International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

ONLINE SHOPPING REMINDER USING NEURAL NETWORK

Selvakumar Kalyanasundaram

Texas, USA
inboxofselva@gmail.com

Abstract:
In the current fast world, grocery purchase in bricks and mortar are slowly declining. People prefer buying online rather than going to shop. All the purchases are done through the grocery App in Mobile. Likewise, the creation of Shopping list was moved from paper to mobile application due to the emergence of new technologies. As an additional improvement, this paper, provides a way to automate the entry of products list in the grocery list App. And reminds the customer exactly on time when he may need to buy. Thereby, it would be reminder tool for the customer. Also, it would be a tool for the merchant to keep the customer stay connected. Thereby keep the sales consistent. This paper shows how it could be achieved using the purchase history data in Feed forward neural network with latest optimization techniques. Hence, this analysis stands apart accuracy and performance wise from the earlier State of Art.

Keywords: Grocery list, Shopping list, Feed forward Neural Network, sklearn, TensorFlow and Keras.

INTRODUCTION

Most of us use grocery list or shopping list while purchasing our regular items. Shopping list is an integral part of the consumers while shopping. The purchasers, whoever buy the products might write down the list of items to be bought before purchase. Few, might also write down the list of items they missed while shopping, which they use on their next purchase. Recent analysis on the shopping shows that nearly 40% of the shoppers are using grocery list while shopping, out of the total item purchased nearly 80% of the products are from their grocery list. 20% of the remaining items were marked to be bought on the next visit. This clearly shows that grocery list is one of the handy tools for efficient shopping. And, the use of the grocery lists clearly impact the shopping behavior. Also, it helps to reduce the expenditure while shopping. [Rfr1]. There are several mobile App products for shopping list. These shopping list applications which are available in the market today are customer centric and it helps the user to note down the list of items to be purchased. So, there is no automatic tool which reminds with the list of items to be purchased without getting prior input from the user.

LITERATURE REVIEW

All the IEEE research papers related to shopping list were reviewed in this thesis to verify whether there is any existing paper related to our problem question. A research named, **Mobile shopping Assistant** in 2007, being developed by integrating mobile application with webservices using XML compression features along with speech and image recognition APIs. And it proposed an idea to create shopping list using mobile. But it is mostly oriented towards the enhancement of the input technologies for creating the shopping list. [Rfr5]
Several researches were done with emerging technologies and one of them in the beginning of 2009, under the topic **Multimodal shopping list**, researchers recommended an interactive system to create shopping list from several input devices like desktop, landline phone and mobile phone using different input formats like text, images and audio files. In addition, they proposed an Interactive system to create and manage the shopping lists. [Rfr4]

In the beginning of 2011, Felix.H and Daniel. S proposed a method in **Digital Grocery list** to reduce the burden of typing the item names in mobile application with the digital shopping list application. Handwritten grocery list or even the paper leaflet from the super market loaded can be recognized by their proposed mobile application. Here the Anoto dot pattern technology has been used to digitize the handwritten input. Using this digital input, the corresponding item names are looked up from the vocabulary database. [Rfr2]

A paper on **Intelligent shopping list** in the year 2011, proposed a novel system to extract the item information automatically from the handwritten information on a special Anoto paper. Using a digital pen, the customer writes the item details that need to be purchased on the Anoto paper. The digital pen strokes were recorded while writing on the special paper. The recorded information from the digital pen is fed into the computer to detect the item and amount by mapping the input signal with the shopping center ontology. Then the final result is fed to shopping database to do shopping. Thereby helps to minimize the effort of customer to carry either paper or mobile containing shopping list. The results are more convincing. [Rfr3]

One of the latest researches in the year 2017 went further and explained about the implementation of data mining techniques in the creation of shopping list. The **Smart Shopping list**, provided an effective mobile solution for shopping list creation. It proposed a new feature called "Recommended Items" in which they proposed to use Apriori Algorithm to recommend the possible missing items when a purchaser creates the shopping list. [Rfr6]

The above research paper made an inspiration on the usage of data in the shopping list, as it helped to increase the sales by recommending new items by applying Association rule mining algorithms. Association Rule mining is one of the data mining techniques which is used for identifying the relation between two different items. The possible percentage of the relation of in a finite set of items gives the new insight so therefore it is very useful in making decisions to enhance the sales of the products or to increase the profit.
All the above research papers are related to technology used in Shopping list and the method in which the data is fed in to the shopping list tool. We also reviewed few existing algorithms which might be suitable for our problem question.

Normally, most of the pattern-based approach starts with the Apriori algorithm [Rfr7] for extracting the patterns. Apriori algorithm is used for identifying what product bought together in most of the transactions. Based on the purchase patterns of the most of the customers, co-occurrence of the product can be determined. Apriori algorithm traverse through the purchase history data iteratively to identify the frequent items sets.
Three matrices are calculated in this algorithm: Support, confidence and lift. Support is the frequency of the combination of the item purchased. With this, less frequent items can be excluded. Confidence, tells how repeatedly the frequent combination occur. New rules can be derived from support and confidence. The best rules for identifying the patterns can be derived from Lift.

Although, the Apriori algorithm helps to find the frequent pattern, there are some disadvantages. The process becomes slow when the dataset size increases. The process needs to be executed every now and then to keep the model up to date. One more **disadvantage of Apriori Algorithm** is that the algorithm does not give importance to recent purchase of the customer that might be bought due to personal interest or climatic change etc.

One of the researches compared the Apriori algorithm and market basket analysis by making series of test over the same set of data. As an outcome of their test, they identified that both the methods provided same set of result. [Rfr8]

Another most popular algorithms for frequent item set mining is the FP-growth algorithm without Candidate Generation, since it condenses the transaction information into a tree structure. Due to the recursive projecting approach of FP-growth, the ExAnte data-reduction is pervasive all over the computation. All the FP-trees built recursively during the FP growth computation can be pruned extensively by using the ExAnte property. The main **drawback of FP growth**, which is its memory requirements on one hand, and on the other hand, the main drawback of ExAMiner which is the I/O cost of iteratively rewriting the reduced datasets to disk. [Rfr9]

On reviewing **Classification and Regression Tree (CART)** prediction method, Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. [Rfr10]

Based on the analysis of existing State of Art, the technology used for creating the shopping list tool was already derived. And the recommendation system in the shopping list was already incorporated. With the above analysis, *the problem statement was refined to derive the* ***user visit frequency*** *and* ***product frequency from the purchase history data using Neural Networks****.*

PROBLEM STATEMENT

Based on the motivation and existing literatures, the problem statement is refined as:

Remembering the exact list of items to purchase at the exact time, when needed, would be difficult. So, it would be good, if there is any reminder tool in mobile that reminds the user with the list of products need to be purchased at the exact time of requirement.

In real time, learning the purchase pattern of different customers across different product categories using the traditional algorithm like Decision Tree, Apriori would provide satisfying results. So, our approach aims to use Neural Networks the learn the complex relation between the user and products.

Unlike the existing shopping list application in the market, the proposed one is merchandise centric. It uses the purchase history as the data to identify the frequently purchased products. And reminds the customer with purchase list, on a couple of days before his predicted purchase date. And also reminds the products that he missed to buy on the previous purchase.

OBJECTIVE

Using Neural Networks with revised performance tuning methodologies, this paper aims to answers the **Core problem questions:**

- **When the customer would purchase next time?**
- **What products he would buy in his next visit / next login?**

The challenge in purchase patterns of the customers is the consistency. Few reasons for this variation are like new products in market, change in customer financial status, change of residence, manual miss to buy a product, dislikes on past purchased product etc., If the customers are reminded with the list of products, the purchase patterns would be kept intact to some extent. Moreover, the customers can be made as a frequent buyer under the same retailer.

The objective of this paper is to learn this inconsistency using neural networks and derive automatically the list of items to be purchased for each customer from the second time onwards based on his first set of manual entry. This approach is novel as it avoids the manual feed for the shopping list creation to a certain extent. And uses neural networks with **latest techniques of hyperparameter optimization and regularization** to achieve this for better accuracy and performance.

METHODOLOGY

Data source - In this section, we report the experiment performed on purchase history data extracted from Kaggle website. [Rfr11]. Totally, there are five files which includes order_products_prior.csv, aisles.csv,

department.csv, order_product_train.csv, orders.csv and products.csv. Altogether, there are 13 features. These datasets were published by Instacart for one of the Competitions in Kaggle.

Data Description - The dataset contains a sample of over 2 million orders of groceries from more than 230,000 users. Each user, provides orders from 5 to 150. List of features provided in the file are given below.
orders file:

- order_id: Identification number of the order
- user_id: Identification number of the customer
- eval_set: Assessment of the order comes under
- order_number: The number which shows the order of products added in the cart
- order_dow: Order day of the week
- order_hour_of_day: Order hour of the day
- order_timestamp: Order timestamp

products file:

- product_id: Unique identification number for the product
- product_name: Name of the product

order_products_prior file

- order_id: foreign key (see order file)
- product_id: foreign key (see products file)
- add_to_cart_order: order in which each product was added to cart users
- user_id
- zip_code
- family_member_cnt

Data Handling Methodology - The plan is to train the models with single category of data. To achieve this, we have planned to take the data of the topmost aisle in which the products are purchased frequently. Then the model would be evaluated with the data from the other departments.
To make sure whether the product is market ready, the model would be fed with entire dataset which contains the data from all the departments. And the accuracy and scalability of the model would then be cross checked with the existing State of Art.

Data preprocessing and Analysis - EDA and cleansing were done on Intel ® Core™ i7-6600 U CPU @ 2.6GHz (24 GB RAM Memory) using Anaconda Jupyter Notebook 6.0.3. The entire analysis and model creation are being done using Anaconda Python 3.7 and scikit 0.11 on Windows 10.
After merging all the five files together pandas in Python 3, the size of the data was around 680MB. Combined all the features from products, orders, users and order_products_prior file into a flat file. There are total of 32434489 records with 3214874 orders from 206209 users for 49677 distinct products. From the order_timestamp, the days after which the product reordered was calculated and added as a new feature. The frequency of reorder for the products varied from 0 to 31 days. By default, in a single cart, the customer can purchase the products from different categories. All these products are categorized under 20 departments. As the customer could purchase from all departments, all the record is fed into the input layer.

All the department data were considered. There are more probabilities for reordering the same product by the customer next time. So existing purchase history of the customer helps. But the new products that are expected by the customer can be predicted from the basket of other customers. For this thesis, all the data are real and no synthesis were done. Most of the features are categorical variables.

The total number of items in a basket in each order and the quantity of each item were missing. Those was derived as a new quantitative feature. Python Label Encoder function was used to deal with the categorical variables.

The purchase order data is in the csv file with order id and customer id along with the days after the prior order. The cleansing rule has been applied on the data and merged with the product file which has the item features. The unwanted features are removed and the records with negligible transactions are removed to get the concrete prediction. Few insights on exploring the raw data are given below.

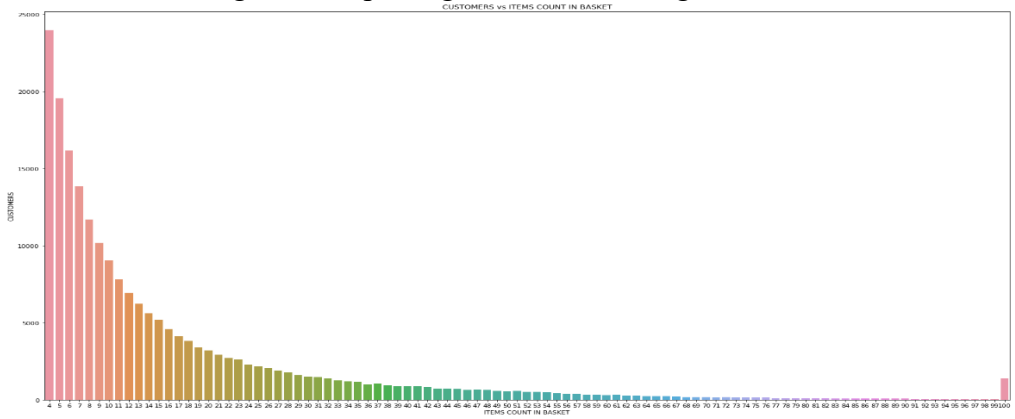

Fig 1. *customer vs item count*

Minimum basket size of each customer is around 4 items. Overall, 8 items per customer on an average.

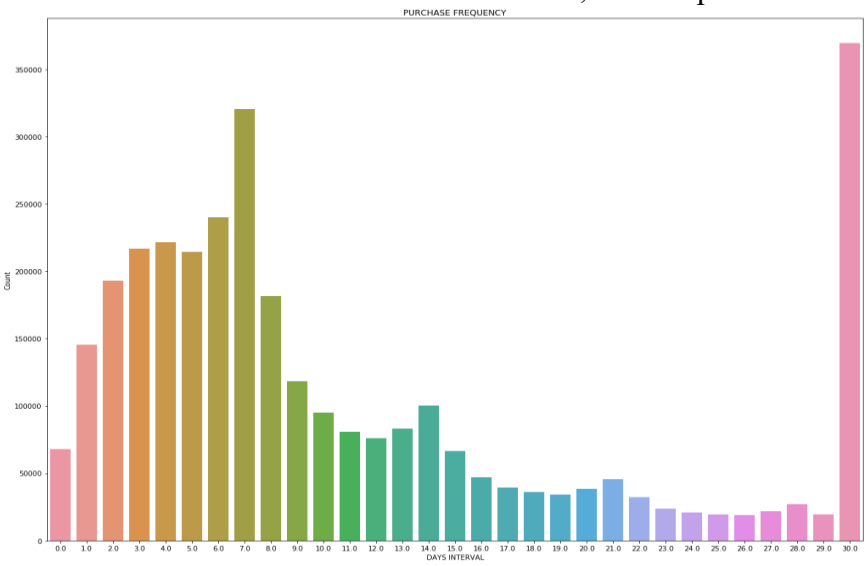

Fig 2. *Days interval purchase frequency*

Purchase frequency is more at the interval of 7 days (Weekly) and 30 days (Monthly)

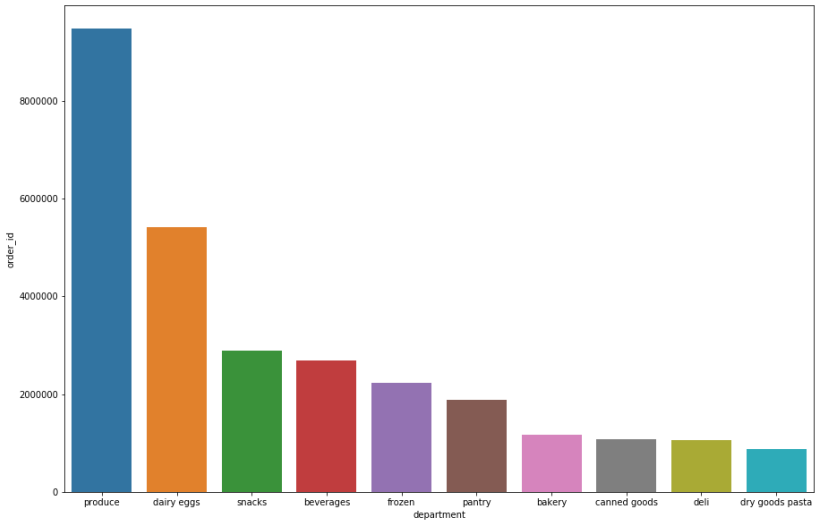

Fig 3. *Department wise purchase frequency*

Frequent products fall under the category of produce and dairy eggs departments.

To build the model, network topology of Multilayered network was used using the input features. Out of the 14 features, only product_id and number of items in the basket are considered. Rest of them are categorical fields. Only top 15 products from fresh vegetables aisle were considered to derive the model. The top 15 products are

Top 15 products
Organic Yellow Onion
Organic Garlic
Organic Zucchini
Cucumber Kirby
Organic Cucumber
Michigan Organic Kale
Carrots
Fresh Cauliflower
Yellow Onions
Organic Small Bunch Celery
Organic Tomato Cluster
Asparagus
Organic Red Onion
Organic Red Bell Pepper
Red Peppers

Table 1*. Top products sold*

Data preparation - Before entering into model generation, the first step is to make the purchase history data in a consumable format for these models. The transaction data contains user details (U_i), Product attributes (like product details and the quantity purchased), It is also having the time of purchase.

$\mathbf{U_i \mid [\, U_iA_1, U_iA_2, U_iA_3] \mid P_i \mid [P_iA_1, P_iA_2, P_iA_3] \mid P_j \mid [P_jA_1, P_jA_2, P_jA_3] \mid T_i}$
$\mathbf{U_j \mid [\, U_jA_1, U_jA_2, U_jA_3] \mid P_i \mid [P_iA_1, P_iA_2, P_iA_3] \mid P_j \mid [P_jA_1, P_jA_2, P_jA_3] \mid T_j}$
$\mathbf{U_k \mid [\, U_kA_1, U_kA_2, U_kA_3] \mid P_i \mid [P_iA_1, P_iA_2, P_iA_3] \mid P_j \mid [P_jA_1, P_jA_2, P_jA_3] \mid T_k}$

Table 2. *Purchase History Format*

Three set of datasets created from the above data. Thereby creating a total of 4 files, one for each model learning with User data, Product data and user product combinations data. This can be achieved by using the below algorithm

```
While Ri. Loc < Len [Dataset]:
    If (Ui ==Uj):
        △T = Ti - Tj
        For I in P:
            PPr[I] = P[I]
            I=I+1
         Write ([ UiA1, UiA2, UiA3] | Pi[list]| △T| PPr [List])
Ri. Loc=Ri.Loc+1
```

Table 3. *Data derivation algorithm*

The datasets generated using the above algorithm are like below.

Usr Frq File

[U_iA_1, U_iA_2, U_iA_3] | T_i

[U_jA_1, U_jA_2, U_jA_3] | T_j

[U_kA_1, U_kA_2, U_kA_3] | T_k

Product Frq File

P_i | [P_iA_1, P_iA_2, P_iA_3] | P_j | [P_jA_1, P_jA_2, P_jA_3] | T_i

P_i | [P_iA_1, P_iA_2, P_iA_3] | P_j | [P_jA_1, P_jA_2, P_jA_3] | T_j

P_i | [P_iA_1, P_iA_2, P_iA_3] | P_j | [P_jA_1, P_jA_2, P_jA_3] | T_k

Product_Classification File

P_i | [P_iA_1, P_iA_2, P_iA_3] | P_j | [P_jA_1, P_jA_3] | $\triangle T_i$ | P_i | [P_iA_1, P_iA_2, P_iA_3] . .

P_1 | [P_1A_1, P_1A_2] | P_j | [P_jA_1, P_jA_2, P_jA_3] | $\triangle T_j$ | P_1 | [P_1A_1, P_1A_2, P_1A_3] . .

P_i | [P_iA_1, P_iA_3] | P_j | [P_jA_1, P_jA_2, P_jA_3] | $\triangle T_k$ | P_i | [P_iA_1, P_iA_2, P_iA_3] . .

Usr_Product_Classification File

[U_iA_1, U_iA_2, U_iA_3] | P_i | [P_iA_1, P_iA_2, P_iA_3] | P_j | [P_jA_1, P_jA_3] | ΔT_i | P_i | [P_iA_1, P_iA_2, P_iA_3] ..

[U_jA_1, U_jA_2, U_jA_3] | P_i | [P_iA_1, P_iA_2] | P_j | [P_jA_1, P_jA_2, P_jA_3] | ΔT_j | P_i | [P_iA_1, P_iA_2, P_iA_3] ..

[U_kA_1, U_kA_2, U_kA_3] | P_i | [P_iA_1, P_iA_3] | P_j | [P_jA_1, P_jA_2, P_jA_3] | ΔT_k | P_i | [P_iA_1, P_iA_2, P_iA_3] ..

Table 4. *File format for machine learning datasets*

Design Overview - The proposed design contains four separate neural network models. They are

- **usr_frq**: NN Model predicts the shopping frequency based on customer data
- **usr_prd_frq**: NN Model predicts the shopping frequency based on customer + corresponding attributes of the product purchased
- **usr_to_prd_clsfy**: NN Model predicts the possible combination of products would be purchased based on customer data
- **usr_prd_to_prd_clsfy**: NN Model predicts the possible combination of products would be purchased based on customer + corresponding attributes of the product purchased

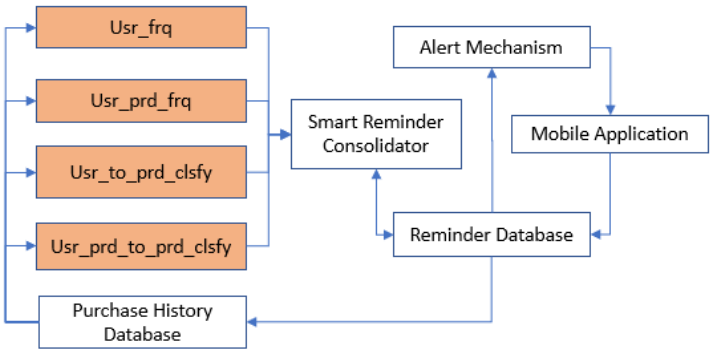

Fig 4. *Proposed Design*

The usr_frq and usr_prd_frq models were generated using sequential feed forward network with linear regression to derive the customer frequency based on customer and product attributes. The usr_to_prd_clsfy and usr_prd_to_prd_clsfy were generated using sequential feed forward network with multiclass classification to derive the possible product combinations from customer and product attributes.
The input data will be fed into the respective models to predict the pattern of the customer and products along with the frequency for each combination of the products. The smart reminder consolidator processes the output from the model and consolidates the predicted products with the minimum frequency generated out of 2 models and feeds the input to 3rd model to get the next probable products and send the data to a database with the data on which the reminder has to be initiated. Alert mechanism application service initiates the reminder with the predicted data from the reminder database as per the timestamp mentioned in the database. The response made by the customer on the reminder would be with purchase or ignore. Those details are recorded and fed into the model for continuous learning.

Model Algorithm - The ANNs have the ability to learn non-linear and complex relationships. And it has been used for several classification and prediction applications, including forecasting, image and character recognition, traffic prediction, pheno-type prediction in genetic data. Deep learning models refer to ANNs with more than one or two hidden layers. It is a representation technique, which gives a learning machine the ability to process raw data and identify its representation for further processing and decision making. Such machines can provide different levels of data representation through their multilayer architecture. More the number of layers provide a more abstract representation of data and eliminates irrelevant variations.
Earlier researches on market basket used the deep learning methodologies to provide solution like product recommendations on various timeframe of the year and shopping location etc.,
There are different kinds of algorithm in neural networks. The most popular one is back propagation. The back-propagation algorithm performs learning on multilayer feed-forward neural network. It iteratively learns the weight and bias from each activity functions. A multilayer feed forward neural network consists of an input layer, one or more hidden layer and an output layer. An example of a multilayer feed-forward network is shown in Fig. 5

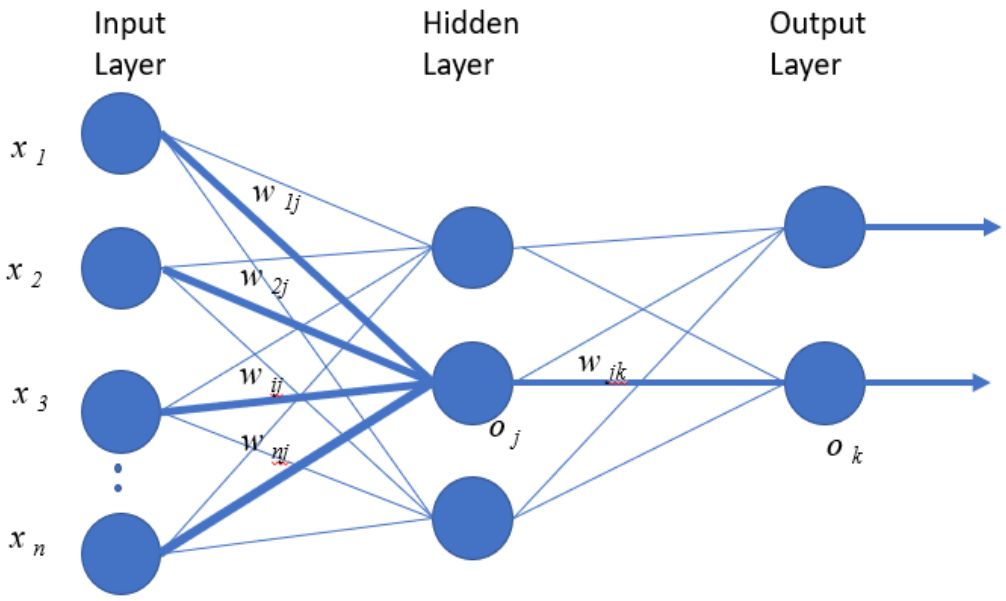

Fig 5. *Multilayer feed-forward neural network*

Back propagation Algorithm. Where the Neural network can learn from the existing data and classify or predict the output for the new incoming data efficiently

Input
D, Tuple containing the input and the corresponding output values.
l, the Learning rate
network, a multilayered feed-forward network

Output
A trained neural network.

```
Method
Initialize all weights and biases in network;
While terminating the condition is not satisfied {
    For each training tuple X in D {
        //propagate the inputs forward:
        For each input layer unit j {
            Oj=Ij; //output of an input is its actual value

        For each hidden or output layer unit j {
            Ij=∑wij Oi + θj; //compute the net input of unit j with respect to the previous layer, I
            Oj= 1 / (1+exp (-Ij));} //compute the output of each unit j
        //Backpropagate the errors:

        For each unit j in the output layer
            Errj=Oj (1 – Oj) (Tj – Oj); //compute the error

        For each unit j in the hidden layers, from the last to first hidden layer
            Errj = Oj (1 – Oj) ∑ Err k wjk; compute the error with respect to next higher layer, k

        For each weight wij in network {
            △wij =(l) Errj Oi ;//weight increment
            Wij = wij + △wij;} //weight update

        For each bias θj in network {
            △θj = (l) Errj; //bias increment
            θj = θj +△ θj ; } //bias update
        }}
```

Table 5. *Back-propagation Algorithm*

The efficiency of this algorithm depends on the learning time spent while calculating the weight and biases. It can be calculated in the form of epoch i.e. O (|T| * w) where |T| tuples and *w* weights.
The given data is preprocessed using the standard scalar method from scikit. To build the neural network, keras was used. Keras is the latest package of python for developing neural network models. It is an open source package. Instead of coding directly in Tensor flow, the Keras can be used as a wrapper to develop NN over Tensor flow. From Keras, sequential model was used to build this NN. Using these three files, four different models are created. The models are optimized by adjusting the hidden layers and number of neurons to provide accurate predictions.

Dense layer implements O_p= Act (D (I_p, K) + bias). Where Op is output and Ip is input D is the Dot product and K is the Kernel. [4C]. For the neural network to initiate learning, the weight and bias has to be initialized. kernel in keras generates tensors with a normal distribution in order to set the weights in each neuron of the dense layer. we can use Random uniform initializers from keras, in order to get the uniform distribution, Using Keras, the weight and bias will get assigned automatically in each layer.
For the product classification, in order to calculate the loss function, we used binary_crossentropy. The loss is the difference between the target value and the predicted value. And for the time frequency prediction, we used mean square error as the loss function.

Adam is one of the optimization tools in NN. We considered the Adam optimizer which is a good default implementation of gradient descent. Gradient descent is a widely used optimization algorithm used to find the minimum of a function. The basic idea is to use the gradient of the function to find the direction of steepest descent, i.e. the direction in which the value of the function decreases most rapidly, and move towards the minima iteratively. Adam stands for Adaptive moment estimation. Adam comprises of RMSProp and

Momentum. Momentum takes the adjacent t-1 gradients into account in order to smooth out the t gradient descent. we use accuracy as the metrics to measure the performance of the model.

Model Training - The given dataset has 230,000 products categorized 21 departments in a total of 134 aisles. Deriving 4 models from this dataset from the system with limited processing power and memory would not be possible. So, we applied the concept of Non-Negative Matrix Factorization. As per which the large matrix of V can be factorized into two small matrices W & H which approximately reconstruct the matrix V.

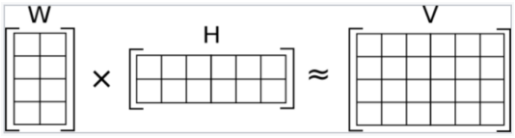

Fig 6. *Non-Negative Matrix Factorization*

In our case, to derive the classification of products, we used Users x 15 and 15 x Products as per NNMF. To fix the train-size for the model generation, the learning curve method was applied. Model generation was iterated with single layer and early stopping mechanism. The scores for different set of data size are given below. The curve seems to be flattened out with data around 780-800. So, instead of data augmentation, the model optimization was preferred.

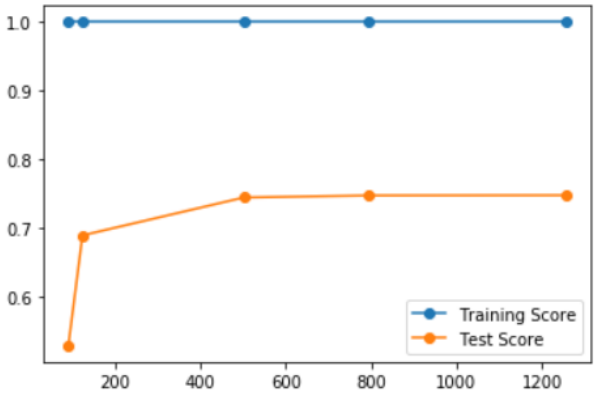

Fig 7. *Learning curves*

Usr_frq Model: To create this model, user data along with frequency of visit was used. Before feeding the data into this model, using keras utility, the user data is converted to categorical values. And fed into the dense feedforward network. The input data is split into 90:10 ratio. Where 90% is for training and 10 % is for testing respectively.

Initially, the model was derived with 3 hidden layers containing 19 neurons in each layer. Rectified Linear unit was used as activation function in each layer. On executing the code, we got more variance between value loss and actual loss. To optimize the value loss the model was model was redefined and tested with 28 ➔ 79 ➔ 200 neurons in each run. The epoch size also modified to improve the accuracy. The model became overfitted. Then we applied dropout and early stopping in call back mechanism to reduce the overfitting and value loss respectively. The final model was created with 80% accuracy.

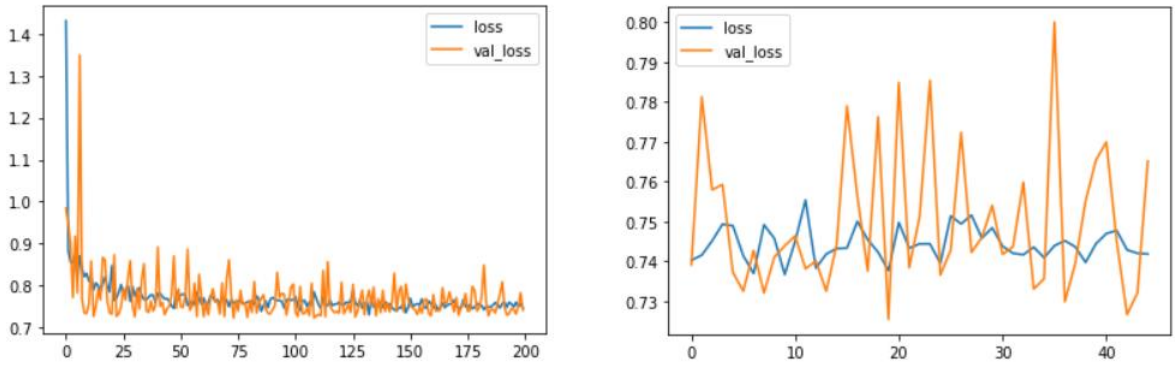

Fig 8. *Value loss error for Usr_frq model*

The same method was applied to predict the time interval with user + product combination data. The code snippet for usr_prd_frq model is given below

```
In [287]: from sklearn.model_selection import train_test_split
          X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.1,random_state=101)
          from sklearn.preprocessing import MinMaxScaler
          scaler = MinMaxScaler()
          X_train= scaler.fit_transform(X_train)
          X_test = scaler.transform(X_test)
          X_train.shape
          X_test.shape

Out[287]: (793, 15)

In [288]: from tensorflow.keras.models import Sequential
          from tensorflow.keras.layers import Dense, Activation
          from tensorflow.keras.optimizers import Adam

In [289]: model = Sequential()

In [290]: model.add(Dense(15,activation='relu'))
          model.add(Dense(19,activation='relu'))
          model.add(Dense(19,activation='relu'))
          model.add(Dense(19,activation='relu'))
          model.add(Dense(1))

In [291]: model.compile(optimizer='adam',loss='mse')

In [292]: model.fit(x=X_train,y=y_train.values,
                    validation_data=(X_test,y_test.values),
                    batch_size=20,epochs=75)

          Train on 7129 samples, validate on 793 samples
          Epoch 1/75
          7129/7129 [==============================] - 1s 182us/sample - loss: 2.4009 - val_loss: 0.3635
          Epoch 2/75
          7129/7129 [==============================] - 1s 84us/sample - loss: 0.4173 - val_loss: 0.2949
          Epoch 3/75
          7129/7129 [==============================] - 1s 90us/sample - loss: 0.3474 - val_loss: 0.2779
          Epoch 4/75
          7129/7129 [==============================] - 1s 85us/sample - loss: 0.3088 - val_loss: 0.3022
          Epoch 5/75
```

Table 6. *Sample Source Code*

In the rest of the 2 models: usr_prd_to_prd_clsfy & usr_prd_to_prd_clsfy the multi-hot encoding was done in each of the product labels. The purpose of this model is to derive the possible combination of the products would be bought by a specific user or by a user who bought a specific set of products in the prior visit. To derive this model **multiclass classification** method was applied.

For these two models, in order to perform Hyperparameters tuning, **Bayesian optimization** Algorithm was used. In Hyperparameter tuning Bayesian Optimization is better than Random search and Grid search methods. It is the latest technique for optimizing the parameters in both regression and classification model. It is based out of Gaussian Process model.

With BO, the model reached its accuracy with 79 neurons in 4 dense layers with Rectified Linear unit activation function in each dense layers and **sigmoid** function in the final layer. **Binary_crossentropy** was applied as a loss function. Here we have not used the **categorical_cross_entropy** as the loss function. This was done to penalize the output from each node independently. And modelled the output of the network as

Bernoulli distribution in each product label. The batch size was kept as 20 with epoch size of 200. The value loss of both the models are given below.

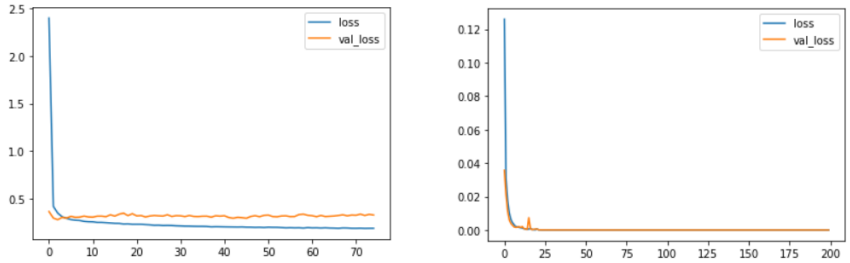

Fig 9. *Value loss for usr_prd_to_prd_clsfy model*

The 4 models can be incorporated in the process layer of the reminder application. The user can be either new to the shop or an existing user. If the user is new, then his profile would be used as input. One of the models recommend the products which he might be interested to purchase. For the existing customer, the model will provide a reminder with the list of products that he might need to purchase. The reminded date would be the least possible date from which he might need the products.

RESULTS

Validation within same aisle data - In usr_frq and usr_prod_frq models, regression technique is used to derive the time interval. So the models are evaluated using explained variance score.

$$explained\_variance(y, \hat{y}) = 1 - \frac{Var\{y - \hat{y}\}}{Var\{y\}}$$

Where y^ is the predicted output, y the corresponding (correct) target output, Var is the variance, square of standard deviation.

The value should be equal to 1.0. The explained variance score of usr_frq is 0.9 which is 90% accurate and explained variance score of usr_prod_frq 0.80 which is 80% accurate is models respectively.

```
predictions=model.predict(X_test)
np.sqrt(mean_squared_error(y_test,predictions))
0.6031898189413805
mean_absolute_error(y_test,predictions)
0.2303483026175
explained_variance_score(y_test,predictions)
0.9077128836788727
```

```
predictions = model.predict(X_test)
np.sqrt(mean_squared_error(y_test,predictions))
0.8801518708380847
mean_absolute_error(y_test,predictions)
0.3688960631588218
explained_variance_score(y_test,predictions)
0.802468822815172
```

Table 7. *usr_frq and usr_prod_frq model evaluation*

The error distribution of the above models is given below

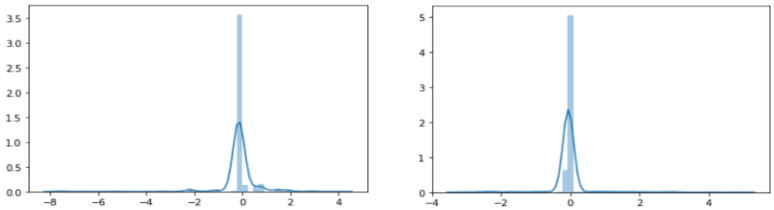

Fig 10. *usr_frq and usr_prod_frq error distribution*

The remaining models are evaluated during the learning process by using the validation feature in Keras. For the product classification using user and product data: usr**_prd_to_prd_clsfy** is around 98%

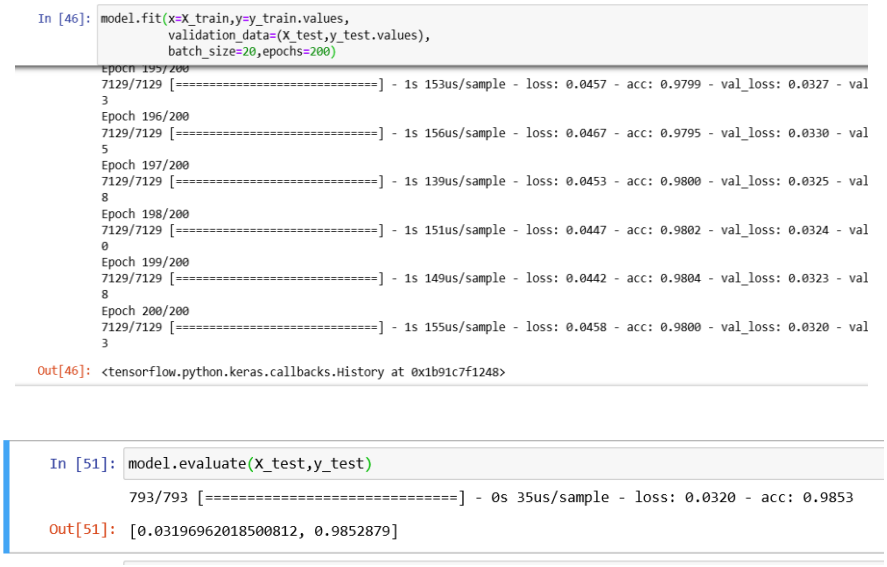

Table 8. *Evaluation of usr_prd_to_prd_clsfy model*

For the product classification using user and product data: usr**_to_prd_clsfy** is around 98%.

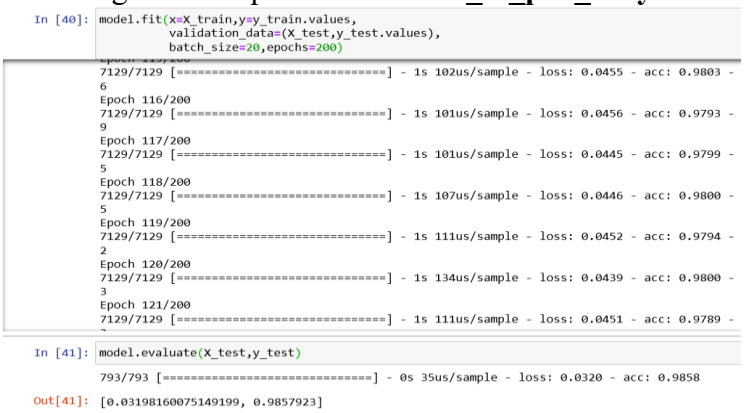

Table 9. *Evaluation of usr_to_prd_clsfy model*

Cross validation with different aisles and department data- Using these models few customers data were cross validated. The results were convincing. Out of 5 customers, the model predicted correctly for 4 customers with the near next date of purchase along with the product combinations that could be purchased in the next visit. To check the robustness of the model for other types of products, the models are validated with data from aisle of fresh fruits.

Top 10 Aisles		Top 10 Departments	
fresh fruits	3642188	produce	9479291
fresh vegetables	3418021	dairy eggs	5414016
packaged vegetables fruits	1765313	snacks	2887550
yogurt	1452343	beverages	2690129
packaged cheese	979763	frozen	2236432
milk	891015	pantry	1875577
water seltzer sparkling water	841533	bakery	1176787
chips pretzels	722470	canned goods	1068058
soy lactosefree	638253	deli	1051249
bread	584834	dry goods pasta	866627

Table 10. *Top Product categories*

The accuracy of the fresh fruit data is given below

Model	Accuracy
usr_frq	0.9617353
usr_prod_frq	0.9534593
usr_to_prd_clsfy	0.9328754
usr_prd_to_prd_clsfy	0.9313247

Table 11*. Accuracy from fresh fruit data*

Further, to make sure whether the model works well with the data from other departments. The models were evaluated with top 4 departments. The consolidated results are given below

Model	produce	dairy eggs	snacks	beverages
usr_frq	0.8272853	0.8229653	0.8038853	0.8975253
usr_prod_frq	0.8300093	0.8256893	0.8066093	0.9002493
usr_to_prd_clsfy	0.8094254	0.8051054	0.7860254	0.8796654
usr_prd_to_prd_clsfy	0.8078747	0.8035547	0.7844747	0.8781147

Table 12. *Accuracy from Top 4 department data*

This shows that the accuracy of the model is varying based on the type of the data fed into the model.
Validation with entire dataset - To proceed further random sample of 800 records across all departments using seed 101 were extracted and evaluated for usr_to_prd_clsfy model. The results are given below

Score	
Accuracy	0.58
Precision	0.57
Recall	0.64
F1 Measure	0.61

Table 13. *Accuracy from entire dataset*

CONCLUSION

Based on the design for creating the reminder list, 4 models were created with user and product combination. Initially, the models were derived with most frequently purchased products. Items from Fresh vegetables aisle

was used, as those are the ones which could be bought frequently. Unlike shopping in brick and motor, the mobile phones make the customer to purchase the products, as and when they need them. So, in our thesis, we have used the fresh vegetable as the prime data. And also, the models were cross verified with different dataset. The results are more convincing. So, these models can be appended with the shopping portal of any merchandise. It will promptly remind the user on time with the list of products that he may need to purchase. We have reviewed most of the researches on grocery list. Nearly 70% of the researches were oriented in providing alternate solution for the paperless grocery list. Couple of researches in the emergence of deep learning provided methods to recommend new products in addition to the products listed by the user in the paperless application. But none of them provided a method to predict the time and the products of purchase by the customer. So, the idea proposed in this paper would be a new feature from the customer point of view.

REFERENCES:

1. [Rfr1]. Authorship or Source (year) Shopping List. [online], date of latest update, 5 November 2019, at 14:47 (UTC): https://en.wikipedia.org/wiki/Shopping_list
2. [Rfr2]. The Hybrid Shopping List: Bridging the Gap between Physical and Digital Shopping Lists Felix Heinrichs, Daniel Schreiber in Conference: Proceedings of the 13th Conference on Human-Computer Interaction with Mobile Devices and Services, Mobile HCI 2011, Stockholm, Sweden, August 30 - September 2, 2011
3. [Rfr3]. An Intelligent Shopping List - Combining Digital Paper with Product Ontologies Marcus Liwicki1 Sandra Thieme1, Gerrit Kahl2, and Andreas Dengel1,3 Knowl. -Based Intell. Inf. Eng. Syst., pp. 187–194, 2011.
4. [Rfr4]. Jain, R. Ghosh, and M. Dekhil, “Multimodal Shopping Lists,” in Proceedings of the 13th International Conference on Human-Computer Interaction. Part II: Novel Interaction Methods and Techniques, Berlin, Heidelberg, 2009, pp. 39–47.
5. [Rfr5]. Huaigu.W and Yuri.N, “Mobile shopping assistant: integration of mobile applications and web services,” 16th international conference on Worldwide Web, 2007, pp. 1259–1260.
6. [Rfr6]. Harsha Jayawilal W.A and Saminda Premeratne "The Smart Shopping List" 2017 IEEE 13th Malaysia International Conference on Communications (MICC), 28-30 Nov. 2017, The Puteri Pacific, Johor Bahru, Malaysia
7. [Rfr7]. R. Agrawal, R. Srikant et al., “Fast algorithms for mining association rules,” in VLDB, vol. 1215, 1994, pp. 487–499.
8. [Rfr8]. Warnia Nengsih Department of Computer Politeknik Caltex Riau-Indonesia "A Comparative Study on Market Basket Analysis and Apriori Association Technique " (IEEE) 03 September 2015 DOI: 10.1109/ICoICT.2015.7231468
9. [Rfr9]. Behera Gayathri "Efficient Market Basket Analysis based on FP-Bonsai " (IEEE) 05 October 2017 DOI: 10.1109/I-SMAC.2017.8058287
10. [Rfr10]. M.D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean et al., “On rectified linear units for speech processing,” in 2013 IEEE International Conference on Acoustics, Speech and Signal Processing. IEEE, 2013, pp. 3517–3521
11. [Rfr11]. Instacart Market Basket Analysis, November 20, 2017 [Online] https://www.kaggle.com/psparks/instacart-market-basket-analysis