CASE STUDY REPORT

Group No.: Group 21

Student Names: Chiung-Chih Lin, Wenyi Jin

Executive Summary

The goal of the study is to predict whether a product purchased by an Instacart customer before will be appeared in his next order. The original data is gained from Instacart blog. The dataset is anonymized and contains over 3 million grocery orders from more than 200,000 Instacart users. We selected five variables to be the predictor and leverage KNN, logistic regression, SVM and neural network algorithms to build the model for this classification problem. The SVM has the highest accuracy - 65.35% and KNN has the lowest - 56.10%. However, based on the consideration of accuracy and time of execution, we recommended the neural network model. Neural network model has accuracy of 65.15%, 0.2% lower than SVM but the computational time is much shorter than SVM.

I. Introduction

Online food delivery services rely on urban transportation to alleviate customers' burden of traveling in highly dense cities. As new business models, these services exploit user-generated contents to promote collaborative consumption among its members. Instacart, a grocery ordering and delivery app, aims to make it easy to fill consumers' refrigerators and pantry with their personal favorites and staples when they need them. After selecting products through the Instacart app, personal shoppers review the order and do the in-store shopping and delivery for.

The market competition becomes more and more fierce so it is significant to improve user experience for customers. Our goal is to use transactional data to develop models that predict whether a user will buy a product again during a session.

On its website Instacart has a recommendation feature, suggesting the users some items that he/she may buy again. Our task is to predict which items will be reordered on the next order.

For this classification problem, we planned to apply KNN, logistic regression, SVM and neural network algorithms to build the model.

The dataset consists of information about 3.4 million grocery orders across 5 csv files, namely orders, order_products, departments, aisles and products. The orders files
give a list of all orders we have in the dataset, 1 row per order. For example, we can see that user 1 has 11 orders. The latest order would be placed in the train set and the

previous ten orders are prior orders. The orders.csv doesn't tell us about which products were ordered. This is contained in the order_products.csv.

order_id	user_id	eval_set	order_number	order_dow	order_hour_of_day	days_since_prior_order
2539329	1	prior	1	2	8	NA
2398795	1	prior	2	3	7	15
473747	1	prior	3	3	12	21
2254736	1	prior	4	4	7	29
431534	1	prior	5	4	15	28
3367565	1	prior	6	2	7	19
550135	1	prior	7	1	9	20
3108588	1	prior	8	1	14	14
2295261	1	prior	9	1	16	0
2550362	1	prior	10	4	8	30
1187899	1	train	11	4	8	14
2168274	2	prior	1	2	11	NA

The order_product.csv gives us information about which products (product_id) were ordered. It also contains information of the order (add_to_cart_order) in which the products were put into the cart and information of whether this product is reordered(1) or not(0). For example, we see below that order_id 1 had 8 products, 4 of which are reorders. Still we don't know what these products are. This information is in the products.csv.

order_id	product_id	add_to_cart_order	reordered
1	49302	1	1
1	11109	2	1
1	10246	3	0
1	49683	4	0
1	43633	5	1
1	13176	6	0
1	47209	7	0
1	22035	8	1
36	39612	1	0
36	19660	2	1

The product.csv contains the names of the products with their corresponding product_id. Furthermore, the aisle and department are included.

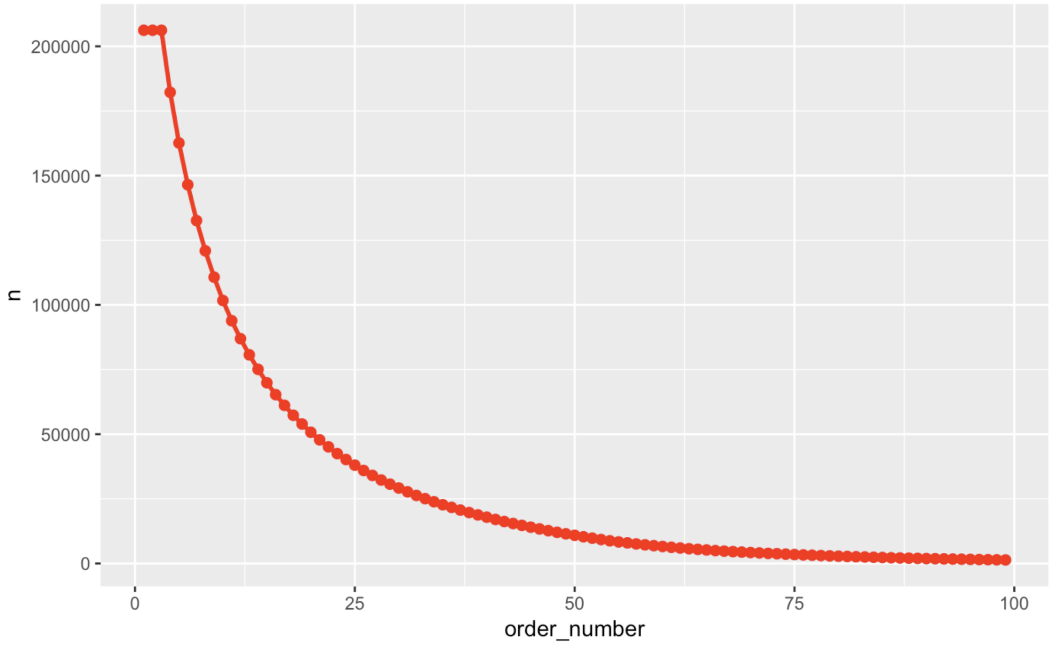
product_id	product_name	aisle_id	department_id
1	Chocolate Sandwich Cookies	61	19
2	All-Seasons Salt	104	13
3	Robust Golden Unsweetened Oolong Tea	94	7
4	Smart Ones Classic Favorites Mini Rigatoni With Vodka Cream Sauce	38	1
5	Green Chile Anytime Sauce	5	13
6	Dry Nose Oil	11	11
7	Pure Coconut Water With Orange	98	7
8	Cut Russet Potatoes Steam N' Mash	116	1
9	Light Strawberry Blueberry Yogurt	120	16
10	Sparkling Orange Juice & Prickly Pear Beverage	115	7

The aisle.csv and department.csv contain the name of the aisles and departments.

II. Data Exploration and Visualization

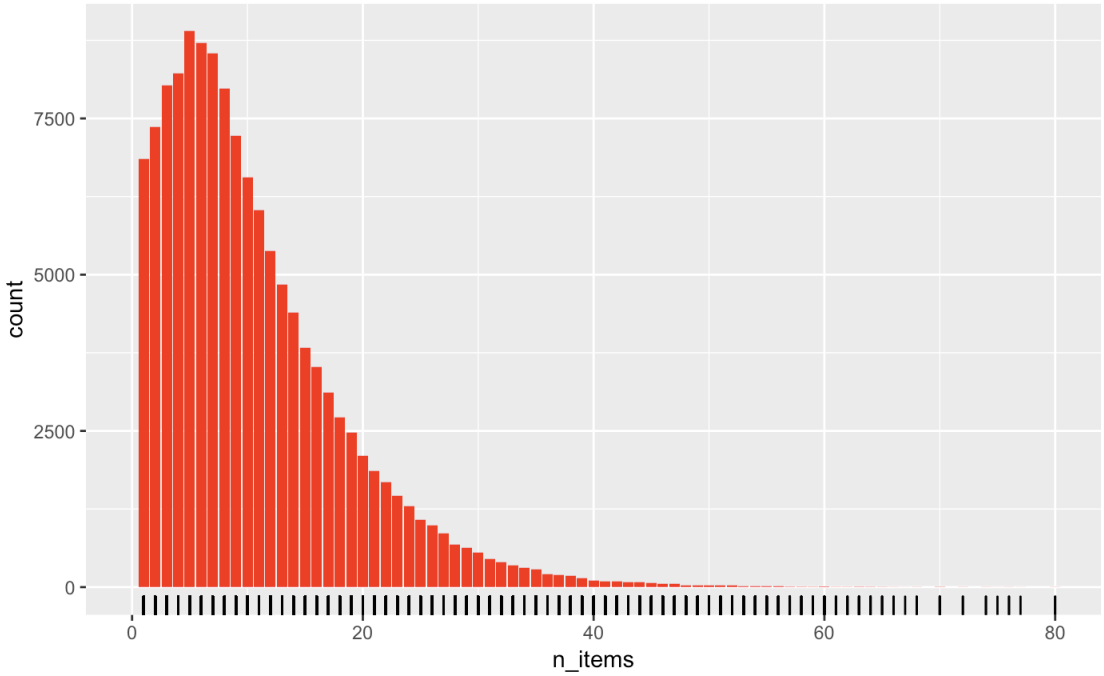
During the data exploration process, we found that we could use data to answer following questions.

- **How many prior orders are there?**
  We can see that there are always at least 3 prior orders.



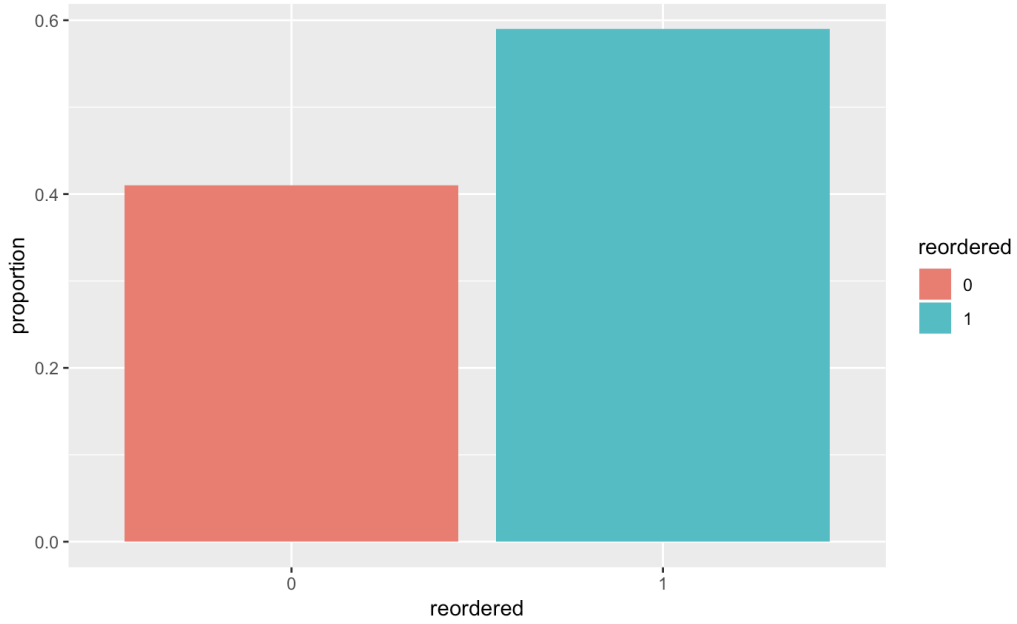
- **How many items do people buy?**

  Let's have a look how many items are in the orders. We can see that people most often order around 5 items.



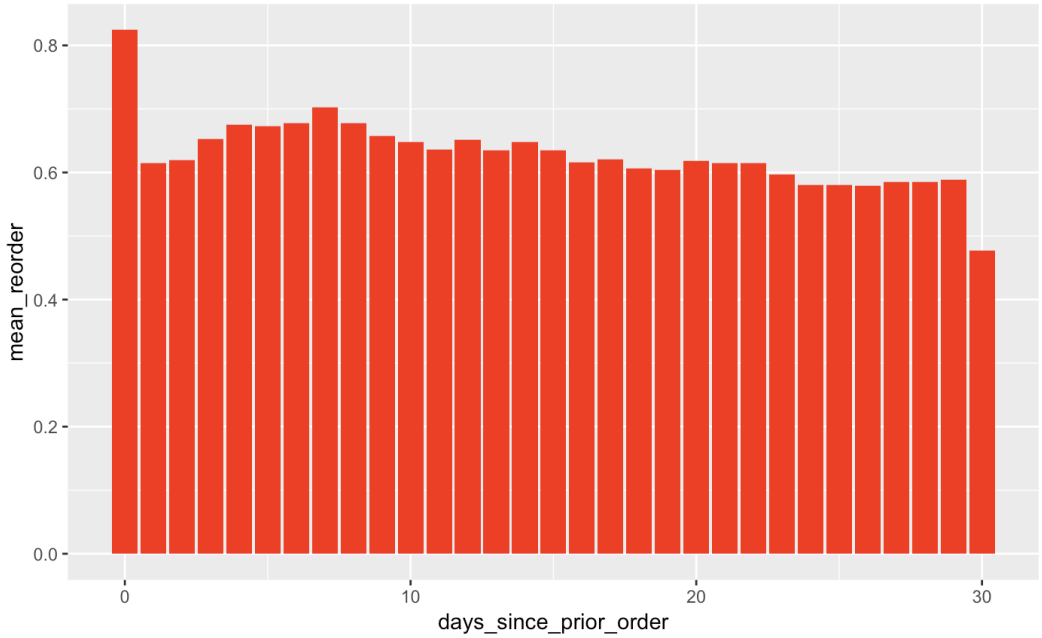
- **How often do people order the same items again?**

  59% of the ordered items are reorders.



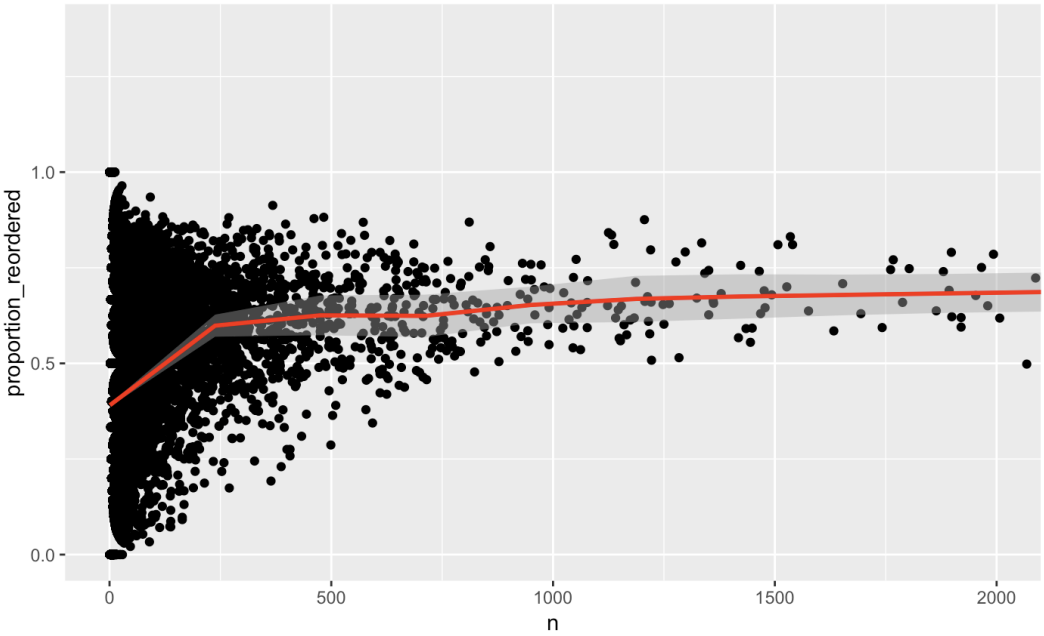
- **Association between time of last order and probability of reorder**

  This is interesting: We can see that if people order again on the same day, they order the same product more often. Whereas when 30 days have passed, they tend to try out new things in their order.



- **Association between number of orders and probability of reordering**

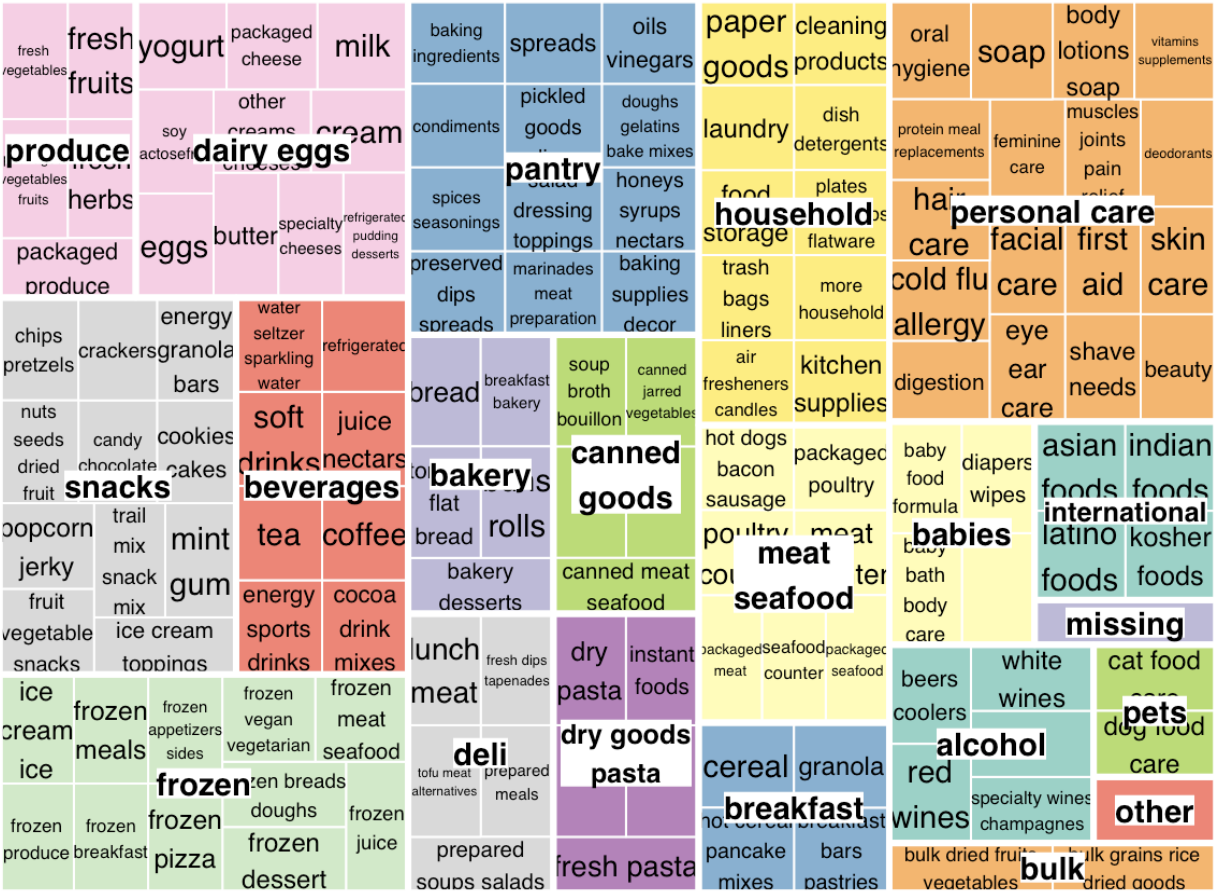
  Products with a high number of orders are naturally more likely to be reordered. However, there seems to be a ceiling effect.



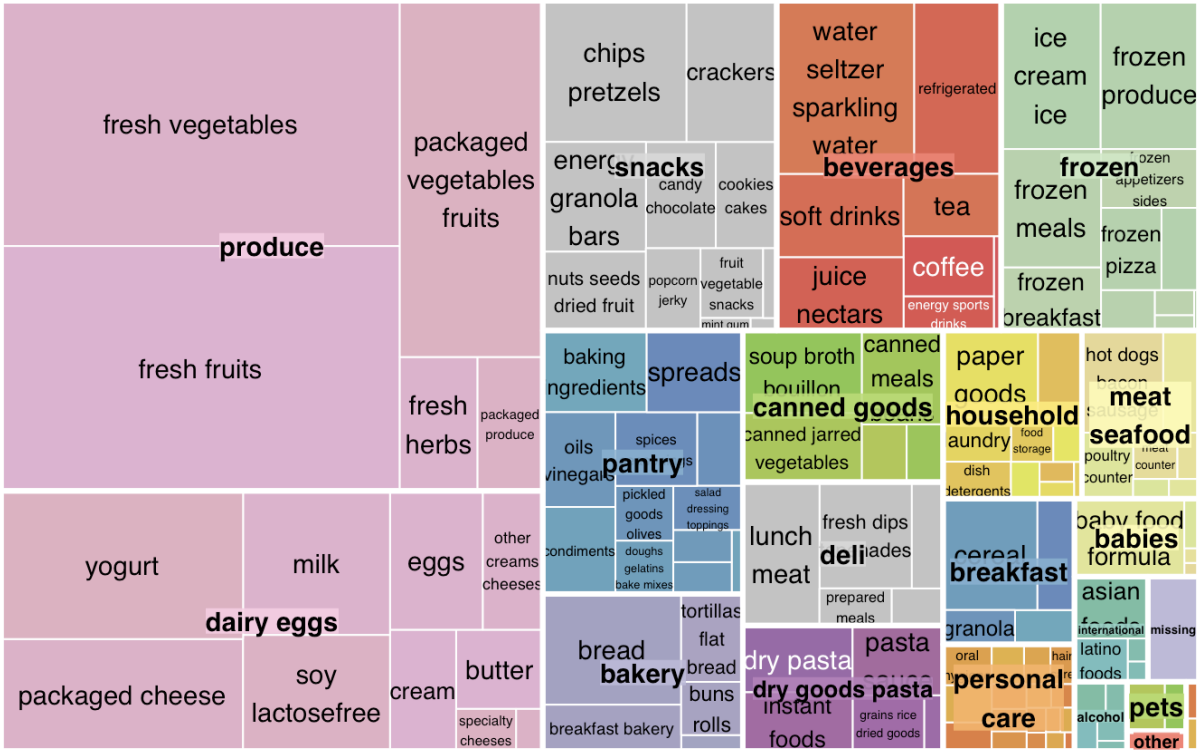
Visualizing the Product Portfolio

Here we used treemap package to visualize the structure of instacarts product portfolio. In total, there are 21 departments containing 134 aisles.

- **The treemap shows how the aisles organized within departments.**



- **How often are products from the department/aisle sold?**
  The size of the boxes shows the number of sales. The products from produce department sold the most.



Exploring Customer Habits

Here we look for customers who just reorder the same products again all the time. To search those, we look at all orders (excluding the first order), where the percentage of reordered items is exactly 1 (This can easily be adapted to look at more lenient thresholds). We can see there are in fact 3,487 customers who always reordering products.

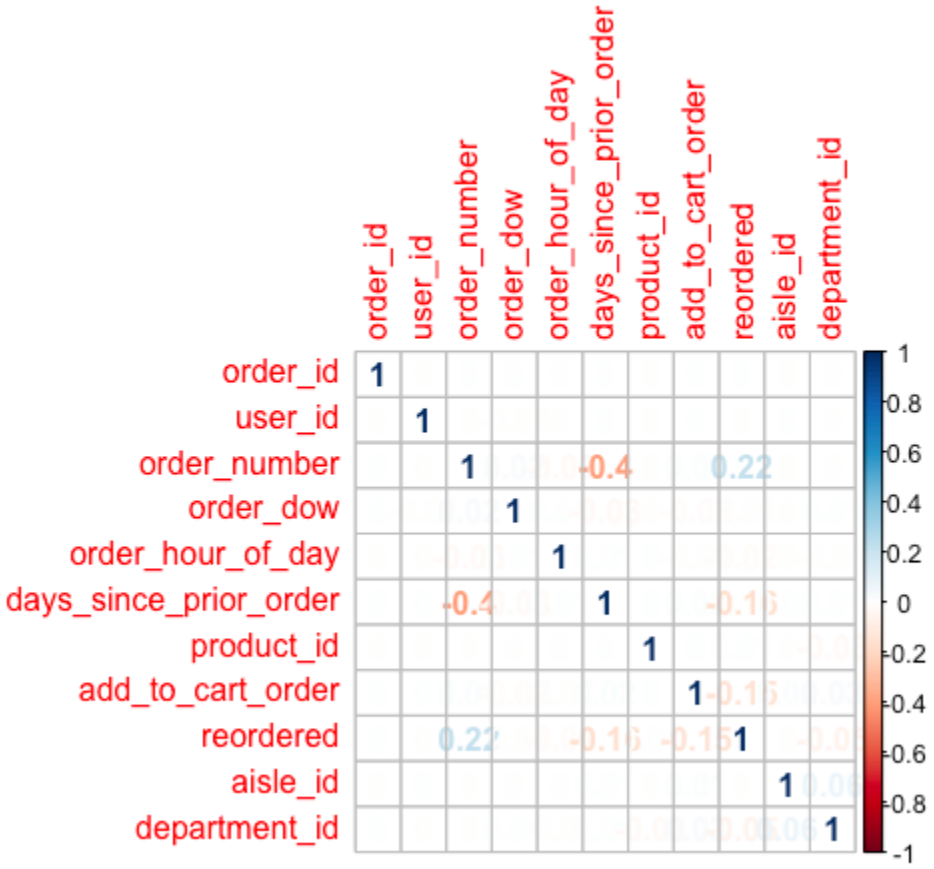
	user_id	n_equal	percent_equal
1	99753	97	1
2	55331	49	1
3	106510	49	1
4	111365	47	1
5	74656	46	1
6	170174	45	1
7	12025	43	1
8	164779	37	1
9	37075	34	1
10	110225	33	1

III. Data Preparation and Preprocessing

We joined all the tables during the preprocessing phrase. NA values are only a small portion of our over-three-million-record dataset so we removed all the missing values.

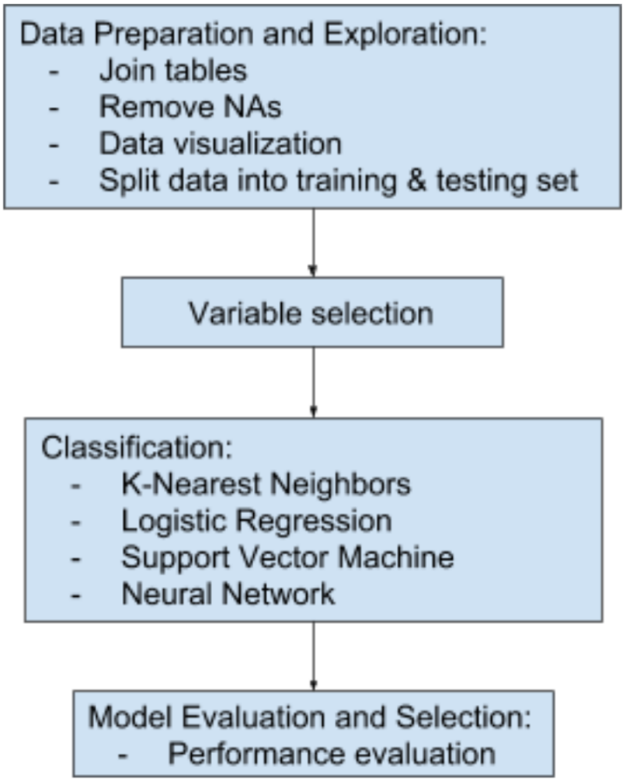
According to the correlation matrix, we can tell that all the numeric variables are not highly correlated. Besides, the dataset doesn't contain many variables so we did not apply PCA to do the dimension reduction.

The categorical variables, department name and aisle name, have corresponding variable - id to uniquely identify so there is no need to create dummy variables and we just removed the name variables. We randomly retrieved 100,000 records for training and testing purposes.



IV. Data Mining Techniques and Implementation

We selected five predictors for this classification problem. The outcome is a variable named reordered. The reordered variable is set to 1 if the product is predicted to be reordered by the customer and 0 otherwise.



V. Performance Evaluation

Based on the consideration of accuracy and the time of execution, we picked neural network model to be our best approach.

Data Mining Techniques	Evaluation Performance (Accuracy)	Computational Time
KNN	56.10%	2 minutes
Logistic Regression	64.22%	5 seconds
SVM	65.35%	10 minutes
Neural Network	65.15%	10 seconds

Classification Matrix of KNN

Total Observations in Table: 100000

prc_test_pred	test$reordered 0	1	Row Total
0	9005	12662	21667
	9.380	6.317	
	0.416	0.584	0.217
	0.224	0.212	
	0.090	0.127	
1	31236	47097	78333
	2.595	1.747	
	0.399	0.601	0.783
	0.776	0.788	
	0.312	0.471	
Column Total	40241	59759	100000
	0.402	0.598	

Classification Matrix of Logistic Regression

Total Observations in Table: 100000

predict_result_class2	test$reordered 0	1	Row Total
0	11911	7454	19365
	2176.487	1465.621	
	0.615	0.385	0.194
	0.296	0.125	
	0.119	0.075	
1	28330	52305	80635
	522.697	351.978	
	0.351	0.649	0.806
	0.704	0.875	
	0.283	0.523	
Column Total	40241	59759	100000
	0.402	0.598	

Classification Matrix of SVM

Total Observations in Table: 100000

pred	test$reordered 0	1	Row Total
0	13685	8095	21780
	2762.445	1860.198	
	0.628	0.372	0.218
	0.340	0.135	
	0.137	0.081	
1	26556	51664	78220
	769.190	517.963	
	0.340	0.660	0.782
	0.660	0.865	
	0.266	0.517	
Column Total	40241	59759	100000
	0.402	0.598	

Classification Matrix of Neural Network

Total Observations in Table: 100000

pred2	test$reordered 0	1	Row Total
0	13272	7877	21149
	2663.891	1793.832	
	0.628	0.372	0.211
	0.330	0.132	
	0.133	0.079	
1	26969	51882	78851
	714.495	481.132	
	0.342	0.658	0.789
	0.670	0.868	
	0.270	0.519	
Column Total	40241	59759	100000
	0.402	0.598	

VI. Discussion and Recommendation

KNN is a simple model and it is effective at capturing complex interactions among variables without having to define a statistical model. However, it might be time consuming as in a large training set as it takes a long time to find distances to all the neighbors and then identifies the nearest one. In our case, it took about 2 minutes to run the model and get the result. The model is computationally expensive and requires high memory as the algorithm store all of the training data.

Logistic regression extends the idea of linear regression to the situation where outcome variable is categorical and it is binary in our case. The model needs two steps. First, it estimates of the probabilities of belonging to each class. Then, it uses a cutoff value on these probabilities in order to classify each case in one of the classes. We set the cutoff value to 0.5 for this study. Although it doesn't achieve the highest accuracy rate among four models, it has good performance when it comes to the time of execution. It costed us less than a minute to run the mode.

SVM is effective in cases where the number of dimensions is greater than the number of samples but it doesn't perform well when we have large data set because the required training time is higher so we won't use all the data.

Neural network model is the one we recommended to consider the accuracy and time of execution. The network architecture has the input layer, the hidden layer which can be more than one and the output layer. The feature allows it to combine input information in a flexible way that captures complicated relationships among variables. Moreover, the hidden layer can be seen as a "distillation layer" that distills some of the important patterns from the input and passes it onto the next layer to see. It makes the network faster and efficient by identifying only the important information from the inputs leaving out the redundant information. It costed us less than a minute the run the model.

The highest accuracy rate we achieved is not so ideal. One of the potential improvements we could make for further study is to collect more related data. There are only a few variables in our original data. For instance, we could capture the gender, age and location of customers. Besides, we could try changing the cutoff value in the logistic regression model.

VII. Summary

Given the dataset and the problem of whether an Instacart customer is going to reorder a product, we recommended the neural network model. The SVM has the highest accuracy - 65.35% and KNN has the lowest - 56.10%. However, SVM model is very time consuming as it took more than 10 minutes while the neural network model runs much faster with only 0.2% accuracy lower.

Appendix: R Code for use case study

```
# Load Data
orders <- fread("/Users/linlcc/Downloads/Data Mining/Case Study/all/orders.csv")
departments <- (fread("/Users/linlcc/Downloads/Data Mining/Case
Study/all/departments.csv"))
products <- (fread("/Users/linlcc/Downloads/Data Mining/Case Study/all/products.csv"))
order_products__prior <- (fread("/Users/linlcc/Downloads/Data Mining/Case
Study/all/order_products__prior.csv"))
order_products_train <- fread("/Users/linlcc/Downloads/Data Mining/Case
Study/all/order_products__train.csv")
aisles <- (fread("/Users/linlcc/Downloads/Data Mining/Case Study/all/aisles.csv"))
samplesub<-(fread("/Users/linlcc/Downloads/Data Mining/Case
Study/all/sample_submission.csv"))
orders <- orders %>% mutate(order_hour_of_day = as.numeric(order_hour_of_day),
eval_set = as.factor(eval_set))
products <- products %>% mutate(product_name = as.factor(product_name))
aisles <- aisles %>% mutate(aisle = as.factor(aisle))
departments <- departments %>% mutate(department = as.factor(department))

# Data Preparation and Preprocessing
set.seed(100)
train <- inner_join(orders,order_products__prior)
train <- inner_join(train, products)
train<- train[c(2,4,7,8,9,10)]
train <- na.omit(train)
train <- sample_n(train, 100000)

test <- inner_join(orders,order_products_train)
test <- inner_join(test, products)
test<- test[c(2,4,7,8,9,10)]
test <- na.omit(test)
test <- sample_n(test, 100000)

# Transforming the dependent variable to a factor
train$reordered = as.factor(train$reordered)
# Setting levels for both training and validation data
levels(train$reordered) <- levels(factor(train$reordered))
levels(test$reordered) <- levels(factor(test$reordered))

# KNN algorithm
prc_test_pred <- knn(train = train[,1:5], test = test[,1:5] ,cl = train[,6], k=10)
DT<- table(prc_test_pred, test[,6])
CrossTable(prc_test_pred, test$reordered)
cat("k=:",10,"\n")
print(DT)
```

```
accuracy<- (DT[1]+DT[4])/(DT[1]+DT[2]+DT[3]+DT[4])
cat("Accuracy=:",accuracy*100,"%","\n")

#Logistic regression
logit.reg <- glm(reordered~ ., data = train, family = "binomial")
predict_result2 <- predict(logit.reg, test, type="response")
predict_result_class2<-ifelse(predict_result2>=0.5,1,0)
CrossTable(predict_result_class2, test$reordered)
cat("Accuracy=:",(1-mean(predict_result_class2!=test$reordered))*100,"%","\n")

#SVM
pred=data.frame
fit_svm <- svm(factor(reordered)~ . ,data=train)
pred <- predict(fit_svm, test,type="class")
CrossTable(pred, test$reordered)
cat("Accuracy=:",(1-mean(pred!=test$reordered))*100,"%","\n")

#Neural network
fit_nn <- nnet(factor(reordered) ~ . , data=train,size=10)
pred2 <- predict(fit_nn, test)
pred2 <-ifelse(pred2>0.5,1,0)
CrossTable(pred2, test$reordered)
cat("Accuracy=:",(1-mean(pred2!=test$reordered))*100,"%","\n")
```