Ryerson University

Master of Data Science and Analytics

Visualization of Exploratory Analysis and Forecasting using Online Retail Data

Table of Contents

Table of Figures

Introduction:

Online retailers can increase their sales and profits faster than a brick and mortar shop as selling online offers the advantage of having an open store, twenty-four hours a day, seven days a week. This advantage allows online retailers to expand their market to global proportions or target an extremely focused segment [1]. To gain profit in online retailing, it is important to set business strategies focusing customer buying attitude along with the sales' trends. In this project we aimed to perform exploratory analysis on 'Online Retail' data set to predict different trends of purchased online products. Based on the exploratory analysis, this project aims to visualize forecast future selling trend.

Dataset:

"The 'Online Retail' data set (available at https://archive.ics.uci.edu/ml/datasets/Online+Retail) is a transnational data set which contains all the transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based and registered non-store online retail. The company mainly sells unique all-occasion gifts. Many customers of the company are wholesalers" [2].

Attribute Information:

InvoiceNo: Invoice number. Nominal, a 6-digit integral number uniquely assigned to each transaction. If this code starts with letter 'c', it indicates a cancellation.
StockCode: Product (item) code. Nominal, a 5-digit integral number uniquely assigned to each distinct product.
Description: Product (item) name. Nominal.
Quantity: The quantities of each product (item) per transaction. Numeric.
InvoiceDate: Invice Date and time. Numeric, the day and time when each transaction was generated.
UnitPrice: Unit price. Numeric, Product price per unit in sterling.
CustomerID: Customer number. Nominal, a 5-digit integral number uniquely assigned to each customer.
Country: Country name. Nominal, the name of the country where each customer resides.

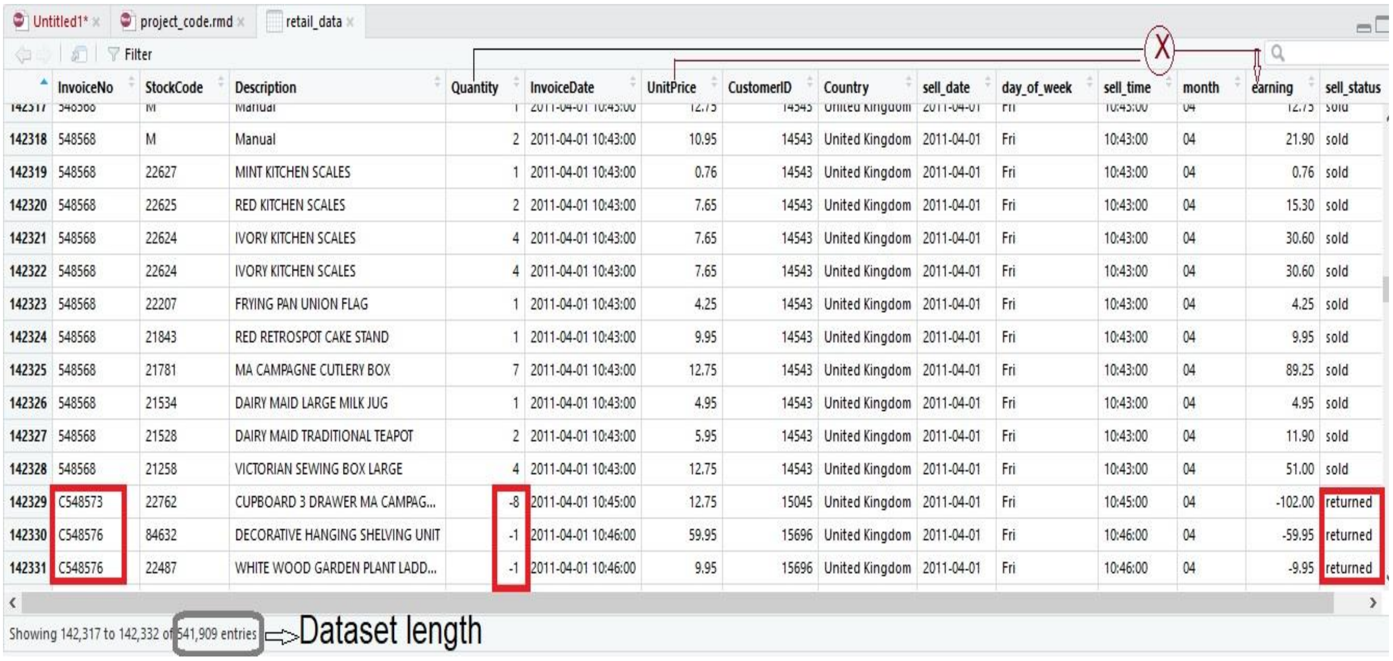
Initial Setup

We choose to implement our visualization using R as it is one of the most powerful tool for statistical analysis. We have installed required packages and libraries at the beginning to setup our project environment to run the code smoothly. List of those packages and libraries are given below.

Libraries	**Purposes**
broom	To prepare a tidy frame of statistical objects those used later by tidyverse and tidyquant.
caret	To perform regression and classification on our dataset.
ggcorrplot	To visualize correlations among the features of our dataset.
grid	To convert the visualization between different grid systems.
gridextra	To visualize multiple grid base plots.
matrixStats	To perform matrix operations on our dataset.
modelr	To integrate modelling into a pipeline of our dataset manipulation and visualization.
prophet	To forecast timeseries.
tidyverse	To provide a set of libraries those used for analysis and visualization.
timetk	To forecast timeseries.
tidyquant	To provide better visualizations of different plots.

Data Loading and Pre-processing:

We have loaded the dataset “Online Retail.csv” and extract some information from the “InvoiceDate”, “Quantity” and “UnitPrice” features for analysis purpose. The “InvoiceDate” column contained date in year-month-date format composed with the transaction time in hh:mm:ss format. From this, we added 4 columns named sell_date (year-month-date), day_of_week (e.g. Mon, Tue, etc.), sell_time (hh:mm:ss) and month (1 to12). We further added one column named ‘earning’ with the corresponding row values of Quantity * UnitPrice. We have cancelled transactions and returned item data in our dataset. The cancelled transactions are marked by ‘C’ at the beginning of the invoice number and placed negative number in the ‘Quantity’ column. Based on these values of ‘Quantity’ feature, we have added one extra column named ‘sell_status’ which indicates the item was sold or not.



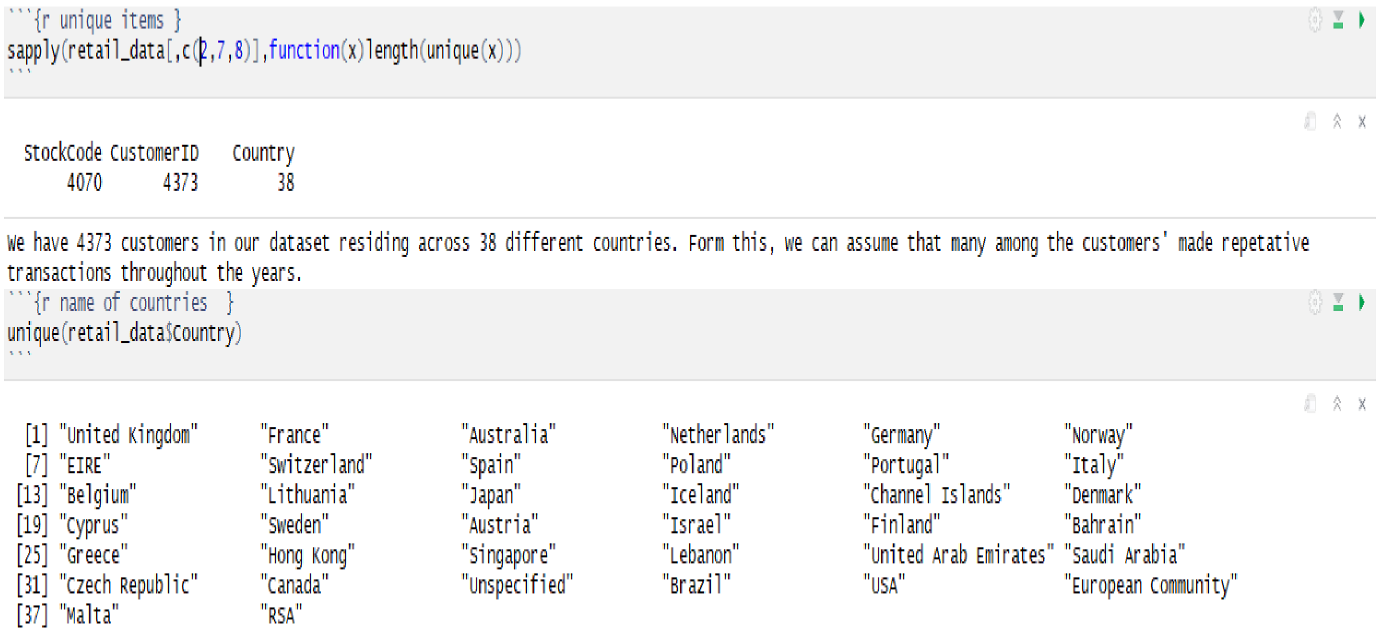
	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	sell_date	day_of_week	sell_time	month	earning	sell_status
142317	548568	M	Manual	1	2011-04-01 10:43:00	12.75	14543	United Kingdom	2011-04-01	Fri	10:43:00	04	12.75	sold
142318	548568	M	Manual	2	2011-04-01 10:43:00	10.95	14543	United Kingdom	2011-04-01	Fri	10:43:00	04	21.90	sold
142319	548568	22627	MINT KITCHEN SCALES	1	2011-04-01 10:43:00	0.76	14543	United Kingdom	2011-04-01	Fri	10:43:00	04	0.76	sold
142320	548568	22625	RED KITCHEN SCALES	2	2011-04-01 10:43:00	7.65	14543	United Kingdom	2011-04-01	Fri	10:43:00	04	15.30	sold
142321	548568	22624	IVORY KITCHEN SCALES	4	2011-04-01 10:43:00	7.65	14543	United Kingdom	2011-04-01	Fri	10:43:00	04	30.60	sold
142322	548568	22624	IVORY KITCHEN SCALES	4	2011-04-01 10:43:00	7.65	14543	United Kingdom	2011-04-01	Fri	10:43:00	04	30.60	sold
142323	548568	22207	FRYING PAN UNION FLAG	1	2011-04-01 10:43:00	4.25	14543	United Kingdom	2011-04-01	Fri	10:43:00	04	4.25	sold
142324	548568	21843	RED RETROSPOT CAKE STAND	1	2011-04-01 10:43:00	9.95	14543	United Kingdom	2011-04-01	Fri	10:43:00	04	9.95	sold
142325	548568	21781	MA CAMPAGNE CUTLERY BOX	7	2011-04-01 10:43:00	12.75	14543	United Kingdom	2011-04-01	Fri	10:43:00	04	89.25	sold
142326	548568	21534	DAIRY MAID LARGE MILK JUG	1	2011-04-01 10:43:00	4.95	14543	United Kingdom	2011-04-01	Fri	10:43:00	04	4.95	sold
142327	548568	21528	DAIRY MAID TRADITIONAL TEAPOT	2	2011-04-01 10:43:00	5.95	14543	United Kingdom	2011-04-01	Fri	10:43:00	04	11.90	sold
142328	548568	21258	VICTORIAN SEWING BOX LARGE	4	2011-04-01 10:43:00	12.75	14543	United Kingdom	2011-04-01	Fri	10:43:00	04	51.00	sold
142329	C548573	22762	CUPBOARD 3 DRAWER MA CAMPAG...	-8	2011-04-01 10:45:00	12.75	15045	United Kingdom	2011-04-01	Fri	10:45:00	04	-102.00	returned
142330	C548576	84632	DECORATIVE HANGING SHELVING UNIT	-1	2011-04-01 10:46:00	59.95	15696	United Kingdom	2011-04-01	Fri	10:46:00	04	-59.95	returned
142331	C548576	22487	WHITE WOOD GARDEN PLANT LADD...	-1	2011-04-01 10:46:00	9.95	15696	United Kingdom	2011-04-01	Fri	10:46:00	04	-9.95	returned

Showing 142,317 to 142,332 of 541,909 entries

Figure 1Dataset Organization

We have also checked for missing values and have spotted missing values in product description and customer ID column. As the missing values of these two columns will have negligible impact on forecasting, we have tried to overlook those.

Exploratory Analysis:



~~~
```{r unique items }
sapply(retail_data[,c(2,7,8)],function(x)length(unique(x)))
```
~~~

```
 StockCode CustomerID    Country 
      4070       4373         38 
```

We have 4373 customers in our dataset residing across 38 different countries. Form this, we can assume that many among the customers' made repetative transactions throughout the years.

~~~
```{r name of countries  }
unique(retail_data$Country)
```
~~~

```
 [1] "United Kingdom"       "France"               "Australia"            "Netherlands"          "Germany"              "Norway"              
 [7] "EIRE"                 "Switzerland"          "Spain"                "Poland"               "Portugal"             "Italy"               
[13] "Belgium"              "Lithuania"            "Japan"                "Iceland"              "Channel Islands"      "Denmark"             
[19] "Cyprus"               "Sweden"               "Austria"              "Israel"               "Finland"              "Bahrain"             
[25] "Greece"               "Hong Kong"            "Singapore"            "Lebanon"              "United Arab Emirates" "Saudi Arabia"        
[31] "Czech Republic"       "Canada"               "Unspecified"          "Brazil"               "USA"                  "European Community"  
[37] "Malta"                "RSA"                 
```

Figure 2 : Number of items and customers from different countries

As from figure 2, we have total 4070 unique items those were purchased by 4373 customers from 38 different countries. We might want to see that the customers from which countries performed most transactions and thus we plotted figure 3.

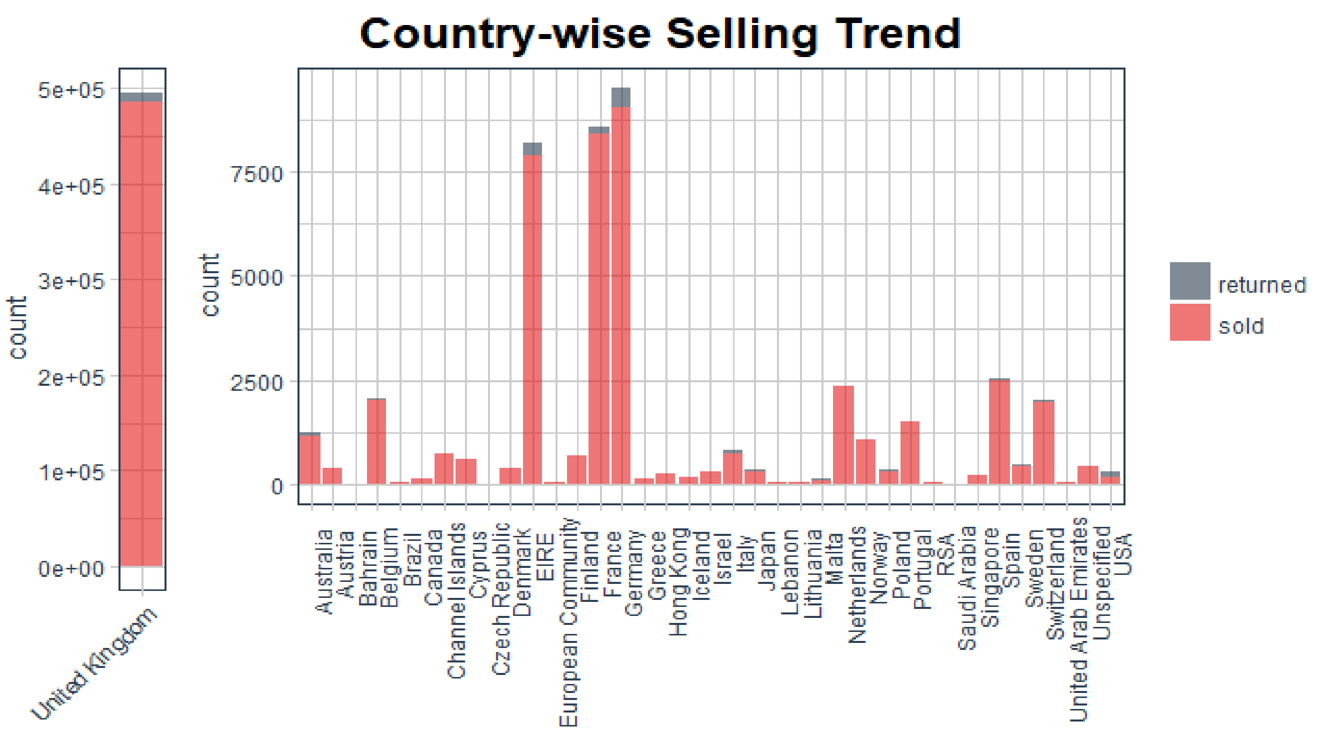


Figure 3 : Country wise selling trend

Figure 3 depicts that the largest product sold in UK, most probably because the retail company was UK based. We have also high number of sold products in Germany, France and Ireland.
As we have around 542K transactions performed by 4373 customers, surely their will be many revisiting customers, and figure 4 shows us the ration of one-time customers and revisiting customers.

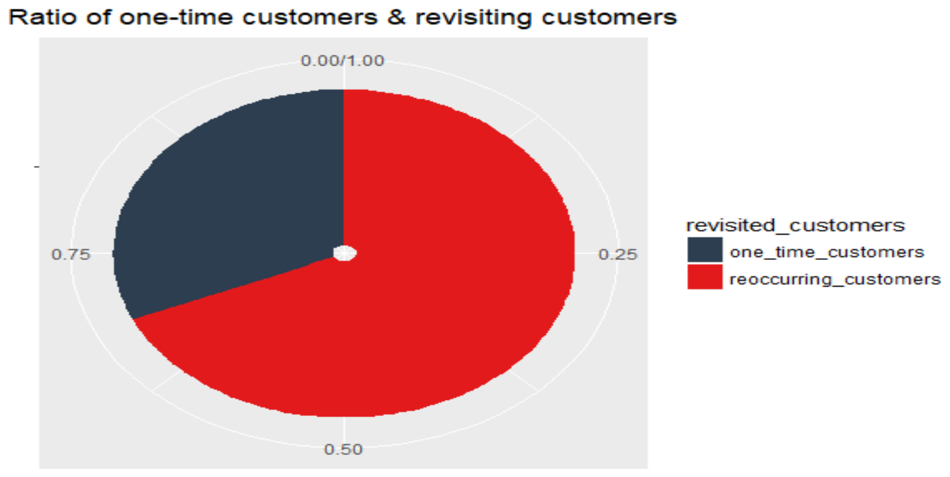


Figure 4 : Distribution of revisiting customers

Figure 5 depicted that the mean earning per customer was just above 500 GBP. The number of different items bought by each customer were approximately 30 and the mean quantity of those items were nearly 300.

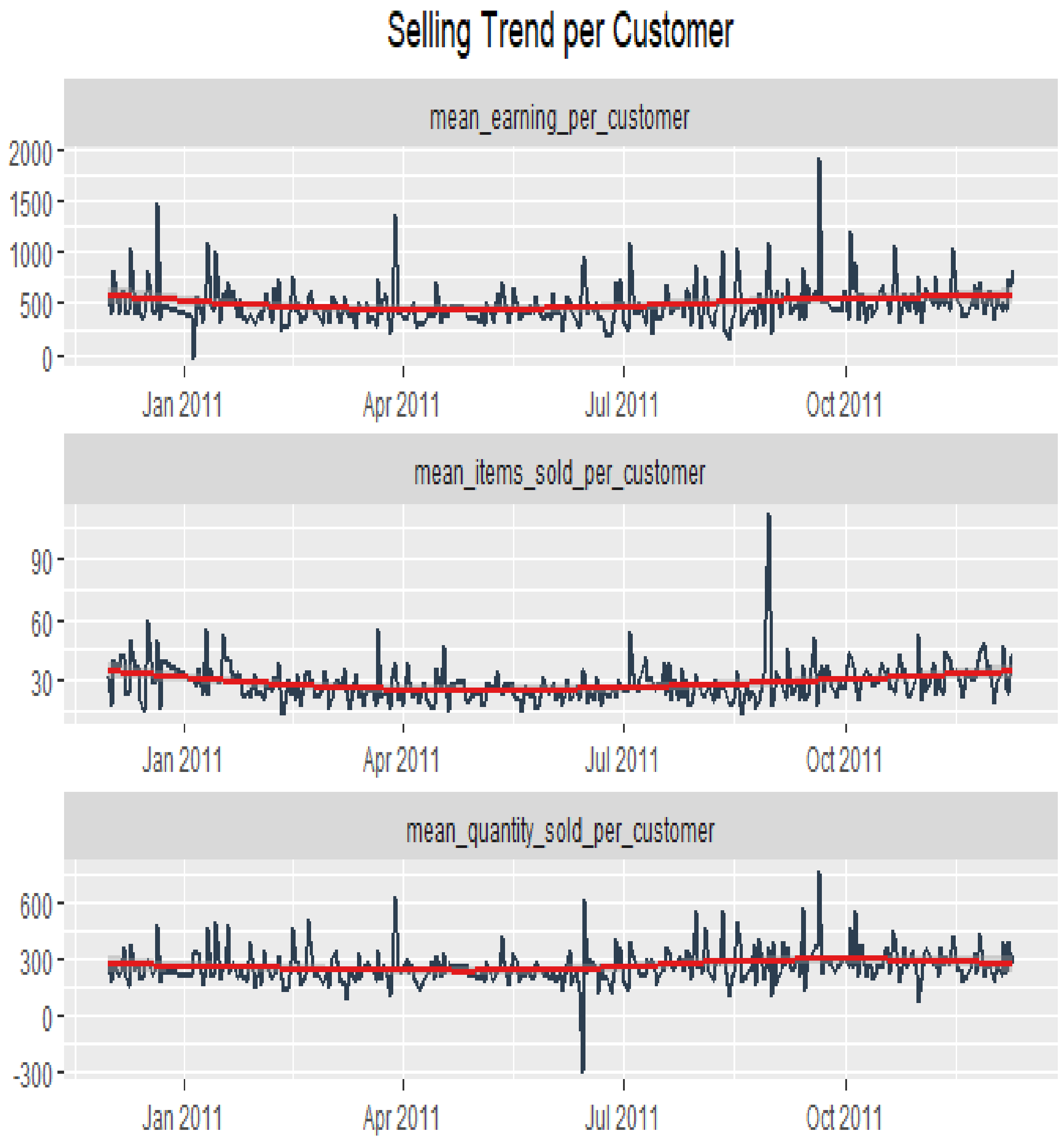


Figure 5: Selling trend overview

As we have many cancelled transactions in our dataset, we might want to see the number of sell and return over the time and the relationship among them.

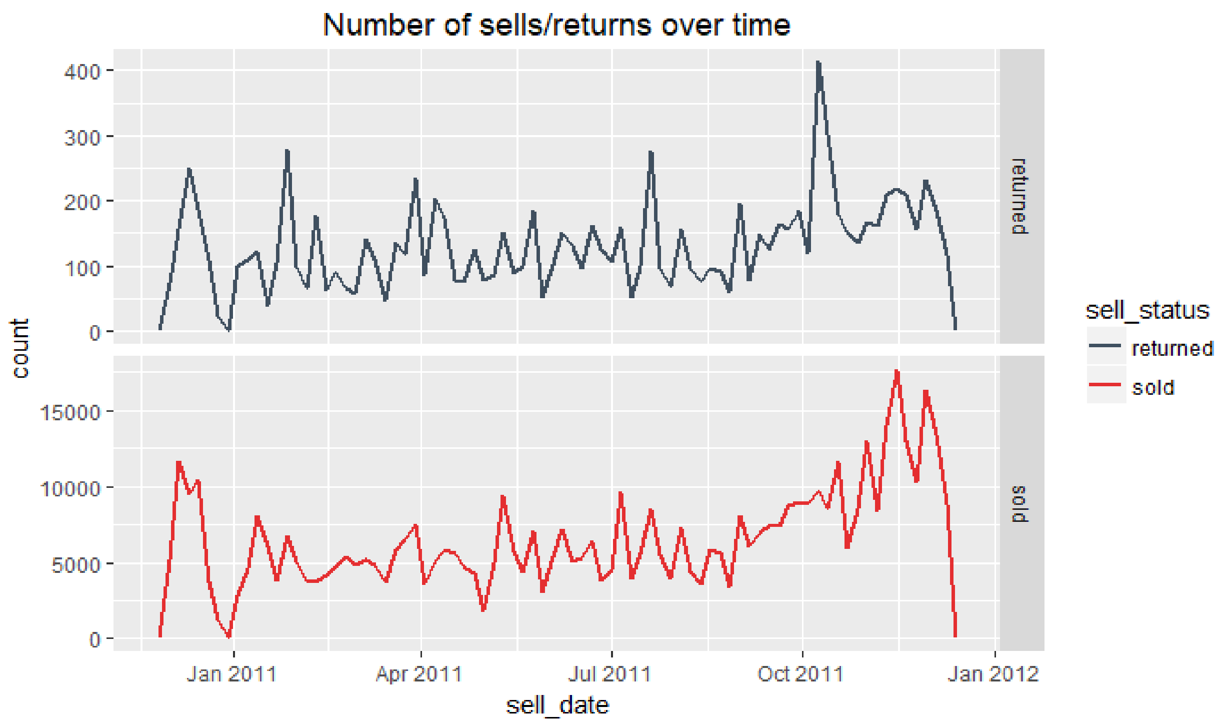


Figure 6: Number of sell/return

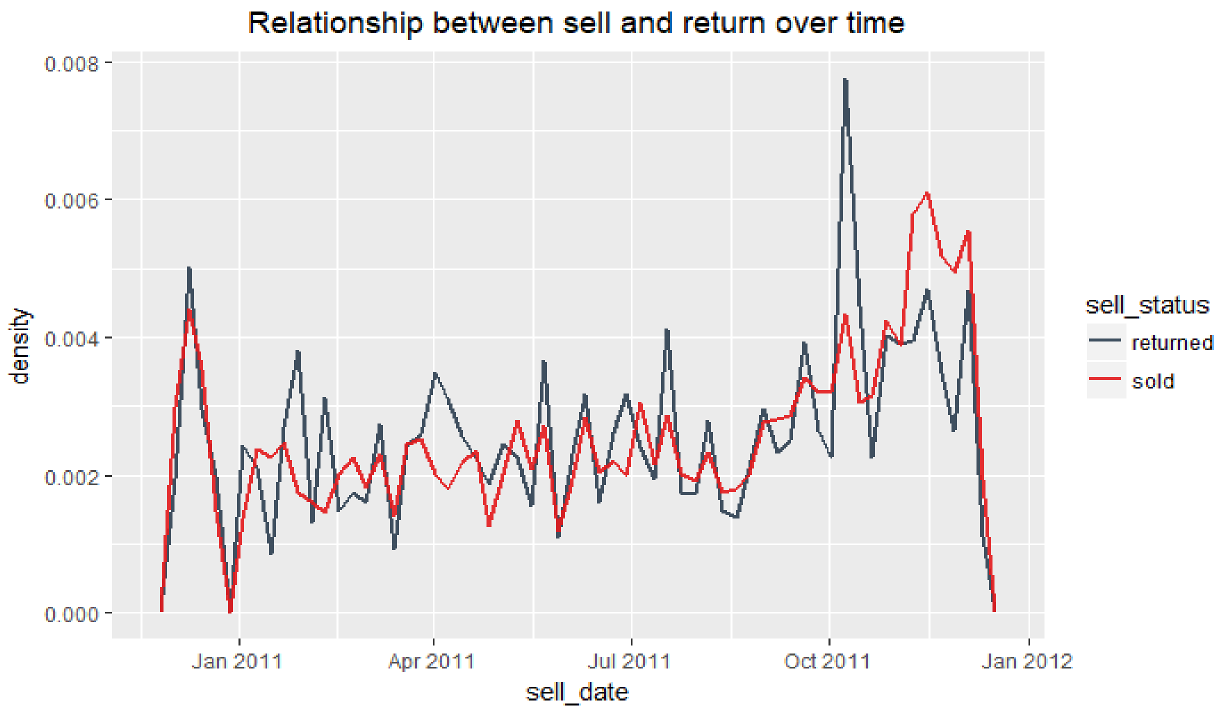


Figure 7: Relationship between sell and return of items

Figure 8 depicted that most transactions were occurred between 10 am to 5 pm. There were no transactions between 10 pm to 6 am. The highest number of transactions occurred between the end of September 2011 to the 9th of December 2011.

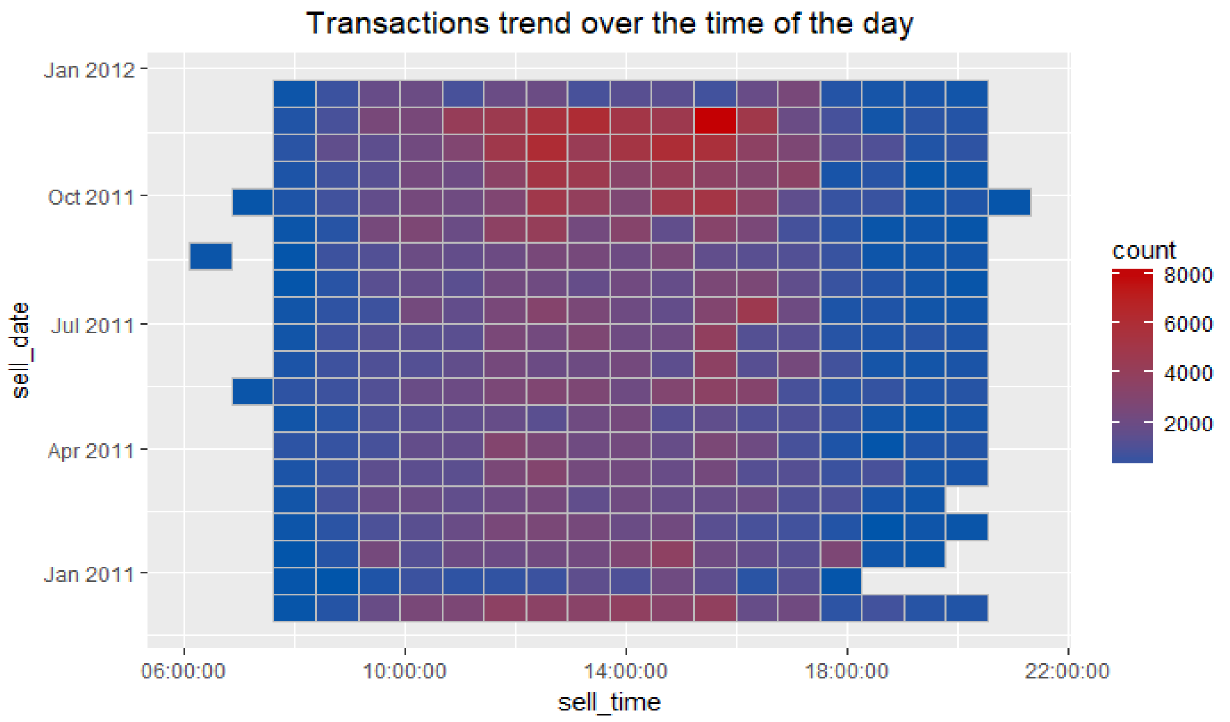


Figure 8: Transactions trend over the time of the day

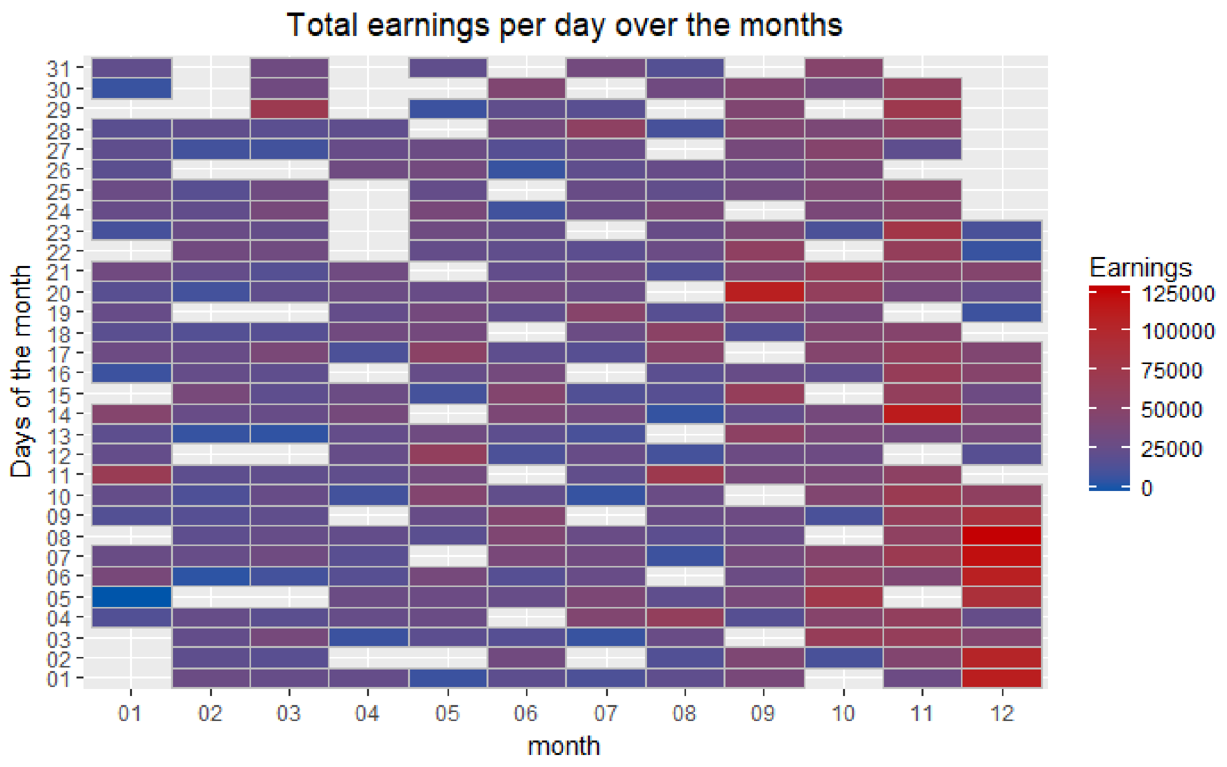


Figure 9: Earnings per day over the months

From figure 9 depicted the highest earning days over the time. The sell increased during the November and reached the to the highest pick on the first week of December 2011. Next, we would like to see the transaction history of the most sold items of our dataset, depicted in figure 10.

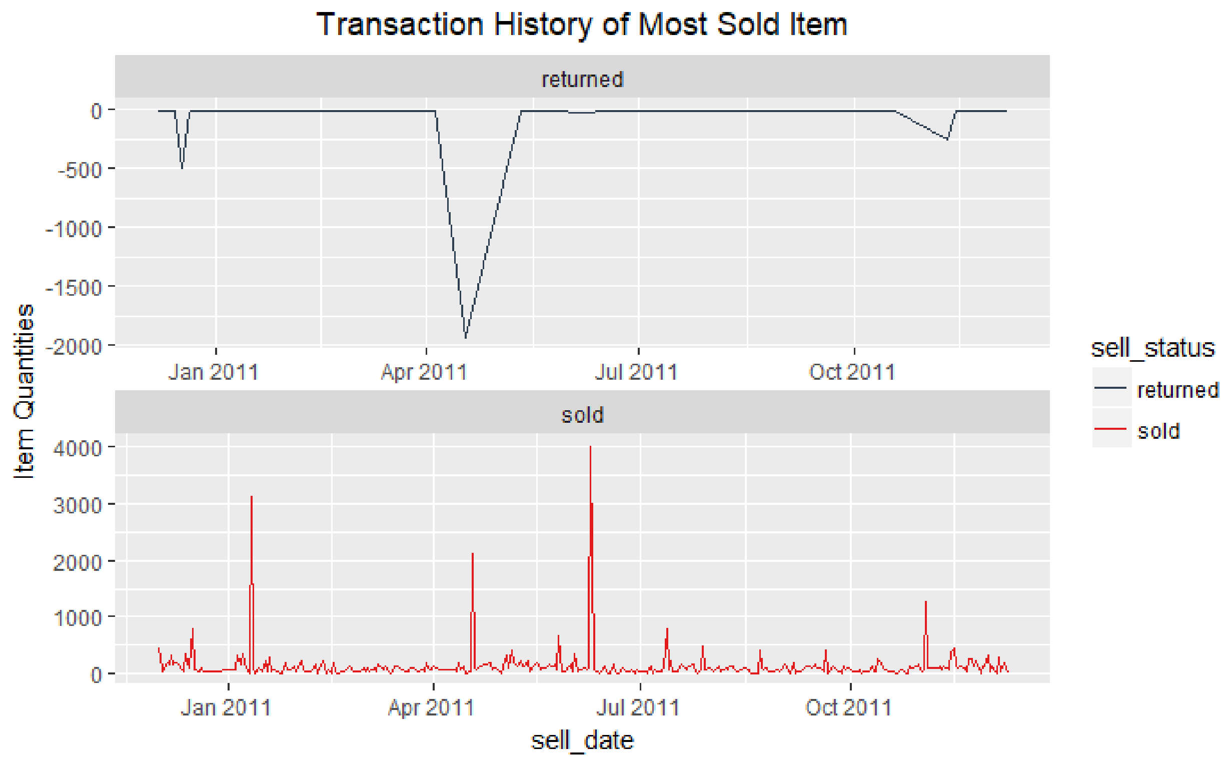


Figure 10: Transaction history of most sold item

By figure 11, we spotted that the transactions of items increased after end of July and in December the number of items were sold on average were reached approximately 1500.

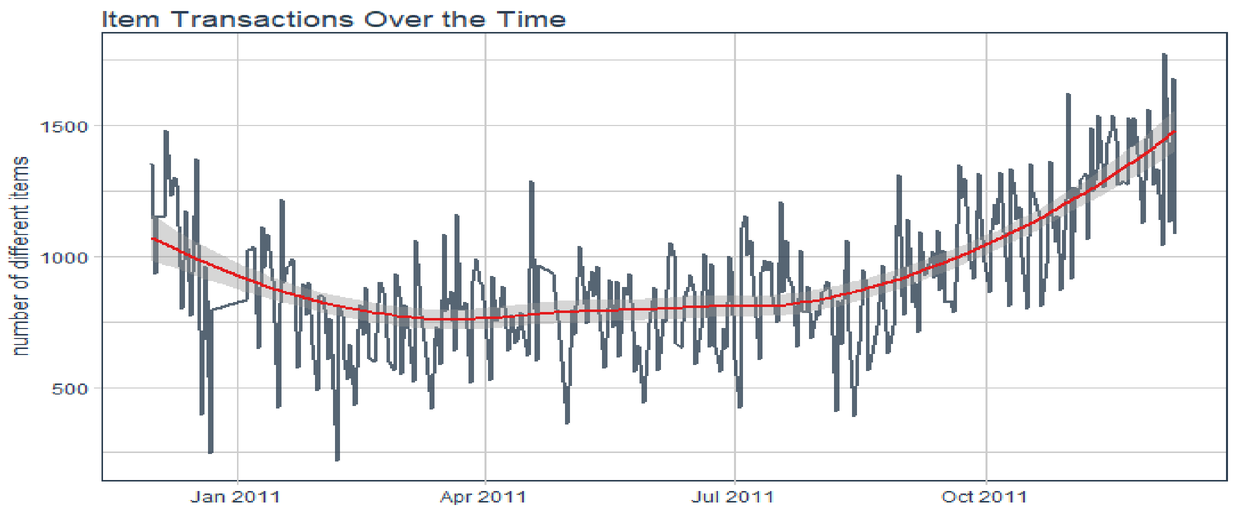


Figure 11: Transaction history of items over the time

The overall earning by sell over the time did not fall below 4000 GBP and the earning ranged between 4K to 7.5 on an average, as depicted in figure 12. There are hike and down in earning throughout the year. We have noticed that end of December 2010 there were less earning for couple of days. The sell earning remained between 500 to 12000 throughout the year with one exception at the end of the December 2011. Surprisingly, the sell earning reached up to 20K in that time.

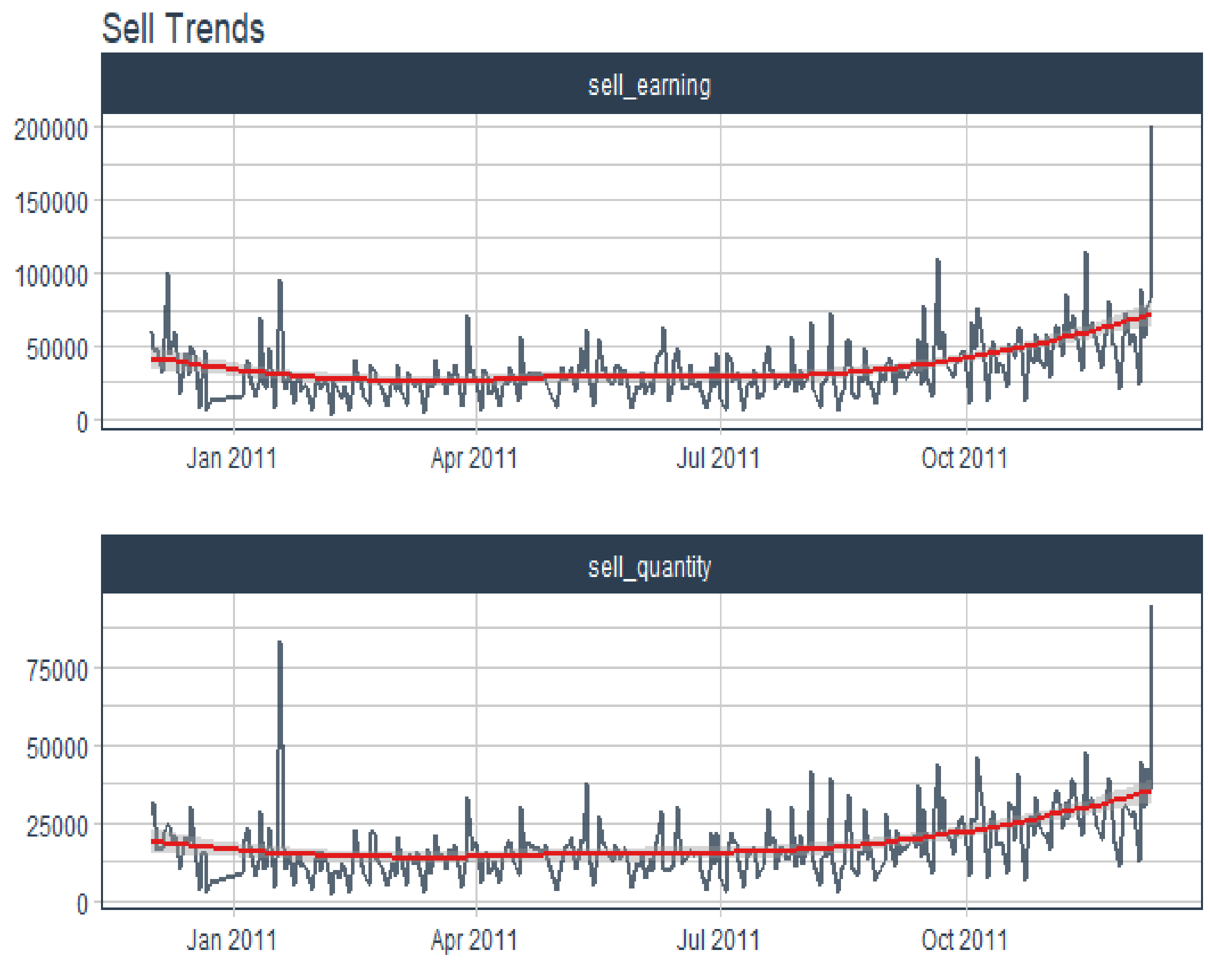


Figure 12: Overall sell trends

The overall quantity of items sold were ranged in between 150 to 50000. We have noticed 2 exceptions. One occurred at the end of January 2011 when the item quantity of sold items reached above 75000 and the second one was at the end of the graph showed in figure 12, which was the month December in 2011 where the quantity reached nearly 100K.

Time Series Analysis:

We have decided to forecast on our sell earnings by using the available data. As a part of this process, we have divided our dataset in to training and testing set. We have used data from 1st of December 2010 to 30th September 2011 to train our model. Data from 1st October 2011 to 9th December 2011 has kept for testing purpose, as figure 13 shows.

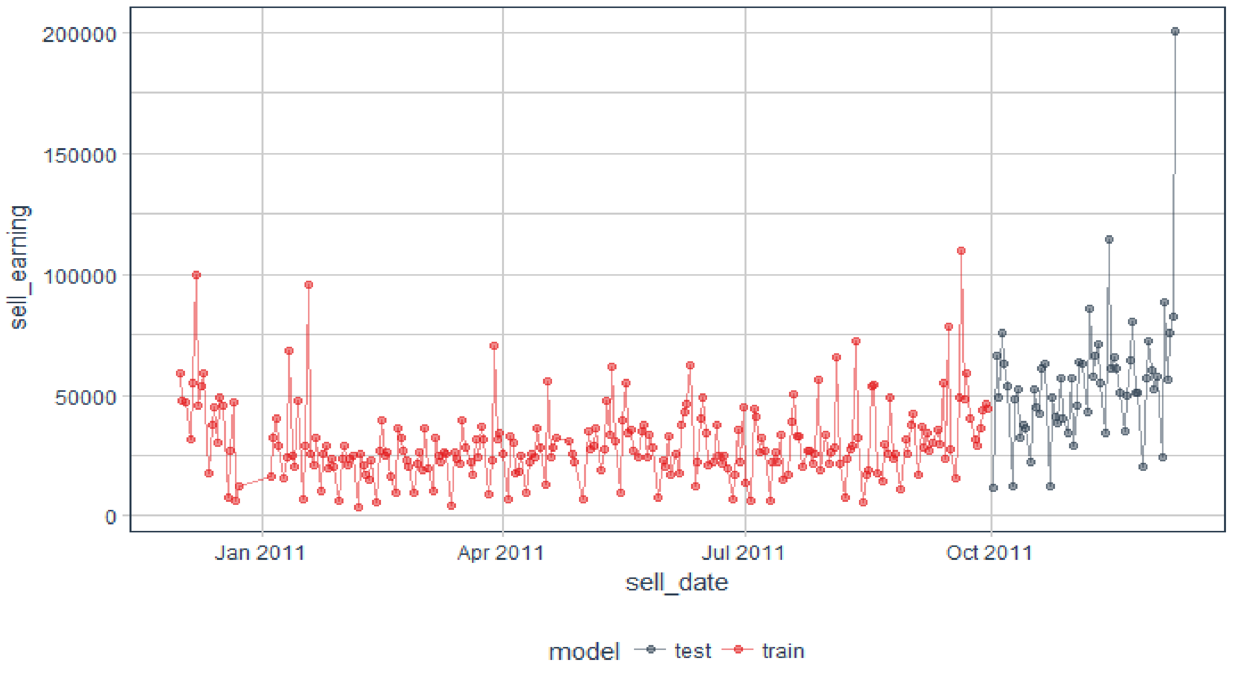


Figure 13: Training and testing data

Using the training data, we have trained our model and evaluated the performance. Our p value and standard error reflects the performance of our model in figure 14.

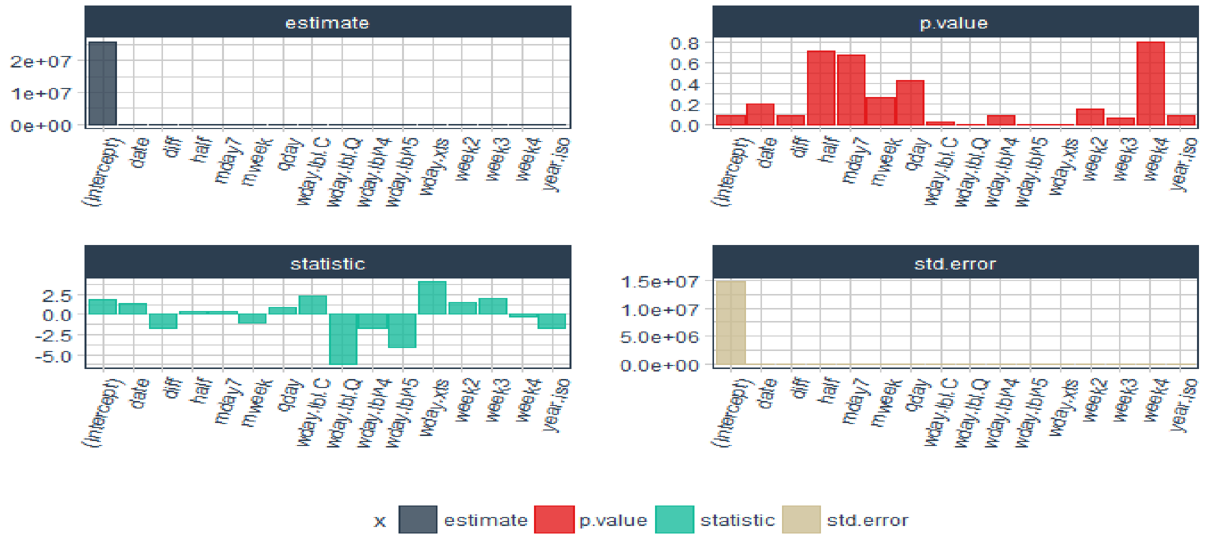


Figure 14: Model evaluation report

We have used simple linear model for this time series analysis where the response variable is 'sell_earning'. Figure 15 and figure 16 shows the working ability of our model. Both figures (15 &16) shows that the difference between expected and predicted is not deviated a lot.

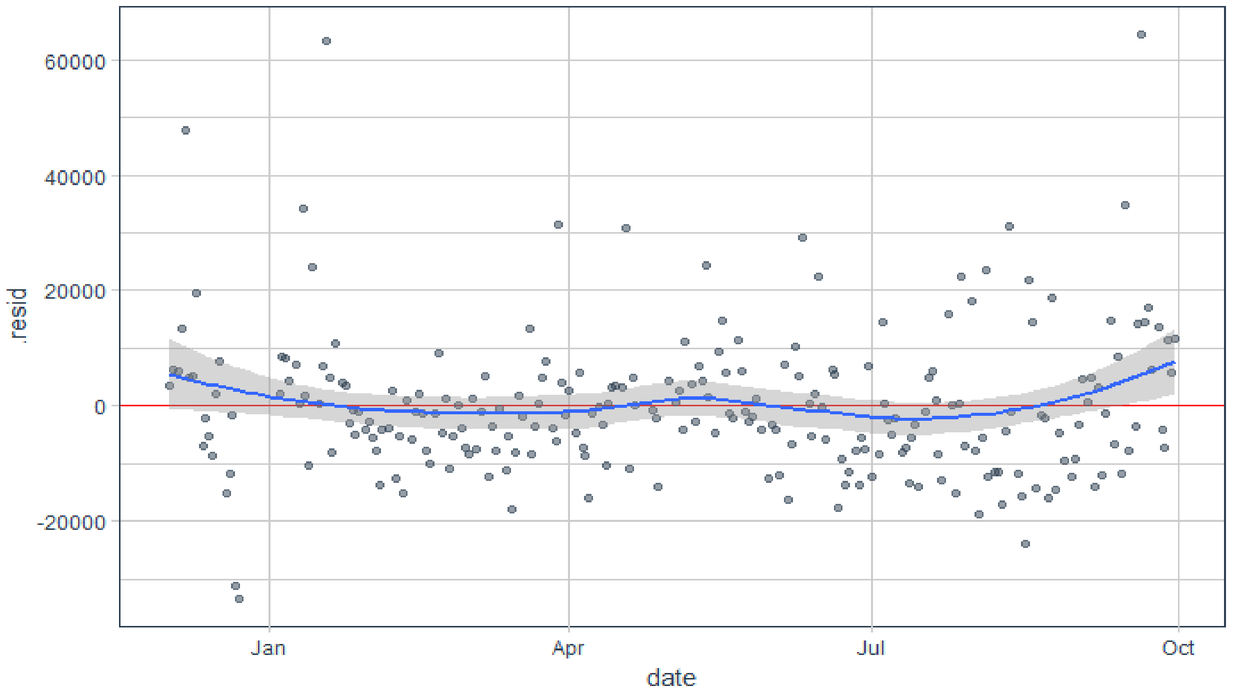


Figure 15: Residual plot of training data

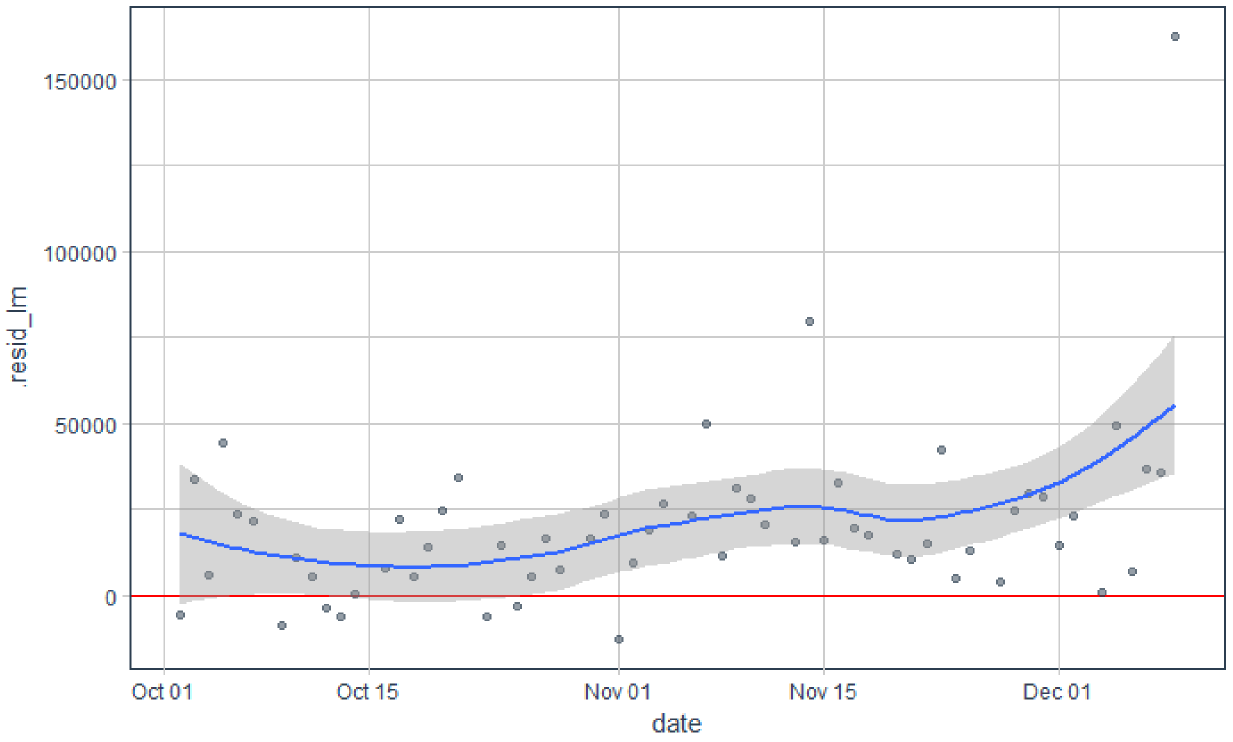


Figure 16: Residual plot of testing data

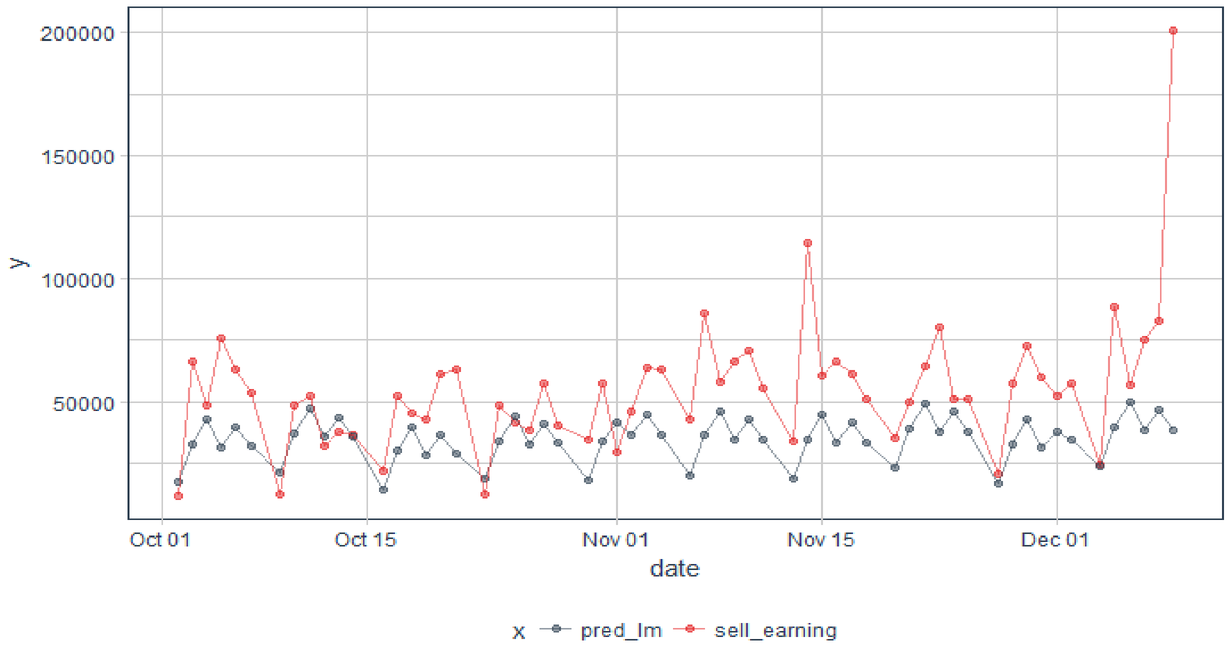


Figure 17: Actual vs predicted earning for test data

We have plotted predicted versus actual sell earning for our test data in figure 17. We have noticed that our model predicted on test data on a steady manner.

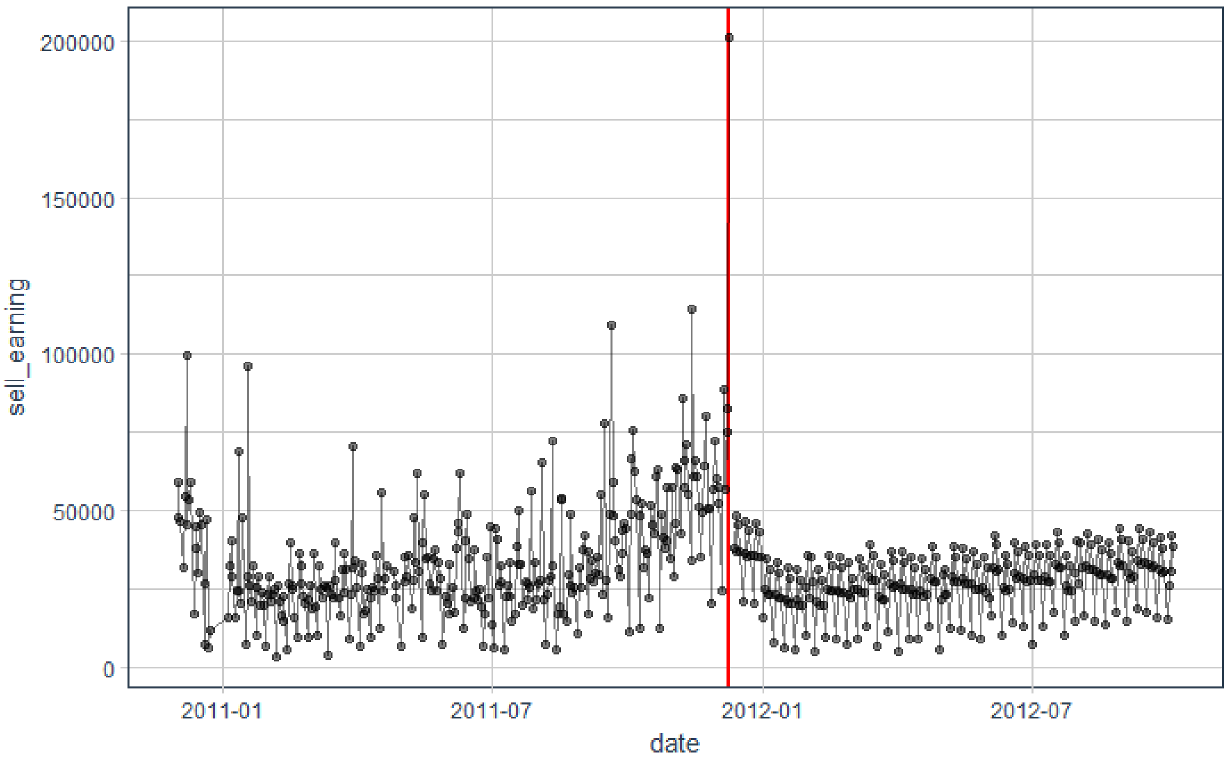


Figure 18: Model's prediction of sell earning

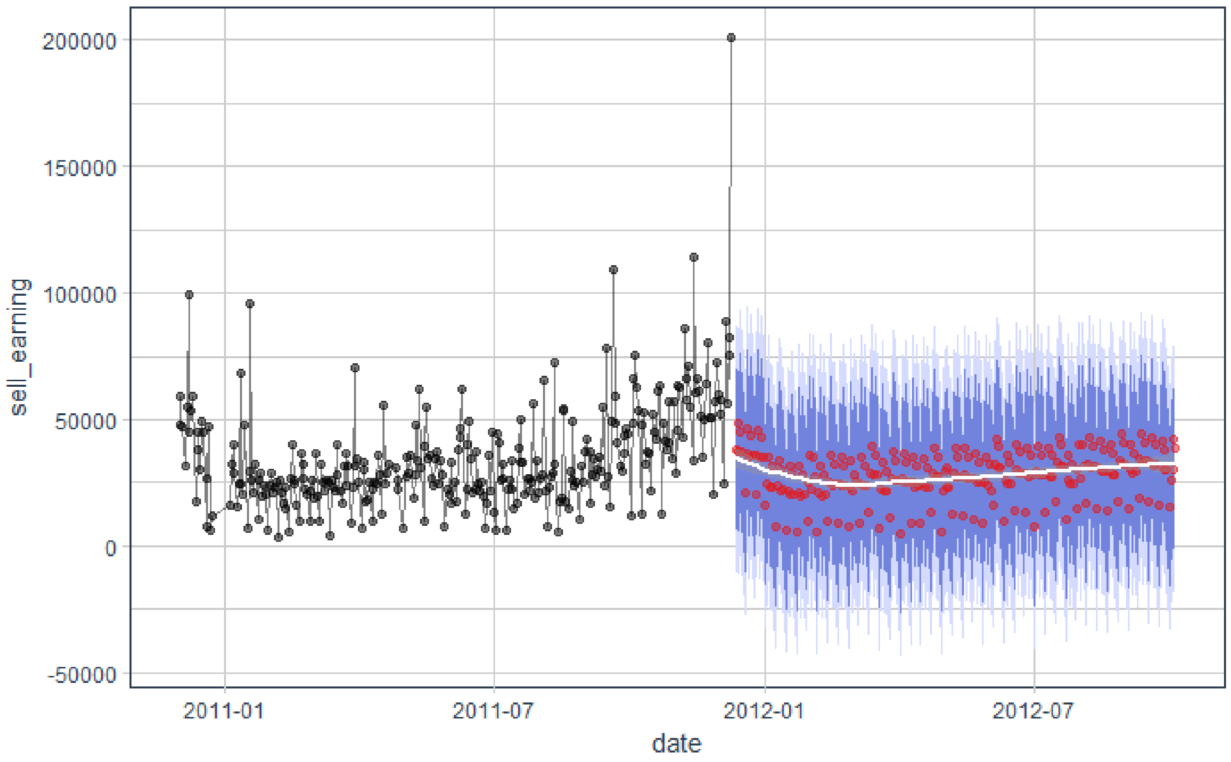


Figure 19: Forecast with confidence interval

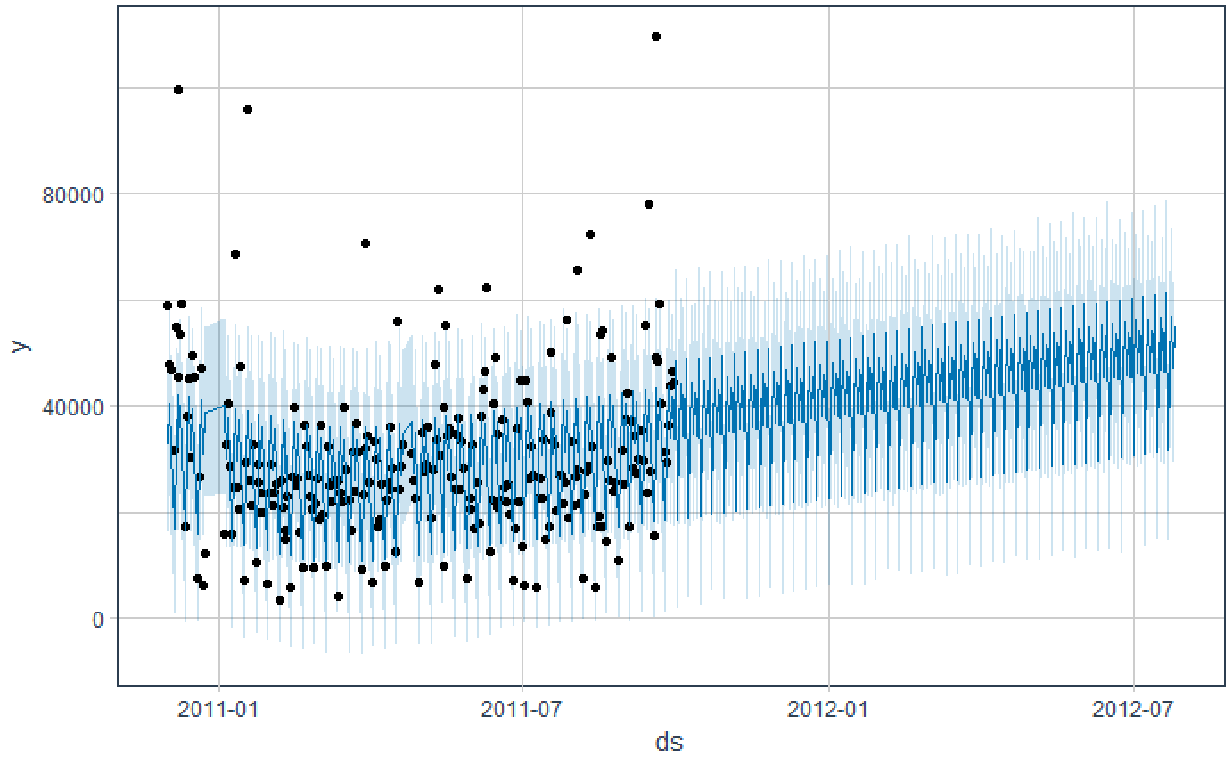


Figure 20: Forecast using prophet model

Discussion:

As in figure 18, we have seen some deviation in accuracy, we would like to consider the standard deviation of the test residual to modify our model. We have then plotted the forecast showing the confidence of intervals in figure 19. This projection shows us that the average earning might be ranged between 0 to 5000 throughout the year. We have plotted another forecasting using Facebook's Prophet Model [3] shown in figure 20. This projection shows that there might be higher trend of earning in future time slot.

Conclusion:

The idea of this project was to learn how to do forecasting using available data. During this project we have faced difficulties during data processing as we have no Saturdays in our data. Also, there were multiple day off for different occasion throughout the year on which the data were not available. Our model could perform better in forecasting if we could train the model with more data. Also, the trend of rise and fall of earning among seasons can not be cross checked due to lack of data.

References:

[1] M. Hudson, "The Pros and Cons of E-Commerce and Selling Online", The Balance, February 15, 2017.

[2] Online Retail Data Set, Machine Learning Repository, Available at, https://archive.ics.uci.edu/ml/datasets/Online+Retail

[3] https://github.com/facebook/prophet

Appendix

Retail_Data_Forcasting

Samiul_Islam_ID_500602494

April 5, 2018

R Markdown

```
#-----------------------------------------Initial Setup----------------------
------------------------
#==========================================================================
========================
#create a function to check for installed packages and install them if they
are not installed
install <- function(packages){
  new.packages <- packages[!(packages %in% installed.packages()[,
"Package"])]
  if (length(new.packages))
    install.packages(new.packages, dependencies = TRUE)
  sapply(packages, require, character.only = TRUE)
}

# usage
required.packages <- c("caret", "broom", "ggcorrplot","matrixStats",
"tidyverse", "timetk", "prophet", "tidyquant", "modelr", "gridExtra", "grid")
install(required.packages)
## Loading required package: caret
## Warning: package 'caret' was built under R version 3.4.3
## Loading required package: lattice
## Loading required package: ggplot2
## Warning: package 'ggplot2' was built under R version 3.4.3
## Loading required package: broom
## Warning: package 'broom' was built under R version 3.4.3
## Loading required package: ggcorrplot
## Warning: package 'ggcorrplot' was built under R version 3.4.4
## Loading required package: matrixStats
## Warning: package 'matrixStats' was built under R version 3.4.4
## Loading required package: tidyverse
## Warning: package 'tidyverse' was built under R version 3.4.4
## -- Attaching packages ---------------------------------------------
------------------------------------------------------------ tidyverse
1.2.1 --
## v tibble  1.4.2     v purrr   0.2.4
## v tidyr   0.7.2     v dplyr   0.7.4
## v readr   1.1.1     v stringr 1.3.0
## v tibble  1.4.2     v forcats 0.2.0
## Warning: package 'tibble' was built under R version 3.4.4
## Warning: package 'readr' was built under R version 3.4.3
## Warning: package 'dplyr' was built under R version 3.4.3
## Warning: package 'stringr' was built under R version 3.4.3
## Warning: package 'forcats' was built under R version 3.4.3
## -- Conflicts ----------------------------------------------------------------
------------------------------------------------------------------
tidyverse_conflicts() --
## x dplyr::count()  masks matrixStats::count()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
## Loading required package: timetk
## Warning: package 'timetk' was built under R version 3.4.4
## Loading required package: prophet
## Warning: package 'prophet' was built under R version 3.4.4
## Loading required package: Rcpp
## Loading required package: tidyquant
## Warning: package 'tidyquant' was built under R version 3.4.4
## Loading required package: lubridate
##
## Attaching package: 'lubridate'
## The following object is masked from 'package:base':
##
##     date
## Loading required package: PerformanceAnalytics
## Warning: package 'PerformanceAnalytics' was built under R version 3.4.3
## Loading required package: xts
## Warning: package 'xts' was built under R version 3.4.3
## Loading required package: zoo
##
## Attaching package: 'zoo'
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
## Attaching package: 'xts'
## The following objects are masked from 'package:dplyr':
##
##     first, last
##
## Attaching package: 'PerformanceAnalytics'
## The following object is masked from 'package:graphics':
##
##     legend
## Loading required package: quantmod
## Warning: package 'quantmod' was built under R version 3.4.3
## Loading required package: TTR
## Warning: package 'TTR' was built under R version 3.4.3
## Version 0.4-0 included new data defaults. See ?getSymbols.
## Loading required package: modelr
## Warning: package 'modelr' was built under R version 3.4.4
##
## Attaching package: 'modelr'
## The following object is masked from 'package:broom':
##
##     bootstrap
## Loading required package: gridExtra
## Warning: package 'gridExtra' was built under R version 3.4.3
##
## Attaching package: 'gridExtra'
## The following object is masked from 'package:dplyr':
```

```
##
##     combine
## Loading required package: grid
##       caret       broom  ggcorrplot matrixStats   tidyverse      timetk
##        TRUE        TRUE        TRUE        TRUE        TRUE        TRUE
##     prophet   tidyquant      modelr   gridExtra        grid
##        TRUE        TRUE        TRUE        TRUE        TRUE
options(na.action = na.warn)
```

Dataset Loading and Preprocessing

Data oraganization

```
write.csv(retail_data, "new_retail_data.csv")
str(retail_data)
## Classes 'tbl_df', 'tbl' and 'data.frame':    541909 obs. of  14 variables:
##  $ InvoiceNo  : chr  "536365" "536365" "536365" "536365" ...
##  $ StockCode  : chr  "85123A" "71053" "84406B" "84029G" ...
##  $ Description: chr  "WHITE HANGING HEART T-LIGHT HOLDER" "WHITE METAL
LANTERN" "CREAM CUPID HEARTS COAT HANGER" "KNITTED UNION FLAG HOT WATER
BOTTLE" ...
##  $ Quantity   : int  6 6 8 6 6 2 6 6 6 32 ...
##  $ InvoiceDate: POSIXct, format: "2010-12-01 08:26:00" "2010-12-01
08:26:00" ...
##  $ UnitPrice  : num  2.55 3.39 2.75 3.39 3.39 7.65 4.25 1.85 1.85 1.69 ...
##  $ CustomerID : int  17850 17850 17850 17850 17850 17850 17850 17850 17850
13047 ...
##  $ Country    : chr  "United Kingdom" "United Kingdom" "United Kingdom"
"United Kingdom" ...
##  $ sell_date  : Date, format: "2010-12-01" "2010-12-01" ...
##  $ day_of_week: Ord.factor w/ 7 levels "Sun"<"Mon"<"Tue"<..: 4 4 4 4 4 4 4
4 4 4 ...
##  $ sell_time  :Classes 'hms', 'difftime'  atomic [1:541909] 30360 30360
30360 30360 30360 ...
##   .. ..- attr(*, "units")= chr "secs"
##  $ month      : chr  "12" "12" "12" "12" ...
##  $ earning    : num  15.3 20.3 22 20.3 20.3 ...
##  $ sell_status: chr  "sold" "sold" "sold" "sold" ...
colSums(is.na(retail_data))
##   InvoiceNo   StockCode Description    Quantity InvoiceDate   UnitPrice
##           0           0        1454           0           0           0
##  CustomerID     Country   sell_date day_of_week   sell_time       month
##      135080           0           0           0           0           0
##     earning sell_status
##           0           0
```

As we can see we have some missing values in our dataset in Description and customer ID column. Though, product description and cutomer IDs will not have an impact on our current analysis, we can keep it as it is.

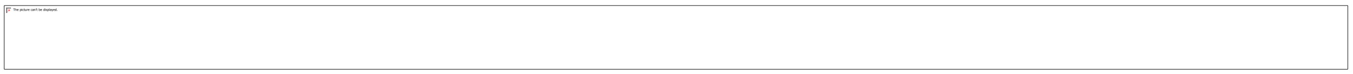
Exploratory Data Analysis

```
sapply(retail_data[,c(2,7,8)],function(x)length(unique(x)))
##   StockCode CustomerID    Country
##        4070       4373         38
```

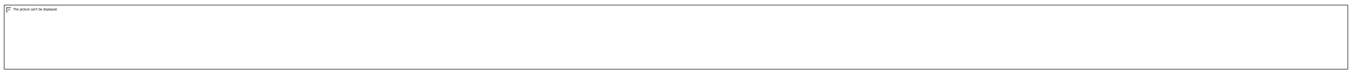
We have 4373 customers in our dataset residing across 38 different countries. Form this, we can assume that many among the customers' made repetative transactions throughout the years.

```
unique(retail_data$Country)
##  [1] "United Kingdom"       "France"               "Australia"
##  [4] "Netherlands"          "Germany"              "Norway"
##  [7] "EIRE"                 "Switzerland"          "Spain"
## [10] "Poland"               "Portugal"             "Italy"
## [13] "Belgium"              "Lithuania"            "Japan"
## [16] "Iceland"              "Channel Islands"      "Denmark"
## [19] "Cyprus"               "Sweden"               "Austria"
## [22] "Israel"               "Finland"              "Bahrain"
## [25] "Greece"               "Hong Kong"            "Singapore"
## [28] "Lebanon"              "United Arab Emirates" "Saudi Arabia"
## [31] "Czech Republic"       "Canada"               "Unspecified"
## [34] "Brazil"               "USA"                  "European Community"
## [37] "Malta"                "RSA"
plot1 <- retail_data %>%
  filter(Country == "United Kingdom") %>%
  ggplot(aes(x = Country, fill = sell_status)) +
    geom_bar(alpha = .6) +
    scale_fill_tq(values = palette_dark()) +
    theme_tq() +
    theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1)) +
    guides(fill = FALSE) +
    labs(x = "")

plot2 <- retail_data %>%
  filter(Country != "United Kingdom") %>%
  ggplot(aes(x = Country, fill = sell_status)) +
    geom_bar(alpha = .6) +
    scale_fill_tq(values = palette_green()) +
    theme_tq() +
    theme(legend.position = "right") +
    theme(axis.text.x = element_text(angle = 90, vjust = 1, hjust = .9)) +
    labs(x = "",fill = "")
title <- textGrob("Country-wise Selling Trend", gp = gpar(fontface = "bold",
cex = 1.5))
grid.arrange(plot1, plot2, top=title ,widths = c(0.2, 1.2))
```

```
retail_data %>%
  ggplot(aes(x = sell_date, color = sell_status)) + ggtitle("Number of
sells/returns over time") +
  theme(plot.title = element_text(hjust = 0.5))+
    facet_grid(sell_status ~ ., scales = "free") +
    geom_freqpoly(bins = 80, size = 1, alpha = 0.9) +
    scale_color_tq(values = palette_dark())
```

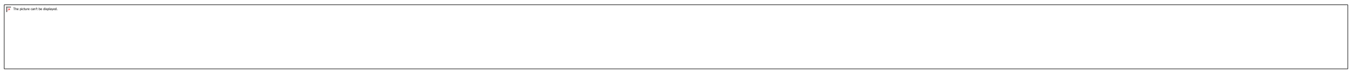


```
retail_data %>%
  ggplot(aes(x = sell_date, y = ..density.., color = sell_status)) +
ggtitle("Relationship between sell and return over time") +
  theme(plot.title = element_text(hjust = 0.5))+
    geom_freqpoly(size = 1, alpha = 0.9, bins = 60) +
    scale_color_manual(values = palette_light())
```



# Purchases over the time of the day

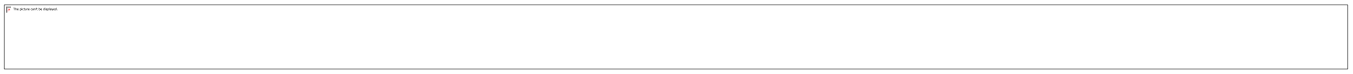
```
retail_data %>%
  ggplot(aes(x = sell_time, y = sell_date)) + ggtitle("Transactions trend
over the time of the day") +   theme(plot.title = element_text(hjust = 0.5))+
    stat_bin_2d(alpha = 1, bins = 19, color = "grey") +
    scale_fill_gradientn(colours = c(palette_green()[[1]],
palette_dark()[[2]]))
```



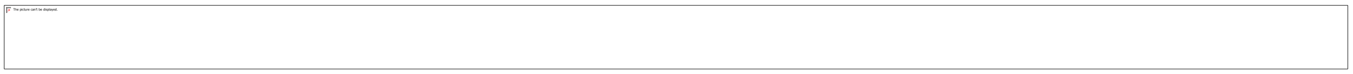
Earnings per day over the months of the year

```
retail_data %>%
  mutate(day = format(InvoiceDate, "%d")) %>%
  group_by(month, day) %>%
  summarise(Earnings = sum(earning)) %>%
  ggplot(aes(x = month, y = day, fill = Earnings)) + ggtitle("Total earnings
per day over the months") +   theme(plot.title = element_text(hjust = 0.5))+
ylab("Days of the month")+
    geom_tile(alpha = 1, color = "grey") +
    scale_fill_gradientn(colours = c(palette_green()[[1]],
palette_dark()[[2]])) +
    theme(legend.position = "right")
```

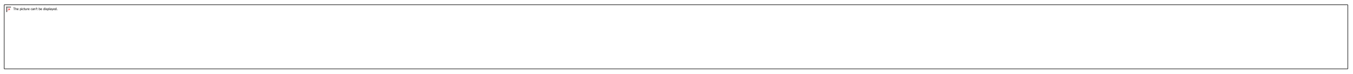
```
retail_data %>%
  group_by(StockCode, Description) %>%
  summarise(Total_Quantity = sum(Quantity)) %>%
  arrange(-Total_Quantity) %>% head()
## # A tibble: 6 x 3
## # Groups:   StockCode [6]
##   StockCode Description                        Total_Quantity
##   <chr>     <chr>                                       <int>
## 1 84077     WORLD WAR 2 GLIDERS ASSTD DESIGNS           53847
## 2 85099B    JUMBO BAG RED RETROSPOT                     47363
## 3 84879     ASSORTED COLOUR BIRD ORNAMENT               36381
## 4 22197     POPCORN HOLDER                              36334
## 5 21212     PACK OF 72 RETROSPOT CAKE CASES             36039
## 6 85123A    WHITE HANGING HEART T-LIGHT HOLDER          35025
p1 <- retail_data %>%
  group_by(StockCode, Description) %>%
  summarise(Total_Quantity = sum(Quantity)) %>%
  ggplot(aes(x = Total_Quantity)) +
    geom_density(fill = palette_light()[[1]], alpha = 0.8) +
    theme_tq()

p2 <- retail_data %>%
  group_by(StockCode, Description) %>%
  summarise(Total_Quantity = sum(Quantity)) %>%
  filter(Total_Quantity > 1) %>%
  ggplot(aes(x = Total_Quantity)) +
    geom_density(fill = palette_light()[[1]], alpha = 0.8) +
    theme_tq()

p3 <- retail_data %>%
  group_by(StockCode, Description) %>%
  summarise(Total_Quantity = sum(Quantity)) %>%
  filter(Total_Quantity > 10000) %>%
  ggplot(aes(x = Total_Quantity)) +
    geom_density(fill = palette_light()[[1]], alpha = 0.8) +
    theme_tq()
title1 <- textGrob("Selling trend of product by quantity", gp = gpar(fontface
= "bold", cex = 1.5))
grid.arrange(p1, p2, p3, ncol = 3,top=title1 ,widths = c(0.6, 0.5, 0.7))
```



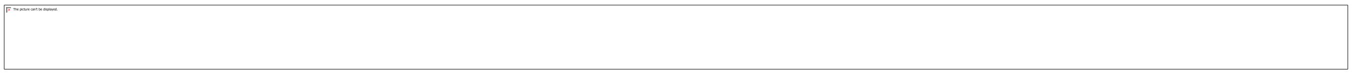
```
top_sold_items <- retail_data %>%
  group_by(sell_date, StockCode, Description) %>%
  summarise(sum = sum(Quantity)) %>%
  group_by(StockCode, Description)%>%
  summarise(n = n()) %>%
  arrange(-n)

top_items_transactions <- retail_data %>%
  filter(StockCode %in% top_sold_items$StockCode[1:5]) %>%
  group_by(sell_date, StockCode) %>%
  summarise(sum = sum(Quantity)) %>%
  spread(key = StockCode, value = sum)

retail_data %>%
  filter(StockCode %in% top_sold_items$StockCode[1:5]) %>%
  group_by(sell_date, StockCode, Description) %>%
  summarise(sum = sum(Quantity)) %>%
  ggplot(aes(x = sell_date, y = sum)) +
    facet_wrap(~ StockCode, ncol = 1, scales = "free") +
    geom_line(color = palette_light()[[1]], size = 1, alpha = 0.8) +
    theme_tq() +
    labs(Title= "Top 5 Items' Selling Trends", x = "",
         y = "Quantity of Sold Items")
```



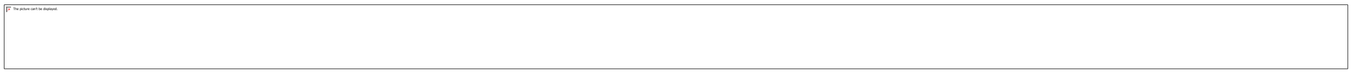
```
retail_data %>%
  filter(StockCode == "85123A") %>%
  group_by(sell_date, sell_status) %>%
  summarise(Most_sold_item = sum(Quantity)) %>%
  ggplot(aes(x = sell_date, y = Most_sold_item, color = sell_status)) +
ggtitle("Transaction History of Most Sold Item") +   theme(plot.title =
element_text(hjust = 0.5))+ ylab("Item Quantities")+
    facet_wrap(~ sell_status, ncol = 1, scales = "free") +
    geom_line(size = .4, alpha = 1) +
    scale_color_tq(values = palette_dark())
```



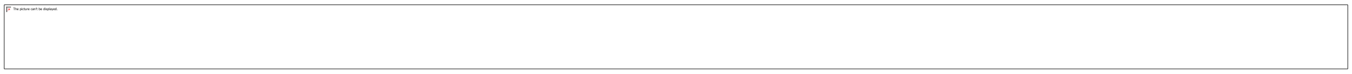
```
multiple_transactions <- retail_data %>%
  group_by(sell_date, CustomerID) %>%
  summarise(sum = sum(Quantity)) %>%
  group_by(CustomerID) %>%
  summarise(n = n()) %>%
  mutate(revisited_customers = ifelse(n > 1, "reoccurring_customers",
"one_time_customers"))

length(which(multiple_transactions$revisited_customers ==
"reoccurring_customers"))
## [1] 2992
revisited_customers_sell_date <- left_join(retail_data,
multiple_transactions, by = "CustomerID") %>%
  distinct(sell_date, CustomerID, revisited_customers) %>%
  group_by(sell_date, revisited_customers) %>%
  summarise(n = n()) %>%
  spread(key = revisited_customers, value = n)
multiple_transactions %>%
  group_by(revisited_customers) %>%
  summarise(n = n()) %>%
  mutate(prop = n / sum(n)) %>%
  ggplot(aes(x = "", y = prop, fill = revisited_customers)) +ggtitle("Ratio
of one-time customers & revisiting customers") +   theme(plot.title =
element_text(hjust = 0.5))+ ylab("")+ xlab("")+
    geom_bar(stat = "identity", alpha = 1) +
    coord_polar("y", start = 0) +
    scale_fill_tq(values = palette_dark())
```



```
purchases <- retail_data %>%
  group_by(sell_date, CustomerID) %>%
  summarise(n = n(),
            Tot_item = sum(Quantity),
            Tot_earning = sum(earning)) %>%
  group_by(sell_date) %>%
  summarise(mean_earning_per_customer = mean(Tot_earning),
            mean_quantity_sold_per_customer = mean(Tot_item),
            mean_items_sold_per_customer = mean(n))
purchases %>%
  gather(x, y, mean_earning_per_customer:mean_items_sold_per_customer) %>%
  ggplot(aes(x = sell_date, y = y)) + ggtitle("Selling Trend per Customer") +
theme(plot.title = element_text(hjust = 0.5))+ ylab("")+ xlab("")+
    facet_wrap(~ x, ncol = 1, scales = "free") +
    geom_line(color = palette_light()[[1]], size = 1, alpha = 1) +
    geom_smooth(color = palette_light()[[2]], method = 'loess')
```

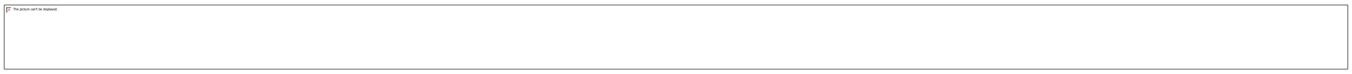


```
returned_items <- retail_data %>%
  group_by(sell_date, sell_status) %>%
  summarise(total_quantity = sum(Quantity)) %>%
  spread(key = sell_status, value = total_quantity)

returned_items%>%
  gather(x, y, sold:returned) %>%
  ggplot(aes(x = sell_date, y = y, color = x)) +
    geom_line(size = 1, alpha = 0.8) +
    scale_color_manual(values = palette_light()) +
    theme_tq() +
    labs(x = "",
         y = "quantity of items",
         title = "Purchase and Return Over the Time")
```

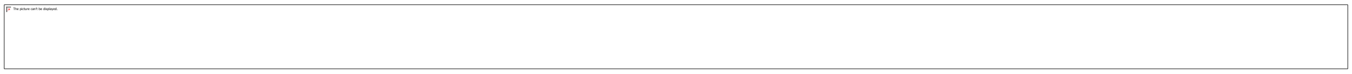


#How many different items are purchased/returned per day?

```
item_transactions <- retail_data %>%
  group_by(sell_date, StockCode) %>%
  summarise(n = n()) %>%
  group_by(sell_date) %>%
  summarise(item_transactions = n())
item_transactions %>%
  ggplot(aes(x = sell_date, y = item_transactions)) +
    geom_line(color = palette_light()[[1]], size = 1, alpha = 0.8) +
    geom_smooth(color = palette_light()[[2]], method = 'loess') +
    theme_tq() +
    labs(title= "Item Transactions Over the Time" ,x = "",
         y = "number of different items",
         color = "")
```

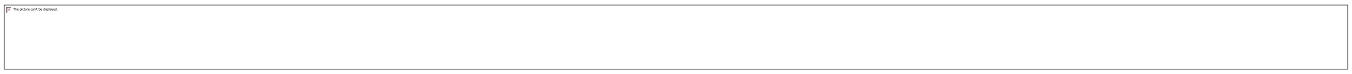


```
net_earning <- retail_data %>%
  group_by(sell_date) %>%
  summarise(sum_earning = sum(earning),
            mean_earning = mean(earning),
            sum_quantity = sum(Quantity),
            mean_quantity = mean(Quantity))
net_earning %>%
  gather(x, y, sum_earning:mean_quantity) %>%
  ggplot(aes(x = sell_date, y = y)) +
    facet_wrap(~ x, ncol = 1, scales = "free") +
    geom_line(color = palette_light()[[1]], size = 1, alpha = 0.8) +
    geom_smooth(color = palette_light()[[2]], method = 'loess') +
    theme_tq() +
    labs(title="Mean of Earning and Itme Quantity" ,x = "",
          y = "")
```

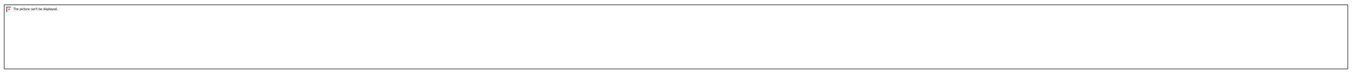


# Profit from purchases and returns

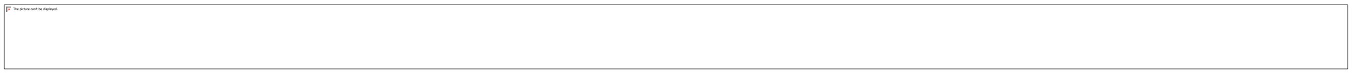
```
sold_items <- retail_data %>%
  filter(earning > 0) %>%
  group_by(sell_date) %>%
  summarise(sell_earning = sum(earning),
            #sell_earning_mean = mean(earning),
            sell_quantity = sum(Quantity))
            #sell_quantity_mean = mean(Quantity))
sold_items %>%
  gather(x, y, sell_earning:sell_quantity) %>%
  ggplot(aes(x = sell_date, y = y)) +
    facet_wrap(~ x, ncol = 1, scales = "free") +
    geom_line(color = palette_light()[[1]], size = 1, alpha = 0.8) +
    geom_smooth(color = palette_light()[[2]], method = 'loess') +
    theme_tq() +
    labs(title= "Sell Trends",x = "",
         y = "")
```



```
return_items <- retail_data %>%
  filter(earning < 0) %>%
  group_by(sell_date) %>%
  summarise(return_items_sum_price = sum(earning),
            mean_return_items_price = mean(earning),
            quantity_return_items = sum(Quantity),
            mean_quantity_return_items = mean(Quantity))
return_items %>%
  gather(x, y, return_items_sum_price:mean_quantity_return_items) %>%
  ggplot(aes(x = sell_date, y = y)) +
    facet_wrap(~ x, ncol = 1, scales = "free") +
    geom_line(color = palette_light()[[1]], size = 1, alpha = 0.8) +
    theme_tq() +
    labs(x = "",
         y = "")
```

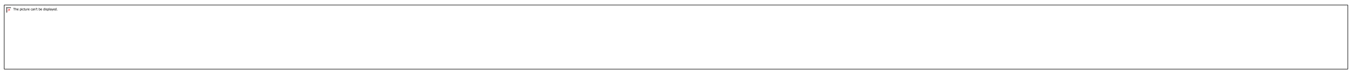


```
unit_quant <- distinct(select(retail_data, sell_date, StockCode, UnitPrice))
%>%
  mutate(unit_quant = paste(sell_date, StockCode, sep = "_")) %>%
  select(unit_quant, UnitPrice)

mean_unit_price <- retail_data %>%
  filter(sell_status == "sold") %>%
  group_by(sell_date, StockCode) %>%
  summarise(n = n()) %>%
  mutate(unit_quant = paste(sell_date, StockCode, sep = "_")) %>%
  left_join(unit_quant, by = "unit_quant") %>%
  group_by(sell_date, StockCode) %>%
  summarise(mean = mean(UnitPrice)) %>%
  group_by(sell_date) %>%
  summarise(mean_unit_price = mean(mean))
mean_unit_price %>%
  ggplot(aes(x = sell_date, y = mean_unit_price)) +
    geom_line(color = palette_light()[[1]], size = 1, alpha = 0.8) +
    theme_tq() +
    labs(x = "",
         y = "mean unit price of sold items")
```



Forcasting

```
transaction_per_day <- distinct(select(retail_data, sell_date, day_of_week,
month)) %>%
  left_join(net_earning, by = "sell_date") %>%
  left_join(mean_unit_price, by = "sell_date") %>%
  left_join(sold_items, by = "sell_date") %>%
  left_join(return_items, by = "sell_date") %>%
  left_join(purchases, by = "sell_date") %>%
  left_join(revisited_customers_sell_date, by = "sell_date") %>%
  left_join(returned_items, by = "sell_date") %>%
  left_join(item_transactions, by = "sell_date") %>%
  left_join(top_items_transactions, by = "sell_date") %>%
  mutate(diff_sum_earning = sell_earning - lag(sell_earning),
         season = ifelse(month %in% c("03", "04", "05"), "spring",
                         ifelse(month %in% c("06", "07", "08"), "summer",
                                ifelse(month %in% c("09", "10", "11"),
"fall", "winter"))))
transaction_per_day <- transaction_per_day %>%
  mutate(model = ifelse(sell_date <= "2011-10-01", "train", "test"))

colnames(transaction_per_day)[grep("^[0-9]+", colnames(transaction_per_day))]
<- paste0("P_", colnames(transaction_per_day)[grep("^[0-9]+",
colnames(transaction_per_day))])
transaction_per_day %>%
  ggplot(aes(x = sell_date, y = sell_earning, color = model)) +
    geom_point(alpha = 0.5) +
    geom_line(alpha = 0.5) +
    scale_color_manual(values = palette_light()) +
    theme_tq()
```



```
augmented_transactions <- transaction_per_day %>%
  rename(date = sell_date) %>%
  select(model, date, sell_earning) %>%
  tk_augment_timeseries_signature() %>%
  select(-contains("month"))
```

```
augmented_transactions <-
augmented_transactions[complete.cases(augmented_transactions), ]
(var <- data.frame(colnames = colnames(augmented_transactions[,
sapply(augmented_transactions, is.numeric)]),
           colvars = colVars(as.matrix(augmented_transactions[,
sapply(augmented_transactions, is.numeric)]))) %>%
  filter(colvars == 0))
##   colnames colvars
## 1     hour       0
## 2   minute       0
## 3   second       0
## 4   hour12       0
## 5    am.pm       0
augmented_transactions <- select(augmented_transactions, -
one_of(as.character(var$colnames)))
```

Removing Highly correlated features

```
relation <- cor(augmented_transactions[, sapply(augmented_transactions,
is.numeric)])
p.relation <- cor_pmat(augmented_transactions[,
sapply(augmented_transactions, is.numeric)])

ggcorrplot(relation,  type = "upper", outline.col = "white", hc.order = TRUE,
p.mat = p.relation,
           colors = c(palette_light()[1], "white", palette_light()[2]))
```

The picture can't be displayed.

```
correlation <- findCorrelation(relation, cutoff=0.8)
augmented_transactions <- select(augmented_transactions, -
one_of(colnames(relation)[correlation]))
train <- filter(augmented_transactions, model == "train") %>%
  select(-model)
test <- filter(augmented_transactions, model == "test")
fit_lm <- glm(sell_earning ~ ., data = train)
tidy(fit_lm) %>%
  gather(x, y, estimate:p.value) %>%
  ggplot(aes(x = term, y = y, color = x, fill = x)) +
    facet_wrap(~ x, scales = "free", ncol = 2) +
    geom_bar(stat = "identity", alpha = 0.8) +
    scale_color_manual(values = palette_light()) +
    scale_fill_manual(values = palette_light()) +
    theme_tq() + labs(x="", y="")+
    theme(axis.text.x = element_text(angle = 75, vjust = 1, hjust = 1))
```

The picture can't be displayed.

```
augment(fit_lm) %>%
  ggplot(aes(x = date, y = .resid)) +
    geom_hline(yintercept = 0, color = "red") +
    geom_point(alpha = 0.5, color = palette_light()[[1]]) +
    geom_smooth() +
    theme_tq()
```

```
## `geom_smooth()` using method = 'loess'
```

The picture can't be displayed.

#Visualizing the Prediction test

```
pred_test <- test %>%
  add_predictions(fit_lm, "pred_lm") %>%
  add_residuals(fit_lm, ".resid_lm")
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
pred_test %>%
    ggplot(aes(x = date, y = .resid_lm)) +
    geom_hline(yintercept = 0, color = "red") +
    geom_point(alpha = 0.5, color = palette_light()[[1]]) +
    geom_smooth() +
    theme_tq()
## `geom_smooth()` using method = 'loess'
```

The picture can't be displayed.

```
pred_test %>%
  gather(x, y, sell_earning, pred_lm) %>%
  ggplot(aes(x = date, y = y, color = x)) +
    geom_point(alpha = 0.5) +
    geom_line(alpha = 0.5) +
    scale_color_manual(values = palette_light()) +
    theme_tq()
```

The picture can't be displayed.

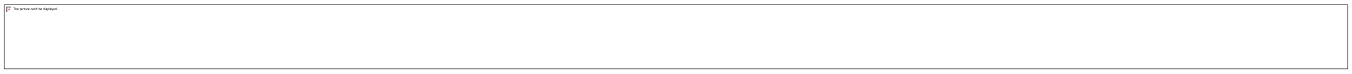
# Forcasting

```
# Extract index
time_index <- transaction_per_day %>%
    tk_index()
augmented_transactions %>%
  ggplot(aes(x = date, y = diff)) +
    geom_point(alpha = 0.5, aes(color = as.factor(diff))) +
    geom_line(alpha = 0.5) +
    scale_color_manual(values = palette_light()) +
    theme_tq()
```

The picture can't be displayed.

```
augmented_transactions %>%
  select(date, wday.lbl, diff) %>%
  filter(wday.lbl != "Sunday" & diff > 86400) %>%
  mutate(days_missing = diff / 86400 -1)
```

```
## # A tibble: 5 x 4
##   date       wday.lbl    diff days_missing
##   <date>     <ord>      <int>        <dbl>
## 1 2011-01-04 Tuesday  1036800          11.
## 2 2011-04-26 Tuesday   432000           4.
## 3 2011-05-03 Tuesday   172800           1.
## 4 2011-05-31 Tuesday   172800           1.
## 5 2011-08-30 Tuesday   172800           1.
off_days <- c("2010-12-24", "2010-12-25", "2010-12-26", "2010-12-27", "2010-
12-28", "2010-12-29", "2010-12-30", "2010-01-01", "2010-01-02", "2010-01-03",
              "2011-04-22", "2011-04-23", "2011-04-24", "2011-04-25", "2011-
05-02", "2011-05-30", "2011-08-29", "2011-04-29", "2011-04-30") %>%
  ymd()
future_time_index <- time_index %>%
    tk_make_future_timeseries(n_future = 300, inspect_weekdays = TRUE,
inspect_months = FALSE, skip_values = off_days)
## pad applied on the interval: day
## The following `skip_values` were not in the future date sequence: 2010-12-
24, 2010-12-25, 2010-12-26, 2010-12-27, 2010-12-28, 2010-12-29, 2010-12-30,
2010-01-01, 2010-01-02, 2010-01-03, 2011-04-22, 2011-04-23, 2011-04-24, 2011-
04-25, 2011-05-02, 2011-05-30, 2011-08-29, 2011-04-29, 2011-04-30
future_time_index %>%
    tk_get_timeseries_signature() %>%
    ggplot(aes(x = index, y = diff)) +
    geom_point(alpha = 0.5, aes(color = as.factor(diff))) +
    geom_line(alpha = 0.5) +
    scale_color_manual(values = palette_light()) +
    theme_tq()
## Warning: Removed 1 rows containing missing values (geom_point).
## Warning: Removed 1 rows containing missing values (geom_path).
```



```
future_data <- future_time_index %>%
    tk_get_timeseries_signature() %>%
    rename(date = index)

prediction <- predict(fit_lm, newdata = future_data)
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
prediction <- future_data %>%
    select(date) %>%
    add_column(sell_earning = prediction)
transaction_per_day %>%
  select(sell_date, sell_earning) %>%
  rename(date = sell_date) %>%
  rbind(prediction) %>%
  ggplot(aes(x = date, y = sell_earning)) +
    scale_x_date() +
    geom_vline(xintercept = as.numeric(max(transaction_per_day$sell_date)),
color = "red", size = 1) +
    geom_point(alpha = 0.5) +
    geom_line(alpha = 0.5) +
    theme_tq()
## Warning: Removed 1 rows containing missing values (geom_point).
```

```
residual_testing <- pred_test$.resid_lm
sd_of_res_testing <- sd(residual_testing, na.rm = TRUE)

prediction <- prediction %>%
    mutate(
        lo.95 = sell_earning - 1.96 * sd_of_res_testing,
        lo.80 = sell_earning - 1.28 * sd_of_res_testing,
        hi.80 = sell_earning + 1.28 * sd_of_res_testing,
        hi.95 = sell_earning + 1.96 * sd_of_res_testing
        )
transaction_per_day %>%
  select(sell_date, sell_earning) %>%
  rename(date = sell_date) %>%
  ggplot(aes(x = date, y = sell_earning)) +
    geom_point(alpha = 0.5) +
    geom_line(alpha = 0.5) +
    geom_ribbon(aes(ymin = lo.95, ymax = hi.95), data = prediction,
                fill = "#D5DBFF", color = NA, size = 0) +
    geom_ribbon(aes(ymin = lo.80, ymax = hi.80, fill = key), data =
prediction,
                fill = "#596DD5", color = NA, size = 0, alpha = 0.8) +
    geom_point(aes(x = date, y = sell_earning), data = prediction,
               alpha = 0.5, color = palette_light()[[2]]) +
    geom_smooth(aes(x = date, y = sell_earning), data = prediction,
                method = 'loess', color = "white") +
    theme_tq()
## Warning: Removed 1 rows containing non-finite values (stat_smooth).
## Warning: Removed 1 rows containing missing values (geom_point).
```

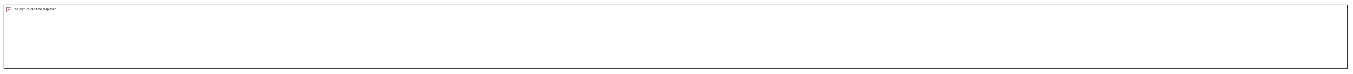
# Prediction using another model

```
transaction_per_day <- transaction_per_day %>%
  mutate(model = ifelse(sell_date <= "2011-10-01", "train2", "test2"))

train2 <- filter(transaction_per_day, model == "train2") %>%
  select(sell_date, sell_earning) %>%
  rename(ds = sell_date,
         y = sell_earning)

test2 <- filter(transaction_per_day, model == "test2") %>%
  select(sell_date, sell_earning) %>%
  rename(ds = sell_date)
closed <- data.frame(ds = as.Date(c("2010-12-24", "2010-12-25", "2010-12-26",
"2010-12-27", "2010-12-28",
                                      "2010-12-29", "2010-12-30", "2010-01-
01", "2010-01-02", "2010-01-03",
                                      "2011-04-22", "2011-04-23", "2011-04-
24", "2011-04-25", "2011-05-02",
                                      "2011-05-30", "2011-08-29", "2011-04-
29", "2011-04-30"))) %>%
  mutate(holiday = paste0("off_day_", seq_along(1:length(ds))))
```

```
prophet_model_test <- prophet(train2,
                              growth = "linear", # growth curve trend
                              n.changepoints = 100, # Prophet automatically
detects changes in trends by selecting changepoints from the data
                              yearly.seasonality = FALSE, # yearly seasonal
component using Fourier series
                              weekly.seasonality = TRUE, # weekly seasonal
component using dummy variables
                              holidays = closed)
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to
override this.
## Initial log joint probability = -9.51268
## Optimization terminated normally:
##   Convergence detected: absolute parameter change was below tolerance
```

Predicting test data

```
forecasting <- predict(prophet_model_test, test2)
## Warning: Unknown or uninitialised column: 'y'.
forecasting %>%
  mutate(.resid2 = trend - yhat) %>%
  ggplot(aes(x = ds, y = .resid2)) +
    geom_hline(yintercept = 0, color = "red") +
    geom_point(alpha = 0.5, color = palette_light()[[1]]) +
    geom_smooth() +
    theme_tq()
## `geom_smooth()` using method = 'loess'
```



```
future <- make_future_dataframe(prophet_model_test, periods = 300)
forecast <- predict(prophet_model_test, future)
plot(prophet_model_test, forecast) +
    theme_tq()
```