Regression on Walmart Sales Data - Notebook 1 Regression

Alper Duru

03/25/2023

Imported data from: Walmart Sales Data

https://www.kaggle.com/datasets/ujjwalchowdhury/walmartcleaned?datasetId=2169207&language=R

GitHub Page

https://alper-enes-duru.github.io/Machine_Learning_Portfolio/

Read the sales data

This R code reads a cleansed Walmart sales data set from a CSV file using the read.csv() function and assigns it to a variable named sales.

Next, use the sample() function to randomly sample 10,000 cases from the sales data set. This random sample is stored in a new variable called mysample. The sample() function uses 1:nrow(sales) to randomly select a set of row indices from 1 to the number of rows in sales. The second parameter 10000 specifies the sample size and the replace=FALSE parameter ensures that each observation is selected only once.

The set.seed() function is used to set a random seed (1234 in this case) to ensure that the same random sample can be reproduced later.

Use the sample() function again to randomly select 80% of the rows from mysample as the training set. The sample() function generates a random set of row indices from 1 to 10,000 using 1:10,000. The second argument, 10000*.80, specifies the sample size (that is, 80% of 10,000). The replace=FALSE parameter ensures that each observation is selected only once. The resulting row index is stored in a new variable called i. The variable train is created by selecting the line of mysample indexed by i. In other words, train contains 8,000 randomly selected observations from mysample to train the model.

Create a test variable by selecting rows in mysample not indexed by i. In other words, test contains the remaining 2,000 observations of mysample that will be used to evaluate model performance.

```
sales <- read.csv('walmart_cleaned.csv')

mysample <- sales[sample(1:nrow(sales), 10000,
   replace=FALSE),]

set.seed(1234)
i <- sample(1:10000, 10000*.80, replace=FALSE)
train <- mysample[i,]
test <- mysample[-i,]
```

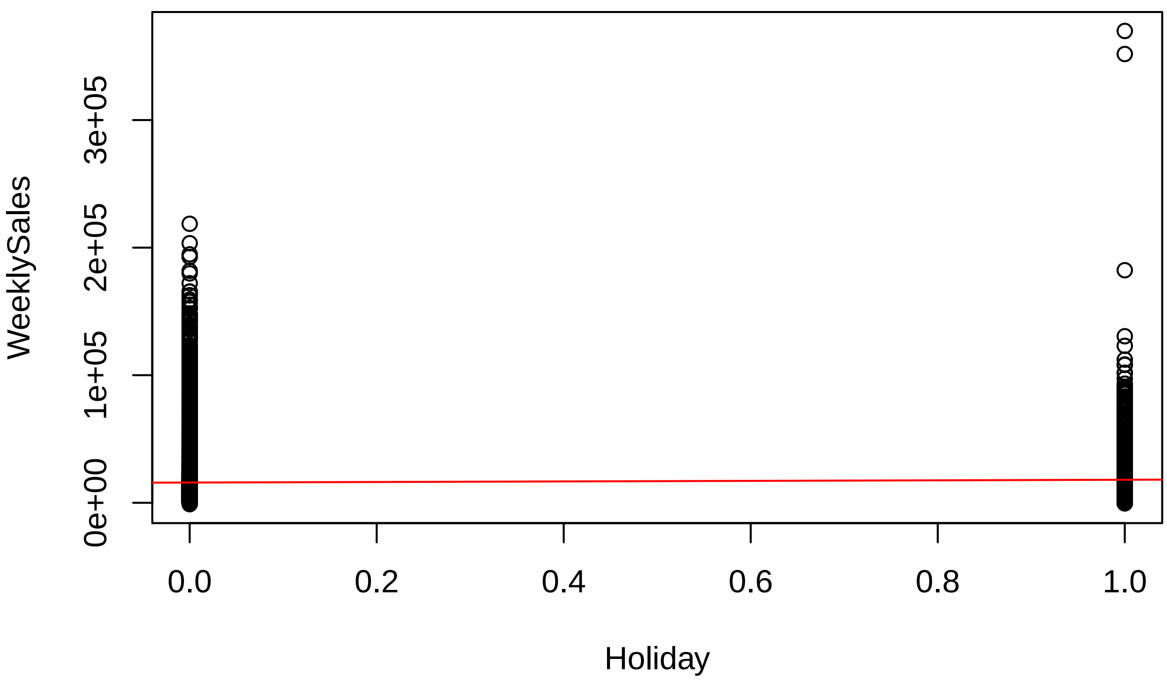
Plot

First, let's look at the data and its structure:

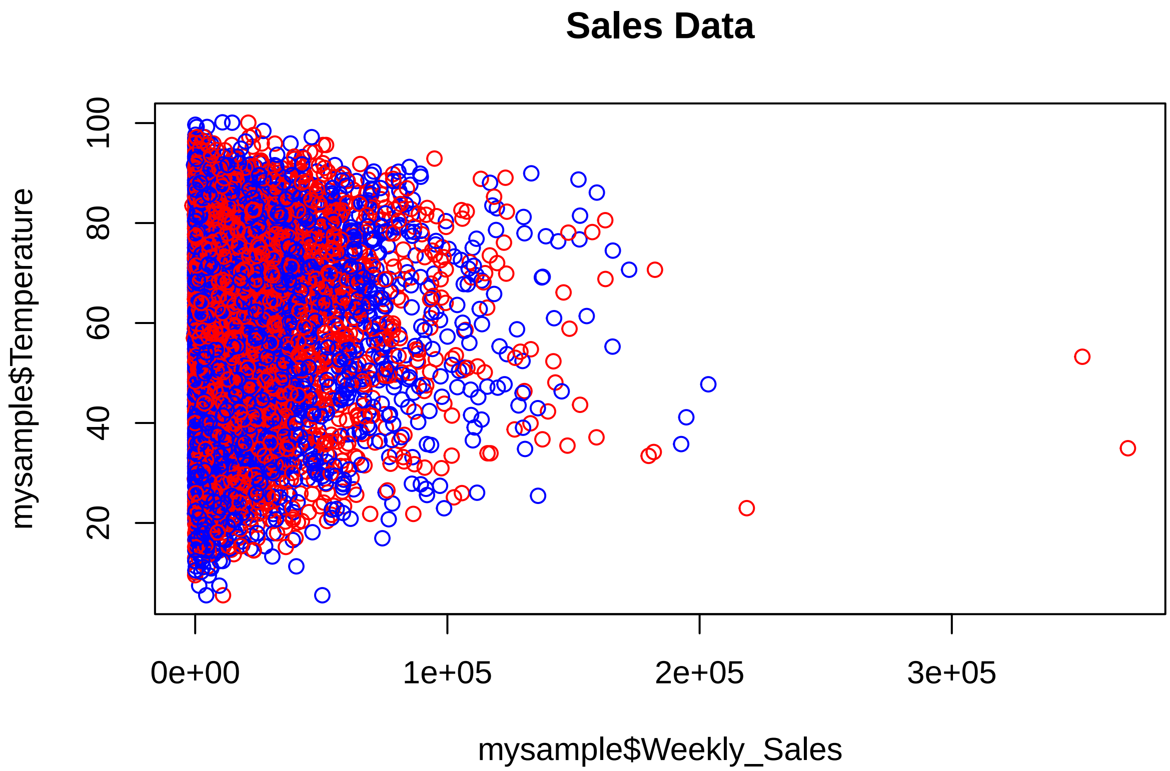
```
str(mysample)
```

```
## 'data.frame':    10000 obs. of  17 variables:
##  $ X           : int  298899 264932 75556 335330 342242 190222 64884 6221 25068 420722 ...
##  $ Store       : int  31 27 8 35 36 20 7 1 3 45 ...
##  $ Date        : chr  "2011-03-04" "2012-01-06" "2011-11-25" "2011-01-07" ...
##  $ IsHoliday   : int  0 0 1 0 0 0 1 0 0 1 ...
##  $ Dept        : num  44 47 41 95 26 49 54 94 81 71 ...
##  $ Weekly_Sales: num  2010 14 1280 30872 139 ...
##  $ Temperature : num  59.5 35.8 49.6 34.4 77.3 ...
##  $ Fuel_Price  : num  3.29 3.58 3.24 3.19 2.77 ...
##  $ MarkDown1   : num  0 9236 438 0 0 ...
##  $ MarkDown2   : num  0 29675 321 0 0 ...
##  $ MarkDown3   : num  0 526 45969 0 0 ...
##  $ MarkDown4   : num  0 2399 110 0 0 ...
##  $ MarkDown5   : num  0 6291 993 0 0 ...
##  $ CPI         : num  213 141 222 137 209 ...
##  $ Unemployment: num  8.03 8.01 6.12 8.55 8.46 ...
##  $ Type        : int  3 3 3 2 3 3 2 3 2 2 ...
##  $ Size        : int  203750 204184 155078 103681 39910 203742 70713 151315 37392 118221 ...
```

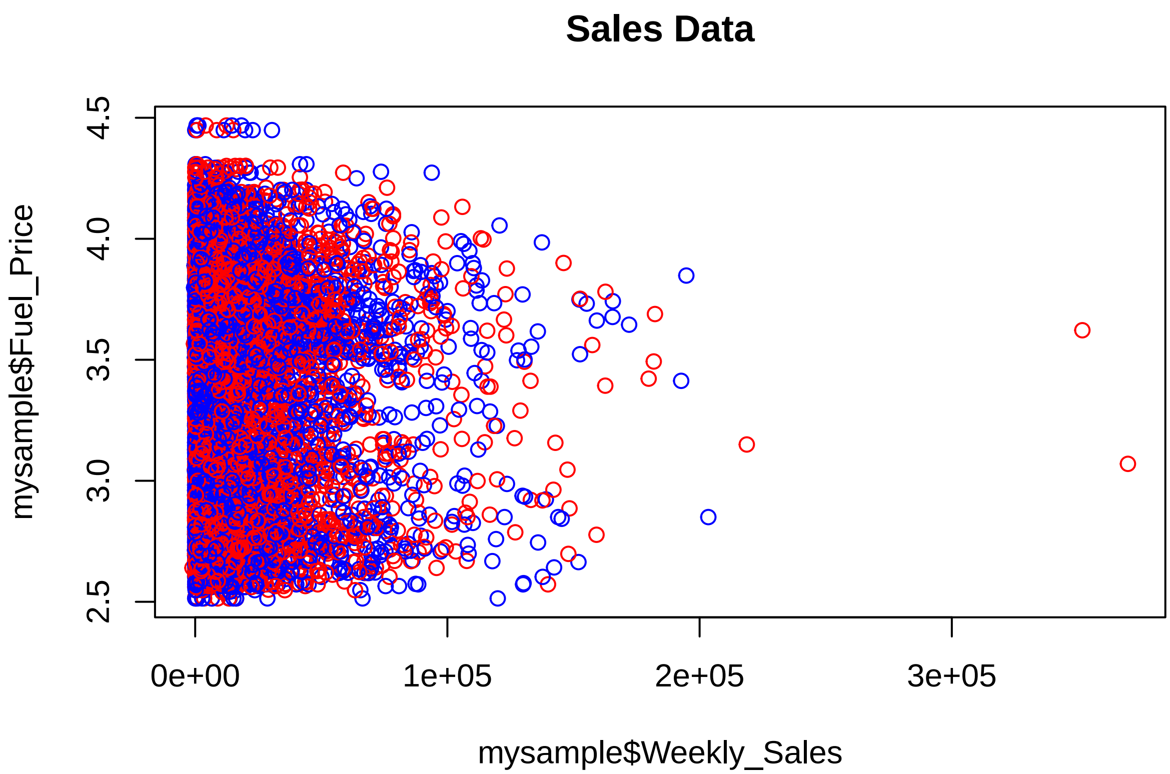
```
plot(mysample$Weekly_Sales~mysample$IsHoliday, xlab="Holiday", ylab="WeeklySales")
abline(lm(mysample$Weekly_Sales~mysample$IsHoliday), col="red")
```



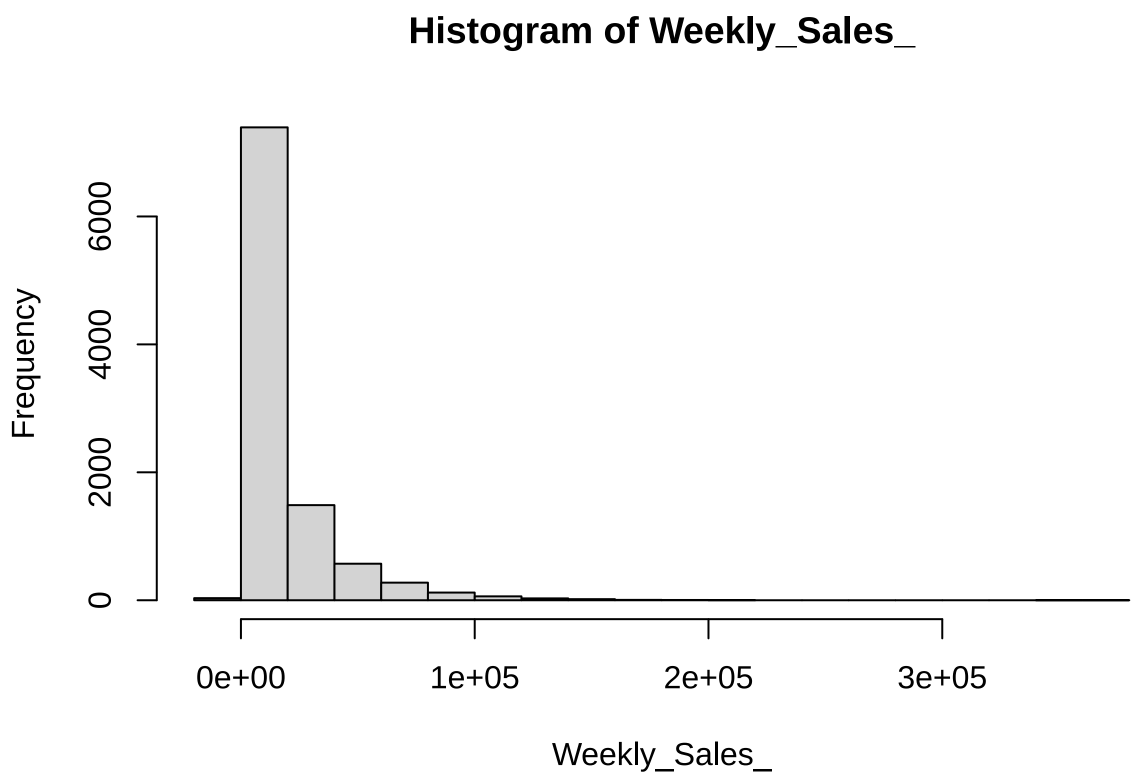

```
plot(mysample$Weekly_Sales,mysample$Temperature, col=c("blue","red"), main="Sales Data")
```



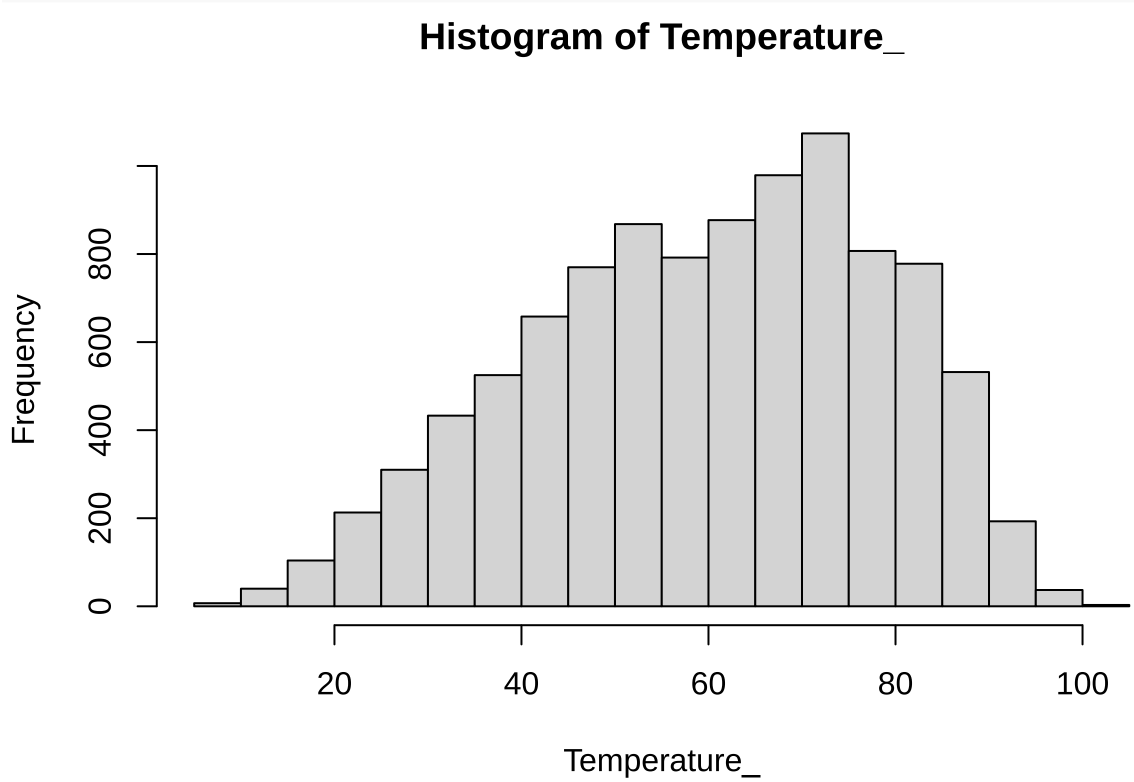

```r
plot(mysample$Weekly_Sales,mysample$Fuel_Price, col=c("blue","red"), main="Sales Data")
```



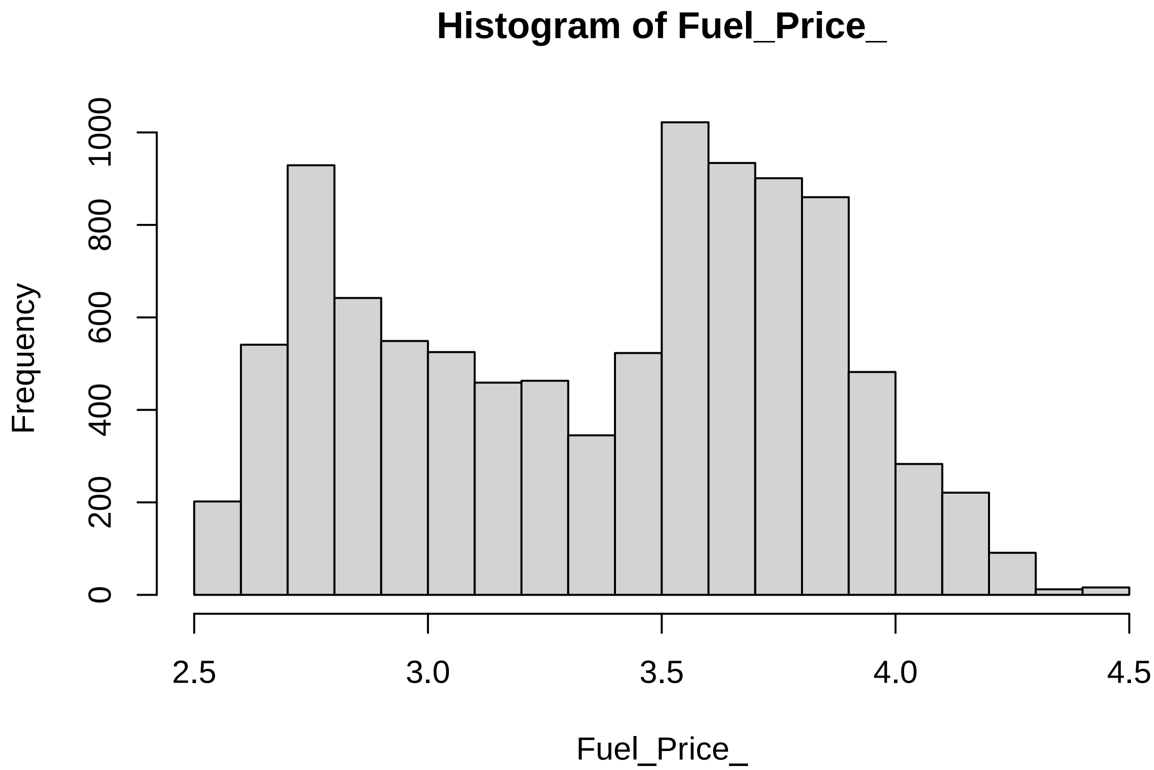

```r
Weekly_Sales_ <- (mysample$Weekly_Sales)
hist(Weekly_Sales_)
```



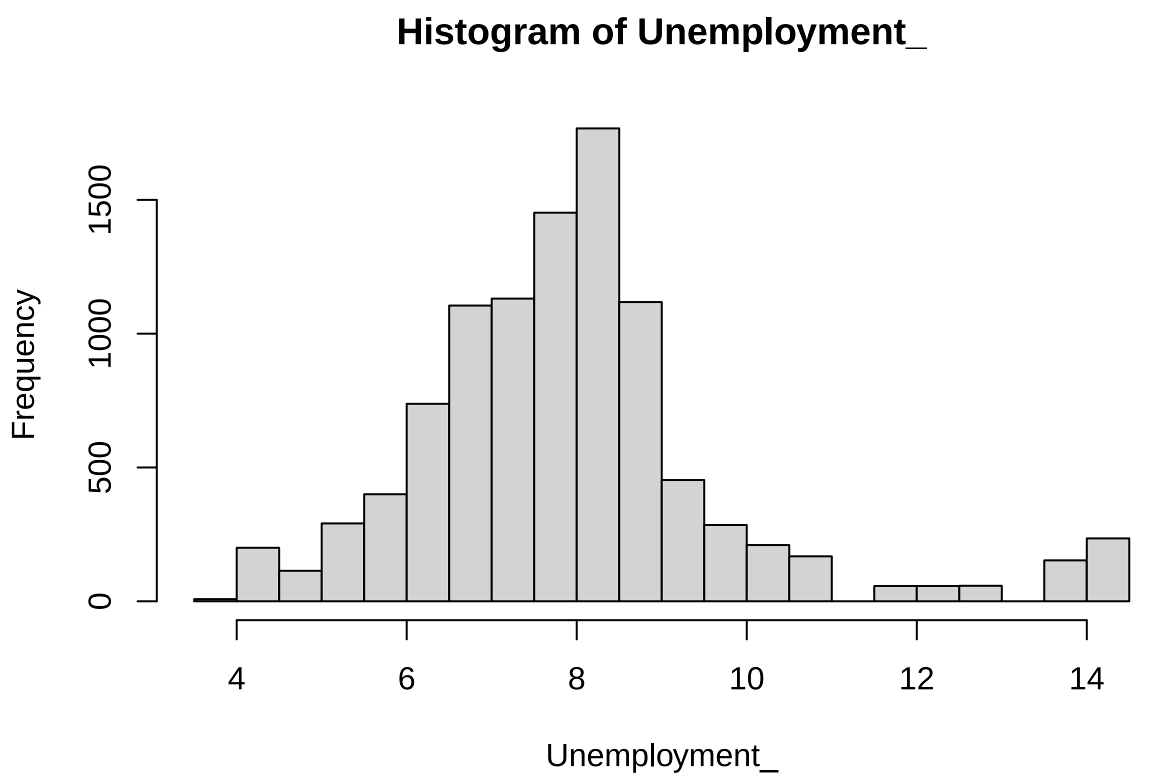

```
Temperature_ <- (mysample$Temperature)
hist(Temperature_)
```




```
Fuel_Price_ <- (mysample$Fuel_Price)
hist(Fuel_Price_)
```




```r
Unemployment_ <- (mysample$Unemployment)
hist(Unemployment_)
```



Correlation

We can check for correlation by creating a table with the cor() function or visually look for correlations by plotting pairs().

```
cor(mysample$Weekly_Sales, mysample$Temperature)
```

```
## [1] 0.004491171
```

```
cor(mysample$Weekly_Sales, mysample$IsHoliday)
```

```
## [1] 0.02461161
```

```
cor(mysample$Weekly_Sales, mysample$Fuel_Price)
```

```
## [1] -0.01077339
```

```
cor(mysample$Weekly_Sales, mysample$CPI)
```

```
## [1] -0.01755624
```

```
cor(mysample$Weekly_Sales, mysample$Unemployment)
```

```
## [1] -0.02518078
```

Linear Regression

The output shows the results of a linear regression analysis performed on the train dataset with the lm() function in R.

The formula used in the regression is Weekly_Sales ~ IsHoliday + Temperature + Fuel_Price + Unemployment + CPI, which means that Weekly_Sales is the dependent variable, and the independent variables are IsHoliday, Temperature, Fuel_Price, Unemployment, and CPI.

The Coefficients table displays the estimated regression coefficients for each independent variable along with their standard errors, t-values, and p-values. The Intercept coefficient represents the estimated mean value of Weekly_Sales when all independent variables are zero.

The results show that Unemployment and CPI have significant negative coefficients, indicating that higher levels of unemployment and CPI are associated with lower weekly sales. Temperature has a significant positive coefficient, indicating that higher temperatures are associated with higher weekly sales. The IsHoliday and Fuel_Price variables are not significant predictors of weekly sales as their p-values are greater than the threshold of 0.05.

The Residuals section provides summary statistics of the residuals, which represent the differences between the observed and predicted values of Weekly_Sales. The minimum, first quartile, median, third quartile, and maximum values of the residuals are provided.

The Residual standard error is the estimated standard deviation of the residuals, which represents the average amount by which the observed values deviate from the predicted values.

The Multiple R-squared value represents the proportion of variance in Weekly_Sales that is explained by the independent variables in the model. The Adjusted R-squared value adjusts for the number of independent variables in the model. In this case, the model explains only a small proportion of the variance in the data.

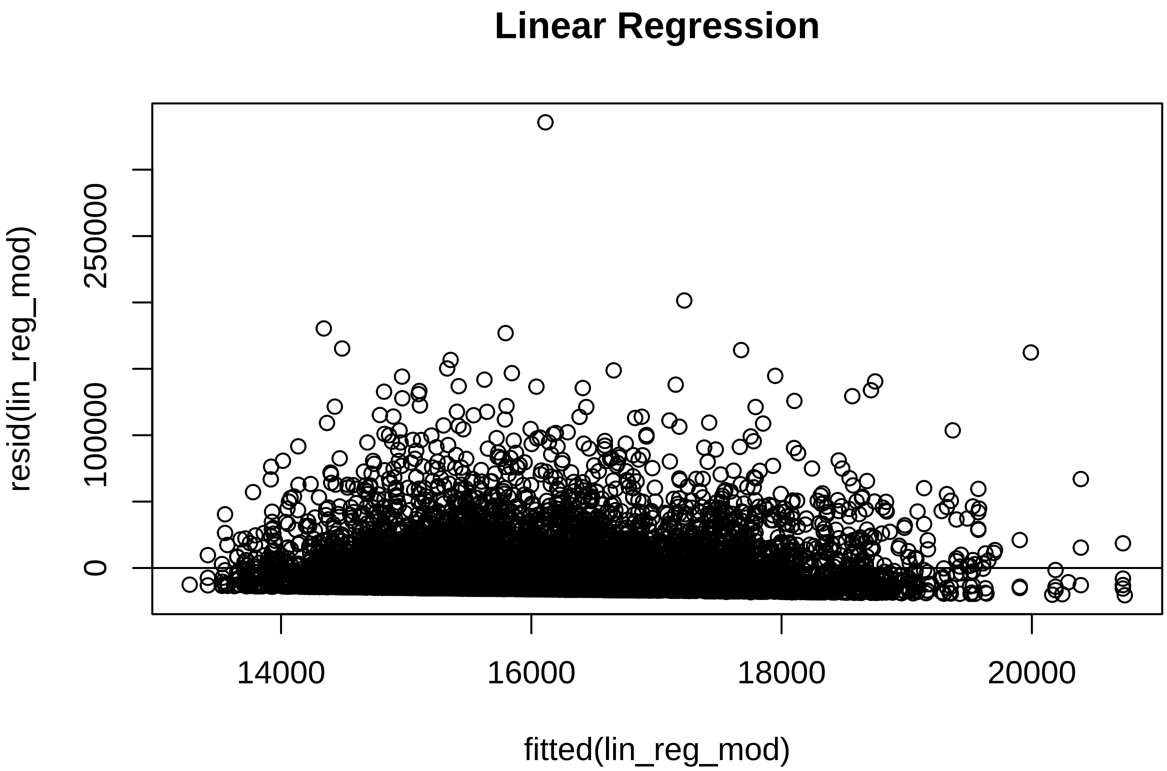
The F-statistic tests the overall significance of the model, and its associated p-value is very small, indicating that the model is significant at the 0.05 level.

The last three lines of output show the correlation, mse (mean squared error), and rmse (root mean squared error) of the model. The correlation value is very small, indicating a weak correlation between the independent variables and Weekly_Sales. The mse and rmse values represent the average squared and root mean squared difference between the observed and predicted values of Weekly_Sales, respectively. The rmse value of approximately 22661 indicates that the model's predictions have an average error of approximately $22,661 in predicting weekly sales.

```
lin_reg_mod <- lm(Weekly_Sales~IsHoliday+Temperature+Fuel_Price+Unemployment+CPI, data=train)
summary(lin_reg_mod)
```

```
##
## Call:
## lm(formula = Weekly_Sales ~ IsHoliday + Temperature + Fuel_Price +
##     Unemployment + CPI, data = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -20531 -13682  -8368   4752 335651
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  25082.789   2840.063   8.832  < 2e-16 ***
## IsHoliday     1868.846    996.041   1.876 0.060654 .
## Temperature     32.330     14.592   2.216 0.026742 *
## Fuel_Price    -851.969    569.773  -1.495 0.134882
## Unemployment  -477.428    145.265  -3.287 0.001018 **
## CPI            -25.574      7.062  -3.621 0.000295 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 22490 on 7994 degrees of freedom
## Multiple R-squared:  0.002742,   Adjusted R-squared:  0.002119
## F-statistic: 4.397 on 5 and 7994 DF,  p-value: 0.0005336
```

```
plot(fitted(lin_reg_mod), resid(lin_reg_mod), main = "Linear Regression")
abline(0,0)
```




```
pred <- predict(lin_reg_mod, newdata=test)
cor <- cor(pred, test$Weekly_Sales)
mse <- mean((pred-test$Weekly_Sales)^2)
rmse <- sqrt(mse)
print(paste('correlation:', cor))
```

```
## [1] "correlation: 0.038701636358345"
```

```
print(paste('mse:', mse))
```

```
## [1] "mse: 523413224.794975"
```

```
print(paste('rmse:', rmse))
```

```
## [1] "rmse: 22878.2259975501"
```

kNN Regression

The output provides an evaluation of two models built using the K-Nearest Neighbors (KNN) algorithm. One model is built using unscaled data, and the other is built using scaled data.

For the scaled data model, the correlation between the predicted values and the actual values is 0.990, indicating a strong positive linear relationship between the two variables. The mean squared error (MSE) of the model is 13,983,015, and the root mean squared error (RMSE) is 3,739. The scaled model seems to perform well, given the high correlation and low RMSE.

For the unscaled data model, the correlation between the predicted values and the actual values is -0.137, indicating a weak negative linear relationship between the two variables. The MSE of the model is 749,324,027, which is much higher than the scaled model, and it indicates that the model is not very accurate in predicting the values.

Overall, it seems that the scaled KNN model is a better choice for this dataset since it has higher accuracy and lower errors in comparison to the unscaled KNN model.

```r
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```r
library(lattice)
library(ggplot2)
scaled_training_data <- train[, c(4:7,13:14)]
means <- sapply(scaled_training_data, mean)
stdvs <- sapply(scaled_training_data, sd)
scaled_training_data <- scale(scaled_training_data, center=means, scale=stdvs)
scaled_test_data <- scale(test[, c(4:7,13:14)], center=means, scale=stdvs)


# Scaled Data
fit <- knnreg(scaled_training_data, train$Weekly_Sales, k=10)
pred2 <- predict(fit, scaled_test_data)
cor_knn2 <- cor(pred2, test$Weekly_Sales)
mse_knn2 <- mean((pred2 - test$Weekly_Sales)^2)
print(paste("scaled cor=", cor_knn2))
```

```
## [1] "scaled cor= 0.982162537115978"
```

```r
print(paste("scaled mse=", mse_knn2))
```

```
## [1] "scaled mse= 23179076.0765356"
```

```r
print(paste("scaled rmse=", sqrt(mse_knn2)))
```

```
## [1] "scaled rmse= 4814.46529497676"
```

```r
# Unscaled Data
fit <- knnreg(train[,c(4:7,13:14)],train[,14],k=50)

pred <- predict(fit, test[,c(4:7,13:14)])
cor_knn1 <- cor(pred, test$Weekly_Sales)
mse_knn1 <- mean((pred - test$Weekly_Sales)^2)
print(paste("cor=", cor_knn1))
```

```
## [1] "cor= -0.0667063761575"
```

```r
print(paste("mse=", mse_knn1))
```

```
## [1] "mse= 762337702.181108"
```

Decision Tree

The output displays the results of a regression tree model applied to a dataset.

The model is constructed using the tree function from the tree package in R. The formula for the model is Weekly_Sales ~ ., meaning that the dependent variable Weekly_Sales is modeled as a function of all other variables in the dataset. The variables that are actually used in the tree construction are Dept, Size, and Type. The tree has 17 terminal nodes and a residual mean deviance of 175900000.

The predict function is used to predict the values of Weekly_Sales for the test dataset. The correlation between the predicted values and the actual test dataset values is computed using the cor function. The correlation value is 0.750968675714858, indicating a moderate positive correlation between the predicted and actual values. The root mean square error (RMSE) of the predicted values is also computed using the sqrt and mean functions. The RMSE value is 14951.781721625.

The plot function is used to visualize the regression tree model. The cv.tree function is used to perform cross-validation on the model. The resulting values are then plotted using the plot function, showing the relationship between the size of the tree and the cross-validation error.

The output also shows a warning message regarding NAs introduced by coercion, indicating that some data may be missing or that there may be issues with the data type.

```r
library(tree)
library(MASS)
tree1 <- tree(Weekly_Sales~., data = train)
```

```
## Warning in tree(Weekly_Sales ~ ., data = train): NAs introduced by coercion
```

```r
summary(tree1)
```

```
##
## Regression tree:
## tree(formula = Weekly_Sales ~ ., data = train)
## Variables actually used in tree construction:
## [1] "Dept"      "Size"      "MarkDown3"
## Number of terminal nodes:  13
## Residual mean deviance:  224700000 = 1.794e+12 / 7987
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -78260   -6211   -2837       0    4104  217700
```

```r
pred <- predict(tree1, newdata=test)
```

```
## Warning in pred1.tree(object, tree.matrix(newdata)): NAs introduced by coercion
```

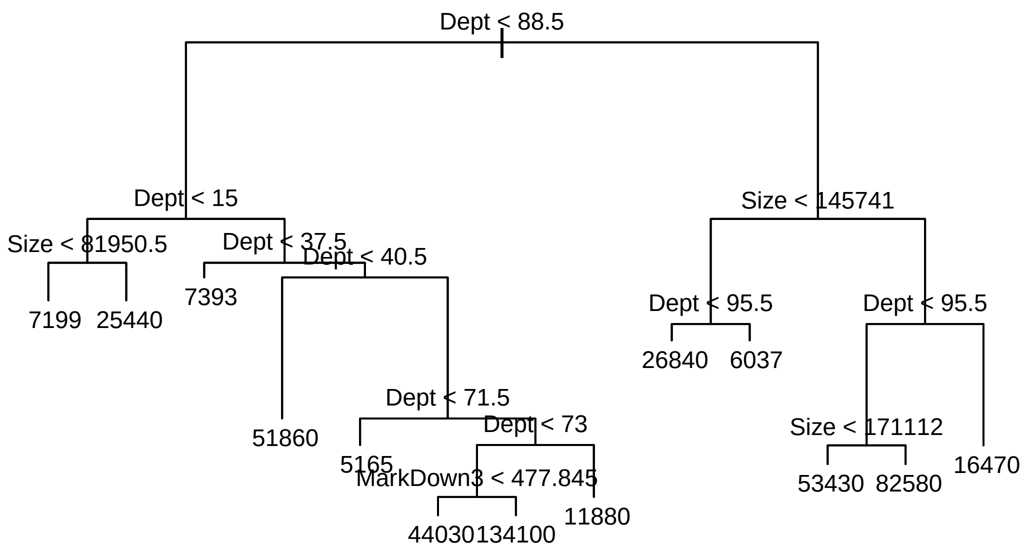
```r
print(paste('correlation:', cor(pred, test$Weekly_Sales)))
```

```
## [1] "correlation: 0.723953945497785"
```

```r
rmse_tree <- sqrt(mean((pred-test$Weekly_Sales)^2))
print(paste('rmse:', rmse_tree))
```

```
## [1] "rmse: 15794.9651371189"
```

```r
plot(tree1, main = "Tree")
text(tree1, cex=0.75, pretty=0)
```




```r
cv_tree <- cv.tree(tree1)
```

```
## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion
```

```
## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion
```

```
plot(cv_tree$size, cv_tree$dev, type='b', main = "Validation")
```