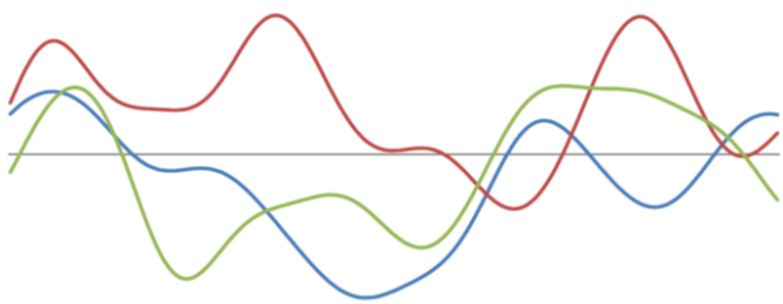
MMA 867 Predictive Modelling

Alfred Consulting Group

Time Series Prediction



Team Alfred
Crystal Fang
Eddie Wang
Gopala Goyal
Faiza Shallwani
Jacqueline Mak
Sushant Karmakar
Sushil Megharaj

Executive Summary

Alfred Consulting Group aims to provide data analytics services and solutions to uncover valuable insights and recommendations on how companies can operate most efficiently and drive efficiency. Our team has been tasked with providing consulting services to one of the largest Russian firms, 1C Company to determine the number of gift sets needed to fulfill the campaign obligations. It is a Black Friday promotional campaign where the firm needs to predict the total number of items sold for 60 regional stores for November 2015. Using a time-series forecasting technique with ARIMA, we have forecasted the total number of items sold per store. This data is aimed to help our client improve the corporate decision-making process and ensure that there is a sufficient inventory of gift sets to maintain the integrity of the campaign.

Exploratory Data Analysis

Our Kaggle dataset consists of daily historical data from January 2013 to October 2015. It has 2935849 data rows, and 5 variables including one response variable, the number of products sold (item_cnt_day). We are predicting a monthly amount of this measure. To prepare the data, we converted the data variable to date format and the number variables into factor variables.

```
> str(sales_train)
'data.frame':	2935849 obs. of  8 variables:
 $ date          : Date, format: "2013-01-02" "2013-01-03" "2013-01-05" ...
 $ date_block_num: int  0 0 0 0 0 0 0 0 0 0 ...
 $ shop_id       : Factor w/ 60 levels "0","1","2","3",..: 60 26 26 26 26 26 26 26 26 26 ...
 $ item_id       : Factor w/ 21807 levels "0","1","2","3",..: 21792 2496 2496 2498 2499 2508 2509 2515 2515 2516
...
 $ item_price    : num  999 899 899 1709 1099 ...
 $ item_cnt_day  : num  1 1 -1 1 1 1 1 1 1 3 ...
 $ month         : Factor w/ 12 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ year          : Factor w/ 3 levels "2013","2014",..: 1 1 1 1 1 1 1 1 1 1 ...
```

The data provided comes with no missing values.

```
> summary(sales_train)
      date            date_block_num     shop_id            item_id          item_price      
 Min.   :2013-01-01   Min.   : 0.00   31     : 235636   20949  :  31340   Min.   :    -1.0  
 1st Qu.:2013-08-01   1st Qu.: 7.00   25     : 186104   5822   :   9408   1st Qu.:   249.0  
 Median :2014-03-04   Median :14.00   54     : 143480   17717  :   9067   Median :   399.0  
 Mean   :2014-04-03   Mean   :14.57   28     : 142234   2808   :   7479   Mean   :   890.9  
 3rd Qu.:2014-12-05   3rd Qu.:23.00   57     : 117428   4181   :   6853   3rd Qu.:   999.0  
 Max.   :2015-10-31   Max.   :33.00   42     : 109253   7856   :   6602   Max.   :307980.0  
                                      (Other):2001714   (Other):2865100                      
  item_cnt_day            month            year        
 Min.   : -22.000   1      : 303561   2013:1267562  
 1st Qu.:   1.000   3      : 284057   2014:1055861  
 Median :   1.000   12     : 274032   2015: 612426  
 Mean   :   1.243   2      : 270251                 
 3rd Qu.:   1.000   8      : 248415                 
 Max.   :2169.000   6      : 237428                 
                    (Other):1318105                 
```

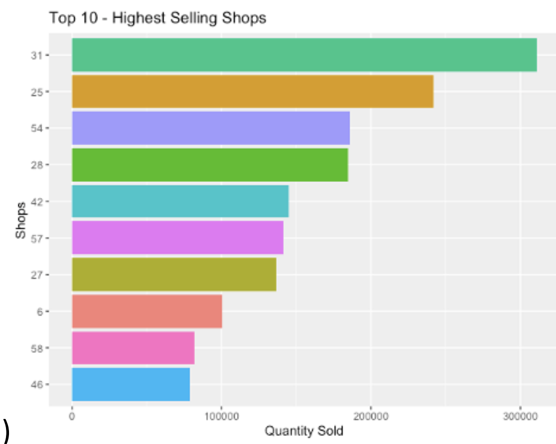
Current State of Business

The main goal of our proposal was to predict the total number of items sold per store in the month of November. Having an idea of the number of potential units sold was important as it helped decide how many gift sets were needed for the event. Prior to building the time series forecasting model, we reviewed the current state of the company.

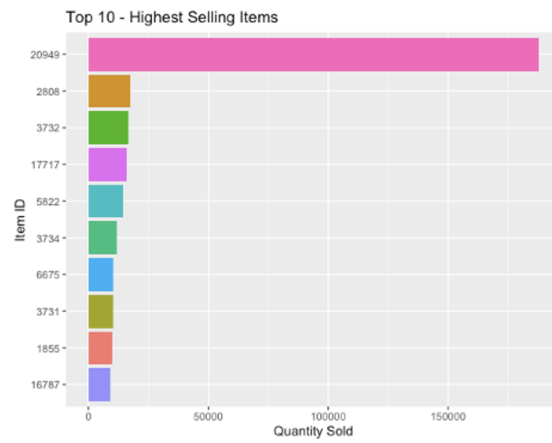
Top 10

(a) Most popular shops with the highest number of items sold
(b) Highest Selling items

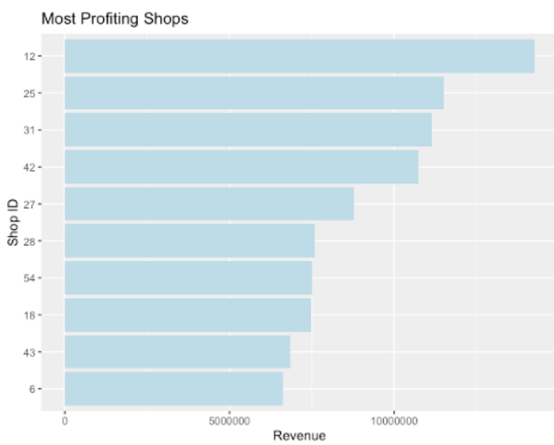
(b) Most Profiting Shops



(a)

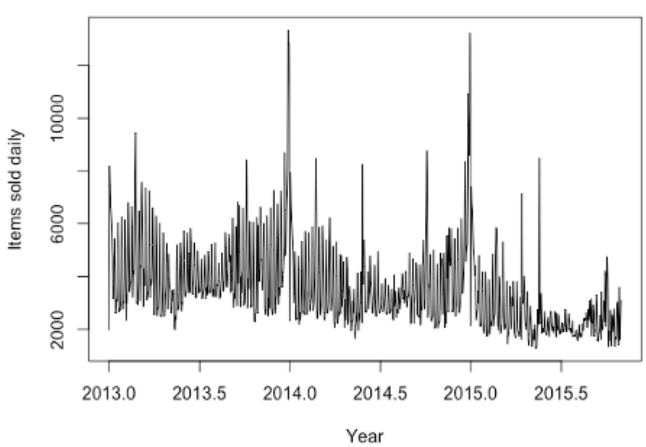


(b)



(c)

Daily historical data from January 2013 to October 2015

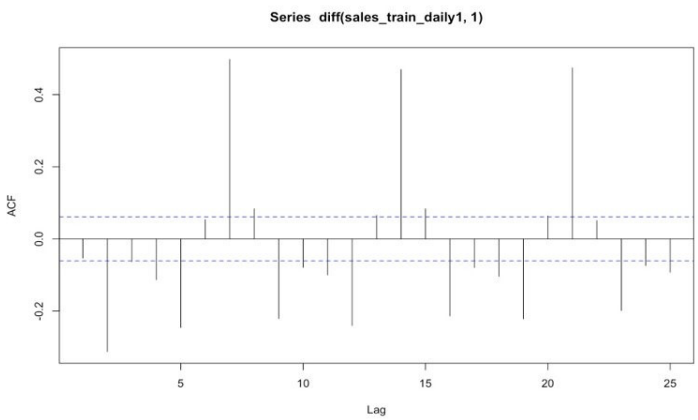


The daily historical data collected from January 2013 to October 2015 has a cyclical pattern, which is also known as seasonality. The last two months of the year have more sales in terms of items sold. The seasonality in the dataset will be removed using seasonal differencing prior to the ARIMA modeling.

Approach & Methodology

Feature Engineering

We started by getting the daily quantity sold for each shop, after grouping the sales data by 'shop_id'. Then we checked heteroskedasticity for changing variance which could interfere with our analysis. We also looked at the ACF plot, where we observed significant spikes coming out at lag 7, 14, and 21. These spikes suggest seasonality (i.e., weekly pattern) in the data. This was expected as, in retail, we often see more sales on weekends and less on weekdays.

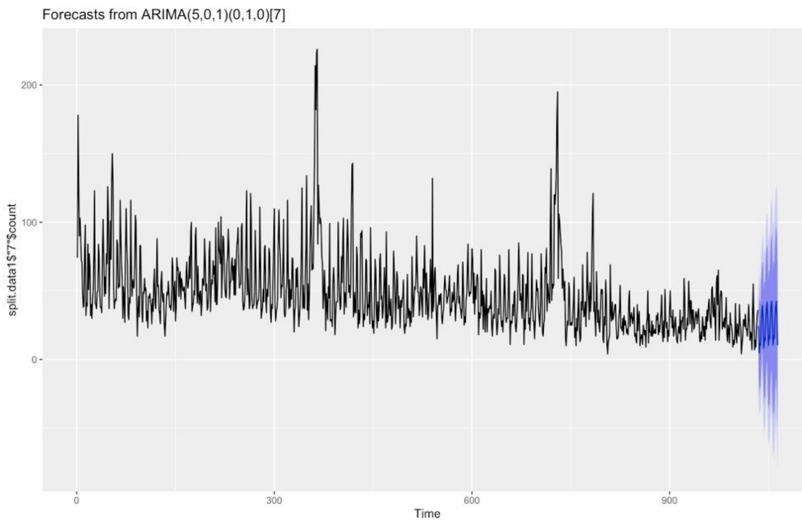


Modeling and Forecasting

Since we are looking to forecast future sales from historical data, we believe that time series analysis is the best method. We first divided the dataset into high- and low-price levels. We had set the threshold at 1000 dollars. The rationale behind this was that when we checked the quartiles of the dataset, 75% is at 1000 dollars. We decided that the third quartile is a good threshold as most of the gifts we expect to give out are the cheaper ones.

```
> quantile(sales_train$item_price)
    0%    25%    50%    75%   100%
    -1    249    399    999 307980
```

For both high- and low-price levels, and for each of the 60 shops, we ran a time series analysis. We used the auto.arima function and set the seasonality to 7 to account for the weekly pattern. The graph below is an example of the forecasted sales, the blue region is the confidence interval. We also observe the seasonality in the forecasted numbers as there are 4 spikes in the 30-day period.



After we have the forecasted sales numbers from each day, we aggregated them into a monthly total for each shop.

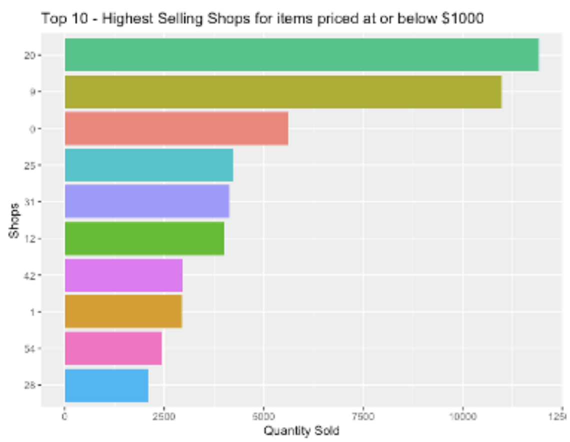
Optimization

We tried to improve the modeling accuracy by changing the p, d, q parameters in the arima model. We first tried adjusting the orders manually in a for-loop but opted for setting the maximum p and q of the auto.arima function to 8 from the default of 5 as it achieves the same results with more ease. For some time series, we got a lower AIC which indicates a better-fit model.

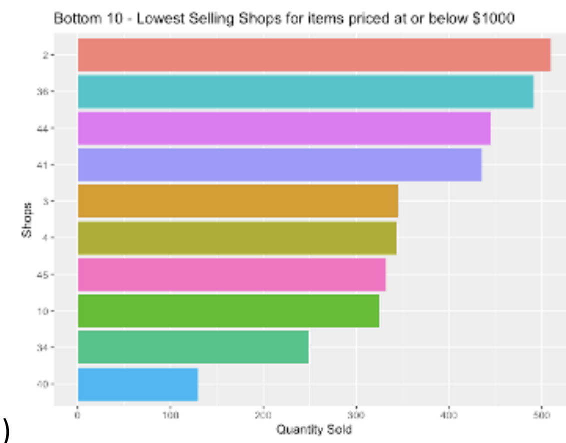
Key Outcomes & Results

From our analysis, we were able to predict the total quantity sold for each store in Nov 2015. We predict that most of the top-selling stores in high-price items will also perform well with low-price items. There is also a strong correlation between whether a shop is top-selling between Jan 2013 and Oct 2015 and whether it's top-selling in Nov 2015, although the order of the shops is different for the two time periods.

To summarize our findings, below are the forecast quantity pertaining to the top 10 and bottom 10 shops:
(a) Top 10 – Highest Selling Shops for items priced at or below $1000
(b) Bottom 10 – Lowest Selling Shops for items priced at or below $1000
(c) Top 10 – Highest Selling Shops for items priced above $1000
(d) Bottom 10 – Lowest Selling Shops for items priced above $1000

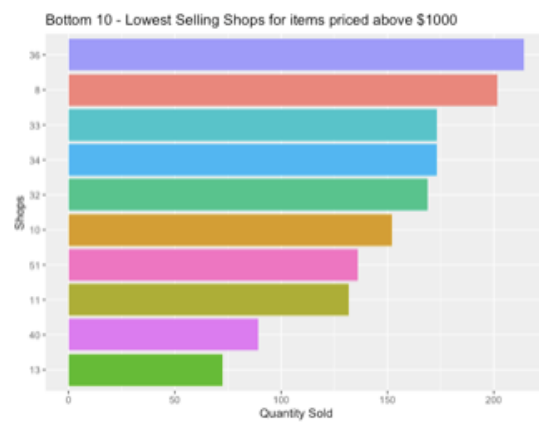


(a)



(b)

(c)



(d)

Recommendations

1. **<u>Promotions & Advertisements</u>**
   Based on the forecasted number of items sold for each shop, we can promote the Black Friday gift with purchase offer on social media and through a themed email/ direct mail to drive sales. For maximization, we recommend distributing the promotion to existing customers in the top 10 highest selling stores. For retention, we can focus on the data collected pertaining to the lowest selling shops and send the promotional offer to customers who have decreased their consumption in shopping.

2. **<u>Segmentation, Targeting & Positioning</u>**
   Depending on the nature of the campaign, we can use a targeting system to effectively communicate to both existing and potential customers who will shop the most in the top 10 highest selling stores. Once a list of customers to be targeted has been defined, the Black Friday gift with purchase promotion can be distributed all at once through an email blast.

3. **<u>Improving Sales in the Weakest Performing Store</u>**
   Regarding the data collected pertaining to the stores that have low item sales, we recommend the following promotional strategies to drive sales:
   - **Location condition**: Using the customer database, the company can offer exclusive discounts or promotions based on where the customer is located.
   - **Quantity condition**: The company can activate a promotion for customers who purchase a specific item that could be based on the brand or category, which would encourage them to buy more than their usual shopping habits.

- **Non-buyer condition**: The company can target inactive customers who have not purchased an item from the store for a certain period of time. For example, we can send BOGO offers or dollar-off digital coupons to the customers.

Technical Appendix

```
#load the libraries

library(dplyr)
library(ggplot2)
library(readxl)
library(fpp)

#load the data

sales_train <- read.csv("/Users/jacquelinemak/MMA /MMA 867 Predictive Modelling/Team
Project/competitive-data-science-predict-future-sales/sales_train.csv")

#explore the data

head(sales_train)
str(sales_train)
summary(sales_train)
dim(sales_train)

#Number of missing data for each variable

lapply(sales_train, function(x)
  sum(is.na(x))) #no missing data

#convert data variable to date format

sales_train <- sales_train %>%
  mutate(date = gsub("[.]", "/", date))%>%
  mutate(date = as.Date.character(date, format="%d/%m/%Y"))

# convert number variable into factor variable

sales_train <- sales_train %>%
  mutate(shop_id=factor(shop_id))%>%
  mutate(item_id=factor(item_id))%>%
  mutate(month=factor(month(date)))%>%
  mutate(year=factor(year(date)))

#Visualizations
#Viz 1. Top 10 - Highest Selling Shops

popular_shops <-sales_train%>%
  group_by(shop_id) %>%
  summarize(total_count = sum(item_cnt_day)) %>%
  ungroup() %>%
  arrange(desc(total_count))
```

```
head(popular_shops,10)
options(scipen=999)

head(popular_shops,10) %>%
  ggplot(aes(x = reorder(as.factor(shop_id), total_count), y = total_count,fill=as.factor(shop_id)))
+
  geom_bar(stat = 'identity') +
  theme(legend.position = "none") +
  labs(y = 'Quantity Sold', x = 'Shops', title = 'Top 10 - Highest Selling Shops') +
  coord_flip()

# Viz 2. Top 10 Highest Selling Items

popular_items <-sales_train%>% group_by(item_id) %>% summarize(Icount =
sum(item_cnt_day)) %>% ungroup() %>% arrange(desc(Icount))
head(popular_items,10)

head(popular_items,10) %>% ggplot(aes(x = reorder(as.factor(item_id), Icount), y =
Icount,fill=as.factor(item_id))) +
  geom_bar(stat = 'identity') +
  theme(legend.position = "none")+
  labs(y = 'Total sales', x = 'Item ID', title = 'Highest Selling Items') +
  coord_flip()

#Viz 3. Most Profiting Shops

pop_items_per_shop <- sales_train %>%
  group_by(shop_id, item_id) %>%
  summarise(Revenue = sum(item_cnt_day*item_price)) %>%
  filter(Revenue == max(Revenue)) %>%
  arrange(desc(Revenue)) %>%
  ungroup()
top_10_revenue <- head(pop_items_per_shop,10)

ggplot(data=top_10_revenue, aes(x = reorder(as.factor(shop_id), Revenue), y = Revenue, fill =
as.factor(item_id))) +
  geom_bar(stat = "identity", fill = 'lightblue') +
  coord_flip() +
  theme(legend.position = "none")+
  labs(title= "Most profiting Item per Shop", x= "Shop ID", y = "Revenue", fill = "Item ID")

#Summarize products sold by days

sales_train_daily <- sales_train %>%
  group_by(date) %>%
  summarise(item_cnt_daily = sum(item_cnt_day))

#assign new variable to item count daily

sales_train_daily_temp <- sales_train_daily$item_cnt_daily
```

```
#change to time series object

sales_train_daily1 <- ts(sales_train_daily_temp, frequency=365, start=c(2013, 1))

#Check for variance

plot.ts(sales_train_daily1, xlab="Year", ylab="Items sold daily")

#Check seasonality

library(fpp)
Acf(diff(sales_train_daily1,1),lag.max =25)
#big spike at lag 7, 14, 21; the pattern is weeks, multiple of 7

#We now remove the seasonality using seasonal differencing

sales_train_daily1.deSeasonality <- diff(sales_train_daily1,7)
plot.ts(sales_train_daily1.deSeasonality, xlab="Date", ylab="Items sold daily after removing
trend and seasonality") #theres no weekly pattern
Acf(sales_train_daily1.deSeasonality,lag.max =25)

#-------------Automatic ARIMA Modeling ------------------
model.auto.daily.sales <- auto.arima(sales_train_daily1.deSeasonality, stepwise=FALSE,
seasonal= FALSE)
model.auto.daily.sales

# It suggests a ARIMA(2,0,3) model with 0 mean

checkresiduals(model.auto.daily.sales)  # Check the quality of fit. Residuals should:
# (1) not have any significant autocorrelation
# (2) follow normal distribution
# (3) have stable variance over time

#we can now fit the model, autoarima suggested 2, 0, 3
# We can use the auto selected model to make forecasting; the arima function can build the
model

fit.daily.sales1 <- Arima(sales_train_daily1.deSeasonality, order=c(2,0,3))
Fit.daily.sales1

#####################################################
#improve the model but manually choosing the p, d, q
#####################################################

fit.daily.sales2 <- Arima(sales_train_daily1.deSeasonality, order=c(3,0,3)) #slightly better
fit.daily.sales2

fit.daily.sales3 <- Arima(sales_train_daily1.deSeasonality, order=c(1,0,3)) #worse than
autoarima
fit.daily.sales3
```

```
fit.daily.sales4 <- Arima(sales_train_daily1,
order=c(20,0,8),seasonal=list(order=c(0,1,0),period=7)) #model with the lowest AIC/BIC/AICc
fit.daily.sales4

#check residuals for final model
checkresiduals(fit.daily.sales4) #looks good

fc1.SALES<-forecast(fit.daily.sales4,30) #Sales Forecast - number of products sold per day in
november 2015

#forecast for all shops

fc1.SALES$mean
autoplot(fc1.SALES)

##########################################################
#Sales Forecast - Number of products sold by shop_id
##########################################################

#check the number of shops

unique(sales_train$shop_id) #60 shops

#split the train dataset based on the threshold for item priced at $1000

#Item priced at <= $1000 and grouped by shop_id and date

sales_train1 <- sales_train %>%
  filter(sales_train$item_price <= 1000) %>%
  group_by(date, shop_id) %>%
  summarise(count = sum(item_cnt_day))

#Item priced at > $1000 and grouped by shop_id and date

sales_train2 <- sales_train %>%
  filter(sales_train$item_price > 1000) %>%
  group_by(date, shop_id) %>%
  summarise(count = sum(item_cnt_day))

#Split the dataset by shop_id

split.data<- split(sales_train1, sales_train1$shop_id)

#For testing purposes - not in final model
#using order 20,0,8  as this is model with the lowest AIC/BIC/AICc for Shop ID 1

split.data1 <- split.data[1]
fit.daily.sales.byshop.self <- Arima(split.data1$'0'$count, order = c(20,0,8))
fit.daily.sales.byshop.self
fc <- forecast(fit.daily.sales.byshop.self, 30)
fc$mean
autoplot(fc)
```

```
##########################################################################
#For loop to forecast sales for items priced at or below $1000 by shop_id
##########################################################################

forecasted_item_count_by_shop <- list()
sum_of_forecasted_item_by_shop <- matrix(nrow = 60, ncol = 2)

for(i in 1:60){
  num_temp <- i-1
  data_temp <-split.data[i]
  ts.data_temp <- ts(data_temp[[as.character(num_temp)]]$count, frequency=365, start=c(2013,
1))
  fit_temp <- auto.arima(ts.data_temp, stepwise=FALSE, seasonal= FALSE, max.p = 8, max.q =
8, max.order = 8, trace = TRUE)
  best_aic <- fit_temp$aic
  best_model <- fit_temp
  print("Forecast for shop")
  print(as.character(num_temp))
  print(fit_temp$arma)
  print(best_model$arma)
  print(fit_temp$aic)
  forecasted_item_count_by_shop[[i]] <- forecast(fit_temp, 30)#our best model
  print(forecasted_item_count_by_shop[[i]])
  sum_of_forecasted_item_by_shop[i,1] <- num_temp
  sum_of_forecasted_item_by_shop[i,2] <- sum(forecasted_item_count_by_shop[[i]]$mean)
}

#Top 10 for items price at or below $1000 by shop_id

top_10_shops <- as.data.frame(sum_of_forecasted_item_by_shop)

desc_order_top_10 <- top_10_shops %>%
  arrange(desc(V2))

top_ten_shops1 <- head(desc_order_top_10,10)

top_ten_shops1 %>%
  ggplot(aes(x = reorder(as.factor(V1), V2), y = V2,fill=as.factor(V1))) +
  geom_bar(stat = 'identity') +
  theme(legend.position = "none") +
  labs(y = 'Quantity Sold', x = 'Shops', title = 'Top 10 - Highest Selling Shops for items priced at
or below $1000') +
  coord_flip()

#Bottom 10 for items price at or below $1000 by shop_id

bottom_10_shops <- as.data.frame(sum_of_forecasted_item_by_shop)

ascen_order_bottom_10 <- bottom_10_shops %>%
  arrange(V2)
```

```
bottom_ten_shops1 <- head(ascen_order_bottom_10,10)

bottom_ten_shops1 %>%
  ggplot(aes(x = reorder(as.factor(V1), V2), y = V2,fill=as.factor(V1))) +
  geom_bar(stat = 'identity') +
  theme(legend.position = "none") +
  labs(y = 'Quantity Sold', x = 'Shops', title = 'Bottom 10 - Lowest Selling Shops for items priced
at or below $1000') +
  coord_flip()

##########################################################################
#For loop to forecast sales for items priced above $1000 by shop_id
##########################################################################

split.data1<- split(sales_train2, sales_train2$shop_id)
forecasted_item_count_by_shop1 <- list()
sum_of_forecasted_item_by_shop1 <- matrix(nrow = 60, ncol = 2)

for(i in 1:60){
  num_temp <- i-1
  data_temp <-split.data1[i]
  ts.data_temp <- ts(data_temp[[as.character(num_temp)]]$count, frequency=365,
start=c(2013, 1))
  fit_temp <- auto.arima(ts.data_temp, stepwise=FALSE, seasonal= FALSE,max.p = 8, max.q = 8,
max.order = 8, trace = TRUE)
  print("Forecast for shop")
  print(as.character(num_temp))
  print(fit_temp)
  forecasted_item_count_by_shop1[[i]] <- forecast(fit_temp, 30)
  print(forecasted_item_count_by_shop1[[i]])
  sum_of_forecasted_item_by_shop1[i,1] <- num_temp
  sum_of_forecasted_item_by_shop1[i,2] <- sum(forecasted_item_count_by_shop1[[i]]$mean)
}

bar_graph1 <-
  ggplot(as.data.frame(sum_of_forecasted_item_by_shop1), aes(x=V1, y=V2))+
  geom_bar(stat = "identity", colour = "black", fill= "lightpink") +
  labs(x="Shop ID", y = "Quantity Sold") +
  ggtitle("Forecasted count for items priced above $1000") +
  theme_minimal()
bar_graph1

#Top 10 for items price above $1000 by shop_id

top_10_shops_b <- as.data.frame(sum_of_forecasted_item_by_shop1)
```

```
desc_order_top_10_b <- top_10_shops_b %>%
  arrange(desc(V2))

top_ten_shops2 <- head(desc_order_top_10_b,10)

top_ten_shops2 %>%
  ggplot(aes(x = reorder(as.factor(V1), V2), y = V2,fill=as.factor(V1))) +
  geom_bar(stat = 'identity') +
  theme(legend.position = "none") +
  labs(y = 'Quantity Sold', x = 'Shops', title = 'Top 10 - Highest Selling Shops for items priced
above $1000') +
  coord_flip()

#Bottom 10 for items priced above $1000 by shop_id

bottom_10_shops_b <- as.data.frame(sum_of_forecasted_item_by_shop1)

ascen_order_bottom_10_b <- bottom_10_shops_b %>%
  arrange(V2)

bottom_ten_shops2 <- head(ascen_order_bottom_10_b,10)

bottom_ten_shops2 %>%
  ggplot(aes(x = reorder(as.factor(V1), V2), y = V2,fill=as.factor(V1))) +
  geom_bar(stat = 'identity') +
  theme(legend.position = "none") +
  labs(y = 'Quantity Sold', x = 'Shops', title = 'Bottom 10 - Lowest Selling Shops for items priced
above $1000') +
  coord_flip()
```