

TUTORIAL #7

PYTHON- PRODUCT RECOMMENDER

DR. ATEF BADER

**-PRESENTATION BY
SNEHAL PRAJAPATI**

Product Recommendation and types:

- A product recommendation is basically a filtering system that seeks to predict and show the items that a user would like to purchase.

Types:

a) Content based Filtering Technique

It focuses on the properties of the items. Similarity of items is determined by measuring similarities in their properties.

b) Collaborative Filtering Technique

It focuses on the relationship between users and items. Similarity of items is determined by the similarity of the ratings of those items by the users who have rated both the items.

Ref : Mining of Massive Datasets - Leskovec, Rajaraman and Ullman

Collaborative Filtering :

There are further 2 types of Collaborative Filtering :

a) **Memory-based CF**

This approach can be divided into two main sections:

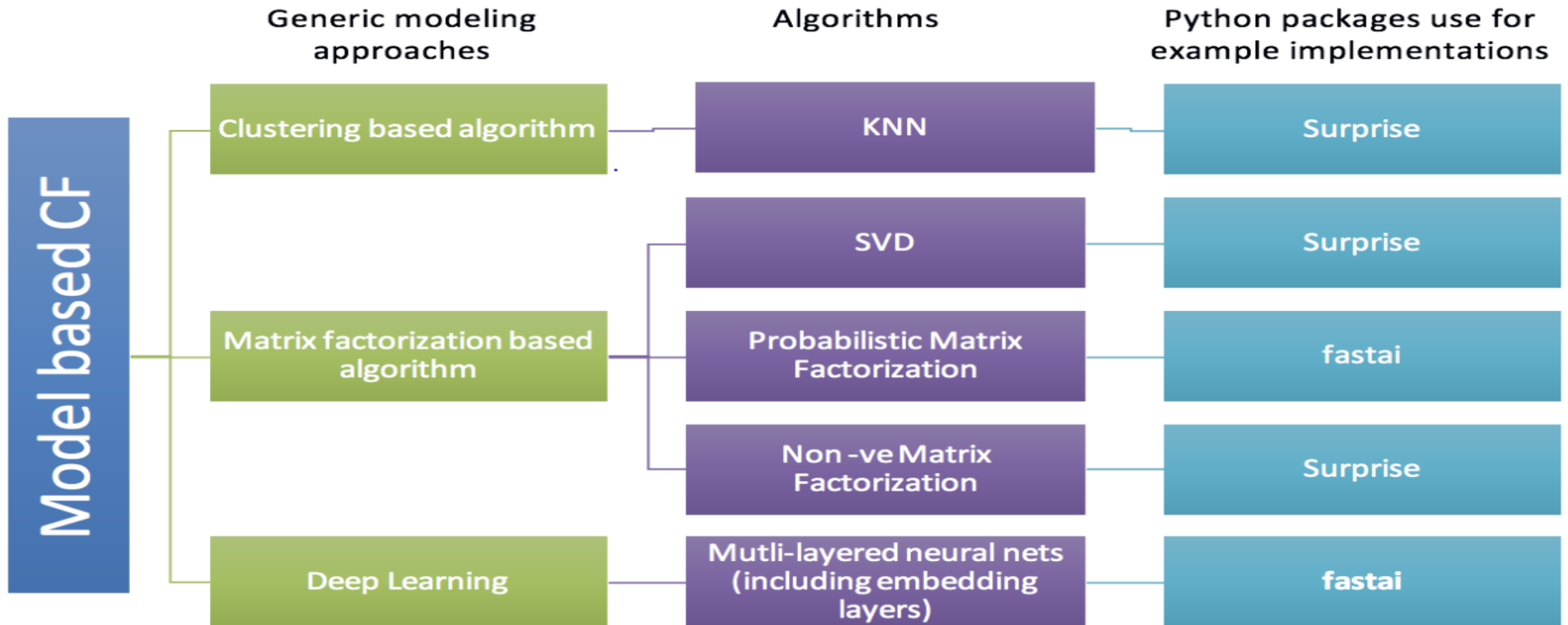
- A **user-item filtering** takes a particular user, find users that are similar to that user based on similarity of ratings, and recommend items that those similar users liked.
- In contrast, **item-item filtering** will take an item, find users who liked that item, and find other items that those users or similar users liked. It takes items and outputs other items as recommendations.

b) **Model-based CF**

In this approach, CF models are developed using machine learning algorithms to predict user's ratings of unrated items.

Ref: [https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0`](https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0)

Figure below shows the further classification of Model based CF.



Ref: <https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0>

Our Approach to Implementing Recommender Systems :

- For this Tutorial implementation , we have used **Matrix Factorization technique** for implementation which will be using **Singular Value Decomposition (SVD)** algorithm already implemented in the **Surprise** package.
- Python Version : **3.6**
- Anaconda Version : **4.2** (higher versions should be ok)

So, let's learn more about Matrix Factorization technique.

Matrix Factorization : The Basics

- Consider below matrix of users as rows and items as columns.
- What matrix factorization does is to come up with two smaller matrices, one representing users and one representing items, which when multiplied together will roughly produce matrix of ratings, ignoring the 0 entries.

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	0	3	0	3	0
User 2	4	0	0	2	0
User 3	0	0	3	0	0
User 4	3	0	4	0	3
User 5	4	3	0	4	0

A matrix of user/item ratings

Matrix Factorization : The Basics

- Let our matrix be $m \times n$, where m is the number of users and n is the number of items.
- So, we need $m \times d$ and a $d \times n$ matrix as our factors, where d is chosen to be small enough for the computation to be efficient and large enough to represent the number of dimensions along which interactions between users and items are likely to vary in some significant way.
- If we choose $d=2$, the predicted rating given by a user for a given item is then the dot product of the vector representing the user and the vector representing the item.
- We also need to add the bias for item i , user u and overall average rating for predicting the rating.

Matrix Factorization : The Basics

Theoretically, the predicted rating is given by below formula :

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u$$

Where, μ is the overall average rating, b_i is the bias for item i , b_u is the bias for user u , and $q_i^T p_u$ is the interaction between item i and user u .

Ref : <http://katbailey.github.io/post/matrix-factorization-with-tensorflow/>

Matrix-factorization : API docs

- The following class implements Matrix-factorization based prediction algorithms:

class `surprise.prediction_algorithms.matrix_factorization.SVD`

- The abstract class that defines the behavior of a prediction algorithm :

class `surprise.prediction_algorithms.algo_base.AlgoBase`

The famous SVD algorithm, as popularized by Simon Funk during the Netflix Prize uses it.

Ref:

https://surprise.readthedocs.io/en/stable/matrix_factorization.html#surprise.prediction_algorithms.matrix_factorization.SVD

Surprise : Basic Note

- Surprise is a Python Scikit building and analyzing package for recommender systems.
- It alleviates the pain of Dataset handling. Users can use both built-in datasets (Movielens, Jester), and their own custom datasets.
- It provides ready-to-use prediction algorithms such as baseline algorithms, SVD, SVD++, neighborhood methods and many more.

Surprise – Installation

- Open “Anaconda Prompt”
- Type the command : ***conda install -c conda-forge scikit-surprise***
- Type ‘y’ when prompted, it should install the required packages inside the “Anaconda3” folder.

```
Proceed ([y]/n)? y

Downloading and Extracting Packages
ca-certificates-2018 | 170 KB      | ##### | 100%
openssl-1.0.2p       | 5.4 MB      | ##### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done

(base) C:\Users\user>
```

- To know more about how conda install works :

<http://docs.anaconda.com/anaconda-cloud/user-guide/howto/#use-packages>

Jupyter notebook

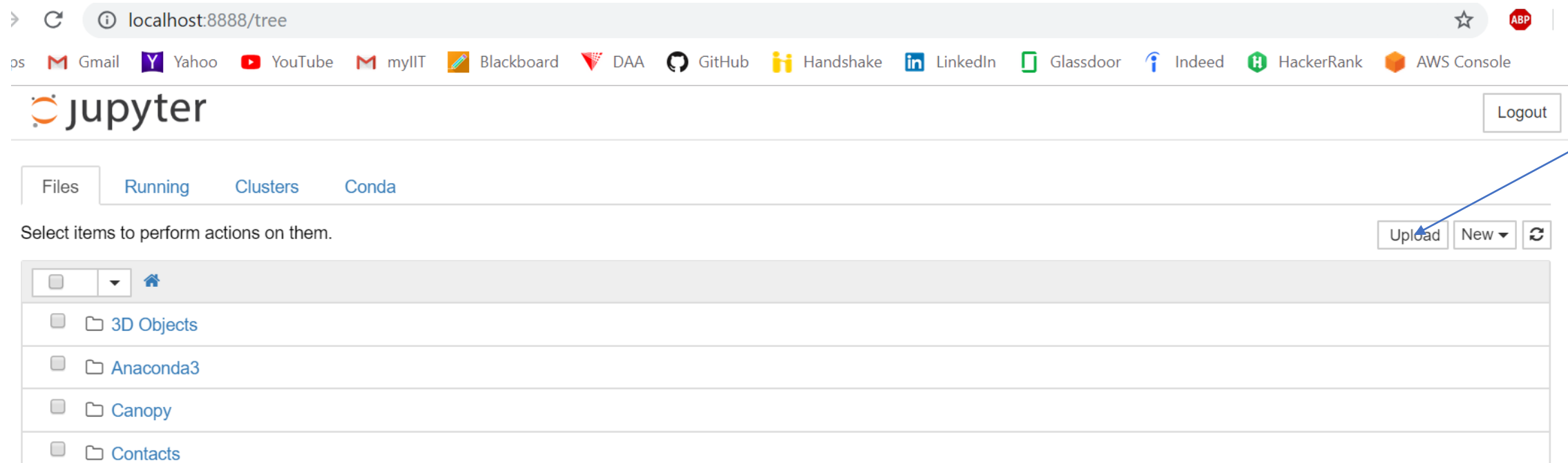
- Type “jupyter notebook” in anaconda prompt and it will open the notebook and you can “upload” any .ipynb file to the notebook

```
(base) C:\Users\user>jupyter notebook
[I 22:50:54.166 NotebookApp] [nb_conda_kernels] enabled, 3 kernels found
[I 22:51:07.866 NotebookApp] [nb_anacondacloud] enabled
[I 22:51:09.677 NotebookApp] \u2713 nbpresent HTML export ENABLED
[W 22:51:09.707 NotebookApp] \u2717 nbpresent PDF export DISABLED: No module named 'nbbrowserpdf'
[I 22:51:09.844 NotebookApp] [nb_conda] enabled
[I 22:51:10.557 NotebookApp] Serving notebooks from local directory: C:\Users\user
[I 22:51:10.557 NotebookApp] 0 active kernels
[I 22:51:10.572 NotebookApp] The Jupyter Notebook is running at: http://localhost:8888/
[I 22:51:10.576 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
```



Jupyter notebook

- Following is the screenshot of the jupyter notebook server home screen where you can upload any “.ipynb” file and Select “**Cell**” from Menu and “**Run All**” to run the notebook file.



Python code:

Link: <https://surprise.readthedocs.io/en/stable/FAQ.html>

Refer 'How to get the top-N recommendations for each user'

Code snippet which imports required packages :

```
In [7]: import os
import csv
from surprise import BaselineOnly
from surprise import Dataset
from surprise import Reader
from surprise import SVD
from surprise import accuracy
from surprise.model_selection import cross_validate
from surprise.model_selection import train_test_split
from collections import defaultdict
```

Python code:

Reviews are stored inside MongoDB. So we need to export the MongoDB data first using the command “mongoexport”.

```
In [2]: pr_file_path="C:/apache-tomcat-7.0.34/webapps/Tutorial_7/"
os.chdir('C:/Program Files/MongoDB/Server/3.2/bin')
os.system(r'mongoexport --db CustomerReviews --collection myReviews --type=csv --fields userName,productName,reviewRating >' + pr_f
```

We read this file and generate a test set of it.

```
with open(pr_file_path+"/mongodata_train.csv", "r") as f:
    reader = csv.DictReader(f, delimiter=',')
    with open(pr_file_path+"/mongodata_test.csv", "w", newline='') as f_out:
        writer = csv.DictWriter(f_out, fieldnames=reader.fieldnames, delimiter=",")
        for row in reader:
            writer.writerow(row)

file_path = os.path.expanduser(pr_file_path+'/mongodata_test.csv')

# As we're loading a custom dataset, we need to define a reader. In the
# movielens-100k dataset, each line has the following format:
# 'user item rating timestamp', separated by '\t' characters.
reader = Reader(line_format='user item rating', sep=',')

#data = Dataset.Load_from_file(file_path, reader=reader)
```

Python code:

We then find top – N recommendation for each user and save them in an ‘output.csv’ file.

```
def get_top_n(predictions, n=10):
    '''Return the top-N recommendation for each user from a set of predictions.

    Args:
        predictions(list of Prediction objects): The list of predictions, as
            returned by the test method of an algorithm.
        n(int): The number of recommendation to output for each user. Default
            is 10.

    Returns:
        A dict where keys are user (raw) ids and values are lists of tuples:
        ... [(raw item id, rating estimation), ...] of size n.
        ...

    # First map the predictions to each user.
    top_n = defaultdict(list)
    for uid, iid, true_r, est, _ in predictions:
        top_n[uid].append((iid, est))

    # Then sort the predictions for each user and retrieve the k highest ones.
    for uid, user_ratings in top_n.items():
        user_ratings.sort(key=lambda x: x[1], reverse=True)
        top_n[uid] = user_ratings[:n]

    return top_n
```


Python code:

```
# First train an SVD algorithm on the dataset.
data = Dataset.load_from_file(file_path, reader=reader)
trainset = data.build_full_trainset()
algo = SVD()
algo.fit(trainset)

# Than predict ratings for all pairs (u, i) that are NOT in the training set.
testset = trainset.build_anti_testset()
predictions = algo.test(testset)

top_n = get_top_n(predictions, n=2)

# Print the recommended items for each user
for uid, user_ratings in top_n.items():
    print(uid, [iid for (iid, _) in user_ratings])

out = open(pr_file_path+'matrixFactorizationBasedRecommendations.csv', 'w', newline='')
output=csv.writer(out)

for uid, user_ratings in top_n.items():
    output.writerow([uid, [iid for (iid, _) in user_ratings]])

out.close()
```

Python Code output :

Output is generated in 'matrixFactorizationBasedRecommendations.csv' file that consists of username and their top – n (n=2 in our case) recommendations based on the similar user ratings.

	A1			
	A	B	C	
1	csp586	['xbox360', 'ipad3']		
2	cs587	['ipad3', 'PS3']		
3	csp584	['PS3', 'Xbox']		
4	snehal	['ipad', 'ipad3']		
5				

Java Code:

ProductRecommenderUtility.java : method that reads output file

```
public HashMap<String,String> readOutputFile() {

    String TOMCAT_HOME = System.getProperty("catalina.home");
    BufferedReader br = null;
    String line = "";
    String cvsSplitBy = ",";
    HashMap<String,String> prodRecmMap = new HashMap<String,String>();
    try {

        br = new BufferedReader(new FileReader(new File(TOMCAT_HOME+"\\webapps\\Tutorial_7\\output.csv")));
        while ((line = br.readLine()) != null) {

            // use comma as separator
            String[] prod_recm = line.split(cvsSplitBy,2);
            prodRecmMap.put(prod_recm[0],prod_recm[1]);

        }

    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (br != null) {
            try {
                br.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    return prodRecmMap;
}
```

Java Code:

ProductRecommenderUtility.java : method that retrieves product details from the database

```
public static Product getProduct(String product){
    Product prodObj = new Product();
    try
    {
        String msg = getConnection();
        String selectProd="select * from Productdetails where Id=?";
        PreparedStatement pst = conn.prepareStatement(selectProd);
        pst.setString(1,product);
        ResultSet rs = pst.executeQuery();

        while(rs.next())
        {
            prodObj = new Product(rs.getString("Id"),rs.getString("productName"),rs.getDouble("productPrice"),rs.getString("productImage"),rs.getString("productManufacturer"),1);
        }
        rs.close();
        pst.close();
        conn.close();
    }
    catch(Exception e)
    {
    }
    return prodObj;
}
```

Java Code:

Display the products in the Carousel below Cart :

```
HashMap<String,String> prodRecmMap = new HashMap<String,String>();
prodRecmMap = prodRecUtility.readOutputFile();

int l =0;
for(String user: prodRecmMap.keySet())
{
    if(user.equals(utility.username()))
    {
        String products = prodRecmMap.get(user);
        products=products.replace("[", "");
        products=products.replace("]", "");
        products=products.replace("\\"", " ");
        ArrayList<String> productsList = new ArrayList<String>(Arrays.asList(products.split(",")));

        myCarousel = "myCarousel"+l;

        sb.append("<div id='content'><div class='post'><h2 class='title meta'>");
        sb.append("<a style='font-size: 24px;'>"+""+" Recommended Products</a>");

        sb.append("</h2>");

        sb.append("<div class='container'>");
        /* Carousels require the use of an id (in this case id="myCarousel") for carousel controls to function properly.
        The .slide class adds a CSS transition and animation effect, which makes the items slide when showing a new item.
        Omit this class if you do not want this effect.
        The data-ride="carousel" attribute tells Bootstrap to begin animating the carousel immediately when the page loads.
        */
        sb.append("<div class='carousel slide' id='"+myCarousel+"' data-ride='carousel'>");
```

Java Code:

Display the products in the Carousel below Cart :

```
int k = 0;
for(String prod : productsList){
    prod= prod.replace("'", "");
    Product prodObj = new Product();
    prodObj = ProductRecommenderUtility.getProduct(prod.trim());
    if (k==0 )
    {
        sb.append("<div class='item active'><div class='col-md-6' style = 'background-color: #58acfa;border :1px solid #cfd1d3'>");
    }
    else
    {
        sb.append("<div class='item'><div class='col-md-6' style = 'background-color: #58acfa ;border :1px solid #cfd1d3' >");
    }
    sb.append("<div id='shop_item'>");
    sb.append("<h3>"+prodObj.getName()+"</h3>");
    sb.append("<strong>"+prodObj.getPrice()+"$</strong><ul>");
    sb.append("<li id='item'><img src='images/'"+prodObj.getType()+"/"+prodObj.getImage()+"' alt='' /></li>");
    sb.append("<li><form method='post' action='Cart'>"+
        "<input type='hidden' name='name' value='"+prodObj.trim()+"'>"+
        "<input type='hidden' name='type' value='"+prodObj.getType()+"'>"+
        "<input type='hidden' name='maker' value='"+prodObj.getRetailer()+"'>"+
        "<input type='hidden' name='access' value='"+prodObj.getAccess()+"'>"+
        "<input type='submit' class='btnbuy' value='Buy Now'></form></li>");
    sb.append("<li><form method='post' action='WriteReview'>"+<input type='hidden' name='name' value='"+prodObj.getName()+"'>"+
        "<input type='hidden' name='type' value='"+prodObj.getType()+"'>"+
        "<input type='hidden' name='maker' value='"+prodObj.getRetailer()+"'>"+
        "<input type='hidden' name='access' value='"+prodObj.getAccess()+"'>"+
        "<input type='hidden' name='price' value='"+prodObj.getPrice()+"'>"+
        "<input type='submit' value='WriteReview' class='btnreview'></form></li>");
    sb.append("<li><form method='post' action='ViewReview'>"+<input type='hidden' name='name' value='"+prodObj.getName()+"'>"+
        "<input type='hidden' name='type' value='"+prodObj.getType()+"'>"+
        "<input type='hidden' name='maker' value='"+prodObj.getRetailer()+"'>"+
        "<input type='hidden' name='access' value='"+prodObj.getAccess()+"'>"+
        "<input type='submit' value='ViewReview' class='btnreview'></form></li>");

    sb.append("</ul></div></div>");
    sb.append("</div>");

    k++;
}
```

Output:


If I login as a user (e.x – Snehal) and add a product to the Cart, it shows me Recommended products in the carousel which can be added to the Cart or can be Reviewed as well.

[Games](#) [Tablets](#) [Trending](#) [ViewOrder](#) [Hello,Snehal](#) [Account](#) [Logout](#) [Cart\(1\)](#)


Cart(1)

1.	xbox360	: 399.99
	Total	399.99
		CheckOut


Recommended Products




Apple Ipad Pro
189.99\$



Apple Ipad 3
189.99\$





Correctness of the Example looking manually:

Let say, users have given following reviews for the products :

	snehal	csp584	cs587	csp586
Xbox	4		2	2
xbox360	5	5	3	
PS3	3			
PS4		2		5
ipad3		4		5
<u>ipad</u>			2	

Correctness of the Example looking manually:

- Looking at the data, clearly we can say that user “snehal” seems more likely to match with “csp584” and hence “snehal” will be recommended ‘PS4’ and ‘ipad3’ which csp584 has rated.
- Also , “cs587” will be recommended with ‘PS4’ and ‘ipad3’ which looks apparent.

	snehal	csp584	cs587	csp586
Xbox	4		2	2
xbox360	5	5	3	
PS3	3			
PS4		2		5
ipad3		4		5
<u>ipad</u>			2	

- As a concluding remark, a machine learning algorithm is expected to have accuracy of more than 50% and not always 100%.

Questions?