International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 05 Issue: 05 | May - 2021 ISSN: 2582-3930

AN ITERATIVE APPROACH FOR PRODUCT RECOMMENDATION USING COLLABORATIVE FILTERING IN MACHINE LEARNING

Preethi Potnuru[1], S Prasad Babu Vagolu[2] and Prof. Vedavathi K[3]

1. PG Student, Department of Computer Science, GITAM Institute of Science, GITAM
2. Asst Professor, Department of Computer Science, GITAM Institute of Science, GITAM
3. Professor and HOD, Department of Computer Science, GITAM Institute of Science, GITAM

ABSTRACT

An accurate and timely demand forecast represents the key input information for most decisions made within a company. It can only be obtained through a systematically managed process of demand forecasting. However, what still has not been researched sufficiently is how to manage this process in the right way in chemical companies that are characterized by implementation of a make-to-stock strategy based on demand forecasting. Therefore, the given area became the subject of a quantitative primary research conducted at 58 Czech manufacturing chemical companies. It aimed to identify the way of organization of the demand forecasting process, the way of developing demand forecasts, and the way of assessment of the accuracy of created forecasts. The research conclusions and their subsequent comparison with modern theoretical approaches made it possible to identify the main directions in improvement of the process of demand forecasting in many companies.

The demand forecasting organization is concerned with how the company organizes development and application of demand forecasts. companies typically organize their demand forecasting function in one of four ways: an independent approach, a concentrated approach, a negotiated approach, or a consensus approach to demand forecasting management.

INTRODUCTION

Product recommendation engines have increasingly determined its importance in boosting revenues by offering consumers chances to have better experience when shopping online. That is also the reason why many online merchants try to make the best of product recommendation engines and gain positive results. Among many e-commerce enterprises applying these tools, Amazon seems to be the most successful one with an impressive 35% of sales comes from the effect of product recommendation engine. Let's see how Amazon has managed to raise customer's demand by product recommendation engines.

Types of recommendation engines

To gain success, Amazon takes advantages of both onsite and offsite recommendations. On-site recommendation is the act of giving suggestions for web browsers during their online sessions. On the other hands, off-site recommendation happens when a series of suggested items are sent to consumers via email after they already bought a product. Although, these two ways of recommending possibly-bought products for visitors have differences in delivery, they have the same intention in increasing the desire to purchase a product among shopping doers. Additionally, besides giving various related products that consumers want to purchase, Amazon includes "add to cart" button for every single product that higher user's demands to buy products immediately.

On-site recommendation

It is noticeable that Amazon has made use of numerous product recommendation engines to offer on-site suggestions to visitors. Some of the most outstanding tools should be listed are: your recommendation, frequently bought together, top best sellers, who bought this item also bought,…

It can be said that this tool makes customer irresistible to purchase more products. Only by a click to "your recommendations", a list of recommended products from different categories that users used to look for will be presented. In other words, with a simple click, web browsers can figure out the products that are of their tastes and preferences. As a result, they have a tendency to purchase more.

This recommendation method functions as a tool to increase the average order value by giving suggested product based on the already chosen items. For example, after a consumer decided to add an Iphone X in their cart, a list of frequently-go-together products will be showed as well such as earphones, charger, case, and so on. In short, "frequently bought together" is highly appreciated to up-sell and cross-sell.

LITERATURE REVIEW

There has been a lot of work done in this field. For example, one very popular algorithm is Collaborative Filtering. One type of collaborative filtering is user-based collaborative filtering, which starts by finding a set of customers who have purchased and rated similar items with the target users purchasing history. The algorithm aggregates items from these similar customers, and uses the ratings from other similar users to predict the ratings from this user. Another type of collaborative filtering is item-based collaborative filtering, which was first brought up by Amazon [4] and focuses on finding similar items instead of similar 1customers. For each of the users purchased and rated items, the algorithm attempts to find similar items. It then aggregates these similar items and recommends them.

There are also other algorithms that try to exploit graph structures to predict links or ratings. Random walks algorithms [2] could be used in predicting links in complex graphs in a very efficient manner. And also, if we model the

user and product graph as a bipartite graph, then it is also feasible to use Bipartite Projection algorithm [5] to calculate the relevance between two customers. So the predicted rating is essentially based on the other relevant customers' ratings. In later sections of this paper, we will introduce three models and algorithms which are derived from the prior work mentioned above with application-specific improvements.

PROBLEM IDENTIFICATION AND OBJECTIVES

As a first step, we implement a simple random recommend system as our baseline system. This system returns a random rating for each (product, customer) pair. Because the way it works, we expect it to have the worst performance. We will use this baseline system for comparisons with other algorithms. Online E-commerce websites like Amazon, Filpkart uses different recommendation models to provide different suggestions to different users. Amazon currently uses item-to-item collaborative filtering, which scales to massive data sets and produces high-quality recommendations in real time. This type of filtering matches each of the user's purchased and rated items to similar items, then combines those similar items into a recommendation list for the user.

The first recommendation system we build is inspired by Amazons item-based collaborative filtering [4]. In Amazons algorithm, they represent each item with a vector showing who bought/reviewed the item. Similarity between these two products is defined by the cosine of the two vectors. After calculating similarity between all product pairs, we will have an item-item matrix showing the similarity between the items. Finally, the similarities can provide a good reference on some of the other products that a customer would buy. The original algorithm is used to predict the next product that a customer would buy. To adopt it in our application, which is to predict the rating given by some customer for some product, we create an algorithm that make use of the item-item similarity. First lets define some terms that we will use later. Let w(i, j) be the similarity between item i and item j; Iu is the set of products customer reviewed, excluding the one we are going to predict with; rateu(i) is the rate for product i given by customer u. S(i) is the most similar items with item i, including i itself, according to the item-item similarities. Finally, let x be the item that we are trying to predict for customer u. We calculate a weighted sum for each j ∈ S(x). For each item i that customer u have a rating, we give them weights using the similarity between i and j:

SYSTEM METHODOLOGY

In this section, we introduce methods and algorithms used in PCFinder, including CBR, clustering analysis, and profile and collaborative filtering. 4.1. Dynamic Order-Based Similarity Measure Local similarity measures largely depend on the application domain, but they all serve the same purpose: to return an estimation between 0 and 1 and to indicate the similarity between a particular attribute of a case and its equivalent in the request [2]. Here, we present a method of Order-Based Similarity Measure. This method is based on Order Based Retrieval [4]. The customer supplies a variety of information (preferred values, values to be avoided, maximum values and minimum values, for example) and we construct an ordering relation from this information. Then we can use this ordering to calculate the local similarity. The advantage of Order-Based Similarity Measure is that it is easy to maintain the similarity attribute value pair table. We can get the similarity attribute value dynamically rather than from pre-initialization. For example, when a new attribute of some case is added in the case base, we do not need to update the similarity attribute-value pair table saved in XML documents or other databases. Consider a set V of values for a specific attribute of some case. Assume that {a a an , , , = 1 2 L } ai−1 < ai for each , , where n is the maximum number of possible values within the range acceptable to the customer. We say that i represents the serial number of value in V. Let q be the customer's "ideal" value for this attribute. If q , let m be its serial number, i.e. i i ≤ n ai ∈V 1< q = am . For example, we wish to find a computer whose processor speed is 800MHz, but we are willing to consider speeds between 700MHz and 1000MHz. The possible values of processor speed and the corresponding local similarity measure are shown in Table 1. In this case, speeds above 1000MHz have zero similarity, so that n = 7, V = {700,750,800,850,900,950,1000}, q = 800, m = 3, and therefore S(q, . 4.2. Modification of the Weight.

The Global Similarity Measure is computed typically by taking a weighted average of the local similarity measures. The weights provide some indication of the relative importance of the different attributes. These are the quantities that we want to modify when the user is not satisfied with some of the proposed specific attributes. Following a critique from the user, the main task of the system consists of computing how much change we should make to the weight(s) of certain attribute(s) that the user considers to be the most important to him. According to this, we should compute the similarity with the weight, , of each attribute using this formula: Simi Sim Where and are the i i c th attributes of the query and the case respectively, and where is the similarity measure between these attributes. Let denote the similarity of the criticized attribute. Once the similarity of each attribute is computed, we cover one by one each attribute whose similarity is greater than . We reduce the weight of these attributes as follows: * Sim * Sim * ωi − (Sim Simi > Sim 3 3. Otherwise, return "cannot explicitly be configured". Finally, if there were at least one attribute whose similarity is greater than , the weight of the criticized attribute is increased by an amount equivalent to the sum of all reductions to the weights of the attributes that lost importance, so that the sum of all weights stays equal to 1: * Sim Adaptation of Dependent Attributes: 1. Search the capability of independent attribute on Finally, if there were at least one attribute whose similarity is greater than , the weight of the criticized attribute is increased by an amount equivalent to the sum of all reductions to the weights of the attributes that lost importance, so that the sum of all weights stays equal to 1

Profile and Collaborative Filtering

The customer's shopping habits, such as preferences or constraints, usually last a long time. We collect them in the long-term profile. This provides very important and useful information to us when we try to get the customer's requirement. Each user is associated with a single profile, and each profile contains user information such as personal identification and selection information. When a consumer configures his computer, he can provide to the search agent a preferred value, a forbidden value, as well as maximum and minimum values for each attribute. For example, a consumer might prefer a notebook computer that has a processor speed of 900MHz but not 1000MHz; and he might expect a price range between $1500 and $2000. But such preferences are temporary, and t PCFinder also applies collaborative recommendation based on user profile. A user profile stores the background of an individual user on the server as a profile database. The key issue in recommendation is the ability to combine a target user with a group of other users that have a profile similar to the target user [6]. The three steps of collaborative recommendation are described as follows: 1. Identify the group in which the given target user belongs. 2. Produce a list of recommendable products or attributes. These products or attributes are ranked according to their appearance in the past purchasing history. 3. Recommend the top n recommendable products or attributes

Clustering Analysis

Although there exist many kinds of clustering techniques, we use a simple one to illustrate how it supports product recommendation. More specifically, PCFinder groups customers according to the profiles of their background, such as the main intended use of computer and occupation of users, etc. These user's attributes are called external attributes. When a new customer signs up, PCFinder provides some suggestions about the computer configurations according to an analysis of the purchasing history of all previous customers whohave a similar profile. These computer configurations are called internal attributes. Some cases and solutions about clustering analysis are described as follows:

Case 1: PCFinder suggests the most popularly used internal attribute by clustering their external attributes. For example, PCFinder groups customers who use their computer for playing games and discovers that most of them buy computers whose processor speed is 1700MHz. Hence, such computers are recommended to new customers who intend to use their computer for playing games.

Case 2: PCFinder suggests the most popularly used internal attribute and gives this internal attribute a higher weight than the others by clustering their external attributes. For example, PCFinder groups customers who work as university professors and discovers that most of them have bought either Compaq or SONY. Hence, these two brands are recommended to new customers who are universityprofessors, and the initial weight of "brand" is increased.

Case 3: This case is more complicated than the first two. In cases 1 and 2, clusters take account of a single attribute. In this case, PCFinder maps groups of related user profiles to groups of related computer configurations. For each new customer, PCFinder finds out to which group he belongs when he fills out his profile form. By analysing cluster correlations between external and internal attributes, PCFinder recommends the configuration of the cluster that correlates best with the customer's external attributes. See Figure 3. Figure 3. The correlation between external and internal attributes For example, PCFinder could group customers who are Implementation of PCFinder In order to illustrate the architecture and methodology of our product recommendation system, we constructed an intelligent agent – PCFinder – that runs on an online notebook computer store to provide suggestions to customers as well as management staff members. 5.1. Running Environment, Developing Tools and Domain To implement our online computer store with PCFinder, we chose Apache Tomcat 4.0 as our Application Server, which was used by 62% of the websites on the Internet in December 2002 [11]; we use Java Server Pages and Java Beans as our developing tools. We also apply XML as a Standard Generalized Markup Language, which is transformed to HTML by a XSLT processor. One hundred and fifty cases are stored in a relational database. Each case includes the following eight attributes: processor speed, memory, hard disk drive, display size

Product Recommendation

In this section, we explain how the agent achieves the product recommendation process. PCFinder provides two ways to assist the customer in refining the result. The first way is weight modification. The product consists of several attributes. Different attributes have different weight in the customer's mind. According to CBR theory, the product is recommended with respect to the integration of similarities of the whole attributes. One of the possibilities occurs if the most important attribute has the same weight as the other attributes at a time when the customer is unsatisfied with the result. In this case, weight modification can be applied to help the customer find a more satisfactory solution. Consider the situation illustrated in Figure 6, for instance. The solution proposed as "result" is closest in similarity to the customer's "query" among all 150 cases in the case base. But the customer declares himself unhappy with the switch from IBM to Compaq. Therefore, PCFinder reduces the weight of all the attributes that had a local similarity higher than that of the brand (in this case, all the attributes) and increases the weight of "brand". The case base is searched again with these new weights and the best match that is found is illustrated in Figure 7. Another way to increase the consumer's satisfaction is adaptation of the result. Some attributes of a product can be adapted, others cannot. Therefore, if the customer is still not satisfied with the attributes, some of them can be adapted. In this case, PCFinder helps the consumer in adapting the recommended computer until he is satisfied. In our system, the attributes of memory, hard drive, multimedia and operating system can be adapted. Figure 8 shows the result of adaptation starting from the unsatisfactory solution that was previously offered in Figure 7. It is important to point out that this is the only time that the system allows itself to recommend a product that may not be in the case base, which explains

why a solution so close to the consumer's query had not been proposed earlier. 5.4. Graphical Analysis To visualize the relationship between user profiles (external attributes) and the product attributes (internal attributes), we construct a graphic-building wizard for management staffs' marketing research. After selecting any items (one or more) of the user profiles and any one attribute of the product (See Figure 9), we get a statistical diagram (See Figure 10). The X-axis represents the selected attribute of the product. The Y-axis represents the quantity of sales. There are two curves in this diagram. For example, the selected item is playing games (the intended use of the computer is to play games); the selected attribute of the 6.

The Recommendation Algorithm

Most recommendation algorithms start by finding a set of customers whose purchased and rated items overlap the user's purchased and rated items.2 The algorithm aggregates items from these similar customers, eliminates items the user has already purchased or rated, and recommends the remaining items to the user. Two popular versions of these algorithms are collaborative filtering and cluster models. Other algorithms — including search-based methods and our own item-to-item collaborative filtering — focus on finding similar items, not similar customers. For each of the user's purchased and rated items, the algorithm attempts to find similar items. It then aggregates the similar items and recommends them

Algorithm

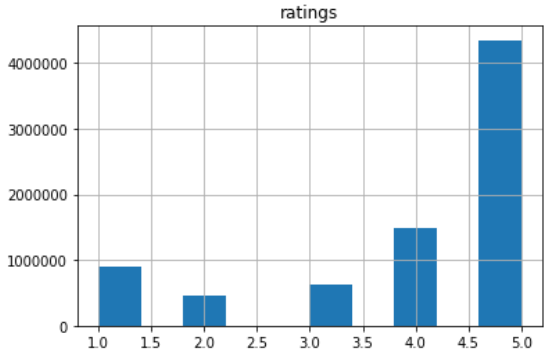
For each item in product catalog, I1
For each customer C who purchased I1
For each item I2 purchased by
customer C
Record that a customer purchased I1 and I2
For each item I2
Compute the similarity between I1 and I2

How well did the algorithm work

The key to item-to-item collaborative filtering's scalability and performance is that it creates the expensive similar-items table offline. The algorithm's online component looking up similar items for the user's purchases and ratings scales independently of the catalog size or the total number of customers; it is dependent only on how many titles the user has purchased or rated. Thus, the algorithm is fast even for extremely large data sets. Because the algorithm recommends highly correlated similar items, recommendation quality is excellent.

Unlike traditional collaborative filtering, the algorithm also performs well with limited user data, producing high-quality recommendations based on as few as two or three items

Implementation



1. Read and explore the given dataset. (Rename column/add headers, plot histograms, find data characteristics)
2. Take a subset of the dataset to make it less sparse/ denser. (For example, keep the users only who has given 50 or more number of ratings)
3. Split the data randomly into train and test dataset. (For example, split it in 70/30 ratio)
4. Build Popularity Recommender model.
5. Build Collaborative Filtering model.
6. Evaluate both the models. (Once the model is trained on the training data, it can be used to compute the error (like RMSE) on predictions made on the test data.) You can also use a different method to evaluate the models.
7. Get top - K (K = 5) recommendations. Since our goal is to recommend new products to each user based on his/her habits, we will recommend 5 new products.
8. Summarize the insights.

```
from sklearn.model_selection import train_test_split
train_data, test_data = train_test_split(top_ratings_df, test_size = 0.30, random_state=0)
```

```
train_data.head()
```

	userID	productID	ratings
3879937	A2W0XCWOK3MW6F	B003Y74AXO	3.0
3603561	A2E1EFNIZL2FVA	B003VANO7C	5.0
6945634	AR3EVUQF0AC7R	B00ARB5FLQ	4.0
2732999	A38RMU1Y5TDP9	B002NGVY8G	1.0
7715921	A1JZFGZEZVWQPY	B00GRNUO2I	3.0

```
test_data.head()
```

	userID	productID	ratings
6562653	AWH2AY17ZU7W2	B009A6CZ30	5.0
1001830	A1SHHQSPOWR00F	B000HGIWN4	3.0
3904732	A1PVJIC1412IN4	B00466X9SY	5.0
7600678	AGYH5U11ZKPFB	B00F3ZN0CC	4.0
2743475	AMKNPIDFLRFMP	B002O3W2OI	2.0

Splitting the Data into test and train

RESULTS AND DISCUSSIONS

We had read and explored the dataset. Considered only first three columns userId, productId, and ratings.

Analysed the data and plotted the histogram based on ratings and usedID. We had Split the data randomly into train and test dataset.

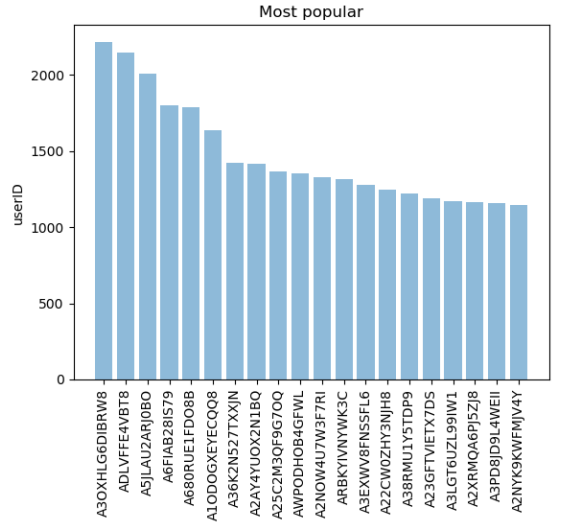
Build Popularity Recommender model and found the RMSE value for Popularity Recommender model as **1.091**

Build Collaborative Filtering model. The RMSE value for Collaborative Filtering model, by **KNN With Means** is **0.9941** and **SVD** is **0.9606**. After parameter tuning of SVD it is **0.858**

We had recommended new products to each user based on his/her habits and have recommended 5 new products.

Between RMSE of Popularity and Collaborative filtering, Collaborative filtering fares better with 0.86 scores.

	userID	productID	ratings
94	A3BY5KCNQZXV5U	0594451647	5.0
118	AT09WGFUM934H	0594481813	3.0
177	A32HSNCNPRUMTR	0970407998	1.0
178	A17HMM1M7T9PJ1	0970407998	4.0
492	A3CLWR1UUZT6TG	0972683275	5.0




```
A28UMA3GW9L124 ['B00HFRMWAM', 'B000M2GYF6', 'B008JJLW4M', 'B0002D6QJO', 'B000N99BBC']
A38NHXL5257E3B ['B00BOHNYU6', 'B00CB2F65O', 'B004Q3R9AQ', 'B003SGCO3E', 'B00829THK0']
A36IHC0K68NS2 ['B00HMREOLK', 'B004ING996', 'B00ATM1MGA', 'B009N8M2P6', 'B005DKZTNK']
A231WM2Z2JL0U3 ['B00005OQMO', 'B00008OE6I', 'B00004RC2D', 'B00004TDL2', 'B00008OE5G']
A2AC66Q24545GA ['B001TOD7ME', 'B009E67J8U', 'B009VV56TY', 'B00152RCWI', 'B008U303BI']
A3OXHLG6DIBRW8 ['B004CLYEFK', 'B001T9NUJE', 'B002VPEIX4', 'B001ID829O', 'B00461673E']
A2XA8CW5DF4MNZ ['B000TKHBDK', 'B00387EW1K', 'B001TOD7ME', 'B000MKKTJK', 'B000AP85BO']
AG35JCCQWDRCS ['B000068O16', 'B0036Q7MV0', 'B003ZSHNE0', 'B0045TYDNC', 'B001MSVPM6']
A2X3L31KCXBHCL ['B00829THEG', 'B0044DEDC0', 'B005EM6IAA', 'B000B63KSM', 'B002V1APJ2']
A33YZNZIRA3H97 ['B00BOHNYTW', 'B000O161X0', 'B00006I5J7', 'B004MT7GGG', 'B008FO14W8']
AX0SDBUBIRUWY ['B007WTAJTO', 'B006W8U2MU', 'B001L1H0SC', 'B00IKCQ0EK', 'B001FO4QHI']
A296QED1MV1V0J ['B0058G40O8', 'B00AXTQQDS', 'B008AST7R6', 'B005BOMTT0', 'B007VGGFZU']
A1CPRP3VFJRS1R ['B003CFATT2', 'B004HO58SO', 'B008C13C4O', 'B002Z4ZDFY', 'B0007QKXYU']
AB094YABX21WQ ['B001TH7GUU', 'B000S5Q9CA', 'B000JM3WV2', 'B004PYD9I2', 'B002Y6CJJ8']
A1MEISNED4NP7U ['B00483NRZ6', 'B004LWYE4Q', 'B007R5YDYA', 'B00834SJSK', 'B007PJ4P4G']
A3LGT6UZL99IW1 ['B000V5P90K', 'B000VUIXOO', 'B004W23KWG', 'B00CDS9HTM', 'B000L47AHG']
A1PPS91NLI7KEH ['B00AAHT8JC', 'B0098PRKA6', 'B0064EL2DK', 'B009XEBBC0', 'B00CD8ADKO']
A1VJ0V58N0698J ['B0075SUK14', 'B000QY9KIS', 'B00328HR7G', 'B002H78JA2', 'B004VM0SE6']
A196JN53PG0C7R ['B003ESSZUU', 'B008JJLW4M', 'B001TH7T2U', 'B001FVI91U', 'B001AYGDCE']
A1901NTE8LFJF6 ['B000M2TAN4', 'B0011U65F2', 'B005DKZTMG', 'B004XZL980', 'B00FF9MLOW']
```

Since our goal is to recommend new products to each user based on his/her habits, we will recommend 5 new products.

CONCLUSION AND FUTURE SCOPE

Recommendation systems help users discover items they might not have found by themselves and promote sales to potential customers, which provide an effective form of targeted marketing by creating a personalized shopping experience for each customer. Lots of companies have such kind of systems, especially for e-commerce companies like Amazon.com, an effective product recommendation system is very essential to their businesses. In this paper, based on the research on some existing models and algorithms, we design three new recommendation systems, Item Similarity, Bipartite Projection and Spanning Tree. They can be used to predict the rating for a product that a customer has never reviewed, based on the data of all other users and their ratings in the system. To examine and compare their effectiveness, we implement these three algorithms and test them on some existing datasets. In our experiments, we found that, in terms of effectiveness measured with mean squared error (MSE), for all users, Item Similarity has the best result, then followed by Spinning Tree, and Bipartite Projection is the worst. For new users, Spinning Tree has the best result, then followed by Item Similarity, and Bipartite Projection cannot even generate result because of lack of data. For old users, Bipartite Projection has the best result, then followed by Item Similarity, and Spinning Tree is the worst. In terms of computational performance, Bipartite Projection is the fastest algorithm that gives result within fraction of seconds, while Item Similarity can be very computationally expensive. In the future, we plan to improve the effectiveness and performance by exploring a hybrid system which will apply different algorithms on different user segments. One concrete thought is to use Spinning Tree on new users and use Bipartite Projection on old users. And we also need to experiment on different criteria to decide whether a user is a new user or an old user, and then choose the criterion that has the best result. We also would like to study how we could control or tweak the outputs of recommendation systems based on application-specific requirements. For example, the company might want to avoid recommending some very popular items to distribute the traffic to other products, or the company would like to promote some newly listed products. In general, it is an promising direction to build recommendation systems that can adapt to more granular and flexible application-specific requirements.

REFERENCES

Stanford large network dataset collection. http://snap.stanford.edu/data/index.html.

[1] Jianfeng Hu, Bo Zhang, Product Recommendation System, 12th December 2012.

[2] Vehmaskoski, Kari, and Toni Pekkola. "Smart Home: A Learning and Development Environment." Handbook of Smart Homes, Health Care and Well-Being (2017): 505-519.

[3] Lars Backstrom and Jure Leskovec. Supervised random walks: Predicting and recommending links in social networks. Proceeding of WSDM 2011, pages 635 644, 2011.

[4] Chandrasegar Thirumalai, Rashad Manzoor, “Cost Optimization using Normal Linear Regression Method for Breast Cancer Type I Skin,” IEEE IPACT 2017.

[5] Software metric Numerical Data analysis using Box plot and control chart methods, VIT University, DOI:10.13140/RG.2.2.27422.95041

[6] P. Dhavachelvan, Chandra Segar T, K. Satheskumar, "Evaluation of SOA Complexity Metrics Using Weyuker’s Axioms," IEEE International Advance Computing (IACC), India, pp. 2325 – 2329, March 2009

[7] Remagnino, Paolo, et al. "Machine Learning for Plant Leaf Analysis." Computational Botany. Springer Berlin Heidelberg, 2017. 57-79.

[8] Chandrasegar Thirumalai, “Physicians Drug encoding

system using an Efficient and Secured Linear Public Key Cryptosystem (ESLPKC),” International journal of pharmacy and technology, Vol. 8 Issue 3, Sep. 2016, pp. 16296-16303
[9] Halstead Metric for Intelligence, Effort, Time predictions, DOI:10.13140/RG.2.2.17988.42881
[10] Yin, Xizhe, et al. "Human activity detection based on multiple smart phone sensors and machine learning algorithms." Computer Supported Cooperative Work in Design (CSCWD), 2015 IEEE 19th International Conference on. IEEE, 2015
[11] Lars Backstrom and Jure Leskovec. Supervised random walks: Predicting and recommending links in social networks. Proceeding of WSDM 2011, pages 635–644, 2011. [3] Jure Leskovec, Lada A.Adamic, and Bernardo A. Huberman. The dynamics of viral marketing. ACM Transactions on the Web (ACMTWEB), 1(1), 2007. [4] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-toitem collaborative filtering. IEEE Internet Computing, 7(1):76–80, 2003. [5] Tao Zhou, Jie Ren, Matus Medo, and Yi-Cheng Zhang. Bipartite network projection and personal recommendation. Physical Review E, page 76, 2007.