



DATA ANALYTICS USING R



HISTORICAL SALES AND ACTIVE INVENTORY

Team 'R'Acers

Dnyanesh Kulkarni

Jay Mehta

Partha Ganrai

Ritika Sinha

Saloni Bhoyar

Yashaswini Krishna Naik

Table of Contents

EXECUTIVE SUMMARY	2
PROBLEM STATEMENT.....	2
APPROACH	3
PREDICTIVE MODELLING.....	6
APPLICATION DEVELOPMENT	8
CONCLUSION.....	11
APPENDIX.....	11



EXECUTIVE SUMMARY

Improving forecasting has always been the main focus of traditional inventory management systems. When inventory is driven by actual demand, in addition to forecast, the supply chain operates at greater efficiency and at increased profit margins. Inventory systems attempt to solve the inventory management dilemma through better forecasting, improvements in the order point / order quantity process, or by adjusting safety stock levels. In order to achieve the same we built multiple predictive models. Sold Flag variable was an indicator to find if the item was sold in past six months or not and Sold Count indicated the count of items sold. File Type variable had two categories – Historical data and Active inventory data. While we tried various models, Random Forest gave the best results with 99% model accuracy. We also tried ABC/XYZ analysis of forecasting. It is a method of grouping items based on their value and dynamics of consumption, sales, trend or seasonal demand. We created an interactive web application using R-shiny which would allow users to plug in values of basic attributes and find out the status of various items in the inventory. This would help in taking further action so as to order or not to order a particular item and to maintain safety stock which is a level of extra stock that is maintained to mitigate risk of stockouts due to uncertainties in supply and demand.

PROBLEM STATEMENT

Our objective was to predict the items to be retained in the inventory and the quantity of items based on analysis of historical and active inventory data. Sold Flag and Sold Count were the two variables to be predicted. To make the process user-friendly to create an interactive web application using R_Shiny which would take inputs like Release Year, Marketing Type, New Release Flag, Strength Factor, Reg Price, Item count in order to predict the items that are likely to be sold and the quantity in which they are likely to be sold.

APPROACH

The dataset contains information about a set of products in the inventory out of which we are trying to determine which products to continue to sell and which products to remove from inventory.

```
> head(SalesData)
  Order File_Type SKU_number SoldFlag SoldCount MarketingType ReleaseNumber New_Release_Flag StrengthFactor PriceReg ReleaseYear
1     2 Historical   1737127      0         0           D           15             1       682743    44.99      2015
2     3 Historical   3255963      0         0           D            7             1      1016014    24.81      2005
3     4 Historical    612701      0         0           D            0             0      340464    46.00      2013
4     6 Historical    115883      1         1           D            4             1      334011   100.00      2006
5     7 Historical    863939      1         1           D            2             1     1287938   121.95      2010
6     8 Historical    214948      0         0           D            0             0     1783153   132.00      2011

  ItemCount LowUserPrice LowNetPrice
1         8       28.97       31.84
2        39         0.00       15.54
3        34       30.19       27.97
4        20      133.93       83.15
5        28         4.00       23.99
6        33      138.98       13.64
```

Order: Sequential counter of orders

File Type: Two types historical and active inventory

Sold Flag: If product is sold in past 6 months then value=1; If not sold value = 0

Sold Count: Number of products sold in past 6 months

SKU Number: Unique identifier for each product

Marketing Type: Categories of how the product is marketed

And variables such as Release Number, Strength Factor, Price Reg, Release Year, Item Count, Low User Price and Low Net Price. The file contains both historical sales data and active inventory, which can be discerned with the column titled "File Type".

1. Packages used

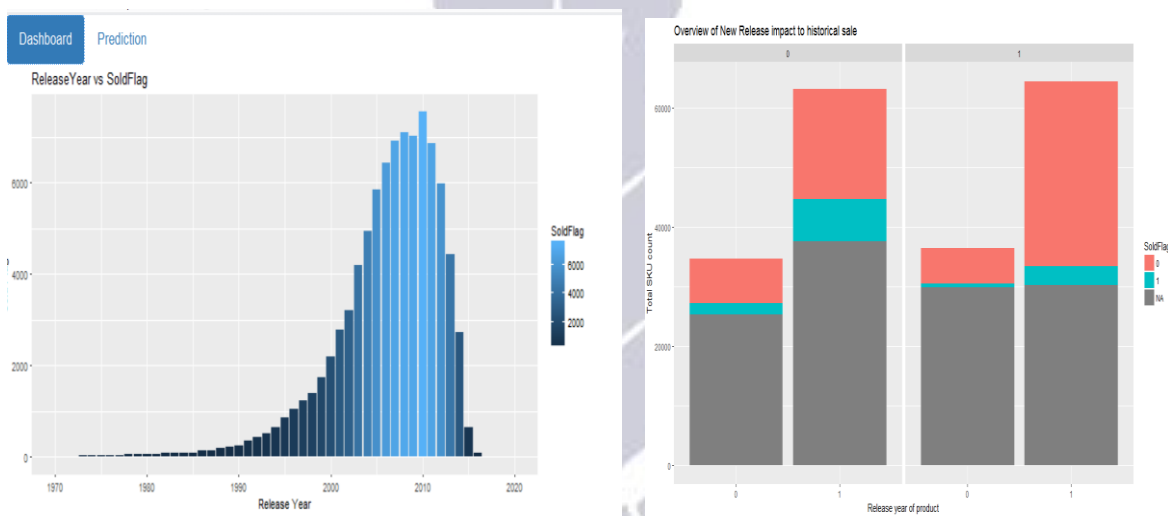
In the process of exploratory data analysis, model building and creating web application we used packages like shiny, corrplot, ggplot2, xgboost, neuralnet, dplyr, randomForest, plotROC, caret, rattle, rpart.plot, e1071 and pROC.

2. Data Pre-processing

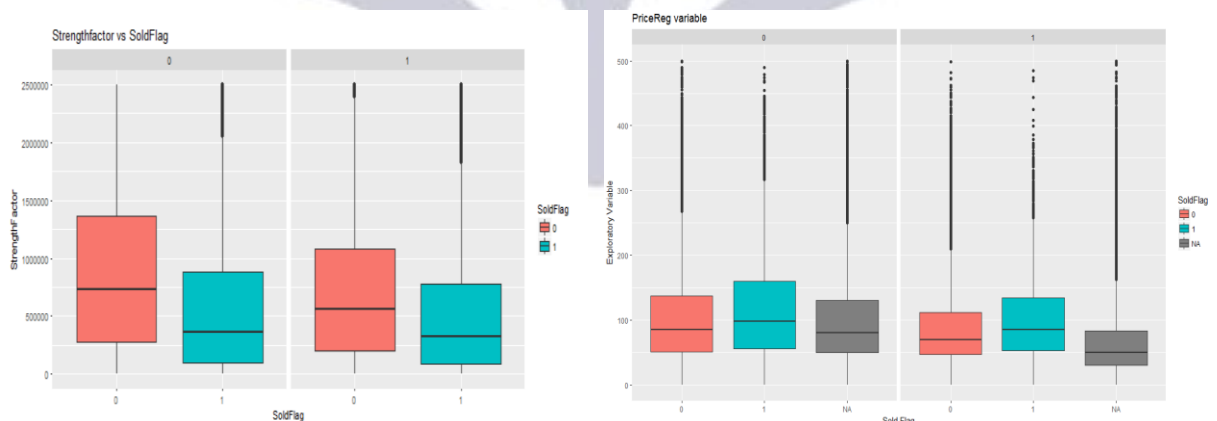
We changed the datatypes of few variables. SoldFlag and NewRelease Flag variables were converted to factor datatypes. The data set was divided into Training data and Test data based on the File_Type being 'Historical' or 'Active'. Rescaling was done on variables ReleaseYear, StrengthFactor, PriceReg, LowUserPrice and LowRegPrice. The variable ReleaseYear was converted to age of the items in inventory.

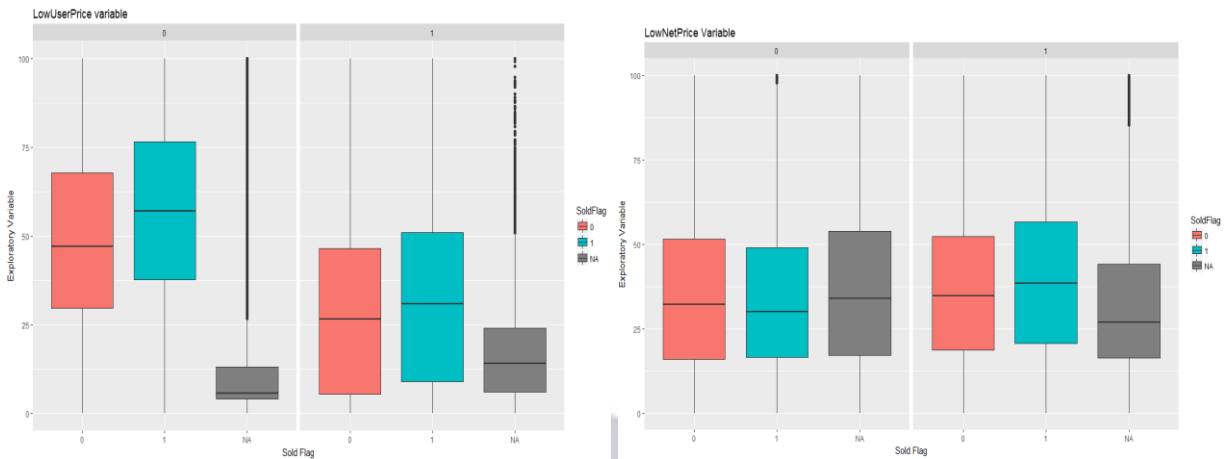
3. Exploratory data analysis

- Ggplot was used to visualize variables that impact SoldFlag across different marketing types



- To visualize exploratory variables such as Strength Factor, Price Reg, Low User Price and Low Net Price

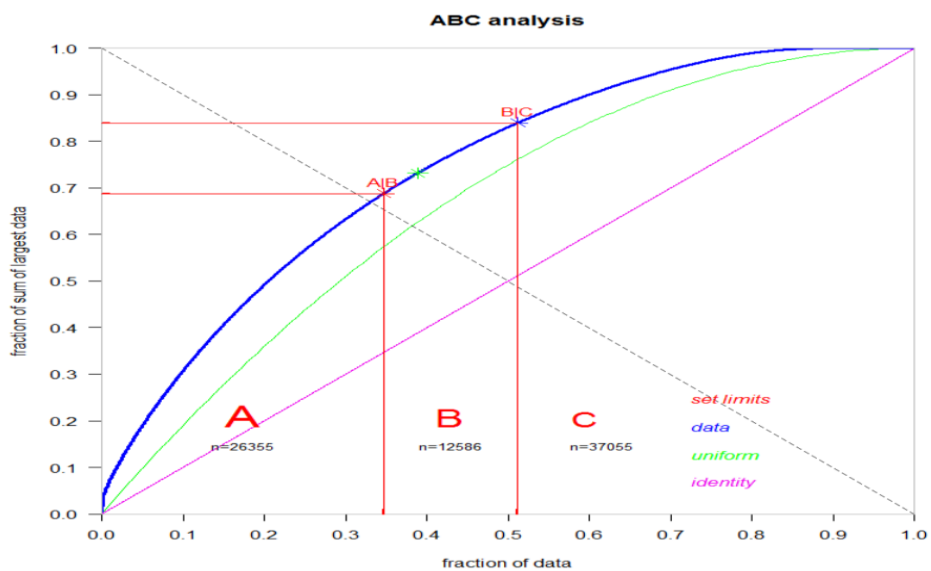




- **ABC Analysis**

The ABC inventory analysis is an inventory categorization technique which determines the importance of items and thus allows “Selective Inventory Control” based on the relative importance of items. The main objective of ABC analysis is to separate essential and non-essential inventory.

- A-Low proportion high value stocks. Requires continuous monitoring.
- B-Intermediate materials. Requires periodic inventory control
- C-Low value high proportion stocks. Cost exceeds benefit



PREDICTIVE MODELLING:

Sold Flag is the Response Variable.

We performed Logistic Regression, Decision Tree and Random Forest for the prediction of the SoldFlag.(Sold Flag as '0' represents Product not being sold and Sold Flag as '1' indicate as the product being sold. Please refer Appendix for the Logistic Regression and Decision Tree Models.

After running all the three models, Random Forest was selected as the best model.

Random Forest:

Random Forest is an ensemble learning method for classification that works by building multitude of the decision tree at training time and outputting the class that is the mode of the classes or mean prediction of the individual trees. The training algorithm uses the technique of Bootstrap aggregating or bagging. This bootstrapping procedure leads to better model performance without increasing the bias. This means that while the predictions of a single tree are highly sensitive to noise in its training set, the average of many trees is not, as long as the trees are not correlated. Random Forest has an additional feature of 'Feature Bagging' in which only a subset of features are selected at random out of the total and the best split feature from the subset is used to split each node in a tree.

For our dataset, we used 'randomForest' package in R to predict whether the product was sold in market in last 6 month or not. Accordingly we can forecast whether those product should be available in the inventory or should be applied special discount offer and moved out from the inventory. We run the randomForest with 500 trees.

```
RandomForest = randomForest(trainData$SoldFlag ~ .,data=trainData, ntree=
500,importance=TRUE)
```

‘getTree’ function used to extract the structure of the tree.

```
getTree(RandomForest,k=1,labelvar=FALSE)
```

Below is the output for the model:

In the above code, the first argument represents the random forest object, the second arguments represents which tree is to be extracted and the third argument checks on the variable name.

Result:

```
> getTree(RandomForest)
      left daughter right daughter split var   split point status prediction
1          2          3          6 3.644123e-02      1          0
2          4          5          5 1.524390e-01      1          0
3          6          7          7 3.464234e-03      1          0
4          8          9          3 3.896269e-02      1          0
5         10         11          3 1.938304e-01      1          0
6         12         13          1 5.000000e-01      1          0
7         14         15          6 6.007879e-02      1          0
8         16         17          3 3.894004e-02      1          0
9         18         19          8 1.645088e-03      1          0
```

Interpretation of the Result:

In numerical predictors, the data with values of the variable less than or equal to splitting node goes to the left daughter node while the other goes to right daughter node. In the above result, the split root is 6 (ItemCount). If the status of the prediction is 1 then it means that we have not reached prediction and if its -1 then it means that we have reached the prediction. The prediction value ‘1’ indicates that the node is terminal while ‘0’ indicates that it is not terminal. We also used the ‘importance’ function which gives single score per variable across the whole forest.

```
> importance(RandomForest)
      0      1 MeanDecreaseAccuracy MeanDecreaseGini
MarketingType  44.68935  75.327154      90.26216      657.0683
New_Release_Flag 33.78095 -8.908975      28.73244      180.6449
StrengthFactor  53.31232 39.920204      81.83933     2723.1178
PriceReg        80.26573 -6.635603      76.97702     2233.1049
ReleaseYear     34.32187 10.540302      40.47904     1309.5368
ItemCount       58.61456 74.453754     103.72298     2427.1300
LowUserPrice    49.23898 14.435111      57.65418     2309.4187
LowNetPrice     83.16818 -20.452672      74.88124     2215.9817
```

The above screenshot shows the contribution of each predictor variable in terms of MeanDecreaseAccuracy

and MeanDecreaseGiniA. Strength factor and ItemCount has the highest prediction based on the above result. Final Analysis is done by the analysing the confusion matrix and accuracy which is maximum for the random forest. Even the cost of error for random forest is much lower than the other two models.

Confusion Matrix and Statistics			Accuracy : 0.9965	
			95% CI : (0.9954, 0.9974)	
Prediction	Reference		No Information Rate : 0.824	
0	0	1	P-Value [Acc > NIR] : < 2.2e-16	
0	11730	50		
1	0	2456		

APPLICATION DEVELOPMENT

We analysed data, visualized data and built models on it. How are these really going to help the end user? Analysing the results of statistical analysis and making predictions from chunk of R codes would complicate the process and perplex the situation to a non-technical business user. A well-defined visually appealing interactive application would serve the purpose well. Is it possible with R to build an application, and the answer is R shiny. It is a package packed with different tools to build interactive web application straight from R, without having prior knowledge of HTML, CSS. We tried to build an application to help users to visually interpret the available data. The developed application has two tabs namely Dashboard and Prediction.

Dashboard

This part of application emphasizes on various plots to give a visual representation of attributes in the Data set. This part of dashboard is static and has no interactivity. Output depends on the available data in server.

Prediction

This part of the application is reactive in nature and supports user interactivity. There is provision for user to input various parameters like Release Year, Marketing Type, New Release Flag, Strength Factor, Reg Price, Item count etc. These inputs serves as an input to a model that predicts whether the product has chances of getting sold and the count of products that can be sold.

App Development :

The app development has two important steps-

1. Development of UI
2. ShinyServer

Step-1 (Development of UI): ShinyUI

Syntax-shinyUI(<<User interface definition>>)

TabPanel- TabPanel is used to describe a tab in the application.

Syntax-tabPanel(title, ..., value = title, icon = NULL)

For this application there are two tabPanels, one for each of the tabs. Shown below is the code for tabPanel of Dashboard. 'fluidRow' is used to define the layout of each row in the tab.

```
tabPanel("Dashboard", fluidRow(splitLayout(cellWidths = c("50%", "50%"),
  plotOutput("plot1", height = "370px", width = "750px"),
  plotOutput("plot2", height = "370px", width = "750px"))),
  fluidRow(splitLayout(cellWidths = c("50%", "50%"),
  plotOutput("plot3", height = "370px", width = "750px"),
  plotOutput("plot4", height = "370px", width = "750px") )))
```

Below is the tabpanel for the Prediction tab.

```
tabPanel("Prediction",
  fluidRow(pageWithSidebar(
    headerPanel(title = "Inventory Management System"),
    sidebarPanel(wellPanel(numericInput("ReleaseYear", "Release Year", value = 0),
      radioButtons("MarketType", "Select the marketing type", list("D", "S"), "D"),
      radioButtons("New_Release_Flag", "New release flag", list(1, 0), "1"),
      numericInput("StrengthFactor", "Strength Factor", value = 0),
      numericInput("PriceReg", "Price Reg", value = 0),
      numericInput("ItemCount", "Item Count", value = 0),
      numericInput("LowUserPrice", "Low User Price", value = 0),
      numericInput("LowNetPrice", "Low Net Price", value = 0),
      actionButton("submit", "Submit")), width = 2),
    mainPanel
```

Next, after defining the tabPanels, both the panels can be placed as tabs in the application using the 'tabsetPanel'. The tabset panel can be used as below.

```
ui= shinyUI(  
  tabsetPanel( TABPANEL 1,TABPANEL 2))
```

Step2:

❖ Development of ShinyServer for the Dashboard

Outputting plots to the UI build earlier. RenderPlot in shiny is used to render a reactive plot that is suitable for assigning to an output plot. Below is a sample code used to render a plot in the UI slot. Similarly, codes can be written for all other outputs

```
output$plot1 = renderPlot({ ggplot(data = plot1,  
  mapping = aes(x = ReleaseYear, y = SoldFlag, fill = SoldFlag)) +  
  labs(x = "Release Year",y = "Sold Flag") +geom_bar(stat = "identity") +  
  ggtitle("ReleaseYear vs SoldFlag") + xlim(1970,2020)  })
```

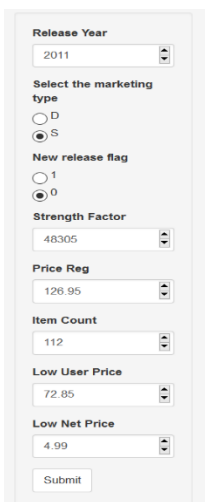
❖ Development of shiny server for the Prediction tab.

This stage incorporated steps to read the inputs from the UI, process it to align with the existing data and then pass on to a predictive model to predict the SoldFlag and SoldCount.

Below code details the above steps.

```
Data = eventReactive(input$submit,{  
  df <- data.frame(MarketingType = ifelse(input$MarketType=="D",1,0),  
    New_Release_Flag = as.integer(input$New_Release_Flag),  
    StrengthFactor = input$StrengthFactor,  
    PriceReg = input$PriceReg,  
    ReleaseYear=input$ReleaseYear,  
    ItemCount = input$ItemCount,  
    LowUserPrice= input$LowUserPrice,  
    LowNetPrice= input$LowNetPrice)  
  
  df = rescale_df(df=df,reference = trainData_original)  
  
  SoldFlag = predict(RandomForest,newdata = df,importance=TRUE)  
  df = cbind(df, SoldFlag)  
  result_SoldCount = predict(rf0, newdata = df, importance=FALSE)  
  df = cbind(df[9], as.integer(result_SoldCount))  
  data_frame = df  
})
```

Application UI



Release Year
2011

Select the marketing type
☐ D
☒ S

New release flag
☐ 1
☒ 0

Strength Factor
48305

Price Reg
126.95

Item Count
112

Low User Price
72.85

Low Net Price
4.99

Submit

Product Required	Count
1	14

CONCLUSION

We developed the best model that could predict the Sold Flag and sold count using Random Forest. R shiny is an excellent package in R which can help in web page development. We used this package and integrated the best model with this. This can help the Inventory Manager to manage the inventory easily. On the basis of different features of the product, he can easily analyze whether the product should be part of the inventory or not. If the product is fit to be part of the inventory, then what can be the sold count of the product? This application can help business to apply special offers and discount on products which were part of inventory but the likelihood of the sale is low. So, customers can be easily lured by the offer. And thus those places can be replaced by the new products whose demand is high. The business can easily understand the market orientation and accordingly source in and out the product in the inventory. All the product higher in sales than the average total sale should be made sure that it is in stock.

APPENDIX

1. Logistic Regression:

We ran logistic regression using 'glm' function in R. The sold count was excluded from the predictor variable. This model had an accuracy of 83.04%.

Confusion Matrix and Statistics

	Reference	
Prediction	0	1
0	11614	2298
1	116	208

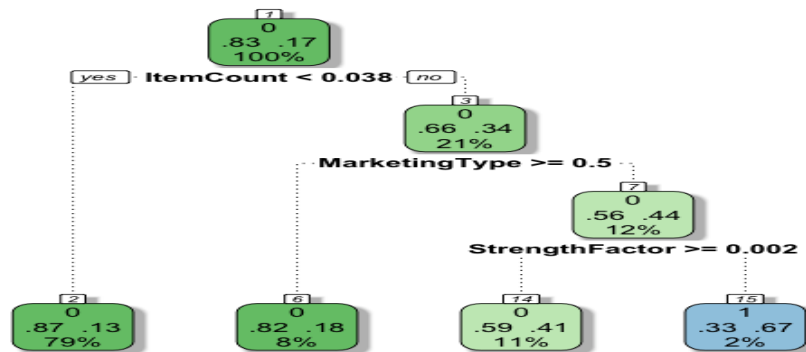
Accuracy : 0.8304
 95% CI : (0.8242, 0.8366)
 No Information Rate : 0.824
 P-Value [Acc > NIR] : 0.02164

Sensitivity : 0.9901
 Specificity : 0.9830
 Pos Pred Value : 0.8348
 Neg Pred Value : 0.6420
 Prevalence : 0.8240
 Detection Rate : 0.8158
 Detection Prevalence : 0.9772
 Balanced Accuracy : 0.5366
 'Positive' Class : 0

2. Decision Tree:

We ran the Decision Tree model using 'rpart' function in R. Here also all the variable after data pre-processing excluding the SoldCount was included as the predictor variable.

The model accuracy



Leaf Report Interpretation:

When the ItemCount < 0.038 then the product not being sold has the Information Gain of 0.79.

When the ItemCount >= 0.038 and MarketingType >= 0.05 then the product not being sold has Information Gain of 0.08.

When the ItemCount >= 0.038, MarketingType < 0.5 and Strength factor >= 0.002 then the product not being sold has Information Gain of 0.11.

When the ItemCount >= 0.038, MarketingType < 0.5 and Strength Factor <= 0.002 then the product being sold has Information Gain of 0.02.

Confusion Matrix and Statistics

	Reference	
Prediction	0	1
0	11663	2351
1	67	155

Accuracy : 0.8301
 95% CI : (0.8239, 0.8363)
 No Information Rate : 0.824
 P-Value [Acc > NIR] : 0.02667

Kappa : 0.0875
 McNemar's Test P-Value : < 2e-16
 Sensitivity : 0.99429
 Specificity : 0.06185
 Pos Pred Value : 0.83224
 Neg Pred Value : 0.69820
 Prevalence : 0.82397
 Detection Rate : 0.81926
 Detection Prevalence : 0.98441
 Balanced Accuracy : 0.52807
 'Positive' Class : 0

