Final Exercise

Gandhar Kothari

2017-12-28

```
include = F
# install.packages("readxl")
# install.packages("forecast")
# install.packages("tseries")
# install.packages("Metrics")
# install.packages("vars")
# install.packages("tsDyn")
# install.packages("DAAG")
# install.packages("lubridate")
# install.packages("caret")
# install.packages("randomForest")
# install.packages("xlsx")
suppressWarnings(library("readxl",verbose =F))
suppressWarnings(library("forecast",verbose =F))
suppressWarnings(library("tseries",verbose =F))
suppressWarnings(library("Metrics",verbose =F))
```

```
##
## Attaching package: 'Metrics'
```

```
## The following object is masked from 'package:forecast':
##
##     accuracy
```

```
suppressWarnings(library("vars",verbose =F))
```

```
## Loading required package: MASS
```

```
## Loading required package: strucchange
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: sandwich
```

```
## Loading required package: urca
```

```
## Loading required package: lmtest
```

```
suppressWarnings(library("lubridate",verbose =F))
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```r
suppressWarnings(library("caret",verbose =F))
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```r
suppressWarnings(library("tsDyn",verbose =F))
```

```
##
## Attaching package: 'tsDyn'
```

```
## The following object is masked from 'package:Metrics':
##
##     mse
```

```r
#library("DAAG")
suppressWarnings(library("randomForest",verbose =F))
```

```
## randomForest 4.6-12
```

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

```
##
## Attaching package: 'randomForest'
```

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

```r
suppressWarnings(library("xlsx",verbose =F))
```

```
## Loading required package: rJava
```

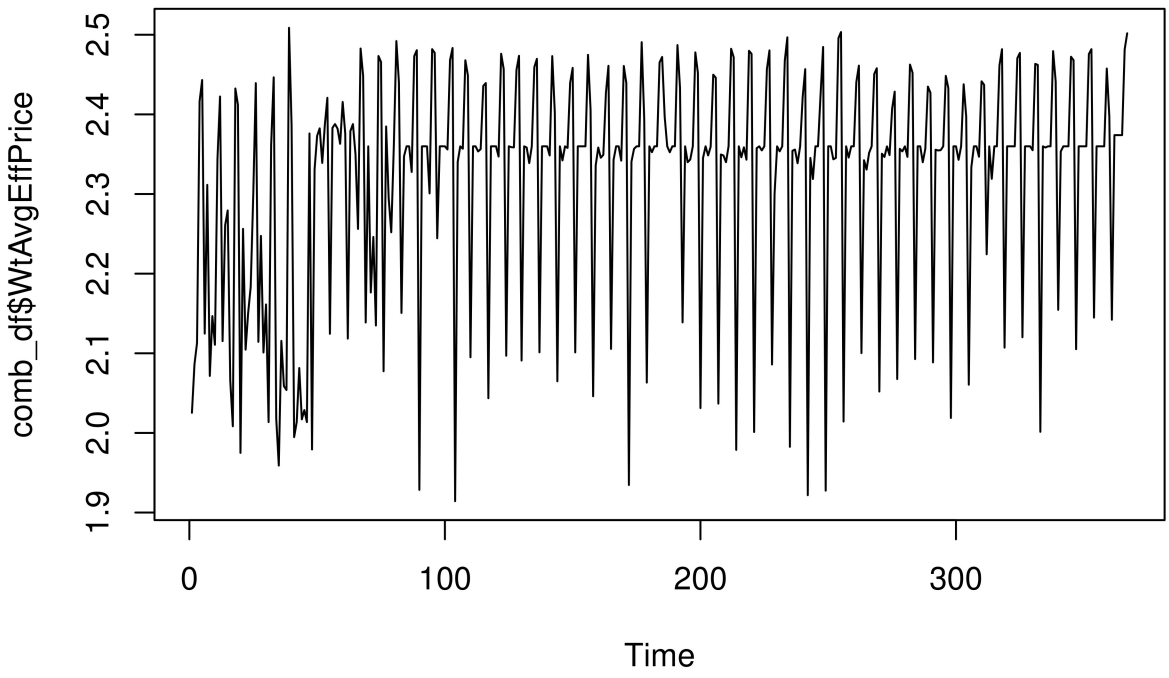
```
## Loading required package: xlsxjars
```

```r
data <- read_xlsx("Test.xlsx")
data$DayOfWeekNumber <- as.factor(data$DayOfWeekNumber)
```

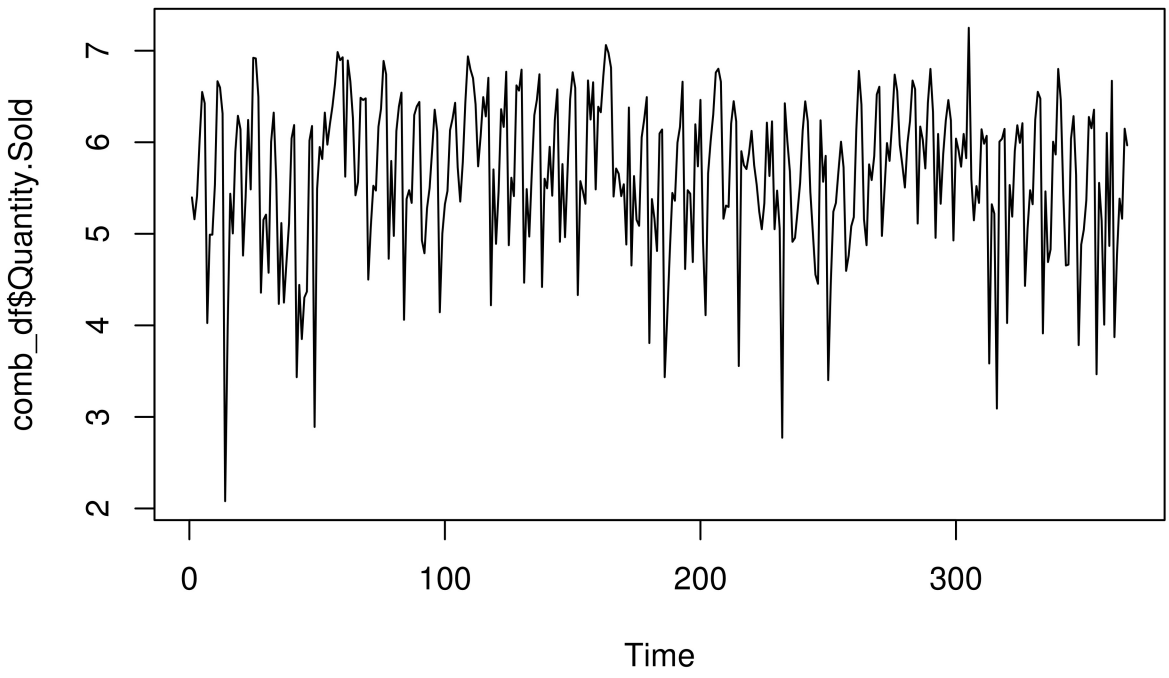
```r
# Data preprocessing

# tranforming data and converting to time series object to apply time series model
combine<- log(data[,5:8])
comb_ts <- ts(combine, start = c(2016,11), frequency = 7)
comb_df <- data.frame(comb_ts)

# plotting time series graphs
plot.ts(comb_df$WtAvgEffPrice)
```




```
plot.ts(comb_df$Quantity.Sold)
```




```
# checking if data is stationary for time series
adf.test(comb_df$WtAvgEffPrice)
```

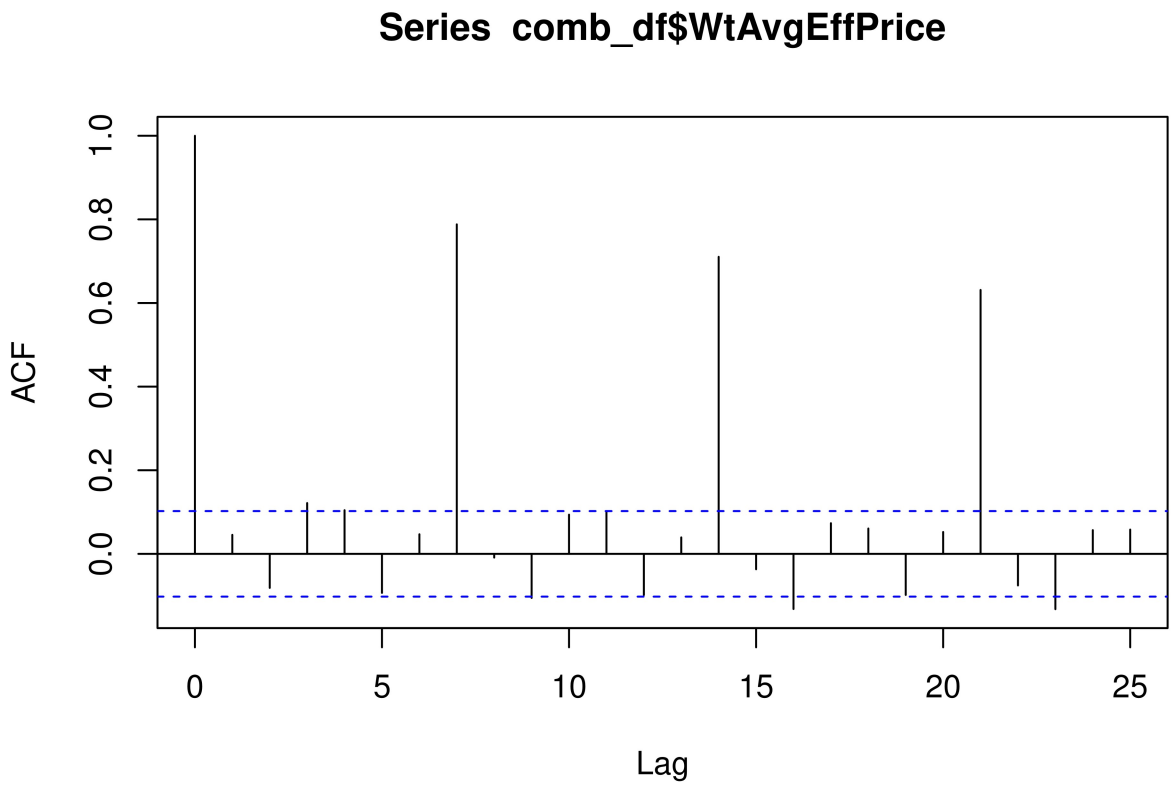
```
##
##  Augmented Dickey-Fuller Test
##
## data:  comb_df$WtAvgEffPrice
## Dickey-Fuller = -3.0499, Lag order = 7, p-value = 0.1338
## alternative hypothesis: stationary
```

```
adf.test(comb_df$Quantity.Sold)
```

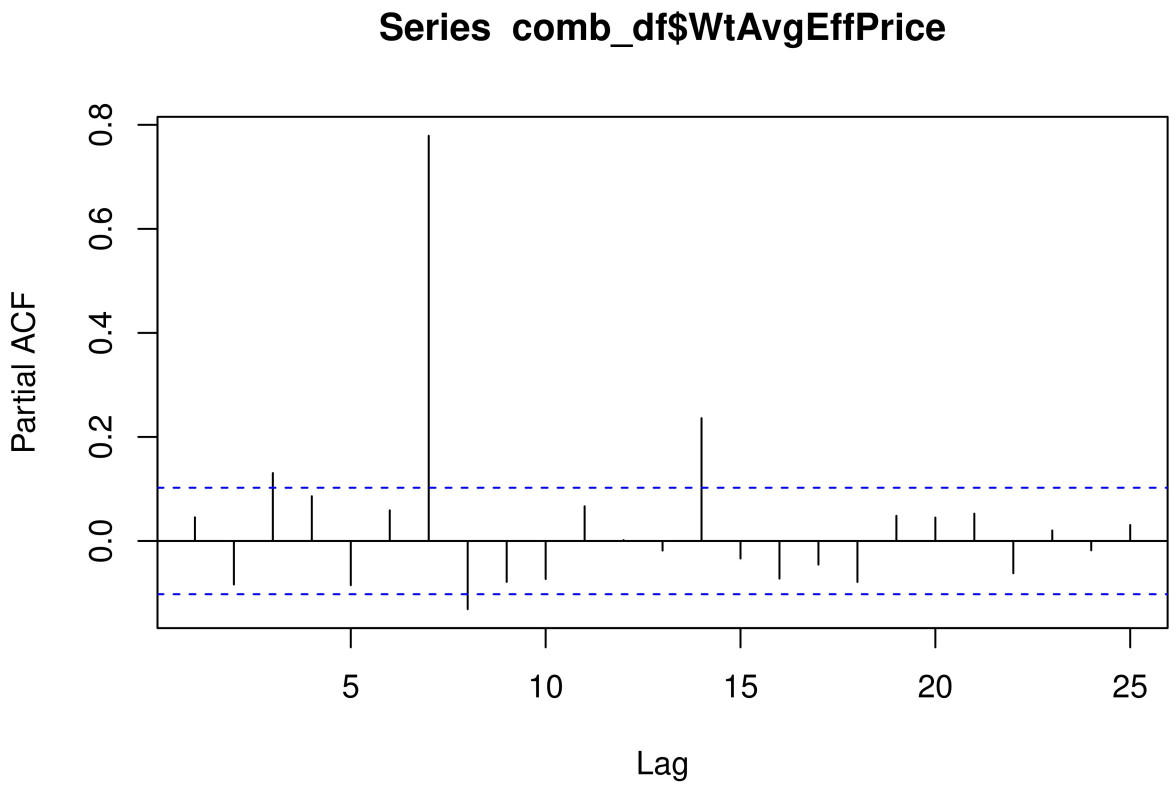
```
## Warning in adf.test(comb_df$Quantity.Sold): p-value smaller than printed p-
## value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  comb_df$Quantity.Sold
## Dickey-Fuller = -4.1194, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

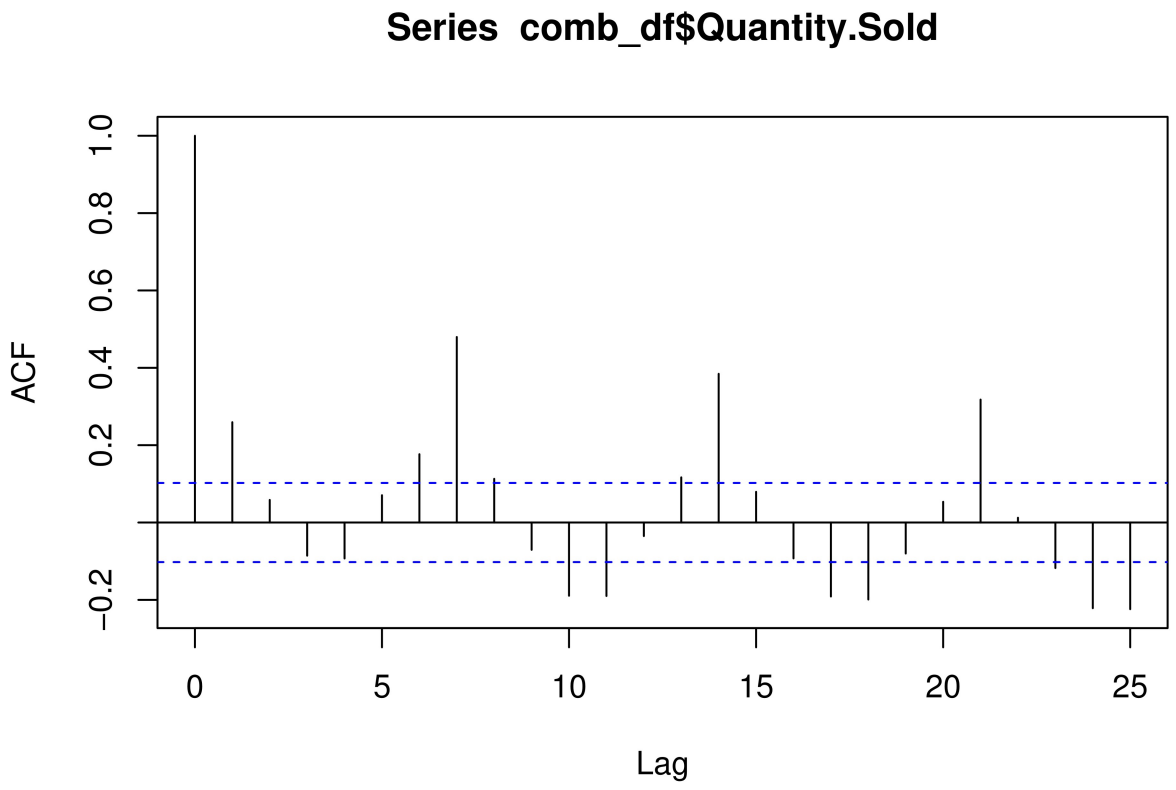
```
# plotting acf and pacf plot to check autocorrelation
acf(comb_df$WtAvgEffPrice)
```



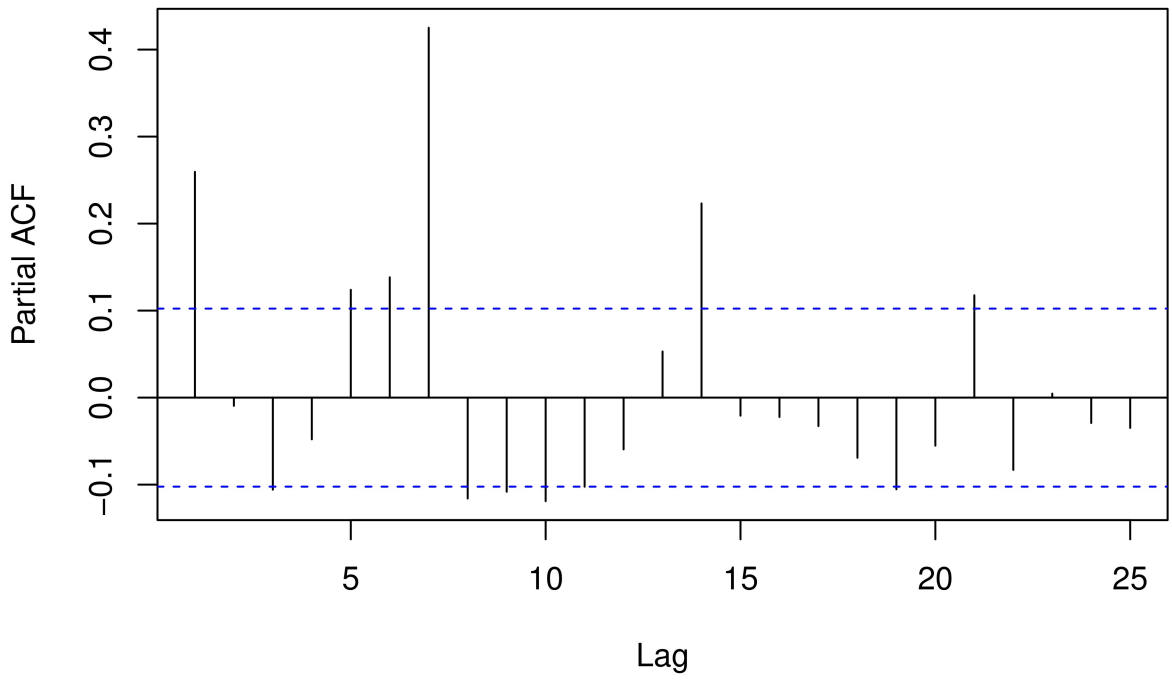

```
pacf(comb_df$WtAvgEffPrice)
```




```
acf(comb_df$Quantity.Sold)
```




```
pacf(comb_df$Quantity.Sold)
```




```
# It is clearly seen from plots that there is high correlation on lag 7, 14, 21 so on hence
# this data has weekly seasonality which will be considered while building time series model

# Partitioning data into train and test
train_comb <- comb_df[1:330,]
test_comb <- comb_df[331:367,]
train_data <- data[1:330,]
test_data <- data[331:367,]
```

```
# VAR model to forecast test data
var1 <- VAR(train_comb,p = 2,type = "both", lag.max = 7, season = 7)

#summary(var1)
pred <- predict(var1, n.ahead = 37)
pred_df <- as.data.frame(pred$fcst)
pred_actual <- exp(pred_df[,c(1,5,9,13)])

# Checking accuracy of VAR model on test data
forecast::accuracy(pred_actual$Quantity.Sold.fcst, test_data$`Quantity Sold`)
```

```
##                 ME     RMSE      MAE       MPE     MAPE
## Test set 3.374091 159.9765 114.646 -54.19023 73.15708
```

```
forecast::accuracy(pred_actual$EffectivePriceRevenue.fcst, test_data$EffectivePriceRevenue)
```

```
##                  ME     RMSE      MAE       MPE     MAPE
## Test set -64.55754 1597.487 1146.837 -57.77541 75.00759
```

```
forecast::accuracy(pred_actual$DailyShoeRental.fcst, test_data$DailyShoeRental)
```

```
##                  ME    RMSE      MAE       MPE     MAPE
## Test set -7.088302 93.6765 64.42428 -69.39631 83.96972
```

```
forecast::accuracy(pred_actual$WtAvgEffPrice.fcst, test_data$WtAvgEffPrice)
```

```
##                   ME      RMSE       MAE       MPE     MAPE
## Test set -0.192544 0.3779335 0.3030169 -2.010035 2.968956
```

```
# VECM model to forecast test data
VECmodel <-  VECM(train_comb,lag = 7, r = 3,include = "const",estim="ML")
```

```
## Warning in x/sqrt(t(x) %*% S11 %*% x): Recycling array of length 1 in vector-array arithmetic is depr
##   Use c() or as.vector() instead.

## Warning in x/sqrt(t(x) %*% S11 %*% x): Recycling array of length 1 in vector-array arithmetic is depr
##   Use c() or as.vector() instead.

## Warning in x/sqrt(t(x) %*% S11 %*% x): Recycling array of length 1 in vector-array arithmetic is depr
##   Use c() or as.vector() instead.

## Warning in x/sqrt(t(x) %*% S11 %*% x): Recycling array of length 1 in vector-array arithmetic is depr
##   Use c() or as.vector() instead.
```

```
#summary(VECmodel)
pred <- predict(VECmodel, n.ahead = 37)
pred_df <- as.data.frame(pred)
pred_actual <- exp(pred_df[,c(1:4)])
forecast::accuracy(pred_actual$Quantity.Sold, test_data$`Quantity Sold`)
```

```
##                  ME     RMSE      MAE       MPE     MAPE
## Test set -28.78852 201.5138 161.4791 -90.62859 110.0625
```

```
forecast::accuracy(pred_actual$EffectivePriceRevenue, test_data$EffectivePriceRevenue)
```

```
##                  ME     RMSE      MAE      MPE    MAPE
## Test set -373.3068 1980.564 1665.873 -91.7722 111.582
```

```
forecast::accuracy(pred_actual$DailyShoeRental, test_data$DailyShoeRental)
```

```
##                  ME     RMSE      MAE       MPE     MAPE
## Test set -15.26191 116.4435 90.84297 -99.07085 118.1397
```

```
forecast::accuracy(pred_actual$WtAvgEffPrice, test_data$WtAvgEffPrice)
```

```
##                    ME      RMSE       MAE        MPE     MAPE
## Test set -0.02186242 0.5411617 0.4131236 -0.7299947 4.083126
```

```
# From both model results it is seen that accuarcy of VAR model is better. I have also tried other
# time series models like ARIMA, ETS and Holtwinters (code not included for those models). But VAR
# model provided better results among all. Hence I have selected VAR model to forecast data for next
# 365 days.
```

```
# 1. Forecasting data for next 365 days using VAR model
combine_df <- ts(comb_ts, start = c(2016,11), frequency = 7)
var2 <- VAR(combine_df,type = "const", lag.max = 7, season = 7)
#summary(var2)
prediction <- predict(var2, n.ahead = 365)
```

```
prediction_df <- as.data.frame(prediction$fcst)
predicted_value <- exp(prediction_df[,c(1,5,9,13)])
predicted_value$Quantity.Sold.fcst <- round(predicted_value$Quantity.Sold.fcst)
predicted_value$DailyShoeRental.fcst <- round(predicted_value$DailyShoeRental.fcst)
#range(data$EffectivePriceRevenue)
dates <- seq(as.Date("2017-11-05"), as.Date("2018-11-04"), by = "day")
date_df <- data.frame(dates)
date_df$dayofweeknumber <- wday(date_df$dates)
forecasting_VAR <- cbind(date_df,predicted_value)
colnames(forecasting_VAR)[3:6] <- c("Quantity Sold", "Net Revenue", "Net Rate", "Daily Shoe Rental")
head(forecasting_VAR, 5)
```

```
##        dates dayofweeknumber Quantity Sold Net Revenue  Net Rate
## 1 2017-11-05               1           541    4611.983  8.524904
## 2 2017-11-06               2           131    1376.223 10.527098
## 3 2017-11-07               3           203    2161.375 10.626912
## 4 2017-11-08               4           200    2124.819 10.603008
## 5 2017-11-09               5           186    1987.035 10.657059
##   Daily Shoe Rental
## 1               297
## 2                70
## 3               100
## 4               100
## 5                96
```

```
# 2. Optimizing net rate an net revenue using linear algorithm
l1 <- lm(forecasting_VAR$`Net Rate` ~ forecasting_VAR$`Quantity Sold`, data = forecasting_VAR)
l1$coefficients
```

```
##                     (Intercept) forecasting_VAR$`Quantity Sold`
##                    10.856296524                     -0.001513362
```

```
ctrl <- trainControl(method = "cv", number = 10 )
lmcv <- train(`Net Rate` ~ `Quantity Sold`, data = forecasting_VAR, method = 'lm', trControl = ctrl,
              metric= 'Rsquared')
# summary(lmcv)
Optimized_rate <- NULL
Optimized_rate_all <- NULL
Optimized_revenue <- NULL
Optimized_revenue_all <- NULL
for(i in 1:nrow(forecasting_VAR)) {
Optimized_rate <-  l1$coefficients[1] + l1$coefficients[2] * forecasting_VAR$`Quantity Sold`[i]
Optimized_rate_all <- as.data.frame(rbind(Optimized_rate_all, Optimized_rate))
colnames(Optimized_rate_all)[1] <- c("rates")
Optimized_revenue <- Optimized_rate_all$rates[i] * forecasting_VAR$`Quantity Sold`[i]
Optimized_revenue_all <- as.data.frame(rbind(Optimized_revenue_all, Optimized_revenue))
}
Optimization <- cbind(Optimized_rate_all$rates, Optimized_revenue_all)
rmse(Optimization$`Optimized_rate_all$rates`, forecasting_VAR$`Net Rate`)
```

```
## [1] 0.7946038
```

```
# Both normal lm and cv methods are giving same results. hence I have calculated net rate using
# intercept of quantity sold and constant of equation. I have also calculated revenue through net rate
# and quantity sold
```

```
# Exporting forecasted and optimized values to excel
# write.xlsx(forecasting_VAR, "Forecasted_values.xlsx")
# write.xlsx(Optimization, "Optimized_values.xlsx")
```