Sentiment Analysis and Product Recommendation on Amazon's Electronics Dataset Reviews

Machine Learning - Capstone Project 2

SENTIMENT ANALYSIS:

Machine Learning models take numerical values as input. The reviews are made of sentences, so in order to extract patterns from the data; we need to find a way to represent it in a way that machine learning algorithm can understand, i.e. as a list of numbers.

FEATURE EXTRACTION:

Feature engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work. Features are usually numeric in nature and can be absolute numeric values or categorical features that can be encoded as binary features for each category in the list using a process called one-hot encoding. The process of extracting and selecting features is both art and science, and this process is called feature extraction or feature engineering.

As a part of this project, Bag of Words model, TF-IDF, Hashing Vectorizer, Word2Vec and adding most common words into the stopwords list, SMOTE, PCA, and Truncated SVD techniques into classification models in the following sections as a part of feature engineering and selection.

DATA PREPROCESSING:

Due to computational considerations, features with good_rating_class_reviews longer than 150 words reviewed earlier 2010 were removed. Final dataset consisted 15000 observations. From the dataset, "clean text" and "rating class" were treated as "X"(feature) and "Y"(variable) respectively. Dataset were divided into 75% as training and 25% as testing.

MACHINE LEARNING:

In this project, the model needs to predict sentiment based on the reviews written by customers who bought headphones from Amazon. This is a supervised binary classification problem. Python's Scikit libraries was used to solve this problem. Following machine learning algorithms were implemented.

1. Logistic Regression

Logistic regression, despite its name, is a linear model for classification rather than regression. Logistic regression is also known in the literature as logit regression, maximum-entropy classification (MaxEnt) or the log-linear classifier. In this model, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function.

2. Naive Bayes

Naive Bayes implements the naive Bayes algorithm for multinomial distributed data, and is one of the two classic naive Bayes variants used in text classification (where the data are typically represented as word vector counts). This algorithm is a special case of the popular naïve Bayes algorithm, which is used specifically for prediction and classification tasks where we have more than two classes.

3. Random Forest Classifier

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True (default).

4. XGBoost Classifier

XGBoost means eXtreme Gradient Boosting. XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks. However, when it comes to small-to-medium structured/tabular data, decision tree based algorithms are considered best-in-class right now.

5. CatBoost Classifier

CatBoost is an algorithm for gradient boosting on decision trees. “CatBoost” name comes from two words “Category” and “Boosting”. the library works well with multiple Categories of data, such as audio, text, image including historical data.

EVALUATION METRICS:

Since we have a data imbalance in our case, the evaluation of the classifier performance must be carried out using adequate metrics in order to consider the class distribution and to pay more attention to the minority class. Based on this thought f1 score was used, which is harmonic average of precision and recall as my evaluation metric.

Understanding the types of errors our model makes are important. A good way to visualize that information is using a Confusion Matrix, which compares the predictions our model makes with the true label. With that in mind, confusion matrix was used besides our evaluation metric (f1 score).

MODELING:

Since the ratings of the reviews were not distributed normally, rating classes from 1-2 were classified as ‘Bad’ and Rating 3-4-5 were classified as 'Good'. For feature selection, threshold for word occurrence with using min_df/max_df, PCA and Singular Value Decomposition were

applied. For feature engineering, CountVectorizer, TF-IDF, Hashing Vectorizer and Word2Vec were applied to the text data in order to turn a collection of text documents into numerical feature vectors.

1. Bag of Words Model

The Bag of Words model is perhaps one of the simplest yet most powerful techniques to extract features from text documents. This specific strategy (tokenization, counting and normalization) is called the Bag of Words or "Bag of n-grams" representation. The essence of this model is to convert text documents into vectors such that each document is converted into a vector that represents the frequency of all the distinct words that are present in the document vector space for that specific document. Fig 1. shows that **Logistic Regression is the winner with 0.896267**

.

				precision	recall	f1-score	support
vectorizer	**model**	**accuracy**	**class**				
CountVect	**LogReg**	**0.896267**	**bad**	0.688581	0.833799	0.754264	716.0
			good	0.958724	0.911009	0.934257	3034.0
			average	0.907144	0.896267	0.899891	3750.0
	Random Forest	**0.857867**	**bad**	0.969231	0.263966	0.414929	716.0
			good	0.851758	0.998022	0.919108	3034.0
			average	0.874188	0.857867	0.822843	3750.0
	Naive Bayes	**0.898400**	**bad**	0.790295	0.636872	0.705336	716.0
			good	0.918059	0.960119	0.938618	3034.0
			average	0.893664	0.898400	0.894077	3750.0
	XGBoost	**0.890933**	**bad**	0.880893	0.495810	0.634495	716.0
			good	0.892142	0.984179	0.935903	3034.0
			average	0.889994	0.890933	0.878355	3750.0
	CatBoost	**0.896800**	**bad**	0.818182	0.590782	0.686131	716.0
			good	0.909372	0.969018	0.938248	3034.0
			average	0.891961	0.896800	0.890111	3750.0

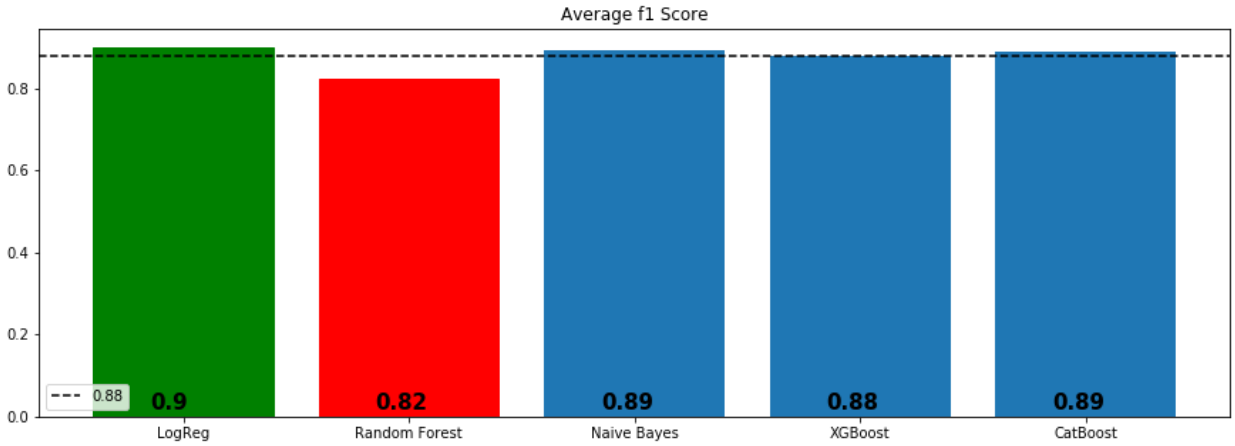


Fig 1. Average F1 score

2. TF-IDF Model

TF-IDF stands for Term Frequency-Inverse Document Frequency, a combination of two metrics: term frequency and inverse document frequency. In order to focus more on meaningful words, TF-IDF score (Term Frequency-Inverse Document Frequency) was used on top of our Bag of Words model. TF-IDF weighs words by how rare they are in our dataset, discounting words that are too frequent and just add to the noise. -IDF works by penalizing these common words by assigning them lower weights while giving importance to words which appear in a subset of a particular document. Fig 2. Shows that **CatBoosting is the winner with 0.896533 score.**

				precision	recall	f1-score	support
vectorizer	model	accuracy	class				
CountVect	LogReg	0.866933	bad	0.602070	0.893855	0.719505	716.0
			good	0.971716	0.860580	0.912777	3034.0
			average	0.901138	0.866933	0.875875	3750.0
	Random Forest	0.852533	bad	0.988024	0.230447	0.373726	716.0
			good	0.846218	0.999341	0.916427	3034.0
			average	0.873294	0.852533	0.812808	3750.0
	Naive Bayes	0.809600	bad	1.000000	0.002793	0.005571	716.0
			good	0.809498	1.000000	0.894721	3034.0
			average	0.845872	0.809600	0.724953	3750.0
	XGBoost	0.892267	bad	0.902062	0.488827	0.634058	716.0
			good	0.891136	0.987475	0.936836	3034.0
			average	0.893222	0.892267	0.879025	3750.0
	CatBoost	0.896533	bad	0.805970	0.603352	0.690096	716.0
			good	0.911637	0.965722	0.937900	3034.0
			average	0.891461	0.896533	0.890586	3750.0

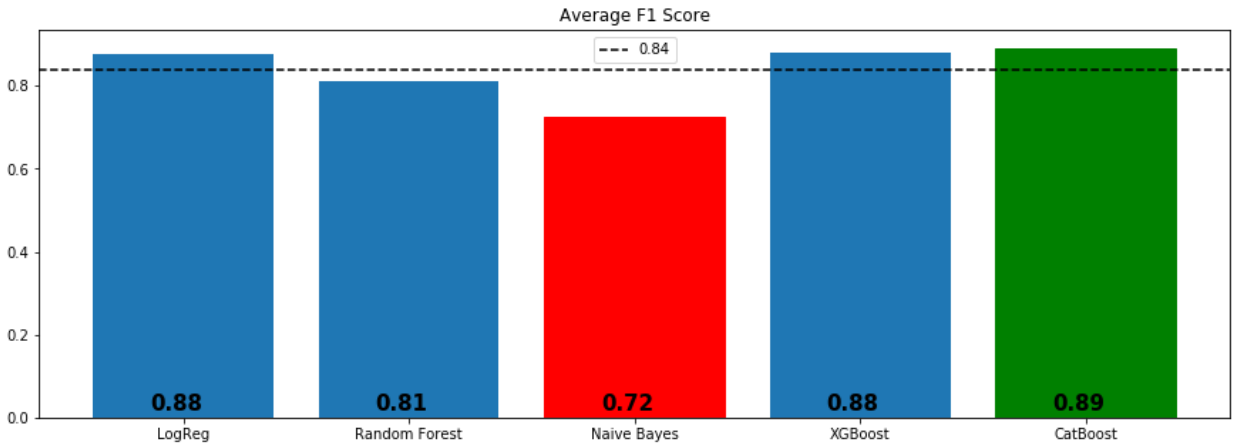


Fig 2. Average F1 score

3. Hash Vectorizer

Hash Vectorizer is designed to be as memory efficient as possible. Instead of storing the tokens as strings, the vectorizer applies the hashing trick to encode them as numerical indexes. The downside of this method is that once vectorized, the features' names can no longer be retrieved. Fig 3 shows that **CatBoosting is the winner with 0.894133 score.**

				precision	recall	f1-score	support
vectorizer	**model**	**accuracy**	**class**				
CountVect	**LogReg**	**0.842400**	**bad**	0.556256	0.863128	0.676519	716.0
			good	0.962865	0.837508	0.895822	3034.0
			average	0.885229	0.842400	0.853950	3750.0
	Random Forest	**0.870667**	**bad**	0.971429	0.332402	0.495317	716.0
			good	0.863623	0.997693	0.925830	3034.0
			average	0.884207	0.870667	0.843630	3750.0
	Naive Bayes	**0.816000**	**bad**	0.964286	0.037709	0.072581	716.0
			good	0.814884	0.999670	0.897869	3034.0
			average	0.843410	0.816000	0.740294	3750.0
	XGBoost	**0.889867**	**bad**	0.912807	0.467877	0.618652	716.0
			good	0.887378	0.989453	0.935640	3034.0
			average	0.892233	0.889867	0.875116	3750.0
	CatBoost	**0.894133**	**bad**	0.812133	0.579609	0.676447	716.0
			good	0.907070	0.968359	0.936713	3034.0
			average	0.888943	0.894133	0.887019	3750.0

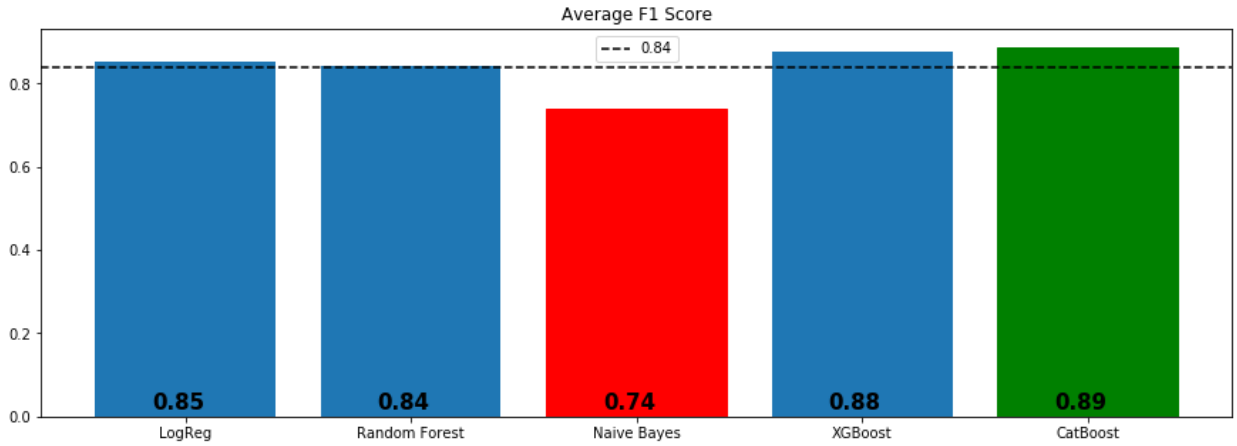


Fig 3. Average F1 score

4. Adding Most Common and Least Common Words to Stopwords List (Count Vectorizer)

Since there were not too many distinguisher words in different classes, the most and least common 70 words added to the stopwords list and models were applied in order to see any changes in evaluation metrics. Fig 4 shows that **CatBoosting is the winner with 0.890133 score**. Adding most and least common words to the stopword list didn't have impact on models' performance.

				precision	recall	f1-score	support
vectorizer	model	accuracy	class				
CountVect	LogReg	0.875200	bad	0.643519	0.776536	0.703797	716.0
			good	0.944560	0.898484	0.920946	3034.0
			average	0.887081	0.875200	0.879485	3750.0
	Random Forest	0.889333	bad	0.841270	0.518156	0.641314	716.0
			good	0.895739	0.976928	0.934574	3034.0
			average	0.885339	0.889333	0.878580	3750.0
	Naive Bayes	0.881600	bad	0.675258	0.731844	0.702413	716.0
			good	0.935440	0.916941	0.926099	3034.0
			average	0.885763	0.881600	0.883389	3750.0
	XGBoost	0.877600	bad	0.890578	0.409218	0.560766	716.0
			good	0.876352	0.988134	0.928892	3034.0
			average	0.879068	0.877600	0.858605	3750.0
	CatBoost	0.890133	bad	0.819328	0.544693	0.654362	716.0
			good	0.900428	0.971655	0.934686	3034.0
			average	0.884943	0.890133	0.881163	3750.0

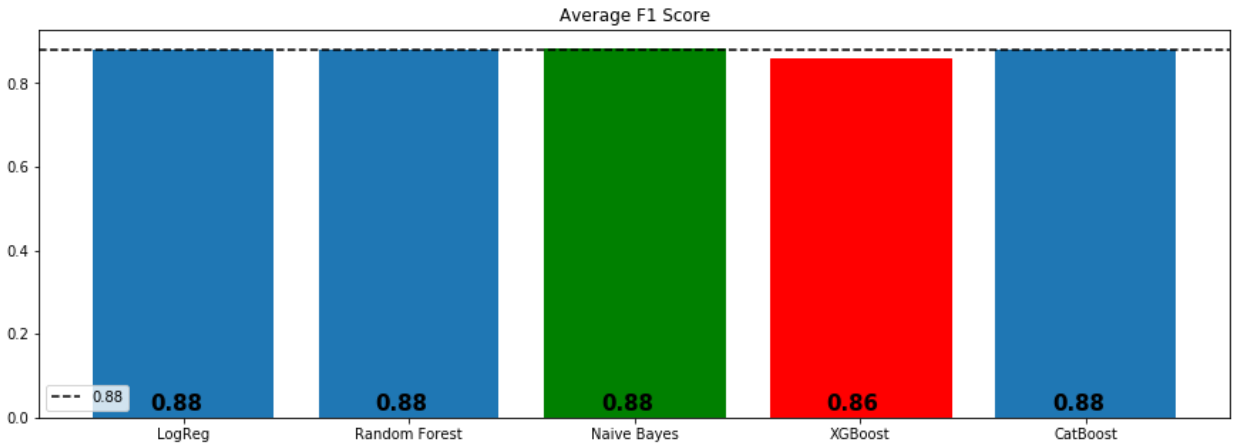
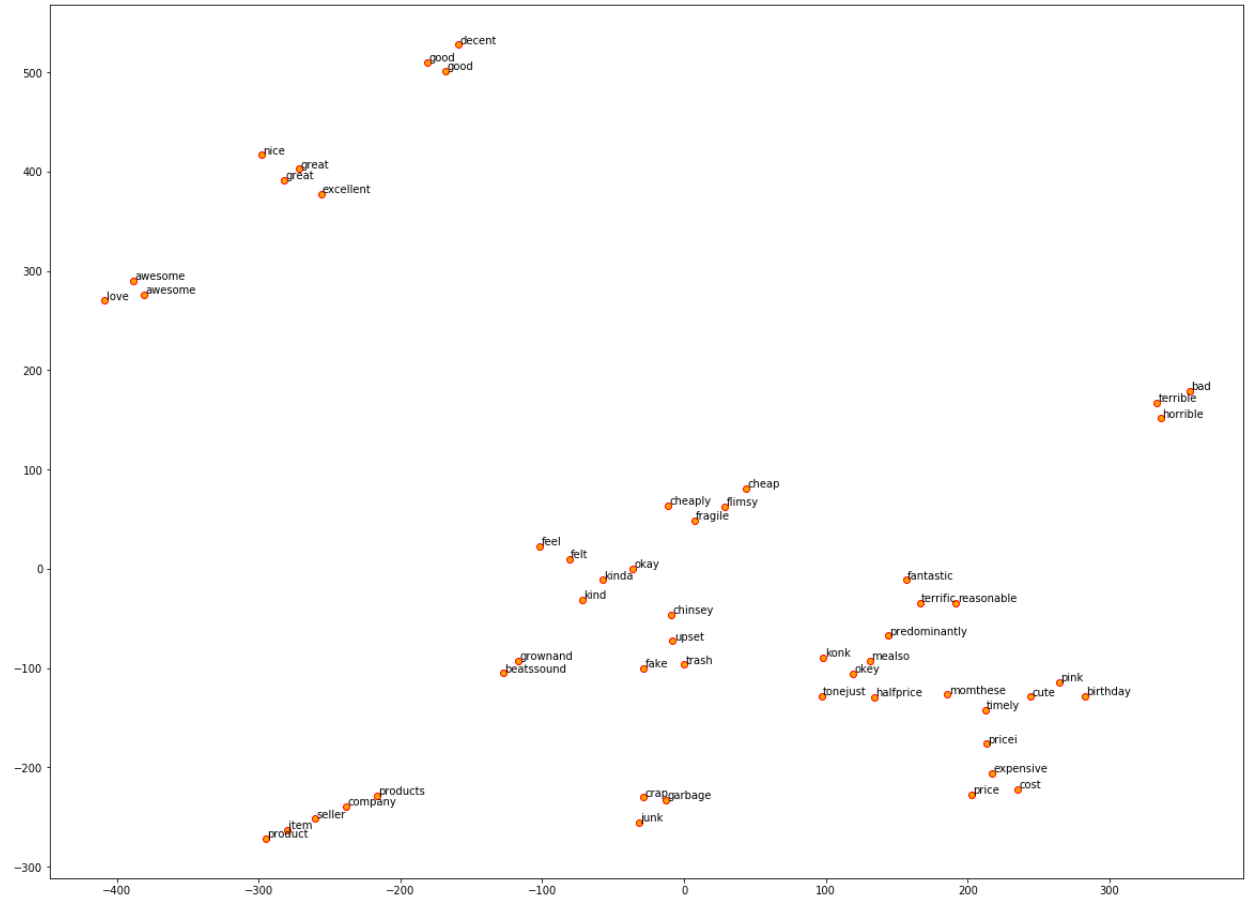


Fig 4. Average F1 score

5. Applying Word2Vec and Simple Neural Network

We created word vectors using Word2Vec and the model has 26548 unique words where each word has a vector length of 100. Then we used these dense vectors - word embeddings - in a simple neural network to predict. In training and validation accuracy graph, the model starts to overfit after 1st epoch. The accuracy for this simple neural network is 0.7992.

```
{'feel': ['felt', 'grownand', 'beatssound', 'kind', 'kinda'],
 'good': ['decent', 'great', 'predominantly', 'reasonable', 'nice'],
 'product': ['item', 'products', 'company', 'timely', 'seller'],
 'cheap': ['flimsy', 'chinsey', 'cheaply', 'fragile', 'mealso'],
 'junk': ['crap', 'garbage', 'upset', 'fake', 'trash'],
 'bad': ['terrible', 'horrible', 'okay', 'konk', 'okey'],
 'great': ['fantastic', 'excellent', 'good', 'awesome', 'terrific'],
 'price': ['cost', 'tonejust', 'expensive', 'pricei', 'halfprice'],
 'love': ['awesome', 'pink', 'cute', 'birthday', 'momthese']}
```



Visualization of the words of interest and their similar words using their embedding vectors after reducing their dimensions to a 2-D space with t-SNE is presented above. Similar words based on gensim's model can be viewed as well.

PRODUCT RECOMMENDATION:

Till recently, people generally tended to buy products recommended to them by their friends or the people they trust. This used to be the primary method of purchase when there was any doubt about the product. But with the advent of the digital age, that circle has expanded to include online sites that utilize some sort of recommendation engine.

A recommendation engine filters the data using different algorithms and recommends the most relevant items to users. It first captures the past behavior of a customer and based on that, recommends products which the users might be likely to buy.

If we can recommend a few items to a customer based on their needs and interests, it will create a positive impact on the user experience and lead to frequent visits. Hence, businesses nowadays are building smart and intelligent recommendation engines by studying the past behavior of their users. In this project, item-item collaborative filtering was used.

DATA PROCESSING:

After merging electronics rating dataset with product metadata, null values were removed from the dataset. Total features were 7530925. The final dataset is shown below.

	userID	prod_ID	rating	prod_name
0	AKM1MP6P0OYPR	0132793040	5.0	Kelby Training DVD: Mastering Blend Modes in A...
1	A2CX7LUOHB2NDG	0321732944	5.0	Kelby Training DVD: Adobe Photoshop CS5 Crash ...
2	A2NWSAGRHCP8N5	0439886341	1.0	Digital Organizer and Messenger
3	A2WNBOD3WNDNKT	0439886341	3.0	Digital Organizer and Messenger
4	A1GI0U4ZRJA8WN	0439886341	1.0	Digital Organizer and Messenger

Fig 1. Final Dataset for product recommendation

ITEM-ITEM COLLABORATIVE FILTERING:

This collaborative filtering is useful when the number of users is more than the items being recommended. In this project, the number of users (4053964) is more than the number of items (469625).

In this filtering, the similarity between each item pair was computed and based on that, similar items were recommended which are liked by the users in the past. The weighted sum of ratings of "item-users" were taken. The tem based filtering process is shown in Fig 2.

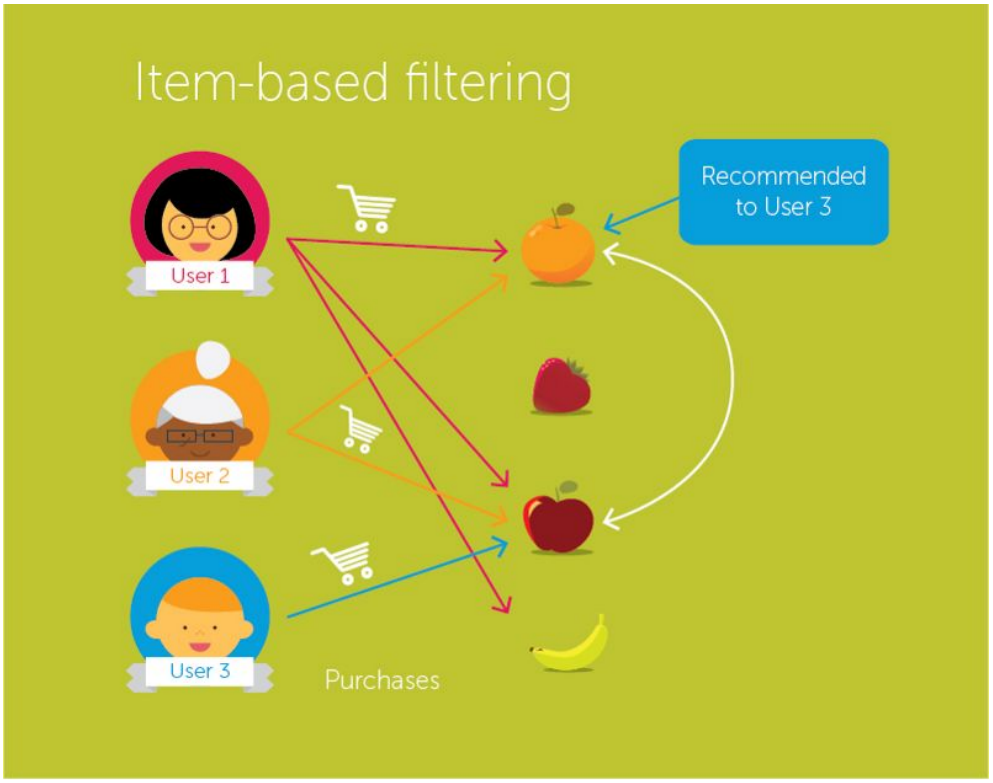


Fig 2. Item based filtering recommendation

1. **Top 10 Popular Products by Sum user Ratings**

	prod_ID	prod_name	ratings_sum
3313486	B003ES5ZUU	AmazonBasics High-Speed HDMI Cable - 15 Feet (...	846.0
6104981	B0088CJT4U	TP-LINK TL-WDR4300 Wireless N750 Dual Band Rou...	814.0
1198226	B000N99BBC	TP-LINK TL-SG1005D 10/100/1000Mbps 5-Port Giga...	755.0
5941380	B007WTAJTO	SanDisk Ultra 64GB MicroSDXC Class 10 UHS Memo...	741.0
6031403	B00829TIEK	Seagate Backup Plus 3TB USB 3.0 Desktop Extern...	626.0
6028195	B00829THK0	Seagate Backup Plus 1TB Desktop External Hard ...	560.0
6190152	B008DWCRQW	D-Link Wireless AC 1750 Mbps Home Cloud App-En...	524.0
4024382	B004CLYEDC	Micra Digital CAT5e Snagless Patch Cable, 5 Fe...	517.0
2796338	B002R5AM7C	Flip MinoHD Video Camera - Brushed Metal, 8 GB...	514.0
2883526	B002V88HFE	eneloop SEC-CSPACER4PK C Size Spacers for use ...	475.0

Fig 3. Top 10 popular products by sum user ratings

2. **Product Recommendation**

```
User A100WO06OQR8BQ has already purchased 91 items.
Recommending the highest 5 predicted  items not already purchased.
```

	prod_ID	prod_name
11745	B004CLYEDC	Micra Digital CAT5e Snagless Patch Cable, 5 Fe...
5649	B000N99BBC	TP-LINK TL-SG1005D 10/100/1000Mbps 5-Port Giga...
5012	B004CLYEFK	Micra Digital USB A to USB B Cable (6 Feet)
1169	B00829THK0	Seagate Backup Plus 1TB Desktop External Hard ...
656	B00834SJSK	Seagate Expansion 500GB Portable External Hard...

Fig 4. Product recommendation

Code:

Sentiment Analysis:

https://github.com/umaraju18/Capstone_project_2/blob/master/code/Amazon_headphones_Sentiment_Analysis_CV_IF_IDF_HASH.ipynb

https://github.com/umaraju18/Capstone_project_2/blob/master/code/Amazon_headphones_wordvec.ipynb

Recommendation System:

https://github.com/umaraju18/Capstone_project_2/blob/master/code/amazon_electronics_recommendation.ipynb