SalesAnalysis

Hardeep Arora

09/12/2017

Datasets

How reliable is this data?

There are definitely some shortcomings with this dataset that would make it unreliable

Firstly

This dataset has sales and inventory data for a business and it's specifically mentioned that

"It is important to note that we have MANY products in our inventory, and very few of them tend to sell (only about 10% sell each year) and many of the products only have a single sale in the course of a year."

But the historical data provided for sales is just for last 6 months, this means we would not have a complete picture of the sales cycle. So this make the dataset not so reliable to make a true prediction on sales.

Also since only 10% of products sell, so there is also a class imbalance issue, as there is very less sale data and we would need to take care of it using downsampling or upsampling techniques.

Secondly

There is no clear indication on meaning of certain fields like

a. StrengthFactor - This limits the understanding and explainability of model that uses this feature.

b. PriceReg, LowUserPrice and LowNetPrice - There is no clear distinction between the three fields and hence it leaves lot of scope of assumptions which might not be true.

Thirdly

There are differences in value between SKU's common across historic and inventory data

a. The LowUserPrice differs a lot between historic and active records.

b. ItemCount also varies without any obvious logic in a lot of cases.

Fourthly

This dataset has very small subset of features, which would limit the depth of analysis that can be carried out.

How did you make an assessment?

I made this assessment based on the given information and eye balling the data.

How does the data quality limit or constrain your ideas for analysis?

The data quality limits the accuracy and completness of the analysis that would be carried out on this dataset and we might observe that the model does not generalize well on the unseen data.

What other datasets could be brought in to enrich these questions?

There are some key other datasets that can be brought to help analyze the below questions

a. First we need to capture one full year of historic sales data

b. Second data set containing the product features or BOM per product, would be a useful addition. It would help in conducting many more kind of analysis like price determination etc.

c. Third data set containing some more information about the customers would be helpful from point of view of retailer to determine the product and price mix.

d. Fourth any information on time/date of sale would be useful to carry out temporal analysis and determine seasonality in sales.

e. Fifth any kind of dataset on returned/defective products.

Stakeholders

What interesting questions could you answer with this data?

We can ask some interesting questions based on this small dataset, however we would need more datasets to answer all the questions appropriately.

If you are a retailer?

a. What products are selling?
   This is also the primary question asked along with this dataset.

b. How much quantity would be sold per sale?
   This is useful to get an idea about inventory to maintain at retailer end.

c. What price should I charge for each product?
   We need some more datasets to do this analysis like product and customer features.

If you are a consumer?

a. What should I pay for an item?
   We would need more features of the product along with its BOM if possible.

b. Is this the right retailer to shop on in terms of product mix and price?
   We would need more features of the product along with its BOM if possible.

If you are a wholesaler?

a. Prediction of the inventory required by this retailer by product?

b. What kind of sales happen at this retailer and what should be the frequency of refills?

c. Prediction of items that would be returned by this retailer?

Analysis

What is something interesting or useful to answer?

In the present shape and form this dataset is most suitable to answer the retailers questions. I would try to answer what products will sell and in how much quantity. I would be leaving the price for sale, because I feel we don't have enough features to answer this question adequately.

I plan to train a model on the historic data and use it to predict the values on inventory data. Given the amount of data, I feel it should be possible to make these predictions reasonably accurately.

Exploration

Let's explore the dataset a bit,

```
# Read the data
df <- fread("../data/SalesKaggle3.csv")

# Separate it into Sales and Inventory data
historicSales <- df[df$File_Type=="Historical",]
activeInv <- df[df$File_Type=="Active",]

dim(historicSales)
```

```
## [1] 75996    14
```

```
dim(activeInv)
```

```
## [1] 122921     14
```

```
colnames(historicSales)
```

```
##  [1] "Order"            "File_Type"        "SKU_number"
##  [4] "SoldFlag"         "SoldCount"        "MarketingType"
##  [7] "ReleaseNumber"    "New_Release_Flag" "StrengthFactor"
## [10] "PriceReg"         "ReleaseYear"      "ItemCount"
## [13] "LowUserPrice"     "LowNetPrice"
```

Since we are going to work on sales prediction, let's first look at the distribution of soldflag.

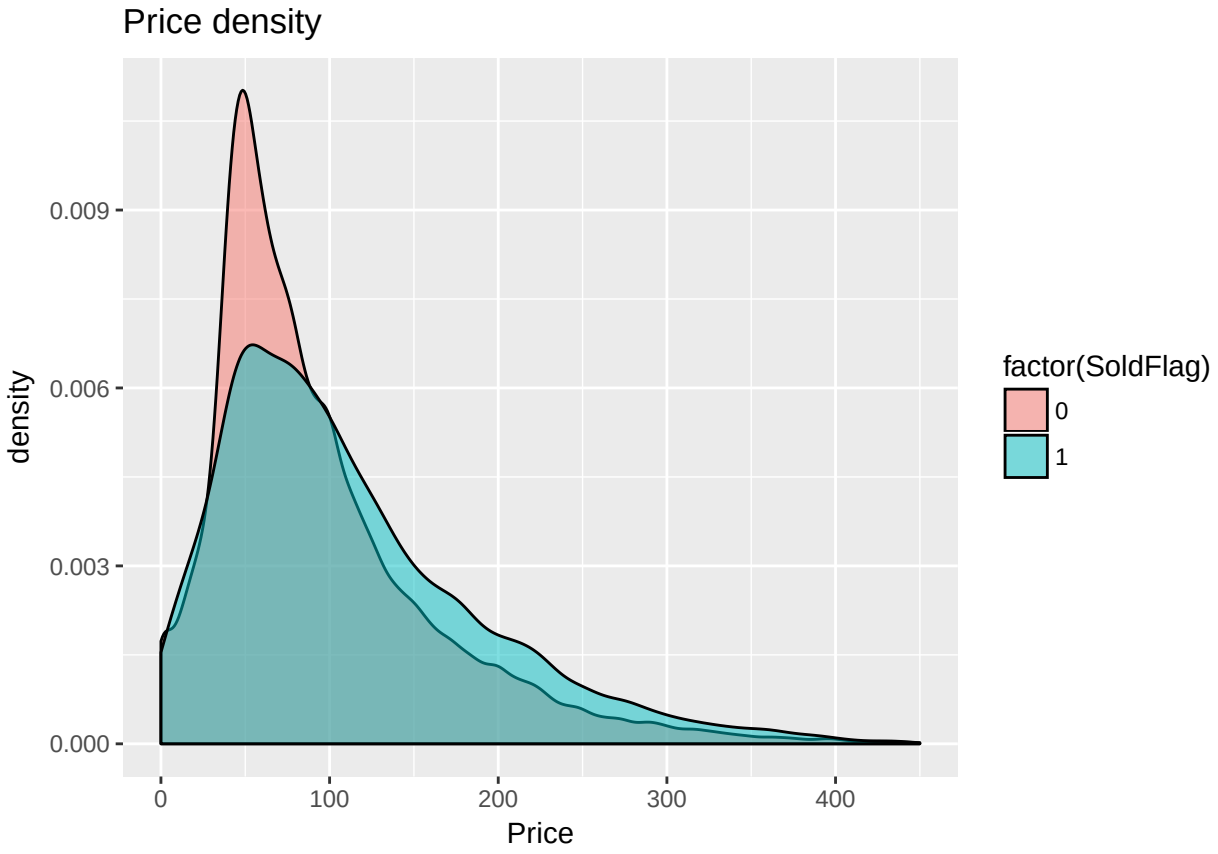
```
ftable(historicSales$SoldFlag)
```

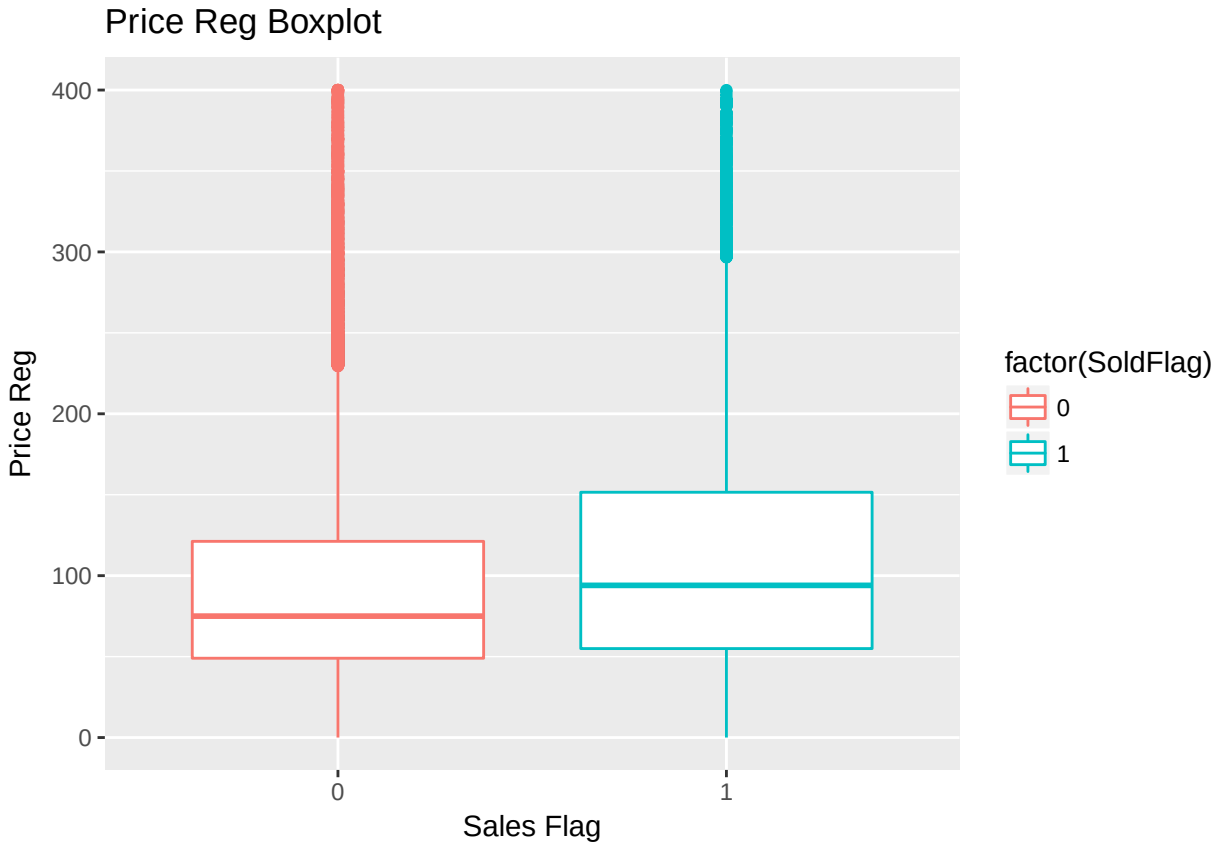
```
##      0     1
##
##  63000 12996
```

'0' means no-sales and '1' means sales and we can clearly conclude that this dataset is imbalanced.

Generally sales are correlated to the price of the product, so let us first plot the distribution of price between sale and no-sale cases.

```
## Scale for 'x' is already present. Adding another scale for 'x', which
## will replace the existing scale.
```





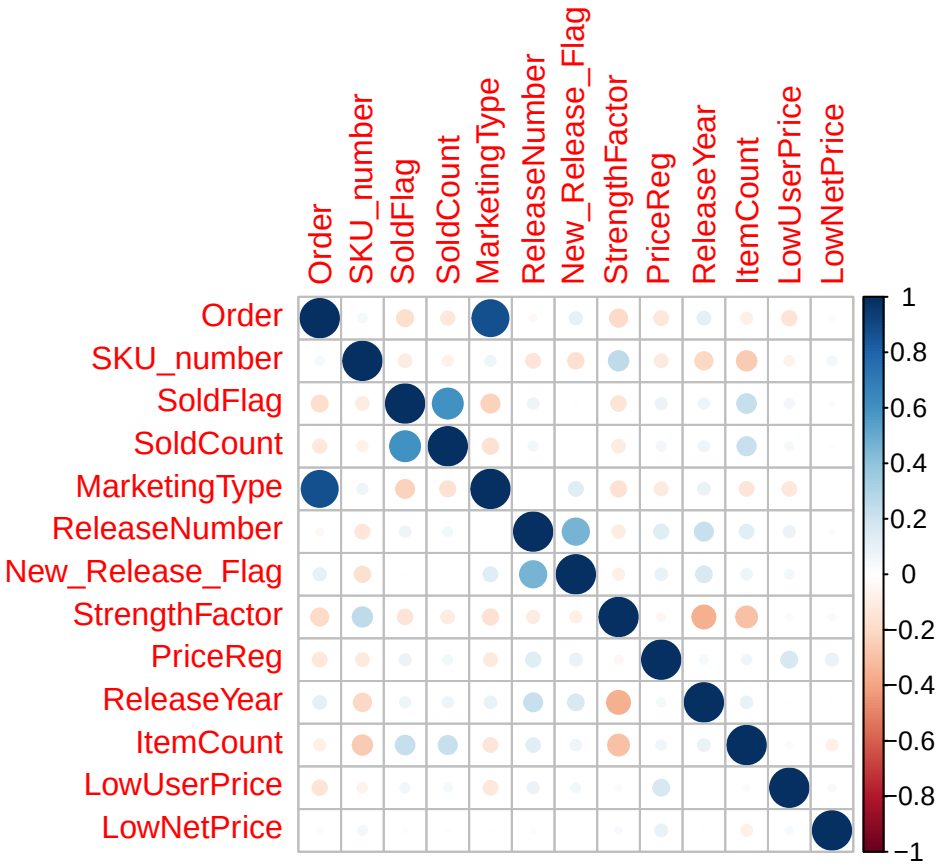
The boxplots above shows the median price for both sale and no-sale cases and there are quite a number of

outliers as well.

Correlation plot

Let's explore the correlation between the features.

```
corrplot(cor(scale(historicSales[,!c("File_Type")])), method="circle")
```

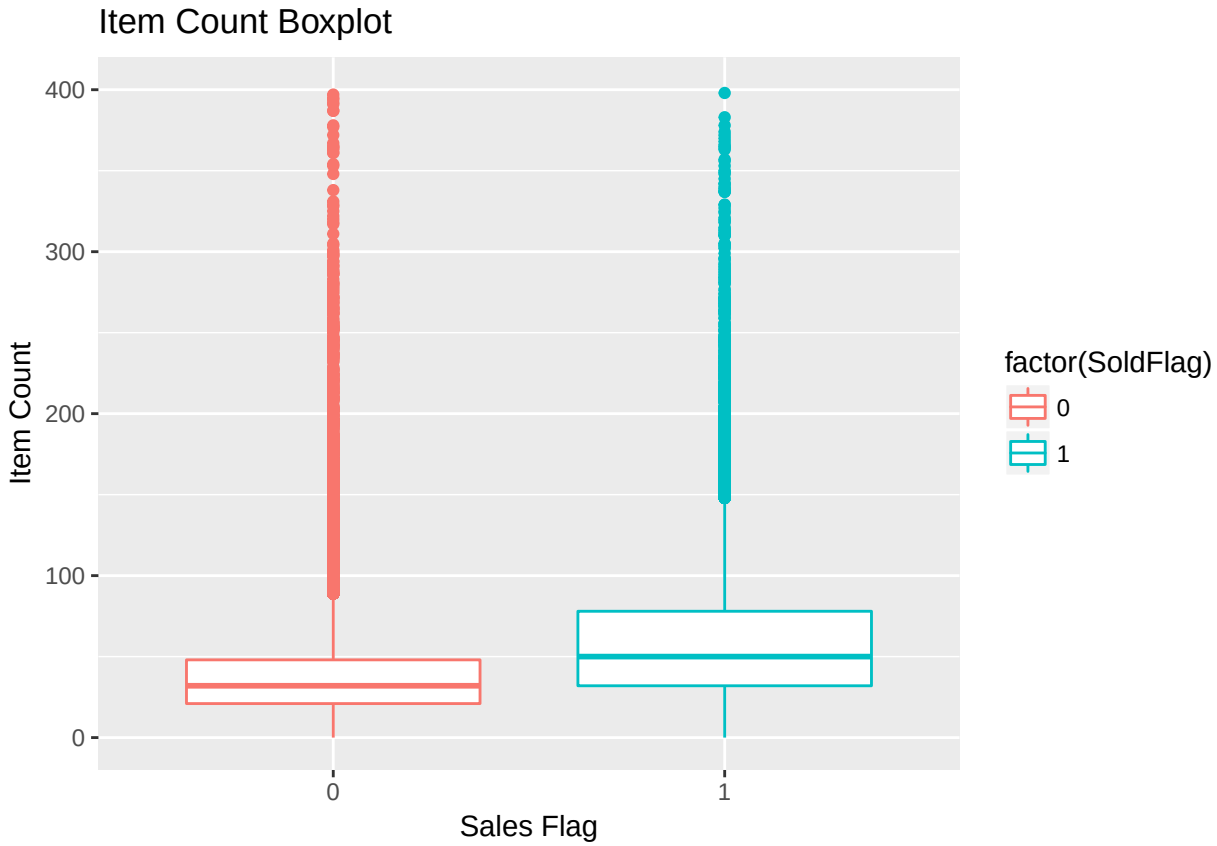


Above plot shows none of the features are highly correlated to SoldFlag or SoldCount.

a. **SoldFlag** - **ItemCount**, **MarketingType**, **Order**, **StrengthFactor** seem to have some small positive/negative correlation. Since MarketingType and Order and highly correlated among themselves, we can drop one of them from prediction. Ofcourse we can't use **SoldCount** as a predictor since it is not available in the inventory data.

b. **SoldCount** - **ItemCount**, **MarketingType**, **Order**, **StrengthFactor** seem to have some small positive/negative correlation. As stated above we would drop Order from the prediction. We can use **SaleFlag** to predict SoldCount for items that are sold, but I have decided not to do so because this would force the algo to make SaleCount close to zero for non-sale cases. In real life case if a sale happens on non-sale case, then we would have wrong indication of the how much sale would happen. Since we don't expect the sale to happen we might not stock enough inventory and end up having lost/less sale due to no-stock.

Since ItemCount is the main feature in both cases let's explore it a bit more



From the above boxplot we see the median ItemCount around 50 for sales and also observe some outliers above 150. We can experiment to see the impact of dropping them on our prediction later.

Data Pre-Processing

Now we would do some pre-processing of data, as listed below

a. Removing File_Type, Order, New_Release_Flag and SKU_numbers as they are not very useful or highly correlated to other features that we are keeping.

b. Saving the SaleFlag and SaleCount as they would be target of predictions.

c. Also since the dataset is imbalanced with 10% of sale cases only, would try to oversample it to make it more balanced. Here we are using SMOTE. I have tried without upsampling and the sensitivity in that case is less than 20%.

```
# Columns to Exclude
cols = c("File_Type","Order","SKU_number","New_Release_Flag")
historicSales <- historicSales[,!cols,with=FALSE]

# Balance the classes using oversampling
historicSales$SoldFlag <- as.factor(historicSales$SoldFlag)
historicSales1 <- SMOTE(SoldFlag ~ . ,
                        data= historicSales, perc.over = 100, perc.under = 200)

target.sales <- as.data.table(as.numeric(as.character(historicSales1$SoldFlag)))
target.sales.count <- as.data.table(historicSales1$SoldCount)

historicSales <- historicSales1
```

```
ftable(historicSales$SoldFlag)
```

```
##      0     1
##
##  25992 25992
```

```
historicSales <- historicSales[,!c("SoldFlag","SoldCount"),with=FALSE]
```

The classes are balanced now.

Data Partitioning

Now let's divide the data into test and train samples in ratio 20-80.

```
# Divide the data into train and test set (0.8,0.2)
testSize <- 0.2
indexes = sample(1:nrow(historicSales), size=testSize*nrow(historicSales))

# Split data
test <- historicSales[indexes,]
train <- historicSales[-indexes,]

lbl_train.sales <- target.sales[-indexes,]
lbl_test.sales <- target.sales[indexes,]

lbl_train.sales.count <- target.sales.count[-indexes,]
lbl_test.sales.count <- target.sales.count[indexes,]
```

SaleFlag Prediction

Now let's train a gradient boosted tree (xgboost) to predict the SoldFlag on this data. It would be binary logistic regression to predict the probability of sale between 0 and 1.

```
# Train the model for SaleFlag Prediction
new_tr <- model.matrix(~.,data = train)
new_ts <- model.matrix(~.,data = test)

lbl_train <- lbl_train.sales$V1
lbl_test <- lbl_test.sales$V1

dtrain <- xgb.DMatrix(data = new_tr,label = lbl_train)
dtest <- xgb.DMatrix(data = new_ts,label=lbl_test)

params <- list(booster = "gbtree",
               objective = "binary:logistic",
               eta=0.01,
               gamma=0,
               max_depth=7,
               min_child_weight=1,
               subsample=1,
               colsample_bytree=1)

set.seed(101)
xgb.sales <- xgb.train (params = params,
```

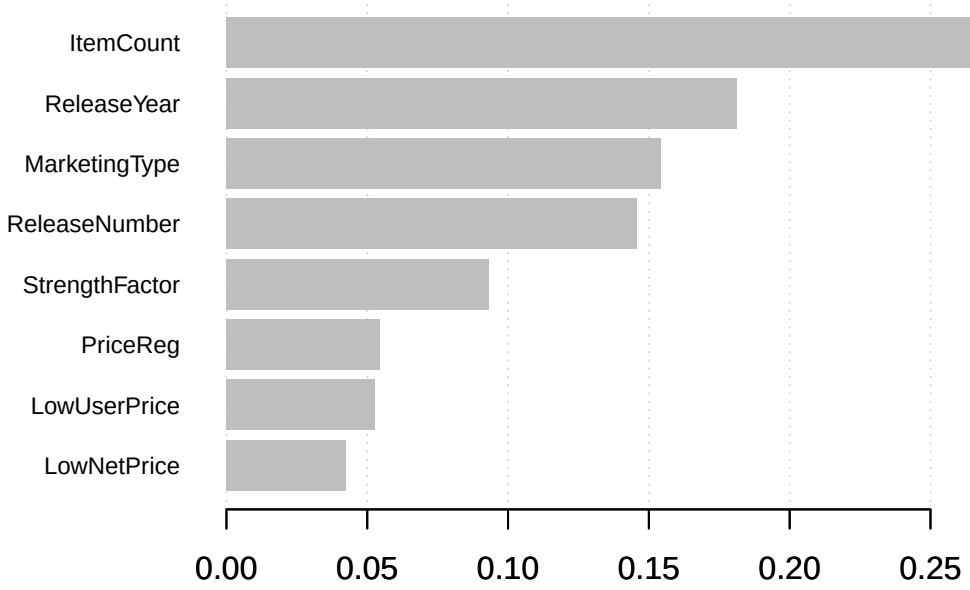
```
                    data = dtrain,
                    nrounds = 1510,
                    watchlist = list(train=dtrain,val=dtest),
                    print_every_n = 10,
                    early_stopping_rounds = 10,
                    maximize = T ,
                    eval_metric = "auc")
```

What evidence or rationale supports your findings?

Now let's print the results

Feature importance

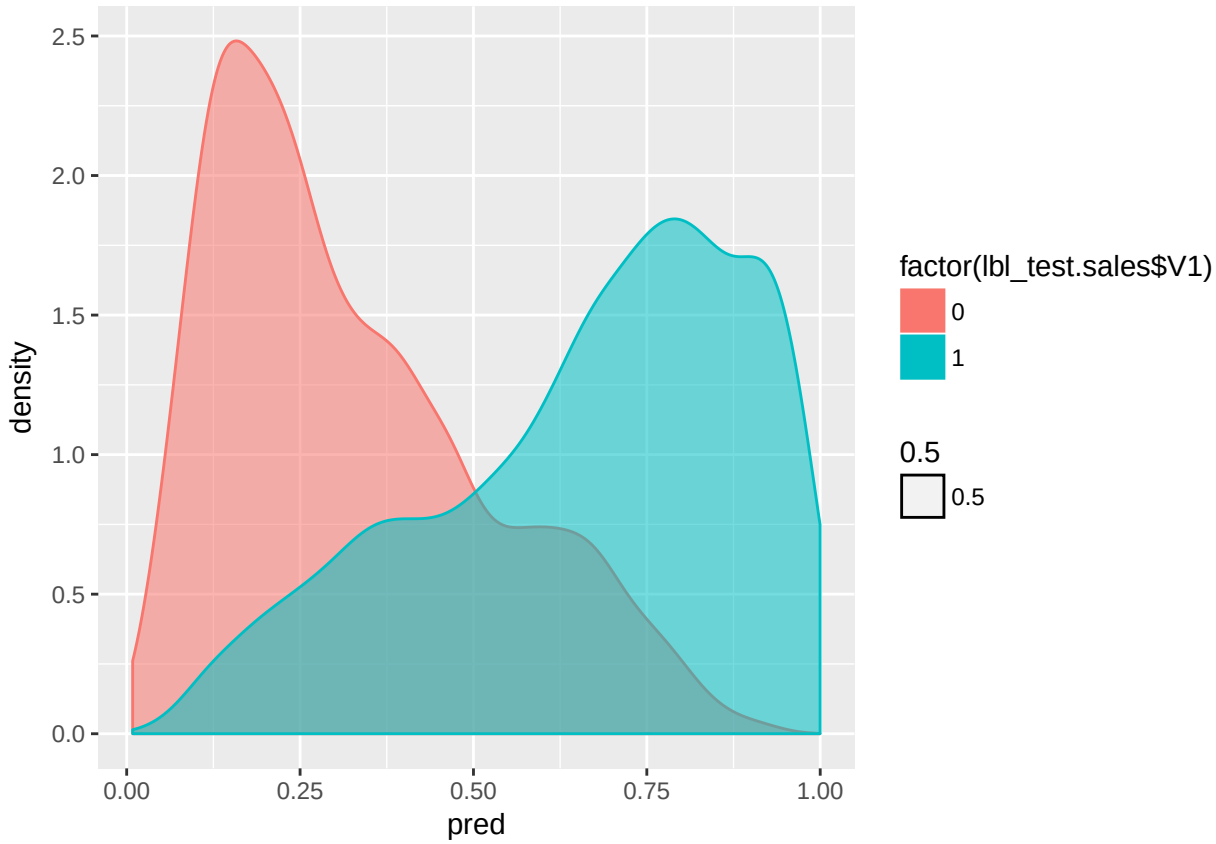
```
mat <- xgb.importance (feature_names = colnames(new_tr),model = xgb.sales)
xgb.plot.importance (importance_matrix = mat)
```



As anticipated from the correlation plots earlier, indeed the feature importance from xgboost show the same results.

Prediction distribution

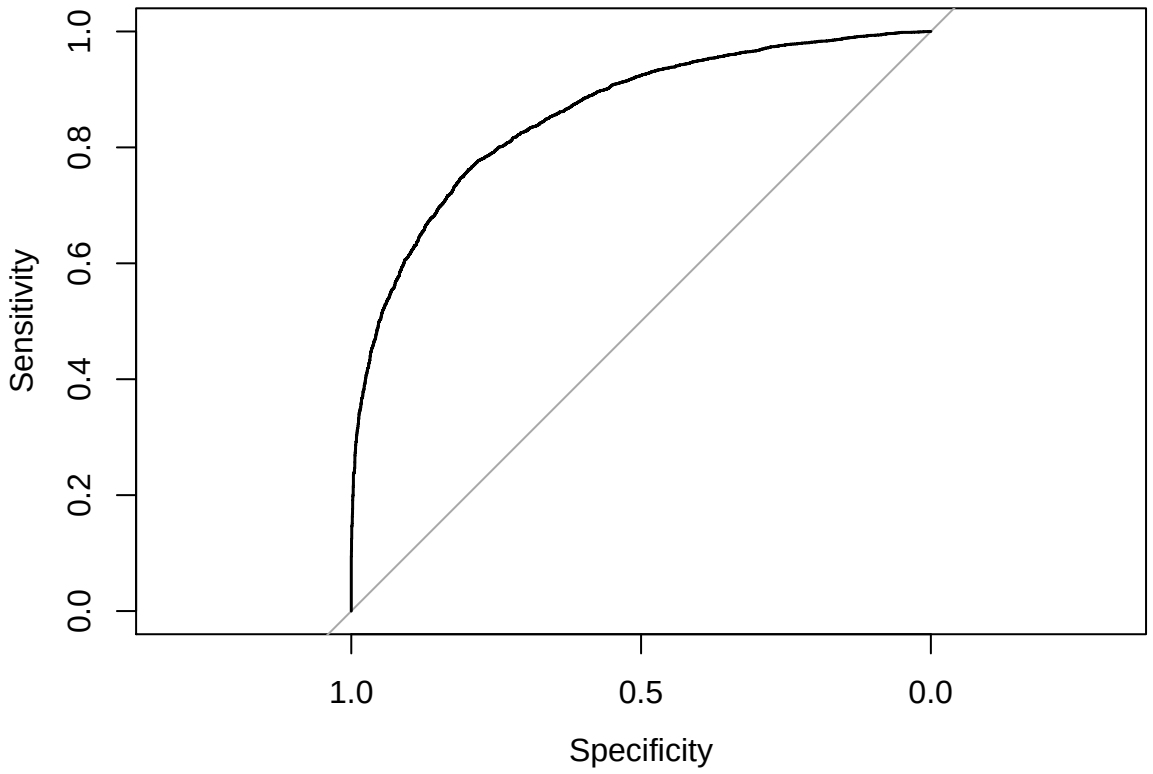
```
xgbpred <- predict (xgb.sales,dtest)
test$pred <- xgbpred

ggplot(test, aes(x=pred, colour =factor(lbl_test.sales$V1),
                 fill = factor(lbl_test.sales$V1), alpha = 0.5)) +
  geom_density()
```



The distribution of predictions are nicely separated at probability of 0.5

ROC curve

```
plot(pROC::roc(response = lbl_test.sales$V1,
               predictor = xgbpred,
               levels=c(0, 1)),lwd=1.5)
```




```
xgbpred <- ifelse (xgbpred > 0.5,1,0)

confusionMatrix (xgbpred, lbl_test)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 4136 1266
##          1 1030 3964
##
##                Accuracy : 0.7791
##                  95% CI : (0.771, 0.7871)
##     No Information Rate : 0.5031
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5584
##  Mcnemar's Test P-Value : 9.373e-07
##
##             Sensitivity : 0.8006
##             Specificity : 0.7579
##          Pos Pred Value : 0.7656
##          Neg Pred Value : 0.7938
##              Prevalence : 0.4969
##          Detection Rate : 0.3978
##    Detection Prevalence : 0.5196
##       Balanced Accuracy : 0.7793
##
##        'Positive' Class : 0
```

##

The AUC is approx 77% and True Positve (Sensitivity) and True Negative (Specificity) is also above 75%.

This looks to be good results for time being.

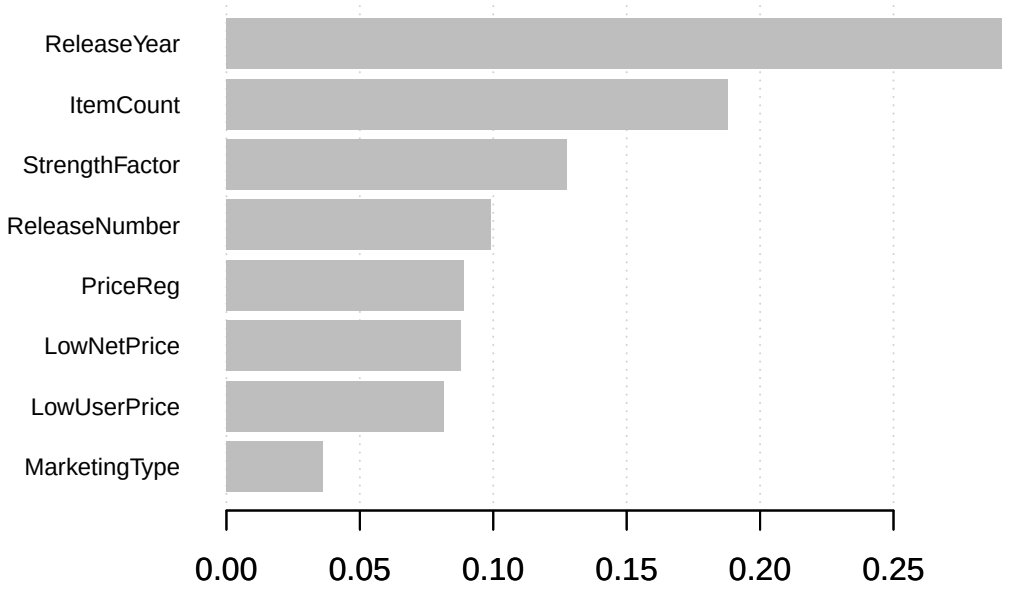
SaleCount Prediction

Now let's train the SaleCount predictor using gradient boosted trees (xgboost). In this case we would be doing a linear regression to predict the SoldCount.

```
new_tr <- model.matrix(~.,data = train)
new_ts <- model.matrix(~.,data = test)

lbl_train <- lbl_train.sales.count$V1
lbl_test <- lbl_test.sales.count$V1

dtrain <- xgb.DMatrix(data = new_tr,label = lbl_train)
dtest <- xgb.DMatrix(data = new_ts,label=lbl_test)

params <- list(booster = "gbtree",
               objective = "reg:linear",
               eta=0.1,
               gamma=0,
               max_depth=7,
               min_child_weight=1,
               subsample=1,
               colsample_bytree=1)

set.seed(101)
xgb.sales.count <- xgb.train (params = params,
                        data = dtrain,
                        nrounds = 1510,
                        watchlist = list(train=dtrain,val=dtest),
                        print_every_n = 10,
                        #early_stopping_rounds = 10,
                        maximize = T ,
                        eval_metric = "rmse")
```

The validation RMSE error is around 1.4 which is not that good and the algo tends to overfit on the training data as the training loss keeps going down.

```
mat <- xgb.importance (feature_names = colnames(new_tr),model = xgb.sales.count)
xgb.plot.importance (importance_matrix = mat)
```



Feature importance shows ReleaseYear as the top feature instead of ItemCount which is surprising as it's not that evident from the correlation analysis above.

Eye balling the predictions

```
xgbpred <- predict (xgb.sales.count,dtest)

test1 <- test
test1$SoldCount <- lbl_test
test1$PredSC <- ifelse(xgbpred < 0.1, 0, xgbpred)

head(test1[,c("SoldCount","PredSC"),with=FALSE])
```

```
##    SoldCount    PredSC
## 1:  2.106947 1.3678658
## 2:  0.000000 1.2134922
## 3:  1.000000 1.9461449
## 4:  0.000000 0.2159537
## 5:  0.000000 5.4016995
## 6:  1.247429 1.1767170
```

This would need some further fine tuning and analysis, but due to lack of time I am leaving it at this point.

Predicting on Inventory Data

Now let's use the trained models to predict SaleFlag and SaleCount on the inventory data

```
### Predict Sale on the Inventory
cols = c("File_Type","Order","SKU_number","SoldCount","SoldFlag")
activeInv <- activeInv[,!cols,with=FALSE]

# Convert MarketingType to numeric
activeInv$MarketingType <- ifelse(activeInv$MarketingType == "D",1,2)

new_inv <- model.matrix(~.,data = activeInv)
dinv <- xgb.DMatrix(data = new_inv)
xgbpredInv <- predict (xgb.sales,dinv)
```

```r
xgbpredInv <- ifelse (xgbpredInv > 0.5,1,0)
activeInv$SoldFlag <- xgbpredInv
salesCount <- predict (xgb.sales.count,dinv)
activeInv$SoldCount <- ifelse(salesCount < 0.1, 0, salesCount)

invSample <- df[df$File_Type=="Active",]
invSample$SoldFlag <- activeInv$SoldFlag
invSample$SoldCount <- activeInv$SoldCount

head(invSample[,c("SKU_number","SoldFlag","SoldCount"),with=FALSE])
```

```
##    SKU_number SoldFlag SoldCount
## 1:     869734        1  13.02462
## 2:    3741319        1  16.45313
## 3:    3517789        1  15.98642
## 4:    1455936        1  12.79040
## 5:    2921480        1  16.46266
## 6:     862455        1  15.98642
```

This finishes the prediction part. Here we could predict the SaleFlag with 77% accuracy but the SaleCount still needs some improvement and work.

If a client wanted to track these insights ongoing

How would you go about designing an operational solution?

In order to build an operational solution we can have 2 possible approaches based on the frequency.

a. **Weekly/Daily/Real-time** : In this case I would prefer to build an end-to-end data pipline with this model automated and working in an unattended manner. Having said that if the metrics of the model fall below a threshold value then we would need to retrain and redeploy the model. This part of retraining can also be automated if required.

b. **Monthly/Quarterly/Yearly** : In this case we can run the model on adhoc basis manually and the data pipline can be manual as well.

On a high-level the solution would entail a data drop in form of csv file followed by some R/Python scripts for data cleaning, followed by the R/Python script for model predictions. The predictions can then be embedded in an automated pipeline or send out to people to take actions.

How would you go about estimating time/cost/skills needed to build something?

Time/Cost/Skills would be a function of what approach with take as listed above. In case of first approach, initial time/cost/skills would be higher both from data science and engineering teams. The on-going support would be less and would depend on how frequently do we need to retrain the model because of distribution shifts etc.

In case of latter approach, the initial time/cost/skills to market would be less, but it would be need periodic effort to manually churn out results.

What are some of the top challenges you would expect?

The main challenges would be as follows

a. **Data sourcing & quality**: Ensuring that there is a data pipeline in place that can generate the required input data in a consistent/accurate manner.

b. **Shifting data distribution**: There is chance that the model needs to be frequently re-trained, in case the data distribution or patterns change in production frequently. In such a case we would need to further invest time in some form of online learning infrastructure.

Appendix

In this analysis for modeling I have used gradient boosted tree's (xgboost)

Other packages used are as follows

a. data.table - This is the new, fast mechanism to read and manipulate data in R.

b. corrplot - Most popular package to plot correlation between features.

c. caret - I just used it for showing the confusion matrix.

d. DMwR - Used it for SMOTE function for up-sampling the data.

e. pROC - Used to plot the ROC curve.

f. ggplot2 - Popular plotting library in R.