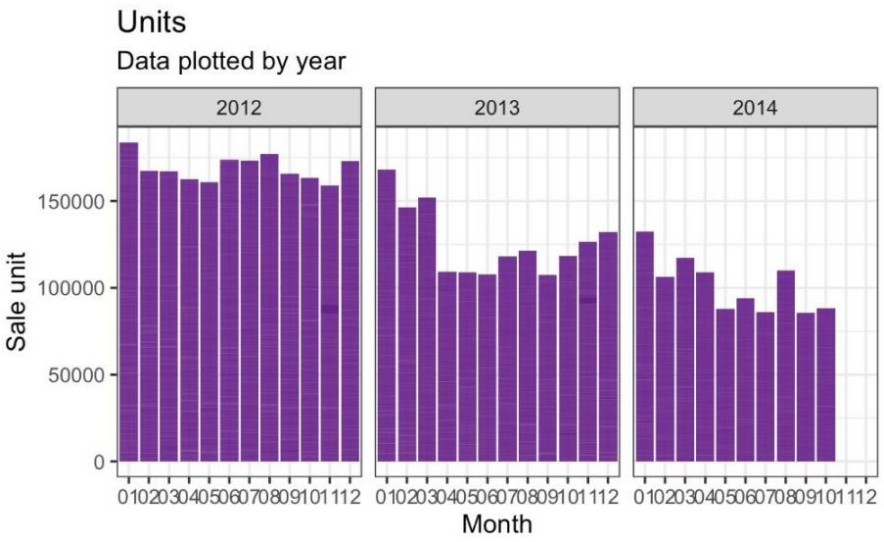

Figure 5: Units

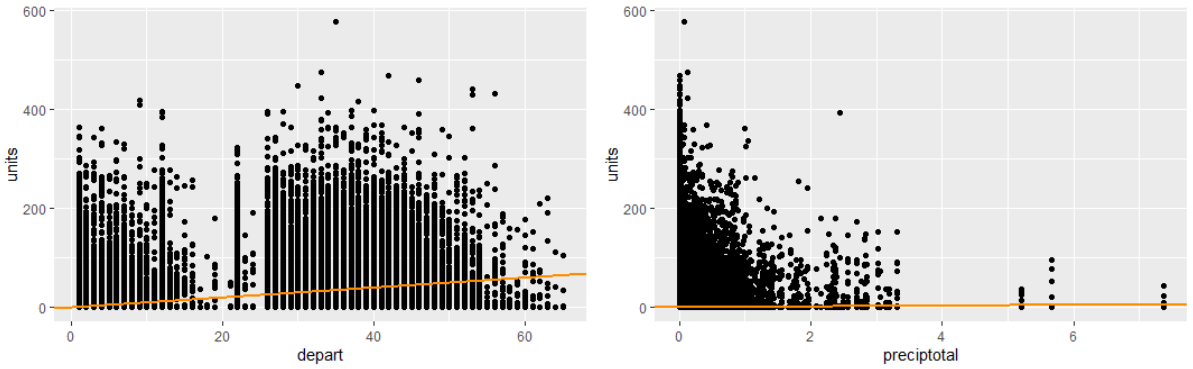

Figure 6: Units versus depart / Figure 7: Units versus preciptotal

We create the scatter plots of unit sales and a couple of weather variables to check the linear relationships. We observe an overall increasing pattern on the units versus depart plot (Figure 6), showing a potential relationship between these two variables. From Figure 7, we can see that many of the precipitation data are close to or equal to zeros. There are certain high sales with very low precipitation. Yet the trend line is very close to zero, showing little information on their relationship.

Linear Regression Model & Diagnostics

Merging Data

We apply the *tidyverse* package and use the *left_join* function to firstly merge the key and weather data by station number. Then we combine the train and merged dataset by date and store number to complete the data. We create a new test data in the same way.

Interpretation of the First OLS

Before building a multiple linear regression model, we first subset the nonzero sale units. The unit sale column includes a large number of zeros, indicating that either values are missing or there is no sale that day. Customers might not buy certain goods normally and thus the sales of those items remain the state of zero in many days throughout the year. So, according to our understanding, it would be better to only use the nonzero sales for more accurate prediction.

We regress the *units* over *date*, *store_nbr*, *item_nbr*, *depart*, *dewpoint*, *heat*, *snowfall*, *preciptotal*, *sealevel*, *stnpressure*, *resultspeed*, and *avgspeed* and get an overall significant model. In the summary of the model, we can see that most of the predictors are significant at the level of 0.05, except *heat*, and *snowfall*. The R-squared is very low, though, with a value of 0.1. The extremely low R-squared value indicates that the predictors altogether explain little of the response in our first linear model.

Among the significant predictors, date, store number and item number appear to be very significant with nearly zero p-values. If we look at the weather variables, we can see that one degree increase in the maximum temperature would cause a 0.0732-unit decrease in the item sold. Change in total precipitation can lead to larger influence on the unit sale, as its coefficient has an absolute value of 2.9421. An increase in sea level, resultant wind speed or average wind speed would increase the unit sales, while increase in dew point, station pressure or dew point would decrease the sales.

Diagnostics

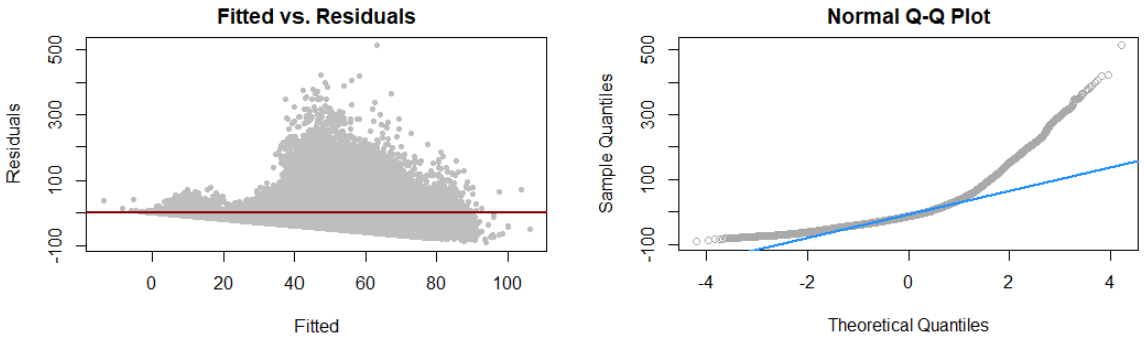

Figure 8: Fitted versus Residuals / Figure 9: Normal Q-Q Plot

To check the linear assumptions, we graph the Fitted versus Residuals and Normal Q-Q plots. From the Fitted versus Residuals plot, we can conclude that the assumption of linearity is not violated, since the residuals seem to center at 0 at any fitted value. But homoscedasticity appears to be violated because the spreads of residuals at different fitted values are not the same. The Normal Q-Q plot also shows the violation of normality assumption, as seldom points lie on the same straight line. We also use the *vif* function in R to check the multicollinearity, and the result shows that no VIF values are higher than 5. We then conclude that the non-collinearity assumption is not violated here.

Kaggle Submission

We use this model along with test data to predict the unit sales and submit the results on Kaggle. The score turns out to be 2.33489, which is too high to prove an accurate estimation.

Improvements

KNN and TREE Model

Recall that the Kaggle score of the above linear regression model is high (0.83930). As above the linear model failed, the reason that first came into our mind was the use of an inappropriate model. As a result, we turn to make use of the other two different modeling methods which are k-nearest neighbors and decision tree with almost every weather variable

in the dataset. Due to the huge size of the original dataset, we decide to randomly subset it and take about 50,000 observations from it. In this case, we calculate the test RMSE value of each model instead of running the Kaggle score. However, with almost all the weather variables included, the RMSE value appears to be unsatisfying and the Kaggle score of the two models is about 0.66375 and 0.75548. Although the two scores are far from enough it is much better than the first OLS model. To further upgrade the model, we try to minimize the size of the predictor by deleting variables that are not strongly related to the units of sale and also variables that might have collinearity.

Test for collinearity

First, we try to detect the existence of collinearity relationships between those weather variables, therefore, we created Figure 1 in the Exploratory Data Analysis section. In this graph, the darker the color, the stronger the collinearity is. As Figure 1 shows, collinearity is severe. Variables containing temperature information like *tmax*, *tmin*, *tavg*, *dewopint*, *wetbulb* and *cool* are positively correlated with each other. The collinearity between those is easy to explain as the *tavg* is, in fact, the average value of *tmax* and *tmin*, the *dewpoint* is calculated from the temperature the humidity (*wetbulb*) and the higher the temperature, the stronger the air conditioning in the room. Despite those variables that are positively related to each other, there are also many variables that are negatively related to each other. As the red area in Figure 1, variable *heat* and *sealevel* are negatively correlated with those temperature variables. Taken the collinearity into consideration, we decide to drop *tmin*, *tmax*, *dewpoint*, *wetbulb*, *heat*, *cool* and *sealevel* from the dataset.

In addition, we observe there is a great number of missing values in some of the variables like *depart*, *sunrise*, *sunset*, and *codesum*. Too much missing information makes them less applicable in the model, so we need to delete them too.

Model selection using stepwise

After deleting some of the 'bad' variables, in order to make our regression model more concise. We build up a linear model with predictors that have not been deleted yet and do the model selection using stepwise methods. As a result of stepwise model selection, we further delete *precipital, depart, resultspeed* and *tavg* from the model. Then we run the summary of the output of stepwise selection and the summary shows only the *stnpressure, store_nbr, station_nbr* and *item_nbr* are significant in the regression model at a 5% level. As the pressure for the same area should be almost the same, the *stnpressure* is collinear with *station_nbr*. In the end, variables that are going to be included in our future model is*, store_nbr, station_nbr,* and *item_nbr*.

Random Forest

From the results above, we can conclude that there is no significant effect of weather variables on the units of sale. Therefore, we have decided to delete all the weather variables from the model and use only the *item_nbr*, *store_nbr*, *station_nbr* as predictors. The variable *date* is also excluded this time as we used to think the value of date is not suitable in prediction. Then we create a random forest model with the *mtry* and *ntree* value tuned by using train function and out of bag method. The tuned *mtry* value is 3 and *ntree* value is 300. Noticed, there are many stores that never sell some specific goods. We, therefore, round our prediction result of the random forest model in order to make those prediction value which is really small to be zero. We upload the result we made from the random forest model to Kaggle and the score is about 0.15720, which is great progress compared to 0.6. This progress proves that removing those weather predictors are correct.

Transformation

To further improve the Kaggle score, we decide to do some transformation on our response variable. We apply the *boxcox* function on the linear model which only includes *item_nbr*, *store_nbr* and *station_nbr*. The output is shown below:

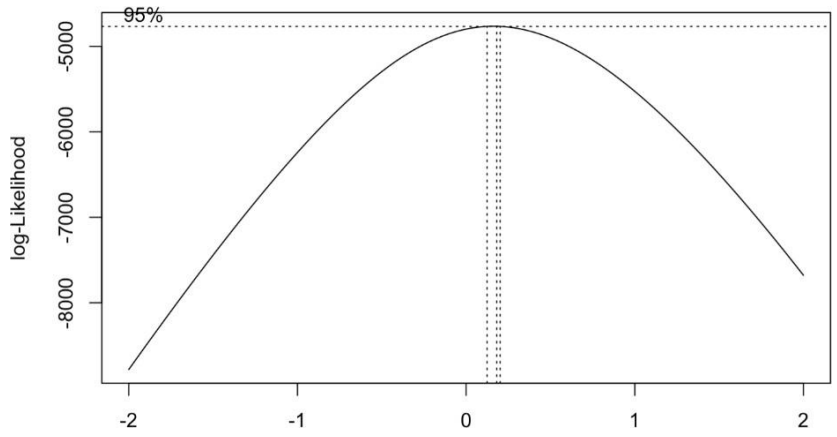

Figure 11: Boxcox

As we can tell from the graph, the lambda value is close to 0, which implies that a log transformation could be applied to the model. What's more, doing log transformation only is not enough in this case. Because for a large portion of the data, the value of units is equal to 0 and the value of log(0) is negative infinity. In order to prevent infinity value in prediction, we decided to apply transformation in the form as log(units +1). Keeping other things constant, we applied the transformed response variable in our random forest model and redo the prediction. As a result of the transformation, our model's Kaggle score improves from 0.15760 to 0.14239.

Classify Date

After the transformation is done, our team encounters a bottleneck in the optimization of the model. Thinking about the potential relationship between predictors and response, we choose to include the *date* information in our model again with its value classified in different methods.

Since people would prefer to buy items during the "Big sales" period, we create another variable *BlackF,* which means "Black Friday". We first search for the date for Black Friday both in 2012 and 2013 (we do not include Black Friday in 2014 because there is no data from 2014-11-01 to 2014-12-31 in the training data). Then, we set the value for these two days as "BlackFriday*".* Other values are like "1day_BlackFriday", "2days_BlackFriday", and

"3days_BlackFriday". In this case, "1day_BlackFriday" means the date that is one day before or after Black Friday and "2days_BlackFriday" means 2 days before or after Black Friday. The dates that are not in the Black Friday week are named as "Normal day". In order to prove that the sales differ a lot between Black Friday week and normal day, we calculate the average sale per day of Normal days and Black Friday week then plot the result into the below bar plot.

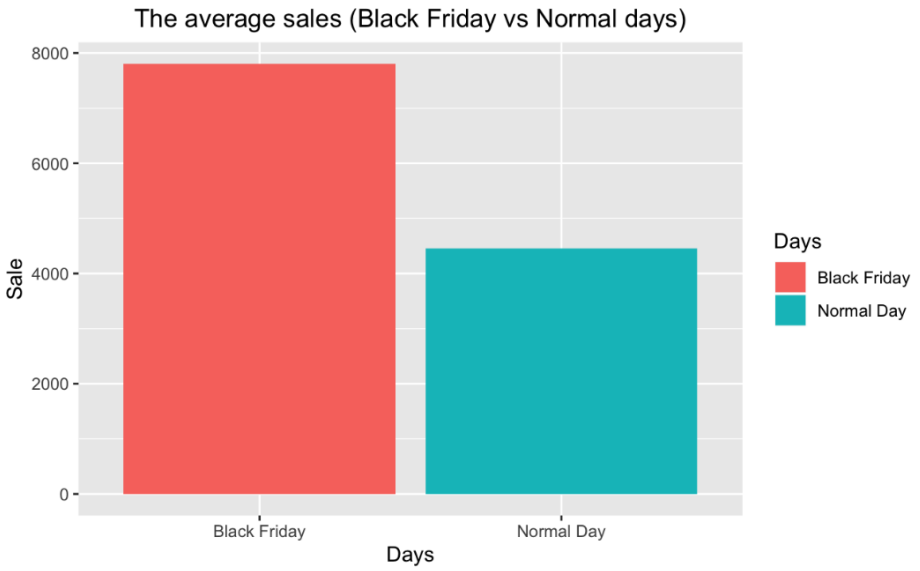

Figure 12: Avg Sales: Black Friday versus Normal Days

From Figure 12 we can tell that the average sales in Black Friday week per day is about 7,800 units and the average sales of a normal day is about 4,400 units. The significant difference between sales of the two categories indicates that it is meaningful to add them into our model.

In common sense, at the weekends, people without busy schedules would be more willing to do shopping than on weekdays. Therefore, the second variable is to classify weekends and weekdays from the *date*. In order to make this classification, we first use the *as.Date* and *weekday* functions to transform the date into weekdays. Monday to Friday are classified as "weekdays" together with Saturday and Sunday are classified as "Weekend". The bar plot of weekend and weekdays are as follows:

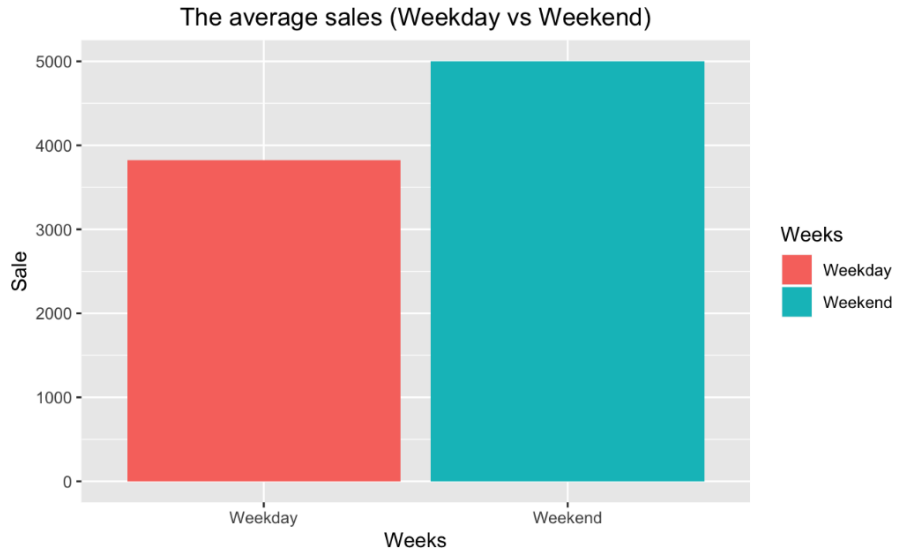

Figure 13: Avg Sales: Weekday versus Weekend

As we can tell from Figure 13, there is also a significant difference between weekday and weekend categories, and it is meaningful to add them to our model.

The third way that we think of is to classify the date by season. Although there is no obvious connection between sales and seasons, our group wants to try our luck. Thus, we set the 12 months to 4 groups: December, January, February as group “Winter”, March, April, May as group “Spring”, June, July, August as group “Summer” and September, October, November as group “Fall”. After this new variable is created, we make a chart with 4 plots to see if the sales of 111 items would be affected by different seasons.

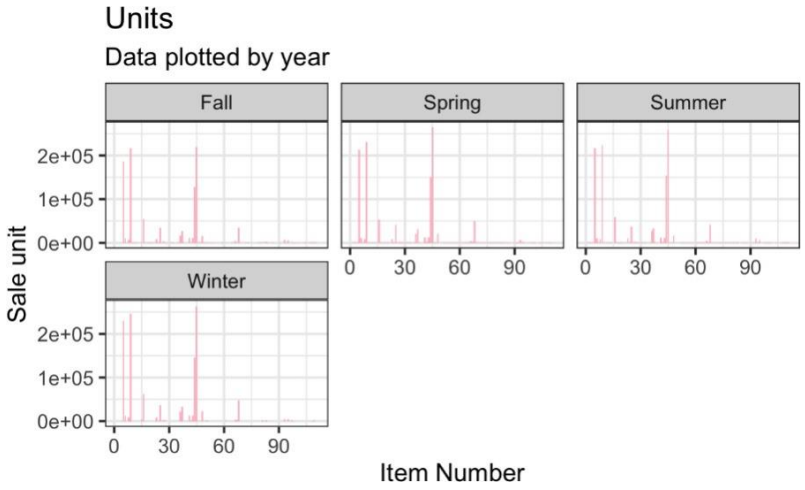

Figure 17: Units by Seasons

By analyzing the charts, we can conclude that *seasons* have no significant effect on the units of sales and only two date classified variables (*BlackF* and *Weeks)* will be included in a future model.

Optimal way to subset the data

Besides creating additional date variables, we also notice that a randomly subset of data is not the optimal way. As the size of nonzero units and zero units are imbalanced in the dataset, compared to zero observations, nonzero observation is more valuable for prediction. If we randomly subset the data, we will lose a lot of nonzero observations and make the prediction less precise. Therefore, we need to come up with another way to better subset the data.

As we have talked above, the reason why there are so many zero observations is because that some stores never sold some specific items. In order to pick those observations, we create a *new_id* variable both in the testing and training dataset, which is the combination of *store_nbr* and *item_nbr*. Then, we sum the units of sales by the *new_id* using an aggregate function. By doing this, we can tell which store has never sold which good by the *new_id* they have. Next, we pick that *new_id* which corresponding to nonzero sales and use it to subset both training data and testing data. By doing data subsampling in this way, we can make the model training and predicting process more focus on observations that are actually selling something. By both adding classified date variables and doing the subset, our random forest model ended up with a Kaggle score of 0.13182.

Extra Models

Linear model with interaction terms

After deleting those weather variables, to verify the interaction between *item_nbr*, *store_nbr*, and days, we use forward AIC to select the "best" model. We set the lower boundary to be lm(units ~ days) to full model to be lm(units ~ store_nbr + days + item_nbr +

store_nbr*item_nbr + days*item_nbr + days*store_nbr). The forward AIC selected the model lm(units ~ days + item_nbr + store_nbr + days*item_nbr + item_nbr*store_nbr). But from the plot below (Figure 10), the normal QQ plot indicates that the normality of the model is violated. The Kaggle score is 0.83930 and this linear model with interaction is not predicting units well.

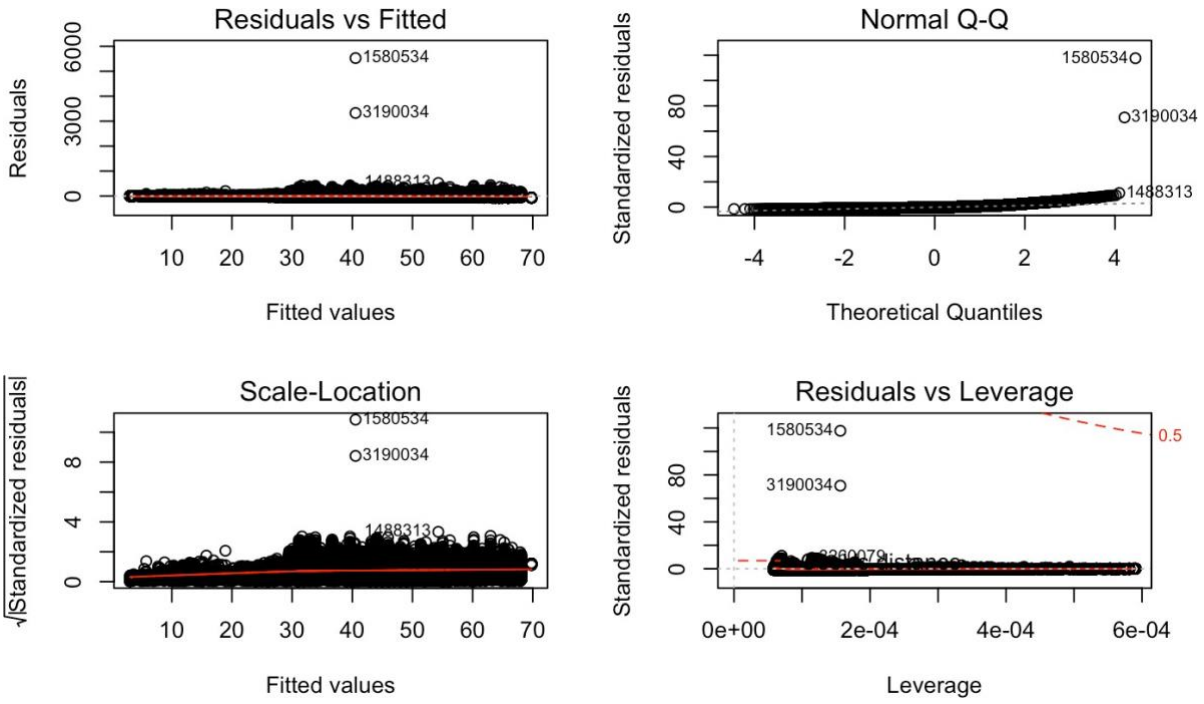

Figure 10: Diagnostics of interaction model

Poisson Regression

We tried to use Poisson regression and assumed the response variable units has a Poisson distribution and the logarithm of its expected value can be modeled by a linear combination of *store_nbr*, days, log(*item_nbr*). All variables have a p-value smaller than 2e-16 and are all significant. From the Normal QQ plot (Figure 11), the model seems to be a straight line and normality is assumed to be achieved at this point. We used this model on the test data set and have a Kaggle score of 0.53907, which does better than a linear model but still needs improvement.

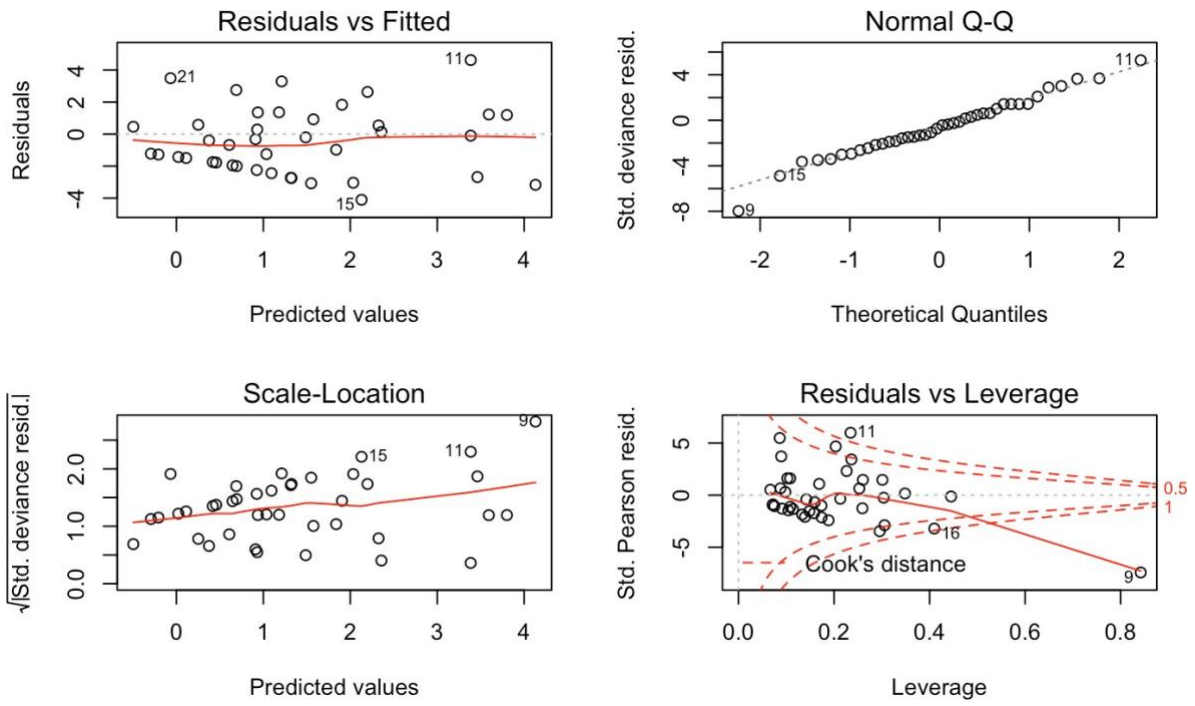

Figure 11: Diagnostics of Poisson Regression

LASSO Regression

We regress the model with predictor *store_nbr*, days and log(*item_nbr*) using LASSO to transform the model with the best lambda. From Figure 12, we can see that the optimal lambda selected is 0.01. The graph of prediction (see Figure 13) indicates that there are several outliers and high deviation from the value. The Kaggle score is 3.49762. This indicates that transformation is needed.

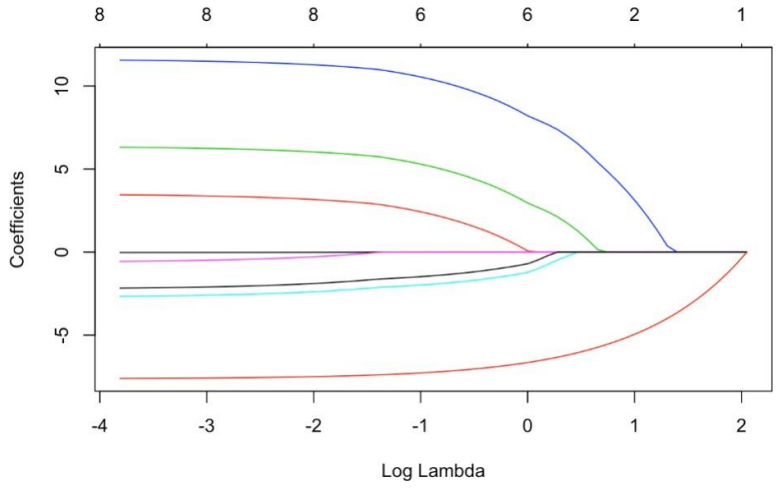

Figure 12: LASSO

Appendix

Data source: https://www.kaggle.com/c/walmart-recruiting-sales-in-stormy-weather/data

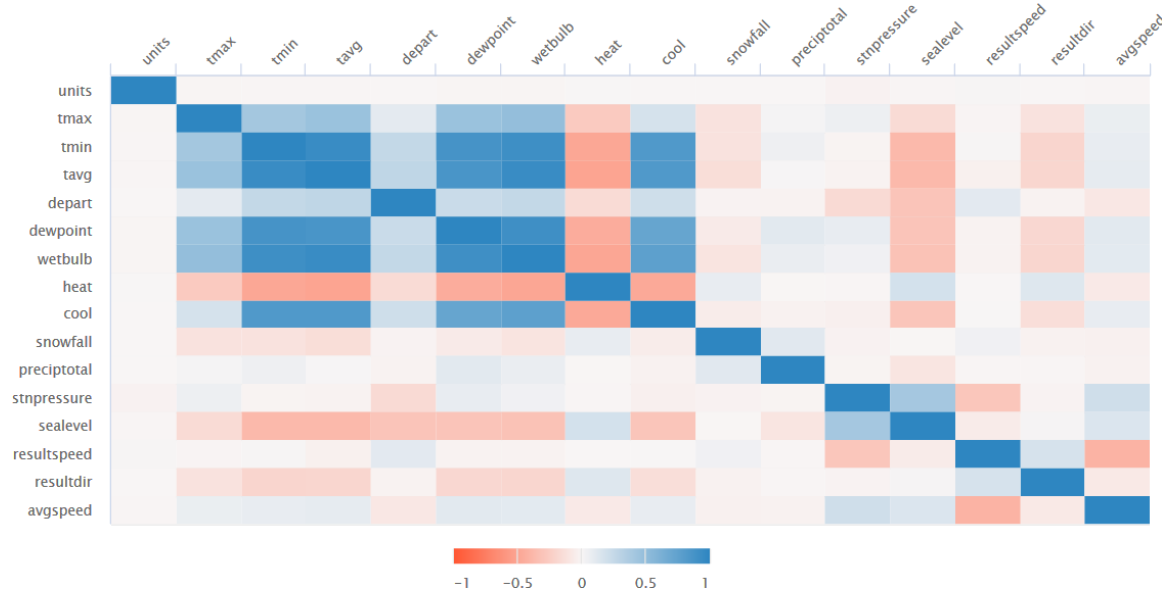

Figure 1: Correlation

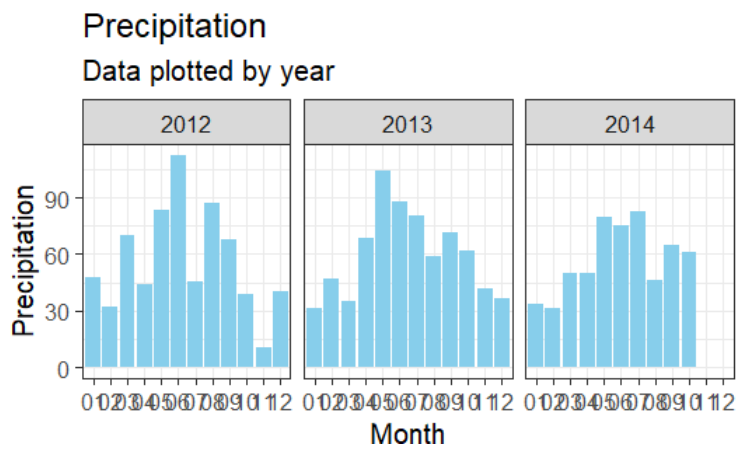

Figure 2: Precipitation

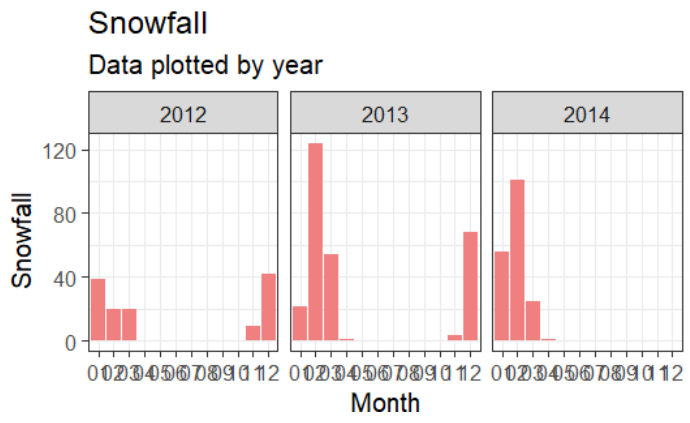

Figure 3: Snowfall

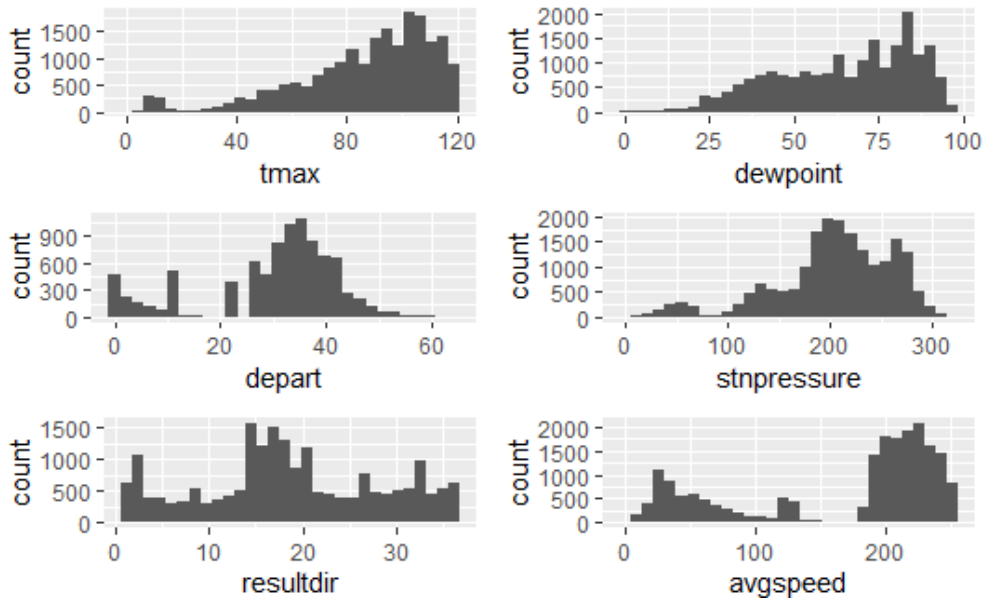

Figure 4: Histograms

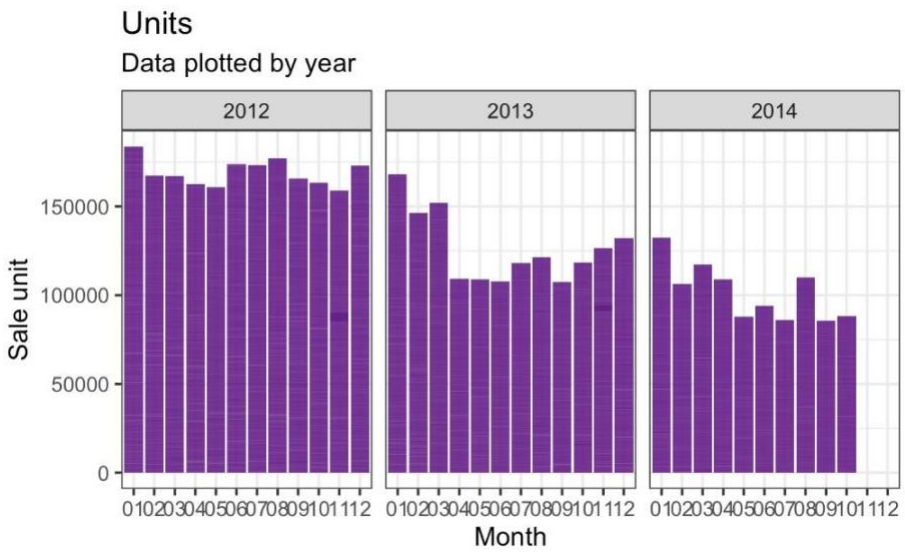

Figure 5: Units

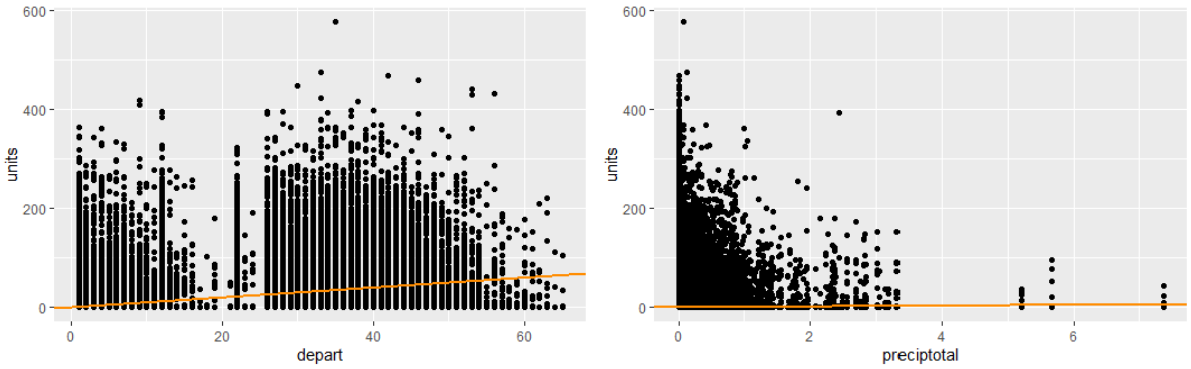

Figure 6: Units versus depart / Figure 7: Units versus preciptotal

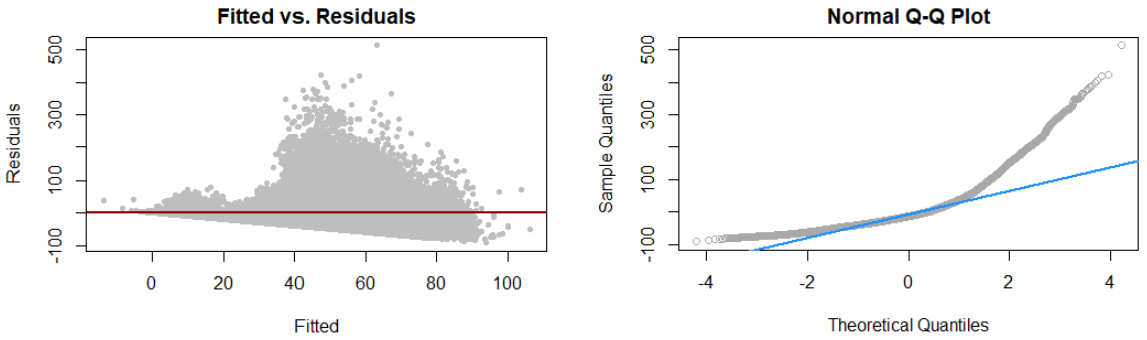

Figure 8: Fitted versus Residuals / Figure 9: Normal Q-Q Plot

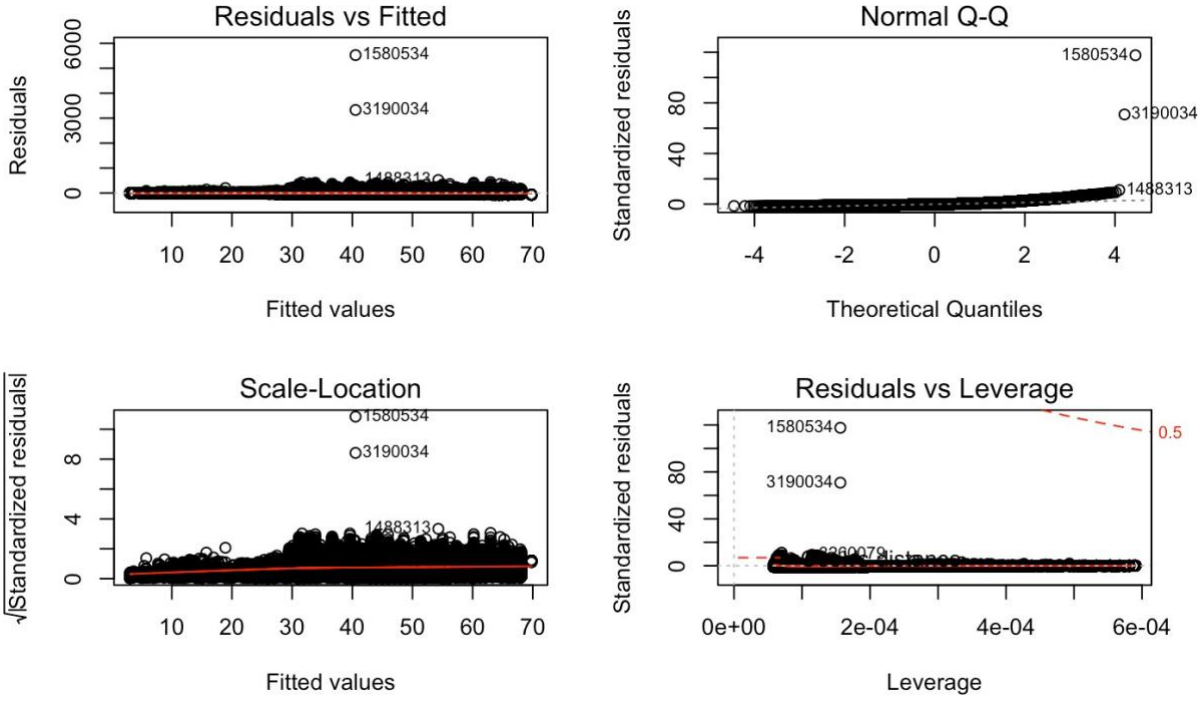

Figure 10: Diagnostics of interaction model

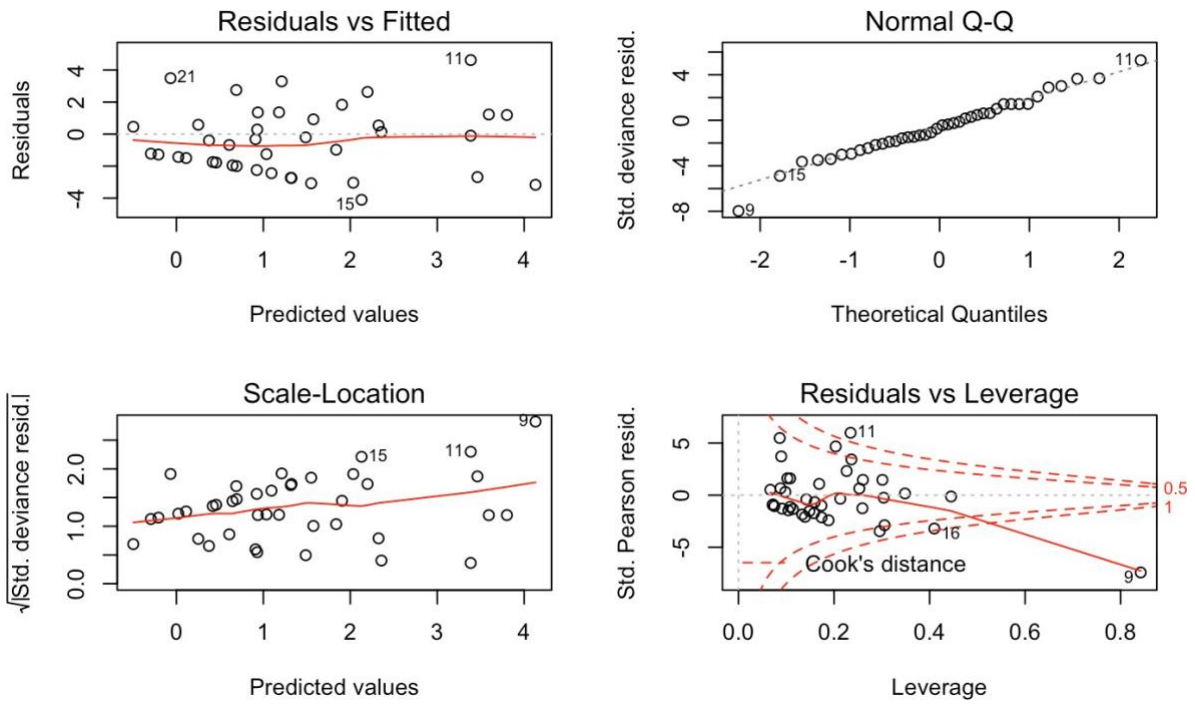

Figure 11: Diagnostics of Poisson Regression

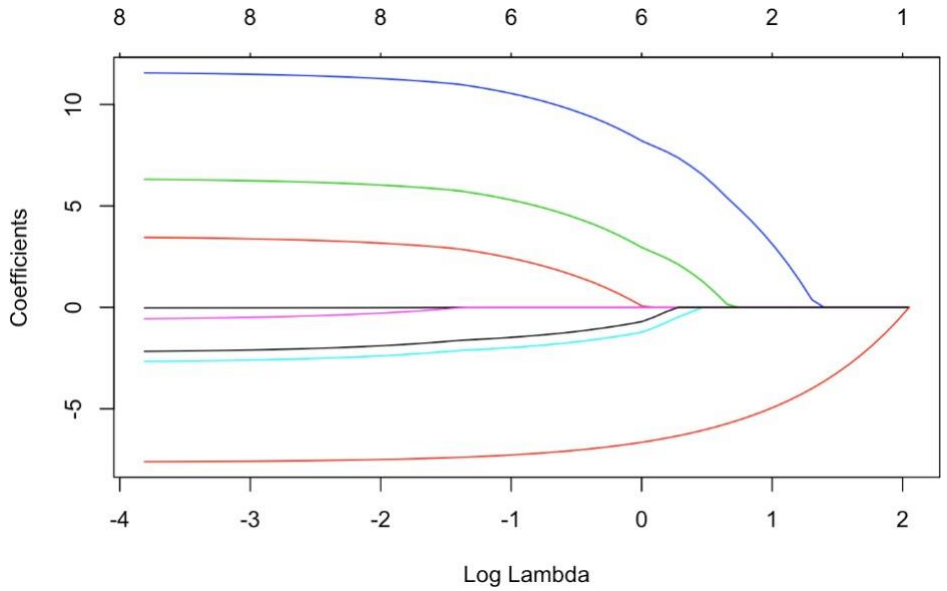

Figure 12: LASSO

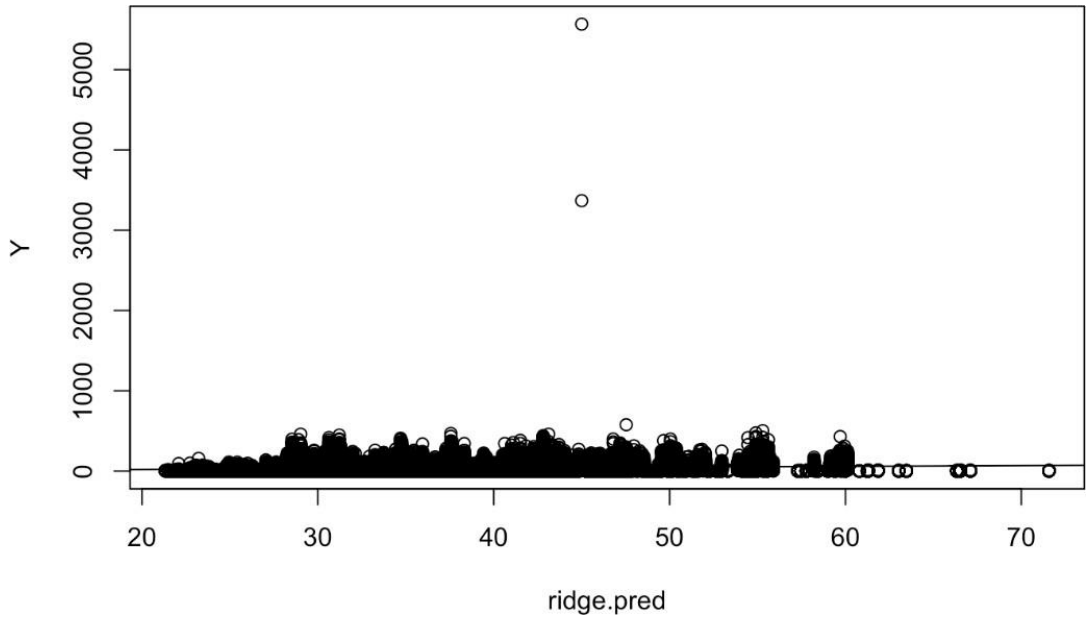

Figure 13: Ridge prediction

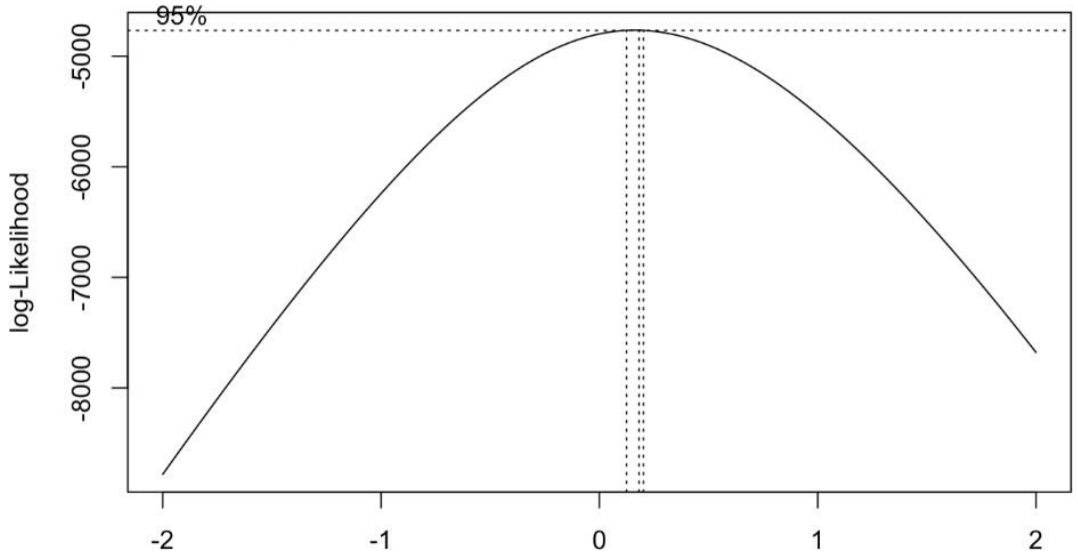

Figure 14: Boxcox

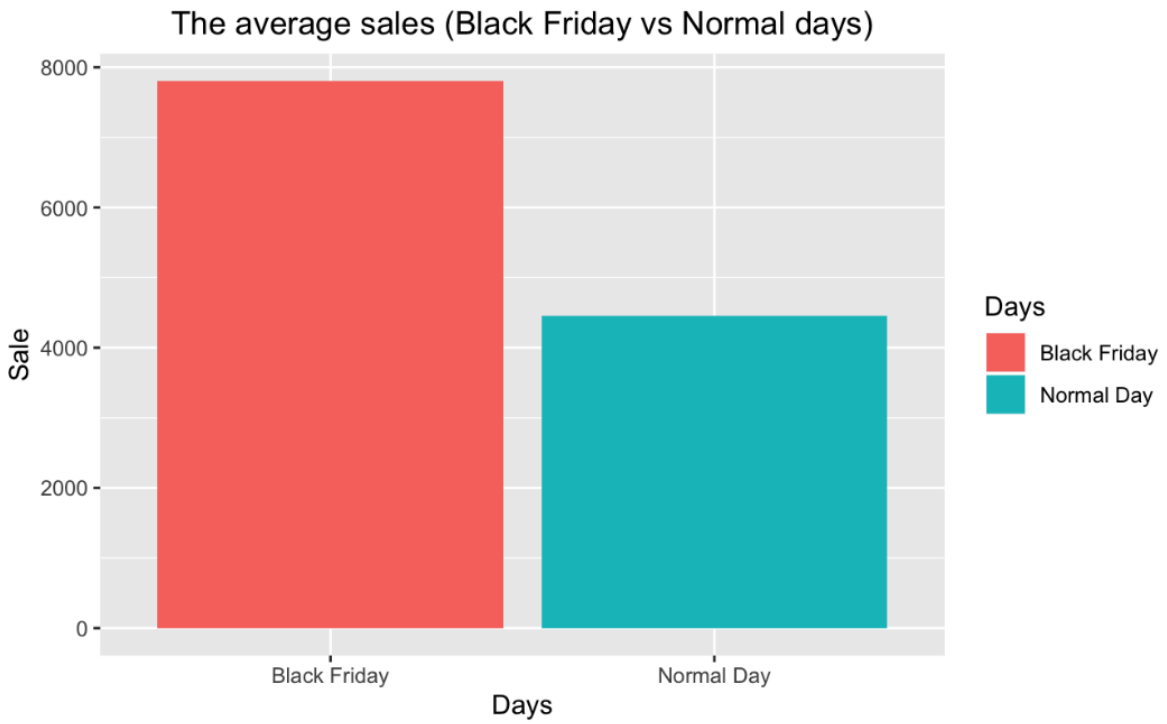

Figure 15: Avg Sales: Black Friday versus Normal Days

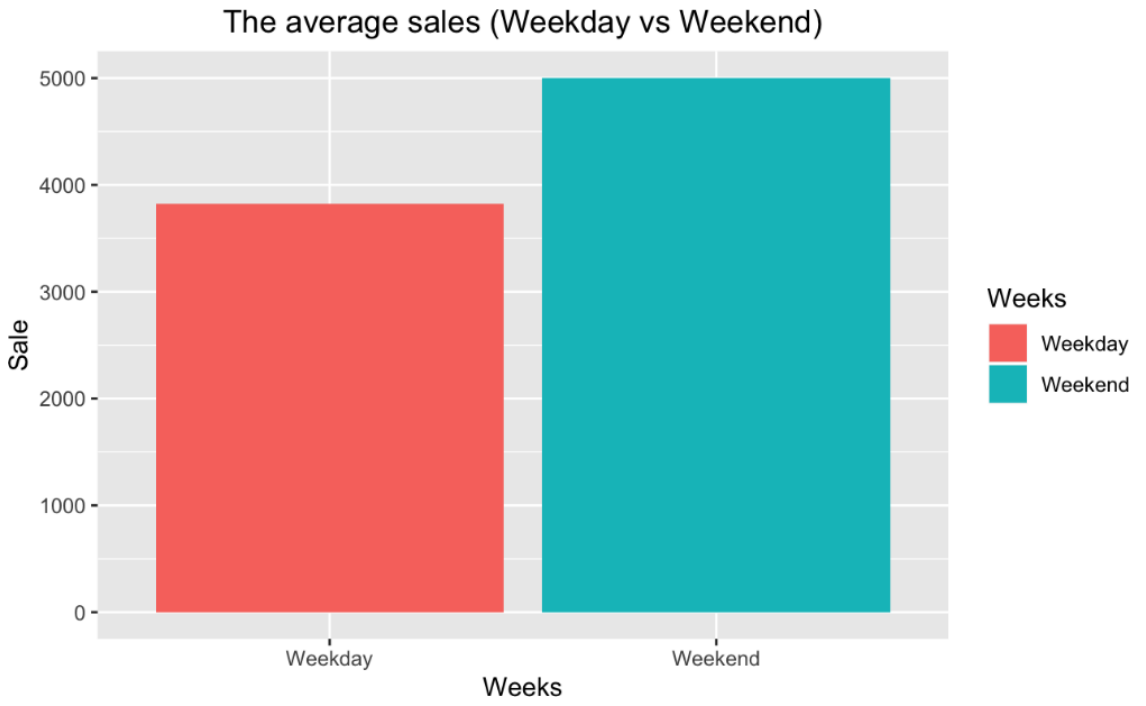

Figure 16: Avg Sales: Weekday versus Weekend

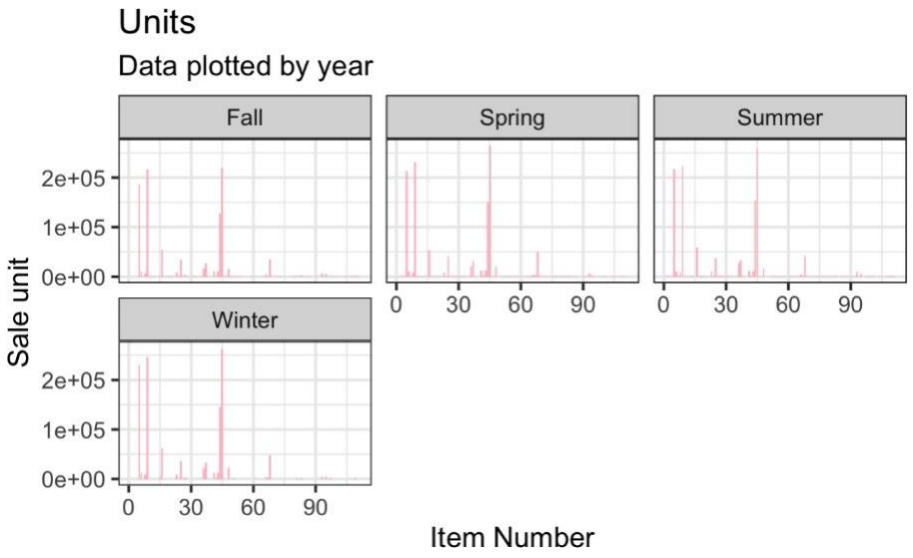

Figure 17: Units by Seasons

Code

```
library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.2.1 ──

## ✔ ggplot2 3.2.1     ✔ purrr 0.3.3
## ✔ tibble 2.1.3      ✔ dplyr 0.8.3
## ✔ tidyr 1.0.0       ✔ stringr 1.4.0
## ✔ readr 1.3.1

## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()

library(ggplot2)
library(glmnet)

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 3.0-1
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift

library(gridExtra)

##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine

library(highcharter)

## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

## Highcharts (www.highcharts.com) is a Highsoft software product which is

## not free for commercial and Governmental use

library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:gridExtra':
##
##     combine

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin

library(faraway)

##
## Attaching package: 'faraway'
```

```
## The following object is masked from 'package:lattice':
##
##     melanoma

knitr::opts_chunk$set(eval = FALSE)

weather = read.csv(file = '~/Documents/STAT 425/Final Project/weather.csv'
)
train = read.csv(file = '~/Documents/STAT 425/Final Project/train.csv')
test = read.csv(file = '~/Documents/STAT 425/Final Project/test.csv')
key = read.csv(file = '~/Documents/STAT 425/Final Project/key.csv')
samplesub = read.csv(file = '~/Documents/STAT 425/Final Project/sampleSubm
ission(1).csv')
```

Exploratory Data Analysis

```
# Data Pre-processing
weather$date = as.Date(weather$date, format = "%Y-%m-%d")

weather$tavg[weather$tavg == "M"] = NA
weather$tavg = as.numeric(weather$tavg)
weather$tmax[weather$tmax == "M"] = NA
weather$tmax = as.numeric(weather$tmax)
weather$tmin[weather$tmin == "M"] = NA
weather$tmin = as.numeric(weather$tmin)
weather$depart[weather$depart == "M"] = NA
weather$depart = as.numeric(weather$depart)
weather$dewpoint[weather$dewpoint == "M"] = NA
weather$dewpoint = as.numeric(weather$dewpoint)
weather$stnpressure[weather$stnpressure == "M"] = NA
weather$stnpressure = as.numeric(weather$stnpressure)
weather$heat[weather$heat == "M"] = NA
weather$heat = as.numeric(weather$heat)
weather$cool[weather$cool == "M"] = NA
weather$cool = as.numeric(weather$cool)

weather$snowfall = gsub("\\s+", "", weather$snowfall)
weather$snowfall[weather$snowfall == "M"] = NA
weather$snowfall[weather$snowfall == "T"] = 0
weather$snowfall[weather$snowfall == "0.0"] = 0
weather$snowfall = as.numeric(weather$snowfall)

weather$preciptotal = gsub("\\s+", "", weather$preciptotal)
weather$preciptotal[weather$preciptotal == "M"] = NA
weather$preciptotal[weather$preciptotal == "T"] = 0
weather$preciptotal[weather$preciptotal == "0.00"] = 0
weather$preciptotal = as.numeric(weather$preciptotal)

weather$wetbulb[weather$wetbulb == "M"] = NA
weather$wetbulb = as.numeric(weather$wetbulb)
weather$sealevel[weather$sealevel == "M"] = NA
weather$sealevel = as.numeric(weather$sealevel)
weather$resultspeed[weather$resultspeed == "M"] = NA
weather$resultspeed = as.numeric(weather$resultspeed)
```

```r
weather$resultdir[weather$resultdir == "M"] = NA
weather$resultdir = as.numeric(weather$resultdir)
weather$avgspeed[weather$avgspeed == "M"] = NA
weather$avgspeed = as.numeric(weather$avgspeed)

# Merge data
train$date = as.Date(train$date, format = "%Y-%m-%d")
test$date = as.Date(test$date, format = "%Y-%m-%d")
train_with_key = left_join(key, weather, by = "station_nbr")
comb_trn = left_join(train, train_with_key, by = c("date", "store_nbr"))
comb_tst = left_join(test, train_with_key, by = c("date", "store_nbr"))

# take a look at the data
glimpse(comb_trn)
summary(comb_trn)

# filter out NAs to graph correlation plot
data.sub = na.omit(comb_trn)
hchart(cor(data.sub[, c(4, 6:13, 17:23)]))

# Visualization: distributions
p1 = ggplot(data = weather) + geom_histogram(aes(x = tmax))
p2 = ggplot(data = weather) + geom_histogram(aes(x = dewpoint))
p3 = ggplot(data = weather) + geom_histogram(aes(x = depart))
p4 = ggplot(data = weather) + geom_histogram(aes(x = stnpressure))
p5 = ggplot(data = weather) + geom_histogram(aes(x = resultdir))
p6 = ggplot(data = weather) + geom_histogram(aes(x = avgspeed))
##grid.arrange(p1, p2, p3, p4, p5, p6, ncol = 2)

# Visualization: relation with unit sales
ggplot(data = data.sub) + geom_point(aes(x = depart, y = units)) + geom_ab
line(color = "darkorange", size = 1)
ggplot(data = data.sub) + geom_point(aes(x = preciptotal, y = units)) + ge
om_abline(color = "darkorange", size = 1)

#Visualization: barplots
weather %>%
  ggplot(aes(x = format(date, "%m"), y = preciptotal)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  facet_wrap( ~ format(date, "%Y"), ncol = 3) +
  labs(
    title = "Precipitation",
    subtitle = "Data plotted by year",
    y = "Precipitation",
    x = "Month") +
  theme_bw(base_size = 15)

weather %>%
  ggplot(aes(x = format(date, "%m"), y = snowfall)) +
  geom_bar(stat = "identity", fill = "lightcoral") +
  facet_wrap( ~ format(date, "%Y"), ncol = 3) +
  labs(
    title = "Snowfall",
    subtitle = "Data plotted by year",
```

```
    y = "Snowfall",
    x = "Month") +
  theme_bw(base_size = 15)

#units
comb_trn$date = as.Date(comb_trn$date)
comb_trn %>%
  ggplot(aes(x = format(comb_trn$date,"%m"), y = comb_trn$units)) +
      geom_bar(stat = "identity", fill = "darkorchid4") +
  facet_wrap(~ format(comb_trn$date,"%Y"), ncol = 3) +
      labs(title = "Units",
           subtitle = "Data plotted by year",
           y = "Sale unit",
           x = "Month") + theme_bw(base_size = 15)
  scale_x_date(date_labels = "%b")
```

Linear Regression Model

```
# subset nonzero sale units
data.mod = comb_trn[comb_trn$units > 0, ]

# linear regression
OLS1 = lm(units ~ date + store_nbr + item_nbr + tmax + depart + dewpoint +
heat + snowfall + preciptotal + sealevel + stnpressure + resultspeed + av
gspeed, data = data.mod)

summary(OLS1)

# Diagnostics
# Fitted vs Residuals
plot(fitted(OLS1), resid(OLS1), col = "grey", pch = 20, xlab = "Fitted", y
lab = "Residuals", main = "Fitted vs. Residuals")
abline(h = 0, col = "darkred", lwd = 2)
# Normal Q-Q
qqnorm(resid(OLS1), main = "Normal Q-Q Plot", col = "darkgrey")
qqline(resid(OLS1), col = "dodgerblue", lwd = 2)
# Multicollinearity
vif(OLS1)

# Prediction
comb_tst$date <- as.Date(comb_tst$date, format = "%Y-%m-%d")
comb_tst$stnpressure = as.numeric(comb_tst$stnpressure)
comb_tst$heat = as.numeric(comb_tst$heat)
comb_tst$dewpoint = as.numeric(comb_tst$dewpoint)
comb_tst$resultspeed = as.numeric(comb_tst$resultspeed)
comb_tst$avgspeed = as.numeric(comb_tst$avgspeed)
comb_tst$depart = as.numeric(comb_tst$depart)
comb_tst$snowfall = as.numeric(comb_tst$snowfall)
comb_tst$preciptotal = as.numeric(comb_tst$preciptotal)
comb_tst$tmax = as.numeric(comb_tst$tmax)
comb_tst$sealevel = as.numeric(comb_tst$sealevel)

predict.test = predict(OLS1, comb_tst)
submission = cbind(samplesub, units = matrix(predict.test))
```

```
submission = submission[,-2]

write.table(submission, file = "LinearRegression.csv", sep = ",")
#kaggle score: 2.33489
```

Improvement

```
#add new variable `day` to conb_trn
comb_trn$days <- format(as.Date(comb_trn$date, "%Y-%m-%d"), "%A")
comb_tst$days <- format(as.Date(comb_tst$date, "%Y-%m-%d"), "%A")
intercept = lm(units ~ days, data = comb_trn)
full = lm(units ~ store_nbr + days + item_nbr + store_nbr*item_nbr + days*
item_nbr + days*store_nbr, data = comb_trn)
forwardaic = step(intercept, direction = "forward", scope = list(lower = i
ntercept, upper = full), data = comb_trn, k = 2)
forwardaic

lm.mod <- lm(units ~ days + item_nbr + store_nbr + days*item_nbr + item_nb
r*store_nbr, data = comb_trn)
summary(lm.mod)
par(mfrow=c(2,2));plot(lm.mod)

#Prediction
prediction <- predict(lm.mod, newdata = comb_tst)
variablePredictionsIds <- paste(test$store_nbr, test$item_nbr, test$date,
sep = "_")
variablePredictions <- round(prediction)
samplesub$id <- variablePredictionsIds
samplesub$units <- variablePredictions
samplesub = subset(samplesub, select = c(id, units))
write.csv(samplesub, file = "raw.csv", row.names = FALSE)
#kaggle score:0.83930

glm.model = glm(units ~ store_nbr + days + log(item_nbr), data = comb_trn,
 family = poisson)
summary(glm.model)
par(mfrow=c(2,2));plot(glm.model)
exp(coef(glm.model))
#kaggle score:0.53907

X = model.matrix(glm(units ~ store_nbr + days + log(item_nbr), data = comb
_trn))
Y = comb_trn$units
lasso = glmnet(x = X, y = Y, family="gaussian", alpha = 1)
summary(lasso)
plot(lasso, xvar = "lambda")
y.pred1 = predict(lasso, X)
lambdas = 10^seq(3, -2, by = -.1)
cv_fit = cv.glmnet(X, Y, alpha = 1, lambda = lambdas)
summary(cv_fit)
opt_lambda = cv_fit$lambda.min
opt_lambda
cv_fit$glmnet.fit
ridge.pred = predict(cv_fit$glmnet.fit, s = opt_lambda, newx = X)
```

```r
ridge.pred
plot(ridge.pred,Y);abline(0,1)

# add a new variable `season`
month = as.numeric(format(comb_trn$date, "%m"))
season = ifelse(month == 1 | month == 2 | month == 12, "Winter",
                ifelse(month == 3 | month == 4 | month == 5, "Spring",
                       ifelse(month == 6 | month == 7 | month == 8, "Summe
r",
                              ifelse(month == 9 | month == 10 | month == 1
1, "Fall",NA))))
comb_trn <- comb_trn %>% mutate(season)

# plot barcharts "season" with storenumber, itemnumber and stationnumber.
comb_trn %>%
  ggplot(aes(x = comb_trn$store_nbr, y = comb_trn$units)) +
  geom_bar(stat = "identity", fill = "pink") +
  facet_wrap( ~ comb_trn$season, ncol = 3) +
  labs(
    title = "Units",
    subtitle = "Data plotted by year",
    y = "Sale unit",
    x = "Store Number") +
  theme_bw(base_size = 15)

scale_x_date(date_labels = "%b")
#Conclusion: The sales of units by store seem unrealted with seasons.

comb_trn %>%
  ggplot(aes(x = comb_trn$station_nbr, y = comb_trn$units)) +
  geom_bar(stat = "identity", fill = "pink") +
  facet_wrap( ~ comb_trn$season, ncol = 3) +
  labs(
    title = "Units",
    subtitle = "Data plotted by year",
    y = "Sale unit",
    x = "Station Number") +
  theme_bw(base_size = 15)

scale_x_date(date_labels = "%b")
#Conclusion: The sales of units grouped by station seem unrealted with sea
sons.

set.seed(425)
random = sample(nrow(comb_trn), 50000)
comb_trn = comb_trn[random, ]
data1 = na.omit(comb_trn)

# knn model with tuned A meanwhile calculating the rmse value
calc_rmse_model = function(model, data, response) {
  actual = data[[response]]
  predicted = predict(model, data)
  sqrt(mean((actual - predicted) ^ 2))
}
```

```r
# knn
k = 2:100
knn_mods = map(k, ~ knnreg(units ~ ., data = data1, k = .x))
knn_rmse = map_dbl(knn_mods, calc_rmse_model, data = data1, response = "un
its")
which.min(knn_rmse)

# tree model
cp = c(0.0001, 0.001, 0.01, 0.1, 0)
tree_mods = map(cp, ~ rpart(units ~ ., data = data1, cp = .x))
tree_rmse = map_dbl(tree_mods, calc_rmse_model, data = data1, response = "
units")
which.min(tree_rmse)

# rf_mod_tuning = train(log(units+1)~item_nbr+store_nbr+station_nbr,
rf_mod_tuning = train(units~item_nbr+store_nbr+station_nbr,
             data = fin_trn1,
             method = "rf",
             trControl = trainControl(method = "oob"))

comb_trn$id_trn  = paste(as.factor(comb_trn$store_nbr), as.factor(comb_trn
$item_nbr), sep="_")
item_units = aggregate(comb_trn$units, by = list(item = comb_trn$id_trn),
FUN = sum)
item_units0 = item_units[which(item_units$x != 0), ]
colnames(item_units) = c("id_s", "units")

comb_tst$id_tst = paste(as.factor(comb_tst$store_nbr), as.factor(comb_tst$
item_nbr), sep="_")

WantedTrain = comb_trn[comb_trn$id_trn %in% item_units0$item, ]

WantedData=comb_tst[comb_tst$id_tst %in% item_units0$item, ]

# add week variable to the train
fin_trn1 = WantedTrain[, c(1, 2, 3, 4, 5,24)]
fin_trn1$date = as.Date(fin_trn1$date, format = "%Y-%m-%d")
fin_trn1$weekend = weekdays(fin_trn1$date)
fin_trn1$weeks = ifelse(fin_trn1$weekend == "Saturday" | fin_trn1$weekend
== "Sunday" , "Weekend",
                        ifelse(fin_trn1$weekend == "Monday" |fin_trn1$week
end == "Tuesday"| fin_trn1$weekend == "Wednesday"|fin_trn1$weekend == "Thu
rsday"|fin_trn1$weekend == "Friday", "Weekday",NA))
fin_trn1$weeks = as.factor(fin_trn1$weeks)

# add `BlackF` to the train
fin_trn1$BlackF = ifelse(fin_trn1$date == "2012-11-23" | fin_trn1$date ==
"2013-11-29", "BlackFriday",
                  ifelse(fin_trn1$date == "2012-11-22"|fin_trn1$date == "201
3-11-28"|fin_trn1$date == "2012-11-24"|fin_trn1$date == "2013-11-30",  "1d
ay_BlackFriday",
                         ifelse(fin_trn1$date == "2012-11-21"|fin_trn1$date
== "2013-11-27"|fin_trn1$date == "2012-11-25"|fin_trn1$date == "2013-12-01
", "2day_BlackFriday",
```

```
                                ifelse(fin_trn1$date == "2012-11-20"|fin_trn
1$date == "2013-11-26"|fin_trn1$date == "2012-11-26"|fin_trn1$date == "201
3-12-02", "3day_BlackFriday","Normal day"))))
fin_trn1$BlackF = as.factor(fin_trn1$BlackF)

# add BlackF and weeks to test
WantedData$date = as.Date(WantedData$date, format = "%Y-%m-%d")
WantedData$weekend = as.factor(weekdays(WantedData$date))
WantedData$weeks = ifelse(WantedData$weekend == "Saturday" | WantedData$we
ekend == "Sunday" , "Weekend",
                          ifelse(WantedData$weekend == "Monday" |WantedData$
 weekend == "Tuesday"| WantedData$weekend == "Wednesday"|WantedData$weekend
 == "Thursday"|WantedData$weekend == "Friday", "Weekday",NA))
WantedData$weeks = as.factor(WantedData$weeks)

WantedData$BlackF = ifelse(WantedData$date == "2012-11-23" | WantedData$da
te == "2013-11-29", "BlackFriday",
                ifelse(WantedData$date == "2012-11-22"|WantedData$date ==
"2013-11-28"|WantedData$date == "2012-11-24"|WantedData$date == "2013-11-3
0",  "1day_BlackFriday",
                       ifelse(WantedData$date == "2012-11-21"|WantedData$d
ate == "2013-11-27"|WantedData$date == "2012-11-25"|WantedData$date == "20
13-12-01", "2day_BlackFriday",
                              ifelse(WantedData$date == "2012-11-20"|Wante
dData$date == "2013-11-26"|WantedData$date == "2012-11-26"|WantedData$date
 == "2013-12-02", "3day_BlackFriday","Normal day"))))
WantedData$BlackF = as.factor(WantedData$BlackF)

# Plot barchart "BlackF".
N_units=sum(data$units[data$BlackF=="Normal day"])
meanN_units=N_units/((366-7)+(365-7)+(365-71))
B_units=sum(data$units[data$BlackF=="BlackFriday"|data$BlackF=="1day_Black
Friday"|data$BlackF=="2day_BlackFriday"|data$BlackF=="3day_BlackFriday"])
meanB_units=B_units/7
bf= data.frame(Days = c("Normal Day","Black Friday"), Sale = c(meanN_units
,meanB_units) )
ggplot(bf)+
  geom_bar(aes(x = Days, y = Sale, fill = Days), stat = "identity")+
ggtitle("The average sales (Black Friday vs Normal days)")+
  theme(plot.title = element_text(hjust = 0.5))

# random forest model tuning
rf_mod_tuning = train(units~item_nbr+store_nbr+station_nbr,
             data = fin_trn1,
             method = "rf",
             trControl = trainControl(method = "oob"))

# random forest model
set.seed(425)
rf_mod = randomForest(log(units+1) ~ item_nbr + store_nbr + weeks + BlackF
,
                      data = fin_trn1,
                      mtry = 3,
```

```
                      ntree = 300,
                      alpha = 1)

# derive the output file
sample2 = cbind(sample, id_tst = comb_tst[23])
WantedSample = sample2[sample2$id_tst %in% item_units0$item, ]
a = matrix(round(exp(predict(rf_mod, WantedData[c(2, 3, 4, 25, 26)]))-1))

sample111 = cbind(WantedSample, units = a)
sample333 = sample111[,-2]

sample444 = left_join(sample, sample333, by = "id")
sample444 = sample444[, -c(2,3)]
colnames(sample444) = c("id", "units")
sample444[is.na(sample444)] = 0
write.table(sample444, file = "sample888.csv", sep = ",", row.names = FALS
E)
```