Randomisation:

```
// random.cpp : définit le point d'entrée pour l'application console.
//

#include "stdafx.h"

int main(int argc, char* argv[])
{

      //Create an array to put all values
      map <int,vector <double>> tabdata;

      if (argv[1] != NULL)
    {
            //File name to convert
            string filename = argv[1];
            cout << "File name to convert : " << filename << endl;

            //Open file
            ifstream file(filename.c_str());
            //Test if the file exist
            if (!file)
            {
                  cout << "File don't exist" << endl;
          }
            else
            {
                  //Tempory string
                  string str_temp = "";
                  int rec = 0;
                  while (getline(file,str_temp))
                  {
                        for (int i = 0 ; i < str_temp.length() ; i++)
                        {
                              int j = 0;
                              while ((str_temp[i+j] != ' ')&&((i+j) !=
(str_temp.length()-1)))
                              {
                                    j++;
                              }
                              string str_temp2 = str_temp.substr(i,(j+1));

      tabdata[rec].push_back(atof(str_temp2.c_str()));
                              if ( (i+j) < str_temp.length())
                              {
                                    i = i + j;
                              }
                              else
                              {
                                    break;
                              }
                        }
                        rec++;
                  }
                  file.close();

                  //Map the class field between from 0 and 1 to 0 and 10
                  for (int i = 0 ; i < tabdata.size() ; i++)
```

```
		{
			int nb_class_field = tabdata[i].size();
			double value = tabdata[i][nb_class_field - 1];

			if((value >= 0.0) && (value <=
0.1))tabdata[i][nb_class_field - 1] = 0;
			else if((value > 0.1)&&(value <=
0.2))tabdata[i][nb_class_field - 1] = 1;
			else if((value > 0.2)&&(value <=
0.3))tabdata[i][nb_class_field - 1] = 2;
			else if((value > 0.3)&&(value <=
0.4))tabdata[i][nb_class_field - 1] = 3;
			else if((value > 0.4)&&(value <=
0.5))tabdata[i][nb_class_field - 1] = 4;
			else if((value > 0.5)&&(value <=
0.6))tabdata[i][nb_class_field - 1] = 5;
			else if((value > 0.6)&&(value <=
0.7))tabdata[i][nb_class_field - 1] = 6;
			else if((value > 0.7)&&(value <=
0.8))tabdata[i][nb_class_field - 1] = 7;
			else if((value > 0.8)&&(value <=
0.9))tabdata[i][nb_class_field - 1] = 8;
			else if((value > 0.9)&&(value <=
1.0))tabdata[i][nb_class_field - 1] = 9;
			else tabdata[i][nb_class_field - 1] = 0;
		}

		//Randomization
		//Compute vector with size of the number of records
		vector <int> rand_number(tabdata.size());
		for(int i = 0; i < rand_number.size(); i ++)
		{
			rand_number[i] = i;
		}
		random_shuffle( rand_number.begin(),rand_number.end());

		/*for(int i = 0; i < rand_number.size(); i ++)
		{
			cout << rand_number[i] << ' ';
		}*/

		//Create an array to put all values
		map <int,vector <double>> tabdatarand;

		for(int i = 0; i < tabdata.size(); i++)
		{
			for (int j = 0; j < tabdata[i].size() ; j++)
			{

	tabdatarand[i].push_back(tabdata[rand_number[i]][j]);
			}
		}

		//Create a file
		cout << "Randomization" << endl;
		ofstream fileoutput("communities_random.data");
		for (int i = 0 ; i < tabdatarand.size() ; i++)
		{
			for (int j = 0 ; j < tabdatarand[i].size() ; j++)
			{
				fileoutput << tabdatarand[i][j];
```

```
                                fileoutput << ' ';
                        }
                        fileoutput << '\n';
                }
                fileoutput.close();
        }
    }
    system("PAUSE");
    return 0;
}
```

Class for correlation, mean and standard deviation:

```
#pragma once
#include <vector>
#include <math.h>

class mathdmml
{
public:
    mathdmml(void);
    ~mathdmml(void);

    double mean(vector <double> field);
    double std(vector <double> field);
    double correlation(vector <double> field1, vector <double> field2);
};
```

```
#include "StdAfx.h"
#include "mathdmml.h"

mathdmml::mathdmml(void)
{
}

mathdmml::~mathdmml(void)
{
}

double mathdmml::mean(std::vector<double> field)
{
    double result = 0;

    for(int i = 0 ; i < field.size() ; i ++)
    {
        result += field[i];
    }

    result /= field.size();

    return result;
}
```

```
double mathdmml::std(std::vector<double> field)
{
	double result = 0;

	//Compute the mean for this field
	double meanval = mean(field);

	for (int i = 0 ; i < field.size() ; i++)
	{
		result += (field[i] - meanval)*(field[i] - meanval);
	}

	result /= field.size();
	result = sqrt(result);

	return result;
}

double mathdmml::correlation(std::vector<double> field1,
std::vector<double> field2)
{
	double result = 0;

	//Compute the mean for each field
	double mean_f1 = mean(field1);
	//cout << "mean f1 = " << mean_f1 << endl;
	double mean_f2 = mean(field2);
	//cout << "mean f2 = " << mean_f2 << endl;

	//Compute the standard deviation for each field
	double std_f1 = std(field1);
	//cout << "std f1 = " << std_f1 << endl;
	double std_f2 = std(field2);
	//cout << "std f2 = " << std_f2 << endl;

	//Compute the correlation
	for (int i = 0 ; i < field1.size() ; i++)
	{
		//cout << "value field1 = " << field1[i] << endl;
		//cout << "value field2 = " << field2[i] << endl;
		result += ((field1[i] - mean_f1)/std_f1)*((field2[i] -
mean_f2)/std_f2);
	}

	//cout << "result inter = " << result << endl;

	result /= field1.size();

	//cout << "result correlation = " << result << endl;

	return abs(result);
}
```

Selection feature top fields:

```
// random.cpp : définit le point d'entrée pour l'application console.
//
```

```
#include "stdafx.h"
#include "mathdmml.h"

int main(int argc, char* argv[])
{

      //Create an array to put all values
      map <int,vector <double>> tabdata;

      if (argv[1] != NULL)
    {
            //File name to convert
            string filename = argv[1];
            cout << "File name to convert : " << filename << endl;

            //Open file
            ifstream file(filename.c_str());
            //Test if the file exist
            if (!file)
            {
                  cout << "File don't exist" << endl;
          }
            else
            {
                  //Tempory string
                  string str_temp = "";
                  int rec = 0;
                  while (getline(file,str_temp))
                  {
                        for (int i = 0 ; i < str_temp.length() ; i++)
                        {
                              int j = 0;
                              while ((str_temp[i+j] != ' ')&&((i+j) !=
(str_temp.length()-1)))
                              {
                                    j++;
                              }
                              string str_temp2 = str_temp.substr(i,(j+1));

      tabdata[rec].push_back(atof(str_temp2.c_str()));
                              if ( (i+j) < str_temp.length())
                              {
                                    i = i + j;
                              }
                              else
                              {
                                    break;
                              }
                        }
                        rec++;
                  }
                  file.close();

                  //Compute the correlation between each field and class
field

                  //Create element to compute correlation
                  mathdmml myMath;

                  //Create vector to store correlation value
```

```
                  vector <double> corre;

                  //Create tempory vector to put both field
                  vector <double> field;
                  vector <double> field_class;

                  //Compute the number of field
                  int nb_fields = tabdata[0].size();

                  //Create class vector
                  for (int i = 0 ; i < tabdata.size() ; i ++)
                  {
                        field_class.push_back(tabdata[i][nb_fields-1]);
                  }

                  for (int i = 0 ; i < (nb_fields-1) ; i ++)
                  {
                        //Create the field to compute the correlation
                        for (int j = 0 ; j < tabdata.size() ; j++)
                        {
                              field.push_back(tabdata[j][i]);
                        }


      corre.push_back(myMath.correlation(field,field_class));

                        field.clear();
                  }

                  //Create a copy of correlation vector
                  vector <double> ordercorre(corre);

                  //Order the previous corre
                  sort(ordercorre.begin(),ordercorre.end());

                  for (int i = 0 ; i < ordercorre.size() ; i++)cout <<
ordercorre[i] << endl;

                  //Find five best fields
                  vector <int> top5(5);

                  for (int i = 0 ; i < 5 ; i ++)
                  {
                        for(int j = 0 ; j < corre.size() ; j++)
                        {
                              if (ordercorre[ordercorre.size()- i - 1] ==
corre[j])
                              {
                                    top5[i] = j;
                                    break;
                              }
                        }
                  }

                  //Find ten best fields
                  vector <int> top10(10);

                  for (int i = 0 ; i < 10 ; i ++)
                  {
                        for(int j = 0 ; j < corre.size() ; j++)
                        {
```

```
                        if (ordercorre[ordercorre.size()- i - 1] ==
corre[j])
                        {
                              top10[i] = j;
                              break;
                        }
                  }
            }

            //Find ten best fields
            vector <int> top20(20);

            for (int i = 0 ; i < 20 ; i ++)
            {
                  for(int j = 0 ; j < corre.size() ; j++)
                  {
                        if (ordercorre[ordercorre.size()- i - 1] ==
corre[j])
                        {
                              top20[i] = j;
                              break;
                        }
                  }
            }

            ofstream fileoutput10("essai.data");
            for(int i = 0 ; i < top20.size() ; i++)
            {
                  fileoutput10 <<top20[i];
                  fileoutput10 << endl;
            }
            fileoutput10.close();

            //Create data with 5 fields
            map <int,vector <double>> tabdata5;

            for(int i = 0 ; i < tabdata.size() ; i++)
            {
                  for (int j = 0 ; j < 5 ; j++)
                  {
                        tabdata5[i].push_back(tabdata[i][top5[j]]);
                  }
                  tabdata5[i].push_back(tabdata[i][tabdata[i].size()
- 1]);
            }

            //Create data with 10 fields
            map <int,vector <double>> tabdata10;

            for(int i = 0 ; i < tabdata.size() ; i++)
            {
                  for (int j = 0 ; j < 10 ; j++)
                  {
                        tabdata10[i].push_back(tabdata[i][top10[j]]);
                  }
                  tabdata10[i].push_back(tabdata[i][tabdata[i].size()
- 1]);
            }
            //Create data with 20 fields
            map <int,vector <double>> tabdata20;
```

```
                    for(int i = 0 ; i < tabdata.size() ; i++)
                    {
                          for (int j = 0 ; j < 20 ; j++)
                          {
                                tabdata20[i].push_back(tabdata[i][top20[j]]);
                          }
                          tabdata20[i].push_back(tabdata[i][tabdata[i].size()
- 1]);
                    }

                    //Create a file
                    cout << "top5" << endl;
                    ofstream fileoutput("top5.data");
                    for (int i = 0 ; i < tabdata5.size() ; i++)
                    {
                          for (int j = 0 ; j < tabdata5[i].size() ; j++)
                          {
                                fileoutput << tabdata5[i][j];
                                fileoutput << ' ';
                          }
                          fileoutput << '\n';
                    }
                    fileoutput.close();

                    //Create a file
                    cout << "top10" << endl;
                    ofstream fileoutput2("top10.data");
                    for (int i = 0 ; i < tabdata10.size() ; i++)
                    {
                          for (int j = 0 ; j < tabdata10[i].size() ; j++)
                          {
                                fileoutput2 << tabdata10[i][j];
                                fileoutput2 << ' ';
                          }
                          fileoutput2 << '\n';
                    }
                    fileoutput2.close();

                    //Create a file
                    cout << "top20" << endl;
                    ofstream fileoutput3("top20.data");
                    for (int i = 0 ; i < tabdata20.size() ; i++)
                    {
                          for (int j = 0 ; j < tabdata20[i].size() ; j++)
                          {
                                fileoutput3 << tabdata20[i][j];
                                fileoutput3 << ' ';
                          }
                          fileoutput3 << '\n';
                    }
                    fileoutput3.close();

                    //Create a file
                    cout << "Correlation" << endl;
                    ofstream fileoutput4("communities_correlation.data");
                    for (int i = 0 ; i < corre.size() ; i++)
                    {
                          fileoutput4 << i ;
                          fileoutput4 << ' ';
                          fileoutput4 << corre[i];
                          fileoutput4 << '\n';
```

```
                }
                fileoutput4.close();
            }
      }

      system("PAUSE");
      return 0;
}
```

Reduction dataset with correlation:

```
// random.cpp : définit le point d'entrée pour l'application console.
//

#include "stdafx.h"
#include "mathdmml.h"

int main(int argc, char* argv[])
{

      //Create an array to put all values
      map <int,vector <double>> tabdata;

      if (argv[1] != NULL)
    {
            //File name to convert
            string filename = argv[1];
            cout << "File name to convert : " << filename << endl;

            //Open file
            ifstream file(filename.c_str());
            //Test if the file exist
            if (!file)
            {
                  cout << "File don't exist" << endl;
          }
            else
            {
                  //Tempory string
                  string str_temp = "";
                  int rec = 0;
                  while (getline(file,str_temp))
                  {
                        for (int i = 0 ; i < str_temp.length() ; i++)
                        {
                              int j = 0;
                              while ((str_temp[i+j] != ' ')&&((i+j) !=
(str_temp.length()-1)))
                              {
                                    j++;
                              }
                              string str_temp2 = str_temp.substr(i,(j+1));

      tabdata[rec].push_back(atof(str_temp2.c_str()));
                              if ( (i+j) < str_temp.length())
                              {
                                    i = i + j;
```

```
                        }
                        else
                        {
                              break;
                        }
                  }
                  rec++;
            }
            file.close();

            //Create element to compute correlation
            mathdmml myMath;

            //Create tempory vector to put both field
            vector <double> field1;
            vector <double> field2;

            //Compute the number of fields
            double nb_fields = tabdata[0].size() - 2;

            //Variable to store correlation
            double corre = 0;

            //Compute correlation between each field
            while (nb_fields > 4)
            {
                  cout << "Number of fields = " << nb_fields << endl;
                  int index = 0;
                  double maximum = 0;
                  for (int i = 0 ; i < nb_fields ; i++)
                  {
                        for (int j = 0 ; j < tabdata.size() ; j++)
                        {
                              field1.push_back(tabdata[j][i]);
                        }
                        //cout << "First field computed" << endl;
                        for (int j = i+1 ; j < nb_fields ; j ++)
                        {
                              for (int k = 0 ; k < tabdata.size() ;
k++)
                              {
                                    field2.push_back(tabdata[k][j]);
                              }
                              //cout << "Second field computed" <<
endl;
                              corre =
myMath.correlation(field1,field2);
                              cout << "Correlation computed between
field " << i << " and field " << j << " = " << corre << endl;
                              if (maximum < corre)
                              {
                                    maximum = corre;
                                    index = i;
                              }
                              field2.clear();
                        }
                        field1.clear();
                  }
                  cout << "Correlation value = " << maximum << endl;
                  cout << "Remove field " << index << endl;
                  for (int i = 0 ; i < tabdata.size() ; i++)
```

```
                        {
                              tabdata[i].erase(tabdata[i].begin() + index);
                        }
                        nb_fields = tabdata[0].size() - 2;
                  }

                  //Create a file
                  cout << "new" << endl;
                  ofstream fileoutput("new.data");
                  for (int i = 0 ; i < tabdata.size() ; i++)
                  {
                        for (int j = 0 ; j < tabdata[i].size() ; j++)
                        {
                              fileoutput << tabdata[i][j];
                              fileoutput << ' ';
                        }
                        fileoutput << '\n';
                  }
                  fileoutput.close();
            }
      }
      system("PAUSE");
      return 0;
}
```