# Customer Behaviour Prediction And Alternate Product Suggestion For E-commerce

Dr. Rajeswari Sridhar,

Shobana Chandrasekaran, Devi Sivakumar, Pallavi Venumadhavan

National Institute of Technology, Tiruchirappalli

## Abstract

The ever-expanding world of E-Commerce presents a treasure trove of opportunities to innovate and improve the shopping experience. With shoppers wanting a more personalized experience, and sellers looking to boost revenue, recommendation systems naturally serve both goals, making it a win-win scenario. Paper describes a system that takes this concept one step further and describes its end-to-end use in the retail sector. The system suggests alternate products that are costlier to some customers, and products that are cheaper to other customers. To do this, it first categorises customers into three groups, and then uses information from that to further suggest products. All of the customer and product related information is stored in ontology, which helps to preserve the relationships between classes. The ontology has been built keeping in mind the required properties to properly capture and understand the required relations. The proposed system is measured using accuracy, precision, recall and f1 score.

**Keywords:** Prediction, Ontology, Recommendation, E-Commerce

## 1. Introduction

All major E-Commerce platforms use a variety of Machine Learning algorithms to analyze customer purchasing patterns and customize each customer's shopping experience [1]. When two different customers search for the same product, they are shown different product recommendations based on their previous purchase history. To maximize the revenue generated through each transaction, we have created a system that categorizes customers when they are looking at a certain product and based on this, suggests various alternative products, that are very similar in nature and use, and are either higher or lower in price than the product they are currently looking at.

Our system first separates those customers who actually have a purchasing intention, and from whom revenue can possibly be maximized, from those customers who do not intend to buy at all. It further categories the customers with a purchasing intention as customers who might abandon the product for some reason, or "at-risk customers", and customers who will go ahead and purchase the item, or "potential customers".

Since price is one of the biggest factors of the product that causes an individual to be categorized as an "at-risk customer" within our scope, once the customer has been put under this category, we identify other alternate products that are the same as what the customer is currently looking at, but cheaper and suggest these products. This increases the probability of that customer generating revenue.

When a customer is categorized as a “potential customer”, we are fairly certain that the customer will buy the present product, and therefore is comfortable with the current price being shown. It is possible that he/she might be willing to pay slightly more for an alternate product in the same category if it has better features. For this reason, to a “potential customer”, we suggest alternate products with a slightly higher price.

This paper is structured as follows : Section 2 presents an overview of the existing approaches to Prediction and Recommendation. Section 3 explains the design of the entire system including the dataset used, the prediction module as well as the recommendation module. Section 4 provides the results obtained with the given setup and makes comparisons between different systems. Section 5 provides a conclusion to the thesis, noting its limitations and provides scope for future work.

## 2. Related Work

### 2.1 Prediction

In physical retailing, a salesperson can offer a range of customized alternatives to shoppers based on the experience gained over time which helps him/her access the customer easily. This experience has an important influence on the purchase conversion rates and hence consequently the sales figures. To bring this experience during online shopping, many E-Commerce companies are interested in analyzing their customers based on all the information they can obtain.

Many academic studies address the problem of predicting the online customer’s behaviour using machine methods. In a study by Mobasher B, Dai H, Luo T and Nakagawa M (2002) [2], they set up two different clustering models based on user transactions and pageviews to derive useful aggregate usage profiles that can be effectively used to take specific actions in real-time. The results showed that the profiles extracted from user clickstream data can be helpful in achieving effective personalization at the early stages of user’s visits in a virtual shopping environment without the benefit of explicit input by these users or deeper knowledge about them.

Suchacka G and Chodak G (2017) [3] aimed to characterize e-customer behaviours based on Web server log data using an online bookstore data. They extracted a set of session features which are session length in terms of the number of Web pages visited in session, session duration in seconds, average time per page in seconds, traffic type representing the page which had referred the user to the bookstore site, three binary variables representing a set of key operations related to the commercial intent, and a set of product categories viewed during the session. On this dataset, association rule mining was applied to assess the probability of visitors making a purchase and understanding the behaviour of different customer profiles.

In the study by Suchacka G, Skolimowska-Kulig M, Potempa A (2015) [4], the problem of predicting purchasing intention was designed as a supervised learning problem and uses the data from an online bookstore to classify the user sessions as browsing and buyer sessions. Every user session is represented as a vector and support vector machines with different kernel types were used for the classification.

The study by İbrahim TOPAL (2019) [5], is aimed to create a meaningful rule by estimating the purchasing behaviour of online consumers with fewer data. The Fisher Score feature was selected in an open database and then training and test data were determined with K fold and a rule was created with Decision Tree. As a result, it suggested that it is possible to determine the purchasing behaviour of online consumers with high accuracy by using a single feature.

In the study conducted by Sakar, C. O., Polat, S. O., Katircioglu, M., and Kastro, Y. (2018) [6], the intention of online consumers to purchase is predicted by data mining and artificial intelligence methods. The customer session information was analyzed using C4.5, random forest, support vector machine and multilayer perceptron classifier methods to predict their purchasing intention. They use a long short-term memory-based recurrent neural network to find out the probability of the visitor’s intention to leave the site.

In our work, we use a similar two-module system to analyze the customer behavior. We predict the likelihood of website abandonment first and then predict the visitor's shopping intent for those consumers who do not abandon the site. We are interested in the at-risk and potential customers and once the type of customer is determined we call the recommendation module in our system to take the actions accordingly.

### 2.2 Recommendation

Recommendation Systems help users to find the items that meet their preferences, among a large number of items available and thereby increases their chances of making a purchase. Techniques such as collaborative filterable-based recommendation, content-based recommendation, demographic-based recommendation, and knowledge-based recommendation are most commonly used. In recent years, techniques such as ontology-based recommenders have gained popularity. Ontology-based recommenders are knowledge-based recommenders which use an ontology to represent knowledge about the user, the products, and the relationship between the two.

Studies of John K. Tarus, Zhendong Niu & Ghulam Mustafa (2018) [7] and Gina George, Anisha M Lal (2019) [8], conclude that for e-learning, ontology-based recommender provides better results in the recommendation process compared to traditional recommendation techniques. They also conclude that hybridization of ontology-based recommenders with other known techniques improve the accuracy and performance of the recommendation process.

Charbel Obeid, Inaya Lahoud, Hicham Khoury and Pierre-Antoine Champin (2018) [9] proposed a hybrid recommendation system, which combines an ontology-based recommender with machine learning techniques, to recommend universities to the students. The proposed approach helps to focus on the students' grades as well as other important features like their skills and interests.

In the study of Márcio Guia, Rodrigo Rocha Silva and Jorge Bernardino (2019) [10] we observe a new approach to recommend products in e-commerce systems, which combines the simplicity of finding users who have similar preferences with the active user, with ontology-based models. This approach generates more knowledge about the active User, the respective Neighbours and the Products recommended to the user and the relation between these three. In this approach the time taken to apply the KNN to find the k-nearest Products is too high.

In our work, we build a similar hybrid system which combines collaborative filtering with ontology-based recommenders. Our recommender first finds the neighbours of the active users and then the products/items that are similar to the product of interest based on the ontology created. To overcome the problem of taking too much time to find the k-nearest Products, we created the top 200 database and product feature database prior to execution.

## 3. Proposed System

Our proposed system has two main modules:

- Prediction module
- Recommendation module

In the prediction module, we have implemented a two-fold prediction technique which simultaneously predicts the visitor's shopping intent and Web site abandonment likelihood. This two-step prediction gives a clear picture of the visitor's behaviour towards the E-Commerce site. If the visitor was identified to abandon the site but has the purchasing intention, then the recommendation module is triggered for suggesting alternate products that are cheaper than the product he/she is looking at ("an at-risk customer"). If the visitor is NOT likely to abandon the site, then the recommendation module is triggered for suggesting alternate products that are slightly costlier than the product he/she is looking at ("a potential customer"). Figure 3.2 explains the aforementioned algorithm of the proposed system.

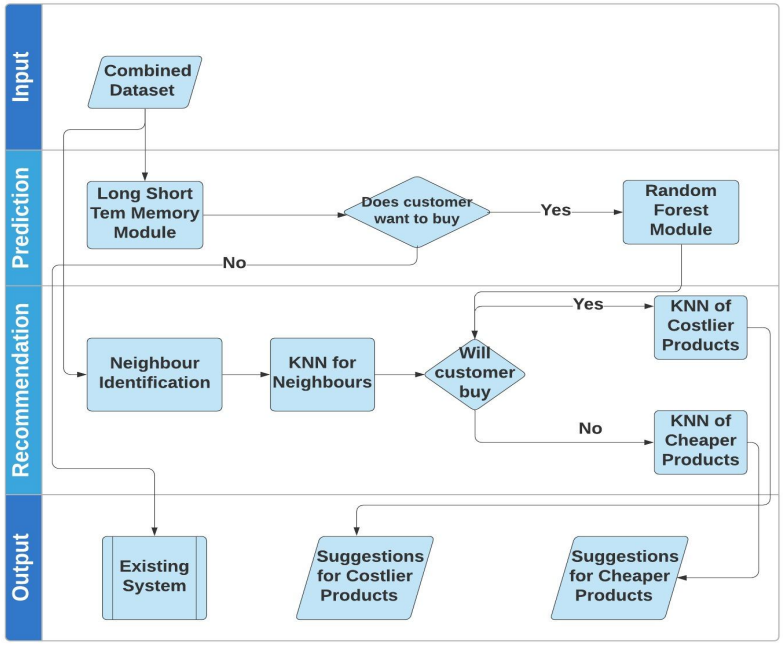


**Figure 3.1** Flow chart of customer behaviour prediction and alternate product suggestion

The recommendation module combines the simplicity of the K-Nearest Neighbor algorithm (KNN) with the efficiency of ontology-based recommenders. The final output of our system is a list of recommended products whose prices are cheaper, for at-risk customers and whose prices are more expensive, for potential customers.

---

**Algorithm: System Implementation**

---

Given initial clickstream data X1; session information data X2;

**1**. Standardize each column of X1
**2**. s1 = LSTM(X1) //abandonment analysis module
**3**. if s1 < 0.5 // visitor likely to abandon
**4**. s2 = RF (X1,X2) //purchasing intention analysis module
**5**. if s2 > 0.5 //visitor has purchasing intention
**6**. call alternate_product_suggestion(less expensive)
**7**. else
**8**. Ignore //visitor has no purchasing intention
**9**. else //visitor is a definite customer
**10**. call alternate_product_suggestion(more expensive)

---

In the following subsections, we provide a detailed explanation about each step of the proposed approach. This includes the dataset used, the prediction module, the ontology described, and the recommendation module.

### 3.1 Dataset

Our system runs on our combined dataset of UCI online shoppers purchasing intention dataset and custom-tailored Amazon Reviews dataset. The dataset used to build the prediction module, the UCI online shoppers purchasing intention dataset, consists of feature vectors belonging to 12,330 sessions. Of the 12,330 sessions in the dataset, 84.5% (10,422) are negative class samples

that did not end with shopping, and the rest (1908) are positive class samples ending with shopping. Table 3.1 and Table 3.2 shows the categorical and numerical features in the UCI dataset respectively.

**Table 3.1** Categorical features used in Customer Behavior Analysis

| Feature name | Feature description | No. of categorical values |
|---|---|---|
| OperatingSystems | Operating system of the visitor | 8 |
| Browser | Browser of the visitor | 13 |
| Region | Geographic region from which session has been started by the visitor | 9 |
| TrafficType | Traffic source by which the visitor arrived at the website | 20 |
| VisitorType | Visitor type as " New Visitor", "Returning visitor" and "Other" | 3 |
| Weekend | Boolean Value indicating whether the date of the visit is weekend | 2 |
| Month | Month value of visit date | 12 |
| Revenue | Class label indicating whether the visit has been finalized with a transaction | 2 |

**Table 3.2** Numerical features used in Customer Behavior Analysis

| Feature name | Feature Description | Min value | Max Value |
|---|---|---|---|
| Administrative | No. of pages visited by the visitor about account management | 0 | 27 |
| Administrative duration | Total amount of time (in sec) spent by the visitor on account management related pages | 0 | 3398 |
| Informational | No. of pages visited by the visitor about Website, communication and address information of the site | 0 | 24 |
| Informational duration | Total amount of time (in sec) spent by the visitor on informational pages | 0 | 2549 |
| Product related | No. of pages visited by the visitor about product related pages | 0 | 705 |
| Product related duration | Total amount of time (in sec) spent by the visitor on product related pages | 0 | 63973 |

| | | | |
|---|---|---|---|
| Bounce rate | Average bounce rate value of pages visited by visitor | 0 | 0.2 |
| Exit rate | Average exit rate value of pages visited by visitor | 0 | 0.2 |
| Page value | Average page value of pages visited by the visitor | 0 | 361 |
| Special day | Closeness of the site visiting time to a special day | 0 | 1.0 |

**3.1.1 Amazon Reviews Dataset Creation**

As there is no pre-existing dataset suitable for our recommendation model, we have manually created our own dataset from the Amazon Reviews repository (2018). The repository contains different kinds of datasets:

- *Raw review data* - contains all 233.1 million reviews
- *Ratings only* - same as above, in csv form without reviews or metadata
- *5-core* - subset of the data in which all users and items have at least 5 reviews (75.26 million reviews)
- *Per-category data* - the review and product metadata for each category.

Our Amazon Reviews dataset was created by first downloading the 5-core dataset along with the corresponding product metadata for the following categories:

- Movies and TV
- Home and Kitchen
- Video Games
- Grocery and Gourmet Food

The attributes of the product metadata and the 5-core dataset are listed in Table 3.3 and Table 3.4 respectively.

**Table 3.3** Attributes of product dataset

| Attributes | Description |
|---|---|
| asin | ID of the product, e.g. 0000031852 |
| title | Name of the product |
| feature | Bullet-point format features of the product |
| description | Description of the product |
| price | Price in US dollars |
| image | URL of the product image |
| related | Related products (also bought, also viewed, bought together, buy after viewing) |
| salesRank | Sales rank information |
| brand | name of the brand |

| | |
|---|---|
| categories | List of categories the product belongs to |
| tech1 | The first technical detail table of the product |
| tech2 | The second technical detail table of the product |
| similar | Similar product table |

**Table 3.4** Attributes of 5-core dataset

| Attributes | Description |
|---|---|
| reviewerID | ID of the reviewer, e.g. A2SUAM1J3GNN3B |
| asin | ID of the product, e.g. 0000013714 |
| reviewerName | Name of the reviewer |
| vote | Helpful votes of the review |
| style | A dictionary of product metadata |
| reviewText | Text of the review |
| overall | Rating of the product |
| summary | Summary of the review |
| unixReviewTime | Time of the review (unix time) |
| reviewTime | Time of the review (raw) |
| image | Images that users post after they have received the product |

Both the datasets, which are available in JSON format, were converted into a pandas dataframe. For each category, the two datasets were combined based on the attribute "asin" which is the productID. Columns that are not used by our recommendation model were removed. Rows with NULL and duplicate values were also removed. Each of these categories were then merged to form our Amazon Reviews dataset which has 16,000 reviews. Figure 3.2 gives one such vector from our Amazon Reviews dataset

| asin | overall | reviewText | reviewerID | reviewerName | main_cat | price | title |
|---|---|---|---|---|---|---|---|
| 0005419263 | 5.0 | The little ones love this | A2CFV9UPFTTM10 | SuzieQ | Movies & TV | $17.26 | Steve Green: Hide 'em in Your Heart Volume 2: ... |

**Figure 3.2** A feature vector of Custom-made Amazon Reviews dataset

### 3.1.2 Databases for Recommendation Module

Our Ontology-based recommendation system creates feature vectors for users and products for the KNN algorithm. For the creation of product feature vectors, we need the top 200 words from the reviews, for each category. To enhance the system performance, we created the top200 database and the product feature database prior to system execution. So, for each category of products, the 200 most common words used in the revisions which have an overall rating of 4 and 5 were stored in the top200 database.

In order to process the reviews, we split them into unique words and remove any punctuation or characters. We, then, remove the stop words, which are words that appear frequently in any document and do not add meaning to a sentence. We also apply Lemmatization, which is the process that obtains the "lemma" of a word by understanding the part of speech and the context of the word in the respective sentence. Finally, we pick only the 200 most common words and save them in the top200 database.

As will be discussed in section 3.3, the product feature vector contains an attribute known as TopFiveWords. This attribute denotes the number of words that are in the 200 most common words of the corresponding product category. The process of selecting these words is similar to the creation of Top200 words. Finally, all the product feature vectors are put together with other product attributes in a new database.

## 3.2 Prediction Module

The customer behaviour prediction uses the UCI online shoppers purchasing intention dataset whose features can be categorized into clickstream data and session information data. According to Table 3.1 and Table 3.2, "Administrative", "Administrative Duration", "Informational","Informational Duration","Product Related","Product Related Duration" and "Page Value" represent the clickstream data. The values of these features are derived from the URL information of the pages visited by the user and updated in real time when a user takes an action, e.g., moving from one page to another.
Rest of the features represent the session information. These metrics are measured by "Google Analytics" for each page in the e-commerce site.

As mentioned earlier, the prediction module comprises abandonment analysis module and purchasing intention analysis module. The first module predicts the likelihood to abandon the site. If the likelihood is greater than the predetermined threshold, the second module, which assigns a score to the purchasing intention of the visitor, is triggered.

### 3.2.1 Data Pre-processing

The dataset is made up of 84.5% of negative class samples that did not end with a purchase, and the rest, positive class samples ending with a purchase. In the E-commerce domain, correctly identifying directed buying visits, which are represented with positive class in our dataset, is as important as identifying negative class samples. Therefore, a balanced classifier is needed to enhance the conversion rates in an E-Commerce Web site. To deal with this class imbalance problem, we use an oversampling method, in which a uniform distribution is achieved by adding more of the minority (positive class in our dataset) class instances.

### 3.2.2 Abandonment Analysis

This module makes use of the clickstream data of the UCI dataset. The page type information is converted into binary value using one-hot encoding technique. The four binary values are appended with their corresponding page duration and page value to generate the feature vector which serves as input for the abandonment analysis. We process the sequential pageview patterns as a

time series data. Also, the knowledge of each pageview information is necessary, throughout, for computing the abandonment likelihood. So, when it comes to retaining the information long short-term memory (LSTM) recurrent neural network (RNN) has been showing greater performance in various applications. Being a classification problem, we have also included well-known classifiers like Support Vector Machines (SVM) and Random Forest (RF) in our analysis. It should be noted that the input to these classifiers function as independent entities, unlike LSTM-RNN whose inputs are related to each other. It is this behaviour to relate inputs that made us choose LSTM-RNN for abandonment likelihood computation.

#### 3.2.2.1 Support Vector Machines

Support vector machine (SVM) classifier, whose classification ability has been shown in many literature studies is also included in our analysis. It is a linear model but is capable of modeling non-linear interactions by mapping the original input space to a higher dimensional space using a kernel function.

The classification report of our SVM implementation is shown in Table 3.5 for classes '0' and '1' depicting 'no purchase made' and 'purchase made' respectively. From Table 3.5, it can be seen that SVM is not suitable for large datasets.

**Table 3.5** Performance of SVM classifier for Abandonment Analysis.

| | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| Class 0 | 0.86 | 0.01 | 0.03 | 411 |
| Class 1 | 0.84 | 1.00 | 0.91 | 2055 |
| accuracy | | | 0.84 | 2466 |
| Macro avg | 0.85 | 0.51 | 0.47 | 2466 |
| Weighted avg | 0.84 | 0.84 | 0.76 | 2466 |

#### 3.2.2.2 Random Forest

Random forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mean prediction of the individual trees. The random forest algorithm proved itself to be effective for many classification problems such as gene classification, remote sensing classification, land-cover classification or image classification. Therefore, random forest is determined to be used as another classification algorithm in our abandonment analysis module.

In our experiment, the number of trees in the forest was set to 100, the maximum depth was 17 and gini-index was the parameter to measure the quality of a split. Table 3.6 shows the performance report of RF for classes '0' and '1'.

**Table 3.6** Performance of Random Forest for Abandonment Analysis

| | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| Class 0 | 0.91 | 0.95 | 0.93 | 2055 |

| | | | | |
|---|---|---|---|---|
| Class 1 | 0.66 | 0.52 | 0.58 | 411 |
| Micro avg | 0.88 | 0.88 | 0.88 | 2466 |
| Macro avg | 0.78 | 0.73 | 0.75 | 2466 |
| Weighted avg | 0.87 | 0.88 | 0.87 | 2466 |

#### 3.2.2.3 Long Short Term Memory Recurrent Neural Network

In a Recurrent Neural Network, the connections between the units form a directed graph along a sequence, allowing it to exhibit dynamic temporal behaviour. It remembers the context while training with the help of its internal state.

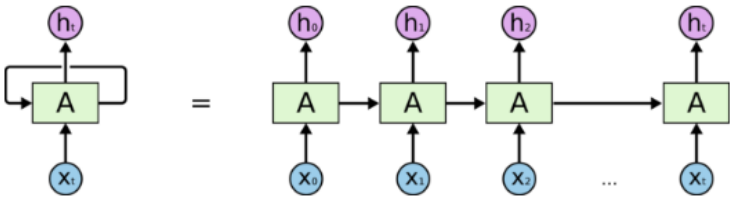


**Figure 3.3** Unrolled neural network

Simple RNNs have a serious drawback of "vanishing gradient". This means that the network experiences difficulty in retaining the words that are far away in the sequence thereby making predictions based on only the recent ones.

LSTMs are capable of learning the long-term dependencies efficiently. In other words, it remembers the information for long periods of time. They are modified versions of Recurrent Neural Networks: the hidden layer of RNN is replaced by an LSTM unit which consists of a memory cell and three types of gates. Our LSTM-RNN network consists of a single hidden layer containing 30 neurons. The timestep of the model has been set to seven. The output is a sigmoid function showing the probability estimate of visitor's intention to leave the site without finalizing the transaction.

If the probability estimate is less than the predetermined threshold (0.5), the purchasing intention module is triggered. If not, the customer is a potential customer. So, the recommendation module is triggered for suggesting products that are slightly more expensive than the currently viewed product. The performance evaluation of LSTM-RNN is shown in Table 3.7

**Table 3.7** Performance of LSTM-RNN for Abandonment Analysis

| | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| Class 0 | 0.91 | 0.96 | 0.93 | 3119 |
| Class 1 | 0.69 | 0.49 | 0.57 | 580 |
| accuracy | | | 0.89 | 3699 |
| Macro avg | 0.80 | 0.72 | 0.75 | 3699 |
| Weighted avg | 0.88 | 0.89 | 0.88 | 3699 |

We present the average accuracy, precision, recall, and F1 Score for each classifier in Section 4

### 3.2.3 Purchasing Intention Analysis

As mentioned earlier in the preprocessing step, the categorical variables are mapped using 1-of-C coding technique and the numerical features are standardized. The dataset is fed to decision trees, support vector machines, and multilayer perceptron classifiers using 70% of the dataset for training and the rest for validation. We employ feature selection techniques to improve the classification performance and scalability of the system. We use the Chi2 function for featuring rankings. It works by taking one feature at a time and tries to classify the output. The following value is computed for each feature and the values are ranked (Figure 3.4):

$$\chi_c^2 = \sum \frac{(O_i - E_i)^2}{E_i} \qquad (3.1)$$

where $O_i$ is the original output and $E_i$ is the predicted output for a particular feature.

We experimented with SVM, Random Forest and MLP using grid search technique for optimizing the hyper parameters with 15, 9 and 6 being the number of features. We observed that better performance was achieved with Random Forest Classifier. Detailed discussion of this is presented in Section 4.

The output of the Random Forest is the likelihood of a visitor making a purchase. If the likelihood estimate is more than the predetermined threshold (0.5) the subsequent action is triggering the recommendation module for at-risk customers. If not, the system halts.

| | | |
|---|---|---|
| 5 | 877404.339415 | ProductRelated_Duration |
| 8 | 175126.808512 | PageValues |
| 1 | 41754.836841 | Administrative_Duration |
| 3 | 35059.775770 | Informational_Duration |
| 4 | 19317.285376 | ProductRelated |
| 0 | 1133.965531 | Administrative |
| 2 | 357.981605 | Informational |
| 24 | 223.548231 | Nov |
| 15 | 115.339482 | New_Visitor |
| 23 | 54.997108 | May |
| 9 | 53.797094 | SpecialDay |
| 22 | 42.613274 | Mar |
| 6 | 29.654336 | BounceRates |
| 7 | 28.985072 | ExitRates |
| 19 | 26.961176 | Feb |
| 25 | 12.571184 | Oct |
| 18 | 11.624839 | Dec |
| 11 | 8.873291 | Browser |
| 14 | 8.120464 | Weekend |
| 21 | 6.432531 | June |
| 26 | 4.744843 | Sep |
| 12 | 3.037565 | Region |
| 17 | 1.428799 | Aug |
| 13 | 1.283194 | TrafficType |
| 10 | 1.037132 | OperatingSystems |
| 16 | 0.728897 | Other |
| 20 | 0.012775 | Jul |

**Figure 3.4** Feature ranking and feature selection

### 3.3 Recommendation Module

Our prediction module categorizes the customer as an “at-risk customer” or a “potential customer”. Based on the prediction module output, the recommendation module is called either for suggesting cheaper products (at-risk customer) or slightly expensive products (potential customer).

Our Ontology-based recommendation model is built using the custom-made Amazon Reviews dataset which has 4 product categories and 16,000 reviews. A product in the dataset has been reviewed by at least 3 customers and a reviewer has reviewed at least 3 products in the same category. This has been made sure for the efficient execution of the KNN algorithm.

Basically we create four classes in our Ontology: Person, User, Neighbour, and Product. These classes are used to represent the knowledge and details about each active user and the relationship between them. In order to recommend items to the active user, firstly we run the KNN algorithm to find the nearest neighbors, and then we rerun it to find the nearest alternate Products to recommend to the active user. Subsection 3.3.2 will talk about the recommendation algorithm in detail.

---

**Algorithm : Alternate Product Suggestion**

---

Given s1, s2 from prediction module and reviewerDF, productDF
**1**. Update reviewerDF and productDF details in ontology
**2**. For active customer, update profile in ontology
**3**. Retrieve Neighbour Information from Ontology
**4**. For each neighbour in Neighbours:
**5**. Retrieve neighbour information
**6**. Build features for each neighbour
**7**. Run KNN on Neighbours
**8**. For each neighbour in Neighbours:
**9**. Retrieve product information
**10**. For each product in Product:
**11**. Retrieve product features
**12**. If (s1 < 0.5 and s2> 0.5)
**13**. KNN on less expensive products
**14**. Else if (sl>0.5)
**15**. KNN on more expensive products

---

The following subsections are organized as follows: subsection 3.3.1 describes our Ontology in detail and subsection 3.3.2 explains the recommendation module implementation.

#### 3.3.1 Ontology model

In order to improve the product recommendation to the active user, we combine an ontology-based recommender with a collaborative filtering approach. An Ontology has domain concepts and relationships between these concepts. With these relationships, more information about the concepts is available. Consider the scenario: a user is looking for earphones in “Beats”. He might also be interested in “SkullCandy”. But “Beats” and “SkullCandy” may not have any relation but they are quite the competitors. Using Ontology that covers these relations are useful in good recommendation of products.

We use Protege v5 software for implementing our Ontology. On the development side, we use OwlReady2, a module for ontology-oriented programming in Python 3.

#### 3.3.1.1 Ontology Classes

In order to develop the ontology-based recommendation system, four main classes are created. The four classes are Person, User, Neighbour, and Product. Figure 3.5 shows the structure of our Ontology with its classes, attributes and relationships.
The classes and attributes are the following:

- **Person:** a class that represents a person who have made reviews of the purchased products, with the following attributes:
    - ReviewerID—identifies the person who made the review
    - ReviewerName—name of the person
    - AveragePrice—the average amount spent by the reviewer
    - ProductId—represents the Ids of products purchased by the reviewer.

- **User:** a subclass of person class that represents the user to whom the system will recommend new products. This class has the same attributes as Person plus the following
    - MinPrice—the price of the product with the lowest value of all products purchased by the user
    - MaxPrice—the price of the product with the highest value of all products purchased by the user

- **Neighbour:** a subclass of person class that represents a reviewer that purchased the product the User is currently looking for. This class has the same attributes as Person plus the following:
    - NeighbourFeatures—represents the features vector that will be used to calculate the Euclidean distance and find the nearest Neighbours of User.
- **Product:** a class that represents a product purchased by a Person. This class has the following attributes:
    - ProductID—identifies the product
    - ProductPrice—the price of the product
    - ProductName—name of the product
    - ProductRating—identifies the score given by the Neighbour in the revision and can be a value between 1 and 5
    - ProductCategory—the category of the product
    - ProductFeatures—represents the features vector that will be used to calculate the Euclidean distance and find the nearest products to recommend to User.

The *hasBoughtProducts* relation is established between the Person and Product class when a Person is identified to have bought products. When the neighbours of a user are identified using KNN algorithm, the *hasNeighbours* relation is established between the User class and the Neighbour class.

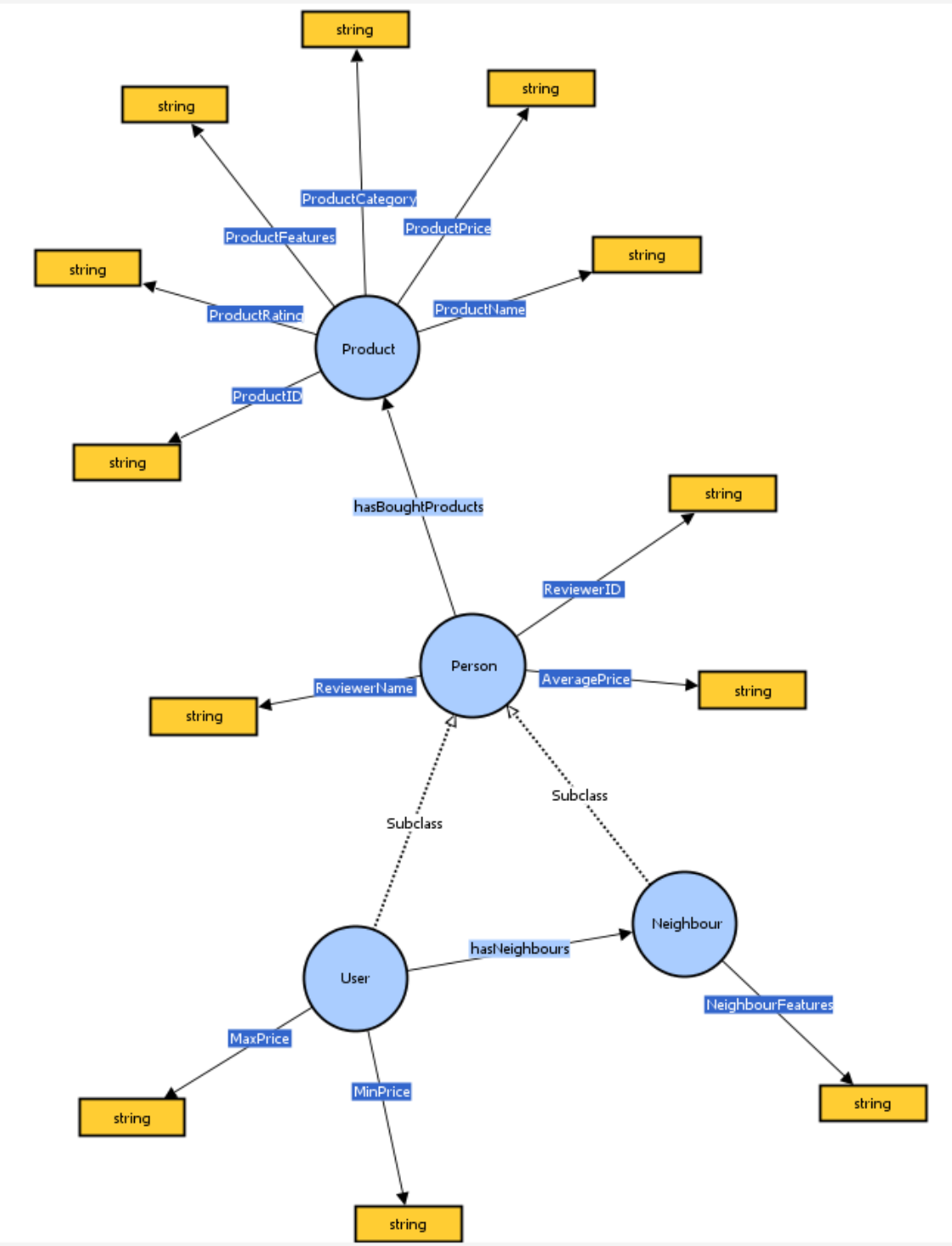


**Figure 3.5** Structure of the Ontology

### 3.3.2 The Hybrid-Ontology based System Design

The implementation of the recommendation system can be broadly divided into 6 steps:

1. Building the Ontology
2. Loading the active-user details
3. Finding Neighbors of the active customer
4. Neighbour KNN
5. Creating the product list
6. Product KNN

*I. Building the Ontology*

As per subsection 3.3.1, we designed with two relations and four classes with its corresponding attributes. As the recommendation system is built from scratch, it is necessary to send all the reviewer and product data to the Ontology. In order to do this, the custom-designed Amazon Reviews dataset is split into Products and Reviewers database. The Reviewers database has "ReviewerName","ReviewerID", "AveragePrice" , "MinPrice", "MaxPrice", "ProductIDs" and "AllCategories" as its attributes. The "AllCategories" attribute is a list of all product categories of the purchased products. The Products database has the features : "ProductID", "ProductCategory", "ProductRating", "ProductPrice", "ProductName" and "ProductFeatures".

As mentioned in subsection 3.1.2, the product features are created beforehand to enhance the performance of the system. For each Product, we create the feature vector with two columns.The first column identifies the overall rating given by the respective Person: a value between 1 and 5. The second column is a number between 0 and 5: 0 if no word of the TopFiveWords is among the most common 200 words of the product category and 5 if all words are. Figure 3.6 shows an instance of a product feature vector.

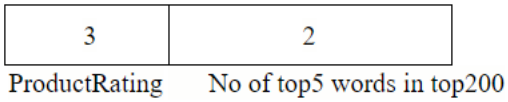


**Figure 3.6** Creation of product feature vectors.

When loading the Products and Reviewers data to the Ontology, the Product details are first loaded in the Product class. Then, the Reviewers details (ReviewerID, ReviewerName, Average Price) are sent to the Person class(shown in Figure 3.7).While sending the Reviewer details, the *hasBoughtProducts* is updated for each and every product bought by a reviewer.

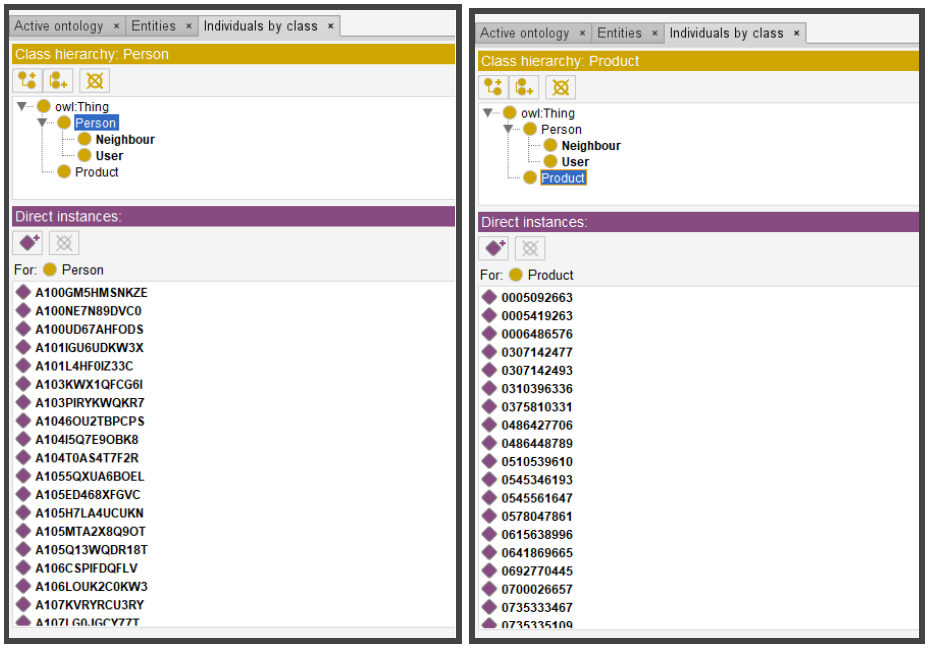


**Figure 3.7** Snapshot of the Ontology after loading Product and Reviewer details

*II. Loading the active-user details*

Active-user is the customer to whom we are recommending the products. This step involves searching the Ontology for the ID same as that of the active user and upgrading him/her from the Person class to User class. This is done by adding "MaxPrice" and "MinPrice" values to the active-user ID. Being a knowledge-based system, the Ontology automatically updates the active-user from Person to User class, upon addition of these values.

*III. Finding Neighbors of the active customer*

The next step is to find those reviewers who have already bought the product the customer is looking for. After extracting these IDs, the *hasNeighbours* relation is updated between the User and the Neighbour classes.

For the subsequent neighbour KNN algorithm, we need to create the feature vectors for each Neighbor. Figure 3.8 shows an example of this procedure.

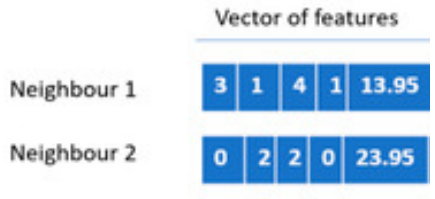


**Figure 3.8** Creation of feature vectors for each neighbor.

The first 4 columns in the Neighbours vector of features concerns the number of products that each neighbor has purchased in each category.The fifth column is the average money spent. After creating all the Neighbours vector features and normalizing the columns in a range of 0 and 1, we send them to the "Neighbor Features" attribute of the Neighbour class.

*IV. Neighbor KNN*For the KNN algorithm, the Neighbour features as well as the active-user's feature vectors are given as input (Figure 3.9). The user's feature vectors are created in the same way as that of neighbour feature vectors. Once the KNN is defined, we calculate the Euclidean distance between each of the Neighbour feature vector and the user vector feature, to identify the nearest Neighbours of the active user.


**Figure 3.9** Neighbour KNN

*V. Creating the Product List*

This is the most important step in the recommendation module as it differentiates product suggestions for an "at-risk" or a " potential" customer. When the nearest Neighbours are identified, as explained in the previous subsection, it is necessary to find out whether the products they have bought are new to the active user. The procedure consists of making sure that the product is not the same as that of the current product, does not belong to a different category, and the product price falls in the 0.5 to 1 range if the user is an at-risk customer or 1 to 1.5 range if the user is a potential customer. Only those that meet the above conditions are stored in a list.

*VI. Product KNN*

After selecting the ProductIDs that the user has not yet bought and are well within the price range, we extract the feature vector of the active product and feature vectors of the products in the product list from Ontology. The last step of our ontology-based recommendation system involves running the KNN algorithm in order to find the nearest Products to recommend to the user.


**Figure 3.10** Product KNN

The Euclidean distance is calculated between the Product feature vectors and the feature vectors of the active product. The system will suggest the K products with the smallest Euclidean distance values to the active User. Therefore, the final output of the system is a list of recommended products whose prices could be either cheaper (for at-risk customer) or slightly expensive (for potential customer). These recommended products are the alternate products that the customer has been looking in the E-commerce site.

# 4. Results

## 4.1 Abandonment Analysis Module

The abandonment analysis module classifies the customers based on their likelihood of abandonment. The SVM, Random forest and LSTM RNN classifiers were tested and we observe the results in Table 4.1. We observe that the SVM Classifier tends to overfit with large datasets and hence is not suitable for large datasets. The Random Forest Classifier gives good results but in a real-world scenario as each page is traversed sequentially, the knowledge of each pageview information is necessary, throughout, for computing the abandonment likelihood. So the classifier needs to be able to retain the clickstream information. Hence we observe that the best results are obtained in case of the LSTM RNN Classifier with an F1-score of 0.88.

**Table 4.1** Performance of the different abandonment analysis modules

| Classifier | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| SVM | 0.84 | 0.84 | 0.84 | 0.76 |
| Random Forest | 0.88 | 0.87 | 0.88 | 0.87 |
| LSTM RNN | 0.89 | 0.88 | 0.89 | 0.88 |

## 4.2 Purchasing Intention Analysis Module

The purchasing intention module classifies the customers who are most likely to abandon the site based on whether or not they have the intention to make a purchase. The module deals with the customer's session information data. The important features are selected using the Chi2 function for featuring rankings. The SVM, Random forest and MLP classifiers were tested taking a different number of features into account using grid search.

The analysis of Support Vector Machines (SVM) for different parameters are presented in Table 4.2. It can be observed that the performance of the SVM classifier is optimum when the number of features is nine.

A Multilayer Perceptron (MLP) is a feedforward neural network which has more than one linear layer. The parameters of the neural network are learned iteratively during the training process. Since it is a two-class classification problem, the output is a sigmoid layer for computing the purchasing likelihood of the visitor. We present results for various number of features fed as input for MLP. As can be observed from Table 4.3, the performance of the MLP classifier is optimum when the number of features is nine.

Random Forest (RF) is made of many decision trees. Decision trees are prone to overfitting when we don't limit the maximum depth as it keeps on growing until it has exactly one leaf node for every observation. As Random Forests combine a lot of Decision trees, each tree will be trained on a different subset of the dataset giving rise to lower correlation between each Decision Tree in a RF. Table 4.4 presents the accuracy, precision, recall and f1-score for different hyper-parameters.

**Table 4.2** Results of SVM for Purchasing Prediction using Grid Search

| No of features | Accuracy | Precision | Recall | F1-score |
|---|---|---|---|---|
| 15 | 0.54 | 0.76 | 0.21 | 0.33 |
| 9 | 0.61 | 0.55 | 0.22 | 0.31 |
| 6 | 0.52 | 0.71 | 0.20 | 0.32 |

**Table 4.3** Results of MLP for Purchasing Prediction using Grid Search

| No of features | Accuracy | Precision | Recall | F1-score |
|---|---|---|---|---|
| 15 | 0.59 | 0.74 | 0.23 | 0.35 |
| 9 | 0.62 | 0.56 | 0.22 | 0.32 |
| 6 | 0.53 | 0.71 | 0.20 | 0.31 |

**Table 4.4** Results of RF for Purchasing Prediction using Grid Search

| No of features | Accuracy | Precision | Recall | F1-Score | max_depth | n_estimators |
|---|---|---|---|---|---|---|
| 15 | 0.88 | 0.79 | 0.56 | 0.66 | 80 | 100 |
| 9 | 0.85 | 0.77 | 0.52 | 0.62 | 90 | 100 |
| 6 | 0.86 | 0.76 | 0.54 | 0.63 | 110 | 1000 |

From Tables 4.2, 4.3 and 4.4 it can be observed that the performance of Random Forest overtakes the performance of SVM and MLP classifiers. In RF, the performance is the best when the number of features is 15, maximum depth is 80 and the number of trees in the Random Forest is 100. The output of the Random Forest is the likelihood of a visitor making a purchase.

### 4.3 Recommendation Module

Our hybrid recommendation module which combines collaborative filtering with ontology-based recommenders takes 4192ms to execute a query and receive the result, the list of similar products. The similarity between two items is observed to be 99.5% and the similarity between two users is observed to be 99.9%. The results are summarized in Table 4.5. We observe that the time taken to perform k-NN on products or items is greatly reduced by the creation of the top 200 database and product feature database prior to execution.

**Table 4.5** Results of the recommendation module

| Model | Query Latency | Item-based cosine similarity | User-based cosine similarity |
|---|---|---|---|
| Ontology-based hybrid model | 4192ms | 99.5% | 99.9% |

## 5. Conclusion

We construct a real-time user behavior analysis system for a virtual shopping environment which consists of two modules. We use the online shoppers intention data to perform the experiments. The system created, analyses the customer by determining their likelihood of abandoning the website and their purchasing intention. It predicts whether the customer is an at-risk customer or a potential customer. It then suggests the customer, a list of products based on the type he/she is. To a potential customer, the system suggests a list of products similar to the one they are viewing, with prices slightly higher than the viewing product. To an at-risk customer, the system suggests a list of products similar to the one they are viewing, with prices slightly lower than the viewing product.

We run oversampling and feature selection preprocessing techniques to enhance the performance and scalability of the algorithms. The best results are achieved with a Random Forest for the first module. In the second module, which uses the clickstream data, we train a long short-term memory-based recurrent neural network (LSTM-RNN) that generates an output showing the probability estimate of visitor's intention to leave the site without finalizing the transaction.

Our proposed approach combines the simplicity of finding users that have similar tastes/preferences with the active user, using a hybrid ontology-based model. In this approach, we combine an ontology-based recommender with a collaborative filtering approach. We build an ontology using the reviewer and product information from the AmazonReviews dataset. We define the four classes: Person, User, Neighbour, and Product, and the relationship between them in our ontology. For an active user, we update the profile in the ontology and retrieve the list of neighbours. We then apply the k-Nearest Neighbours algorithm. From the k-nearest neighbours, we obtain a list of products that they have bought and apply the k-Nearest Neighbours algorithm to get the list of products that we can recommend to the active user. This approach generates more knowledge about the Active User, the respective Neighbours and the Products and the relation between these three.

The proposed system does not address the cold-start problem of users. The system does not face a cold-start problem for products as new products can be easily added to the Ontology. For new users, the system will require inputs like name,age, interested products/categories etc.. from the user to suggest products. So the system can be further improved by taking care of the cold-start condition where the user is new to the site.

## References


1. Shyna K and Vishal M (2017). "A Study on Artificial Intelligence in ECommerce", International Journal of Advances in Engineering & Scientific Research, Volume 4,(Issue 4, Jun-2017), pp 62-. ISSN: 2349 –3607 (Online) , ISSN: 2349 –4824 (Print)
2. Mobasher, B., Dai, H., Luo, T. et al. (2002). Discovery and Evaluation of Aggregate Usage Profiles for Web Personalization. Data Mining and Knowledge Discovery 6, 61–82.
3. Suchacka G, Chodak G (2017) Using association rules to assess purchase probability in online stores. IseB 15(3):751–780
4. Suchacka G, Skolimowska-Kulig M, Potempa A (2015) Classification of e-customer sessions based on support vector machine. ECMS 15:594–600
5. Topal, Ibrahim. (2019). Estimation of Online Purchasing Intention Using Decision Tree - Karar Ağacı Kullanarak Çevrimiçi Satın Alma Niyetinin Tahmini. 17. 269-280. 10.11611/yead.542249.
6. Sakar, C. Okan & Polat, S. & Katircioglu, Mete & Kastro, Yomi. (2018). Real-time prediction of online shoppers' purchasing intention using multilayer perceptron and LSTM recurrent neural networks. Neural Computing and Applications. 10.1007/s00521-018-3523-0.
7. Tarus, J.K., Niu, Z., & Mustafa, G. (2017). Knowledge-based recommendation: a review of ontology-based recommender systems for e-learning. Artificial Intelligence Review, 50, 21-48.
8. George, G.; Lal, A.M. (2019) Review of ontology-based recommender systems in e-learning. Comput. Educ. 2019, 42, 103642.

9. Obeid, C.; Lahoud, I.; el Khoury, H.; Champin, P.-A. (2018). Ontology-Based Recommender System in Higher Education. In Proceedings of the Companion Proceedings of The Web Conference 2018, Lyon, France, 23–27 April 2018; pp. 1031–1034.
10. Guia, Márcio & Silva, Rodrigo & Bernardino, Jorge. (2019). A Hybrid Ontology-Based Recommendation System in e-Commerce. Algorithms. 12. 239. 10.3390/a12110239.