Shopee_Order_Brush

Kwok Cheong, Shu Min, Amy

4/10/2022

Bring in library

```
library(foreign)
library(ggplot2)
library(dplyr)
library(e1071)
library(corrplot)
library(randomForest)
library(naivebayes)
library(caret)
library(vcd)
library(gridExtra)
library(ggeffects)
library(smotefamily)
```

Data Preprocess

```
data <- read.csv("main_data_cleaned_v4.csv", header = TRUE)

# data preprocess
data$Orderbrush <- as.factor(data$Orderbrush)
data$Verified <- as.factor(data$Verified)
data$Shop.Followers.Cat <- as.factor(data$Shop.Followers.Cat)
data$Shop.Rating <- as.factor(data$Shop.Rating)

str(data)
```

```
## 'data.frame':    16969 obs. of  15 variables:
##  $ shopid             : int  122238789 193872265 178877065 120981896 69575583 45828982 84710867 7175
##  $ Shop_name          : chr  "FILA Official Store" "Sasa Official Store" "Xiaomi Global Official Sto
##  $ Orderbrush         : Factor w/ 2 levels "No","Yes": 2 1 1 1 1 1 1 1 1 1 ...
##  $ Shop.Follower.Count: int  132751 116488 113943 105817 102473 97993 97136 93640 90425 86740 ...
##  $ Shop.Followers.Cat : Factor w/ 11 levels "<5000",">50000",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ Verified           : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Shop.Rating.Value  : num  4.91 4.92 4.87 4.9 4.94 ...
##  $ Bad.Comments       : int  30 177 323 361 500 704 195 234 196 1004 ...
##  $ Normal.Comments    : int  6776 42792 34661 42999 166004 81110 31924 64501 90312 141063 ...
##  $ Good.Comments      : int  122 549 752 662 1675 2485 513 529 922 5555 ...
```

```
##  $ Total.Comments     : int  6928 43518 35736 44022 168179 84299 32632 65264 91430 147622 ...
##  $ Shop.Rating        : Factor w/ 4 levels "Above Average",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Shop.Response.Rate : int  99 92 100 91 100 81 97 98 54 98 ...
##  $ Count.of.Order     : int  6 12 54 85 66 241 6 7 84 430 ...
##  $ Hour               : num  9.39 6.37 2.06 1.41 1.8 ...
```

```
numData <- data[,3:15]

set.seed(122)
train_index <- createDataPartition(numData$Orderbrush, p=0.75, list= FALSE)
train <- numData[train_index, ] # 75% of the data
test <- numData[-train_index, ] # 25% of the data
```

Data preprocess 2

```
train_numeric <- subset(train, select = -c(Shop.Followers.Cat, Verified, Shop.Rating))
test_numeric <- subset(test, select = -c(Shop.Followers.Cat, Verified, Shop.Rating))

train_numeric$Orderbrush <- as.numeric(train_numeric$Orderbrush) - 1

# 0 is non fraud, 1 is fraud
table(train_numeric$Orderbrush)
```

```
##
##     0     1
## 12438   290
```

```
# perform smote on train data

train_smote <- SMOTE(X = train_numeric, target=train_numeric$Orderbrush, dup_size = 10)
train_smote_data <- train_smote$data
train_smote_data$Orderbrush <- as.factor(train_smote_data$Orderbrush)
levels(train_smote_data$Orderbrush) <- c("No", "Yes")
train_smote_data <- subset(train_smote_data, select = -c(class))
table(train_smote_data$Orderbrush)
```

```
##
##    No   Yes
## 12438  3190
```

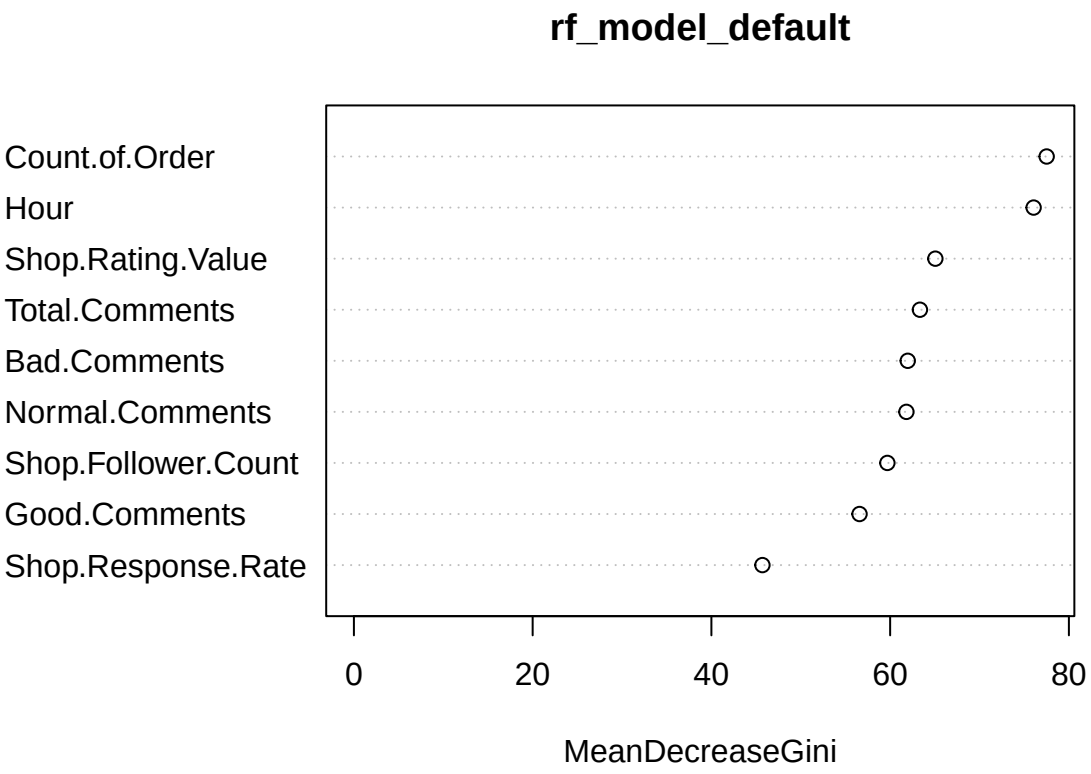
Random Forest (Default - Train test)

```
train_numeric$Orderbrush <- as.factor(train_numeric$Orderbrush)
test_numeric$Orderbrush <- as.factor(test_numeric$Orderbrush)
levels(train_numeric$Orderbrush) <- c("No", "Yes")
levels(test_numeric$Orderbrush) <- c("No", "Yes")

rf_model_default <- randomForest(formula = Orderbrush ~ ., data=train_numeric)
p_rf <- predict(rf_model_default, newdata=test_numeric)
confusionMatrix(as.factor(test_numeric$Orderbrush), p_rf ,mode = "prec_recall", positive="Yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  4142    3
##        Yes   92    4
##
##                Accuracy : 0.9776
##                  95% CI : (0.9727, 0.9818)
##     No Information Rate : 0.9983
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0748
##
##  Mcnemar's Test P-Value : <2e-16
##
##               Precision : 0.0416667
##                  Recall : 0.5714286
##                      F1 : 0.0776699
##              Prevalence : 0.0016506
##          Detection Rate : 0.0009432
##    Detection Prevalence : 0.0226362
##       Balanced Accuracy : 0.7748499
##
##        'Positive' Class : Yes
##
```

```
varImpPlot(rf_model_default)
```




```
varImp(rf_model_default)
```

```
##                       Overall
## Shop.Follower.Count  59.68872
## Shop.Rating.Value    65.05568
## Bad.Comments         61.96940
## Normal.Comments      61.81881
## Good.Comments        56.57318
## Total.Comments       63.33114
## Shop.Response.Rate   45.70774
## Count.of.Order       77.51015
## Hour                 76.05145
```

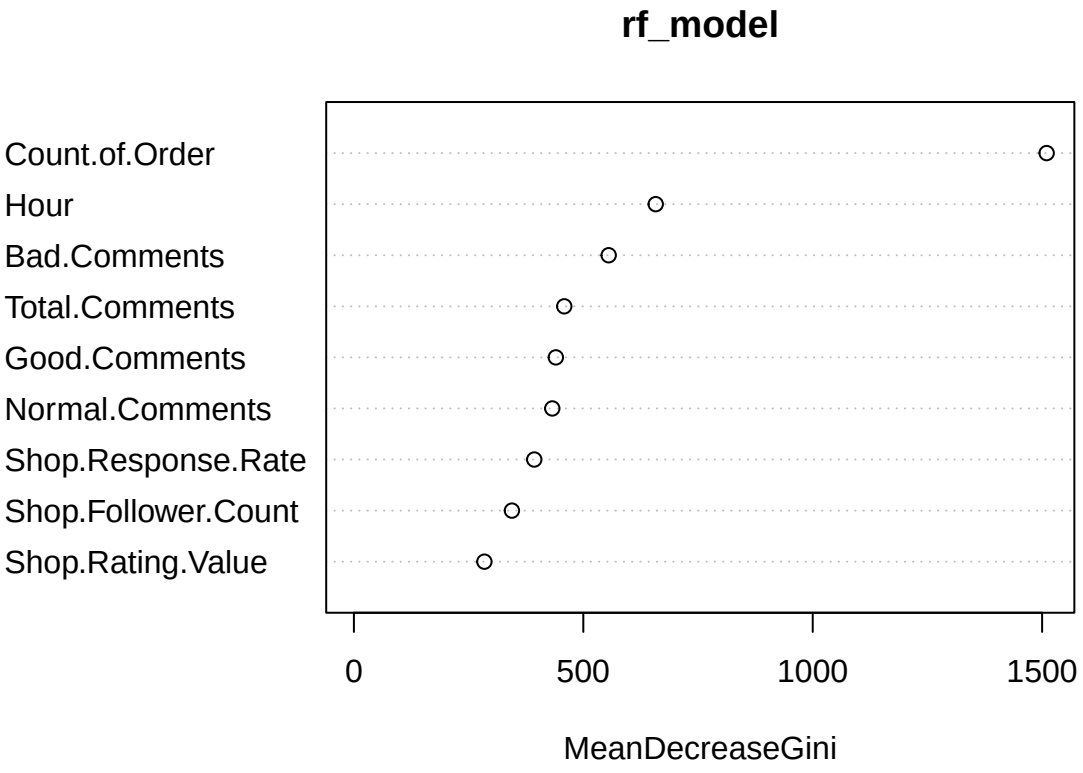
Random Forest (SMOTE)

```
rf_model <- randomForest(formula = Orderbrush ~ ., data=train_smote_data )
p_rf <- predict(rf_model, newdata=test_numeric)
confusionMatrix(test_numeric$Orderbrush, p_rf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
```

```
##        No  4027  118
##        Yes   77   19
##
##                Accuracy : 0.954
##                  95% CI : (0.9473, 0.9601)
##     No Information Rate : 0.9677
##     P-Value [Acc > NIR] : 0.999999
##
##                   Kappa : 0.1402
##
##  Mcnemar's Test P-Value : 0.004177
##
##             Sensitivity : 0.9812
##             Specificity : 0.1387
##          Pos Pred Value : 0.9715
##          Neg Pred Value : 0.1979
##              Prevalence : 0.9677
##          Detection Rate : 0.9495
##    Detection Prevalence : 0.9774
##       Balanced Accuracy : 0.5600
##
##        'Positive' Class : No
##
```

```
varImpPlot(rf_model)
```



```
varImp(rf_model)
```

```
##                       Overall
## Shop.Follower.Count  344.3633
## Shop.Rating.Value    284.3881
## Bad.Comments         555.2905
## Normal.Comments      432.3778
## Good.Comments        440.0902
## Total.Comments       458.4793
## Shop.Response.Rate   393.0763
## Count.of.Order      1509.8510
## Hour                 657.8308
```

Random Forest (With undersample and preprocessing)

```
# RF - CARET
ctrl <- trainControl(method = "repeatedcv",
                     number = 10,
                     repeats = 10,
                     verboseIter = FALSE,
                     sampling = "down")

random_f_model <- caret::train(Orderbrush ~ .,
                                    data = train,
                                    method = "rf",
                                    preProcess = c("scale", "center"),
                                    trControl = ctrl)

p_rf2 <- predict(random_f_model, newdata=test)
confusionMatrix(as.factor(test$Orderbrush), p_rf2, mode = "prec_recall", positive="Yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  3458  687
##        Yes   31   65
##
##                Accuracy : 0.8307
##                  95% CI : (0.8191, 0.8419)
##     No Information Rate : 0.8227
##     P-Value [Acc > NIR] : 0.08843
##
##                   Kappa : 0.1179
##
##  Mcnemar's Test P-Value : < 2e-16
##
##               Precision : 0.67708
##                  Recall : 0.08644
##                      F1 : 0.15330
##              Prevalence : 0.17732
```

```
##          Detection Rate : 0.01533
##    Detection Prevalence : 0.02264
##       Balanced Accuracy : 0.53878
##
##        'Positive' Class : Yes
##
```

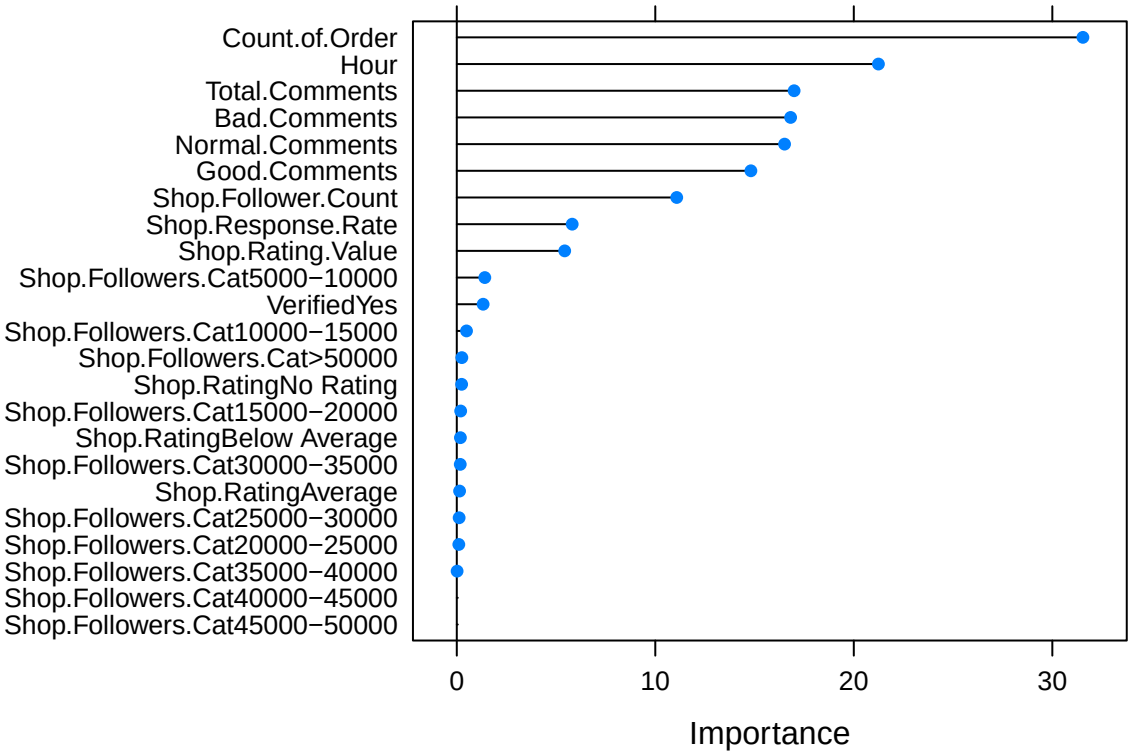
```
table(test$Orderbrush)
```

```
##
##   No  Yes
## 4145   96
```

```
varImp(random_f_model, scale=FALSE)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 23)
##
##                               Overall
## Count.of.Order                31.5416
## Hour                          21.2409
## Total.Comments                16.9935
## Bad.Comments                  16.8169
## Normal.Comments               16.5124
## Good.Comments                 14.8155
## Shop.Follower.Count           11.0803
## Shop.Response.Rate             5.8143
## Shop.Rating.Value              5.4366
## Shop.Followers.Cat5000-10000   1.4118
## VerifiedYes                    1.3272
## Shop.Followers.Cat10000-15000  0.4915
## Shop.Followers.Cat>50000       0.2561
## Shop.RatingNo Rating           0.2451
## Shop.Followers.Cat15000-20000  0.1961
## Shop.RatingBelow Average       0.1839
## Shop.Followers.Cat30000-35000  0.1754
## Shop.RatingAverage             0.1420
## Shop.Followers.Cat25000-30000  0.1189
## Shop.Followers.Cat20000-25000  0.1023
```

```
plot(varImp(random_f_model, scale=FALSE))
```



Data Hyperparameter tuning

Naive Bayes