

# Data Mining Final Project (FALL 2019)

## 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 shows the RStudio interface. The main editor displays a data frame with columns: date, date\_block\_num, item\_id, item\_category\_id, item\_cnt\_day, item\_price, and shop\_id. The environment pane on the right lists several data frames: mydata1 (2935849 obs., 6 variables), mydata2 (22170 obs., 3 variables), mydata3 (84 obs., 2 variables), mydata4 (60 obs., 2 variables), mydata5 (1048575 obs., 4 variables), mytempdata (2935849 obs., 10 variables), and mytempdata1 (2935849 obs., 7 variables). The console shows 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)
[1] "shop_id" "item_category_id" "item_id" "date" "date_block_num" "item_price"
[7] "item_cnt_day" "item_name" "item_category_name" "shop_name"
> 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.

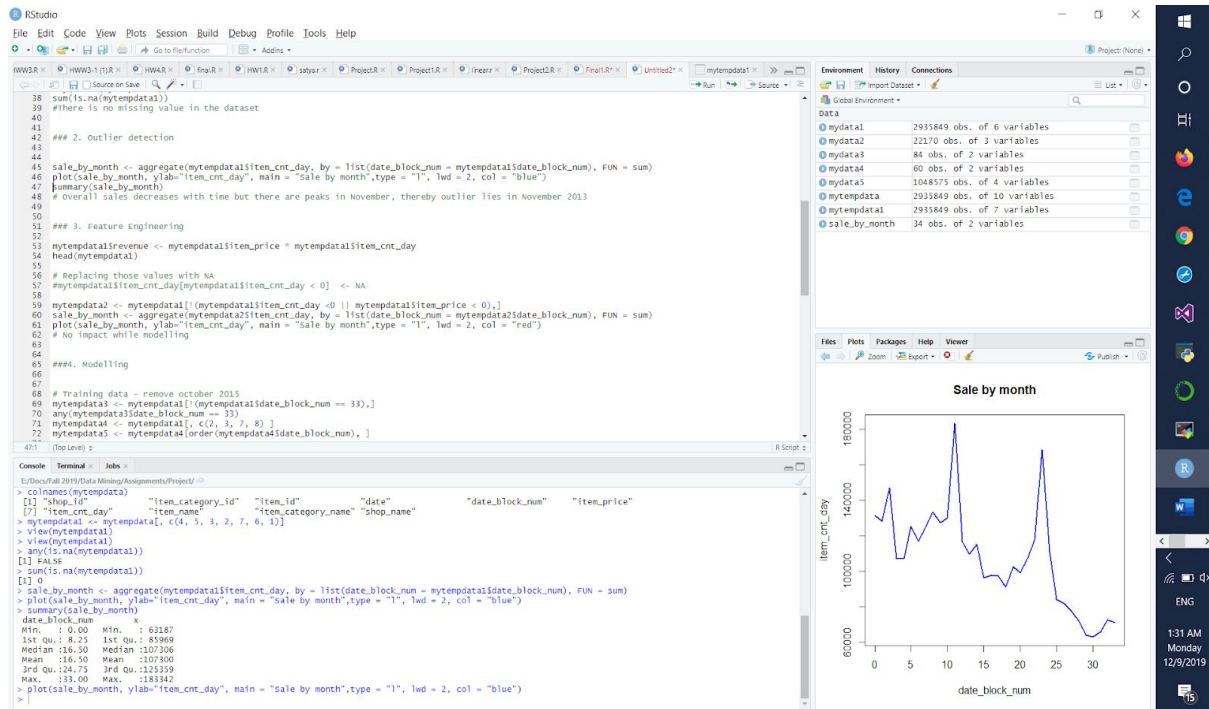
- Removed categorial attributes that were not required (item\_name, item\_category\_name, shop\_name).

The screenshot displays the RStudio environment with the following components:

- Source Editor:** Contains R code for data pre-processing, including loading libraries (ggthemes, data.table, xlsx, readxl, forecast), setting the directory, loading data from CSV files, merging datasets, and checking for missing values.
- Environment Pane:** Lists the loaded datasets: 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 (104875 obs. of 4 variables), mytempdata (2935849 obs. of 10 variables), and mytempdata1 (2935849 obs. of 7 variables).
- Console:** Shows the execution of the R code, including the output of the `sum(is.na(mytempdata1))` command, which returns 0, indicating no missing values.
- Files Pane:** Displays the file structure of the project, including the `data` directory.
- Help Pane:** Shows the documentation for the `lapply` function, describing its usage and arguments.

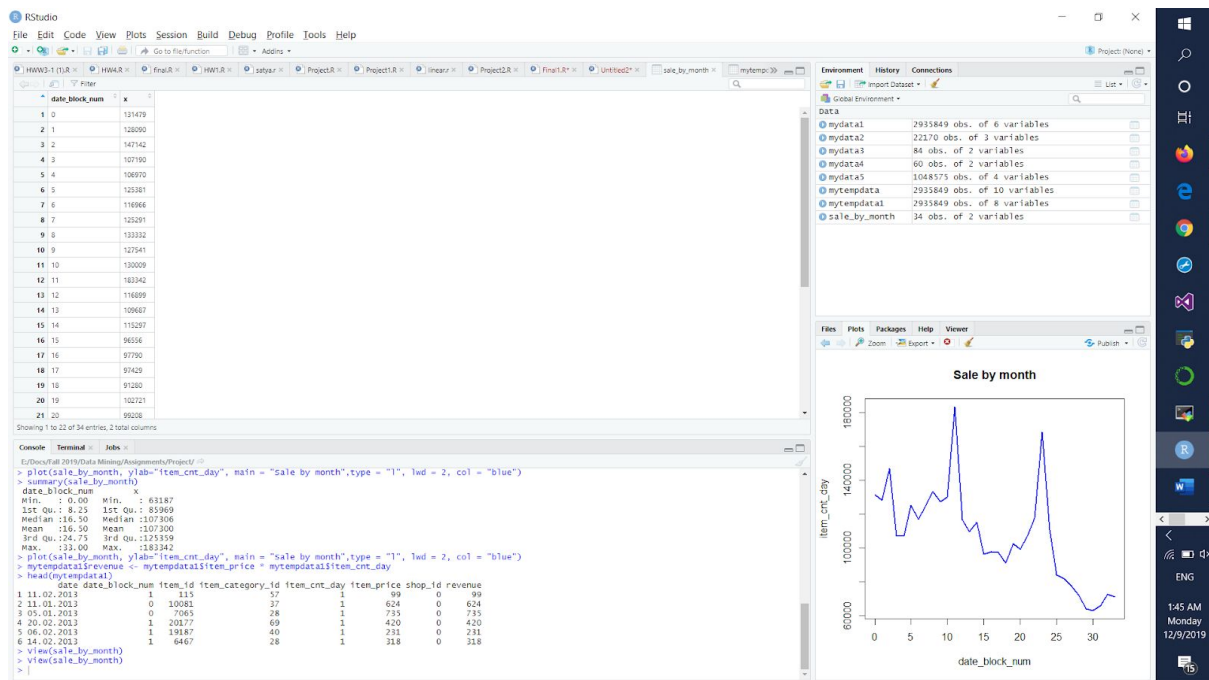
## 2. OUTLIER DETECTION:

- 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.





## 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))

```

## o Building Model and output:

- Following values shown are the predicted item\_cnt\_day values.

```

# Building model and results:
require(class)
model_knn=knn(train = mytempdata3_n,test = mytempdata4_n, cl=mytempdata3_n$item_cnt_day,k=length(unique(mytempdata3_n$item_cnt_day)))
model_knn

```

The screenshot displays the RStudio environment with a script editor on the left and the Environment pane on the right. The script editor contains R code for building a K-NN model and validating it. The Environment pane shows the loaded datasets and the results of the model training.

```

142 colnames(mytempdata3)
143 mytempdata3_nas.data.frame(tapply(mytempdata3[,c(2,3,4,5)],normalize))
144
145 mytempdata3_n2 <- merge.data.frame(mytempdata3, mytempdata3_n)
146 mytempdata3_n2 = merge(mytempdata3[, c(1,5)], mytempdata3_n2)
147
148 mytempdata4 <- mytempdata4[,c(2,3,5,6,7)]
149 colnames(mytempdata4)
150 mytempdata4_nas.data.frame(tapply(mytempdata4[,c(2,3,4,5)],normalize))
151
152 mytempdata6-mytempdata3[,c(2,5)]
153 colnames(mytempdata6)
154 mytempdata6_nas.data.frame(tapply(mytempdata6[, ], normalize))
155
156
157 # Building model and results:
158 require(class)
159 model_knn=knn(train = mytempdata3_n, test = mytempdata4_n, cl=mytempdata3_n$item_cnt_day,k=length(unique(mytempdata3_n$item_cnt_day)))
160 model_knn
161 #summary(model_knn)
162 #predict(model_knn)
163 #table(mytempdata3_n$item_cnt_day,model_knn)
164 #crossTable(x = mytempdata3_n$item_cnt_day, y = model_knn, prop.chisq=FALSE)
165
166 ## 6. Validation:
167
168 ## ARIMA Model:
169 summary(autorimais)
170
171 ## KNN:
172 rmse <- mytempdata4_n$item_cnt_day - model_knn
173
174
175 ## 5. Computing confidence interval of Model 1 and Model 2 for the following different confidence levels: 80%, 90%, 95%
176 (Top Level) :
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

The Environment pane shows the following objects:

- Global Environment
- mytempdata2: 2935849 obs. of 8 variables
- mytempdata3: 2882335 obs. of 5 variables
- mytempdata3\_n: 2882335 obs. of 4 variables
- 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 4 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.

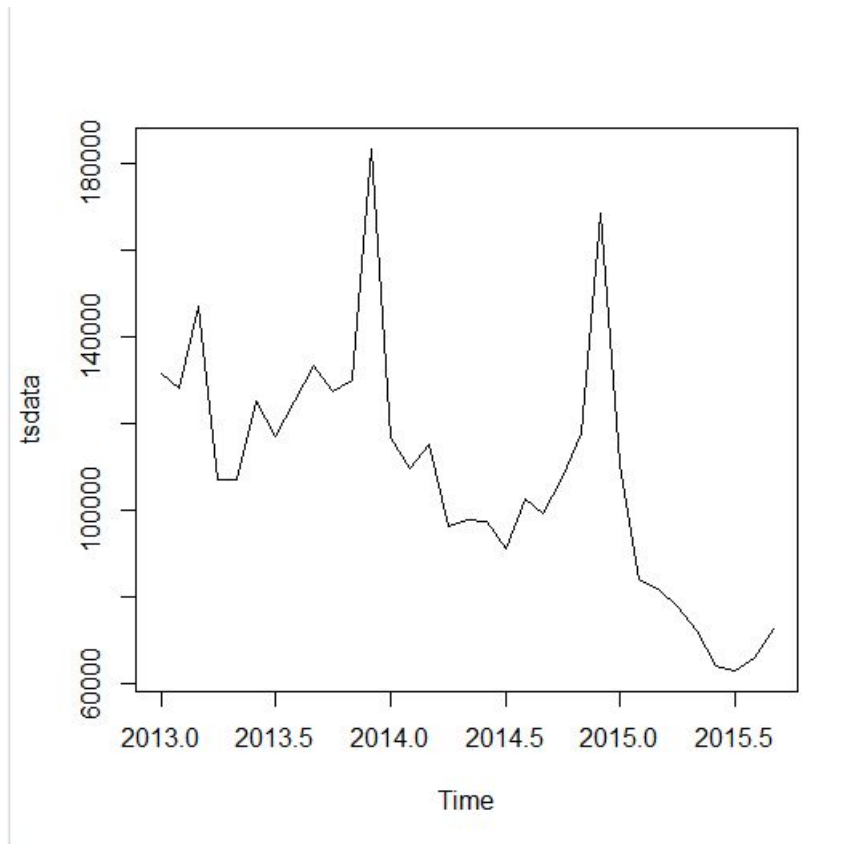
- **ARIMA(Auto Regressive Integrated Moving Average):**
  - Arima is useful in our case since sales\_by\_month plot consists of moving averages and is not constant.
  - **Integrated** in the ARIMA refers to the difference in the timestamps. e.g. **Timestamp\_1= Sales(February 2013 – January 2013)**
  - **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:**

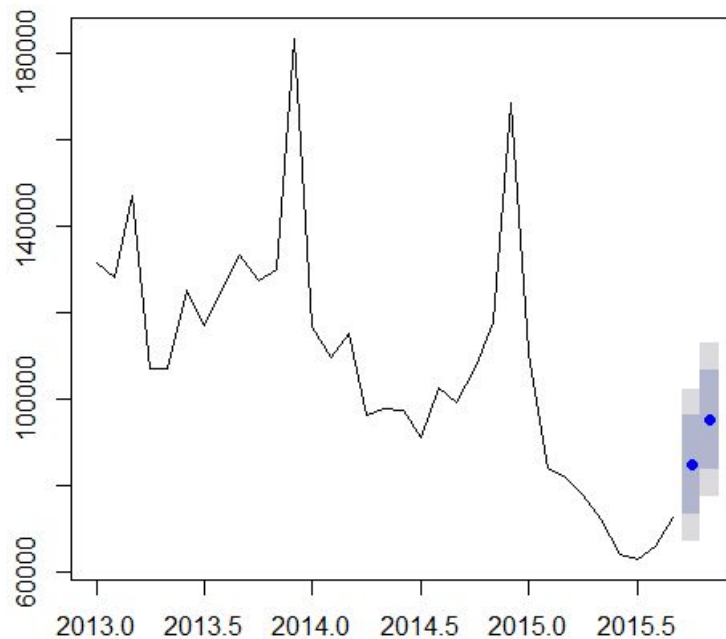
```
> forecast1 <- forecast(autoarima1,h=2)
> forecast1
```

	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

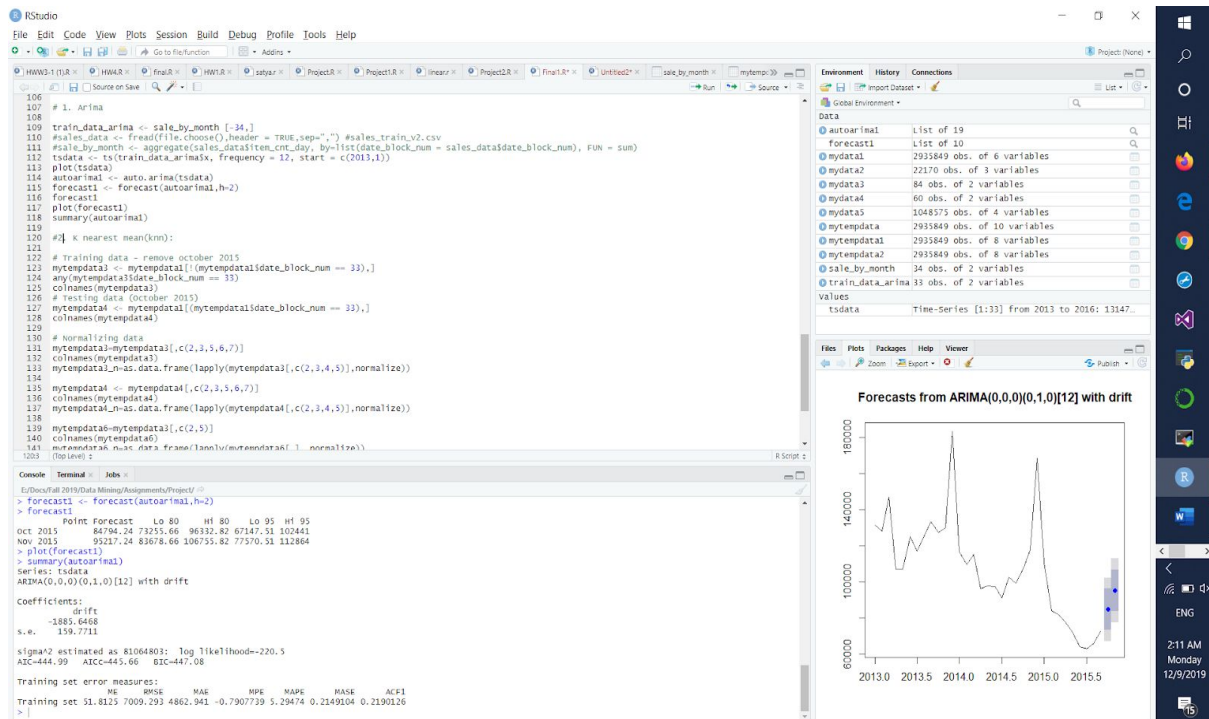
- **Plotting the Forecasts from ARIMA(0,0,0)(0,1,0)[12] with drift:**



## 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 processing and model evaluation. Key lines include:
 

```
## 2. ARIMA:
train_data_arima <- sale_by_month[-34,]
#sales_data <- fread(file.choose(),header = TRUE,sep="," ) #sales_train.csv
#sale_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$item_cnt_day ~ as.numeric(as.character(model_knn))
RMSE=RMSE^2
RMSEout=mean(RMSE)
RMSEout
# 1. Mean Square Error:
RMSEout^1/2

# Validation for Arima model( RMSE and MSE):
summary(autoarima)
```
- Environment Pane:** Lists global environment variables:
  - 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
- Values Pane:** Shows values for variables like `all_means`, `i`, `j`, `model_knn`, `RMSE`, `RMSEout`, and `normalize`.
- Files, Plots, Packages, Help, Viewer:** Standard RStudio toolbars.
- Console:** Displays the output of the R code, including the RMSE calculation and the summary of the ARIMA model.
- R Documentation:** Shows 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

Source Editor: `rmseout1/2`

```
154 # Validation for Arima model ( RMSE and MSE):
155
156 summary(autoarima)
157
158 ## 6. computing confidence interval of Model 1 and Model 2 for the following different confidence levels: 80%, 90%, 95%
159
160
161 library(companion)
162
163 ## 1. KNN:
164
165 t.test(mytempdata4,
166       conf.level=0.80)
167
168 t.test(mytempdata4,
169       conf.level=0.90)
170
171 t.test(mytempdata4,
172       conf.level=0.95)
173
174 t.test(mytempdata4,
175       conf.level=0.95)
176
177 ## 2. ARIMA:
178
179 t.test(tsdata,
180       conf.level=0.80)
181
182 t.test(tsdata, model_knn, paired = TRUE,
183       conf.level=0.90)
184
185 t.test(tsdata, model_knn, paired = TRUE,
186       conf.level=0.95)
187
188
```

Console

```

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

> 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

```

Environment

Object	Class	Attributes
mytempdata4	data.frame	53514 obs. of 3 variables
mytempdata4_n	data.frame	53514 obs. of 4 variables
mytempdata5	data.frame	2935849 obs. of 4 variables
mytempdata6	data.frame	2882335 obs. of 2 variables
mytempdata6_n	data.frame	2882335 obs. of 2 variables
mytempdata7	data.frame	53514 obs. of 8 variables
mytempdata8	data.frame	53514 obs. of 4 variables
sale_by_month	data.frame	34 obs. of 2 variables
train_data_arima	data.frame	33 obs. of 2 variables

Values

all_means	Named num [1:33]	NA	25.1	25.4	25	25.2	...
i	1L						
j	1L						
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 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, k = NULL, boot = FALSE,
  traditional = TRUE, normal = FALSE, basic = FALSE,
  percentile = FALSE, box = FALSE, digits = 3, ...)
```

Arguments

Argument	Description
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.
k	The number of bootstrap replicates to use for bootstrapped statistics.



- **ARIMA Model:**

- Confidence level (80%):

```
> t.test(tsdata,  
+        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(tsdata,  
+        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(tsdata,  
+        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 a script editor, console, and environment pane.

**Script Editor:**

```
## Normalizing data
mytempdata3=mytempdata3[,c(2,3,5,6,7)]
colnames(mytempdata3)
mytempdata3=as.data.frame(apply(mytempdata3[,c(2,3,4,5)],normalize))
mytempdata4 <- mytempdata4[,c(2,3,5,6,7)]
colnames(mytempdata4)
mytempdata4=as.data.frame(apply(mytempdata4[,c(2,3,4,5)],normalize))
mytempdata5=mytempdata5[,c(2,5)]
colnames(mytempdata5)
mytempdata6=as.data.frame(apply(mytempdata6[,], normalize))

# Building model and results:
require(class)
model_knn=knn(train = mytempdata3_n,test = mytempdata4_n, c1=mytempdata3_n$ten_cnt_day,k=length(unique(mytempdata3_n$ten_cnt_day)))
table(mytempdata3_n$ten_cnt_day,model_knn)

# Computing confidence interval of Model 1 and Model 2 for the following different confidence levels: 80%, 90%, 95%
# Arima Model:
library(rcompanion)
t.test(tsdata,
       conf.level=0.80)
t.test(tsdata,
       conf.level=0.90)
t.test(tsdata,
       conf.level=0.95)
```

**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-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
```

**Environment:**

Object	Class	Attributes
mytempdata3	matrix	1048375 obs. of 4 variables
mytempdata4	matrix	2935849 obs. of 10 variables
mytempdata1	matrix	2935849 obs. of 8 variables
mytempdata2	matrix	2935849 obs. of 8 variables
mytempdata3	matrix	2882335 obs. of 8 variables
mytempdata4	matrix	2935849 obs. of 4 variables
mytempdata5	matrix	2935849 obs. of 4 variables
mytempdata7	matrix	53514 obs. of 8 variables
mytempdata8	matrix	53514 obs. of 4 variables
sale_by_month	matrix	34 obs. of 2 variables
train_data_arima	matrix	33 obs. of 2 variables

**Files:**

- all\_means
- i
- j
- tsdata

**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, basic = FALSE,
              percentiles = FALSE, box = FALSE, digits = 3, ...)
```

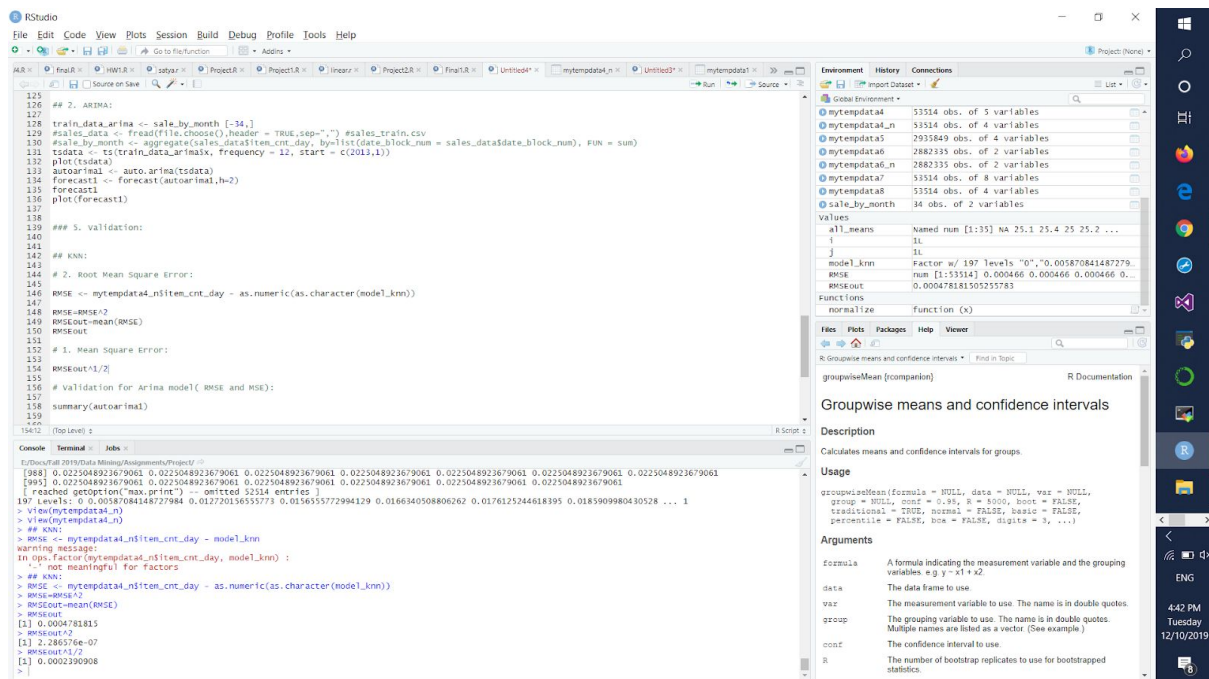
**Arguments**

Argument	Description
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.

## 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



- 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.