{

 "cells": [

  {

   "cell\_type": "markdown",

   "metadata": {},

   "source": [

    "# Recommendation Engine\n",

    "## Building a Movie Recommendation Engine using MovieLens dataset \n",

    "We will be using a MovieLens dataset. This dataset contains 100004 ratings across 9125 movies for 671 users. All selected users had at least rated 20 movies. \n",

    "We are going to build a recommendation engine which will suggest movies for a user which he hasn't watched yet based on the movies which he has already rated. We will be using k-nearest neighbour algorithm which we will implement from scratch."

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 45,

   "metadata": {},

   "outputs": [],

   "source": [

    "import pandas as pd"

   ]

  },

  {

   "cell\_type": "markdown",

   "metadata": {},

   "source": [

    "Movie file contains information like movie id, title, genre of movies and ratings file contains data like user id, movie id, rating and timestamp in which each line after header row represents one rating of one movie by one user."

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 71,

   "metadata": {},

   "outputs": [

    {

     "data": {

      "text/html": [

       "<div>\n",

       "<style scoped>\n",

       "    .dataframe tbody tr th:only-of-type {\n",

       "        vertical-align: middle;\n",

       "    }\n",

       "\n",

       "    .dataframe tbody tr th {\n",

       "        vertical-align: top;\n",

       "    }\n",

       "\n",

       "    .dataframe thead th {\n",

       "        text-align: right;\n",

       "    }\n",

       "</style>\n",

       "<table border=\"1\" class=\"dataframe\">\n",

       "  <thead>\n",

       "    <tr style=\"text-align: right;\">\n",

       "      <th></th>\n",

       "      <th>movieId</th>\n",

       "      <th>title</th>\n",

       "    </tr>\n",

       "  </thead>\n",

       "  <tbody>\n",

       "    <tr>\n",

       "      <th>0</th>\n",

       "      <td>1</td>\n",

       "      <td>Toy Story (1995)</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>1</th>\n",

       "      <td>2</td>\n",

       "      <td>Jumanji (1995)</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>2</th>\n",

       "      <td>3</td>\n",

       "      <td>Grumpier Old Men (1995)</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>3</th>\n",

       "      <td>4</td>\n",

       "      <td>Waiting to Exhale (1995)</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>4</th>\n",

       "      <td>5</td>\n",

       "      <td>Father of the Bride Part II (1995)</td>\n",

       "    </tr>\n",

       "  </tbody>\n",

       "</table>\n",

       "</div>"

      ],

      "text/plain": [

       "   movieId                               title\n",

       "0        1                    Toy Story (1995)\n",

       "1        2                      Jumanji (1995)\n",

       "2        3             Grumpier Old Men (1995)\n",

       "3        4            Waiting to Exhale (1995)\n",

       "4        5  Father of the Bride Part II (1995)"

      ]

     },

     "execution\_count": 71,

     "metadata": {},

     "output\_type": "execute\_result"

    }

   ],

   "source": [

    "movie\_file = \"data\\movie\_dataset\\movies.csv\"\n",

    "movie\_data = pd.read\_csv(movie\_file, usecols = [0, 1])\n",

    "movie\_data.head()"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 72,

   "metadata": {},

   "outputs": [

    {

     "data": {

      "text/html": [

       "<div>\n",

       "<style scoped>\n",

       "    .dataframe tbody tr th:only-of-type {\n",

       "        vertical-align: middle;\n",

       "    }\n",

       "\n",

       "    .dataframe tbody tr th {\n",

       "        vertical-align: top;\n",

       "    }\n",

       "\n",

       "    .dataframe thead th {\n",

       "        text-align: right;\n",

       "    }\n",

       "</style>\n",

       "<table border=\"1\" class=\"dataframe\">\n",

       "  <thead>\n",

       "    <tr style=\"text-align: right;\">\n",

       "      <th></th>\n",

       "      <th>userId</th>\n",

       "      <th>movieId</th>\n",

       "      <th>rating</th>\n",

       "    </tr>\n",

       "  </thead>\n",

       "  <tbody>\n",

       "    <tr>\n",

       "      <th>0</th>\n",

       "      <td>1</td>\n",

       "      <td>31</td>\n",

       "      <td>2.5</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>1</th>\n",

       "      <td>1</td>\n",

       "      <td>1029</td>\n",

       "      <td>3.0</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>2</th>\n",

       "      <td>1</td>\n",

       "      <td>1061</td>\n",

       "      <td>3.0</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>3</th>\n",

       "      <td>1</td>\n",

       "      <td>1129</td>\n",

       "      <td>2.0</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>4</th>\n",

       "      <td>1</td>\n",

       "      <td>1172</td>\n",

       "      <td>4.0</td>\n",

       "    </tr>\n",

       "  </tbody>\n",

       "</table>\n",

       "</div>"

      ],

      "text/plain": [

       "   userId  movieId  rating\n",

       "0       1       31     2.5\n",

       "1       1     1029     3.0\n",

       "2       1     1061     3.0\n",

       "3       1     1129     2.0\n",

       "4       1     1172     4.0"

      ]

     },

     "execution\_count": 72,

     "metadata": {},

     "output\_type": "execute\_result"

    }

   ],

   "source": [

    "ratings\_file = \"data\\\\movie\_dataset\\\\ratings.csv\"\n",

    "ratings\_info = pd.read\_csv(ratings\_file, usecols = [0, 1, 2])\n",

    "ratings\_info.head()"

   ]

  },

  {

   "cell\_type": "markdown",

   "metadata": {},

   "source": []

  },

  {

   "cell\_type": "code",

   "execution\_count": 76,

   "metadata": {},

   "outputs": [

    {

     "data": {

      "text/html": [

       "<div>\n",

       "<style scoped>\n",

       "    .dataframe tbody tr th:only-of-type {\n",

       "        vertical-align: middle;\n",

       "    }\n",

       "\n",

       "    .dataframe tbody tr th {\n",

       "        vertical-align: top;\n",

       "    }\n",

       "\n",

       "    .dataframe thead th {\n",

       "        text-align: right;\n",

       "    }\n",

       "</style>\n",

       "<table border=\"1\" class=\"dataframe\">\n",

       "  <thead>\n",

       "    <tr style=\"text-align: right;\">\n",

       "      <th></th>\n",

       "      <th>movieId</th>\n",

       "      <th>title</th>\n",

       "      <th>userId</th>\n",

       "      <th>rating</th>\n",

       "    </tr>\n",

       "  </thead>\n",

       "  <tbody>\n",

       "    <tr>\n",

       "      <th>0</th>\n",

       "      <td>1</td>\n",

       "      <td>Toy Story (1995)</td>\n",

       "      <td>7</td>\n",

       "      <td>3.0</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>1</th>\n",

       "      <td>1</td>\n",

       "      <td>Toy Story (1995)</td>\n",

       "      <td>9</td>\n",

       "      <td>4.0</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>2</th>\n",

       "      <td>1</td>\n",

       "      <td>Toy Story (1995)</td>\n",

       "      <td>13</td>\n",

       "      <td>5.0</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>3</th>\n",

       "      <td>1</td>\n",

       "      <td>Toy Story (1995)</td>\n",

       "      <td>15</td>\n",

       "      <td>2.0</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>4</th>\n",

       "      <td>1</td>\n",

       "      <td>Toy Story (1995)</td>\n",

       "      <td>19</td>\n",

       "      <td>3.0</td>\n",

       "    </tr>\n",

       "  </tbody>\n",

       "</table>\n",

       "</div>"

      ],

      "text/plain": [

       "   movieId             title  userId  rating\n",

       "0        1  Toy Story (1995)       7     3.0\n",

       "1        1  Toy Story (1995)       9     4.0\n",

       "2        1  Toy Story (1995)      13     5.0\n",

       "3        1  Toy Story (1995)      15     2.0\n",

       "4        1  Toy Story (1995)      19     3.0"

      ]

     },

     "execution\_count": 76,

     "metadata": {},

     "output\_type": "execute\_result"

    }

   ],

   "source": [

    "movie\_info = pd.merge(movie\_data, ratings\_info, left\_on = 'movieId', right\_on = 'movieId')\n",

    "movie\_info.head()"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 49,

   "metadata": {},

   "outputs": [

    {

     "data": {

      "text/html": [

       "<div>\n",

       "<style scoped>\n",

       "    .dataframe tbody tr th:only-of-type {\n",

       "        vertical-align: middle;\n",

       "    }\n",

       "\n",

       "    .dataframe tbody tr th {\n",

       "        vertical-align: top;\n",

       "    }\n",

       "\n",

       "    .dataframe thead th {\n",

       "        text-align: right;\n",

       "    }\n",

       "</style>\n",

       "<table border=\"1\" class=\"dataframe\">\n",

       "  <thead>\n",

       "    <tr style=\"text-align: right;\">\n",

       "      <th></th>\n",

       "      <th>movieId</th>\n",

       "      <th>title</th>\n",

       "      <th>userId</th>\n",

       "      <th>rating</th>\n",

       "    </tr>\n",

       "  </thead>\n",

       "  <tbody>\n",

       "    <tr>\n",

       "      <th>0</th>\n",

       "      <td>1</td>\n",

       "      <td>Toy Story (1995)</td>\n",

       "      <td>7</td>\n",

       "      <td>3.0</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>1</th>\n",

       "      <td>1</td>\n",

       "      <td>Toy Story (1995)</td>\n",

       "      <td>9</td>\n",

       "      <td>4.0</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>2</th>\n",

       "      <td>1</td>\n",

       "      <td>Toy Story (1995)</td>\n",

       "      <td>13</td>\n",

       "      <td>5.0</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>3</th>\n",

       "      <td>1</td>\n",

       "      <td>Toy Story (1995)</td>\n",

       "      <td>15</td>\n",

       "      <td>2.0</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>4</th>\n",

       "      <td>1</td>\n",

       "      <td>Toy Story (1995)</td>\n",

       "      <td>19</td>\n",

       "      <td>3.0</td>\n",

       "    </tr>\n",

       "  </tbody>\n",

       "</table>\n",

       "</div>"

      ],

      "text/plain": [

       "   movieId             title  userId  rating\n",

       "0        1  Toy Story (1995)       7     3.0\n",

       "1        1  Toy Story (1995)       9     4.0\n",

       "2        1  Toy Story (1995)      13     5.0\n",

       "3        1  Toy Story (1995)      15     2.0\n",

       "4        1  Toy Story (1995)      19     3.0"

      ]

     },

     "execution\_count": 49,

     "metadata": {},

     "output\_type": "execute\_result"

    }

   ],

   "source": [

    "movie\_info.loc[0:10, ['userId']]\n",

    "movie\_info[movie\_info.title == \"Toy Story (1995)\"].head()"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 50,

   "metadata": {},

   "outputs": [

    {

     "data": {

      "text/html": [

       "<div>\n",

       "<style scoped>\n",

       "    .dataframe tbody tr th:only-of-type {\n",

       "        vertical-align: middle;\n",

       "    }\n",

       "\n",

       "    .dataframe tbody tr th {\n",

       "        vertical-align: top;\n",

       "    }\n",

       "\n",

       "    .dataframe thead th {\n",

       "        text-align: right;\n",

       "    }\n",

       "</style>\n",

       "<table border=\"1\" class=\"dataframe\">\n",

       "  <thead>\n",

       "    <tr style=\"text-align: right;\">\n",

       "      <th></th>\n",

       "      <th>movieId</th>\n",

       "      <th>title</th>\n",

       "      <th>userId</th>\n",

       "      <th>rating</th>\n",

       "    </tr>\n",

       "  </thead>\n",

       "  <tbody>\n",

       "    <tr>\n",

       "      <th>246</th>\n",

       "      <td>1</td>\n",

       "      <td>Toy Story (1995)</td>\n",

       "      <td>671</td>\n",

       "      <td>5.0</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>2111</th>\n",

       "      <td>36</td>\n",

       "      <td>Dead Man Walking (1995)</td>\n",

       "      <td>671</td>\n",

       "      <td>4.0</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>2843</th>\n",

       "      <td>50</td>\n",

       "      <td>Usual Suspects, The (1995)</td>\n",

       "      <td>671</td>\n",

       "      <td>4.5</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>6715</th>\n",

       "      <td>230</td>\n",

       "      <td>Dolores Claiborne (1995)</td>\n",

       "      <td>671</td>\n",

       "      <td>4.0</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>7809</th>\n",

       "      <td>260</td>\n",

       "      <td>Star Wars: Episode IV - A New Hope (1977)</td>\n",

       "      <td>671</td>\n",

       "      <td>5.0</td>\n",

       "    </tr>\n",

       "  </tbody>\n",

       "</table>\n",

       "</div>"

      ],

      "text/plain": [

       "      movieId                                      title  userId  rating\n",

       "246         1                           Toy Story (1995)     671     5.0\n",

       "2111       36                    Dead Man Walking (1995)     671     4.0\n",

       "2843       50                 Usual Suspects, The (1995)     671     4.5\n",

       "6715      230                   Dolores Claiborne (1995)     671     4.0\n",

       "7809      260  Star Wars: Episode IV - A New Hope (1977)     671     5.0"

      ]

     },

     "execution\_count": 50,

     "metadata": {},

     "output\_type": "execute\_result"

    }

   ],

   "source": [

    "movie\_info = pd.DataFrame.sort\_values(movie\_info, ['userId', 'movieId'], ascending = [0, 1])\n",

    "movie\_info.head()"

   ]

  },

  {

   "cell\_type": "markdown",

   "metadata": {},

   "source": [

    "Let us see the number of users and number of movies in our dataset"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 51,

   "metadata": {},

   "outputs": [

    {

     "name": "stdout",

     "output\_type": "stream",

     "text": [

      "671\n",

      "163949\n"

     ]

    }

   ],

   "source": [

    "num\_users = max(movie\_info.userId)\n",

    "num\_movies = max(movie\_info.movieId)\n",

    "print(num\_users)\n",

    "print(num\_movies)"

   ]

  },

  {

   "cell\_type": "markdown",

   "metadata": {},

   "source": [

    "how many movies were rated by each user and the number of users rated each movie"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 52,

   "metadata": {

    "scrolled": true

   },

   "outputs": [

    {

     "data": {

      "text/plain": [

       "547    2391\n",

       "564    1868\n",

       "624    1735\n",

       "15     1700\n",

       "73     1610\n",

       "Name: userId, dtype: int64"

      ]

     },

     "execution\_count": 52,

     "metadata": {},

     "output\_type": "execute\_result"

    }

   ],

   "source": [

    "movie\_per\_user = movie\_info.userId.value\_counts()\n",

    "movie\_per\_user.head()"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 53,

   "metadata": {},

   "outputs": [

    {

     "data": {

      "text/plain": [

       "Forrest Gump (1994)                          341\n",

       "Pulp Fiction (1994)                          324\n",

       "Shawshank Redemption, The (1994)             311\n",

       "Silence of the Lambs, The (1991)             304\n",

       "Star Wars: Episode IV - A New Hope (1977)    291\n",

       "Name: title, dtype: int64"

      ]

     },

     "execution\_count": 53,

     "metadata": {},

     "output\_type": "execute\_result"

    }

   ],

   "source": [

    "users\_per\_movie = movie\_info.title.value\_counts()\n",

    "users\_per\_movie.head()"

   ]

  },

  {

   "cell\_type": "markdown",

   "metadata": {},

   "source": [

    "Function to find top N favourite movies of a user"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 54,

   "metadata": {},

   "outputs": [

    {

     "name": "stdout",

     "output\_type": "stream",

     "text": [

      "['Grease (1978)', 'Little Mermaid, The (1989)', 'Sound of Music, The (1965)']\n"

     ]

    }

   ],

   "source": [

    "def fav\_movies(current\_user, N):\n",

    "    # get rows corresponding to current user and then sort by rating in descending order\n",

    "    # pick top N rows of the dataframe\n",

    "    fav\_movies = pd.DataFrame.sort\_values(movie\_info[movie\_info.userId == current\_user], \n",

    "                                          ['rating'], ascending = [0]) [:N]\n",

    "    # return list of titles\n",

    "    return list(fav\_movies.title)\n",

    "\n",

    "print(fav\_movies(5, 3))\n",

    "    \n",

    "    "

   ]

  },

  {

   "cell\_type": "markdown",

   "metadata": {},

   "source": [

    "Lets build recommendation engine now\n",

    "\n",

    "- We will use a neighbour based collaborative filtering model. \n",

    "- The idea is to use k-nearest neighbour algorithm to find neighbours of a user\n",

    "-  We will use their ratings to predict ratings of a movie not already rated by a current user.\n",

    "\n",

    "We will represent movies watched by a user in a vector - the vector will have values for all the movies in our dataset.\n",

    "If a user hasn't rated a movie, it would be represented as NaN."

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 55,

   "metadata": {},

   "outputs": [

    {

     "data": {

      "text/html": [

       "<div>\n",

       "<style scoped>\n",

       "    .dataframe tbody tr th:only-of-type {\n",

       "        vertical-align: middle;\n",

       "    }\n",

       "\n",

       "    .dataframe tbody tr th {\n",

       "        vertical-align: top;\n",

       "    }\n",

       "\n",

       "    .dataframe thead th {\n",

       "        text-align: right;\n",

       "    }\n",

       "</style>\n",

       "<table border=\"1\" class=\"dataframe\">\n",

       "  <thead>\n",

       "    <tr style=\"text-align: right;\">\n",

       "      <th>movieId</th>\n",

       "      <th>1</th>\n",

       "      <th>2</th>\n",

       "      <th>3</th>\n",

       "      <th>4</th>\n",

       "      <th>5</th>\n",

       "      <th>6</th>\n",

       "      <th>7</th>\n",

       "      <th>8</th>\n",

       "      <th>9</th>\n",

       "      <th>10</th>\n",

       "      <th>...</th>\n",

       "      <th>161084</th>\n",

       "      <th>161155</th>\n",

       "      <th>161594</th>\n",

       "      <th>161830</th>\n",

       "      <th>161918</th>\n",

       "      <th>161944</th>\n",

       "      <th>162376</th>\n",

       "      <th>162542</th>\n",

       "      <th>162672</th>\n",

       "      <th>163949</th>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>userId</th>\n",

       "      <th></th>\n",

       "      <th></th>\n",

       "      <th></th>\n",

       "      <th></th>\n",

       "      <th></th>\n",

       "      <th></th>\n",

       "      <th></th>\n",

       "      <th></th>\n",

       "      <th></th>\n",

       "      <th></th>\n",

       "      <th></th>\n",

       "      <th></th>\n",

       "      <th></th>\n",

       "      <th></th>\n",

       "      <th></th>\n",

       "      <th></th>\n",

       "      <th></th>\n",

       "      <th></th>\n",

       "      <th></th>\n",

       "      <th></th>\n",

       "      <th></th>\n",

       "    </tr>\n",

       "  </thead>\n",

       "  <tbody>\n",

       "    <tr>\n",

       "      <th>1</th>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>...</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>2</th>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>4.0</td>\n",

       "      <td>...</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>3</th>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>...</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>4</th>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>4.0</td>\n",

       "      <td>...</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "    </tr>\n",

       "    <tr>\n",

       "      <th>5</th>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>4.0</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>...</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "      <td>NaN</td>\n",

       "    </tr>\n",

       "  </tbody>\n",

       "</table>\n",

       "<p>5 rows × 9066 columns</p>\n",

       "</div>"

      ],

      "text/plain": [

       "movieId  1       2       3       4       5       6       7       8       \\\n",

       "userId                                                                    \n",

       "1           NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN   \n",

       "2           NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN   \n",

       "3           NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN   \n",

       "4           NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN   \n",

       "5           NaN     NaN     4.0     NaN     NaN     NaN     NaN     NaN   \n",

       "\n",

       "movieId  9       10       ...    161084  161155  161594  161830  161918  \\\n",

       "userId                    ...                                             \n",

       "1           NaN     NaN   ...       NaN     NaN     NaN     NaN     NaN   \n",

       "2           NaN     4.0   ...       NaN     NaN     NaN     NaN     NaN   \n",

       "3           NaN     NaN   ...       NaN     NaN     NaN     NaN     NaN   \n",

       "4           NaN     4.0   ...       NaN     NaN     NaN     NaN     NaN   \n",

       "5           NaN     NaN   ...       NaN     NaN     NaN     NaN     NaN   \n",

       "\n",

       "movieId  161944  162376  162542  162672  163949  \n",

       "userId                                           \n",

       "1           NaN     NaN     NaN     NaN     NaN  \n",

       "2           NaN     NaN     NaN     NaN     NaN  \n",

       "3           NaN     NaN     NaN     NaN     NaN  \n",

       "4           NaN     NaN     NaN     NaN     NaN  \n",

       "5           NaN     NaN     NaN     NaN     NaN  \n",

       "\n",

       "[5 rows x 9066 columns]"

      ]

     },

     "execution\_count": 55,

     "metadata": {},

     "output\_type": "execute\_result"

    }

   ],

   "source": [

    "user\_movie\_rating\_matrix = pd.pivot\_table(movie\_info, values = 'rating', index=['userId'], columns=['movieId'])\n",

    "user\_movie\_rating\_matrix.head()"

   ]

  },

  {

   "cell\_type": "markdown",

   "metadata": {},

   "source": [

    "Now, we will find the similarity between 2 users by using correlation "

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 62,

   "metadata": {},

   "outputs": [],

   "source": [

    "from scipy.spatial.distance import correlation\n",

    "import numpy as np\n",

    "def similarity(user1, user2):\n",

    "    # normalizing user1 rating i.e mean rating of user1 for any movie\n",

    "    # nanmean will return mean of an array after ignore NaN values \n",

    "    user1 = np.array(user1) - np.nanmean(user1) \n",

    "    user2 = np.array(user2) - np.nanmean(user2)\n",

    "    \n",

    "    # finding the similarity between 2 users\n",

    "    # finding subset of movies rated by both the users\n",

    "    common\_movie\_ids = [i for i in range(len(user1)) if user1[i] > 0 and user2[i] > 0]\n",

    "    if(len(common\_movie\_ids) == 0):\n",

    "        return 0\n",

    "    else:\n",

    "        user1 = np.array([user1[i] for i in common\_movie\_ids])\n",

    "        user2 = np.array([user2[i] for i in common\_movie\_ids])\n",

    "        return correlation(user1, user2)"

   ]

  },

  {

   "cell\_type": "markdown",

   "metadata": {},

   "source": [

    " We will now use the similarity function to find the nearest neighbour of a current user"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 64,

   "metadata": {},

   "outputs": [],

   "source": [

    "# nearest\_neighbour\_ratings function will find the k nearest neighbours of the current user and\n",

    "# then use their ratings to predict the current users ratings for other unrated movies \n",

    "\n",

    "def nearest\_neighbour\_ratings(current\_user, K):\n",

    "    # Creating an empty matrix whose row index is userId and the value\n",

    "    # will be the similarity of that user to the current user\n",

    "    similarity\_matrix = pd.DataFrame(index = user\_movie\_rating\_matrix.index, \n",

    "                                    columns = ['similarity'])\n",

    "    for i in user\_movie\_rating\_matrix.index:\n",

    "        # finding the similarity between user i and the current user and add it to the similarity matrix\n",

    "        similarity\_matrix.loc[i] = similarity(user\_movie\_rating\_matrix.loc[current\_user],\n",

    "                                             user\_movie\_rating\_matrix.loc[i])\n",

    "    # Sorting the similarity matrix in descending order\n",

    "    similarity\_matrix = pd.DataFrame.sort\_values(similarity\_matrix,\n",

    "                                                ['similarity'], ascending= [0])\n",

    "    # now we will pick the top k nearest neighbou\n",

    "    nearest\_neighbours = similarity\_matrix[:K]\n",

    "\n",

    "    neighbour\_movie\_ratings = user\_movie\_rating\_matrix.loc[nearest\_neighbours.index]\n",

    "\n",

    "    # This is empty dataframe placeholder for predicting the rating of current user using neighbour movie ratings\n",

    "    predicted\_movie\_rating = pd.DataFrame(index = user\_movie\_rating\_matrix.columns, columns = ['rating'])\n",

    "\n",

    "    # Iterating all movies for a current user\n",

    "    for i in user\_movie\_rating\_matrix.columns:\n",

    "        # by default, make predicted rating as the average rating of the current user\n",

    "        predicted\_rating = np.nanmean(user\_movie\_rating\_matrix.loc[current\_user])\n",

    "\n",

    "        for j in neighbour\_movie\_ratings.index:\n",

    "            # if user j has rated the ith movie\n",

    "            if(user\_movie\_rating\_matrix.loc[j,i] > 0):\n",

    "                predicted\_rating += ((user\_movie\_rating\_matrix.loc[j,i] -np.nanmean(user\_movie\_rating\_matrix.loc[j])) \*\n",

    "                                                    nearest\_neighbours.loc[j, 'similarity']) / nearest\_neighbours['similarity'].sum()\n",

    "\n",

    "        predicted\_movie\_rating.loc[i, 'rating'] = predicted\_rating\n",

    "\n",

    "    return predicted\_movie\_rating"

   ]

  },

  {

   "cell\_type": "markdown",

   "metadata": {},

   "source": [

    "Predicting top N recommendations for a current user"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 79,

   "metadata": {},

   "outputs": [],

   "source": [

    "def top\_n\_recommendations(current\_user, N):\n",

    "    predicted\_movie\_rating = nearest\_neighbour\_ratings(current\_user, 10)\n",

    "    movies\_already\_watched = list(user\_movie\_rating\_matrix.loc[current\_user]\n",

    "                                  .loc[user\_movie\_rating\_matrix.loc[current\_user] > 0].index)\n",

    "    \n",

    "    predicted\_movie\_rating = predicted\_movie\_rating.drop(movies\_already\_watched)\n",

    "    \n",

    "    top\_n\_recommendations = pd.DataFrame.sort\_values(predicted\_movie\_rating, ['rating'], ascending=[0])[:N]\n",

    "    \n",

    "    top\_n\_recommendation\_titles = movie\_data.loc[movie\_data.movieId.isin(top\_n\_recommendations.index)]\n",

    "\n",

    "    return list(top\_n\_recommendation\_titles.title)"

   ]

  },

  {

   "cell\_type": "markdown",

   "metadata": {},

   "source": [

    "finding out the recommendations for a user"

   ]

  },

  {

   "cell\_type": "code",

   "execution\_count": 80,

   "metadata": {

    "scrolled": true

   },

   "outputs": [

    {

     "name": "stderr",

     "output\_type": "stream",

     "text": [

      "C:\\Users\\erchh\\Anaconda3\\envs\\tensorflow\\lib\\site-packages\\scipy\\spatial\\distance.py:644: RuntimeWarning: invalid value encountered in double\_scalars\n",

      "  dist = 1.0 - uv / np.sqrt(uu \* vv)\n"

     ]

    },

    {

     "name": "stdout",

     "output\_type": "stream",

     "text": [

      "User's favorite movies are :  ['Shawshank Redemption, The (1994)', 'Father of the Bride Part II (1995)', 'Cast Away (2000)', 'Parent Trap, The (1998)', \"Ocean's Eleven (2001)\"] \n",

      "User's top recommendations are:  ['Godfather, The (1972)', 'Star Wars: Episode V - The Empire Strikes Back (1980)', 'Godfather: Part II, The (1974)']\n"

     ]

    }

   ],

   "source": [

    "current\_user = 140\n",

    "print(\"User's favorite movies are : \", fav\_movies(current\_user, 5),\n",

    "      \"\\nUser's top recommendations are: \", top\_n\_recommendations(current\_user, 3))"

   ]

  },

  {

   "cell\_type": "markdown",

   "metadata": {},

   "source": [

    "## Conclusion\n",

    "We have built a movie recommendation engine using k-nearest neighbour algorithm implemented from scratch. "

   ]

  }

 ],

 "metadata": {

  "kernelspec": {

   "display\_name": "Python 3",

   "language": "python",

   "name": "python3"

  },

  "language\_info": {

   "codemirror\_mode": {

    "name": "ipython",

    "version": 3

   },

   "file\_extension": ".py",

   "mimetype": "text/x-python",

   "name": "python",

   "nbconvert\_exporter": "python",

   "pygments\_lexer": "ipython3",

   "version": "3.5.4"

  }

 },

 "nbformat": 4,

 "nbformat\_minor": 2

}