Walmart Sales Prediction in Stormy Weather

12/15/2019

Group Members

- Aastha Nargas(anargas2)
- Kusum Vanwani(vanwani2)
- Pankaj Sharma(pankajs2)
- Shashi Roshan(sroshan2)

Introduction

Weather plays an important role in the consumer preferences while shopping. During extreme weather events, it would be beneficial for the stores to be stocked with products which are essential to cope up with the event. This project aims to predict the sales of 111 products sold in 45 different walmart locations. We are trying to model the effect on the weather conditions on the sales of these products. The 45 locations are covered by 20 weather stations. The task is to predict the amount of each product sold around the time of major weather events.

We are provided the full observed weather data so we don't need to forecast weather for the sake of this project. More information on the dataset can be found at the kaggle page for this competition, https://www.kaggle.com/c/walmart-recruiting-sales-in-stormy-weather/data (https://www.kaggle.com/c/walmart-recruiting-sales-in-stormy-weather/data).

Other than the competitive goal of the project, this project is also aimed at solidifying our understanding of the intricacies of the linear model and we will be testing a least squares model on this dataset. We will begin with understanding the datasets and draw insights from the data. By analyzing the data well we are able to create additional variables for our modelling process. Initially, we start with a simple linear model for prediction, then create diagnostics for our model and based on the inferences drawn will make modifications to the model in accordance with the material covered in the class.We will be building different models and testing their score and prediction accuracy. We will also be testing out some advanced models like Random Forest and Gradient Boosting to make the predictions and will compare with the performance of the linear model.

Intuitively, we may expect an increase in the sales of certain products before a big weather event, but it's difficult for supply-chain managers to correctly predict the level of inventory needed to avoid being out-of-stock or overstock during and after that storm. Walmart relies on a variety of vendor tools to predict sales around extreme weather events, but it's an ad-hoc and time-consuming process that lacks a systematic measure of effectiveness. Helping Walmart better predict sales of weather-sensitive products will keep valued customers out of the rain.

Data

File Descriptions

- key.csv - the relational mapping between stores and the weather stations that cover them
- train.csv - sales data for all stores & dates in the training set
- test.csv - stores & dates for forecasting (missing 'units', which you must predict)
- weather.csv - a file containing the NOAA weather information for each station and day

Sales data for 111 products sold in 45 different stores whose sales may be impacted by weather such as dairy products, rain essentials etc. are provided. There might be cases where same product is being sold with a different id in a different store. The 45 store locations are covered by 20 weather staions mapping of which is provided in the key file. The full observed weather conditions covering the time period of both training and test data are provided. The training data contains ~ 4.6M rows of data while test data contains ~ 0.52M rows of data.

A quick look at the datasets.

Table 0.01: A sample of the Training Dataset

date	store_nbr	item_nbr	units
2012-01-01	1	1	0
2012-01-01	1	2	0
2012-01-01	1	3	0
2012-01-01	1	4	0
2012-01-01	1	5	0
2012-01-01	1	6	0

depart	dewpoint	wetbulb	heat	cool	sunrise	sunset	codesum	snowfall	preciptotal	stnpressure	sealevel	resultspeed	resultdir	avgspeed
4	28	43	10	0	0728	1742		0.0	0.00	29.79	30.48	8.0	35	8.2
0	31	43	16	0	0727	1742		0.0	0.00	29.95	30.47	14.0	36	13.8
M	26	35	25	0	•	•		0.0	0.00	29.15	30.34	10.3	32	10.2

EDA

We want to start with investigating the data and identify some patterns which can help us the understand the data better. We want to make sure that the data we have does not have missing values since the linear model that we may aim to fit will remove the rows which contain the missing values. We first look at the missing values in train data. **Fig 0.01** is a visualization of the number of records containing missing values by predictor variables. We can see that there are no missing values in the train data.

Fig 0.02 shows us the missing values in the weather data. We can see that most of the variables have some missing values. We need to think about the implications of these missing values. The missing values need to be imputed because the linear model by default is not very good at handling missing values. We use `MICE` package in `R` to impute the missing values in the weather data. This package helps us to impute the missing values with plausible data values. These plausible values are drawn from a distribution specifically designed for each missing data point.

Before we do further exploratory analysis on the train data, we need to do some preprocessing. First we need to change the type of the date variable and convert it into a `date` object. We then use this `date` object to extract the day and month as the intuition dictates that sales will depend on the month of the year and the day of the week. People are more likely to go shopping on weekends than weekdays. **Fig .03** confirms this as we see that the total sales are highest on Saturday and Sunday as compared to the rest of the week.

Fig 0.04 is very similar to the earlier plot but instead of seeing sales by day we instead check the sales by month. We can clearly see that `January` has the highest sales as compared to the rest of the months and `November` and `December` have the least amount of units sold. An insight from this could be drawn, since January is the month with harsher winters, the requirement for weather related products might be more and hence more units maybe sold. Due to holidays and some products' demand being seasonal, we also wanted to see the effect of season on sales. For this, we created a `season` variable in the dataset and plotted it against units sold. **Fig 0.05** plots the relationship between the season and the units sold. We see that the sales are considerably low in *Fall* but for the rest of the seasons there is not much difference.

We want to check the correlation among the predictors in the weather dataset. for this we create a correlation plot in `R`. **Fig 0.06** shows the correlation of all the variables with each other. We can see that time of sunrise and sunset are highly correlated with the temperature variables. Sunset and Sunrise temperatures are highly correlated. As per expectations, average temperature is almost perfectly correlated to maximum and minimum temperatures. Wind parameters are negatively correlated with the temperature variables which is expected as a stronger wind will bring the temperature down. We will be calculating variance inflation factors in this analysis to further assess the correlations and then remove the variables with very high inflation factors.

Before we start the Exploratory analysis on the weather data, we need to impute the weather data and merge it with the train data so that we can understand the relationship between the weather parameters and units sold. **Fig 0.07** shows that the wind speed isn't necessarily a defining factor but the there no points in the upper right half of the graph showing that large number of units are only sold with low wind speed. With high wind speed, people probably prefer to stay at home instead of going shopping.**Fig 0.08** shows us that with very high and very low temperatures the number of units sold goes down which makes sense as people would try to avoid extreme temperatures.

Missing rows in Train Data

	x
date	0
store_nbr	0
item_nbr	0
units	0

Missing rows in Weather Data

	x
station_nbr	0

	x
date	0
tmax	906
tmin	908
tavg	1469
depart	11511
dewpoint	666
wetbulb	1252
heat	1469
cool	1469
sunrise	9656
sunset	9656
codesum	0
snowfall	7224
preciptotal	860
stnpressure	929
sealevel	1724
resultspeed	589
resultdir	589
avgspeed	875

Modelling

In the data we notice that there are a large number of items for which sales are zero throughout the time period for which the data is available. Since we are not learning any new information from these observations, we would not consider them to train our model, instead we will just consider the items for which we have atleast some non-zero sales. This will also allow us to reduce the size of the training set quite a bit and significantly reduce the model training time. We will start with a simple linear regression model and then investigate it to assess the quality of the fit and will add additional features and improvements to our base model as necessary.

We study the summary of the units variable, and observe that there maybe some outliers in the data. There are two observation with units sold greater than 1000, which we have removed to prevent the model from being skewed.

Table 0.03: A summary of the Units

Measure	Value
Min	0.000
1st Quantile	0.000
Median	0.000
Mean	0.987
3rd Quantile	0.000
Max	5568.000

Summary of the Simple Linear Regression is shown below:

Observations	236036
Dependent variable	units
Type	OLS linear regression

F(189,235846)	3166.41
R²	0.72
Adj. R²	0.72

	Est.	S.E.	t val.	p
(Intercept)	170.09	2.73	62.34	0.00

Standard errors: OLS

	Est.	S.E.	t val.	p
date	-0.01	0.00	-58.73	0.00
store_nbr2	19.17	0.88	21.74	0.00
store_nbr3	7.22	0.87	8.31	0.00
store_nbr4	42.54	0.69	61.54	0.00
store_nbr5	9.74	0.56	17.42	0.00
store_nbr6	5.28	0.93	5.66	0.00
store_nbr7	17.87	0.63	28.22	0.00
store_nbr8	5.43	0.64	8.43	0.00
store_nbr9	7.85	0.61	12.89	0.00
store_nbr10	14.31	0.54	26.52	0.00
store_nbr11	4.49	0.57	7.92	0.00
store_nbr12	14.26	0.55	25.77	0.00
store_nbr13	17.66	0.60	29.52	0.00
store_nbr14	-14.11	0.50	-28.02	0.00
store_nbr15	14.94	1.24	12.02	0.00
store_nbr16	26.57	0.74	35.80	0.00
store_nbr17	78.40	0.71	111.07	0.00
store_nbr18	-9.12	0.71	-12.90	0.00
store_nbr19	5.13	0.55	9.31	0.00
store_nbr20	17.71	0.89	19.90	0.00
store_nbr21	-16.11	0.69	-23.39	0.00
store_nbr22	7.25	0.55	13.08	0.00
store_nbr23	-16.23	0.67	-24.17	0.00
store_nbr24	43.08	1.14	37.94	0.00
store_nbr25	20.50	1.25	16.40	0.00
store_nbr26	-4.63	0.66	-7.01	0.00
store_nbr27	2.65	0.56	4.70	0.00
store_nbr28	-6.98	0.91	-7.69	0.00
store_nbr29	-0.04	0.72	-0.05	0.96
store_nbr30	19.49	0.63	30.96	0.00
store_nbr31	12.75	0.59	21.45	0.00
store_nbr32	2.61	1.24	2.10	0.04
store_nbr33	48.50	0.63	77.35	0.00
store_nbr34	16.27	0.60	27.29	0.00
store_nbr35	10.32	0.62	16.56	0.00
store_nbr36	-32.39	0.67	-48.36	0.00
store_nbr37	1.22	1.25	0.98	0.33
store_nbr38	19.41	0.86	22.68	0.00
store_nbr39	-46.62	0.69	-67.97	0.00
store_nbr40	6.65	1.24	5.35	0.00
store_nbr41	11.28	0.54	20.94	0.00
store_nbr42	-0.61	0.94	-0.64	0.52
store_nbr43	14.45	0.56	25.97	0.00
store_nbr44	7.62	0.55	13.74	0.00
store_nbr45	-13.87	0.63	-22.11	0.00
item_nbr2	-30.50	0.93	-32.77	0.00
item_nbr3	-47.17	1.10	-42.90	0.00
item_nbr4	-19.66	1.13	-17.36	0.00
item_nbr5	12.31	0.83	14.89	0.00
item_nbr6	-33.75	1.44	-23.50	0.00
item_nbr7	-50.42	1.12	-45.00	0.00
item_nbr8	-0.23	1.15	-0.20	0.85

Standard errors: OLS

	Est.	S.E.	t val.	p
item_nbr9	15.30	0.80	19.04	0.00
item_nbr10	-40.47	1.09	-37.28	0.00
item_nbr11	-51.55	1.09	-47.21	0.00
item_nbr12	-52.63	1.06	-49.72	0.00
item_nbr13	-52.59	1.10	-47.80	0.00
item_nbr14	-38.08	1.15	-33.18	0.00
item_nbr15	-34.96	0.87	-40.28	0.00
item_nbr16	-15.16	0.85	-17.81	0.00
item_nbr17	-46.89	1.05	-44.60	0.00
item_nbr18	-50.23	1.12	-44.99	0.00
item_nbr19	-34.74	1.06	-32.90	0.00
item_nbr20	-42.18	1.11	-38.14	0.00
item_nbr21	-40.86	0.97	-42.25	0.00
item_nbr22	-19.85	1.13	-17.53	0.00
item_nbr23	-26.34	0.98	-26.86	0.00
item_nbr24	-39.66	1.10	-35.89	0.00
item_nbr25	42.94	0.96	44.72	0.00
item_nbr26	-16.42	1.13	-14.50	0.00
item_nbr27	-64.80	1.12	-57.93	0.00
item_nbr28	-28.57	1.10	-25.93	0.00
item_nbr29	-16.30	1.12	-14.55	0.00
item_nbr30	0.24	0.90	0.27	0.79
item_nbr31	-49.06	1.08	-45.55	0.00
item_nbr32	-42.53	1.11	-38.46	0.00
item_nbr33	-19.42	1.07	-18.10	0.00
item_nbr34	-19.91	1.13	-17.58	0.00
item_nbr35	-25.25	1.13	-22.40	0.00
item_nbr36	26.06	0.99	26.39	0.00
item_nbr37	-31.09	0.87	-35.66	0.00
item_nbr38	-32.02	1.06	-30.09	0.00
item_nbr39	-49.41	0.97	-50.85	0.00
item_nbr40	-33.56	1.10	-30.46	0.00
item_nbr41	-16.42	1.00	-16.47	0.00
item_nbr42	-36.77	1.09	-33.87	0.00
item_nbr43	-28.38	1.44	-19.76	0.00
item_nbr44	86.79	0.80	108.01	0.00
item_nbr45	27.91	0.83	33.64	0.00
item_nbr46	-50.17	1.10	-45.61	0.00
item_nbr47	-31.24	1.10	-28.36	0.00
item_nbr48	-30.64	1.14	-26.98	0.00
item_nbr49	-47.59	0.91	-52.47	0.00
item_nbr50	-38.33	0.87	-43.91	0.00
item_nbr51	-39.59	0.88	-45.02	0.00
item_nbr52	-49.92	0.96	-52.18	0.00
item_nbr53	-33.34	1.06	-31.32	0.00
item_nbr54	-40.07	1.10	-36.54	0.00
item_nbr55	-38.39	1.06	-36.29	0.00
item_nbr56	-24.30	1.00	-24.24	0.00
item_nbr57	-23.80	1.14	-20.84	0.00
item_nbr58	-34.27	1.06	-32.44	0.00
item_nbr59	-37.85	1.10	-34.28	0.00
item_nbr60	-76.14	1.44	-53.01	0.00

Standard errors: OLS

	Est.	S.E.	t val.	p
item_nbr61	-42.79	0.91	-46.93	0.00
item_nbr62	-40.41	1.10	-36.64	0.00
item_nbr63	-40.60	1.10	-36.75	0.00
item_nbr64	-59.62	1.10	-54.16	0.00
item_nbr65	-52.73	1.10	-47.93	0.00
item_nbr66	-26.83	1.10	-24.28	0.00
item_nbr67	-39.00	1.08	-36.24	0.00
item_nbr68	-1.47	0.88	-1.68	0.09
item_nbr69	-38.08	1.11	-34.43	0.00
item_nbr70	-19.23	1.07	-17.93	0.00
item_nbr71	-111.21	1.14	-97.94	0.00
item_nbr72	-23.77	1.14	-20.81	0.00
item_nbr73	-47.45	1.10	-43.16	0.00
item_nbr74	-49.91	1.12	-44.71	0.00
item_nbr75	-38.56	1.11	-34.86	0.00
item_nbr76	-76.54	1.44	-53.29	0.00
item_nbr77	-59.36	1.10	-53.92	0.00
item_nbr78	-39.67	1.10	-36.17	0.00
item_nbr79	-46.92	1.10	-42.73	0.00
item_nbr80	-35.56	1.10	-32.27	0.00
item_nbr81	-48.55	1.06	-45.86	0.00
item_nbr82	-32.02	1.16	-27.67	0.00
item_nbr83	-29.57	1.11	-26.73	0.00
item_nbr84	-47.31	0.91	-52.04	0.00
item_nbr85	-65.07	0.90	-71.93	0.00
item_nbr86	-44.49	0.85	-52.14	0.00
item_nbr87	-36.70	1.10	-33.29	0.00
item_nbr88	-28.96	0.96	-30.07	0.00
item_nbr89	-33.66	1.10	-30.56	0.00
item_nbr90	-75.21	1.12	-67.23	0.00
item_nbr91	-16.70	1.11	-15.08	0.00
item_nbr92	-81.18	1.08	-75.39	0.00
item_nbr93	-45.54	0.82	-55.82	0.00
item_nbr94	-40.15	1.10	-36.40	0.00
item_nbr95	-33.28	1.12	-29.70	0.00
item_nbr96	-75.85	1.12	-67.80	0.00
item_nbr97	-75.65	1.12	-67.62	0.00
item_nbr98	-48.52	0.96	-50.74	0.00
item_nbr99	-33.30	1.10	-30.23	0.00
item_nbr100	-38.10	1.10	-34.50	0.00
item_nbr101	-75.87	1.12	-67.82	0.00
item_nbr102	-40.30	1.09	-36.81	0.00
item_nbr103	13.59	1.15	11.83	0.00
item_nbr104	-42.57	0.89	-48.03	0.00
item_nbr105	-41.35	0.87	-47.76	0.00
item_nbr106	-44.65	0.96	-46.46	0.00
item_nbr107	-37.80	1.15	-32.94	0.00
item_nbr108	-42.39	1.10	-38.69	0.00
item_nbr109	-26.41	0.97	-27.35	0.00
item_nbr110	-36.25	1.10	-32.88	0.00
item_nbr111	13.60	1.15	11.85	0.00
dayMonday	1.42	0.15	9.35	0.00

Standard errors: OLS

	Est.	S.E.	t val.	p
daySaturday	3.70	0.15	24.29	0.00
daySunday	6.06	0.15	39.89	0.00
dayThursday	-2.02	0.15	-13.27	0.00
dayTuesday	-0.76	0.15	-4.97	0.00
dayWednesday	-1.73	0.15	-11.40	0.00
month2	-0.73	0.19	-3.77	0.00
month3	-1.60	0.20	-7.97	0.00
month4	-1.51	0.23	-6.57	0.00
month5	-1.50	0.26	-5.77	0.00
month6	-0.30	0.30	-1.00	0.32
month7	-0.19	0.31	-0.60	0.55
month8	0.49	0.30	1.63	0.10
month9	-0.36	0.27	-1.34	0.18
month10	-0.54	0.23	-2.40	0.02
month11	-0.90	0.22	-4.08	0.00
month12	0.67	0.21	3.14	0.00
seasonSpring	NA	NA	NA	NA
seasonSummer	NA	NA	NA	NA
seasonWinter	NA	NA	NA	NA
tmax	-0.01	0.00	-4.46	0.00
tmin	-0.00	0.01	-0.55	0.58
tavg	-0.06	0.01	-7.28	0.00
depart	-0.01	0.00	-1.71	0.09
dewpoint	0.01	0.00	1.85	0.06
wetbulb	0.03	0.01	3.00	0.00
heat	0.00	0.00	0.09	0.93
cool	0.07	0.01	6.33	0.00
sunrise	-0.00	0.00	-0.73	0.46
sunset	-0.00	0.00	-0.12	0.90
snowfall	0.02	0.01	1.68	0.09
preciptotal	-0.00	0.00	-0.68	0.49
stnpressure	0.00	0.01	0.61	0.54
sealevel	-0.01	0.01	-1.38	0.17
resultspeed	0.00	0.00	0.05	0.96
resultdir	0.01	0.00	1.78	0.08
avgspeed	0.00	0.00	0.55	0.58

Standard errors: OLS

Simple Linear Regression: We start with a basic linear model with all the variables at our disposal.

This model has an R^2 of 0.72, which implies that the predictors explain the model to certain extent. We also observe that most weather variables having a p-value of greater than 0.05. This implies that they are not significant for the model, and could be removed for further analysis. Variables like store number, item number, days in a week, month are significant in the model. The interpretation of the coefficients can be as follows, if we take a look at the coefficient of `tavg` , a one unit increase in the average temperature, results in decline of units sold by a factor of - 0.06. However, to gauge the performance of the linear model we run some diagnostics to check if the assumptions of the model hold. Based on our assesment of these assumption we make further improvements to our model.

Assumptions in SLR:

- Linear relationship
- Multivariate normality
- No or little multicollinearity
- No auto-correlation
- Homoscedasticity(Error terms have constant variance)

For all the item numbers which we did not consider for training the model, we will assign a prediction of zero. We will then merge this with the item numbers for which we made the predictions using our model and get the final submission file. Submitting this file on Kaggle, we were able to achieve a score of 0.23442 which is significantly better than the baseline score of 0.52 which is attainable with all zero predictions. This parameter used to assess the accuracy of the models on kaggle is Root Mean Squared Logarithmic Error(RMSLE).

Diagnostics

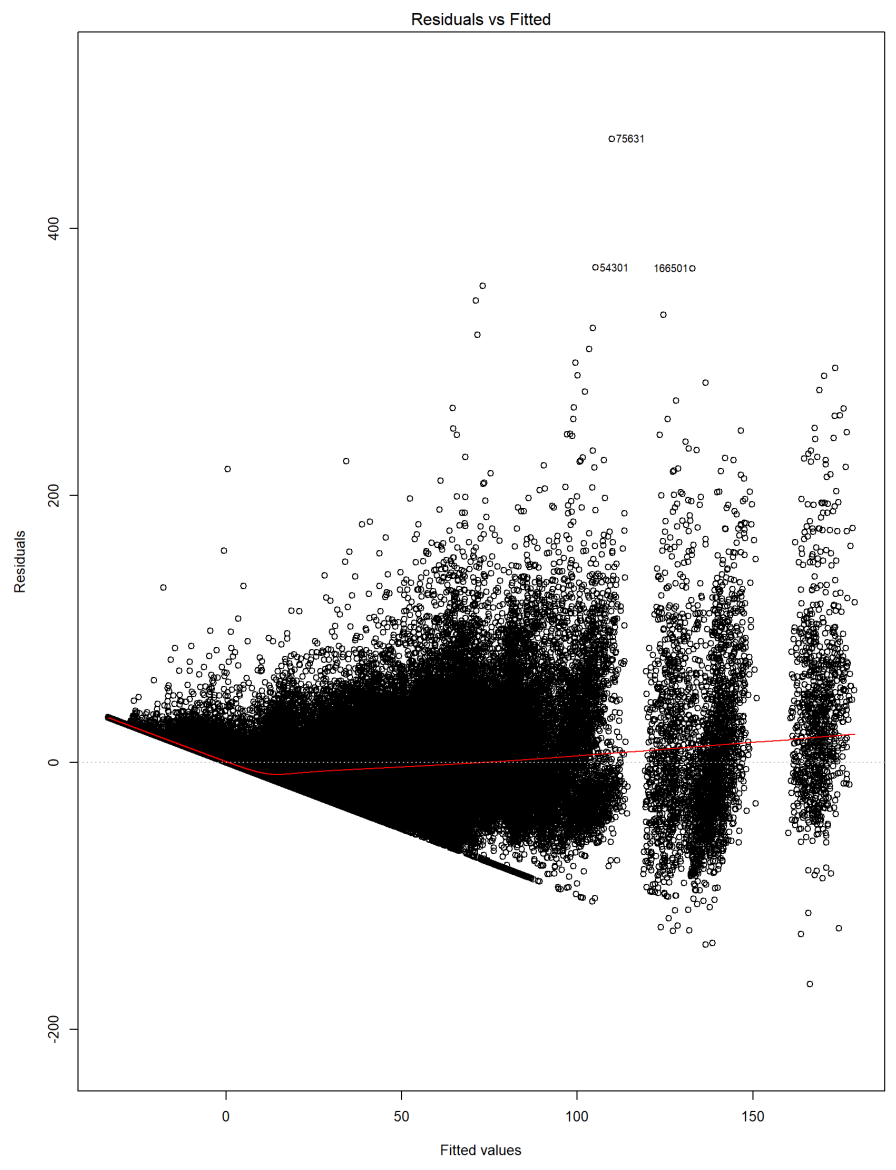
Below the diagnostics plots are shown.

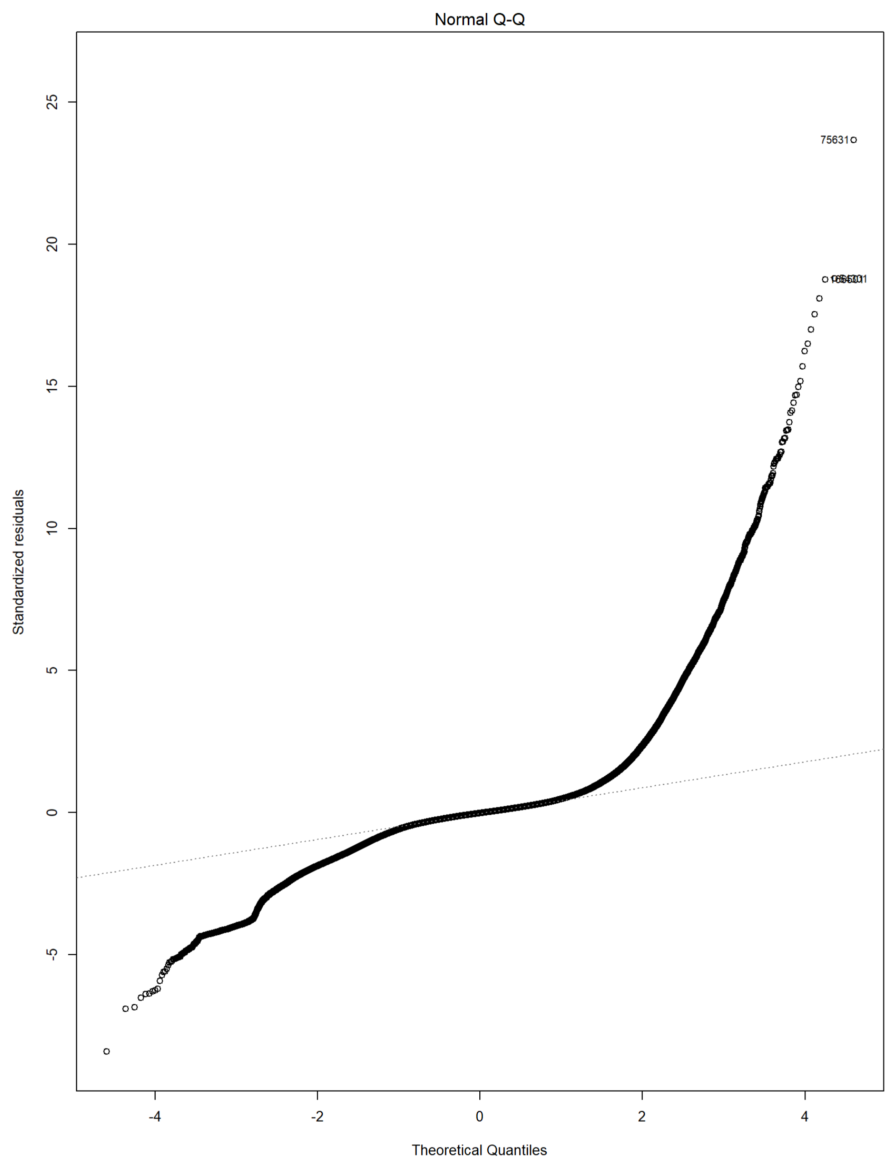
Plot 01: Residuals vs Fitted This plot shows if residuals have non-linear patterns. There could be a non-linear relationship between the response variable and the predictor variables and if it exist then we will see a clear pattern in the plot. However, in our plot we see that there are no obvious patterns in the points plotted and its pretty random. We can conclude with reasonable confidence that the assumption of linear relationship stands corrected. if we find equally spaced residuals around a horizontal line without distinct patterns then it is a good indication that non-linear relationships are not present.

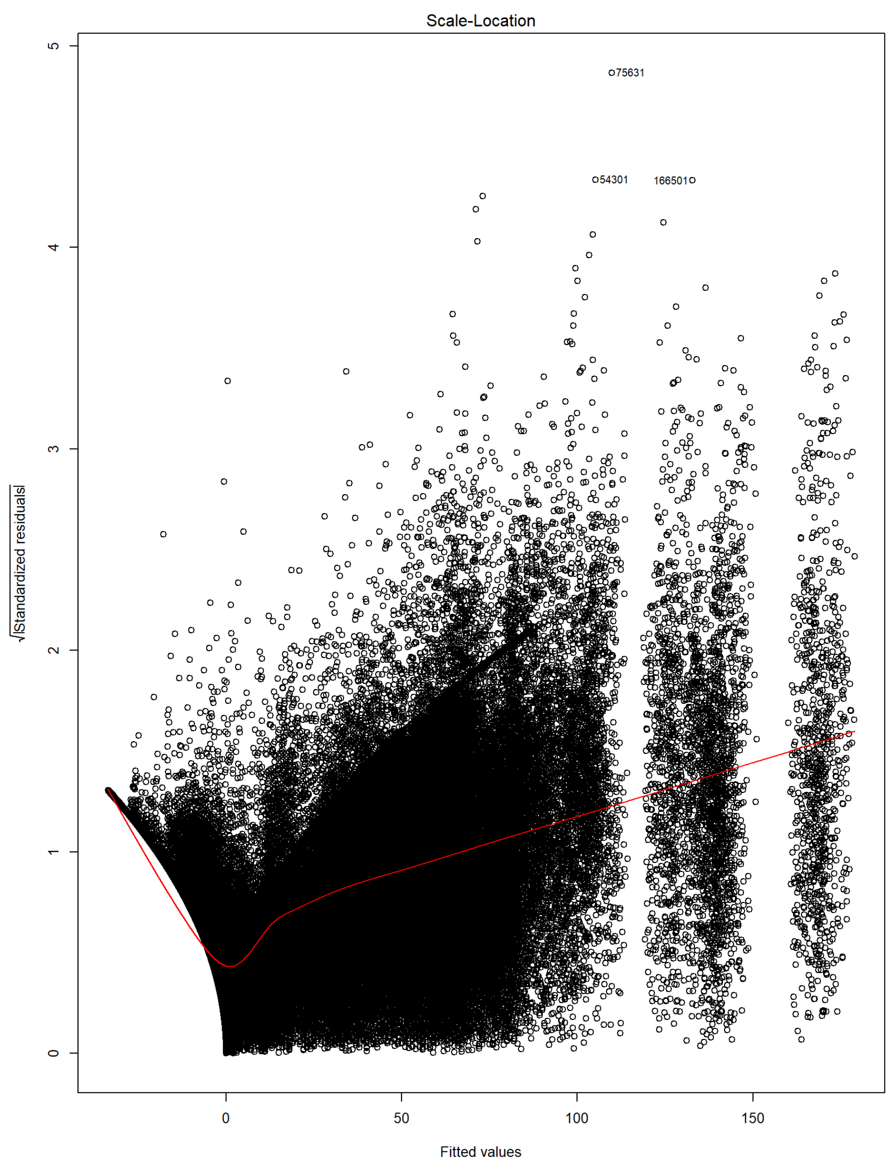
Plot 02: Normal QQ This plot shows if the residuals are normally distributed which test the assumption of normality in the simple linear regression. If the residuals follow a straight line, we can conclude that they satisfy the normality criteria but the curve that we see here in the plot below deviates from the straight line. While the curve may not always follow a straight line perfectly but the deviation is too much to ignore and we can safely conclude that the residuals do not satisfy the normalcy criteria. We will employ transformation techniques like box-cox to correct for this assumption fail.

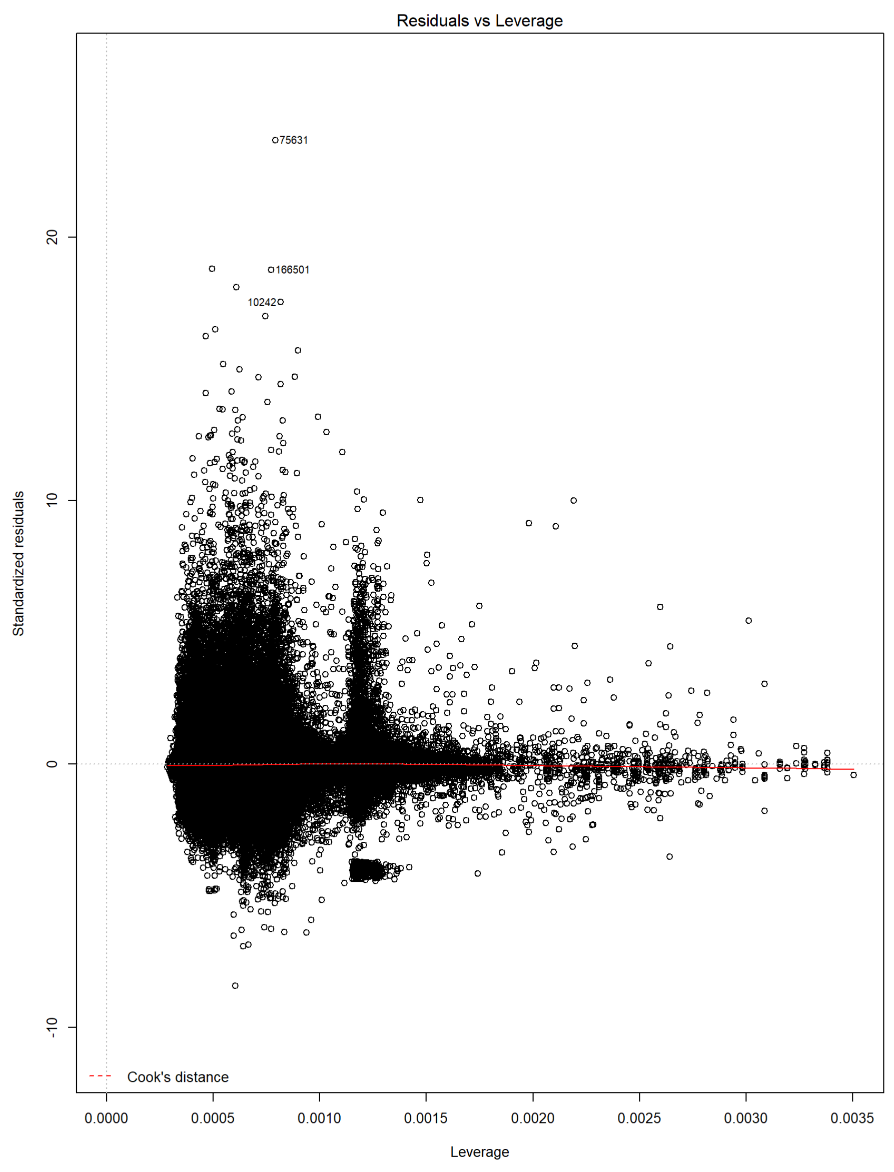
Plot 03: Scale Location This plot is also called spread-location plot. This plot if residuals are spread equally along the ranges of predictions. Using this graph, we can check the assumption of homoscedasticity(equal variance). It is generally considered good if the points are equally distributed on along the horizontal line in the middle. In this case, the residuals appear to spread out more as we move along the horizontal axis. Because the points are not spread equally we don’t see a smooth straight horizontal line through the middle.

Plot 04: Residuals vs Leverage This plot helps us to find influential points. In Regression, there are two kinds of outliers, influential and non-influential. Even there may exist some extreme points but they may not have enough influence to shift the regression line much. This means that the results wouldn’t change much with the inclusion or exclusion of these points. To look for the influential points, we look at the cook’s distance which is represented by the red curves in the graph however those are not visible in this graph due to being out of scale as none of our points are even close to being influential. Unlike the first three plots, the pattern in the plot is not relevant here. We watch for outlying values at the upper right and lower right corner.









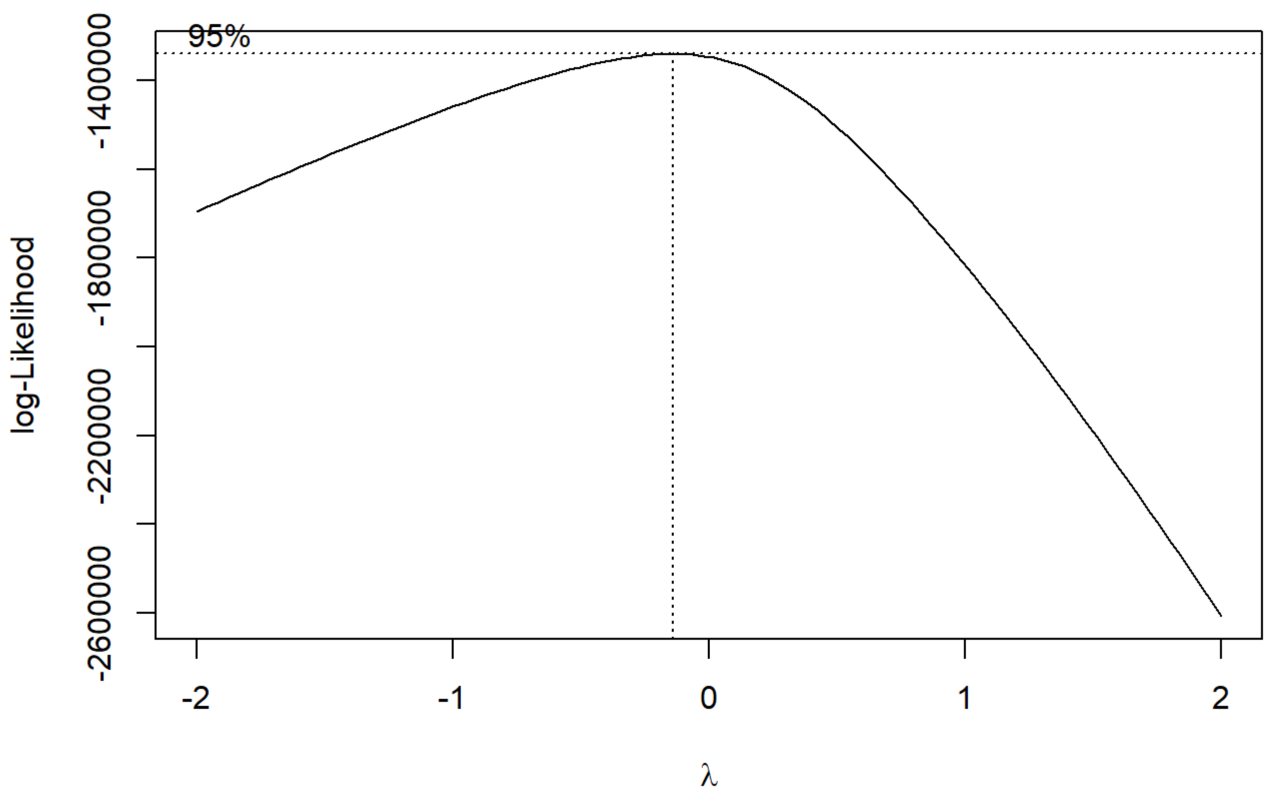
Studentized Breusch-Pagan Test

Name	Value
BP	42837.96
Degrees of Freedom	189.00
P-Value	0.00

Breusch-Pegan Test THE BP test is a test conducted to check the homoscedasticity(constant variance assumption). Its basically a hypothesis test with the null hypothesis being that the variance of the residuals is constant. If our p-value is sufficiently small, in this case smaller than 0.05 significance level, we reject the null hypothesis and conclude that the variance is not constant. We have already seen through the diagnostic plots that the variance is not constant and the BP test confirms this as our p-value is basically zero. Hence we reject the null hypothesis and conclude that the alternate hypothesis is true i.e the assumption of homoscedasticity is incorrect.

Improvements

Box-Cox



Linear Regression assumes that the errors follow a normal distribution with mean= 0 and with a constant variance. This assumption might be false in various ways. One way to fix this is to transform Y to a function h(Y) such that the variance of h(Y) is constant. As we can see from the results of the Breusch-Pegan Test, assumption of the errors being homoskedastic does not hold true. Hence we perform the Box cox transformation.

Box cox transformation provides a way in which we can solve for the heteroskesticity (non constant variance) and non normality of errors. Suppose that if we know the dependent variable is positive, and consider the model:

$$y^{\lambda} = XB + \epsilon$$

where $var(\epsilon)$ is constant and y^{λ} is the element-wise power of lambda. When lambda will be equal to 1, we will get the original linear model.

Box Cox transformation method tries to choose a lambda that aims at maximizing the likelihood of the data and assumes that the erros are normally distributed. We plot the Likelihood on y axis verses the lambda on x axis and choose the lambda for which we get the likelihood as maximum.

From the above graph, we observe that the lambda for which the likelihood is maximum falls between -1 and 0 and closer to zero. Since the lambda is closer to zero, we can consider the log transformation of Y, to achieve a constant variance of errors.

After deciding on the log-transformation, we transformed the units variable to $ln(units + 1)$ and fit the simple linear regression again. With this model we were able to achieve an R^2 score of 0.8771235. We also achieved a kaggle score of 0.16624.Looking at the diagnostics plot for the log transformed model, we can clearly see that we are closer to linear model assumptions as compared to simple least regression without transformation. One point to note here is that while making predictions we need to convert the $ln(units + 1)$ back to units using the exp function. Since we are decided on the log-transformation, we will consider this model as the base of all further improvements or modification we make.

Feature Engineering

To facilitate a better understanding of the underlying structure of the data, new predictor variables are created based on the observations through exploratory analysis and observation of the data. It is intuitive that the sales in each day will vary with the position of the day in the month or in a year. Hence variables like month day and day in year were generated in the data. Position of the day in the week was also created as it makes sense that sales might be more on a weekend as compared to weekdays. This is confirmed by our analysis in the EDA section.

Additionally, from observing the data, we noticed that the sales in each month vary significantly. Hence, we created a monthly average sales for each product which will serve as a new feature. Based on this, we can also analyse of the monthly sales for any product are zero.

Temperature is also another important feature as we noticed in the EDA section that people avoid shopping in extreme cold or hot days. In addition, “feels like” temperature might be more relevant to this particular analysis which is related to the moisture in the air hence two new features were created identifying the moisture in the air. The first predictor calculates the difference between average temperature and dew point temperature as it signifies how far away the moisture in the air is from saturation. The second predictor calculates the difference between the wet bulb temperature and average temperature. This signifies the relative humidity in the air. The larger the difference, the lower the relative humidity is.

Predictors like precipitation, snowfall and average wind speed are included without any modifications. Later in the analysis we removed the variables with high correlation based on Variance Inflation Factor. The feature `codesum` was ignored for the purpose of this analysis as the data contained too many missing values and most of the info was already covered in the precipitation, snowfall and wind features.

Because some of the store numbers and if numbers combinations have zero sales throughout the time period covered by the data, we have excluded them from the training data we have used for training the models. Prediction of zero units was assigned to all such cases and merged with the model predictions to prepare the submission file.

All these features that we have created and modified for the training dataset, we also need to create those for the test dataset to ensure consistency while making predictions with the trained model.

Multicollinearity

Multicollinearity is difficult to identify. When two or more independent variables are related to each them, we call them multicollinear. This results in unreliable and highly variable regression coefficients. Also, with multicoliinearity. it becomes difficult to interpret the impact of change of one unit in a variable on the change in the response variable. Some of the issues with multicollinearity are: - If some of the predictors are collinear, then model matrix becomes singular and the inverse of such matrix does not exist. - The least square estimate of the coefficients will not be unique

One way to check for multicollinearity is to use the Variance Inflation factor(VIF). VIF gives a measure of collinearity of a given predictor with all the other predictors. VIF of a given predictor is calculated by creating a linear regression model with the given predictor as the response variable and all the other predictors as the independent variables. If the coefficient of determination of this model is large, then we can say that the given predictor is collinear with other predictors.

$VIF = 1/(1 - R^2)$

VIF ranges from 1 to infinity. Predictors with the VIF greater than 5, are considered to be highly collinear with the other predictors. We have shown the VIF of all the predictors. Before checking for VIF we used the alias function in R to find linearly dependent terms in our model. We remove these terms and then proceed with VIF. We have shown the VIF of all the predictors. Predictor with high VIF per degree of freedom are removed in a stepwise manner. We then build a linear regression model excluding these predictors one at a time, and then finalising the model.

Variance Inflation Factor

Variable	GVIF	DF	GVIF^(1/(2*DF))
store_nbr	1.654714e+14	44	1.45070
item_nbr	4.976034e+12	110	1.14212
day	1.009980e+00	6	1.00083
season	8.265880e+00	3	1.42194
tmax	2.148820e+00	1	1.46588
tmin	6.468710e+00	1	2.54337
tavg	2.310519e+01	1	4.80679
depart	1.216620e+00	1	1.10300
dewpoint	7.463050e+00	1	2.73186
wetbulb	2.906607e+01	1	5.39130
heat	1.465720e+00	1	1.21067
cool	4.419530e+00	1	2.10227
sunrise	3.467500e+00	1	1.86212
sunset	3.726240e+00	1	1.93035
snowfall	1.045930e+00	1	1.02270
preciptotal	1.118780e+00	1	1.05773
stnpressure	1.094569e+02	1	10.46217
sealevel	7.303700e+00	1	2.70254
resultspeed	1.566100e+00	1	1.25144
resultdir	1.128680e+00	1	1.06239

Variable	GVIF	DF	GVIF^(1/(2*DF))
avgspeed	1.273880e+00	1	1.12866
monthly_average	8.142090e+00	1	2.85343
day_in_month	1.023180e+00	1	1.01152
day_in_year	2.229390e+00	1	1.49312
year	1.073710e+00	1	1.03620
temp_diff_dew	2.750880e+00	1	1.65858
temp_diff_wb	3.064210e+00	1	1.75049

Variance Inflation Factor

Variable	GVIF	DF	GVIF^(1/(2*DF))
store_nbr	4.162274e+12	44	1.39124
item_nbr	4.975873e+12	110	1.14212
day	1.009720e+00	6	1.00081
season	7.968090e+00	3	1.41327
tmax	2.042390e+00	1	1.42912
tmin	6.236260e+00	1	2.49725
depart	1.199650e+00	1	1.09529
dewpoint	7.068240e+00	1	2.65862
wetbulb	1.407098e+01	1	3.75113
heat	1.446160e+00	1	1.20256
cool	4.184270e+00	1	2.04555
sunrise	3.463960e+00	1	1.86117
sunset	3.705640e+00	1	1.92500
snowfall	1.043420e+00	1	1.02148
preciptotal	1.110930e+00	1	1.05401
sealevel	1.476310e+00	1	1.21503
resultspeed	1.558080e+00	1	1.24823
resultdir	1.128280e+00	1	1.06221
avgspeed	1.273600e+00	1	1.12854
monthly_average	8.141780e+00	1	2.85338
day_in_month	1.023170e+00	1	1.01152
day_in_year	2.228510e+00	1	1.49282
year	1.069880e+00	1	1.03435
temp_diff_dew	2.357230e+00	1	1.53533
temp_diff_wb	3.048490e+00	1	1.74599

We removed the variables with the high variance inflation factor per degree of the freedoms and then for the log transformed linear regression model. We achieved an R^2 of 0.8869016. With this model we achieved an kaggle score of 0.16354 which is not much different from the log model that we fit earlier in the analysis which is expected because we knew that since these variables were correlated with other predictors, most of the information that they provided was already provided through other variables so we won't see any big differences in the results.

Model Selection Methods

AIC - BIC

In general, adding a new predictor results in increase of the coefficient of determination even if the predictor is not significant. As a result, R^2 always tends to favor large models, which results in very complex models and tend to overfit the training data resulting in poor performance on new previously unseen data. To avoid overfitting in such cases, we want to penalize the additional parameters which are increasing the performance on the training data set but are in fact resulting in overfitting.

To penalize addition of new predictors, we have considered the following two measures:

- *Akaike Information Criteria (AIC)* AIC is the penalized log likelihood measure.

$AIC = 2k - 2ln(L)$

where L is the maximum likelihood and k is the number of parameters. As we increase the number of parameters, the value of AIC will increase. Hence we prefer the model with the lowest AIC value.

- *Bayesian Information Criteria(BIC)* BIC is another measure for model comparison. It is similar to AIC but it has a higher penalty term as compared to AIC. As a result of this larger penalty, BIC chooses smaller models as compared to AIC.

$BIC = 2nln(k) - 2ln(L)$

where L is the maximum likelihood and k is the number of parameters and n is the number of rows in the data. As we increase the number of parameters, the value of BIC will increase. Hence we prefer the model with the lowest BIC value.

Feature Subset Selection:

In the linear regression model, not all the predictors are significant in predicting the response variable. Removing the non significant predictors can result in improving the accuracy of the least square fit. There are several ways to select the significant predictors.

- Best Subset Selection
- Stepwise Selection

In the best subset selection method, all possible combinations of the predictors are considered. We then consider the best model out of all these possible models. This can be computationally expensive as the number of predictors becomes large.

Another way of selecting predictors for linear regression is the stepwise Selection. In stepwise selection, we iteratively add or remove variables till we get the subset of predictors that results in the best models with lowest prediction error. There are two ways in which we can start adding or removing predictors: - Forward Selection - Backward Selection

In forward selection, we start with a null model and keep on adding predictors one at a time till we reach a stopping condition.

- Let M0 be the null model.
- Consider all the remaining predictors. Add one predictor at a time to the existing model.
- Choose the predictor that results in the lowest AIC or BIC value for the model.
- Repeat step 2 and 3 till we do not see any improvement in the AIC or BIC value.

In Backward selection, we start with a full model and keep on dropping those predictors one at a time till we reach a stopping condition.

- Let M0 be the full model.
- Consider all remaining predictors. Drop one predictor at a time from the existing model.
- Choose the predictor that results in the lowest AIC or BIC value for the model.
- Repeat step 2 and 3 till we do not see any improvement in the AIC or BIC value.

Since we are starting with the full model, we have chosen to use backward selection method for the purpose of feature selection. We have compared the performance of the models by choosing both AIC and BIC criteria to select the subset of predictors.

We made predictions with both the aic and bic based step regression models and received kaggle score of 0.16354 and 0.16353 respectively which are not much different from the the last model we submitted after removing all the factors with high variance inflation factor. This shows that most of the features that we are left with post vif analysis are contributing to the predictions. We have stored these variables in the cache of the markdown file so that we don’t have to rerun the process everytime we knit the file as the process can be time-consuming.

Ridge/Lasso Regularization

Ridge and Lasso regression are regularization methods to reduce the complexity of the model and shrink the coefficients so that model is simpler, more interpretative and less prone to overfitting. In ridge regression, the cost function is modified by adding a penalty equivalent to the square of the magnitude of the coefficients. This is saying to minimize the normal RMSE function within the constraint on the coefficients:

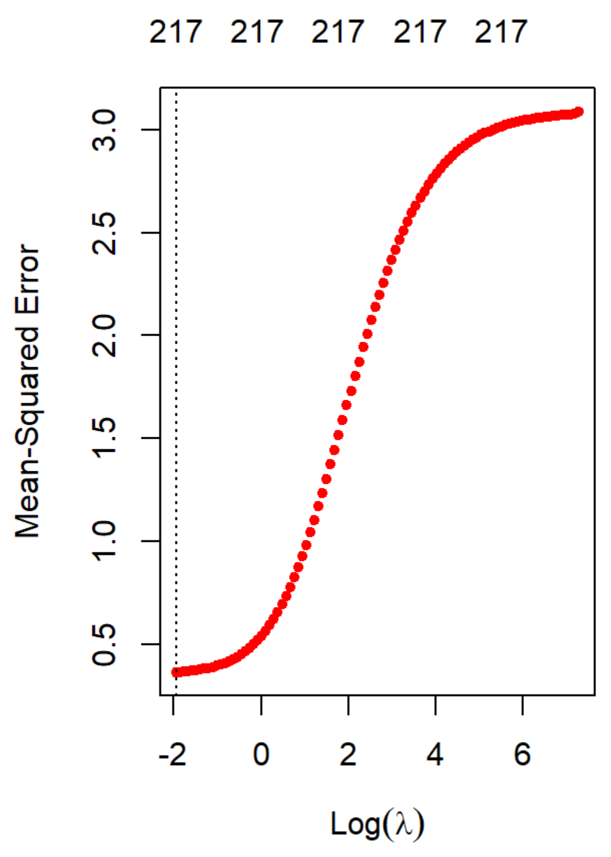
$$\sum_{j=0}^{p} w_i^2 < c$$

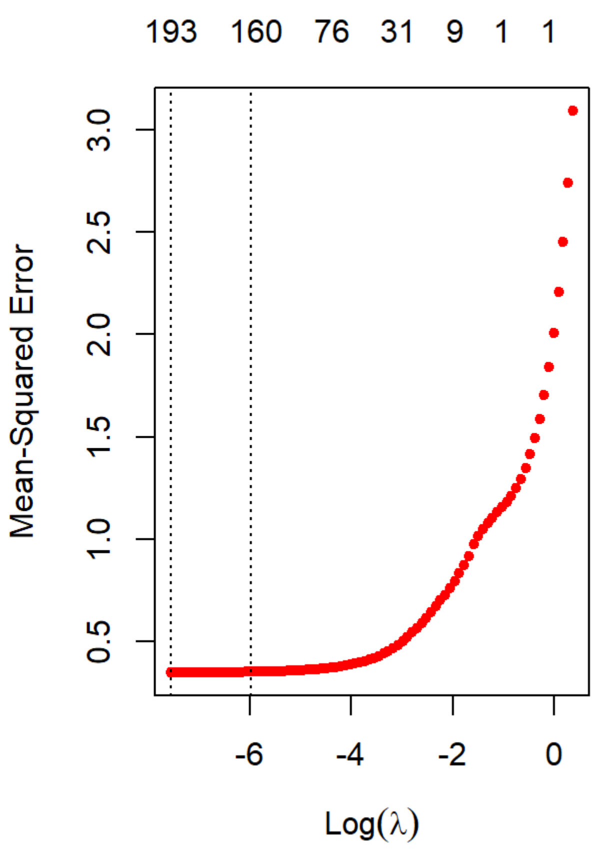
The penalty term λ regularizes the coefficients such that if the coefficients take large values the optimization function is penalized. This results in the shrinkage of the coefficients and helps to reduce the model complexity and multi-collinearity. A lower constraint will make the model resemble the simple linear regression.

The cost function of Lasso(least absolute shrinkage and selection operator) regression is minimizing the rmse with the constraint on the coefficients in the form below:

$$\sum_{j=0}^{p} |(w_i)| < t$$

The only difference here is that instead of taking the square of the the coefficients, magnitudes are taken into account. This type of regularization can lead to zero coefficients. This means some of the features are completely removed from the model. So Lasso regression not only helps us avoid overfitting but also helps in feature selection. We have plotted the graphs for both ridge and lasso below and we can see that lasso reduces the number of features significantly.





Best Model post Improvements

The best model post improvements is the Stepwise Linear Regression with AIC. As explained above, we first transformed our response variable to log, then removed variables based on VIF and further used the stepwise methodsuing AIC criteria.

This model has a R^2 of 0.8869011, which is much better than our initial model. We have `store_nbr` , `item_nbr` , `day` , `season` , and some weather variables as predictors. THe variables are significant with p-values less than 0.05. This model give us a kaggle score of 0.16354.

Advanced Methods

Additional Features

Before we move on to some advanced machine learning algorithms, we created a couple of extra features which based on our evaluation of the data so far we think may be useful predictors. We created the features in a separate code file and then created a csv file to be used with the mail analysis.

The first feature is *ppr_fitted* is calculated using the package `ppr` in `R` . *PPR* stands for Pursuit Regression Model. We use this model to make a baseline model for the units. This feature is created by using the projection pursuit regression model in R. The model is fitted by using only date as the predictor and log(units +1) as the response variable. This is sort of like getting a time series version of the projections and treating it as the base case. The predicted values from this model are used as the baseline. Linear Regression assumes the linear relationship between the response and the predictor variables. In the real-world generally, this assumption may not hold true. PPR is a nonparametric approach that overcomes the limitation of linear assumption. These predicted values result in the smoothing of the response variable.

rmean : This is the rolling mean of log(1+units) over a period of 12 days. This is calculated for each unique combination of store and item number. This can be thought of as the moving average of the log(1+units) to understand the impact of units sold on the days before and after a particular day. It is calculated by taking the unweighted mean of the values before and after the particular day. This also results in smoothing of the response variable that is the log(units+1)

Once we have prepared our final training and test datasets, we can move on to testing some advanced machine learning algorithms on our datasets and compare their performance to the linear models we fitted before this. We considered Random Forests and Gradient Boosting Method as the two methods for the purpose of this analysis.

Random Forest: Random forest is a supervised machine learning algorithm. It builds a “forest” of decision trees which is generally referred to as the bagging method. Bagging is a machine learning ensembling method designed to increase the stability and accuracy of the statistical algorithms and reduce the variance of the algorithms while having low bias. Not only does it help in reducing variance but it also helps in avoiding overfitting. The general idea is to combine a bunch of learning methods to get better and more accurate results.

In simple words, Random Forests builds multiple decision trees and merges them together to get a more accurate and stable precision. Random forest basically builds on the idea of the wisdom of the crowd. For example: If we ask one person to guess a number between 1 and 1000, the number guessed can be wildly different from person to person meaning high variance in your predictions. But if I ask 1000 people the same question and combine the results to get the average as the final prediction the variation is reduced considerably and the probability of the final prediction being closer to the actual number will be higher.

Random Forest can be used for both classification and regression problems. Random forests select a random subset of data with randomly selected subset of the features to build each tree and then combines them at the end. This allows Random Forest to have a large number of deep uncorrelated trees which increases the effectiveness of the forest of the combined trees. To elaborate more, Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of predictors. This results in wide diversity that generally results in a better model.

Another big advantage of random forest algorithm is that it is very convenient to measure the relative importance of each predictor in making effective predictions for the response variable. This importance score is calculated based on looking at how much the tree nodes that use the feature reduce the impurity across all trees in the forest. This score is calculated for each feature and the results are scaled so the sum of all importance adds up to one. By looking at the feature importance you can decide which features to keep in the model and which features are safe to be dropped as their impact or contribution is not significant to the predictions. This is crucial because the more variables you have in the model, the more the chances of overfitting, so by removing features which don't contribute enough we avoid overfitting.

Breaking it further down to the individual tree, each internal node in a decision tree represents a function which performs a test on the feature and each branch represents the outcome of that test. Each leaf node represents the final label or value based on the computation of all attributes. One disadvantage of Random Forest is that because a large number of trees are being created and then the subtrees are being combined, it makes the computation slower depending on the number of trees being built.

Some of the major hyperparameters/tuning parameters of random forest are:

- **n_estimators** is just the number of trees the algorithm builds before combining them to take the average of predictions or taking the maximum voting in case of classification. In general, a higher number of trees increases the performance but slows down the computation.
- **max_features** is the maximum number of features random forest considers to split a single node. Another important feature is **min_sample_leaf** which determines the minimum number of leafs required to split an internal node.

Another advantage of random forest is that the default hyperparameters it uses often produces very good results and understanding the hyperparameters is very easy and they are relatively less in number. There is a running joke in the data science community that if you are confused fit random forest and if that doesn't work move on to deep learning. I feel that this joke speaks to the versatility and power of the random forests.

Gradient Boosting Method: GBM is one of the most popular methods used in Data Science competitions. But most developers still treat it as a black box method. We will try to break it down and lay an intuitive framework for this machine learning technique.

Boosting is a method of converting a series of weak learners into strong learners. In boosting, each new tree is a fit on on a modified version of the original dataset. GBM can be easily explained by first explaining AdaBoost algorithm. The adaboost algorithm begins by training a decision tree in which each observations are equally weighted. Post evaluation of the first tree, the weights of difficult to classify and lower the weights of the observations which are easier to classify. The second tree is therefore grown on the weighted data. The idea is to improve the predictions of the first tree. So the predictions become average of the first and second tree. We then calculate the Root Mean Square error in the case of regression and grow a third tree to predict the revised residuals. This process is repeated for for a specific number of iterations. Subsequent trees will help us make predictions for observations which are not well predicted by the previous trees. Predictions of the final ensemble model is therefore the weighted sum of the predictions made by the previous tree models.

Gradient Boosting trains many models in a gradual, additive and sequential manner. The major difference in gbm as compared to adaboost is in how the two algorithms identify the shortcomings of weak learners. While the adaboost model identifies the shortcomings by using high weight data points, gbm performs the same by using gradients in the loss function. The loss function is a measure indication how good are model's coefficients are at fitting the underlying data. For example, in this case, the loss function would be the error between the true and predicted units. One of the biggest motivations of using a gradient boosting machine is that it allows one to optimize a user define cost function, instead of the default loss functions which may not be suitable to the particular application.

There are two important hyperparameters in the case of gradient boosting, *interaction.depth* and *shrinkage*. Interaction Depth specifies the maximum depth of each tree in the sequence of trees and Shrinkage is considered as the learning rate used for reducing or shrinking the impact of each additional fitted base learner. It reduces the size of incremental steps and thus penalizes the importance of each consecutive iteration.

Hyperparameter tuning is especially significant for gbm modelling since they are prone to overfitting. The special process of tuning the number of iterations for an algorithm such as gbm and random forest is called "Early Stopping". Early stopping performs model optimization by monitoring the model's performance on a separate test data set and stopping the training procedure once the performance on the test data stops improving beyond a certain number of iterations.

It avoids overfitting by identifying the inflection point where performance starts to stagnate on the test data set while the performance keeps improving on the training dataset. The ideal time to stop training the model is when the validation error has decreased and started to stabilize before it starts increasing due to overfitting.

Conclusion

To conclude, we fitted various models to the walmart sales data in stormy weather and tried to predict the sales of different items in different stores. Random Forest performed the best with a kaggle score of 0.14, the best ever score on kaggle is around 0.10 so this is pretty good performance from our model. We were somewhat limited by the computation power we had as were not able to consider the interactions as it created too big a vector to be handled by the computational power we had at our disposal. Apart from the predictions and the score, we applied most of the concepts learned in the Applied Regression class(STAT 425) and understood them much better after applying all these concepts to real world data. The predictions made from this tool can be used by supply managers to assess the demand of certain products in extreme weather conditions and make supply chain decisions accordingly.

Kaggle Results for all Models Tested

Model Name	Kaggle Private Score	Kaggle Public Score
Simple Linear Regression	0.23326	0.23442
SLR with Log Transformation	0.16714	0.16624
SLR with Log transformation post VIF Analysis	0.16395	0.16354
Stepwise Linear Regression with AIC	0.16395	0.16354
Stepwise Linear Regression with BIC	0.16397	0.16353
Random Forest	0.14379	0.14313
Gradient Boosting	0.14394	0.14303

Appendix

Fig 0.02: Missing rows in Training Data

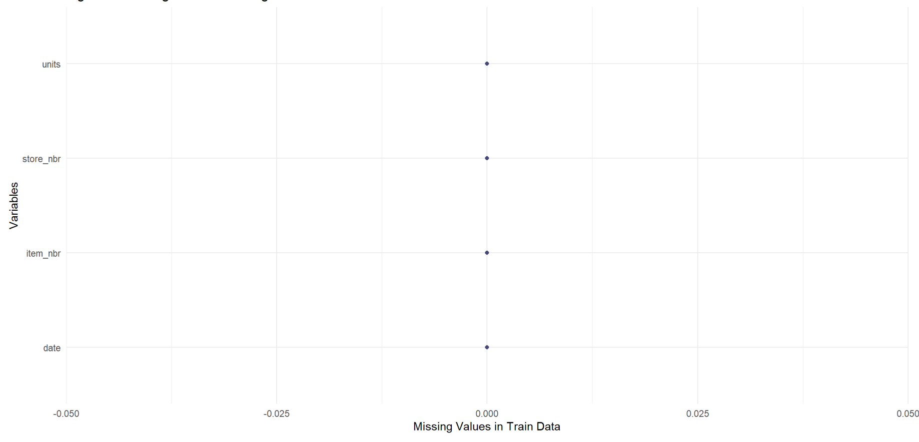


Fig 0.02: Missing rows in Weather Data

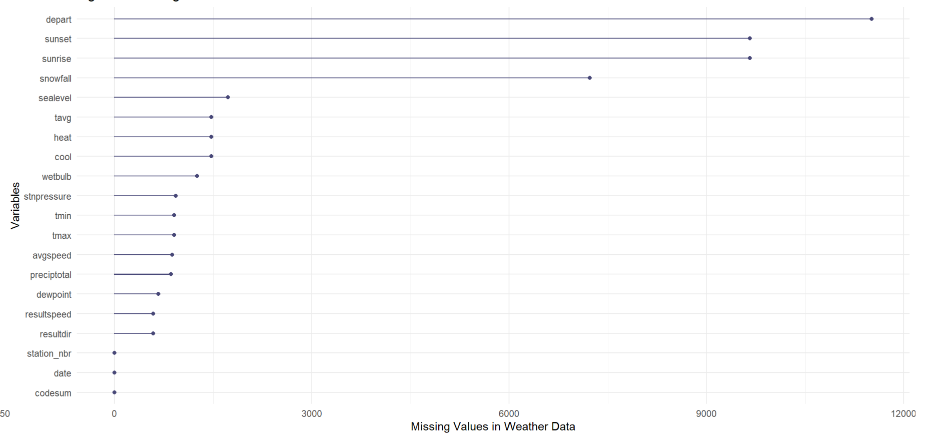


Fig 0.03: Daily Units Sold
Data plotted by Weekday

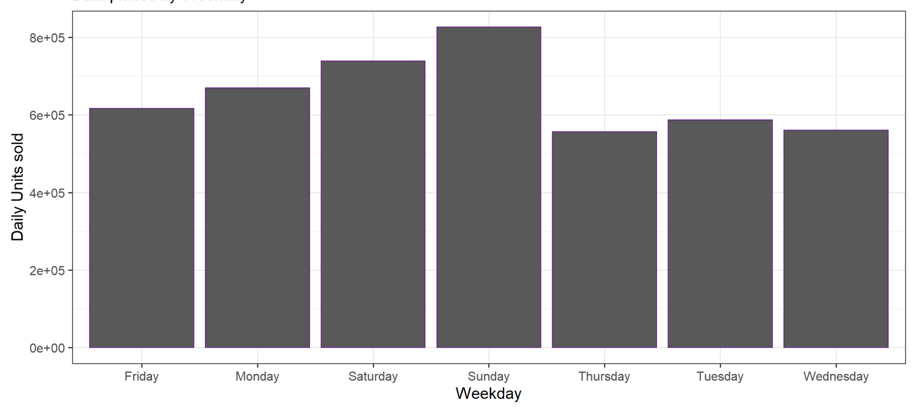


Fig 0.04: Monthly Units Sold
Data plotted by Month

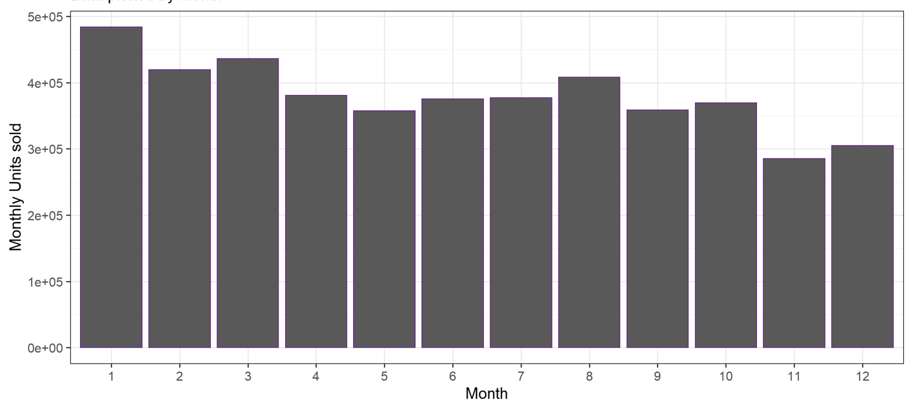


Fig 0.05: Units sold by Season
Data plotted by Season

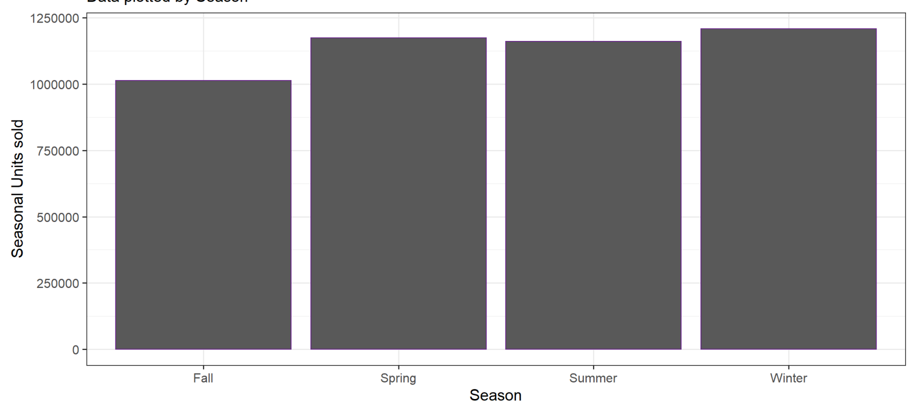


Fig 0.06: Correlation plot for weather data

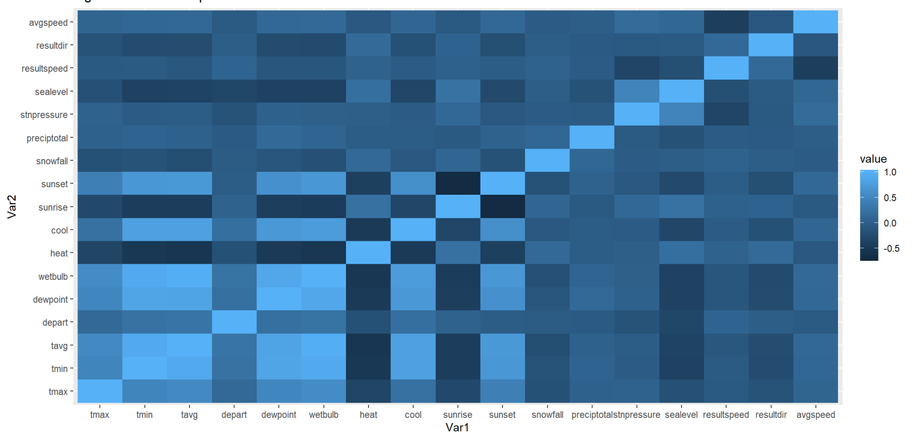


Fig 0.07: Units vs Wind Speed

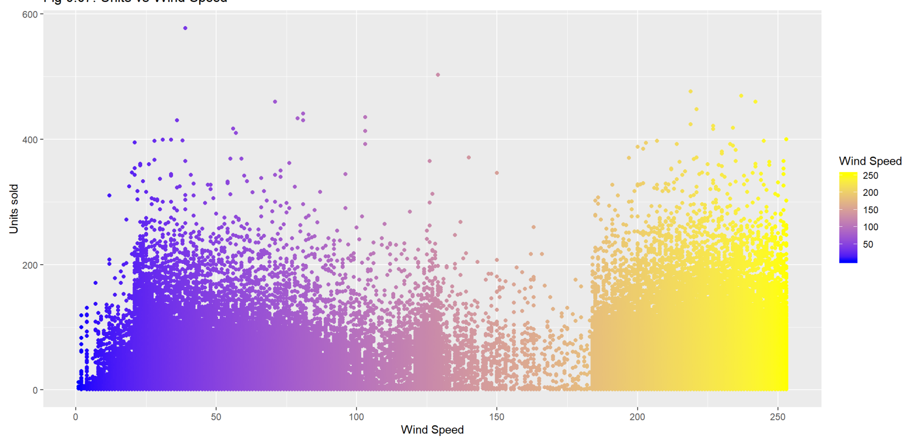
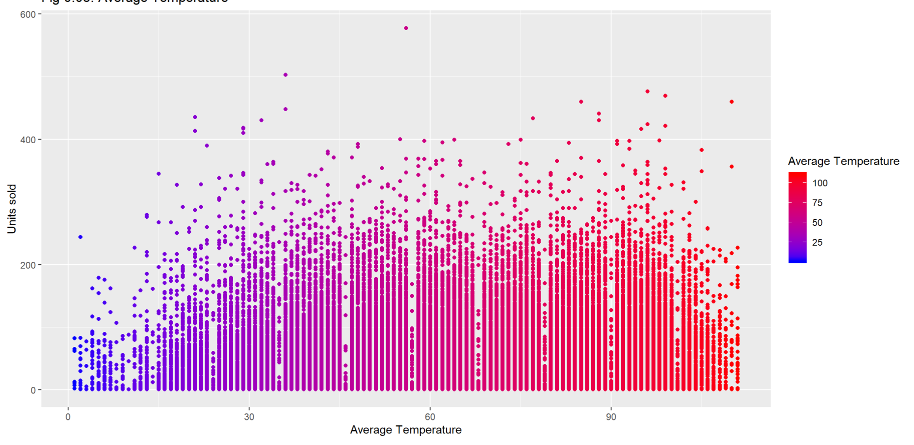


Fig 0.08: Average Temperature


Units sold
Average Temperature