RECOMMENDER SYSTEM FOR INSTACART

A project carried out in partial fulfilment of the IDAF Data Science Fellowship

Toluwalase Okuwoga

Princewill Okechukwu

Olatunde Olanayo

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Instacart, the company that half of American households use to get groceries from their favorite grocery stores like Petco, Costco, Whole Foods, Target and many more without even leaving their house. With its headquarters in San Francisco, Instacart operates in 5,500 cities in all 50 states in America and in some Canadian provinces. The company's unique business model makes it a great case study on how data science can be used in the industry to tackle rocky business problems.

Instacart's customers and new users can access its platform using a mobile and the Instacart website.

Team 6 was chosen to do a project on building a recommender system for Instacart by making recommendations of products to users on its website as a targeted marketing tool. The recommendation algorithms are used to personalize the online store for each customer based on products that the customer likes and is most likely to purchase.

BUSINESS UNDERSTANDING

In order for us to build a recommender system for Instacart, we need to have a good understanding of what the Instacart business is all about. This process will help us to understand the importance of a recommender tool and how it will impact the business as a whole.

Online shopping is one of the biggest changes to happen in the commerce sector this past decade. It is convenient because people can shop from anywhere and at any time they choose while saving time and effort. Customers are also exposed to a wider range of products than if they were to shop physically from store to store. They can get detailed information about each product without having to deal with the pressure from sales representatives trying to influence their decisions.

We have learned that Instacart is one of the bigger online grocery stores that allow customers to order from shops in their neighborhood (usually at the same prices). Orders are fulfilled and delivered by an Instacart personal shopper, who picks, packs, and delivers the order within the customer's designated time frame—within one hour or up to five days in advance. Customers pay with personal debit or credit cards, Google Pay and Apple Pay. For orders of $35 or more, the delivery fee is $3.99, plus a 5% service fee. With an Instacart Express membership for $9.99/month or an annual fee of $99, customers get waived delivery fees on orders over $35, but still must pay the service fee. Customers are also requested to leave a gratuity. Retailers set the price of individual items on the Instacart marketplace, which are mostly the same prices as in-store. Instead of delivery, using the Instacart Pickup option, customers can pick up their packaged orders from the store.

A 2019 study done by NERA Economic Consulting determined that Instacart was responsible for more than 23,000 jobs across the 4 states of California, Illinois, Washington, and New York. This increase was referred to as the "Instacart Effect". As of its most recent funding round, in November 2018, the company was valued at $7.87 billion.

INSTACART'S BUSINESS MODEL

1. Download the Instacart App.
2. Choose a store. There's lots of stores — Whole Foods, Costco, Petco, Wegman's, etc.
3. From the store's inventory, select which items you want to add to your cart.
4. Select a one-hour time window during which you want your groceries to be delivered, which could be within the next hour.
5. Within one hour, your groceries are at your front door. The UX is incredibly simple and easy to navigate.

Graphic by Andre Ye

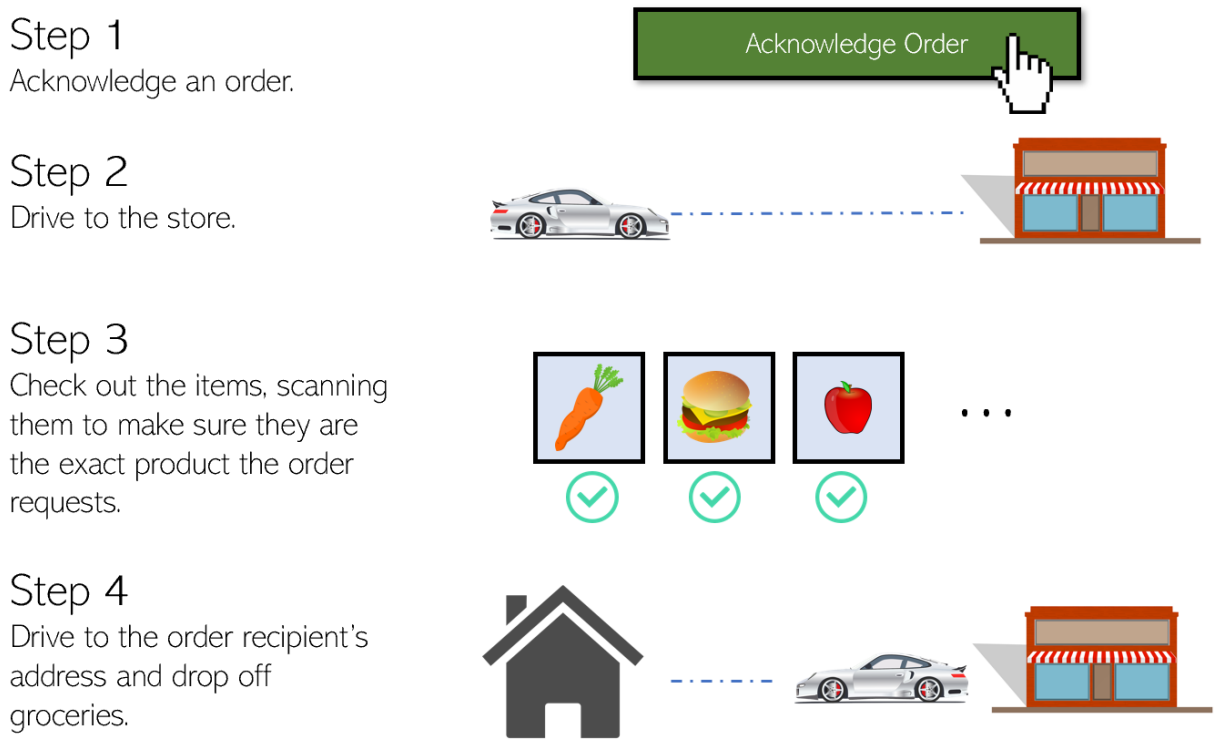
The shopper's experience is a little bit more complicated.

1. In the Instacart app, the shopper is on their shift, and are given a chance to acknowledge an order when it comes in the system.

2. They drive to the store of that order and are presented with a list of the items you would like to purchase.

3. As they find items, they pick them up, and scan the barcode to make sure it is the exact version of the product that you want.

4. They would check out and drive to your address to deliver your groceries.



Understanding the business will help us to better understand how the recommender system will add value to the business and by targeting its customers. The main advantage of the

recommender tool is to improve Instacart's marketing strategy towards its customers by personalizing their shopping experiences. This can be done by using different methods to recommend products that the customer is most likely to buy on their respective pages on the Instacart online shopping platform.

Deployment

In the deployment phase, our data mining analysis would be uploaded to R-shiny. The dataset is quite huge, so this serves as a good base for a recommender system, not limited to just Instacart.

Business Value Proposition

Instacart and other groceries delivery companies can leverage our results and derive the following benefits:

1) An effective recommender system with an area under the curve of *? And TPR = *?
2) Our recommender system recommends items that are important, novel with a touch of serendipity.

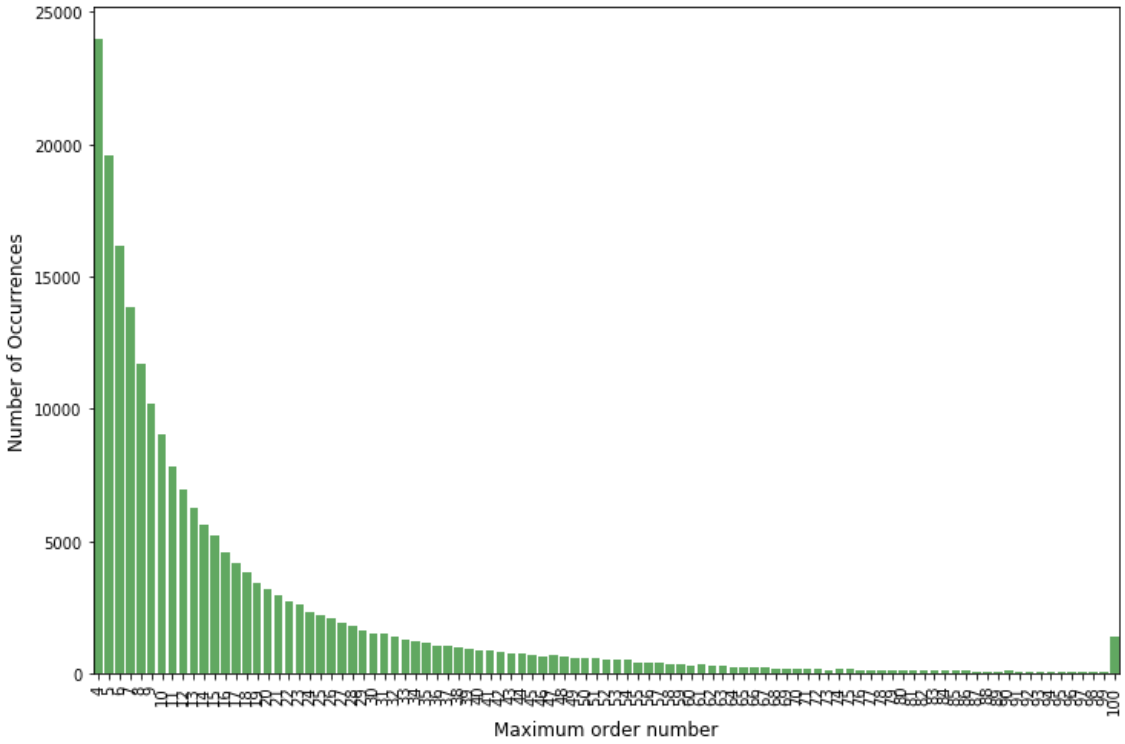
DATA UNDERSTANDING

We spent a significant amount of time understanding the Instacart data which was available on Kaggle. From this data, we had the following information:

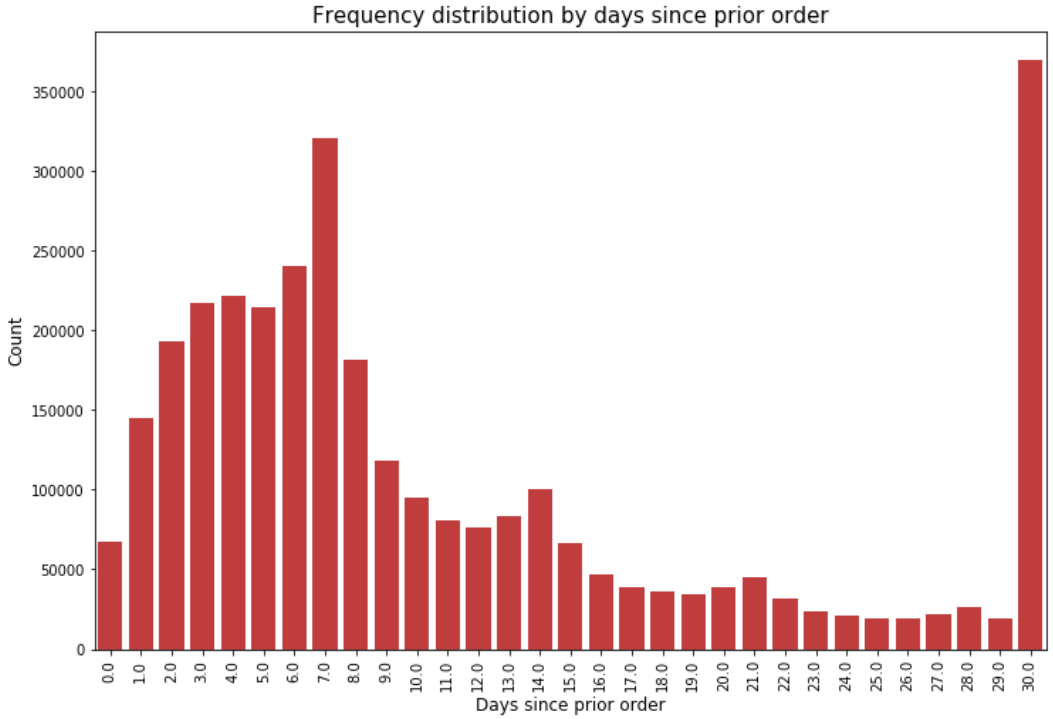
1. There are a total of 134 aisles in the store holding different products and each of them have a unique aisle id.
2. There are a total of 21 departments where all the products are categorized into (e.g. canned goods, personal care, and alcohol) and these departments also have unique department ids.
3. There are a total of 49688 products and every product has a unique product id
4. Our data also contained information on customers' user id which is a unique number assigned to each user at the point of registration.
5. The number of times each customer added a particular product to their cart
6. We also had data on all the products that each customer purchased and the frequency of purchase.
7. Whether or not a customer reordered a particular product
8. Each order placed by a customer was assigned a unique order number
9. The hour of the day that each customer's order was placed as well as the number of days that have passed since they placed their previous order.

In order to understand the data better and explore the relationships between the features, we created an Exploratory Data Analysis (EDA) using Python. From this analysis, we were able to derive the following information and relationships:

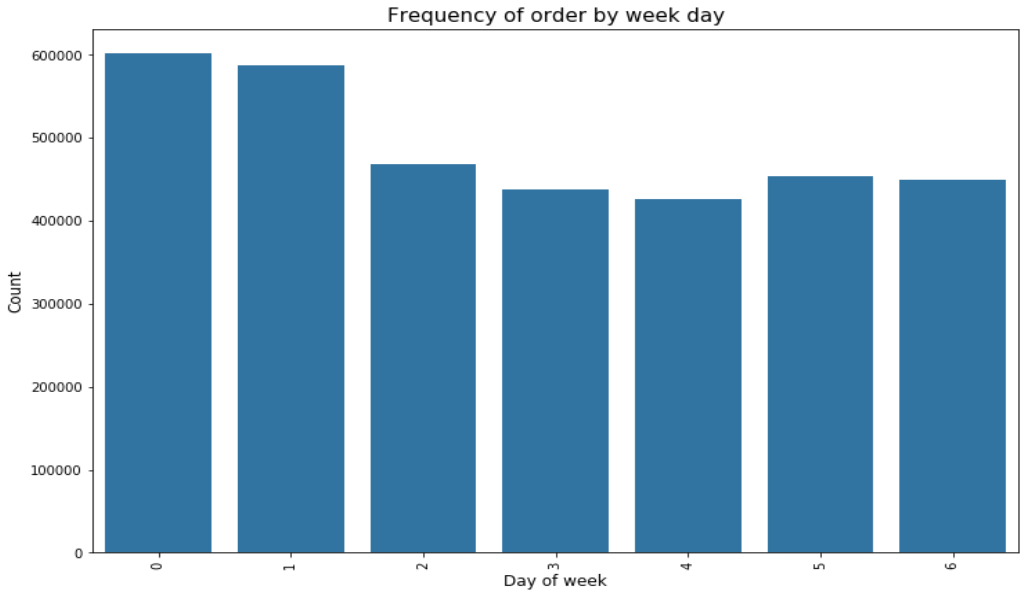
1. There were over 3 million orders placed by all customers
2. The minimum order number placed by the customers was 4 orders, occurring almost 24,000 times
3. The peak order times occurred between Saturday evening and Sunday mornings while Wednesdays saw the least orders
4. Customers order once every week (orders peak at 7 days) or once in a month (orders peak at 30 days)
5. The highest number of products bought in a single order was 80 and number of products bought in a single order with the highest frequency was 5 with a total of 8895 times.
6. Customer mostly placed orders for organic products, mostly fruits, with Banana having the highest number of orders at 472565 orders
7. The top two aisles that most of the ordered products came from were fresh fruits and fresh vegetables
8. Out of all 21 departments, the Produce department is the largest taking up 29.2% of the departments distribution
9. It appears that products that are added to the cart initially are more likely to be reordered compared to the ones added later



The distribution of Orders amongst customers. The lowest orders per customer is 4 with maximum 125.

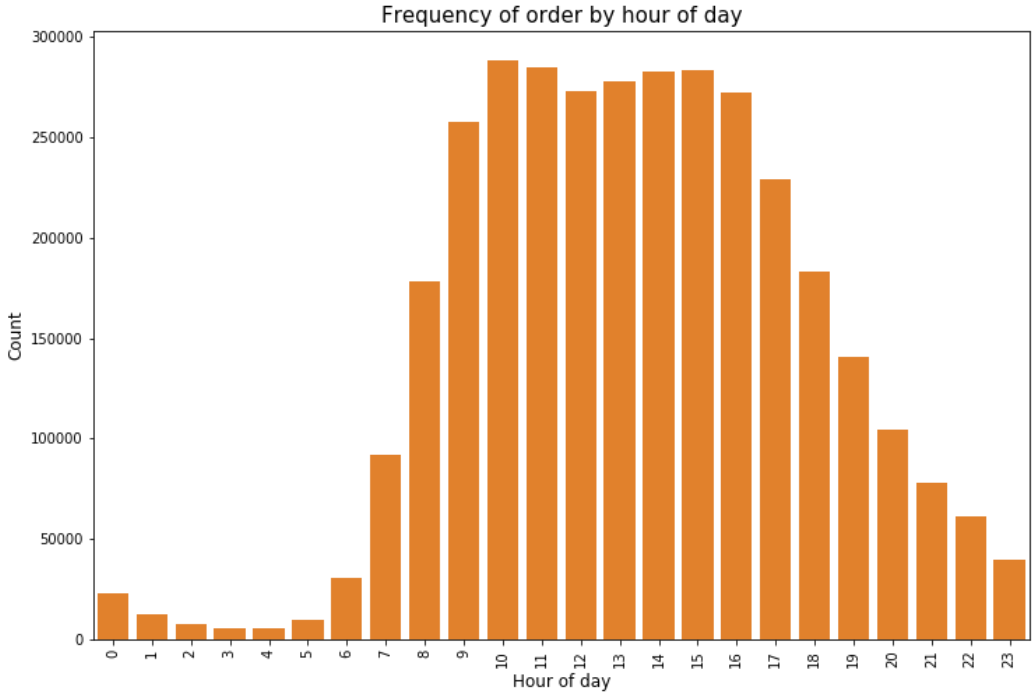


Frequency distribution by days since prior order, it is obvious that there is a significant increase in orders every 7 days and every month end.



Seems like 0 and 1 is Saturday and Sunday when the orders are high and low during Wednesday.

Order distribution with respect to time of the day.



DATA PREPARATION

Our journey to data preparation was founded upon the Business Understanding and Data Understanding phases of Instacart customer data. The data was provided in a structured format which saved the time and effort of cleaning. But, we needed to understand and organize the customer data in order to utilize it properly in our models.

Data segmentation was the key factor in identifying the vision for data preparation. By segmenting Instacart's customer and order information into meaningful populations of fields we were then able to understanding the business problem and gain a better understanding of the data. The first thing we did was to create data frames with the files. We the merged different data frames together in order to derive useful insights from the data.

To further understand Instacart's data we had to figure important information such which the number of times a customer has ordered a particular product as well as the frequency of order. We will discuss the basics of recommender systems below before going to the modelling part.

Recommender systems

Web is becoming increasingly important for electronic and business transactions. This is the driving force behind recommender systems.

The basic idea of a recommender systems is to utilize the various sources of data to infer customer interests.

Where the entity the recommendation is provided to is the user and the item is what is being recommended. The gist of recommendation is that there is a significant relationship that exists between users and item centric activities. These activities could range from shopping for specific products, watching a genre of movies more than other genes and so on. In many cases several items may show significant correlations which can be used to make tailored recommendations.

These relationships can be learned in a data- driven manner from the ratings matrix and the resulting model is used to make recommendations to other users. The larger the number of the recorded ratings between the user and the item, the easier it is to make future recommendations. The collective n-ratings of several users can be used to make groups of similar users that are

interested in similar products. The algorithm just described is based on a simple family of recommendation algorithms called neighborhood models.

This family belongs to a broader class called collaborative filtering.

Collaborative filtering entails using a collective ratings of several users in a collaborative way to predict missing ratings.

The operational and technical goals of recommender systems are:

1. Relevance: the goal of a recommender system is to recommend items that are relevant to the user at hand.

2. Novelty: recommender systems should add a mix of novelty to user recommendations. Recommending items users haven't seen in the past

3. Serendipity: recommender systems should have a mix of serendipity in that recommendation are truly surprising to the user not just something they didn't know about before (novel).

4. Increasing recommendation diversity: when all the recommended items the model suggests are very similar, it increases the risk that the user might not like any of these items. But when the model suggests different items there is a greater chance the user might like at least one of them

INSTACART DATA

As seen from the data understanding stage, Instacart's data is not only very large but it is data rich as well, this eliminates the task of dealing with missing values from the data set.

Since our task is to build a recommendation engine, we will need to do some feature engineering from the dataset. After merging the datasets and extracting useful features we were down to the remaining variables below:

	order_id	user_id	eval_set	order_number	product_id	add_to_cart_order	reordered	product_name
0	2539329	1	prior	1	196	1	0	Soda
1	2539329	1	prior	1	14084	2	0	Organic Unsweetened Vanilla Almond Milk
2	2539329	1	prior	1	12427	3	0	Original Beef Jerky
3	2539329	1	prior	1	26088	4	0	Aged White Cheddar Popcorn
4	2539329	1	prior	1	26405	5	0	XL Pick-A-Size Paper Towel Rolls

To model build an implicit recommendation system, we need to aggregate the number of interactions for each user and the type of products. In our implicit case, the way we will judge user – product interaction is by how many times a user reorders a particular product. The variable of interest is the reordered column. Aggregating the features, we came up with

	user_id	product_name	reordered
0	1	0% Greek Strained Yogurt	1
1	1	Aged White Cheddar Popcorn	2
2	1	Bag of Organic Bananas	2
3	1	Bartlett Pears	1
4	1	Cinnamon Toast Crunch	3

This gives us the count of the number of times a user reorders a particular item, which this we can build an implicit recommendation engine using our preferred library.

Sparsity of the Instacart Data

In order to build an implicit recommendation system, we have to ensure our sparsity is within range so as not to get a biased result. Sparsity is defined as the complement of the percentage of interactions divided by the number of users multiplied by the number of items. Recommender experts advice to keep the sparsity less than or equal to 99.5, for a good collaborative filtering recommender system.

Sparsity = 1 – density = 1 – (total interactions / user x items)

For the Instacart dataset, we have *206,209* users with *49,677* products. For these user/product interactions, *13,307,953* of these product had a purchase. This gives us a sparsity of 99.87%. Although this is quite higher than the sparsity threshold, but left room for more variable reduction to get a lower sparsity.

MODELLING

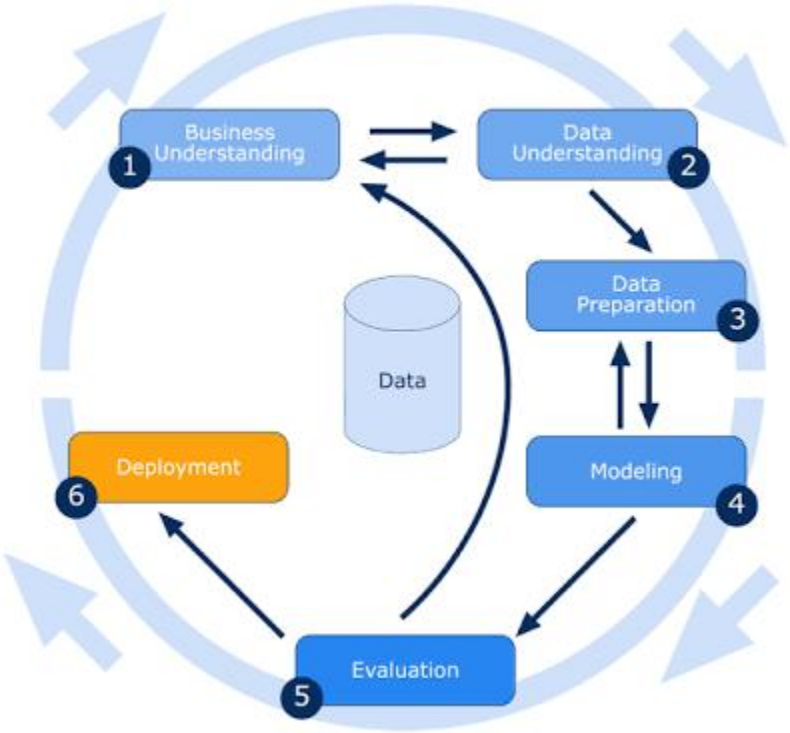
Approach

Cross Industry Standard Process for Data Mining (CRISP-DM) was used as the data mining process shown in the image below.

CRISP-DM is a cross-industry process for data mining. The CRISP-DM methodology provides a structured approach to planning a data mining project.

This is done to ensure, we follow and do not miss any of the required steps for creating our Machine Learning Model.

Phases of the CRISP-DM Reference Model



Source: (LinkedIn by Anshul Roy)

Modelling Tool: RecommenderLab

For the modeling, we used RecommenderLab which is a library in R Programming language used as a wrapper to help with data preparation, model selection, model training, and evaluation. This library was designed by Michael Hahsler. The recommender lab is robust enough to carry out data preparation, experiment design, model selection model training and evaluation. It offers a range of important recommendation algorithms to choose from, they include: Re-recommends Items, Item-Based Collaborative Filtering, User-Based Collaborative Filtering, Association Rules, Popular items, Random items, and the recently added Alternating Least Squares for both implicit binary and ratings data. The popular and random algorithms provides baseline for comparison of our models.

Recommenderlab also offers the opportunity to choose the method of finding similarity between Jaccard Similarity, Pearson Correlation and Cosine Similarity. However, only Jaccard Similarity was relevant for our binary "implicit" data. The Popular models offered a baseline for comparison of our models.

The recommender lab's evaluation function allows for several training and testing methods including split, bootstrap and cross validation. When setting up the evaluation function the data is divided into two groups: the known and unknowns, with group size determined by a "given" parameter. Then, the full dataset is divided into 5 training-testing folds. Once the model is trained on one of the training folds, predictions are generated using the model on items from the user's "known" group to be validated on their actual purchases from "unknown" group of the test set.

EVALUATION

Model evaluation was performed using the RecommenderLab library in R, we used all the algorithms offered by RecommenderLab on binary ratings, out of all the 7 algorithms used, User-Based Collaborative Filtering has the highest performance follow by Item-Based Collaborative Filtering, then Alternating Least Squares on their default parameters.

UBCF takes the "nn" most similar users (nearest neighbors) and computes their distances from the active user. In our case, only Jaccard Similarity because our data is implicit (binary), but Pearson and Cosine can be used to measure distance with ratings data. The items purchased corresponding to these "similar" users are then aggregated into a set of items purchased by the group, along with the frequency of their purchase. UBCF then uses this set of items to generate top N lists of item recommendations for the related user. Known as being "memory-based," the challenges for UBCF are scalability and time because the entire data set had to be stored in memory and used to generate top N predictions when given a particular user (Su & Khoshgoftaar, 2009).

Evaluation Metrics

The evaluation metrics used in the modelling process is the one suitable for Top N systems in binary ratings collaborative filtering which are Receiver Operating Characteristic (ROC), precision, recall and Area Under Curve (AUC)

True Positive Rate (TPR): This is the percentage of purchased items that have been recommended. It's the number of TP divided by the number of purchased items (TP + FN)

False Positive Rate (FPR): This is the percentage of not purchased items that have been recommended. It's the number of FP divided by the number of not purchased items (FP + TN).

Precision: This is the percentage of recommended items that have been purchased. It's the number of FP divided by the total number of positives (TP + FP).

Recall: This is the percentage of purchased items that have been recommended. It's the number of TP divided by the total number of purchases (TP + FN). It's also equal to the True Positive Rate.

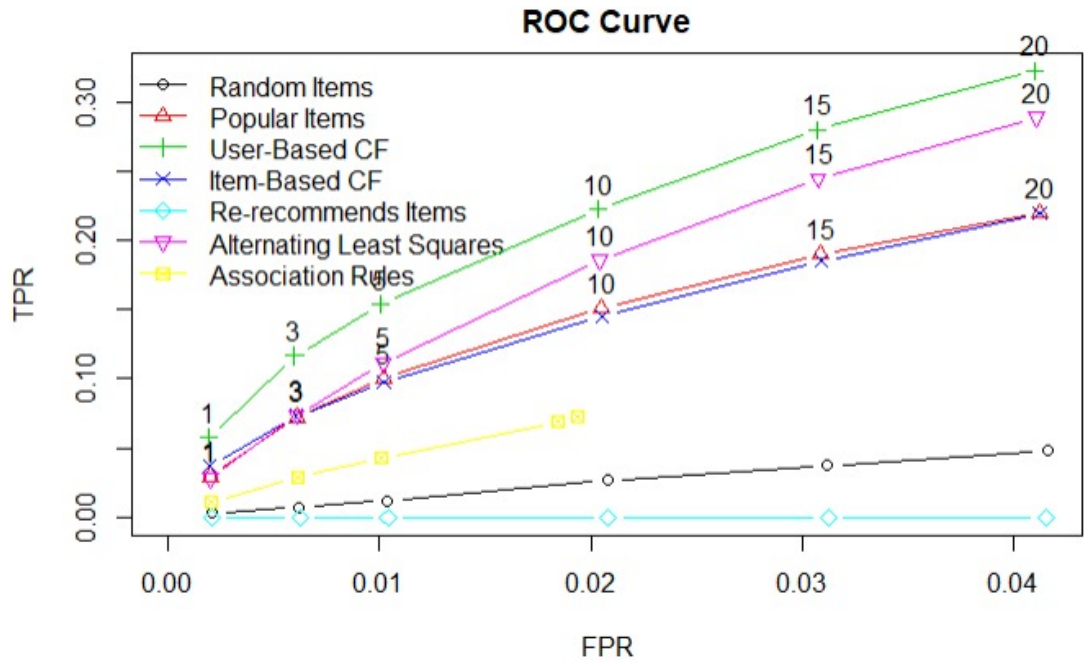
Cross validation: This is a very popular technique for model evaluation for almost all models. In this technique, we divide the data into two datasets: a training dataset and a test dataset. The model is built using the training dataset and evaluated using the test dataset. This process is repeated many times. The test errors are calculated for every iteration. The averaged test error is calculated to generalize the model accuracy at the end of all the iterations.

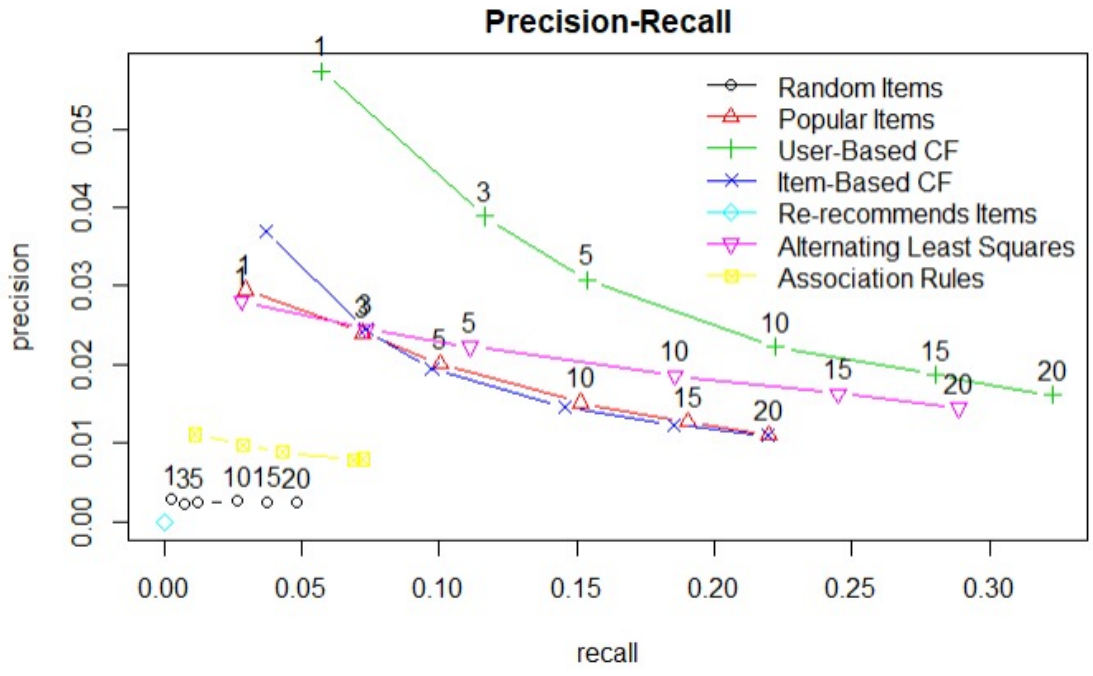
Evaluation Process

We used the Recommenderlab evaluation scheme and set k = 5 which divides our data sets into 5 separate folds. As is normally the case with cross-validation, each fold alternatively acts as the test set for one iteration of model training and testing, and each user and all their purchases are included in at least one test set fold (so all data points are used). The part which may be a little different from the usual training/testing process, is that the test data is divided into two separate groups of “known” and “unknown” transactions.

The transactions withheld for the "known" set are selected randomly to be given to the trained model to generate sets of top N predictions. The remaining transactions in the "unknown" set are used to represent positive instances for comparison against the predictions to determine the hit rate. The “given” parameter allows us to set the relative size of the "known" and "unknown" groups. The user is represented by only those items in the “known” part of the test set when shown to the trained model to make predictions.

We discovered that AUC improved significantly using the “All But One” (given = -1) setting for the parameter rather than using a random number.

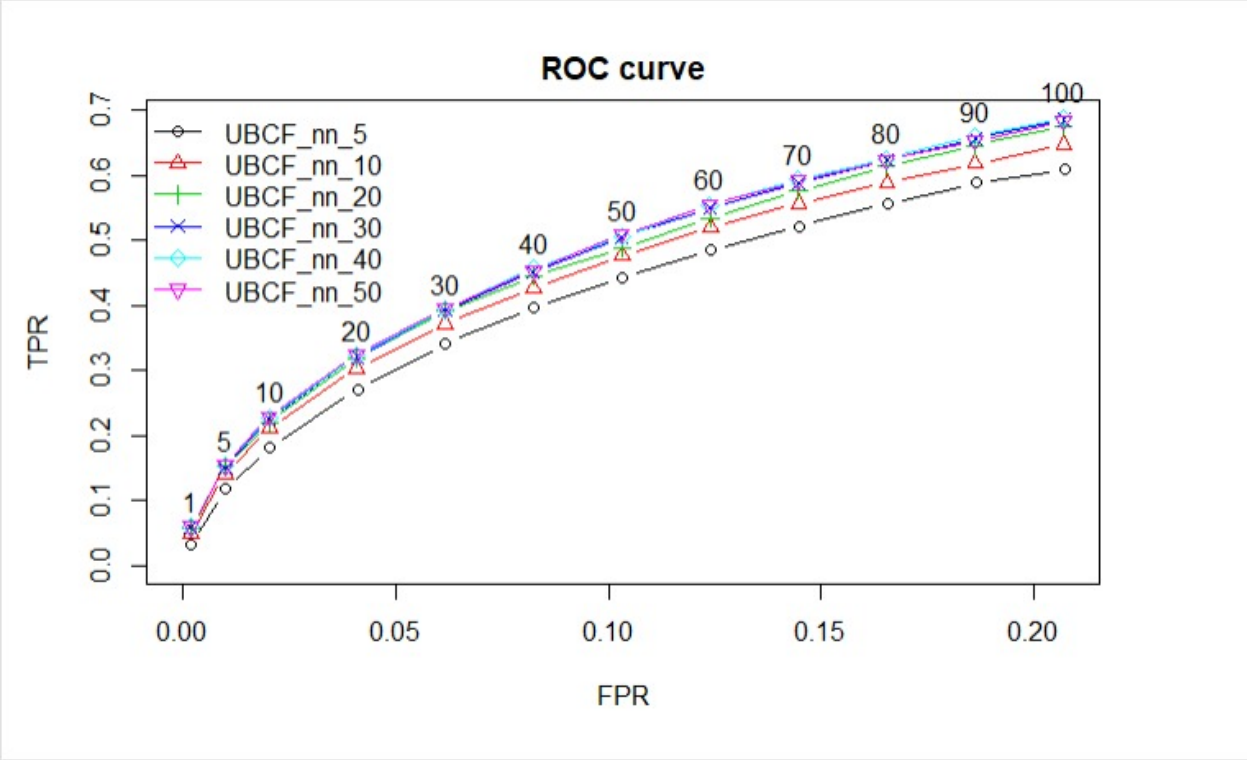

ROC Curve
Random Items
Popular Items
User-Based CF
Item-Based CF
Re-recommends Items
Alternating Least Squares
Association Rules
TPR
FPR
0.00
0.10
0.20
0.30
0.00
0.01
0.02
0.03
0.04

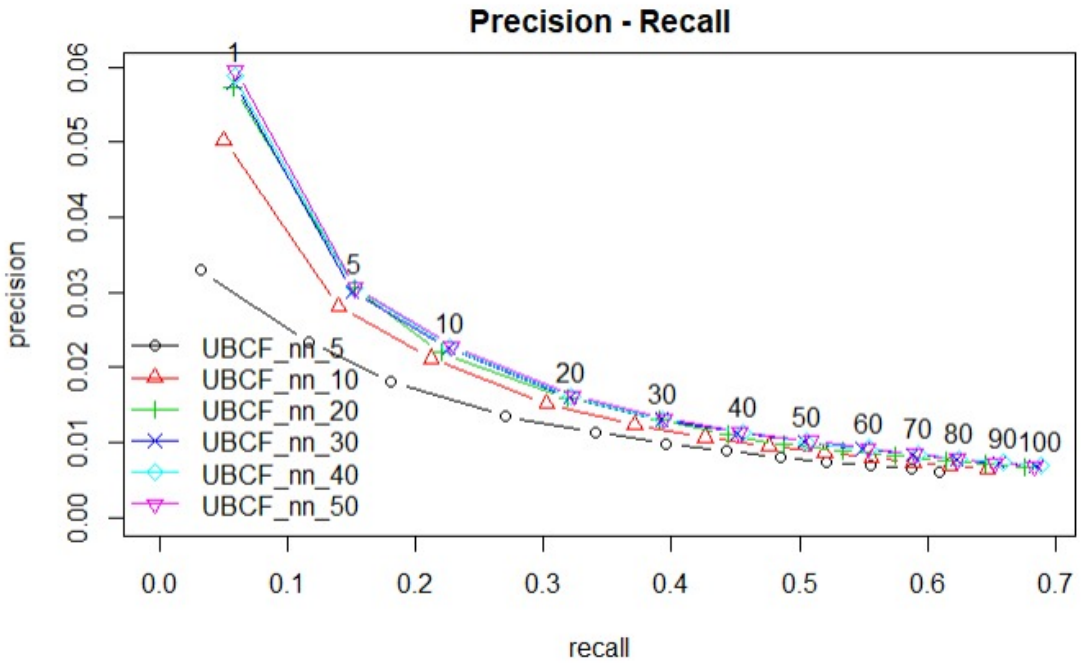

Precision-Recall
Random Items
Popular Items
User-Based CF
Item-Based CF
Re-recommends Items
Alternating Least Squares
Association Rules
precision
recall
0.00
0.01
0.02
0.03
0.04
0.05
0.00
0.05
0.10
0.15
0.20
0.25
0.30

Model Tuning

In this section, we will discussed how we tuned UBCF since is the algorithm that perform best when we use the default parameters.

We ran six experiments with nn at 5, 10, 20, 30, 40 and 50, and Build a chart with the ROC curve and Precision-Recall to identify the best-performing nn and we found the performance of nn = 30 to be better. See the two images below




Precision - Recall
precision
recall
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
1
5
10
20
30
40
50
60
70
80
90
100
UBCF_nn_5
UBCF_nn_10
UBCF_nn_20
UBCF_nn_30
UBCF_nn_40
UBCF_nn_50

DEPLOYMENT

After several iterations using the CRISP-DM process, we are convinced that the implementation of the recommender system meets the business objective. We will discuss the steps taken to deploy our solution in the R Shiny App.

Shiny is a powerful tool in R for you to show off your work to the world, without explaining all the complicated code behind your analysis. We choose Shiny App because its free and open-source development and deployment structure, which allows us to share our work online easily.

When every component of our application is finished, we then published our shiny app online by creating an account on https://shinyapps.io, which is needed for publishing our application, and connect it to RStudio after everything is done, we click the Publish button and all work was uploaded to the shiny server and is accessible from our account, but also from a specific URL. The URL for our app is https://instacart.shinyapps.io/instacart_idaf/

Recommender System For Instacart Online Grocery Store

A Capstone Project Presented to Information and Data Analytics Foundation

Purchase item(s) from the store, by selecting from the dropdown list

Item 1
100% Whole Wheat Bread

Item 2
Aluminum Foil

Item 3
Bag of Organic Bananas

Item 4

Item 5

Item 6

Item 7

Item 8

Item 9

Item 10

Complete Purchase

Because you purchased those item(s), we recommend this:

Recommended Items
100% Pure Pumpkin
Asparation/Broccolini/Baby Broccoli
Banana
Bartlett Pears
Blackberries
Blueberries
Boneless Skinless Chicken Breasts
Cherubs Heavenly Salad Tomatoes

Designed by Okuwoga Toluwalase, Okechukwu Princewill, Olanayo Olatunde. For the full code, please visit the team GitHub Page

The Instacart Online Grocery Shopping Dataset 2017, Accessed from Here

REFERENCES

1. Su, X., & Khoshgoftaar, T. (2009, August 3). A Survey of Collaborative Filtering Techniques.
2. Retrieved from Advances in Artificial Intelligence: https://www.hindawi.com/journals/aai/2009/421425/