Basic recommender system using C#

We have this set of data which represents users' ratings for different products.

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
User 1	7	6	7	?	5	7	7	1	1	?
User 2	?	3	2	2	3	?	?	1	1	3
User 3	3	4	1	3	?	2	1	1	2	3
User 4	?	?	1	4	?	2	3	?	?	1
User 5	5	?	?	?	5	?	7	1	0	6
User 6	?	1	4	0	1	4	2	6	6	1

User1 gave P1 a 7/10 rating, P2 a 6/10 rating… but he didn't rate P4, so we don't know if User1 would like us to show him P4 or similar products or not. Same thing for P10, User2 and P1… and so on.

Our goal is to find a relationship between these users and predict these missing values.

Similarity between users:

First, we will measure the similarity between the users by calculating the Pearson Correlation Coefficient using this formula.

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

r = correlation coefficient

x_i = values of the x-variable in a sample

$\bar{x}$ = mean of the values of the x-variable

y_i = values of the y-variable in a sample

$\bar{y}$ = mean of the values of the y-variable

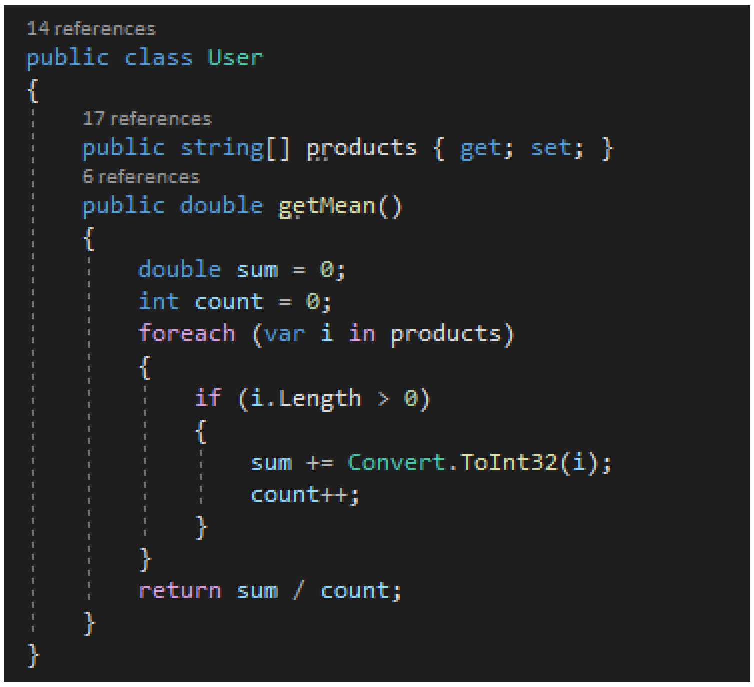
The correlation coefficient takes values between 1 and -1. A correlation of 1 shows a perfect positive correlation, while a correlation of -1 shows a perfect negative correlation. A correlation of 0 shows no linear relationship between the variables.

Implementation in C#:

** The data is loaded from a comma separated csv file, each entry is an instance of class named User and each row is a vector stored in that instance.

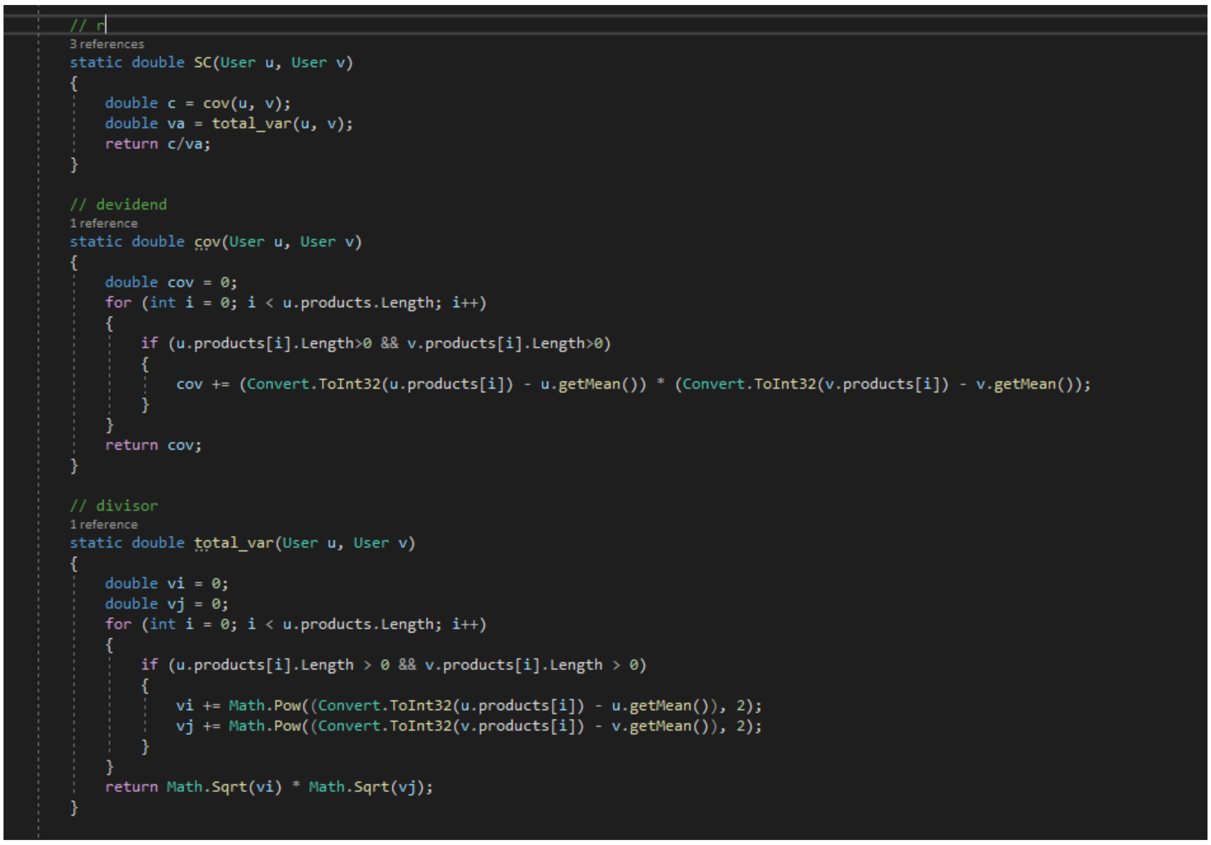
To calculate r I made two separate functions one to calculate the dividend, in this case the covariance of the 2 variables, and another one to calculate the divisor (the sum of the standard deviations).

User class:



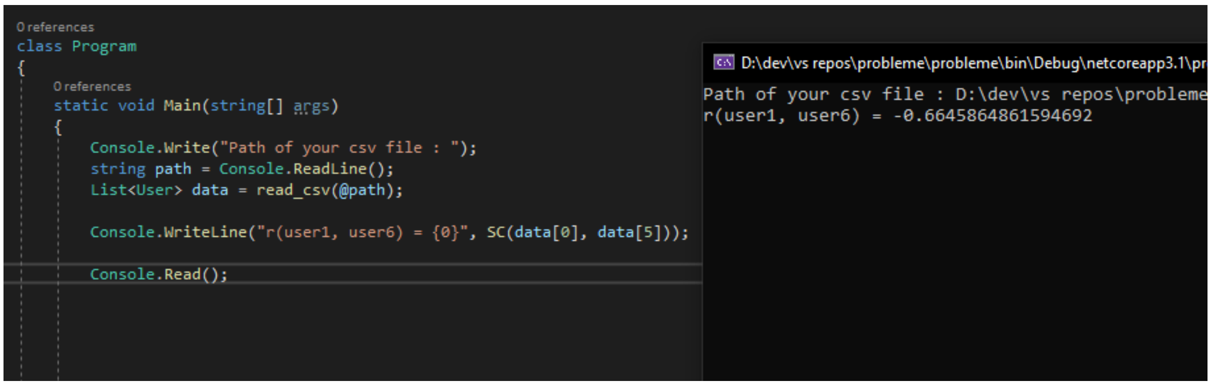
```
14 references
public class User
{
    17 references
    public string[] products { get; set; }
    6 references
    public double getMean()
    {
        double sum = 0;
        int count = 0;
        foreach (var i in products)
        {
            if (i.Length > 0)
            {
                sum += Convert.ToInt32(i);
                count++;
            }
        }
        return sum / count;
    }
}
```

Correlation coefficient implementation:



```
// r
3 references
static double SC(User u, User v)
{
    double c = cov(u, v);
    double va = total_var(u, v);
    return c/va;
}

// devidend
1 reference
static double cov(User u, User v)
{
    double cov = 0;
    for (int i = 0; i < u.products.Length; i++)
    {
        if (u.products[i].Length>0 && v.products[i].Length>0)
        {
            cov += (Convert.ToInt32(u.products[i]) - u.getMean()) * (Convert.ToInt32(v.products[i]) - v.getMean());
        }
    }
    return cov;
}

// divisor
1 reference
static double total_var(User u, User v)
{
    double vi = 0;
    double vj = 0;
    for (int i = 0; i < u.products.Length; i++)
    {
        if (u.products[i].Length > 0 && v.products[i].Length > 0)
        {
            vi += Math.Pow((Convert.ToInt32(u.products[i]) - u.getMean()), 2);
            vj += Math.Pow((Convert.ToInt32(v.products[i]) - v.getMean()), 2);
        }
    }
    return Math.Sqrt(vi) * Math.Sqrt(vj);
}
```

Using this function, we can now measure how similar two users are, lets take for example User1 and User6, we can tell by just looking at the data that these two users are very different.


```
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        Console.Write("Path of your csv file : ");
        string path = Console.ReadLine();
        List<User> data = read_csv(@path);

        Console.WriteLine("r(user1, user6) = {0}", SC(data[0], data[5]));

        Console.Read();
```

```
D:\dev\vs repos\probleme\probleme\bin\Debug\netcoreapp3.1\pr
Path of your csv file : D:\dev\vs repos\probleme
r(user1, user6) = -0.6645864861594692
```

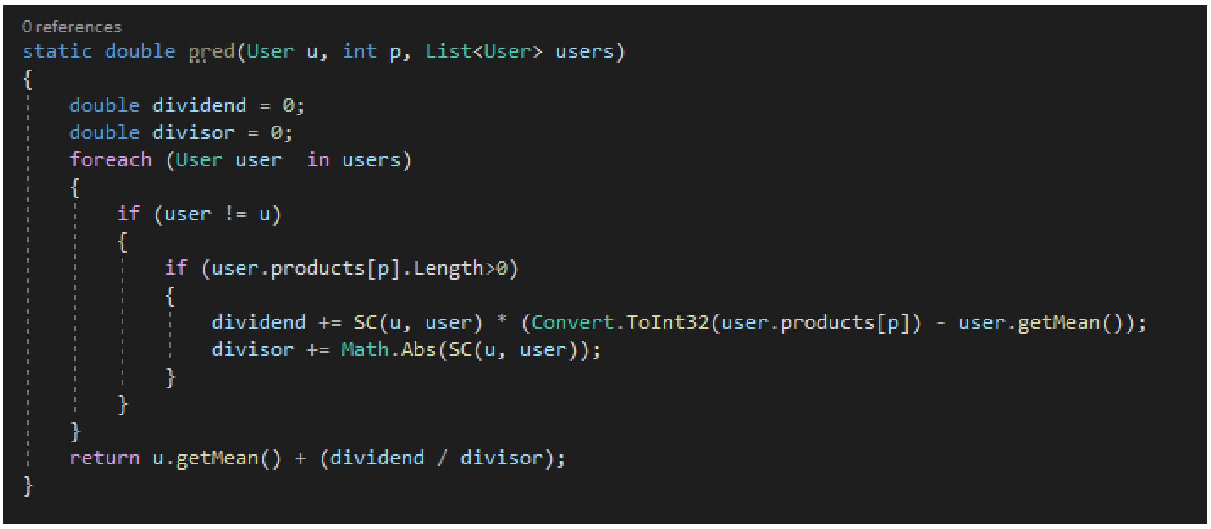
As you can see, we got r ~= -0.65 which means that there is a negative correlation between these two variables.

The recommendation part:

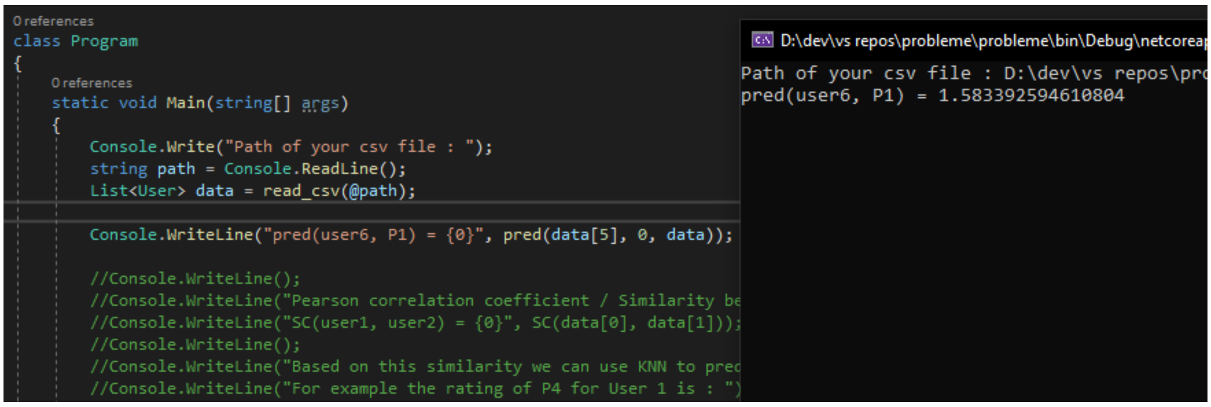
Now to make our prediction we will apply this formula to retain the k nearest neighbours within the meaning of the similarity SC and calculate a weighted average of their evaluations.

$$P_{u,j} = \bar{r}_u + \frac{\sum_{v=1}^{k} SC(u,v)(r_{v,j} - \bar{r}_v)}{\sum_{v=1}^{k} |SC(u,v)|}$$

Implementation in c#:


```
0 references
static double pred(User u, int p, List<User> users)
{
    double dividend = 0;
    double divisor = 0;
    foreach (User user  in users)
    {
        if (user != u)
        {
            if (user.products[p].Length>0)
            {
                dividend += SC(u, user) * (Convert.ToInt32(user.products[p]) - user.getMean());
                divisor += Math.Abs(SC(u, user));
            }
        }
    }
    return u.getMean() + (dividend / divisor);
}
```

Now we can use this function to make our prediction, let's take for example user6's rating for P1, we know that user 6 has a negative correlation with user1 and knowing that user1 gave a rating of 7 to P1 we are expecting a lower rating than that.

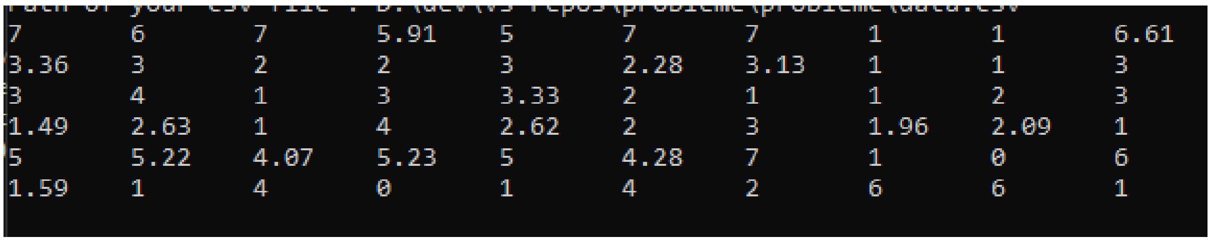


```
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        Console.Write("Path of your csv file : ");
        string path = Console.ReadLine();
        List<User> data = read_csv(@path);

        Console.WriteLine("pred(user6, P1) = {0}", pred(data[5], 0, data));

        //Console.WriteLine();
        //Console.WriteLine("Pearson correlation coefficient / Similarity be
        //Console.WriteLine("SC(user1, user2) = {0}", SC(data[0], data[1]));
        //Console.WriteLine();
        //Console.WriteLine("Based on this similarity we can use KNN to pred
        //Console.WriteLine("For example the rating of P4 for User 1 is : ")
```

```
D:\dev\vs repos\probleme\probleme\bin\Debug\netcorea
Path of your csv file : D:\dev\vs repos\pro
pred(user6, P1) = 1.583392594610804
```

We got a prediction of 1.59, which makes sense based on the corelation between user6 and user1.

Now we apply this method to predict all the missing values in the first table, and this is our result.



```
Path of your csv file : D:\dev\vs repos\probleme\probleme\data.csv
7       6       7       5.91    5       7       7       1       1       6.61
3.36    3       2       2       3       2.28    3.13    1       1       3
3       4       1       3       3.33    2       1       1       2       3
1.49    2.63    1       4       2.62    2       3       1.96    2.09    1
5       5.22    4.07    5.23    5       4.28    7       1       0       6
1.59    1       4       0       1       4       2       6       6       1
```

Full code here

Motassim Othman
IIR-4
G2