Recommendation System Manish Garg

October 5th, 2019

# Definition

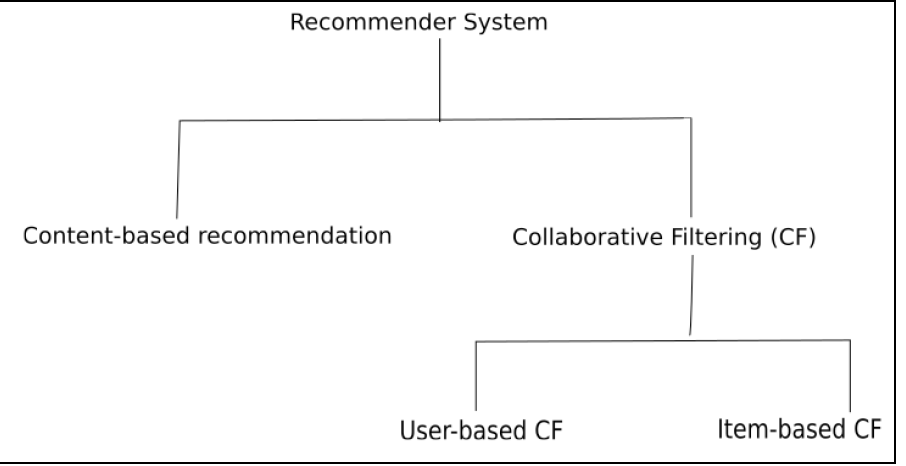
**Project Overview**

Recommendation systems are information filtering systems that deal with the problem of digital data overload to pull out items or information according to the user's preferences, interests, and behavior, as inferred from previous activities.

In machine learning one of the most common use case is recommendation to users that they'll be interested in. For example, the way Amazon recommends books based on the book that you are currently Browsing or Netflix recommending movies for you.

A recommendation system is one that learns about what items might be of interest to a user, and then recommends those items for buying, renting, listening, watching, and so on. Recommendation systems are broadly classified into two categories:

* content-based filtering
* collaborative filtering.



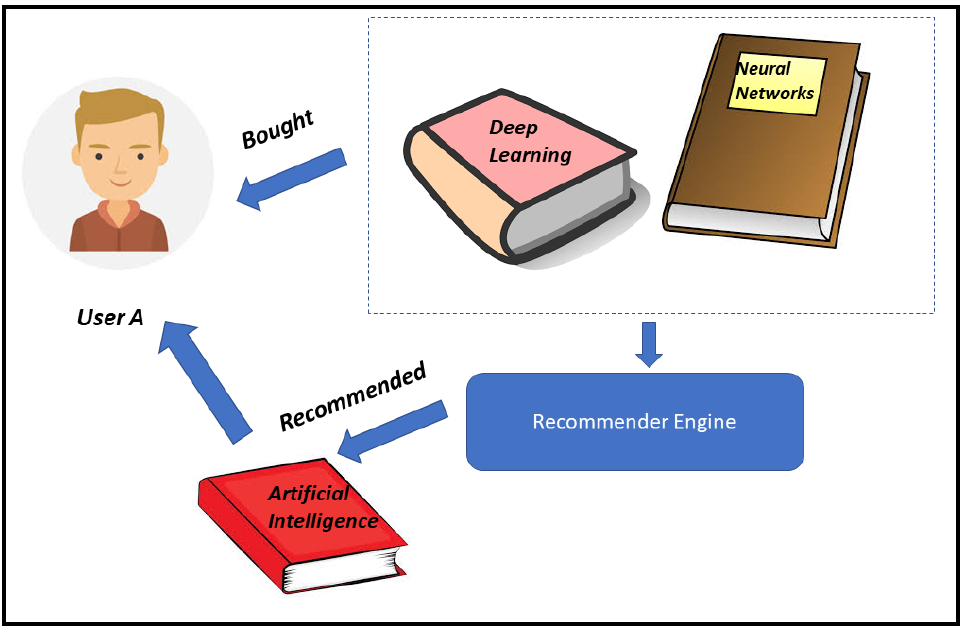
**Content-based filtering**

Content-based filtering is based on creating a detailed model of the content from which recommendations are made, such as the text of books, attributes of movies, or information

about music.

The content model is generally represented as a vector space model. Some of the common models for transforming content into vector space models are TFIDF, the Bagof-words model, Word2Vec, GloVe, and Item2Vec.

Along with the content model, a user profile is also created using information about the user. Content is recommended based on matching the user profile with the content model.



User **A** has bought books named **Deep Learning** and **Neural Networks**. Since the content of the book **Artificial Intelligence** is similar to the two books, the content-based recommender system has recommended the book **Artificial Intelligence to User A.** As we can see, in content-based filtering, the user is recommended items based on their preferences. This doesn't involve how other users have rated the book.

**Advantages of content-based filtering algorithms**

The following are the advantages of content-based filtering algorithms:

* Eliminates the cold-start problem for new items: If we have enough information about the users, and detailed information about the new content, then the cold start problem found in collaborative filtering algorithms does not affect content-based algorithms. The recommendation can be made based on the user profile and the information about the content.
* The recommendations are explainable and transparent: Using content representation models, we'll be able to explain how certain items are selected for recommendations.

**Disadvantages of content-based filtering algorithms**

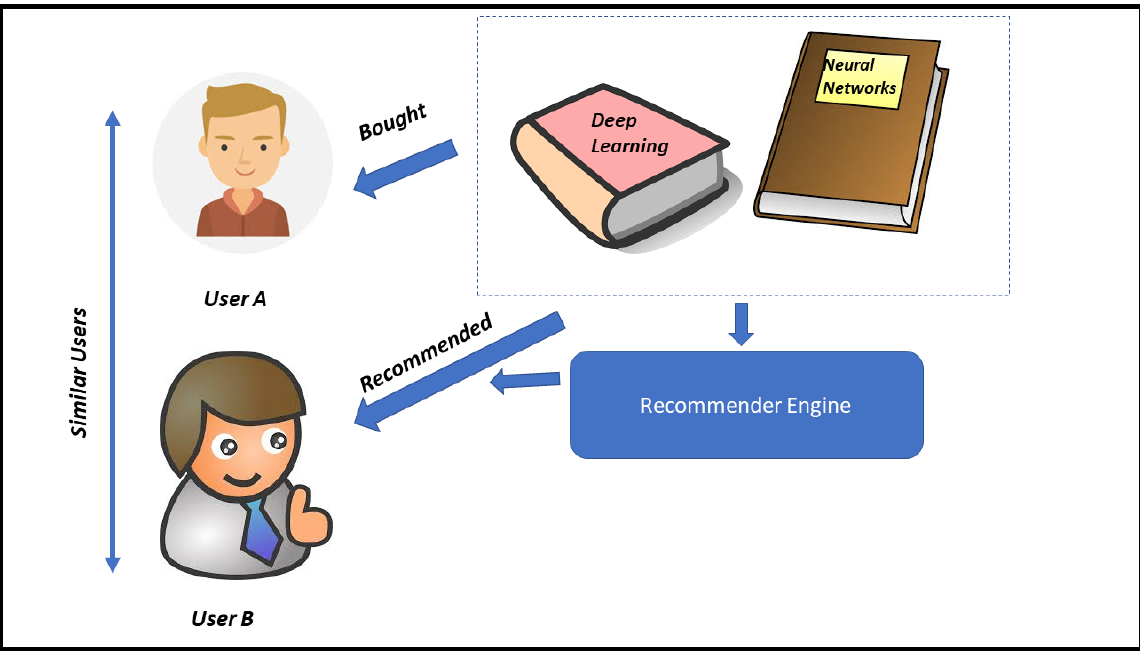
The following are the disadvantages of content-based filtering algorithms:

* Content-based filtering algorithms require detailed information about items and content, which is sometimes not available
* Content-based filtering algorithms are prone to overspecialization

**Collaborative filtering**

Collaborative filtering algorithms do not need detailed information about the user or the items. They build models based on user interactions with items such as song listened, item viewed, link clicked, item purchased or video watched.

It tries to identify similar users pertaining to a given user, and then recommends the user items that similar users have liked, bought, or rated highly. This is generally called **user-user collaborative** filtering. The opposite is to find items similar to a given item and recommend items to users who have also liked, bought, or rated other similar items highly. This goes by the name **item-item collaborative** filtering:



**User** **A** and **User** **B** are very similar in terms of their taste in buying books. User A has recently bought the books **Deep Learning** and **Neural Networks**. Since **User** **B** is very similar to **User** **A,** the user-user collaborative recommender system recommends these books to **User B** as well.

**Problem Statement**

A domain that mainly uses the recommendation system is e-commerce. So, in our basic version of the recommendation engine specifically, we will be building an algorithm that can suggest the name of the products based on the category of the product.

Our recommendation engine, will **suggest books** in the same way as the Amazon website.

We will be building three versions of the recommendation algorithm. The baseline approach is simple but intuitive so that readers can learn what exactly the recommendation algorithm is capable of doing. Baseline is easy to implement. In the second and third approach, we will be building the book recommendation engine using ML algorithms.

We will be using both **content-based** recommendation and **collaborative filtering**.

Metrics

The performance of each classification model is evaluated using statistical measures;

classification accuracy, recall, sensitivity and specificity. It is using true positive (TP), true negative (TN), false positive (FP) and false negative (FN).

We will also summaries the performance of a classification algorithm using confusion

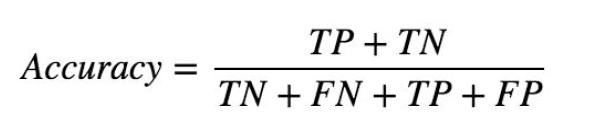
metrics as calculating a confusion matrix can give you a better idea of what your

classification model is getting right and what types of errors it is making.

|  |  |  |
| --- | --- | --- |
|  | **Predicted - Positive** | **Predicted - Negative** |
| **Actual - Positive** | True Positive | False Negative |
| **Actual - Negative** | False Positive | True Negative |

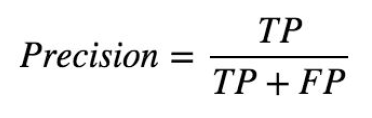
Classification accuracy is defined as the ratio of the number of correctly classified cases and is equal to the

sum of TP and TN divided by the total number of cases (TN + FN + TP + FP).



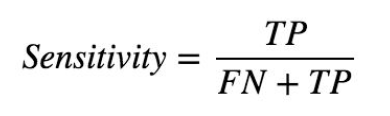
Precision is defined as the number of true positives (TP) over the number of true positives plus the number of

false positives (FP).



Sensitivity refers to the rate of correctly classified positive and is equal to TP divided by the sum of TP and FN.

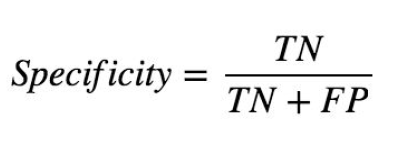
Sensitivity may be referred as a True Positive Rate.



Specificity refers to the rate of correctly classified negative and is equal to the ratio of TN to the sum of TN and

FP

<https://www.sciencedirect.com/science/article/pii/S1877050918308226>



# **Data Description**

In this project we are using two datasets, as follows:

• E-commerce item data

• Book-Crossing dataset

# **2.1 e-commerce Item Data**

This dataset contains data items taken from actual stock keeping units (SKUs).

It is from an outdoor apparel brand's product catalog. We are building the recommendation engine for this outdoor apparel brand's product catalog. You can access the dataset by using this link:

<https://www.kaggle.com/cclark/productitem-data/data>

This dataset contains 500 data items. There are two columns in the dataset.

• ID: This column indicates the indexing of the data item. In layman's terms, it is the serial number of the dataset.

• Description: This column has all the necessary descriptions about the products, and we need to use this data to build the recommendation engine.

The description column has textual data, and we need to process this textual dataset in order to build the recommendation engine

# **2.2 The Book-Crossing dataset**

The Book-Crossing dataset is widely used to build recommendation systems. You

can access it at - <http://www2.informatik.uni-freiburg.de/~cziegler/BX/>

This dataset is available in two formats, as follows:

• SQL dump

• CSV dump

We are using the CSV dump of the dataset. Both formats have three tables with different data attributes. The names of these three files are as follows:

• BX-Book-Ratings.csv

• BX-Books.csv

• BX-Users.csv

**BX-Books.csv**

This file contains all the details regarding the books. The table contains the following

data attributes:

• ISBN: The ISBN is provided to identify the book. All invalid ISBNs have already been deleted. This data table contains only valid ISBNs.

• Book-Title: This data attribute contains the name of the book.

• Book-Author: This data attribute contains the name of the author of the book.

• Year-Of-Publication: This indicates the year of publication of the book and is in the YYYY format.

• Publisher: This data column has the name of the publisher who has published the book.

• Image-URL-S: This data attribute has the URL for the image of the book's

cover page. S indicates a small size of cover page image.

• Image-URL-M: This data attribute has the URL for the image of the book's cover page. M indicates a medium size of cover image.

• Image-URL-L: This data attribute has the URL for the image of the book's cover page. L indicates a large size of cover image.

**BX-Book-Ratings.csv**

This CSV file contains data related to the rating of the book. This table contains three

data attributes, which are as follows:

• User-ID: This data attribute indicates the unique user ID. This column has a numeric value. The length of the user ID is six.

• ISBN: The full form of ISBN is International Standard Book Number. This data attribute indicates the unique identification number of the book.

• Book rating: This data attribute indicates the user rating for the book. The rating of the book varies from 0 to 10. 0, with 0 indicating less appreciation and 10.0 indicating the highest appreciation.

**BX-Users.csv**

This is the third data table of the Book-Crossing dataset. This file contains information about the users.

This particular data file contains the following data attributes:

• User-ID: This data column indicates the user ID, which is a six-digit integer number.

• Location: This data is the part of the demographic details regarding the user. The location indicates the name and abbreviation of the city. The location details for all users are not available, so you will find the null value for those users whose locations haven't been found.

• Age: This is also a demographic data point. If the user's age is tracked, then it is present in the dataset; if not, then the value of the age is null.

# **Approach and Techniques**

As the retrieval and rule based chatbot do not work well for the unseen queries and we need to hard code each and every scenario which is not feasible in short duration. Also, it cannot understand the long context so we will not be using this technique.

We will use **generative-based** approaches for our solution. Both the approaches will use **Deep Learning** techniques.

## **Building the baseline approach**

In order to develop the baseline approach, we will be using the **content-based** approach. It is based on the assumption that if a person has bought one type of item, then he will probably like a similar product(s) as well. For example, if a person is buying a pair of jeans, then there is a high chance that he will also like to buy t-shirts or tops, as well as formal trousers or other types of trousers.

Basically, the recommendation for the products is based on the content that you have explored, bought, or are interested in. This approach works well when the context and properties of each of the item can be determined easily. So, when we need to build a system that can recommend items or products that

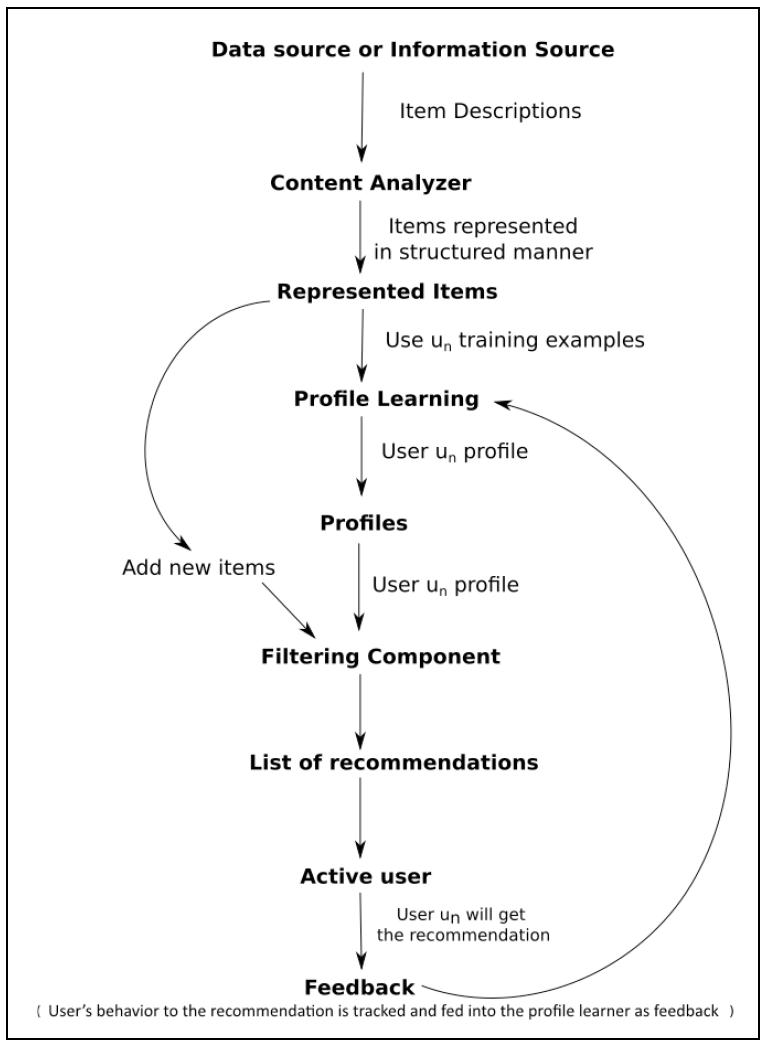
are similar to the user's buying pattern or browsing pattern, we use this approach.

The reason for choosing this approach is that this type of recommendation is not influenced by choices of other users. This will provide a personalized experience for users. A recommendation is totally based on the items and its features that users like. This approach helps the e-commerce company increase their sales with less effort. It needs less manual work, which is a good point to note here.

## **3.1.1 Implementation**

1. **Architecture of the recommendation system**

The basic architecture for the content-based recommendation system will as shown below:



<https://github.com/groveco/content-engine>

These are the steps that we need to follow:

1. Loading the dataset
2. Generating the feature using TF-IDF the cosine similarity matrix
3. Generating the prediction

We will print the value of cos θ as our scoring values for each item to its related items. If the score is close to one, then it can be said that these items are more similar and there is a higher chance that the user will like the recommendation. If the score is closer to 0 or –1, then items appear less attractive to the users. So just note that here, the score indicates the value of cos θ and not the angle directly.

1. **Testing matrix**

Here, the cosine similarity score is the biggest testing score for us. That is because with the help of that score, we can easily come to learn whether the algorithm can suggest the items whose cosine similarity score is close to 1 close to 0.

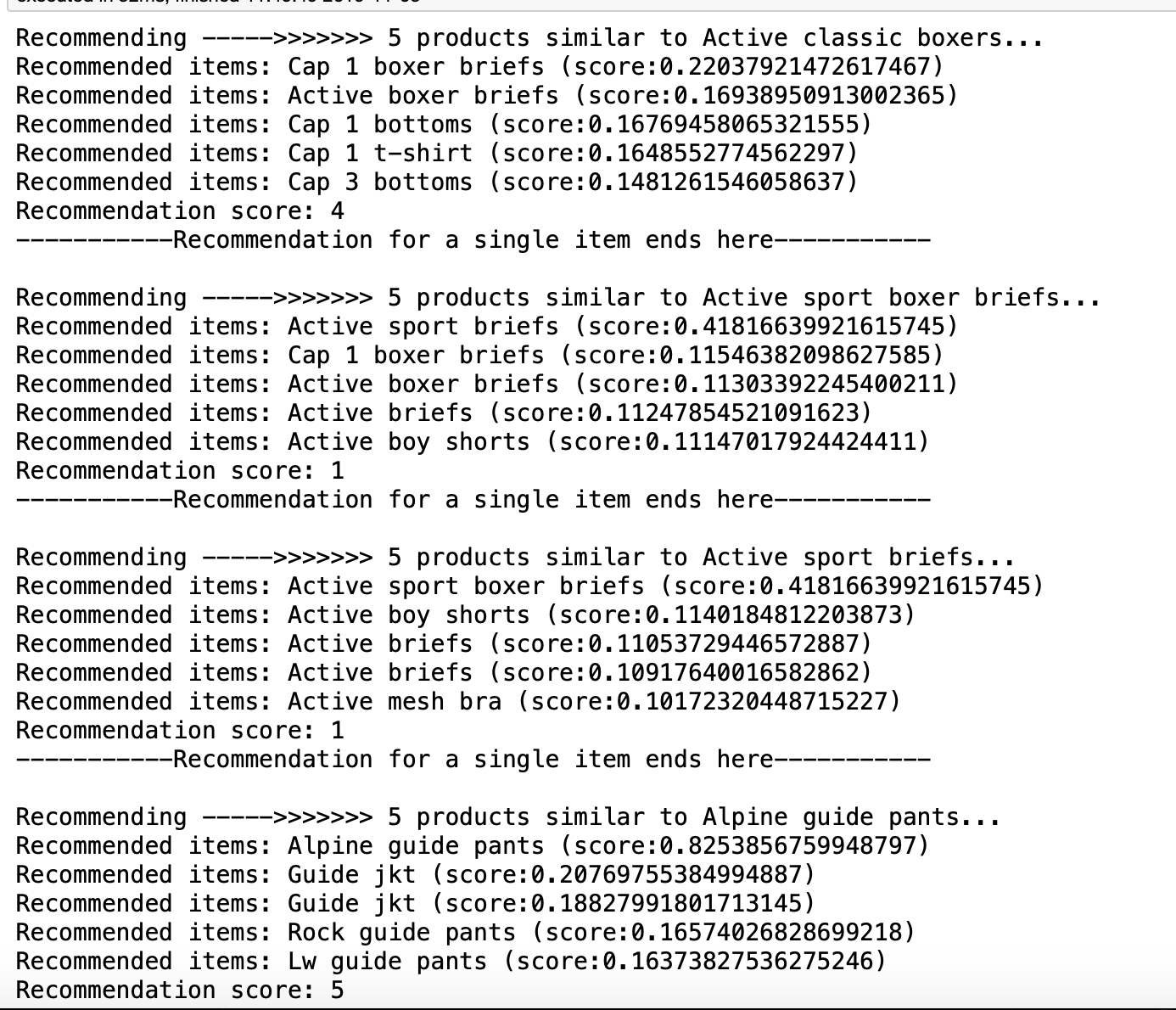
For some items, we will obtain a score that is close to 1, and for other items, we obtain a score that is close to 0. So, we need to focus on this cosine score in order to get an idea of how well or badly the recommendation engine is doing.

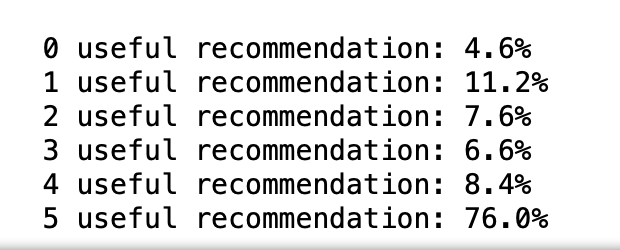
* *0 – 1 : Items are more likely to be similar*
* *0 : Items are orthogonal and not similar*
* *0 - -1 : Items are opposite and are not similar*

We will count the number of items with more than a certain score, which means that we can decide the threshold value for the cosine similarity score and count how many items the recommendation engine is suggesting above that threshold value.

As of initial, we decide a **cut-off score of 0.15**. In this case, **all items whose cosine score is above 0.15 are considered a good recommendation**. Here, the trick is that you need to experiment with this threshold value because based on the user's activity, you may change it later on. This **parameter will be a tunable** parameter for us.

**Result of items**





As you can see in the preceding figure, this approach gives us useful recommendations 76.0% of the time, and it provides four useful suggestions 8.4% of the time. After looking at the analysis of the result, we can say that the baseline approach is doing well, and we can definitely improve the results with the help of

the other approach.

## End-to-end memory networks

# Benchmark Model

The given dataset is a supervised classification learning problem for which we first tried with simple K-nearest neighbours to get base value of score.

KNN is an algorithm that's used in pattern recognition for object classification based on the characteristics of the nearest objects. An object is classified according to the majority of the votes of its neighbouring k cluster. **KNeighborsClassifier** gave model accuracy for the data as 0.5934.

# **Implementation**

## **Sequence-to-sequence model**

In this implementation, we don't need to generate features because the seq2seq model generates its internal representation for sequences of words given in a sentence. Our implementation part has the following steps:

• Data preparation

• Implementing the seq2seq model

During this implementation, we will be using the Cornell movie-dialogs dataset. First of all, we need to prepare data in a format that we can use for training.

## **5.1.2 Data preparation**

In this step we will be doing following:

• Generating question-answer pairs

• Preprocessing the dataset

• Splitting the dataset into the training dataset and the testing dataset

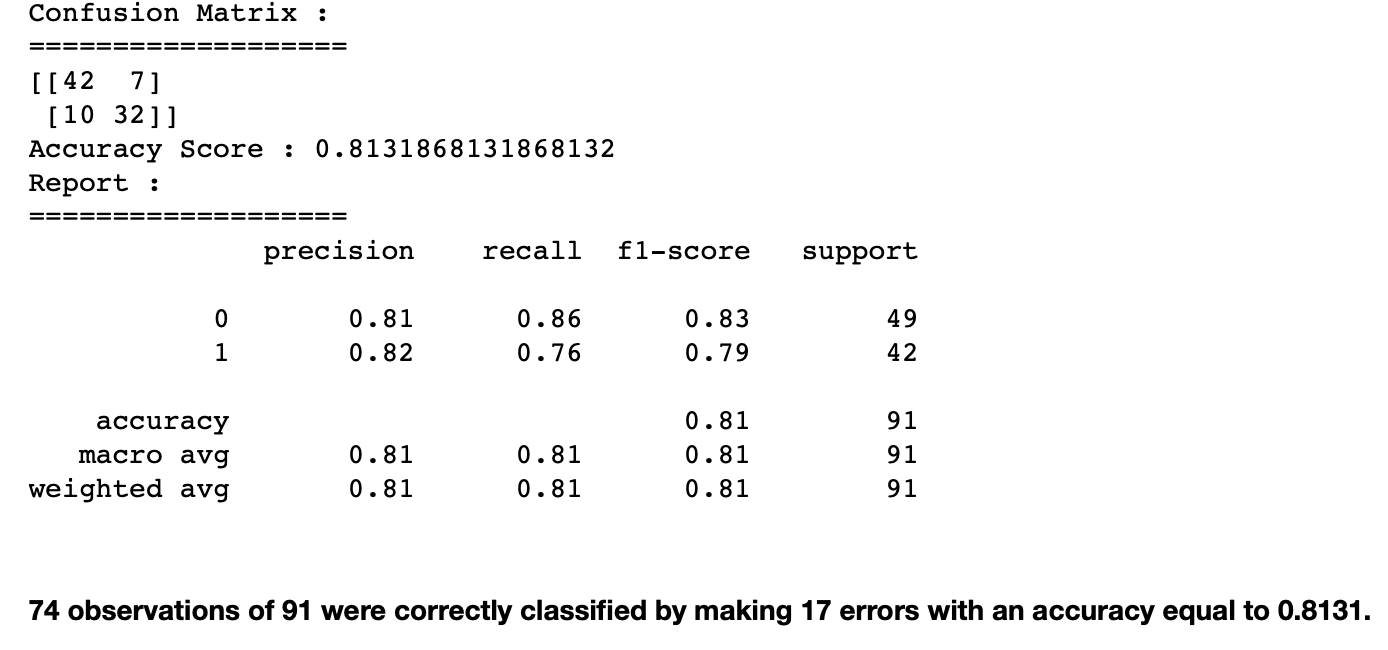
• Building a vocabulary for the training and testing datasets

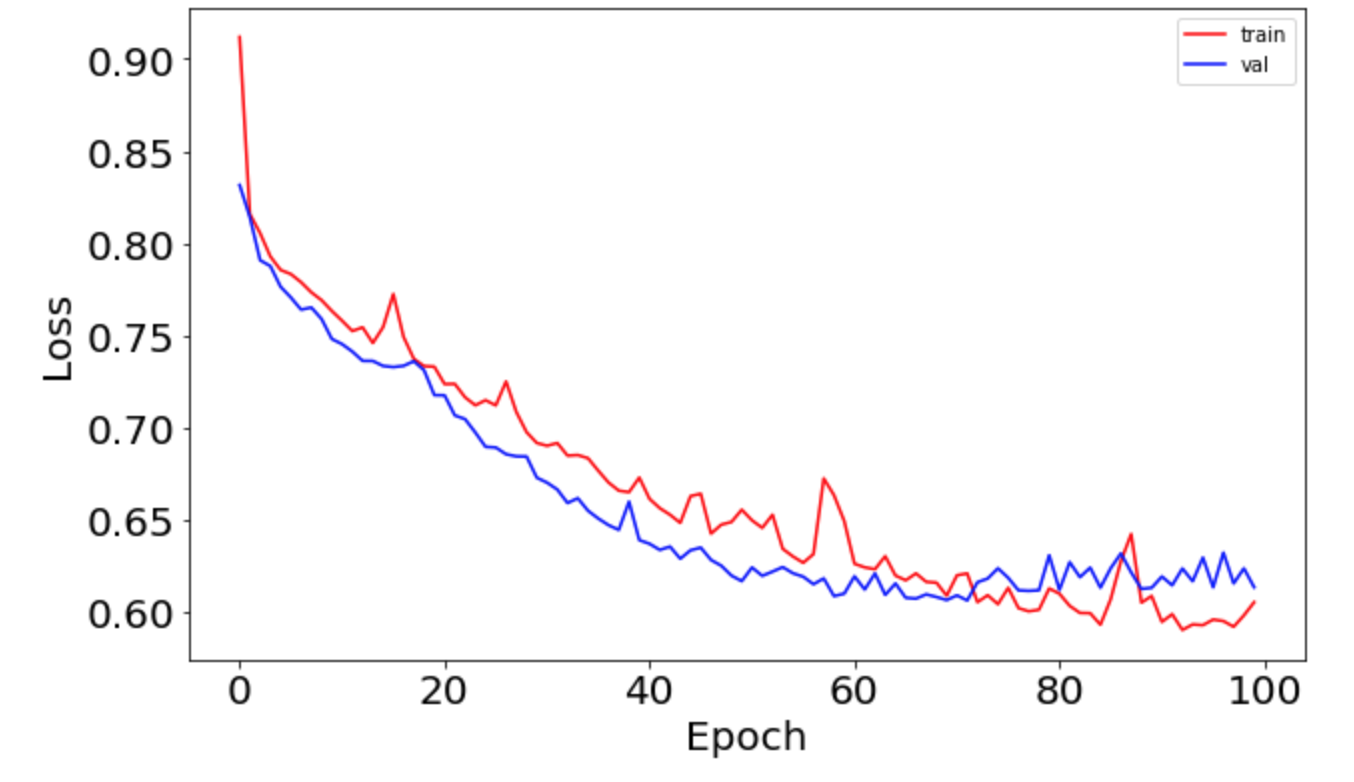
## Refinement and Results

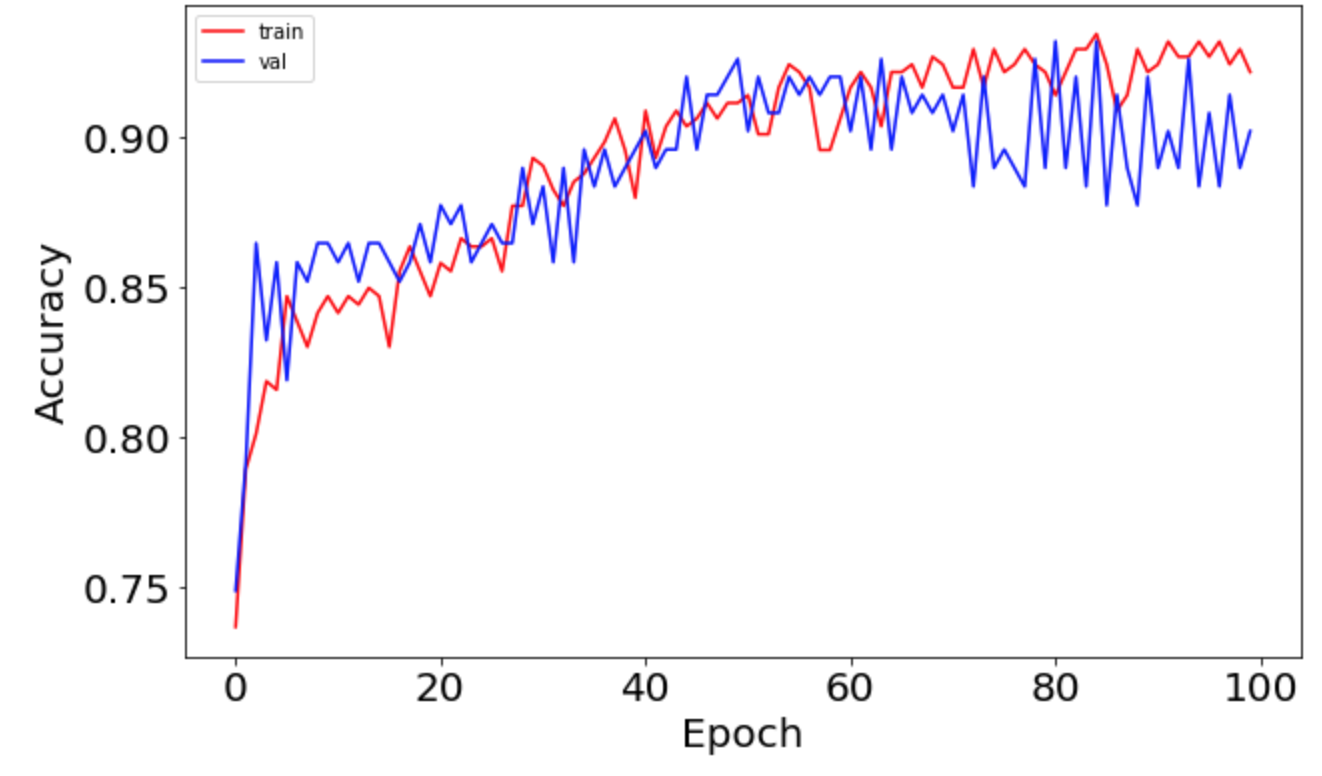
### Deep Neural Network

We performed hyper tuning of parameters of **Deep Neural Network** for binary classification by implementing the interface *tensorflow.keras.wrappers.scikit\_learn.KerasClassifier* and using the *sklearn.model\_selection.RandomizedSearchCV*

|  |  |  |
| --- | --- | --- |
| Parameter | Values Tested | Best Value |
| batch\_size | [10, 20, 40] | 40 |
| epochs | [100, 500, 1000] | 100 |
| optimizer | ['SGD', 'RMSprop', 'Adagrad', 'Adam'] | Adam |
| learning\_rate | [0.001, 0.01, 0.1] | 0.001 |
| init\_mode | ['uniform', 'normal', 'zero'] | 'normal' |

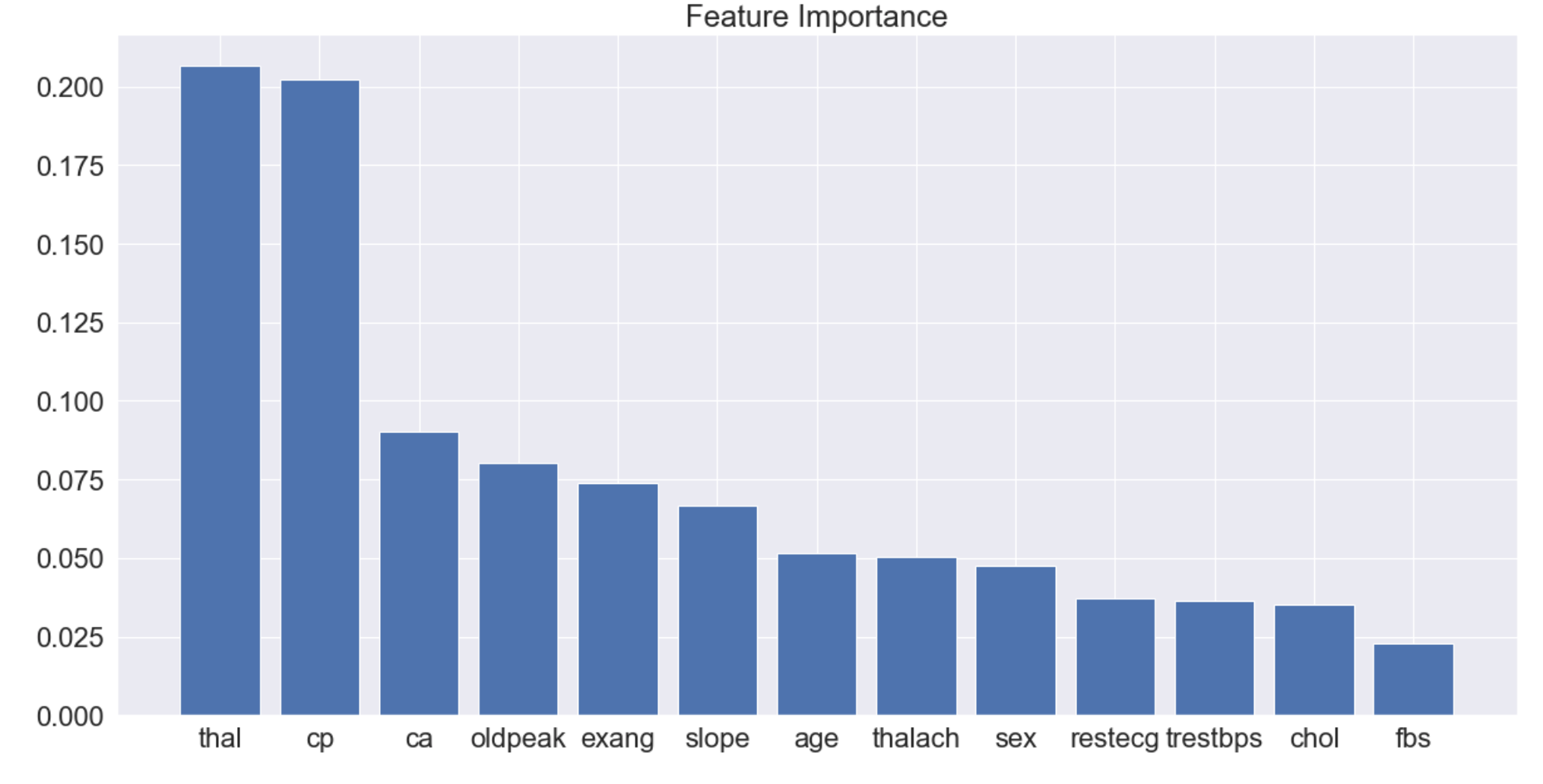






## Conclusion

The feature importance is visualized as:



The graph depicts the most influencing factors for the Heart Disease. Higher values mean the feature is more important in determining the Heart Disease.

As per the above graph 'thal','cp' and 'ca' are the highest influencing factor for the Heart Disease. This also matches with our initial finding in the data visualization section.

* 'cp' is Chest pain type, which determines the type angina. This is one of the major issues reported by the Heart Disease patients. Therefore, Heart Disease dependency on this feature is natural.
* 'thal' Thalassemia is an inherited blood disorder characterized by less hemoglobin and fewer red blood cells in your body than normal. Thalassemia is one of the main causes of Heart disease as Low levels of hemoglobin may cause anemia. Anemia is associated with a special risk in proatherosclerotic conditions and heart disease.
* 'ca' A noninvasive method for coronary artery diseases diagnosis. It indicates Number of major vessels to be get narrowed and are not able to convey enough fresh blood to this blood-pumping organ. This is also quite major factor to determine the heart disease.

**Model evaluations:**

We have used three different models for classification- KNN, Neural Network binary classifier, and XGBoost classifier. Based upon various evaluations matrix we see the XGBoost classifier performs best with highest accuracy of cross validation accuracy score 83.40%. Also, it is much easier to tune XGBoost classifier for the given problem and the dataset.

### Reflection

The most important and time-consuming part of the problem was data cleansing and processing as we need to ensure that we have correctly prepared the data well for the algorithm to work in its full potential.

Once the data was prepared and ready, the next challenge was to pick the optimized parameters for the selected algorithm and the data set.

As the data set was complex but small so may be Deep Neural Network was not the correct algorithm for the problem as it has not given any major advantage and XGBoost classifier has given better results.

The other issue I see is the inconsistency of the models to given consistent results in-spite of similar data set, parameters, setting of initializers and seed values.

There are many other ways to improve the model further like :

* Having more training data.
* Better model tuning using various hyper parameters
* Trying more different Algorithms

<https://arxiv.org/pdf/1506.05869v3>.