INSTACART MARKET BASKET ANALYSIS

MICHAEL AJITH

Introduction:

Instacart is an app for on-demand grocery shopping with same-day delivery service. Instacart uses a crowdsourced marketplace model, akin to that of Uber or Lyft.

The Instacart shopping process is as follows. First, an app user places their grocery order through the app. Then, a locally crowdsourced “shopper” is notified of the order, goes to a nearby store, buys the groceries, and delivers them to the user.

There are three ways that Instacart generates revenue: delivery fees, membership fees, and mark-ups on in-store prices.

Objective:

The objective of this Kaggle competition/project is to use the anonymized data on customer orders over time to predict which previously purchased products will be in a user’s next order

Data Source:

Instacart released a public dataset, **"The Instacart Online Grocery Shopping Dataset 2017"**. The dataset contains over 3 million anonymized grocery orders from more than 200,000 Instacart users. This analysis will make use of this datasets.

Data Dictionary:

The dataset for this competition is a relational set of files describing customers' orders over time. They are anonymized and contains a sample of over **3 million grocery orders** from more than **200,000 Instacart users**. For each user, Instacart provided **between 4 and 100** of their orders, with the sequence of products purchased in each order, the **week and hour of day** the order was placed, and a **relative measure of time between orders**.

Total six datasets were imported. Follwing section will explore each datasets in further detail. These datasets were sourced from an existing Kaggle competition.

orders (3.4m rows, 206k users):

- order_id: order identifier
- user_id: customer identifier
- eval_set: which evaluation set this order belongs in (see SET described below)
- order_number: the order sequence number for this user (1 = first, n = nth)
- order_dow: the day of the week the order was placed on
- order_hour_of_day: the hour of the day the order was placed on
- days_since_prior: days since the last order, capped at 30 (with NAs for order_number = 1)

products (50k rows):

- product_id: product identifier
- product_name: name of the product
- aisle_id: foreign key
- department_id: foreign key

aisles (134 rows):

- aisle_id: aisle identifier
- aisle: the name of the aisle

deptartments (21 rows):

- department_id: department identifier
- department: the name of the department

order_products__SET (30m+ rows):

- order_id: foreign key
- product_id: foreign key
- add_to_cart_order: order in which each product was added to cart
- reordered: 1 if this product has been ordered by this user in the past, 0 otherwise

Understanding Datasets:

Aisles

There are **134 ailes** in this dataset. Here are few sample names of the ailes.

Departments

There are **21 departments** in this dataset.Names of all deparments are listed below in aphabetically ordered.

Products

There are **49,688 products** in the catalogue within **134 aisles and 21 departments**.

Solution Approach:

At high level, the idea is to come up with features related to reorder metrics for individual products as well as for user preferences for products. Also, we create orders based on the orders of the users.

E.g Everyone likes say an iphone, so the individual product's metrics should capture the quality of a product getting reordered on it's merit. Then, Mr.X likes some exotic food rarely liked by anyone else, the reorder metrics for Mr. X should capture this liking and the reordering metrics specific to him. Other metrics based on orders placed would include frequency of ordering etc., time (day of week/hour of the day) preferences etc.

Once the features are created a denormalized structure is created with the keys as the user id and product id. The problem then becomes a CLASSIFICATION problem that needs to be solved using a classfier algorithm. We choose XGBoost as the classifier

R SCRIPT/CODING:

```
#Instacart Market Basket Analysis


#Loading the libraries

library(data.table)

library(dplyr)

library(tidyr)


#setting the directory

setwd("D:/MBA REF/edwisor/Kaagle")

# Loading the datasets


aisles <- read.csv("aisles.csv")

departments <- read.csv("departments.csv")

orderp <- read.csv("order_products__prior.csv")

ordert <- read.csv("order_products__train.csv")

orders <- read.csv("orders.csv")

products <- read.csv("products.csv")




# Reshape data

aisles$aisle <- as.factor(aisles$aisle)
```

```
departments$department <- as.factor(departments$department)
orders$eval_set <- as.factor(orders$eval_set)
products$product_name <- as.factor(products$product_name)

products <- products %>%
inner_join(aisles) %>% inner_join(departments) %>%
select(-aisle_id, -department_id)
rm(aisles, departments)

ordert$user_id <- orders$user_id[match(ordert$order_id, orders$order_id)]

orders_products <- orders %>% inner_join(orderp, by = "order_id")

rm(orderp)
gc()


# Products
prd <- orders_products %>%
arrange(user_id, order_number, product_id) %>%
group_by(user_id, product_id) %>%
mutate(product_time = row_number()) %>%
ungroup() %>%
group_by(product_id) %>%
```

```
summarise(
prod_orders = n(),
prod_reorders = sum(reordered),
prod_first_orders = sum(product_time == 1),
prod_second_orders = sum(product_time == 2)
)

prd$prod_reorder_probability <- prd$prod_second_orders / prd$prod_first_orders
prd$prod_reorder_times <- 1 + prd$prod_reorders / prd$prod_first_orders
prd$prod_reorder_ratio <- prd$prod_reorders / prd$prod_orders

prd <- prd %>% select(-prod_reorders, -prod_first_orders, -prod_second_orders)

rm(products)
gc()

# Users
users <- orders %>%
filter(eval_set == "prior") %>%
group_by(user_id) %>%
summarise(
user_orders = max(order_number),
user_period = sum(days_since_prior_order, na.rm = T),
user_mean_days_since_prior = mean(days_since_prior_order, na.rm = T)
```

```
)

us <- orders_products %>%
group_by(user_id) %>%
summarise(
user_total_products = n(),
user_reorder_ratio = sum(reordered == 1) / sum(order_number > 1),
user_distinct_products = n_distinct(product_id)
)

users <- users %>% inner_join(us)
users$user_average_basket <- users$user_total_products / users$user_orders

us <- orders %>%
filter(eval_set != "prior") %>%
select(user_id, order_id, eval_set,
time_since_last_order = days_since_prior_order)

users <- users %>% inner_join(us)

rm(us)
gc()
```

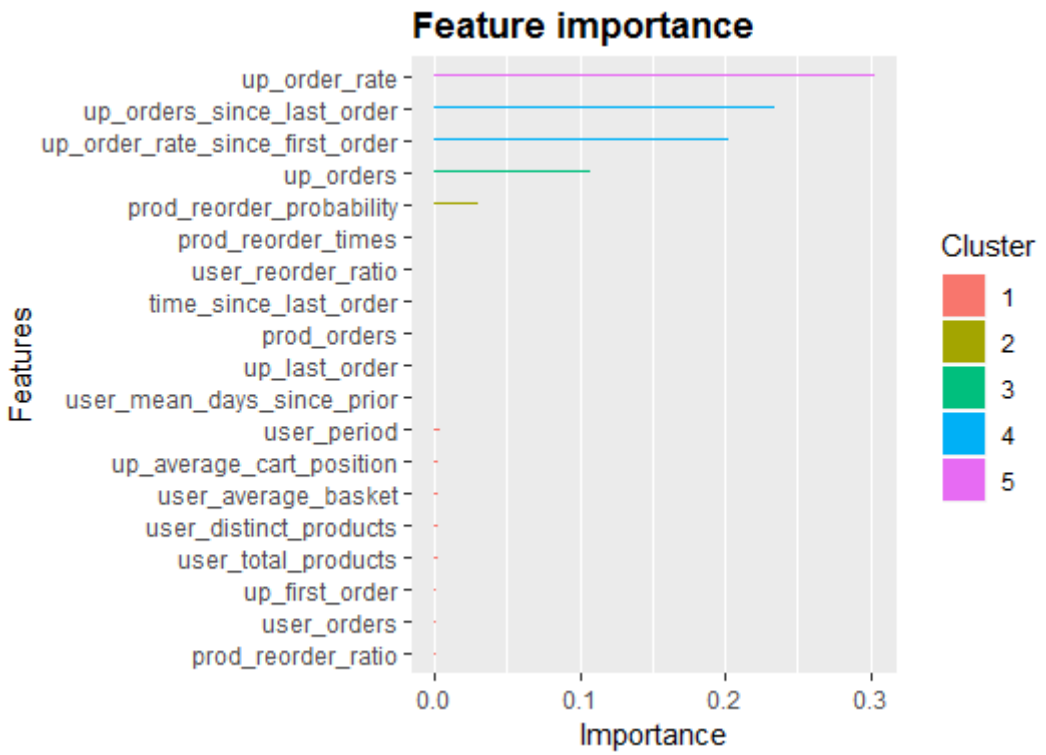
```
# Database

data <- orders_products %>%

group_by(user_id, product_id) %>%

summarise(

up_orders = n(),

up_first_order = min(order_number),

up_last_order = max(order_number),

up_average_cart_position = mean(add_to_cart_order))


rm(orders_products, orders)


data <- data %>%

inner_join(prd, by = "product_id") %>%

inner_join(users, by = "user_id")


data$up_order_rate <- data$up_orders / data$user_orders

data$up_orders_since_last_order <- data$user_orders - data$up_last_order

data$up_order_rate_since_first_order <- data$up_orders / (data$user_orders -
data$up_first_order + 1)


data <- data %>%

left_join(ordert %>% select(user_id, product_id, reordered),

by = c("user_id", "product_id"))


rm(ordert, prd, users)
```

```
gc()



# Train / Test datasets
train <- as.data.frame(data[data$eval_set == "train",])
train$eval_set <- NULL
train$user_id <- NULL
train$product_id <- NULL
train$order_id <- NULL
train$reordered[is.na(train$reordered)] <- 0

test <- as.data.frame(data[data$eval_set == "test",])
test$eval_set <- NULL
test$user_id <- NULL
test$reordered <- NULL

rm(data)
gc()



# Model
library(xgboost)

params <- list(
```

```
"objective"           = "reg:logistic",

"eval_metric"         = "logloss",

"eta"                 = 0.1,

"max_depth"           = 6,

"min_child_weight"    = 10,

"gamma"               = 0.70,

"subsample"           = 0.76,

"colsample_bytree"    = 0.95,

"alpha"               = 2e-05,

"lambda"              = 10

)



subtrain <- train %>% sample_frac(0.1)

X <- xgb.DMatrix(as.matrix(subtrain %>% select(-reordered)), label =
subtrain$reordered)

model <- xgboost(data = X, params = params, nrounds = 80)



importance <- xgb.importance(colnames(X), model = model)

xgb.ggplot.importance(importance)



rm(X, importance, subtrain)

gc()



# Apply model
```

```
X <- xgb.DMatrix(as.matrix(test %>% select(-order_id, -product_id)))

test$reordered <- predict(model, X)


test$reordered <- (test$reordered > 0.21) * 1


submission <- test %>%

filter(reordered == 1) %>%

group_by(order_id) %>%

summarise(

products = paste(product_id, collapse = " ")

)


missing <- data.frame(

order_id = unique(test$order_id[!test$order_id %in% submission$order_id]),

products = "None"

)


submission <- submission %>% bind_rows(missing) %>% arrange(order_id)

write.csv(submission, file = "submit.csv", row.names = F)
```

Important Features:



Conclusion:

As I used XGBoost Classifier algorithm to predict which previously purchased products will be in a user's next order. The result is in the csv file which holds the order_id and products_id provides predictions