EC3067D

MATHEMATICS FOR MACHINE LEARNING

COURSE PROJECT

BOOK RECOMMENDATION SYSTEM

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

GROUP MEMBERS:

S.NO	ROLL NUMBER	NAME OF THE STUDENT
1.	B201032EC	PRANITH KUMAR BOGE
2.	B201015EC	PULYAPUDI ADITYA
3.	B201036EC	MOHINDER CHAWAN
4.	B200929EC	RISHI DHARMESHKUMAR SHAH

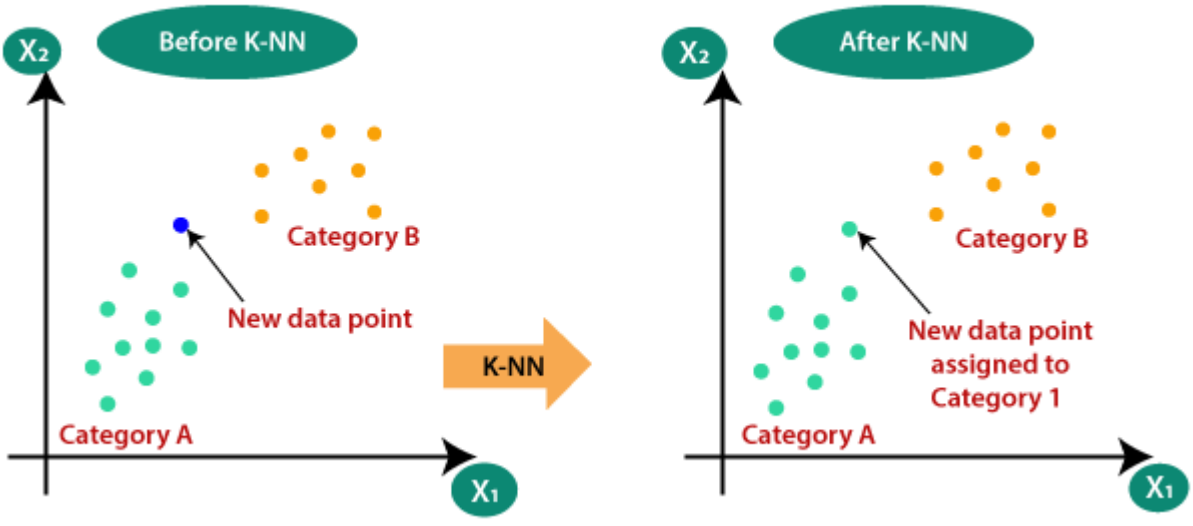
Aim:

The aim of this project is to develop a book recommendation system utilizing the K-Nearest Neighbours (KNN) algorithm to suggest books based on user ratings and book similarities.

Theory Background:

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. This algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. This algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

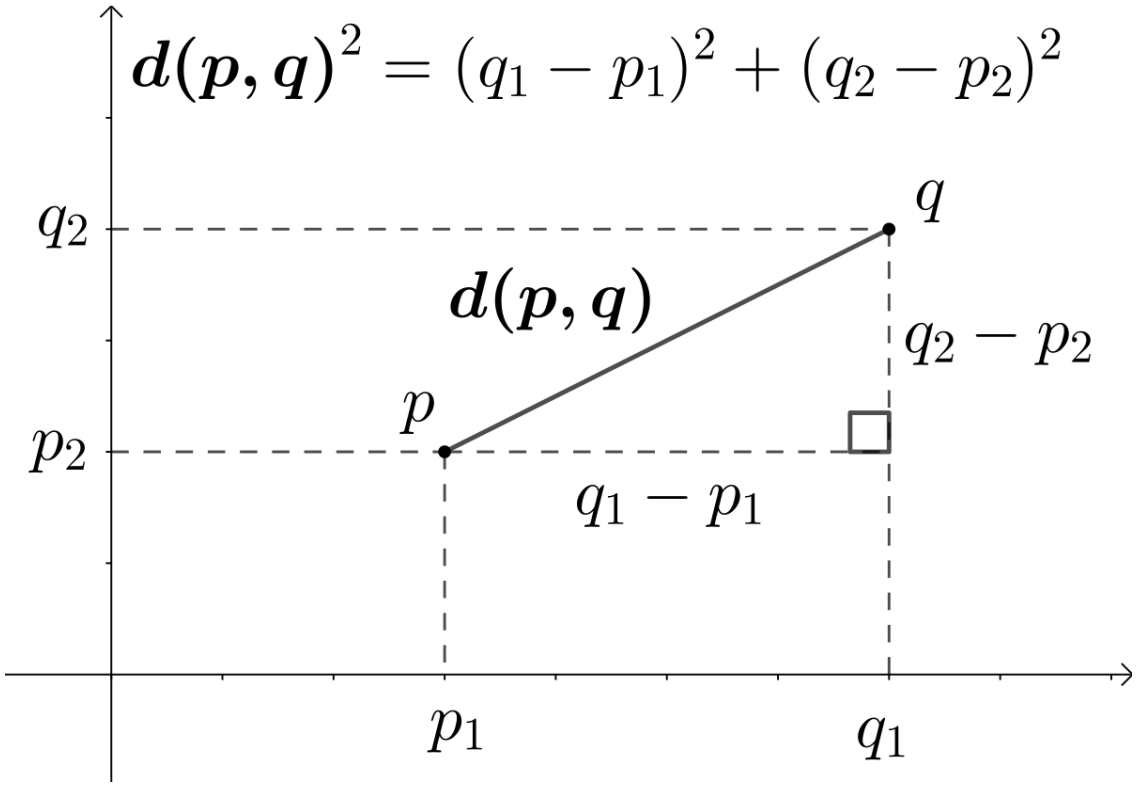
This algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems. K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data. It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset. KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.



Key Parameters:

K: The number of neighbours to consider. A common choice is an odd number to avoid ties in voting for classification.

Distance Metric: The measure used to calculate the distance between data points, such as Euclidean distance or Manhattan distance.

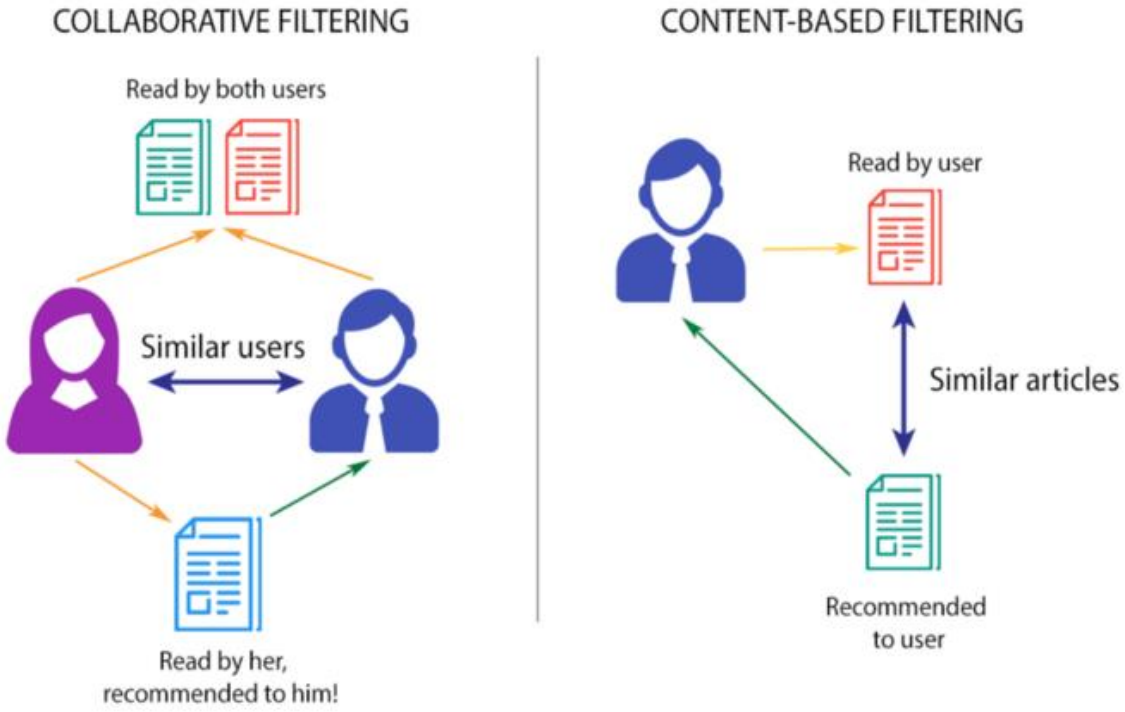


- Calculate the distance between the new data point and every point in the training dataset.
- Identify the 'k' nearest neighbours based on the calculated distances.
- For classification, assign the class label that occurs most frequently among the k neighbours to the new data point. For regression, predict the average of the target values of the k neighbours.

Book Recommendation System:

A recommendation system, a pivotal application of data science, provides relevant suggestions to users. It categorically utilizes three techniques: content-based filtering, collaborative filtering, and a hybrid of both.

- **Content-Based Filtering:** This technique recommends items similar to ones a user has interacted with. By analysing item attributes like tags or categories, it suggests similar products.
- **Collaborative-Based Filtering:** These systems base recommendations on user-item interactions. They identify users with similar preferences and suggest items based on what similar users have chosen. We are doing collaborative filtering for this project.



- **Hybrid Filtering:** This method combines both content-based and collaborative-based approaches. By leveraging user preferences and similarities between users, it offers recommendations based on individual history and similar user preferences.

A book recommendation system is a type of recommendation system where we have to recommend similar books to the reader based on his interest. The books recommendation system is used by online websites which provide ebooks like google play books, open library, good Read's, etc.

Practical Implementation of Recommendation System:

Dataset Description:

- **Books** – first are about books which contain all the information related to books like an author, title, publication year, etc.
- **Users** – The second file contains registered user's information like user id, location.
- **Ratings** – Ratings contain information like which user has given how much rating to which book.

The dataset is reliable and can consider as a large dataset. we have 271360 books data and total registered users on the website are approximately 278000 and they have given near about 11 lakh rating. Hence we can say that the dataset we have is nice and reliable.

Preprocessing Data:

- **Data Trimming:** Remove unnecessary columns (e.g., image URLs) from the books file and standardize column names to simplify usage.
- **Extract Users with More Than 200 Ratings:** Identify users who have given more than 200 ratings from the available data. Extract ratings from these specific users.
- **Merge Ratings with Books:** Merge the ratings data with the books data based on ISBN. This step is crucial to match ratings with respective books.
- **Filter Books with Over 50 Ratings:** Once merged, filter out books that have received more than 50 ratings. Aggregate ratings based on book titles.
- **Drop Duplicate Values:** Remove duplicate entries in case the same user has rated the same book multiple times. Ensure a clean dataset.
- **Create a Pivot Table:** Generate a pivot table where columns represent user IDs, the index corresponds to book titles, and the values denote ratings. Handle missing ratings (not rated by certain users) by replacing them with zero.
- **Sparse Matrix:** As the pivot table contains numerous zero values, which could affect computational efficiency during distance calculations, convert the pivot table into a sparse matrix. This transformation optimizes the process by minimizing the computation of distances for zero values.

Modeling:

- We have prepared our dataset for modeling. we will use the ***nearest neighbours algorithm*** which is the same as K nearest which is used for clustering based on Euclidian distance.
- Utilize the nearest neighbours algorithm, specifically specifying the ***brute-force method to compute distances between every point***. Train the model using the sparse matrix derived from the pivot table.
- **Generating Predictions:** Once the model is trained, make predictions using the trained model. Provide an input book ID and retrieve the nearest neighbours to that input. Fetch the top 5 books closest to the input book ID, displaying their respective distances and book IDs at those distances.

Algorithm Description:

Data Loading and Preprocessing:

- Data files are loaded using pd.read_csv.
- Columns are selected and renamed for Books and Users data.
- Books, Users, and Ratings data are read into Pandas DataFrames.
- The code prepares the necessary datasets by selecting specific columns, renaming columns, and filtering based on certain conditions.

Filtering Data for Recommendations:

- Filtering ratings to include users who have rated more than 200 books and books with at least 50 ratings.
- Creating a pivot table for books and their respective user ratings.

Building the Recommendation Model:

- Creating a sparse matrix from the pivot table using csr_matrix.
- Using NearestNeighbors from scikit-learn to build the recommendation model.

Recommendation Function:

- Defining a function recommend_books(book_name) that recommends similar books based on the input book name.

Code:

```
import pandas as pd
import numpy as np

book_path = "/content/drive/MyDrive/DataMML/Books.csv"
ratings_path = "/content/drive/MyDrive/DataMML/Ratings.csv"
users_path = "/content/drive/MyDrive/DataMML/Users.csv"

# Books CSV
books = pd.read_csv(book_path , error_bad_lines=False )
books = books[['ISBN', 'Book-Title', 'Book-Author', 'Year-Of-Publication',
'Publisher']]
books.rename(columns = {'Publisher' : 'publisher'},inplace = True)
books.rename(columns = {'Book-Title' : 'title'},inplace = True)
books.rename(columns = {'Book-Author' : 'author'},inplace = True)
books.rename(columns = {'Year-Of-Publication' : 'year'},inplace = True)

# Users CSV
users =  pd.read_csv(users_path , error_bad_lines=False )
users.rename(columns = {'User-ID' : 'user-id' , 'Location' : 'location'  ,
'Age' : 'age'},inplace = True)

# Rating CSV
ratings =  pd.read_csv(ratings_path , error_bad_lines=False )
ratings.rename(columns = {'User-ID' : 'user-id' , 'Book-Rating' :
'rating'},inplace = True)

FILTERING RATINGS TO INCLUDE USERS WHO HAVE RATED MORE THAN 200
BOOKS AND BOOKS WITH AT LEAST 50 RATINGS.


# Extract Users with More Than 200 Ratings
x = ratings['user-id'].value_counts()>200
y= x[x].index
ratings = ratings[ratings['user-id'].isin(y)]
ratings_with_books =  ratings.merge(books , on = 'ISBN')

# Merge Ratings with Books
number_rating =
ratings_with_books.groupby('title')['rating'].count().reset_index()
number_rating.rename(columns ={'rating':'number of ratings'} , inplace =True)
final_rating = ratings_with_books.merge(number_rating , on = 'title')

# Filter Books with Over 50 Ratings
new_final_rating = final_rating[final_rating['number of ratings']>=50]

# Drop Duplicate Values
new_final_rating.drop_duplicates(['user-id' , 'title'] , inplace = True)
```

```
# Create a Pivot Table
book_pivot = new_final_rating.pivot_table(columns = 'user-id' , index =
'title' , values = 'rating')
book_pivot.fillna(0 , inplace = True)

book_pivot.shape
(742, 888)

# Creating a sparse matrix from the pivot table using csr_matrix
from scipy.sparse import csr_matrix
book_sparse = csr_matrix(book_pivot)
type(book_sparse)

USING NEARESTNEIGHBORS FROM SCIKIT-LEARN TO BUILD THE RECOMMENDATION
MODEL

from sklearn.neighbors import NearestNeighbors
model = NearestNeighbors( algorithm = 'brute')
model.fit(book_sparse)

DEFINING A FUNCTION RECOMMEND_BOOKS(BOOK_NAME) THAT RECOMMENDS
SIMILAR BOOKS BASED ON THE INPUT BOOK NAME.

def recommend_books(book_name):
  book_id = np.where(book_pivot.index == book_name)[0][0]
  distances , suggestions = model.kneighbors (book_pivot.iloc[book_id ,
:].values.reshape(1, -1) , n_neighbors = 6)

  for i in range(len(suggestions)):
    if i ==0:
      print("The suggestions for" , book_name , "are :")
    if not i :
      print(book_pivot.index[suggestions[i]])
```

Results:

```
recommend_books('Harry Potter and the Chamber of Secrets (Book 2)')

The suggestions for Harry Potter and the Chamber of Secrets (Book 2) are:
Index(['Harry Potter and the Chamber of Secrets (Book 2)',
       'Harry Potter and the Prisoner of Azkaban (Book 3)',
       'Harry Potter and the Goblet of Fire (Book 4)',
       'Harry Potter and the Sorcerer's Stone (Book 1)', 'Exclusive',
       'The Cradle Will Fall'],
      dtype='object', name='title')
```