

Data Mining Final Project (FALL 2019)

Group Members:

1. Harsh Uday Dodia(hd262)
2. Sagar Lathiya (sl828)

Tasks to be completed:

1. Data Preparation and Exploratory analysis:

- **Merging Data:**

- Merging all the data items:

Merging all the data frames and rearrangement of columns.

The screenshot displays the RStudio environment with several open files and a console window. The 'Environment' pane on the right lists data objects: mydata1 (2935849 obs. of 6 variables), mydata2 (22170 obs. of 3 variables), mydata3 (84 obs. of 2 variables), mydata4 (60 obs. of 2 variables), mydata5 (1048575 obs. of 4 variables), mytempdata (2935849 obs. of 10 variables), and mytempdata1 (2935849 obs. of 7 variables). The 'Files' pane shows the project structure. The 'Console' window contains the following R code:

```
setwd("E:/Docs/Fall 2019/Data Mining/Assignments/Project")
# Loading the given data
mydata1 = read.csv("sales_train_v2.csv", header=T, sep=",")
mydata2 = read.csv("items.csv", header=T, sep=",")
mydata3 = read.csv("item_categories.csv", header=T, sep=",")
mydata4 = read.csv("shops.csv", header=T, sep=",")
mydata5 = read.csv("sales.csv", header=T, sep=",")
# Merging the data
mytempdata = merge(mydata1, mydata2)
mytempdata = merge(mytempdata, mydata3)
mytempdata = merge(mytempdata, mydata4)
# Arranging the merged data
colnames(mytempdata) = c("shop_id", "item_category_id", "item_id", "date", "date_block_num", "item_price")
mytempdata1 = mytempdata[, c(4, 5, 3, 2, 7, 6, 1)]
View(mytempdata1)
View(mytempdata1)
```

- **Cleaning Data:**

- **Look for any missing data:**

- Identify observation on missing data:**

While checking for the missing values, we found out that there are no missing values in general like NULL or NA values. The output was FALSE.

```
> any(is.na(mytempdata1))
[1] FALSE
> sum(is.na(mytempdata1))
[1] 0
```

- **Graph the representation of missing data:**

As we found that there is no missing data and occurrence of negative values like -1 is seen the dataset. So we considered negative values as missing values.

- **Decide whether to populate or remove missing data:**

Since the missing values as very small as compared to the entire dataset they don't affect the dataset as whole. So removal or population of the missing values won't make any difference.

- **Identify the possible of impact of it while modelling:**

As mentioned above since the missing values as less as compared to the entire dataset it has negligible or no impact while modelling.

- **Removed categorial attributes that were not required (item_name, item_category_name, shop_name).**

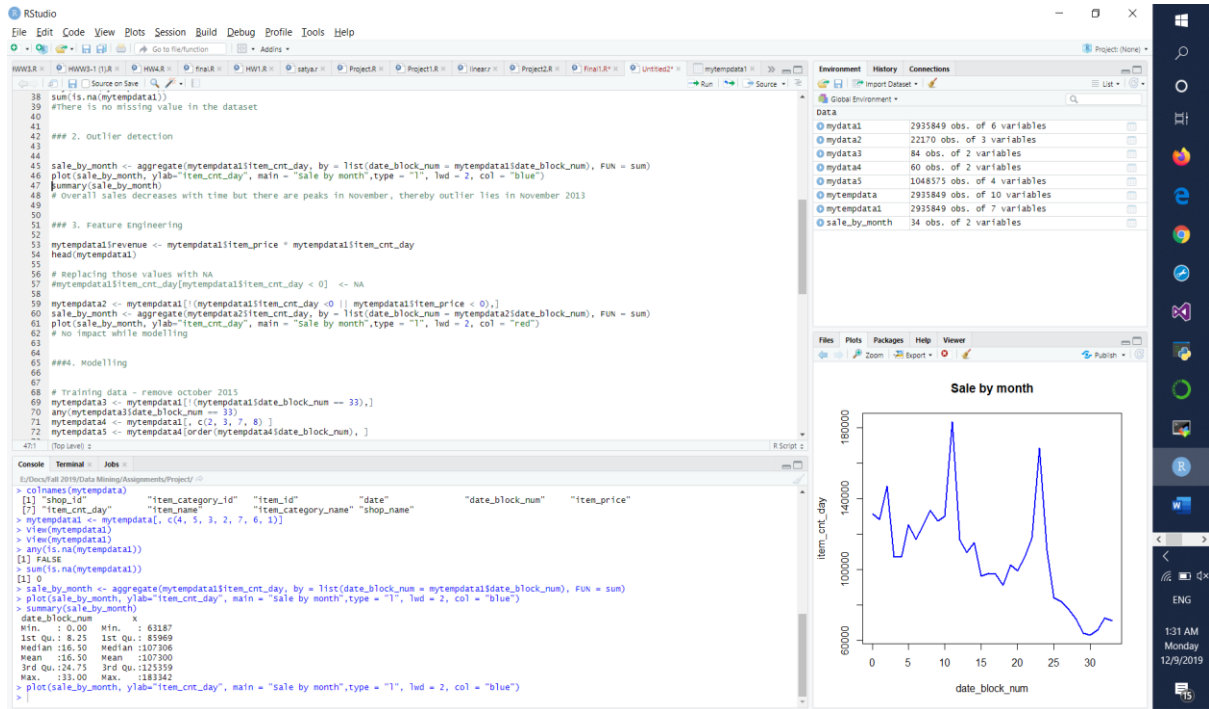
The screenshot displays the RStudio environment with the following components:

- Source Editor:** Contains R code for data pre-processing, including loading libraries (ggplot2, data.table, tidyr, readr, forecast), setting the working directory, loading data from CSV files, merging datasets, and checking for missing values using `any(is.na(mytempdata1))` and `sum(is.na(mytempdata1))`.
- Environment Pane:** Lists the objects in the global environment:
 - `mydata1`: 2935849 obs. of 6 variables
 - `mydata2`: 22170 obs. of 3 variables
 - `mydata3`: 84 obs. of 2 variables
 - `mydata4`: 60 obs. of 2 variables
 - `mydata5`: 1048375 obs. of 4 variables
 - `mytempdata`: 2935849 obs. of 10 variables
 - `mytempdata1`: 2935849 obs. of 7 variables
- Console:** Shows the execution output of the R code, including the results of the missing value checks:

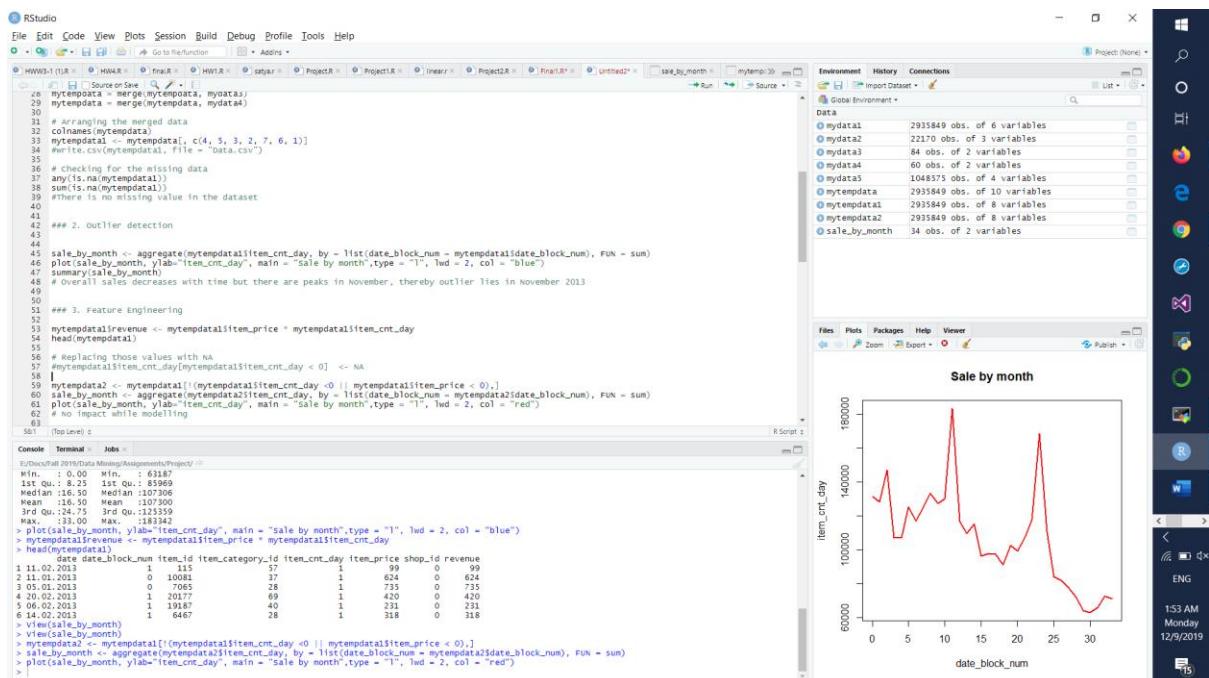

```
[1] FALSE
[1] 0
```
- Files Pane:** Displays the file explorer for the current project.
- Help Pane:** Shows the documentation for the `lapply` function, describing its usage and return value.

2. OUTLIER DETECTION:

- Graph represents the number of products sold for every consecutive month
- From the graph it is observed that the overall sales decrease with time but there are peaks in November months for consecutive years.
- It is found that there are two peaks in consecutive November 2013 and November 2014 which are considered as Outliers.

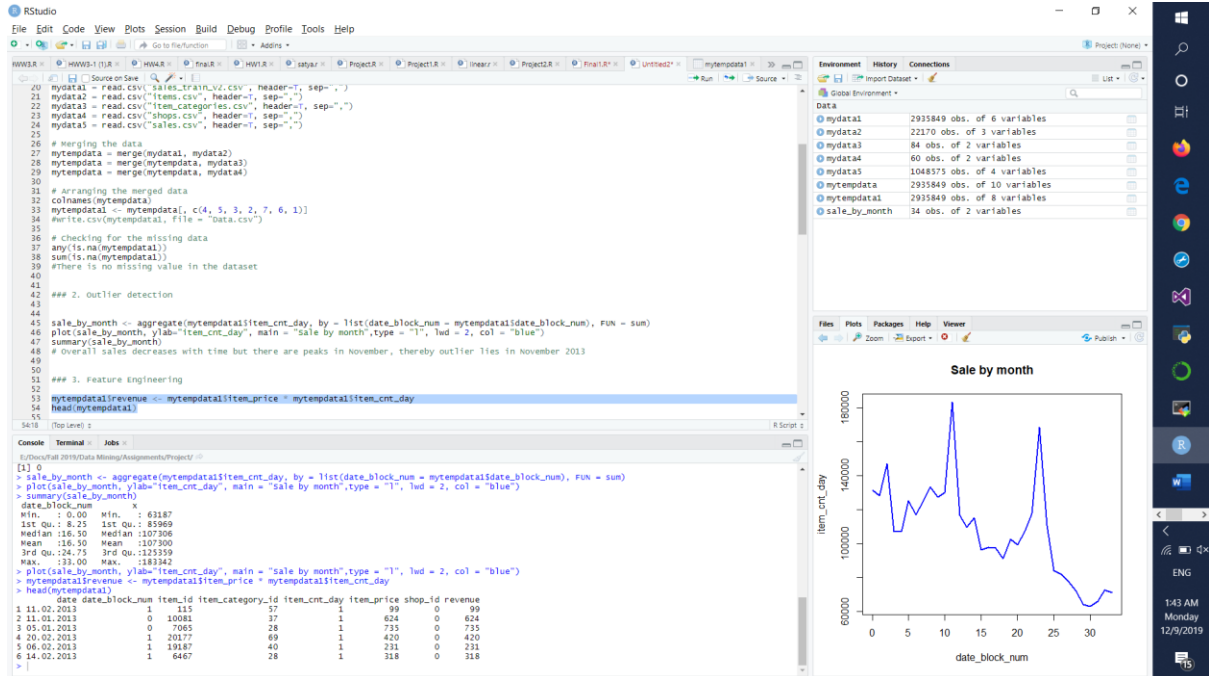


- Replacing the negative values with NA and then plotting the graph (represented by red line) and we found out that it remains the same.

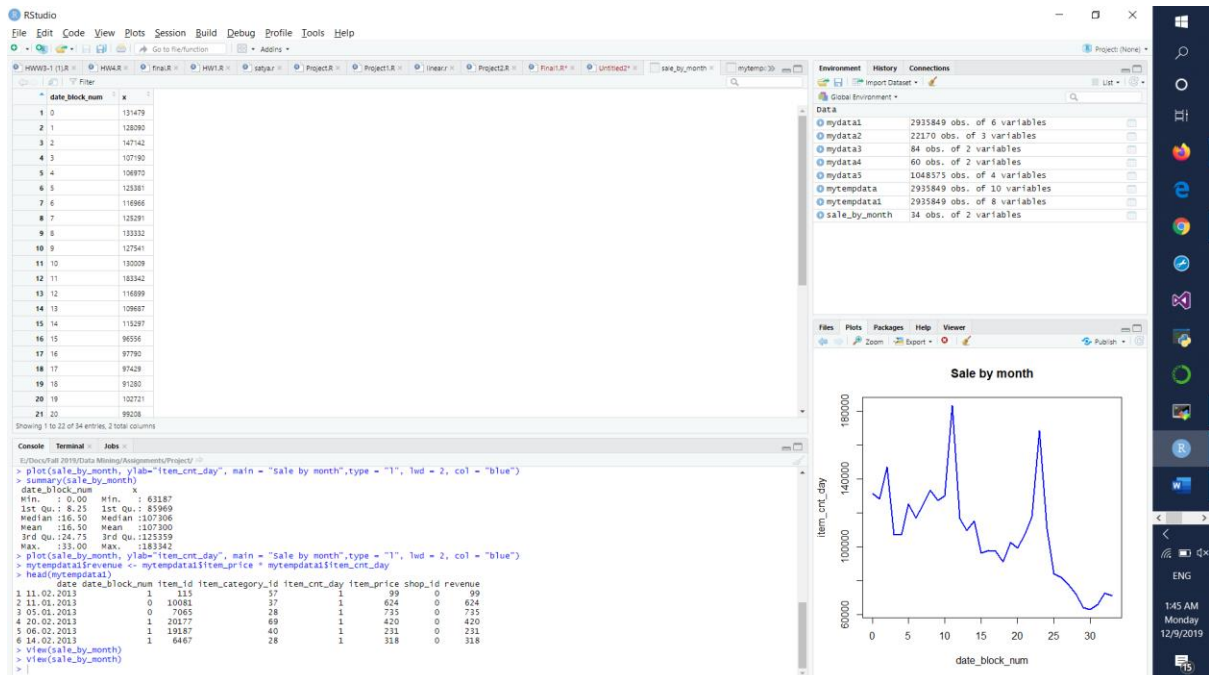


3. Feature Selection/ Engineering:

- We created a new data field Revenue which is the product of no. of products sold (item_cnt_day) and current price of an item (item_price).



- Created Sales_by_month dataset which consists of all the aggregated date_block_num and number of items sold for every consecutive month.



4. Modelling:

- **K Nearest Neighbours (KNN):**

- **Training data:** The merged dataset(mytempdata1) dataset except October 2015.
- **Testing data:** The merged dataset(mytempdata1) dataset for the month of October 2015.

```
# Training data - remove october 2015
mytempdata3 <- mytempdata1[!(mytempdata1$date_block_num == 33),]
any(mytempdata3$date_block_num == 33)
colnames(mytempdata3)
# Testing data (October 2015)
mytempdata4 <- mytempdata1[(mytempdata1$date_block_num == 33),]
colnames(mytempdata4)
```

- **Normalizing data:** We defined the normalize function to normalize data.

```
# Normalize function
normalize=function(x){
  return ((x-min(x))/(max(x)-min(x)))
}
```

- Normalizing the datasets mytempdata3, mytempdata4, mytempdata6 using the normalizing function to mytempdata3_n, mytempdata4_n, mytempdata6_n.

```
# Normalizing data
mytempdata3=mytempdata3[,c(2,3,5,6,7)]
colnames(mytempdata3)
mytempdata3_n=as.data.frame(lapply(mytempdata3[,c(2,3,4,5)],normalize))

mytempdata4 <- mytempdata4[,c(2,3,5,6,7)]
colnames(mytempdata4)
mytempdata4_n=as.data.frame(lapply(mytempdata4[,c(2,3,4,5)],normalize))

mytempdata6=mytempdata3[,c(2,5)]
colnames(mytempdata6)
mytempdata6_n=as.data.frame(lapply(mytempdata6[,], normalize))
```

- Following values shown are the predicted item_cnt_day values.

```
# Building model and results:
```

```
require(class)
model1_knn=knn(train = mytempdata3_n,test = mytempdata4_n, cl=mytempdata3_n$item_cnt_day,k=length(unique(mytempdata3_n$item_cnt_day)))
model1_knn
```

The screenshot displays the RStudio interface. The left pane shows the R script editor with the following code:

```
## Building model and results:
require(class)
model_knn <- knn.train ~ mympdata3.n.test ~ mympdata4.n, c1~mympdata3.n5iten_cnt_day,k~length(unique(mympdata3.n5iten_cnt_day)))
model_knn
pred1 <- predict(model_knn)
res1 <- residuals(mympdata3.n5iten_cnt_day,model_knn)
#crossval <- cv.knn(mympdata3.n5iten_cnt_day, y ~ model_knn, prop.chisq=FALSE)

## 6. validation:
## 6.1. AEMPA model:
library(autopair)

## KNN:
knnse <- mympdata3.n5iten_cnt_day ~ model_knn

## 5. computing confidence interval of model 1 and model 2 for the following different confidence levels: 80%, 90%, 95%
(Top Level >)
```

The right pane shows the R Documentation page for the `groupwiseMeans` function. The title is "Groupwise means and confidence intervals". The description states: "Calculates means and confidence intervals for groups". The usage is: `groupwiseMeans(formula = NULL, data = NULL, var = NULL, conf.level = 0.95, test = FALSE, traditional = TRUE, normal = FALSE, basic = FALSE, percentile = FALSE, box = FALSE, digits = 3, ...)`. The arguments section lists: `formula` (A formula indicating the measurement variable and the grouping variables e.g. `y ~ x1 + x2`), `data` (The data frame to use), `var` (The measurement variable to use. The name is in double quotes), `group` (The grouping variable to use. The name is in double quotes. Multiple names are listed as a vector. (See example)), `conf` (The confidence interval to use), and `R` (The number of bootstrap replicates to use for bootstrapped statistics).

- **ARIMA(Auto Regressive Integrated Moving Average):**

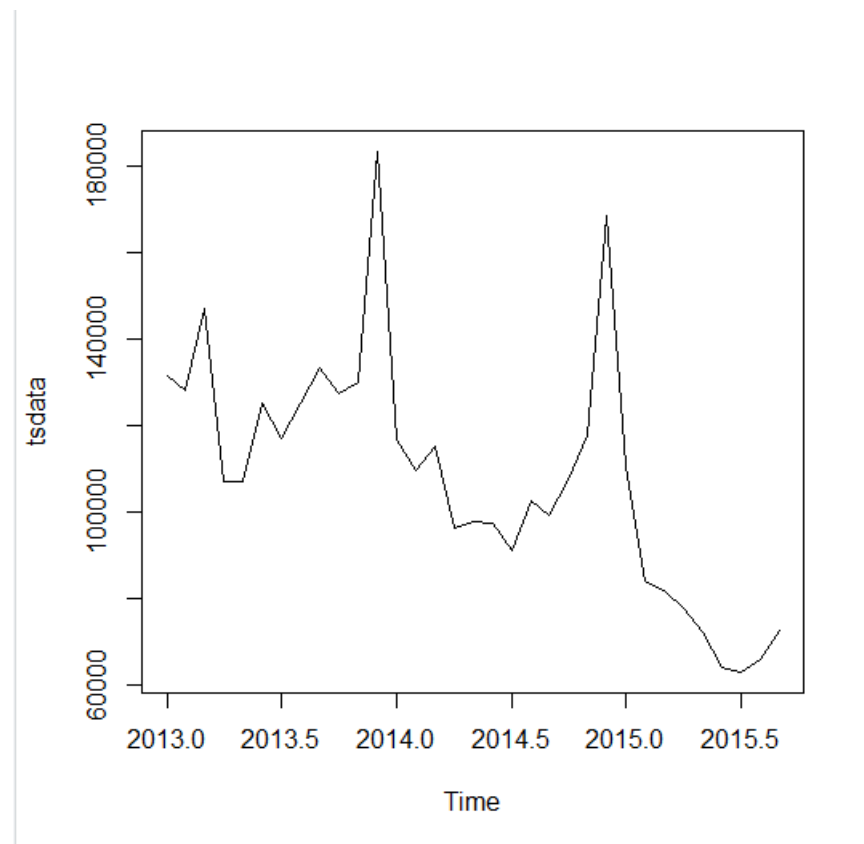
- Arima is useful in our case since sales_by_month plot consists of moving averages and is not constant.
- **Integrated** in the ARIMA refers to the difference in the timestamps.
e.g. Timestamp_1= Sales(February 2013 – January 2013)
- **Training data:** Sales_by_month dataset excluding October 2015.

```
train_data_arima <- sale_by_month [-34,]
```

- **Testing data:** Aggregated number of items sold count for the month of October 2015.

```
tsdata <- ts(train_data_arima$x, frequency = 12, start = c(2013,1))
```

- **Plot of test data:**



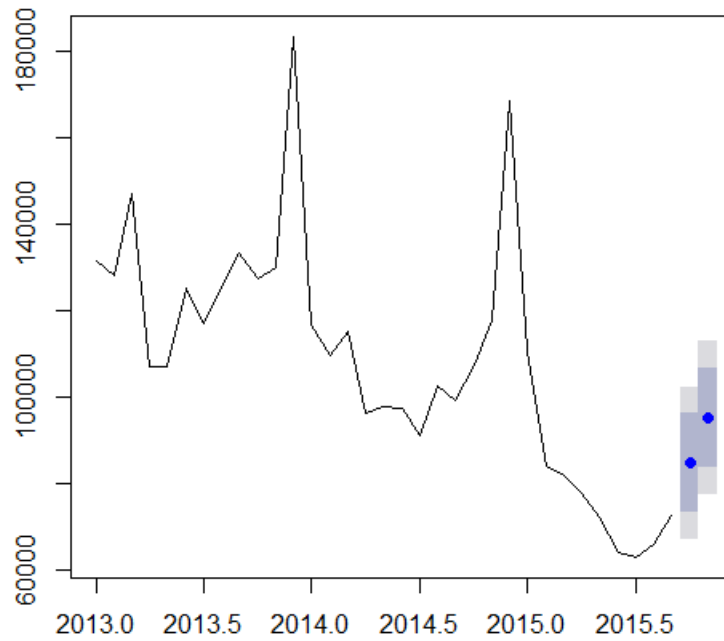
- **Training time for the data set : 2.75 seconds.**

- **Actual forecasting results:**

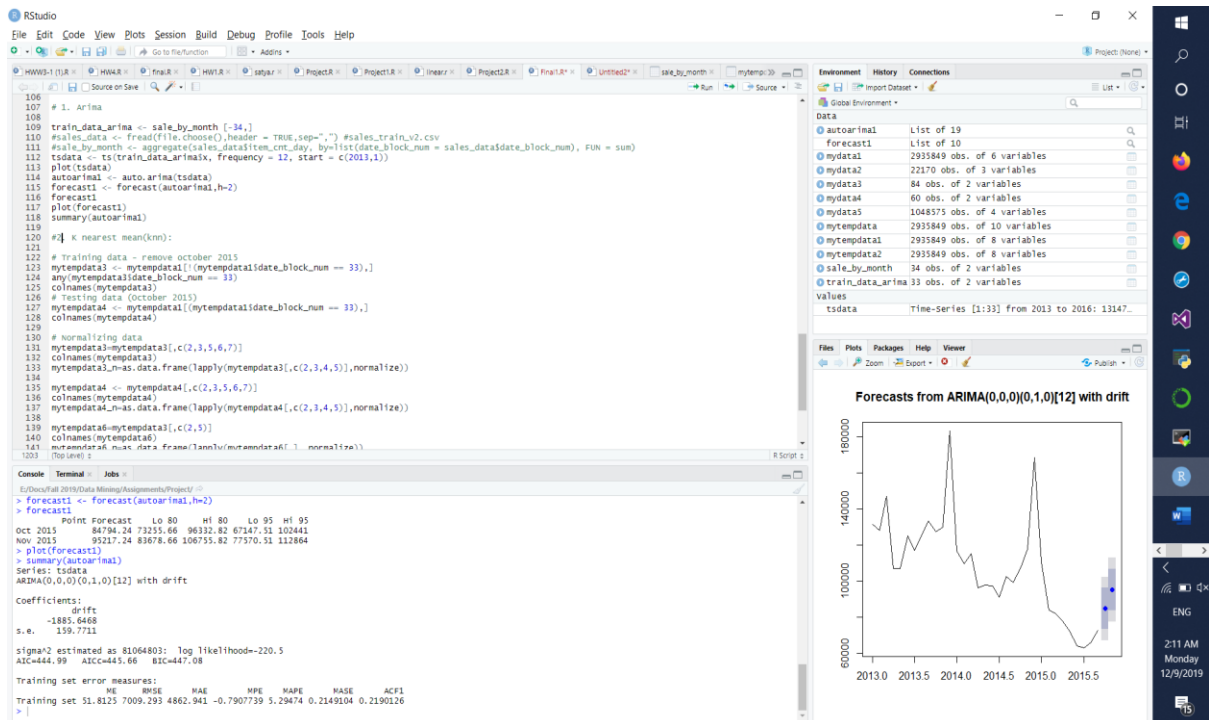
```
> forecast1 <- forecast(autoarima1,h=2)
> forecast1
      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
Oct 2015      84794.24 73255.66 96332.82 67147.51 102441
Nov 2015      95217.24 83678.66 106755.82 77570.51 112864
.
```

- Plotting the Forecasts from ARIMA(0,0,0)(0,1,0)[12] with drift:

Forecasts from ARIMA(0,0,0)(0,1,0)[12] with drift



- Here we can observe the actual and predicted values for months October 2015 and November 2015 in the graph.



5. Validation:

- **KNN Model:**

- **Root Mean Square Error(RMSE):** We got the value around 4.781.

- **Mean Square Error(MSE):** We got this around 2.391

```
> ## KNN:
> RMSE <- mytempdata4_n$item_cnt_day - as.numeric(as.character(model_knn))
> RMSE=RMSE^2
> RMSEout=mean(RMSE)
> RMSEout
[1] 0.0004781815
> RMSEout^2
[1] 2.286576e-07
> RMSEout^1/2
[1] 0.0002390908
```

The screenshot displays the RStudio environment with the following components:

- Source Editor:** Contains R code for data loading, model training, and validation. The code includes comments and function calls like `read.csv`, `aggregate`, `forecast`, and `summary`.
- Environment:** Lists loaded objects such as `mytempdata4`, `mytempdata4_n`, `mytempdata5`, `mytempdata6`, `mytempdata7`, `mytempdata8`, `sale_by_month`, and `tsdata`.
- Console:** Shows the output of the executed code, including the RMSE calculation results: `[1] 0.0004781815`, `[1] 2.286576e-07`, and `[1] 0.0002390908`.
- Documentation:** Displays the documentation for the `groupwiseMean` function, including its description, usage, and arguments.

- **ARIMA Model:**

- ME (Mean Error): The mean error is the term that usually refers to the average of all the errors in a set.
- RMSE (Root Mean Square Error): It is a frequently used measure of the differences between values predicted by a model or an estimator and the values observed.
- MPE (Mean Percentage Error): It is the computed average of percentage errors by which forecasts of a model differ from actual values.
- MAPE (Mean Absolute Percentage Error): It is a measure of prediction accuracy of a forecasting method
- MASE (Mean Absolute Scaled Error): It is a measure of the accuracy of forecasts
- ACF1 (Autocorrelation of errors at lag 1): It is a measure of how much is the current value influenced by the previous values in a time series.
- Here, we observed following validation values for the Arima model:

Training set error measures:

| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 |
|--------------|---------|----------|----------|------------|---------|-----------|-----------|
| Training set | 51.8125 | 7009.293 | 4862.941 | -0.7907739 | 5.29474 | 0.2149104 | 0.2190126 |

6. Compute confidence interval of Model 1 and Model 2 for the following different confidence levels: 90%, 95%

- **KNN Model:**

- Confidence level (80%):

```
> t.test(mytempdata4,  
+        conf.level=0.80)  
  
One Sample t-test  
  
data: mytempdata4  
t = 243.15, df = 267569, p-value < 2.2e-16  
alternative hypothesis: true mean is not equal to 0  
80 percent confidence interval:  
 2405.349 2430.839  
sample estimates:  
mean of x  
 2418.094
```

- Confidence level (90%):

```
> t.test(mytempdata4,  
+        conf.level=0.90)  
  
One Sample t-test  
  
data: mytempdata4  
t = 243.15, df = 267569, p-value < 2.2e-16  
alternative hypothesis: true mean is not equal to 0  
90 percent confidence interval:  
 2401.736 2434.452  
sample estimates:  
mean of x  
 2418.094
```

- Confidence level (95%):

```
> t.test(mytempdata4,  
+        conf.level=0.95)  
  
One Sample t-test  
  
data: mytempdata4  
t = 243.15, df = 267569, p-value < 2.2e-16  
alternative hypothesis: true mean is not equal to 0  
95 percent confidence interval:  
 2398.602 2437.586  
sample estimates:  
mean of x  
 2418.094
```

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Global Environment | History | Connections

Global Environment

- mytempdata4 53514 obs. of 5 variables
- mytempdata4_n 53514 obs. of 4 variables
- mytempdata5 2955849 obs. of 4 variables
- mytempdata6 2882335 obs. of 2 variables
- mytempdata6_n 2882335 obs. of 2 variables
- mytempdata7 53514 obs. of 8 variables
- mytempdata8 53514 obs. of 4 variables
- sale_by_month 34 obs. of 2 variables
- train_data_arima 33 obs. of 2 variables

values

all_means Named num [1:35] NA 25.1 25.4 25 25.2 ...

i IL

j IL

model_knn Factor w/ 197 levels "0","0.005870841487279...

rmse num [1:53514] 0.000466 0.000466 0.000466 0...

rmseout 0.000478181505255783

tsdata Time-series [1:33] From 2013 to 2016: 13147...

Files Plots Packages Help Viewer

R-Groupwise means and confidence intervals

groupwiseMean (rcompanion) R Documentation

Groupwise means and confidence intervals

Description

Calculates means and confidence intervals for groups

Usage

```
groupwiseMean(formula = NULL, data = NULL, var = NULL,
  group = NULL, conf = 0.95, B = 5000, boot = FALSE,
  traditional = TRUE, normal = FALSE, banjo = FALSE,
  presmooth = FALSE, box = FALSE, digits = 3, ...)
```

Arguments

formula A formula indicating the measurement variable and the grouping variables e.g. y ~ x1 + x2.

data The data frame to use.

var The measurement variable to use. The name is in double quotes.

group The grouping variable to use. The name is in double quotes. Multiple names are listed as a vector. (See example.)

conf The confidence interval to use.

B The number of bootstrap replicates to use for bootstrapped statistics.

```
154 RMSEOUT=1/2
155
156 # Validation for Arima model( RMSE and MSE):
157 summary(autoarima)
158
159 ## 6. Computing confidence interval of Model 1 and Model 2 for the following different confidence levels: 80%, 90%, 95%
160
161 library(rcompanion)
162
163 ## 1. KNN:
164 t.test(mytempdata4,
165       conf.level=0.80)
166 t.test(mytempdata4,
167       conf.level=0.90)
168 t.test(mytempdata4,
169       conf.level=0.95)
170
171 ## 2. ARIMA:
172 t.test(tsdata,
173       conf.level=0.80)
174 t.test(tsdata, model_knn, paired = TRUE,
175       conf.level=0.90)
176 t.test(tsdata, model_knn, paired = TRUE,
177       conf.level=0.95)
178
179 Console Terminal Jobs
180
181 t = 243.15, df = 287569, p-value < 2.2e-16
182 alternative hypothesis: true mean is not equal to 0
183 90 percent confidence interval:
184 2401.736 2434.452
185 sample estimates:
186 mean of x
187 2418.094
188
189 > t.test(mytempdata4,
190       conf.level=0.95)
191
192 One Sample t-test
193
194 data: mytempdata4
195 t = 243.15, df = 287569, p-value < 2.2e-16
196 alternative hypothesis: true mean is not equal to 0
197 90 percent confidence interval:
198 2398.602 2437.586
199 sample estimates:
200 mean of x
201 2418.094
```

- **ARIMA Model:**

- Confidence level (80%):

```
> t.test(tsddata,  
+        conf.level=0.80)  
  
One Sample t-test  
  
data:  tsdata  
t = 22.116, df = 32, p-value < 2.2e-16  
alternative hypothesis: true mean is not equal to 0  
80 percent confidence interval:  
 101984.8 114812.2  
sample estimates:  
mean of x  
 108398.5
```

- Confidence level (90%):

```
> t.test(tsddata,  
+        conf.level=0.90)  
  
One Sample t-test  
  
data:  tsdata  
t = 22.116, df = 32, p-value < 2.2e-16  
alternative hypothesis: true mean is not equal to 0  
90 percent confidence interval:  
 100096.2 116700.8  
sample estimates:  
mean of x  
 108398.5
```

- Confidence level (95%):

```
> t.test(tsddata,  
+        conf.level=0.95)  
  
One Sample t-test  
  
data:  tsdata  
t = 22.116, df = 32, p-value < 2.2e-16  
alternative hypothesis: true mean is not equal to 0  
95 percent confidence interval:  
 98414.84 118382.13  
sample estimates:  
mean of x  
 108398.5
```

RStudio interface showing R code execution and environment details.

Code Editor:

```
137 # Normalizing data
138 mytempdata3=mytempdata3[,c(2,3,5,6,7)]
139 colnames(mytempdata3)
140 mytempdata3_n=as.data.frame(lapply(mytempdata3[,c(2,3,4,5)],normalize))
141
142 mytempdata4 = mytempdata4[,c(2,3,5,6,7)]
143 colnames(mytempdata4)
144 mytempdata4_n=as.data.frame(lapply(mytempdata4[,c(2,3,4,5)],normalize))
145
146 mytempdata6=mytempdata3[,c(2,5)]
147 colnames(mytempdata6)
148 mytempdata6_n=as.data.frame(lapply(mytempdata6[,], normalize))
149
150 #
151 # Building model and results:
152 require(class)
153 model_knn_knn(train = mytempdata3_n, test = mytempdata4_n, cl=mytempdata3_n$item_cnt_day,k=length(unique(mytempdata3_n$item_cnt_day)))
154 table(mytempdata3_n$item_cnt_day,model_knn)
155
156 # computing confidence interval of Model 1 and Model 2 for the following different confidence levels: 80%, 90%, 95%
157
158 # Arima Model:
159
160
161 library(rcompanion)
162
163 t.test(tsdata,
164       conf.level=0.80)
165
166 t.test(tsdata,
167       conf.level=0.90)
168
169 t.test(tsdata,
170       conf.level=0.95)
171
```

Console:

```
alternative hypothesis: true mean is not equal to 0
90 percent confidence interval:
 100096.2 116700.8
sample estimates:
mean of x
108398.5

> t.test(tsdata,
+       conf.level=0.95)
one sample t-test

data: tsdata
t = 22.116, df = 32, p-value = 2.2e-18
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 96534.84 116182.13
sample estimates:
mean of x
108398.5
```

Environment:

- Global Environment
- mydata5: 1048375 obs. of 4 variables
- mytempdata5: 2935849 obs. of 10 variables
- mytempdata1: 2935849 obs. of 8 variables
- mytempdata2: 2935849 obs. of 8 variables
- mytempdata3: 2862335 obs. of 8 variables
- mytempdata4: 2935849 obs. of 4 variables
- mytempdata5: 2935849 obs. of 4 variables
- mytempdata7: 53514 obs. of 8 variables
- mytempdata8: 53514 obs. of 4 variables
- sale_by_month: 34 obs. of 2 variables
- train_data_arima: 33 obs. of 2 variables

Values:

```
all_means Named num [1:35] NA 25.1 25.4 25 25.2 ...
i         1L
j         1L
tsdata    Time-Series [1:33] From 2013 to 2016: 13147...
```

Files, Plots, Packages, Help, Viewer:

Groupwise means and confidence intervals

Description:

Calculates means and confidence intervals for groups.

Usage:

```
groupwiseMean(formula = NULL, data = NULL, var = NULL,
              group = NULL, conf = 0.95, R = 5000, boot = FALSE,
              traditional = TRUE, normal = FALSE, basis = FALSE,
              percentile = FALSE, box = FALSE, digits = 3, ...)
```

Arguments:

- formula**: A formula indicating the measurement variable and the grouping variables e.g. `y ~ x1 + x2`
- data**: The data frame to use.
- var**: The measurement variable to use. The name is in double quotes.
- group**: The grouping variable to use. The name is in double quotes. Multiple names are listed as a vector. (See example.)
- conf**: The confidence interval to use.
- R**: The number of bootstrap replicates to use for bootstrapped statistics.

7. Comparison between ARIMA and KNN model:

1. KNN Model:

- **Error:**
 - a. **Root Mean Square Error(RMSE):** We got the value around 4.781.
 - b. **Mean Square Error(MSE):** We got this around 2.391

The screenshot displays the RStudio interface with a script editor, console, and environment pane. The script defines functions for training and validating both ARIMA and KNN models. The console output shows the results of these functions, including RMSE and MSE values.

```
# 2. ARIMA:
train_data_arima <- sales_by_month[1:34]
#sales_data <- read.csv(file.choose(), header = TRUE, sep = ",") #sales_train.csv
#sales_by_month <- aggregate(sales_data[,item_cnt_day, by=list(date_block_num = sales_data$date_block_num), FUN = sum)
tsdata <- ts(train_data_arima[, frequency = 12, start = c(2013,1)])
plot(tsdata)
autoarima <- auto.arima(tsdata)
forecast1 <- forecast(autoarima,h=2)
plot(forecast1)

## 5. validation:
# 2. Root Mean Square Error:
RMSE <- mytempdata4_n[1:34] - as.numeric(as.character(model_knn))
RMSEout <- RMSE^2
RMSEout <- mean(RMSEout)
RMSEout

# 1. Mean Square Error:
RMSEout <- RMSE^2
RMSEout <- mean(RMSEout)
RMSEout

# validation for Arima model( RMSE and MSE):
summary(autoarima)
```

Console output:

```
Warning message:
In Ops.factor(mytempdata4_n[1:34], model_knn) :
'not meaningful for factors'
## KNN:
RMSE <- mytempdata4_n[1:34] - as.numeric(as.character(model_knn))
RMSEout <- RMSE^2
RMSEout <- mean(RMSEout)
RMSEout
[1] 0.0004781815
RMSEout^2
[1] 2.286576e-07
RMSEout^2
[1] 0.0002390908
```

Environment pane shows the following objects:

- mytempdata4: 53514 obs. of 5 variables
- mytempdata4_n: 53514 obs. of 4 variables
- mytempdata5: 2935849 obs. of 4 variables
- mytempdata6: 2882335 obs. of 2 variables
- mytempdata6_n: 2882335 obs. of 2 variables
- mytempdata7: 53514 obs. of 8 variables
- mytempdata8: 53514 obs. of 4 variables
- sale_by_month: 34 obs. of 2 variables

The R Documentation pane shows the documentation for the `groupwiseMean` function, which calculates means and confidence intervals for groups.

a. Efficiency in training time (scalability):

Training time for the data set : 5 minutes.

2. ARIMA Model:

b. Errors:

- a. ME (Mean Error): The mean error is the term that usually refers to the average of all the errors in a set.
- b. RMSE (Root Mean Square Error): It is a frequently used measure of the differences between values predicted by a model or an estimator and the values observed.
- c. MPE (Mean Percentage Error): It is the computed average of percentage errors by which forecasts of a model differ from actual values.
- d. MAPE (Mean Absolute Percentage Error): It is a measure of prediction accuracy of a forecasting method
- e. MASE (Mean Absolute Scaled Error): It is a measure of the accuracy of forecasts
- f. ACF1 (Autocorrelation of errors at lag 1): It is a measure of how much is the current value influenced by the previous values in a time series.
- g. Here, we observed following validation values for the Arima model:

```
Training set error measures:
      ME      RMSE      MAE      MPE      MAPE      MASE      ACF1
Training set 51.8125 7009.293 4862.941 -0.7907739 5.29474 0.2149104 0.2190126
```

c. Efficiency in training time (scalability):

Training time for the data set : 2.75 seconds.