



**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук  
Департамент программной инженерии

**СОГЛАСОВАНО**  
Научный руководитель,  
старший преподаватель департамента  
программной инженерии факультета  
компьютерных наук

  
\_\_\_\_\_ А.В. Меликян  
«22» \_\_\_\_\_ 2019 г.

**УТВЕРЖДАЮ**  
Академический руководитель  
образовательной программы  
«Программная инженерия»  
профессор департамента программной  
инженерии, канд. техн. наук

  
\_\_\_\_\_ В.В. Шилов  
«22» \_\_\_\_\_ 2019 г.


Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл	

**ПРОГРАММА ДЛЯ КЛАСТЕРИЗАЦИИ РОССИЙСКИХ ВУЗОВ ПО  
ПОКАЗАТЕЛЯМ ИХ НАУЧНО-ОБРАЗОВАТЕЛЬНОЙ ДЕЯТЕЛЬНОСТИ  
НА ОСНОВЕ ИЕРАРХИЧЕСКОГО АГЛОМЕРАТИВНОГО МЕТОДА**

**Текст программы**

**ЛИСТ УТВЕРЖДЕНИЯ**

**RU.17701729.04.13-01 12 01-1-ЛУ**

Исполнитель  
студент группы БПИ181  
/А.А. Матевосян /  
  
«22» \_\_\_\_\_ 2019 г.

**Москва 2019**

УТВЕРЖДЕН  
RU.17701729.04.13-01 12 01-1-ЛУ

**ПРОГРАММА ДЛЯ КЛАСТЕРИЗАЦИИ РОССИЙСКИХ ВУЗОВ ПО  
ПОКАЗАТЕЛЯМ ИХ НАУЧНО-ОБРАЗОВАТЕЛЬНОЙ ДЕЯТЕЛЬНОСТИ  
НА ОСНОВЕ ИЕРАРХИЧЕСКОГО АГЛОМЕРАТИВНОГО МЕТОДА**

**Текст программы**

**RU.17701729.04.13-01 12 01-1**

**Листов 64**

<i>Подп. и дата</i>	
<i>Инв. № дубл.</i>	
<i>Взам. инв. №</i>	
<i>Подп. и дата</i>	
<i>Инв. № подл</i>	

**Москва 2019**

## АННОТАЦИЯ

В данном программном документе приведен текст «Программы для кластеризации российских вузов по показателям их научно-образовательной деятельности на основе иерархического агломеративного метода».

Текст программы реализован в виде символической записи на исходном языке. Исходным языком данной разработки является C# 7.0. Среда разработки - Microsoft Visual Studio 2017 Community.

Данная программа является инструментом для кластерного анализа данных используя агломеративный метод иерархической кластеризации:

Настоящий документ разработан в соответствии с требованиями:

- 1) ГОСТ 19.101-77 Виды программ и программных документов [1];
- 2) ГОСТ 19.102-77 Стадии разработки [2];
- 3) ГОСТ 19.103-77 Обозначения программ и программных документов [3];
- 4) ГОСТ 19.104-78 Основные надписи [4];
- 5) ГОСТ 19.105-78 Общие требования к программным документам [5];
- 6) ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом [6];
- 7) ГОСТ 19.401-78 Текст программы. Требования к содержанию и оформлению [7].

Изменения к данному Руководству оператора оформляются согласно ГОСТ 19.604-78 [8], ГОСТ 19.603-78 [9].

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

## СОДЕРЖАНИЕ

1.	ТЕКСТ ПРОГРАММЫ.....	4
1.1.	AboutBox.cs .....	4
1.2.	Agnes.cs .....	6
1.3.	Cluster.cs.....	9
1.4.	ClusterDistance.cs.....	12
1.5.	ClusterizeForm .....	14
1.6.	ClusterPair.cs.....	17
1.7.	ClusterSet.cs .....	19
1.8.	Configuration.cs .....	21
1.9.	CSVData.cs .....	22
1.10.	CSVRow.cs .....	27
1.11.	CustomException.cs.....	28
1.12.	Datapoint.cs.....	29
1.13.	DendogramForm.cs.....	30
1.14.	DissimilarityMatrix.cs.....	37
1.15.	Distance.cs.....	38
1.16.	MainForm.cs.....	39
1.17.	Node.cs.....	52
1.18.	Program.cs.....	53
1.19.	StatisticsForm.cs.....	54
1.20.	Tools.cs .....	57
2.	СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ.....	62
	ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ.....	63

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

## 1. ТЕКСТ ПРОГРАММЫ

### 1.1. AboutBox.cs

```

1. using System;
2. using System.Collections.Generic;
3. using System.ComponentModel;
4. using System.Drawing;
5. using System.Linq;
6. using System.Reflection;
7. using System.Threading.Tasks;
8. using System.Windows.Forms;
9.
10. namespace Clusterizer
11. {
12.     /// <summary>
13.     /// About Box of Programm
14.     /// </summary>
15.     /// <seealso cref="System.Windows.Forms.Form" />
16.     public partial class AboutBox : Form
17.     {
18.         /// <summary>
19.         /// Initializes a new instance of the <see cref="AboutBox"/> class.
20.         /// </summary>
21.         public AboutBox()
22.         {
23.             InitializeComponent();
24.             Text = $"About {AssemblyTitle}";
25.             labelProductName.Text = AssemblyProduct;
26.             labelVersion.Text = $"Version {AssemblyVersion}";
27.             labelCopyright.Text = AssemblyCopyright;
28.             labelCompanyName.Text = AssemblyCompany;
29.             textBoxDescription.Text = AssemblyDescription;
30.         }
31.
32.         #region Assembly Attribute Accessors
33.
34.         public string AssemblyTitle
35.         {
36.             get
37.             {
38.                 object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(
39.                     typeof(AssemblyTitleAttribute), false);
40.                 if (attributes.Length > 0)
41.                 {
42.                     AssemblyTitleAttribute titleAttribute = (AssemblyTitleAttribute)att
43.                         rIBUTES[0];
44.                     if (titleAttribute.Title != "")
45.                     {
46.                         return titleAttribute.Title;
47.                     }
48.                     return System.IO.Path.GetFileNameWithoutExtension(Assembly.GetExecuting
49.                         Assembly().CodeBase);
50.                 }
51.             }
52.
53.             public string AssemblyVersion => Assembly.GetExecutingAssembly().GetName().Vers
54.                 ion.ToString();
55.
56.             public string AssemblyDescription
57.             {
58.                 get

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

56.         {
57.             object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes
es(typeof(AssemblyDescriptionAttribute), false);
58.             if (attributes.Length == 0)
59.             {
60.                 return "";
61.             }
62.             return ((AssemblyDescriptionAttribute)attributes[0]).Description;
63.         }
64.     }
65.
66.     public string AssemblyProduct
67.     {
68.         get
69.         {
70.             object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes
es(typeof(AssemblyProductAttribute), false);
71.             if (attributes.Length == 0)
72.             {
73.                 return "";
74.             }
75.             return ((AssemblyProductAttribute)attributes[0]).Product;
76.         }
77.     }
78.
79.     public string AssemblyCopyright
80.     {
81.         get
82.         {
83.             object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes
es(typeof(AssemblyCopyrightAttribute), false);
84.             if (attributes.Length == 0)
85.             {
86.                 return "";
87.             }
88.             return ((AssemblyCopyrightAttribute)attributes[0]).Copyright;
89.         }
90.     }
91.
92.     public string AssemblyCompany
93.     {
94.         get
95.         {
96.             object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes
es(typeof(AssemblyCompanyAttribute), false);
97.             if (attributes.Length == 0)
98.             {
99.                 return "";
100.            }
101.            return ((AssemblyCompanyAttribute)attributes[0]).Company;
102.        }
103.    }
104.    #endregion
105. }
106. }

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

## 1.2. Agnes.cs

```

1. using System;
2. using System.Collections.Generic;
3. using System.Diagnostics;
4. using System.IO;
5. using System.Text;
6. using System.Threading.Tasks;
7.
8. namespace Clusterizer
9. {
10.     /// <summary>
11.     /// Class for AGNES Clustering
12.     /// </summary>
13.     public class Agnes
14.     {
15.
16.         #region Fields
17.         /// <summary>
18.         /// The clusters
19.         /// </summary>
20.         private readonly ClusterSet _clusters;
21.
22.         /// <summary>
23.         /// The dissimilarity matrix
24.         /// </summary>
25.         private DissimilarityMatrix _dissimilarityMatrix;
26.
27.         /// <summary>
28.         /// The distance metric
29.         /// </summary>
30.         private readonly DistanceMetric _distanceMetric;
31.
32.         /// <summary>
33.         /// The strategy
34.         /// </summary>
35.         private readonly MergeStrategy _strategy;
36.
37.         /// <summary>
38.         /// The initial number of clusters
39.         /// </summary>
40.         private readonly int _initialNumberOfClusters;
41.
42.         /// <summary>
43.         /// The list of indexes for selected CH value
44.         /// </summary>
45.         private readonly List<int> _chIndex;
46.
47.         /// <summary>
48.         /// The list of CH values
49.         /// </summary>
50.         private readonly List<double> _chValue;
51.         #endregion
52.
53.         #region Constructor
54.         /// <summary>
55.         /// Initializes a new instance of the <see cref="Agnes"/> class.
56.         /// </summary>
57.         /// <param name="clusters">The clusters.</param>
58.         /// <param name="distanceMetric">The distance metric.</param>
59.         /// <param name="strategy">The strategy.</param>
60.         public Agnes(ClusterSet clusters, DistanceMetric distanceMetric, MergeStrategy
            strategy)
61.         {
62.             _clusters = clusters;

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

63.         _initialNumberOfClusters = clusters.Count;
64.         _distanceMetric = distanceMetric;
65.         _strategy = strategy;
66.
67.         // creating initial dissimilarity matrix from _clusters
68.         BuildDissimilarityMatrix();
69.
70.         _chValue = new List<double>();
71.         _chIndex = new List<int>();
72.     }
73.     #endregion
74.
75.     #region Methods
76.
77.     /// <summary>
78.     /// Builds the dissimilarity matrix.
79.     /// </summary>
80.     private void BuildDissimilarityMatrix()
81.     {
82.         _dissimilarityMatrix = new DissimilarityMatrix();
83.
84.         for (int i = 0; i < _clusters.Count - 1; i++)
85.         {
86.             for (int j = i + 1; j < _clusters.Count; j++)
87.             {
88.                 var clusterPair = new ClusterPair(_clusters.GetCluster(i), _clusters.GetCluster(j));
89.
90.                 var distanceBetweenTwoClusters = ClusterDistance.ComputeDistance(clusterPair.Cluster1, clusterPair.Cluster2, _distanceMetric);
91.                 _dissimilarityMatrix.AddClusterPairAndDistance(clusterPair, distanceBetweenTwoClusters); // adds distance to matrix
92.             }
93.         }
94.     }
95.
96.     /// <summary>
97.     /// Updates the dissimilarity matrix after adding new cluster to matrix.
98.     /// </summary>
99.     /// <param name="newCluster">The new cluster.</param>
100.    private void _UpdateDissimilarityMatrix(Cluster newCluster)
101.    {
102.        for (int i = 0; i < _clusters.Count; i++)
103.        {
104.            // distance between newCluster and one of the _clusters one
105.            var distanceBetweenClusters = ClusterDistance.ComputeDistance(_clusters.GetCluster(i), newCluster, _dissimilarityMatrix, _strategy);
106.            _dissimilarityMatrix.AddClusterPairAndDistance(new ClusterPair(newCluster, _clusters.GetCluster(i)), distanceBetweenClusters); // adds distance to matrix
107.
108.            // removes old cluster distance because they are useless
109.            _dissimilarityMatrix.RemoveClusterPair(new ClusterPair(newCluster.GetSubCluster(0), _clusters.GetCluster(i)));
110.            _dissimilarityMatrix.RemoveClusterPair(new ClusterPair(newCluster.GetSubCluster(1), _clusters.GetCluster(i)));
111.        }
112.
113.        // remove the distance of the the initial cluster subclusters
114.        _dissimilarityMatrix.RemoveClusterPair(new ClusterPair(newCluster.GetSubCluster(0), newCluster.GetSubCluster(1)));
115.    }
116.
117.
118.    /// <summary>
119.    /// Builds the hierarchical clustering.

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата



```

120.          /// </summary>
121.          /// <param name="indexNewCluster">The index new cluster.</param>
122.          /// <param name="k">The k.</param>
123.          /// <param name="isWithIndex">if set to <c>true</c> [is with index].</pa
ram>
124.          private void BuildHierarchicalClustering(int indexNewCluster, int k, boo
l isWithIndex = false)
125.          {
126.              ClusterPair closestClusterPair = _dissimilarityMatrix.GetClosestClus
terPair(); // gets the clusterpair with minimal distance
127.
128.              // creates new cluster by merging clusters from closestClusterPair
129.              Cluster newCluster = new Cluster();
130.              newCluster.AddSubCluster(closestClusterPair.Cluster1);
131.              newCluster.AddSubCluster(closestClusterPair.Cluster2);
132.              newCluster.Id = indexNewCluster;
133.              newCluster.SetCentroid();
134.
135.              // removes cluster pair from _clusters
136.              _clusters.RemoveClusterPair(closestClusterPair);
137.              _updateDissimilarityMatrix(newCluster);
138.              // add new cluster to _clusters
139.              _clusters.AddCluster(newCluster);
140.
141.              if (isWithIndex) // checks is executed for calculating CH index
142.              {
143.                  _chValue.Add(GetCHIndex()); // adds index to array of CH values
144.
145.                  _chIndex.Add(_clusters.ClustersList.Count); // adds number of cl
usters for current CH value
146.              }
147.
148.              // exit point of algorithm (where _clusters count is equal to k)
149.              if (_clusters.Count > k)
150.                  BuildHierarchicalClustering(indexNewCluster + 1, k, isWithIndex)
151.              ;
152.          }
153.
154.          /// <summary>
155.          /// Executes the clustering.
156.          /// </summary>
157.          /// <param name="k">The k.</param>
158.          /// <param name="isWithIndex">if set to <c>true</c> [is with index].</pa
ram>
159.          /// <returns>Computed ClustersSet</returns>
160.          public ClusterSet ExecuteClustering(int k, bool isWithIndex = false)
161.          {
162.              BuildHierarchicalClustering(_clusters.Count, k, isWithIndex);
163.              return _clusters;
164.          }
165.
166.          /// <summary>
167.          /// Gets the index of the ch.
168.          /// </summary>
169.          /// <returns>Current CH Index</returns>
170.          private double GetCHIndex()
171.          {
172.              // merging all clusters into one
173.              Cluster overallCluster = new Cluster();
174.              _clusters.ClustersList.ForEach(c => overallCluster.AddSubCluster(c))
175.              ;
176.              overallCluster.SetCentroid();
177.
178.              int currentNumberOfClusters = _clusters.Count;
179.              if (_clusters.ClustersList.Count < 2) // CH can't be computed for on
e cluster

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

177.         return double.NaN;
178.
179.         double withinSumOfSquares = 0,
180.             betweenSumOfSquares = 0;
181.
182.         foreach (var cluster in _clusters.ClustersList)
183.         {
184.             // computes sum of squares within cluster
185.             withinSumOfSquares += cluster.GetSumOfSquaredError(_distanceMetric);
186.             // computes som of squares with overallcluster (outside of cluster)
187.             betweenSumOfSquares += Math.Pow(Distance.GetDistance(overallCluster.Centroid, cluster.Centroid, _distanceMetric), 2);
188.         }
189.
190.         // checks if withinSumOfSquares is less then epsilon (CH is NaN)
191.         // else returns CH using formula
192.         return Math.Abs(withinSumOfSquares) < double.Epsilon
193.             ? double.NaN
194.             : (betweenSumOfSquares / withinSumOfSquares / (currentNumberOfClusters - 1)) *
195.               (_initialNumberOfClusters - currentNumberOfClusters);
196.     }
197.
198.     /// <summary>
199.     /// Gets the recomend count of clusters by calculating local Max f.
200.     /// </summary>
201.     /// <returns></returns>
202.     public int GetRecommendedCountOfClusters()
203.     {
204.         int maxIndex = _initialNumberOfClusters - 1; // index of Local Max CH
205.         double maxCoeff = 0; // local Max CH
206.
207.         // finds local Max of CH Values
208.         for (int i = 1; i < _chValue.Count - 1; i++)
209.         {
210.             if (_chValue[i] > _chValue[i - 1] && _chValue[i] > _chValue[i + 1] && _chValue[i] > maxCoeff)
211.             {
212.                 maxCoeff = _chValue[i];
213.                 maxIndex = _chIndex[i];
214.             }
215.         }
216.
217.         return maxIndex;
218.     }
219.     #endregion
220. }
221. }

```

### 1.3. Cluster.cs

```

1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4.
5. namespace Clusterizer
6. {
7.     /// <summary>
8.     /// Data structure for presenting cluster
9.     /// </summary>
10.    public class Cluster

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

11.  {
12.      #region Fields
13.      /// <summary>
14.      /// Gets or sets the sub clusters.
15.      /// </summary>
16.      /// <value>
17.      /// The sub clusters.
18.      /// </value>
19.      public List<Cluster> SubClusters { get; set; }
20.
21.      /// <summary>
22.      /// Gets or sets the data points.
23.      /// </summary>
24.      /// <value>
25.      /// The data points.
26.      /// </value>
27.      public List<DataPoint> DataPoints { get; set; }
28.
29.      /// <summary>
30.      /// Gets or sets the centroid.
31.      /// </summary>
32.      /// <value>
33.      /// The centroid.
34.      /// </value>
35.      public DataPoint Centroid { get; set; }
36.      #endregion
37.
38.      #region Properties
39.      /// <summary>
40.      /// Gets or sets the identifier.
41.      /// </summary>
42.      /// <value>
43.      /// The identifier.
44.      /// </value>
45.      public int Id { get; set; }
46.
47.      /// <summary>
48.      /// Gets the quantity of data points.
49.      /// </summary>
50.      /// <value>
51.      /// The quantity of data points.
52.      /// </value>
53.      public int QuantityOfDataPoints => DataPoints.Count;
54.
55.
56.      /// <summary>
57.      /// Gets the quantity of sub clusters.
58.      /// </summary>
59.      /// <value>
60.      /// The quantity of sub clusters.
61.      /// </value>
62.      public int QuantityOfSubClusters => SubClusters.Count;
63.      #endregion
64.
65.
66.      #region Конструктор
67.      /// <summary>
68.      /// Initializes a new instance of the <see cref="Cluster"/> class.
69.      /// </summary>
70.      public Cluster()
71.      {
72.          DataPoints = new List<DataPoint>();
73.          SubClusters = new List<Cluster>();
74.      }
75.      #endregion
76.

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

77.      #region Methods
78.      /// <summary>
79.      /// Adds the data point.
80.      /// </summary>
81.      /// <param name="dataPoint">The data point.</param>
82.      public void AddDataPoint(DataPoint dataPoint)
83.      {
84.          DataPoints.Add(dataPoint);
85.      }
86.
87.      /// <summary>
88.      /// Adds the sub cluster.
89.      /// </summary>
90.      /// <param name="subCluster">The sub cluster.</param>
91.      public void AddSubCluster(Cluster subCluster)
92.      {
93.          SubClusters.Add(subCluster);
94.          DataPoints.AddRange(subCluster.DataPoints);
95.      }
96.
97.      /// <summary>
98.      /// Gets the sub cluster.
99.      /// </summary>
100.     /// <param name="index">The index.</param>
101.     /// <returns></returns>
102.     public Cluster GetSubCluster(int index)
103.     {
104.         return SubClusters.ElementAt(index);
105.     }
106.
107.     /// <summary>
108.     /// Sets the centroid of cluster.
109.     /// </summary>
110.     public void SetCentroid()
111.     {
112.         int dataPointsCount = DataPoints[0].Count;
113.         double[] tmpPoints = Enumerable.Repeat(0.0, dataPointsCount).ToArray
114.     (
115.         // sum all datapoints of cluster
116.         foreach (var dataPoint in DataPoints)
117.             for (int i = 0; i < dataPointsCount; i++)
118.                 tmpPoints[i] += dataPoint[i];
119.
120.         // get mean of datapoints
121.         for (int i = 0; i < dataPointsCount; i++)
122.             tmpPoints[i] /= QuantityOfDataPoints;
123.
124.         Centroid = new DataPoint(tmpPoints.ToList());
125.     }
126.
127.     /// <summary>
128.     /// Gets the sum of squared error.
129.     /// </summary>
130.     /// <param name="distanceMetric">The distance metric.</param>
131.     /// <returns>Sum of squared error of cluster</returns>
132.     public double GetSumOfSquaredError(DistanceMetric distanceMetric)
133.     {
134.         double squaredErrorSum = 0;
135.
136.         //distance of each element to clustercenter
137.         foreach (var pattern in DataPoints)
138.         {
139.             var distToCenter = Distance.GetDistance(Centroid, pattern, dista
140.         nceMetric);
141.             squaredErrorSum += Math.Pow(distToCenter, 2);

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

141.         }
142.         return squaredErrorSum;
143.
144.     }
145.     #endregion
146.
147.
148. }
149. }

```

#### 1.4. ClusterDistance.cs

```

1. namespace Clusterizer
2. {
3.     /// <summary>
4.     /// Static class for computing distance between clusters
5.     /// </summary>
6.     public static class ClusterDistance
7.     {
8.         /// <summary>
9.         /// Computes the distance between singleton clusters.
10.        /// </summary>
11.        /// <param name="cluster1">The cluster1.</param>
12.        /// <param name="cluster2">The cluster2.</param>
13.        /// <param name="distanceMetric">The distance metric.</param>
14.        /// <returns>Distance between singleton clusters</returns>
15.        public static double ComputeDistance(Cluster cluster1, Cluster cluster2, DistanceMetric distanceMetric)
16.        {
17.            double distance = 0;
18.
19.            // check if clusters are singleton
20.            if (cluster1.QuantityOfDataPoints == 1 && cluster2.QuantityOfDataPoints == 1)
21.                distance = Distance.GetDistance(cluster1.DataPoints[0], cluster2.DataPoints[0], distanceMetric);
22.
23.            return distance;
24.        }
25.
26.
27.        /// <summary>
28.        /// Computes the distance.
29.        /// </summary>
30.        /// <param name="cluster1">The cluster1.</param>
31.        /// <param name="cluster2">The cluster2.</param>
32.        /// <param name="dissimilarityMatrix">The dissimilarity matrix.</param>
33.        /// <param name="strategy">The strategy.</param>
34.        /// <returns>Distance between clusters</returns>
35.        public static double ComputeDistance(Cluster cluster1, Cluster cluster2, DissimilarityMatrix dissimilarityMatrix, MergeStrategy strategy)
36.        {
37.            double distance = 0;
38.            var distance1 = dissimilarityMatrix.ReturnClusterPairDistance(new ClusterPair(cluster1, cluster2.GetSubCluster(0)));
39.            var distance2 = dissimilarityMatrix.ReturnClusterPairDistance(new ClusterPair(cluster1, cluster2.GetSubCluster(1)));
40.
41.            // computes distance by using merge strategy
42.            switch (strategy)
43.            {
44.                case MergeStrategy.SingleLinkage:
45.                    distance = _MinValue(distance1, distance2); // Min(x, y)
46.                    break;

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

47.         case MergeStrategy.CompleteLinkage:
48.             distance = _MaxValue(distance1, distance2); // Max(x, y)
49.             break;
50.         case MergeStrategy.AverageLinkageWpgma:
51.             distance = (distance1 + distance2) / 2; // Avg(x, y)
52.             break;
53.         case MergeStrategy.AverageLinkageUpgma:
54.             distance = ((cluster2.GetSubCluster(0).QuantityOfDataPoints * dista
55. nce1) / cluster2.QuantityOfDataPoints)
56.             + ((cluster2.GetSubCluster(1).QuantityOfDataPoints * dis
57. tance2) / cluster2.QuantityOfDataPoints); // WeightedAvg(x, y)
58.             break;
59.         case MergeStrategy.CentroidMethod:
60.             cluster1.SetCentroid();
61.             cluster2.SetCentroid();
62.             distance = Distance.GetDistance(cluster1.Centroid, cluster2.Centroi
63. d,
64.         DistanceMetric.SquareEuclidianDistance); // Distance of centri
65. ds
66.             break;
67.         case MergeStrategy.WardsMethod:
68.
69.             Cluster newCluster = new Cluster();
70.             newCluster.AddSubCluster(cluster1);
71.             newCluster.AddSubCluster(cluster2);
72.             newCluster.SetCentroid();
73.
74.             distance = newCluster.GetSumOfSquaredError(DistanceMetric.Euclidian
75. Distance)
76.             -
77.             cluster1.GetSumOfSquaredError(DistanceMetric.EuclidianDistance)
78.             -
79.             cluster2.GetSumOfSquaredError(DistanceMetric.EuclidianDistance);
80.             // SE0(xy) - SE0(x) - SE0(y)
81.             break;
82.         }
83.     }
84.     return distance;
85. }
86.
87. /// <summary>
88. /// Calculate the minimal of the given values
89. /// </summary>
90. /// <param name="value1">The value1.</param>
91. /// <param name="value2">The value2.</param>
92. /// <returns>Minimum of the values</returns>
93. private static double _MinValue(double value1, double value2)
94. {
95.     return value1 < value2 ? value1 : value2;
96. }
97.
98. /// <summary>
99. /// Calculate the minimal of the given values
100. /// </summary>
101. /// <param name="value1">The value1.</param>
102. /// <param name="value2">The value2.</param>
103. /// <returns>Maximum of the values</returns>
104. private static double _MaxValue(double value1, double value2)
105. {
106.     return value1 > value2 ? value1 : value2;
107. }
108. }
109.
110. #region Merge Strategy
111. /// <summary>

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

106.      /// Enum of merge strategies
107.      /// </summary>
108.      public enum MergeStrategy
109.      {
110.          SingleLinkage, // Single Linkage
111.          CompleteLinkage, // Complete Linkage
112.          AveragelinkageWpgma, // Average Linkage (WPGMA)
113.          AveragelinkageUpgma, // Average Linkage (UPGMA)
114.          CentroidMethod, // Centroid Method
115.          WardsMethod // Wards Method
116.      }
117.      #endregion
118.  }

```

## 1.5. ClusterizeForm

```

1.  using System;
2.  using System.Collections.Generic;
3.  using System.ComponentModel;
4.  using System.Data;
5.  using System.Drawing;
6.  using System.Linq;
7.  using System.Text;
8.  using System.Threading.Tasks;
9.  using System.Windows.Forms;
10.
11. namespace Clusterizer
12. {
13.     /// <summary>
14.     /// Form for selecting parameters of clustering
15.     /// </summary>
16.     /// <seealso cref="System.Windows.Forms.Form" />
17.     public partial class ClusterizeForm : Form
18.     {
19.         #region Fields
20.
21.         /// <summary>
22.         /// The merge strategy
23.         /// </summary>
24.         internal MergeStrategy strategy;
25.
26.         /// <summary>
27.         /// The distance metric
28.         /// </summary>
29.         internal DistanceMetric distanceMetric;
30.
31.         /// <summary>
32.         /// The normalize method
33.         /// </summary>
34.         internal NormalizeMethod normalizeMethod;
35.
36.         /// <summary>
37.         /// The count of clusters
38.         /// </summary>
39.         internal int countOfClusters = 1;
40.
41.         /// <summary>
42.         /// Indicates if parameters selected
43.         /// </summary>
44.         internal bool isParametersSelected;
45.
46.         /// <summary>
47.         /// Indicates if TreeView busy
48.         /// </summary>

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

49.     private bool _isTreeViewBusy;
50.
51.     #endregion
52.
53.     #region Конструктор
54.
55.     /// <summary>
56.     /// Initializes a new instance of the <see cref="ClusterizeForm"/> class.
57.     /// </summary>
58.     public ClusterizeForm()
59.     {
60.         InitializeComponent();
61.
62.         isParametersSelected = false;
63.         pointsSelectTreeView.CheckBoxes = true;
64.
65.         // Loads Parameters from defined configuration
66.         for (int i = 0; i < Tools.GroupNames.Length; i++)
67.         {
68.             TreeNode rootNode = new TreeNode(Tools.GroupNames[i]) {Checked = true};
69.
70.             for (int j = 0; j < Tools.GroupItemsNames[i].Length; j++)
71.             {
72.                 TreeNode node = new TreeNode(Tools.GroupItemsNames[i][j]);
73.                 rootNode.Nodes.Add(node);
74.                 node.Checked = true;
75.             }
76.             pointsSelectTreeView.Nodes.Add(rootNode);
77.         }
78.
79.         distanceSelectComboBox.SelectedIndex = 0;
80.         strategySelectComboBox.SelectedIndex = 0;
81.         normalizeMethodSelectComboBox.SelectedIndex = 0;
82.     }
83.     #endregion
84.
85.     #region Events
86.     /// <summary>
87.     /// Handles the Click event of the doClusteringButton control.
88.     /// </summary>
89.     /// <param name="sender">The source of the event.</param>
90.     /// <param name="e">The <see cref="EventArgs"/> instance containing the event d
91.     ata.</param>
92.     /// <exception cref="Clusterizer.CustomException">
93.     /// Введите правильное количество кластеров. -
94.     Ошибка при вводе числа кластеров
95.     /// or
96.     /// Не было выбрано не одного показателя. - Ошибка при выборе показателей
97.     /// </exception>
98.     private void doClusteringButton_Click(object sender, EventArgs e)
99.     {
100.         // Gets selected parameters of clustering
101.         distanceMetric = (DistanceMetric) distanceSelectComboBox.SelectedIndex;
102.         strategy = (MergeStrategy) strategySelectComboBox.SelectedIndex;
103.         normalizeMethod = (NormalizeMethod) normalizeMethodSelectComboBox.Se
104.         lectedIndex;
105.
106.         // checks for correct cluster number
107.         if (int.TryParse(clusterCountTextBox.Text, out var tmp) && tmp > 0 &
108.             & tmp < Tools.Data.Rows.Count)
109.             countOfClusters = tmp;
110.         else
111.             throw new CustomException("Введите правильное количество кластеро
112.             в.", "Ошибка при вводе числа кластеров");
113.     }

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата



```

109.         // gets selected datapoints
110.         var isChosen = new bool[Tools.NumericDataHeadings.Length];
111.         bool isAllFalse = true;
112.         int ind = 0;
113.         for (int i = 0; i < pointsSelectTreeView.Nodes.Count; i++)
114.         {
115.             for (int j = 0; j < pointsSelectTreeView.Nodes[i].Nodes.Count; j
116.             ++
117.             {
118.                 isChosen[ind] = pointsSelectTreeView.Nodes[i].Nodes[j].Check
119.                 ed;
120.                 if (isAllFalse)
121.                     isAllFalse = !isChosen[ind];
122.                 ind++;
123.             }
124.         }
125.         // check if no datapoint is selected
126.         if (isAllFalse)
127.             throw new CustomException("Не было выбрано не одного показателя.
128. ", "Ошибка при выборе показателей");
129.         Tools.isChosen = isChosen;
130.         isParametersSelected = true;
131.         Close();
132.     }
133.     /// <summary>
134.     /// Handles the Click event of the calculateClusterCountButton control.
135.     /// </summary>
136.     /// <param name="sender">The source of the event.</param>
137.     /// <param name="e">The <see cref="EventArgs"/> instance containing the
138.     event data.</param>
139.     /// <exception cref="Clusterizer.CustomException">Не было выбрано ни одн
140.     ого показателя. - Ошибка при выборе показателей</exception>
141.     private void calculateClusterCountButton_Click(object sender, EventArgs
142.     e)
143.     {
144.         // gets selected parameters of clustering
145.         distanceMetric = (DistanceMetric)distanceSelectComboBox.SelectedIndex
146.         x;
147.         strategy = (MergeStrategy)strategySelectComboBox.SelectedIndex;
148.         normalizeMethod = (NormalizeMethod)normalizeMethodSelectComboBox.Sel
149.         ectedIndex;
150.         // gets selected datapoints
151.         var isChosen = new bool[Tools.NumericDataHeadings.Length];
152.         bool isAllFalse = true;
153.         int ind = 0;
154.         for (int i = 0; i < pointsSelectTreeView.Nodes.Count; i++)
155.         {
156.             for (int j = 0; j < pointsSelectTreeView.Nodes[i].Nodes.Count; j
157.             ++
158.             {
159.                 isChosen[ind] = pointsSelectTreeView.Nodes[i].Nodes[j].Check
160.                 ed;
161.                 if (isAllFalse)
162.                     isAllFalse = !isChosen[ind];
163.                 ind++;
164.             }
165.         }
166.         // check if no datapoint is selected
167.         if (isAllFalse)

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

163.         throw new CustomException("Не было выбрано ни одного показателя.
164.         ", "Ошибка при выборе показателей");
165.
166.         // gets cluster set from data
167.         var clusters = Tools.Data.GetClusterSet(Tools.isChosen);
168.         clusters.Normalize(normalizeMethod);
169.
170.         // executes clustering for determining recommended count of clusters
171.
172.         Agnes agnes = new Agnes(clusters,
173.             distanceMetric, strategy);
174.         agnes.ExecuteClustering(2, true);
175.
176.         // gets recommended count of clusters
177.         countOfClusters = agnes.GetRecommendedCountOfClusters();
178.         clusterCountTextBox.Text = $"{countOfClusters}";
179.     }
180.
181.     /// <summary>
182.     /// Handles the AfterCheck event of the pointsSelectTreeView control.
183.     /// </summary>
184.     /// <param name="sender">The source of the event.</param>
185.     /// <param name="e">The <see cref="TreeViewEventArgs"> instance contain
186.     ing the event data.</param>
187.     private void pointsSelectTreeView_AfterCheck(object sender, TreeViewEven
188.     tArgs e)
189.     {
190.         if (_isTreeViewBusy) return;
191.         _isTreeViewBusy = true;
192.         try
193.         {
194.             CheckNodes(e.Node, e.Node.Checked);
195.         }
196.         finally
197.         {
198.             _isTreeViewBusy = false;
199.         }
200.     }
201.     #endregion
202.
203.     #region Methods
204.     /// <summary>
205.     /// Checks the nodes.
206.     /// </summary>
207.     /// <param name="node">The node.</param>
208.     /// <param name="check">if set to <c>true</c> [check].</param>
209.     private void CheckNodes(TreeNode node, bool check)
210.     {
211.         foreach (TreeNode child in node.Nodes)
212.         {
213.             child.Checked = check;
214.             CheckNodes(child, check);
215.         }
216.     }
217.     #endregion
218. }

```

## 1.6. ClusterPair.cs

```

1. using System;
2. using System.Collections.Generic;
3.

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

4. namespace Clusterizer
5. {
6.     /// <summary>
7.     /// Data structure for presenting pair of clusters
8.     /// </summary>
9.     public class ClusterPair
10.    {
11.
12.        #region Constructor
13.        /// <summary>
14.        /// Initializes a new instance of the <see cref="ClusterPair"/> class.
15.        /// </summary>
16.        /// <param name="cluster1">The cluster1.</param>
17.        /// <param name="cluster2">The cluster2.</param>
18.        /// <exception cref="ArgumentNullException">
19.        /// cluster1
20.        /// or
21.        /// cluster2
22.        /// </exception>
23.        public ClusterPair(Cluster cluster1, Cluster cluster2)
24.        {
25.            Cluster1 = cluster1 ?? throw new ArgumentNullException(nameof(cluster1));
26.            Cluster2 = cluster2 ?? throw new ArgumentNullException(nameof(cluster2));
27.        }
28.        #endregion
29.
30.        #region Properties
31.        /// <summary>
32.        /// Gets or sets the cluster1.
33.        /// </summary>
34.        /// <value>
35.        /// The cluster1.
36.        /// </value>
37.        public Cluster Cluster1 { get; set; }
38.
39.        /// <summary>
40.        /// Gets or sets the cluster2.
41.        /// </summary>
42.        /// <value>
43.        /// The cluster2.
44.        /// </value>
45.        public Cluster Cluster2 { get; set; }
46.
47.        #endregion
48.
49.        #region EqualityComparer
50.        /// <summary>
51.        /// EqualityComparer for class ClusterPair
52.        /// </summary>
53.        /// <seealso cref="System.Collections.Generic.IEqualityComparer{Clusterizer.Clus
sterPair}" />
54.        public class EqualityComparer : IEqualityComparer<ClusterPair>
55.        {
56.            /// As ClusterPair is defined class, we need to specify its equality for usi
ng in Dictionary as keys
57.
58.            /// <summary>
59.            /// Determines whether the specified objects are equal.
60.            /// </summary>
61.            /// <param name="x">The first object of type <paramref name="T" /> to compa
re.</param>
62.            /// <param name="y">The second object of type <paramref name="T" /> to comp
are.</param>
63.            /// <returns>
64.            /// <see langword="true" /> if the specified objects are equal; otherwise
, <see langword="false" />.

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

65.         /// </returns>
66.     public bool Equals(ClusterPair x, ClusterPair y)
67.     {
68.         return x.Cluster1.Id == y.Cluster1.Id && x.Cluster2.Id == y.Cluster2.Id
69.     ;
69.     }
70.
71.     /// <summary>
72.     /// Returns a hash code for this instance.
73.     /// </summary>
74.     /// <param name="x">The x.</param>
75.     /// </returns>
76.     /// A hash code for this instance, suitable for use in hashing algorithms a
    nd data structures like a hash table.
77.     /// </returns>
78.     public int GetHashCode(ClusterPair x)
79.     {
80.         return x.Cluster1.Id ^ x.Cluster2.Id;
81.     }
82. }
83. #endregion
84.
85. }
86. }

```

## 1.7. ClusterSet.cs

```

1. using System;
2. using System.Collections;
3. using System.Collections.Generic;
4. using System.Linq;
5.
6. namespace Clusterizer
7. {
8.     /// <summary>
9.     /// Data structure of Cluster's set
10.    /// </summary>
11.    public class ClusterSet : IEnumerable
12.    {
13.        #region Properties
14.        /// <summary>
15.        /// Gets the count.
16.        /// </summary>
17.        /// <value>
18.        /// The count.
19.        /// </value>
20.        public int Count => ClustersList.Count;
21.
22.        /// <summary>
23.        /// Gets or sets the clusters list.
24.        /// </summary>
25.        /// <value>
26.        /// The clusters list.
27.        /// </value>
28.        public List<Cluster> ClustersList { get; set; }
29.        #endregion
30.
31.        #region Конструктор
32.        /// <summary>
33.        /// Initializes a new instance of the <see cref="ClusterSet"/> class.
34.        /// </summary>
35.        public ClusterSet()
36.        {
37.            ClustersList = new List<Cluster>();

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

38.     }
39.
40.     #endregion
41.
42.     #region Method
43.     /// <summary>
44.     /// Adds the cluster.
45.     /// </summary>
46.     /// <param name="cluster">The cluster.</param>
47.     public void AddCluster(Cluster cluster)
48.     {
49.         ClustersList.Add(cluster);
50.     }
51.
52.     /// <summary>
53.     /// Removes the cluster.
54.     /// </summary>
55.     /// <param name="cluster">The cluster.</param>
56.     public void RemoveCluster(Cluster cluster)
57.     {
58.         ClustersList.Remove(cluster);
59.     }
60.
61.     /// <summary>
62.     /// Gets the cluster.
63.     /// </summary>
64.     /// <param name="index">The index.</param>
65.     /// <returns></returns>
66.     public Cluster GetCluster(int index)
67.     {
68.         return ClustersList.ElementAt(index);
69.     }
70.
71.     /// <summary>
72.     /// Gets the <see cref="Cluster"/> at the specified index.
73.     /// </summary>
74.     /// <value>
75.     /// The <see cref="Cluster"/>.
76.     /// </value>
77.     /// <param name="index">The index.</param>
78.     /// <returns></returns>
79.     public Cluster this[int index] => ClustersList[index];
80.
81.     /// <summary>
82.     /// Removes the cluster pair.
83.     /// </summary>
84.     /// <param name="clusterPair">The cluster pair.</param>
85.     public void RemoveClusterPair(ClusterPair clusterPair)
86.     {
87.         RemoveCluster(clusterPair.Cluster1);
88.         RemoveCluster(clusterPair.Cluster2);
89.     }
90.
91.     /// <summary>
92.     /// Returns an enumerator that iterates through a collection.
93.     /// </summary>
94.     /// <returns>
95.     /// An <see cref="T:System.Collections.IEnumerator" /> object that can be used
    to iterate through the collection.
96.     /// </returns>
97.     public IEnumerator GetEnumerator()
98.     {
99.         return ClustersList.GetEnumerator();
100.    }
101.
102.    /// <summary>

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

103.          /// Normalizes cluster's datapoints using the specified normalize method
104.          /// </summary>
105.          /// <param name="normalizeMethod">The normalize method.</param>
106.          public void Normalize(NormalizeMethod normalizeMethod)
107.          {
108.              // if there is no normalization
109.              if (normalizeMethod == NormalizeMethod.None)
110.                  return;
111.
112.
113.              int pointCount = ClustersList[0].DataPoints[0].Count; // datapoints
count
114.              int clustersCount = ClustersList.Count; // clusters count
115.
116.              // gets all data subgrouped by their datapoints
117.              double[][] dataArray = new double[pointCount][];
118.
119.              for (int i = 0; i < pointCount; i++)
120.              {
121.                  dataArray[i] = new double[clustersCount];
122.                  for (int j = 0; j < clustersCount; j++)
123.                      dataArray[i][j] = ClustersList[j].DataPoints[0][i];
124.              }
125.
126.              // normalizes data
127.              if (normalizeMethod == NormalizeMethod.MinMax)
128.              {
129.                  for(int i = 0; i < pointCount; i++)
130.                      Tools.MinMaxNormalize(ref dataArray[i]);
131.              } else if (normalizeMethod == NormalizeMethod.ZScore)
132.              {
133.                  for (int i = 0; i < pointCount; i++)
134.                      Tools.ZScoreNormalize(ref dataArray[i]);
135.              }
136.
137.              // updates data with normalized one
138.              for(int i = 0; i < pointCount; i++) {
139.                  for (int j = 0; j < ClustersList.Count; j++)
140.                  {
141.                      ClustersList[j].DataPoints[0][i] = dataArray[i][j];
142.                  }
143.              }
144.          }
145.          #endregion
146.      }
147.
148.      /// <summary>
149.      /// Enum for Normalize Method
150.      /// </summary>
151.      public enum NormalizeMethod
152.      {
153.          None, // No Normalization
154.          MinMax, // Min-Max Normalization
155.          ZScore // Z-Score Normalization
156.      }
157.  }

```

## 1.8. Configuration.cs

```

1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4. using System.Text;

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

5. using System.Threading.Tasks;
6. using System.Xml.Serialization;
7.
8. namespace Clusterizer
9. {
10.     /// <summary>
11.     /// Data structure for storing data file configuration
12.     /// </summary>
13.     public class Configuration
14.     {
15.         /// <summary>
16.         /// The string headings
17.         /// </summary>
18.         public string[] StringHeadings;
19.         /// <summary>
20.         /// The numeric headings
21.         /// </summary>
22.         public string[] NumericHeadings;
23.         /// <summary>
24.         /// The group names
25.         /// </summary>
26.         public string[] GroupNames;
27.         /// <summary>
28.         /// The group items count
29.         /// </summary>
30.         public int[] GroupItemsCount;
31.     }
32. }

```

## 1.9. CSVData.cs

```

1. using System;
2. using System.Collections.Generic;
3. using System.Data;
4. using System.IO;
5. using System.Linq;
6.
7. namespace Clusterizer
8. {
9.     /// <summary>
10.    /// Class for working with input data
11.    /// </summary>
12.    public class CSVData
13.    {
14.        #region Свойства
15.
16.        /// <summary>
17.        /// Gets the fields count.
18.        /// </summary>
19.        /// <value>
20.        /// The fields count.
21.        /// </value>
22.        public int FieldsCount => StringHeadings.Length + NumericHeadings.Length;
23.
24.        /// <summary>
25.        /// Gets or sets the rows.
26.        /// </summary>
27.        /// <value>
28.        /// The rows.
29.        /// </value>
30.        public List<CSVRow> Rows { get; set; }
31.
32.        /// <summary>
33.        /// Строковые столбцы

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

34.      /// </summary>
35.      public string[] StringHeadings { get; set; }
36.
37.      /// <summary>
38.      /// Gets or sets the numeric headings.
39.      /// </summary>
40.      /// <value>
41.      /// The numeric headings.
42.      /// </value>
43.      public string[] NumericHeadings { get; set; }
44.
45.      /// <summary>
46.      /// The data set table
47.      /// </summary>
48.      public DataTable DataSetTable;
49.
50.      /// <summary>
51.      /// Gets or sets the file path.
52.      /// </summary>
53.      /// <value>
54.      /// The file path.
55.      /// </value>
56.      public string FilePath { get; set; }
57.
58.      /// <summary>
59.      /// The undo stack
60.      /// </summary>
61.      public Stack<List<CSVRow>> UndoStack;
62.
63.      /// <summary>
64.      /// The redo stack
65.      /// </summary>
66.      public Stack<List<CSVRow>> RedoStack;
67.
68.      /// <summary>
69.      /// The maximum stack count
70.      /// </summary>
71.      private const int MaxStackCount = 30;
72.
73.      #endregion
74.
75.      #region Constructor
76.      /// <summary>
77.      /// Initializes a new instance of the <see cref="CSVData"/> class.
78.      /// </summary>
79.      /// <param name="filePath">The file path.</param>
80.      public CSVData(string filePath)
81.      {
82.          // initializes main components
83.          FilePath = filePath;
84.          Rows = new List<CSVRow>();
85.          DataSetTable = new DataTable();
86.          UndoStack = new Stack<List<CSVRow>>();
87.          RedoStack = new Stack<List<CSVRow>>();
88.          StringHeadings = Tools.StringDataHeadings;
89.          NumericHeadings = Tools.NumericDataHeadings;
90.
91.          // load data from file
92.          using (var fileStream = new FileStream(filePath, FileMode.Open))
93.          {
94.              var streamReader = new StreamReader(fileStream);
95.              string line;
96.              // adds single line data to rows
97.              while ((line = streamReader.ReadLine()) != null && line != "") Rows.Add
98.              (GetRowFromLine(line));
99.          }

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата



```

99.         }
100.
101.         #endregion
102.
103.         #region Methods
104.         /// <summary>
105.         /// Generates cluster set using selected datapoints
106.         /// </summary>
107.         /// <param name="isChosen">Is datapoint is chosen</param>
108.         /// <returns></returns>
109.         public ClusterSet GetClusterSet(bool[] isChosen)
110.         {
111.             var clusterSet = new ClusterSet();
112.
113.             // generates datapoint
114.             for (var i = 0; i < Rows.Count; i++)
115.             {
116.                 var dataPoint = new DataPoint { Id = i };
117.                 // gets all points of datapoint
118.                 for (var j = StringHeadings.Length; j < FieldsCount; j++)
119.                     dataPoint.Add(double.Parse(Rows[i][j]));
120.
121.                 // new cluster
122.                 var cluster = new Cluster { Id = i };
123.                 cluster.AddDataPoint(dataPoint);
124.                 cluster.SetCentroid();
125.
126.                 // add to set
127.                 clusterSet.AddCluster(cluster);
128.             }
129.
130.             return clusterSet;
131.         }
132.
133.         /// <summary>
134.         /// Gets the row from line.
135.         /// </summary>
136.         /// <param name="line">The line.</param>
137.         /// <returns></returns>
138.         /// <exception cref="InvalidDataException"></exception>
139.         public CSVRow GetRowFromLine(string line)
140.         {
141.             // gets all fields from line
142.             var fields = line.Split(new[] { ';' }, StringSplitOptions.RemoveEmptyEntries)
143.                 .Select(x => x.Trim(' ', '\\'))
144.                 .ToList();
145.             // checks for correctness
146.             if (fields.Count != FieldsCount)
147.                 throw new InvalidDataException();
148.             var csvRow = new CSVRow(fields);
149.             return csvRow;
150.         }
151.
152.         /// <summary>
153.         /// Gets the CSV line from row.
154.         /// </summary>
155.         /// <param name="csvRow">The CSV row.</param>
156.         /// <returns></returns>
157.         public string GetCsvLineFromRow(CSVRow csvRow)
158.         {
159.             return string.Join(";", csvRow.Fields);
160.         }
161.
162.         /// <summary>
163.         /// Updates the rows.
164.         /// </summary>

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

164.     public void UpdateRows()
165.     {
166.         Rows.Clear();
167.         foreach (DataRow row in DataSetTable.Rows)
168.         {
169.             var fieldList = new List<string>();
170.             for (var i = 0; i < row.ItemArray.Length; i++)
171.                 fieldList.Add(row[i].ToString());
172.             Rows.Add(new CSVRow(fieldList));
173.         }
174.     }
175.
176.     /// <summary>
177.     /// Creates the data table columns.
178.     /// </summary>
179.     public void CreateDataTableColumns()
180.     {
181.         foreach (var heading in StringHeadings)
182.             DataSetTable.Columns.Add(heading, typeof(string));
183.
184.         foreach (var numericHeading in NumericHeadings)
185.             DataSetTable.Columns.Add(numericHeading, typeof(double));
186.     }
187.
188.     /// <summary>
189.     /// Creates the data table.
190.     /// </summary>
191.     public void CreateDataTable()
192.     {
193.         DataSetTable = new DataTable();
194.         CreateDataTableColumns();
195.         UpdateData();
196.     }
197.
198.     /// <summary>
199.     /// Updates the data.
200.     /// </summary>
201.     public void UpdateData()
202.     {
203.         DataSetTable.Clear();
204.         foreach (var row in Rows) DataSetTable.Rows.Add(row.Fields.ToArray()
205.     );
206.     }
207.
208.     /// <summary>
209.     /// Gets the chosen data point names.
210.     /// </summary>
211.     /// <param name="isChosen">The is chosen.</param>
212.     /// <returns></returns>
213.     public string[] GetChosenDataPointNames(bool[] isChosen)
214.     {
215.         var tmpList = new List<string>();
216.         for (var i = 0; i < isChosen.Length; i++)
217.             if (isChosen[i])
218.                 tmpList.Add(NumericHeadings[i]);
219.         return tmpList.ToArray();
220.     }
221.
222.     /// <summary>
223.     /// Saves data to CSV file.
224.     /// </summary>
225.     /// <param name="data">The data.</param>
226.     /// <param name="filePath">The file path.</param>
227.     public static void SaveToCsv(CSVData data, string filePath)
228.     {
229.         using (var fileStream = new FileStream(filePath, FileMode.Create))

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

229.        {
230.            var streamWriter = new StreamWriter(fileStream);
231.            foreach (var row in data.Rows) streamWriter.WriteLine(data.GetCs
vLineFromRow(row));
232.            streamWriter.Flush();
233.        }
234.    }
235.
236.
237.    /// <summary>
238.    /// Sorts the specified index.
239.    /// </summary>
240.    /// <param name="index">The index.</param>
241.    /// <param name="isAscending">if set to <c>true</c> [is ascending].</par
am>
242.    public void Sort(int index, bool isAscending)
243.    {
244.        SaveToStack();
245.        IEnumerable<CSVRow> sorted = isAscending ? Rows.OrderBy(row => row.F
ields[index]) : Rows.OrderByDescending(row => row.Fields[index]);
246.
247.        Rows = sorted.ToList();
248.        UpdateData();
249.    }
250.
251.    /// <summary>
252.    /// Filters the specified expression.
253.    /// </summary>
254.    /// <param name="expression">The expression.</param>
255.    /// <param name="index">The index.</param>
256.    /// <param name="selectedOperation">The selected operation.</param>
257.    public void Filter(string expression, int index, int selectedOperation)
258.    {
259.        SaveToStack();
260.        List<CSVRow> filteredList;
261.        switch (selectedOperation)
262.        {
263.            case 0:
264.                filteredList = Rows.FindAll(row => row.Fields[index].Contain
s(expression));
265.                break;
266.            case 1:
267.                filteredList = Rows.FindAll(row => double.Parse(row.Fields[i
ndex]) >= double.Parse(expression));
268.                break;
269.            case 2:
270.                filteredList = Rows.FindAll(row => double.Parse(row.Fields[i
ndex]) <= double.Parse(expression));
271.                break;
272.            case 3:
273.                filteredList = Rows.FindAll(row => double.Parse(row.Fields[i
ndex]) > double.Parse(expression));
274.                break;
275.            default:
276.                filteredList = Rows.FindAll(row => double.Parse(row.Fields[i
ndex]) < double.Parse(expression));
277.                break;
278.        }
279.
280.        Rows = filteredList;
281.        UpdateData();
282.    }
283.
284.    /// <summary>
285.    /// Redoes this instance.

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

286.         /// </summary>
287.     public void Redo()
288.     {
289.         if (RedoStack.Count != 0 && UndoStack.Count < MaxStackCount)
290.         {
291.             UndoStack.Push(Rows.GetRange(0, Rows.Count));
292.             Rows = RedoStack.Pop();
293.             UpdateData();
294.         }
295.     }
296.
297.     /// <summary>
298.     /// Undoes this instance.
299.     /// </summary>
300.     public void Undo()
301.     {
302.         if (UndoStack.Count != 0 && RedoStack.Count < MaxStackCount)
303.         {
304.             RedoStack.Push(Rows.GetRange(0, Rows.Count));
305.             Rows = UndoStack.Pop();
306.             UpdateData();
307.         }
308.     }
309.
310.     /// <summary>
311.     /// Saves to stack.
312.     /// </summary>
313.     public void SaveToStack()
314.     {
315.         if (UndoStack.Count < MaxStackCount)
316.         {
317.             RedoStack.Clear();
318.             UndoStack.Push(Rows.GetRange(0, Rows.Count));
319.         }
320.     }
321.
322.     #endregion
323.
324. }
325. }

```

## 1.10. CSVRow.cs

```

1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4. using System.Text;
5. using System.Threading.Tasks;
6.
7. namespace Clusterizer
8. {
9.     /// <summary>
10.    /// Data structure for one line of CSV File
11.    /// </summary>
12.    public class CSVRow
13.    {
14.        /// <summary>
15.        /// Gets or sets the fields.
16.        /// </summary>
17.        /// <value>
18.        /// The fields.
19.        /// </value>
20.        public List<string> Fields { get; set; }
21.    }

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

22.     /// <summary>
23.     /// Initializes a new instance of the <see cref="CSVRow"/> class.
24.     /// </summary>
25.     /// <param name="fields">The fields.</param>
26.     public CSVRow(List<string> fields)
27.     {
28.         Fields = fields;
29.     }
30.
31.     /// <summary>
32.     /// Gets or sets the <see cref="System.String"/> at the specified index.
33.     /// </summary>
34.     /// <value>
35.     /// The <see cref="System.String"/>.
36.     /// </value>
37.     /// <param name="index">The index.</param>
38.     /// <returns>Field with specified index.</returns>
39.     public string this[int index]
40.     {
41.         get => Fields[index];
42.         set => Fields[index] = value;
43.     }
44. }
45. }

```

### 1.11. CustomException.cs

```

1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4. using System.Text;
5. using System.Threading.Tasks;
6.
7. namespace Clusterizer
8. {
9.     /// <summary>
10.    /// Custom exception for showing errors via MessageBox
11.    /// </summary>
12.    /// <seealso cref="System.Exception" />
13.    public class CustomException : Exception
14.    {
15.        /// <summary>
16.        /// Gets or sets the text.
17.        /// </summary>
18.        /// <value>
19.        /// The text.
20.        /// </value>
21.        public string Text { get; set; }
22.
23.        /// <summary>
24.        /// Gets or sets the caption.
25.        /// </summary>
26.        /// <value>
27.        /// The caption.
28.        /// </value>
29.        public string Caption { get; set; }
30.
31.        /// <summary>
32.        /// Initializes a new instance of the <see cref="CustomException"/> class.
33.        /// </summary>
34.        /// <param name="text">The text.</param>
35.        /// <param name="caption">The caption.</param>
36.        public CustomException(string text, string caption)
37.        {

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

38.         Text = text;
39.         Caption = caption;
40.     }
41. }
42. }

```

## 1.12. Datapoint.cs

```

1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4.
5. namespace Clusterizer
6. {
7.     /// <summary>
8.     /// Data structure for presenting set of points
9.     /// </summary>
10.    public class DataPoint
11.    {
12.
13.        #region Properties
14.        /// <summary>
15.        /// Gets or sets the points.
16.        /// </summary>
17.        /// <value>
18.        /// The points.
19.        /// </value>
20.        public List<double> Points { get; set; }
21.
22.        /// <summary>
23.        /// Gets or sets the identifier.
24.        /// </summary>
25.        /// <value>
26.        /// The identifier.
27.        /// </value>
28.        public int Id { get; set; }
29.
30.        /// <summary>
31.        /// Gets the Point's count.
32.        /// </summary>
33.        /// <value>
34.        /// The count.
35.        /// </value>
36.        public int Count => Points.Count;
37.        #endregion
38.
39.        #region Конструктор
40.        /// <summary>
41.        /// Initializes a new instance of the <see cref="DataPoint"/> class.
42.        /// </summary>
43.        public DataPoint()
44.        {
45.            Points = new List<double>();
46.        }
47.
48.        /// <summary>
49.        /// Initializes a new instance of the <see cref="DataPoint"/> class.
50.        /// </summary>
51.        /// <param name="points">The points.</param>
52.        public DataPoint(List<double> points)
53.        {
54.            Points = points;
55.        }
56.        #endregion

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

57.
58.     #region Methods
59.     /// <summary>
60.     /// Adds the specified point.
61.     /// </summary>
62.     /// <param name="point">The point.</param>
63.     public void Add(double point)
64.     {
65.         Points.Add(point);
66.     }
67.
68.     /// <summary>
69.     /// Gets or sets the <see cref="System.Double"/> at the specified index.
70.     /// </summary>
71.     /// <value>
72.     /// The <see cref="System.Double"/>.
73.     /// </value>
74.     /// <param name="index">The index.</param>
75.     /// <returns></returns>
76.     public double this[int index]
77.     {
78.         get => Points[index];
79.         set => Points[index] = value;
80.     }
81.     #endregion
82.
83. }
84. }

```

### 1.13. DendogramForm.cs

```

1. using System;
2. using System.Collections.Generic;
3. using System.ComponentModel;
4. using System.Data;
5. using System.Diagnostics;
6. using System.Drawing;
7. using System.Drawing.Imaging;
8. using System.IO;
9. using System.Linq;
10. using System.Text;
11. using System.Threading.Tasks;
12. using System.Windows.Forms;
13.
14. namespace Clusterizer
15. {
16.     /// <summary>
17.     /// Form for visualizing Dendogram
18.     /// </summary>
19.     /// <seealso cref="System.Windows.Forms.Form" />
20.     public partial class DendrogramForm : Form
21.     {
22.         #region Properties
23.         /// <summary>
24.         /// The drawarea
25.         /// </summary>
26.         private Graphics _drawarea;
27.
28.         /// <summary>
29.         /// The root
30.         /// </summary>
31.         private Node<string> _root;
32.
33.         /// <summary>

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

34.    /// The current count of leaves
35.    /// </summary>
36.    private int _leaves;
37.
38.    /// <summary>
39.    /// The current count of levels
40.    /// </summary>
41.    private int _levels;
42.
43.    /// <summary>
44.    /// The curent color
45.    /// </summary>
46.    private Color _color;
47.
48.    /// <summary>
49.    /// The width per level
50.    /// </summary>
51.    private int _widthPerLevel;
52.
53.    /// <summary>
54.    /// The maximum level
55.    /// </summary>
56.    private int _maxLevel;
57.
58.    /// <summary>
59.    /// The current y
60.    /// </summary>
61.    private int _currentY;
62.
63.    /// <summary>
64.    /// The height
65.    /// </summary>
66.    private int _height;
67.
68.    /// <summary>
69.    /// The width
70.    /// </summary>
71.    private int _width;
72.
73.    /// <summary>
74.    /// The root list
75.    /// </summary>
76.    private readonly List<Node<string>> _rootList;
77.
78.    /// <summary>
79.    /// The leaves list
80.    /// </summary>
81.    private readonly List<int> _leavesList;
82.
83.    /// <summary>
84.    /// The levels list
85.    /// </summary>
86.    private readonly List<int> _levelsList;
87.
88.    /// <summary>
89.    /// The colors
90.    /// </summary>
91.    private readonly List<Color> _colors;
92.
93.    /// <summary>
94.    /// The random
95.    /// </summary>
96.    static readonly Random random = new Random();
97.    #endregion
98.
99.    #region Constants

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата



```

100.    /// <summary>
101.    /// The height per leaf
102.    /// </summary>
103.    private const int HeightPerLeaf = 40;
104.
105.    /// <summary>
106.    /// The drawing area margin
107.    /// </summary>
108.    private const int DrawingAreaMargin = 25;
109.
110.    /// <summary>
111.    /// The contest offset
112.    /// </summary>
113.    private const int ContestOffset = 80;
114.
115.    /// <summary>
116.    /// The drawing area offset
117.    /// </summary>
118.    private const int DrawingAreaOffset = 35;
119.    #endregion
120.
121.    #region Constructor
122.    /// <summary>
123.    /// Initializes a new instance of the <see cref="DendrogramForm"/> class
124.    /// </summary>
125.    public DendrogramForm()
126.    {
127.        InitializeComponent();
128.        _drawarea = dendrogramControl.CreateGraphics();
129.        _rootList = new List<Node<string>>();
130.        _leavesList = new List<int>();
131.        _levelsList = new List<int>();
132.        _colors = new List<Color>();
133.    }
134.    #endregion
135.
136.
137.    #region Methods
138.    /// <summary>
139.    /// Creates the specified contents.
140.    /// </summary>
141.    /// <param name="contents">The contents.</param>
142.    /// <returns></returns>
143.    private Node<string> Create(string contents)
144.    {
145.        return new Node<string>(contents);
146.    }
147.
148.    /// <summary>
149.    /// Creates the specified child0.
150.    /// </summary>
151.    /// <param name="child0">The child0.</param>
152.    /// <param name="child1">The child1.</param>
153.    /// <returns></returns>
154.    private Node<string> Create(Node<string> child0, Node<string> child1)
155.    {
156.        return new Node<string>(child0, child1);
157.    }
158.
159.    /// <summary>
160.    /// Counts the leaves.
161.    /// </summary>
162.    /// <param name="node">The node.</param>
163.    /// <returns></returns>
164.    private int CountLeaves(Node<string> node)

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

165.         {
166.             List<Node<string>> children = node.ChildrenNodes;
167.             if (children.Count == 0)
168.             {
169.                 return 1;
170.             }
171.
172.             Node<string> child0 = children.ElementAt(0);
173.             Node<string> child1 = children.ElementAt(1);
174.
175.             return CountLeaves(child0) + CountLeaves(child1);
176.         }
177.
178.         /// <summary>
179.         /// Counts the levels.
180.         /// </summary>
181.         /// <param name="node">The node.</param>
182.         /// <returns></returns>
183.         private int CountLevels(Node<string> node)
184.         {
185.             List<Node<string>> children = node.ChildrenNodes;
186.             if (children.Count == 0)
187.             {
188.                 return 1;
189.             }
190.
191.             Node<string> child0 = children.ElementAt(0);
192.             Node<string> child1 = children.ElementAt(1);
193.
194.             return 1 + Math.Max(CountLevels(child0), CountLevels(child1));
195.         }
196.
197.
198.         /// <summary>
199.         /// Builds the dendrogram.
200.         /// </summary>
201.         /// <param name="clusters">The clusters.</param>
202.         /// <returns>Return overall node</returns>
203.         private Node<string> BuildDendrogram(Cluster[] clusters)
204.         {
205.             Node<string> child0;
206.             Node<string> child1;
207.
208.             // cluster with two clusters and without subcluster
209.             if (clusters.Length == 2 && clusters.ElementAt(0).QuantityOfSubClusters == 0)
210.             {
211.                 child0 = GetNodeFromCluster(clusters.ElementAt(0));
212.                 child1 = GetNodeFromCluster(clusters.ElementAt(1));
213.
214.                 return Create(child0, child1);
215.             }
216.
217.             // singleton cluster with two subclusters
218.             if (clusters.Length == 1 && clusters.ElementAt(0).QuantityOfSubClusters == 2)
219.             {
220.                 return BuildDendrogram(clusters.ElementAt(0).SubClusters.ToArray());
221.             }
222.
223.             // cluster with two clusters and by 2 subclusters
224.             if (clusters.Length == 2 && clusters.ElementAt(0).QuantityOfSubClusters == 2)
225.             {

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

226.         child0 = BuildDendrogram(clusters.ElementAt(0).SubClusters.ToArr
ay());
227.
228.         child1 = clusters.ElementAt(1).QuantityOfSubClusters == 2 ? Buil
dDendrogram(clusters.ElementAt(1).SubClusters.ToArray()) : GetNodeFromCluster(clusters.
ElementAt(1));
229.
230.
231.         return Create(child0, child1);
232.     }
233.
234.
235.     return _root;
236.
237. }
238.
239.     /// <summary>
240.     /// Gets the node from cluster.
241.     /// </summary>
242.     /// <param name="cluster">The cluster.</param>
243.     /// <returns></returns>
244.     private Node<string> GetNodeFromCluster(Cluster cluster)
245.     {
246.         return Create("Cluster" + cluster.Id.ToString());
247.     }
248.
249.     /// <summary>
250.     /// Setups this instance.
251.     /// </summary>
252.     public void Setup()
253.     {
254.         for (int i = 0; i < Tools.Clusters.ClustersList.Count; i++)
255.         {
256.             // get overall root
257.             _root = BuildDendrogram(new Cluster[] { Tools.Clusters.ClustersL
ist.ElementAt(i) });
258.
259.             // check if singleton cluster
260.             if (Tools.Clusters.GetCluster(i).QuantityOfSubClusters == 0)
261.                 _root = GetNodeFromCluster(Tools.Clusters.GetCluster(i));
262.
263.             // calculate values
264.             _leaves = CountLeaves(_root);
265.             _levels = CountLevels(_root);
266.
267.             // get maxLevel count
268.             if (_levels > _maxLevel)
269.                 _maxLevel = _levels;
270.
271.             // add calculated values to list
272.             _rootList.Add(_root);
273.             _levelsList.Add(_levels);
274.             _leavesList.Add(_leaves);
275.
276.             // generate new random color
277.             Random rand = random;
278.             int max = byte.MaxValue + 1; // 256
279.             int r = rand.Next(max);
280.             int g = rand.Next(max);
281.             int b = rand.Next(max);
282.             Color c = Color.FromArgb(r, g, b);
283.             _colors.Add(c);
284.         }
285.     }
286.
287.     /// <summary>

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

288.         /// Draws the specified g.
289.         /// </summary>
290.         /// <param name="g">The g.</param>
291.         /// <param name="node">The node.</param>
292.         /// <param name="y">The y.</param>
293.         /// <returns></returns>
294.         private Point Draw(Graphics g, Node<string> node, int y)
295.         {
296.             // children of node
297.             List<Node<string>> children = node.ChildrenNodes;
298.             // pen for drawing
299.             Pen pen = new Pen(_color);
300.             // brush for drawing
301.             SolidBrush brush = new SolidBrush(_color);
302.
303.             // if singleton cluster
304.             if (children.Count == 0)
305.             {
306.                 int x = Width - ContestOffset - _widthPerLevel -
307.                 2 * DrawingAreaMargin;
308.                 // draws string with cluster nam
309.                 g.DrawString(node.Contents, new Font("Times New Roman", 8.0f), b
310.                 rush, x + 8, _currentY - 8);
311.                 int resultX = x;
312.                 int resultY = _currentY;
313.                 _currentY += HeightPerLeaf;
314.                 return new Point(resultX, resultY);
315.             }
316.
317.             // if cluster have subclusters
318.             if (children.Count >= 2)
319.             {
320.                 Node<string> child0 = children.ElementAt(0);
321.                 Node<string> child1 = children.ElementAt(1);
322.
323.                 // get points of the childs by order
324.                 Point p0 = Draw(g, child0, y);
325.                 Point p1 = Draw(g, child1, y + HeightPerLeaf);
326.
327.                 // fill rectanbles of them
328.                 g.FillRectangle(brush, p0.X - 2, p0.Y - 2, 4, 4);
329.                 g.FillRectangle(brush, p1.X - 2, p1.Y - 2, 4, 4);
330.
331.                 // calculate parameters
332.                 int dx = _widthPerLevel;
333.                 int vx = Math.Min(p0.X - dx, p1.X - dx);
334.
335.                 // combine line with each other
336.                 Pen blackPen = pen;
337.                 g.DrawLine(blackPen, vx, p0.Y, p0.X, p0.Y);
338.                 g.DrawLine(blackPen, vx, p1.Y, p1.X, p1.Y);
339.                 g.DrawLine(blackPen, vx, p0.Y, vx, p1.Y);
340.
341.                 // returns new point
342.                 Point p = new Point(vx, p0.Y + (p1.Y - p0.Y) / 2);
343.                 return p;
344.             }
345.
346.             return new Point();
347.         }
348.
349.         #endregion
350.
351.         #region Events
352.         /// <summary>
353.         /// Handles the SizeChanged event of the DendrogramFrm control.

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

352.          /// </summary>
353.          /// <param name="sender">The source of the event.</param>
354.          /// <param name="e">The <see cref="EventArgs"/> instance containing the
event data.</param>
355.          private void DendrogramForm_SizeChanged(object sender, EventArgs e)
356.          {
357.              // gets size of form
358.              _width = Width - DrawingAreaOffset;
359.              _height = 0;
360.
361.              for (int i = 0; i < _levelsList.Count; i++)
362.              {
363.                  _height += (HeightPerLeaf * _leavesList[i]);
364.              }
365.
366.              // calculates new parameters of drawing
367.              _height += 2 * DrawingAreaMargin + 50;
368.              _widthPerLevel = (_width - ContestOffset - DrawingAreaMargin -
DrawingAreaMargin) / _maxLevel;
369.
370.              // redraw
371.              dendrogramControl.Invalidate();
372.          }
373.
374.          /// <summary>
375.          /// Handles the FormClosing event of the DendrogramForm control.
376.          /// </summary>
377.          /// <param name="sender">The source of the event.</param>
378.          /// <param name="e">The <see cref="FormClosingEventArgs"/> instance containing the event data.</param>
379.          private void DendrogramForm_FormClosing(object sender, FormClosingEventArgs e)
380.          {
381.              // save image to file before closing
382.              Bitmap bmp = new Bitmap(dendrogramControl.Width, dendrogramControl.Height);
383.              dendrogramControl.DrawToBitmap(bmp, new Rectangle(0, 0, dendrogramControl.Width, dendrogramControl.Height));
384.              bmp.Save("tmpDendogram.png");
385.          }
386.
387.          /// <summary>
388.          /// Handles the Paint event of the dendrogramControl control.
389.          /// </summary>
390.          /// <param name="sender">The source of the event.</param>
391.          /// <param name="e">The <see cref="PaintEventArgs"/> instance containing the event data.</param>
392.          private void dendrogramControl_Paint(object sender, PaintEventArgs e)
393.          {
394.              // setup draw area
395.              dendrogramControl.Size = new Size(_width, _height);
396.              _currentY = 0;
397.              _drawarea = e.Graphics;
398.              _drawarea.TranslateTransform(DrawingAreaMargin, DrawingAreaMargin);
399.
400.              // gets all self-dependent elements and draw them
401.              for (int i = 0; i < Tools.Clusters.ClustersList.Count; i++)
402.              {
403.                  _root = _rootList[i];
404.                  _levels = _levelsList[i];
405.                  _leaves = _leavesList[i];
406.                  _color = _colors[i];
407.
408.                  _widthPerLevel = (_width - ContestOffset - DrawingAreaMargin -
DrawingAreaMargin) / _maxLevel;

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

409.
410.             // draw element
411.             Draw(_drawarea, _root, _currentY);
412.         }
413.     }
414.     #endregion
415.
416.
417.     }
418. }

```

#### 1.14. DissimilarityMatrix.cs

```

1. using System;
2. using System.Collections.Concurrent;
3. using System.Collections.Generic;
4. using System.Linq;
5.
6. namespace Clusterizer
7. {
8.     /// <summary>
9.     /// Data structure for presenting dissimilarity matrix
10.    /// </summary>
11.    public class DissimilarityMatrix
12.    {
13.        #region Fields
14.        /// <summary>
15.        /// The distance matrix
16.        /// </summary>
17.        private readonly Dictionary<ClusterPair, double> _distanceMatrix;
18.        #endregion
19.
20.        #region Constructor
21.        /// <summary>
22.        /// Initializes a new instance of the <see cref="DissimilarityMatrix"/> class.
23.        /// </summary>
24.        public DissimilarityMatrix()
25.        {
26.            _distanceMatrix = new Dictionary<ClusterPair, double>(new ClusterPair.Equal
ityComparer());
27.        }
28.        #endregion
29.
30.        #region Methods
31.        /// <summary>
32.        /// Adds the cluster pair and distance.
33.        /// </summary>
34.        /// <param name="clusterPair">The cluster pair.</param>
35.        /// <param name="distance">The distance.</param>
36.        public void AddClusterPairAndDistance(ClusterPair clusterPair, double distance)
37.        {
38.            _distanceMatrix.Add(clusterPair, distance);
39.        }
40.
41.        /// <summary>
42.        /// Removes the cluster pair.
43.        /// </summary>
44.        /// <param name="clusterPair">The cluster pair.</param>
45.        public void RemoveClusterPair(ClusterPair clusterPair)
46.        {
47.            _distanceMatrix.Remove(_distanceMatrix.ContainsKey(clusterPair)
? clusterPair
48.

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

49.         : new ClusterPair(clusterPair.Cluster2, clusterPair.Cluster1));
50.     }
51.
52.     /// <summary>
53.     /// Gets the closest cluster pair.
54.     /// </summary>
55.     /// <returns></returns>
56.     public ClusterPair GetClosestClusterPair()
57.     {
58.         double minDistance = double.MaxValue;
59.         ClusterPair closestClusterPair = null;
60.
61.         foreach (var item in _distanceMatrix)
62.         {
63.             if (item.Value < minDistance)
64.             {
65.                 minDistance = item.Value;
66.                 closestClusterPair = item.Key;
67.             }
68.         }
69.
70.         return closestClusterPair;
71.     }
72.
73.
74.     /// <summary>
75.     /// Returns the cluster pair distance.
76.     /// </summary>
77.     /// <param name="clusterPair">The cluster pair.</param>
78.     /// <returns></returns>
79.     public double ReturnClusterPairDistance(ClusterPair clusterPair)
80.     {
81.         double clusterPairDistance;
82.
83.         clusterPairDistance = _distanceMatrix.ContainsKey(clusterPair) ? _distanceM
84.         atrix[clusterPair] : _distanceMatrix[new ClusterPair(clusterPair.Cluster2, clusterPair.
85.         Cluster1)];
86.         return clusterPairDistance;
87.     }
88. }

```

### 1.15. Distance.cs

```

1. using System;
2.
3. namespace Clusterizer
4. {
5.     public static class Distance
6.     {
7.         /// <summary>
8.         /// Gets the distance.
9.         /// </summary>
10.        /// <param name="x">The x.</param>
11.        /// <param name="y">The y.</param>
12.        /// <param name="distanceMetric">The distance metric.</param>
13.        /// <returns>Datapoints distance</returns>
14.        /// <exception cref="ArgumentException">Неравное количество точек.</exception>
15.        public static double GetDistance(DataPoint x, DataPoint y, DistanceMetric dista
16.        nceMetric)
17.        {
18.            double distance = 0;

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

18.         double diff;
19.
20.         // checks for dimensions match
21.         if (x.Count != y.Count)
22.             throw new ArgumentException("Неравное количество точек.");
23.
24.         switch (distanceMetric)
25.         {
26.             case DistanceMetric.EuclidianDistance: // calculates by using Euclidian
Distance
27.                 for (var i = 0; i < x.Count; i++)
28.                 {
29.                     diff = x[i] - y[i];
30.                     distance += diff * diff;
31.                 }
32.
33.                 distance = Math.Sqrt(distance);
34.                 break;
35.             case DistanceMetric.SquareEuclidianDistance: // calculates by using Squ
are of Euclidian Distance
36.                 for (var i = 0; i < x.Count; i++)
37.                 {
38.                     diff = x[i] - y[i];
39.                     distance += diff * diff;
40.                 }
41.
42.                 break;
43.             case DistanceMetric.ManhattanDistance: // calculates by using Manhattan
Distance
44.                 for (var i = 0; i < x.Count; i++)
45.                 {
46.                     diff = x[i] - y[i];
47.                     distance += Math.Abs(diff);
48.                 }
49.
50.                 break;
51.             case DistanceMetric.ChebyshevDistance: // calculates by using Chebyshev
Distance
52.                 for (var i = 0; i < x.Count; i++)
53.                 {
54.                     diff = Math.Abs(x[i] - y[i]);
55.                     distance = distance > diff ? distance : diff;
56.                 }
57.
58.                 break;
59.         }
60.
61.         return distance;
62.     }
63. }
64.
65. #region DistanceMetric
66.
67. public enum DistanceMetric
68. {
69.     EuclidianDistance, // Euclidian Metric
70.     SquareEuclidianDistance, // Square of Euclidian Metric
71.     ManhattanDistance, // Manhattan Metric
72.     ChebyshevDistance // Chebyshev Metric
73. }
74.
75. #endregion
76. }

```

## 1.16. MainForm.cs

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата



```

1. using System;
2. using System.Collections.Generic;
3. using System.Collections.Specialized;
4. using System.ComponentModel;
5. using System.Data;
6. using System.Drawing;
7. using System.IO;
8. using System.Linq;
9. using System.Runtime.Serialization.Formatters.Binary;
10. using System.Text;
11. using System.Threading;
12. using System.Threading.Tasks;
13. using System.Windows.Forms;
14. using Clusterizer.Properties;
15.
16. namespace Clusterizer
17. {
18.     /// <summary>
19.     /// The application's Main Form
20.     /// </summary>
21.     /// <seealso cref="System.Windows.Forms.Form" />
22.     public partial class MainForm : Form
23.     {
24.         #region Fields
25.         /// <summary>
26.         /// The data points before normalization
27.         /// </summary>
28.         private List<DataPoint> _dataPointsBeforeNormalization;
29.
30.         /// <summary>
31.         /// The statistics form
32.         /// </summary>
33.         private StatisticsForm statisticsForm;
34.         #endregion
35.
36.
37.         #region Constructor
38.         /// <summary>
39.         /// Initializes a new instance of the <see cref="MainForm"/> class.
40.         /// </summary>
41.         public MainForm()
42.         {
43.             InitializeComponent();
44.             ResetComponents();
45.             ResetData();
46.             Tools.Load();
47.         }
48.
49.         #endregion
50.
51.
52.         #region Methods
53.         /// <summary>
54.         /// Resets the components.
55.         /// </summary>
56.         private void ResetComponents()
57.         {
58.             // deletes temporary files
59.             DeleteTempFiles();
60.
61.             // resets all UI components
62.             dataTableGridView.DataSource = null;
63.
64.             filteredColumnSelectComboBox.Items.Clear();
65.             filteredColumnSelectComboBox.SelectedItem = null;
66.             sortColumnSelectComboBox.Items.Clear();

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

67.         sortColumnSelectComboBox.SelectedItem = null;
68.         operandSelectComboBox.Items.Clear();
69.
70.         clusterTreeView.Nodes.Clear();
71.         clusterTableGridView.Rows.Clear();
72.         clusteringToolStrip.Enabled = false;
73.         searchExpressionTextBox.Text = "";
74.
75.         // Disables using clustering
76.         DisableClustering();
77.     }
78.
79.     /// <summary>
80.     /// Resets the data.
81.     /// </summary>
82.     private void ResetData()
83.     {
84.         Tools.Data = null;
85.         Tools.Clusters = null;
86.         Tools.isChosen = null;
87.     }
88.
89.     /// <summary>
90.     /// Restores the cluster set before normalization.
91.     /// </summary>
92.     /// <param name="cluster">The cluster.</param>
93.     private void RestoreClusterSet(Cluster cluster)
94.     {
95.         for (var i = 0; i < cluster.DataPoints.Count; i++)
96.             cluster.DataPoints[i] = _dataPointsBeforeNormalization[cluster.DataPoin
ts[i].Id];
97.
98.         foreach (var subCluster in cluster.SubClusters)
99.             RestoreClusterSet(subCluster);
100.
101.         cluster.SetCentroid();
102.     }
103.
104.     /// <summary>
105.     /// Deletes the temporary files.
106.     /// </summary>
107.     public void DeleteTempFiles()
108.     {
109.         if (File.Exists("tmpOverview.csv"))
110.             File.Delete("tmpOverview.csv");
111.         if (File.Exists("tmpTable.csv"))
112.             File.Delete("tmpTable.csv");
113.         if (File.Exists("tmpDendogramm.png"))
114.             File.Delete("tmpDendogramm.png");
115.     }
116.
117.     /// <summary>
118.     /// Loads the data.
119.     /// </summary>
120.     public void LoadData()
121.     {
122.         // resets components
123.         ResetComponents();
124.
125.         // loads data table
126.         dataTableGridView.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsM
ode.None;
127.         dataTableGridView.ColumnHeadersVisible = false;
128.         dataTableGridView.DataSource = Tools.Data.DataSetTable;
129.         dataTableGridView.ColumnHeadersVisible = true;

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

130.         dataTableGridView.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsM
ode.DisplayedCells;
131.         dataTableGridView.Refresh();
132.
133.         // load filter and sort fields
134.         filteredColumnSelectComboBox.Items.AddRange(Tools.Data.StringHeading
s);
135.         filteredColumnSelectComboBox.Items.AddRange(Tools.Data.NumericHeadin
gs);
136.         sortColumnSelectComboBox.Items.AddRange(Tools.Data.StringHeadings);
137.
138.         filteredColumnSelectComboBox.SelectedIndex = 0;
139.         sortColumnSelectComboBox.SelectedIndex = 0;
140.
141.         // enables clustering
142.         EnableClustering();
143.     }
144.
145.     /// <summary>
146.     /// Builds the result table.
147.     /// </summary>
148.     private void BuildResultTable()
149.     {
150.         // resets result table
151.         clusterTableGridView.Rows.Clear();
152.         clusterTableGridView.AutoSizeColumnsMode = DataGridViewAutoSizeColum
nsMode.None;
153.         clusterTableGridView.ColumnHeadersVisible = false;
154.
155.         // saves table to temp file
156.         using (var fileStream = new FileStream("tmpTable.csv", FileMode.Crea
te))
157.         {
158.             var streamWriter = new StreamWriter(fileStream);
159.             // gets table from clusters
160.             for (var i = 0; i < Tools.Clusters.ClustersList.Count; i++)
161.             {
162.                 var baseId = Tools.Clusters.GetCluster(i).Id;
163.                 foreach (var pattern in Tools.Clusters.GetCluster(i).DataPoi
nts)
164.                 {
165.                     clusterTableGridView.Rows.Add(Tools.Data.Rows[pattern.Id
].Fields[0], $"Cluster{pattern.Id}",
166.                     $"Cluster{baseId}");
167.                     streamWriter.WriteLine(
168.                         $"{Tools.Data.Rows[pattern.Id].Fields[0]};Cluster{pa
ttern.Id};Cluster{baseId}");
169.                 }
170.             }
171.
172.             streamWriter.Flush();
173.         }
174.
175.         // load data to table
176.         clusterTableGridView.ColumnHeadersVisible = true;
177.         clusterTableGridView.AutoSizeColumnsMode = DataGridViewAutoSizeColum
nsMode.AllCells;
178.
179.         clusterTableGridView.Refresh();
180.         exportClusterTableToolStripMenuItem.Enabled = true;
181.     }
182.
183.     /// <summary>
184.     /// Builds the TreeView.
185.     /// </summary>

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

186.         private void BuildTreeView()
187.         {
188.             clusterTreeView.Nodes.Clear();
189.             foreach (var cluster in Tools.Clusters.ClustersList)
190.             {
191.                 var rootNode = clusterTreeView.Nodes.Add($"Cluster{cluster.Id}")
192.             ;
193.                 AddNodes(cluster.SubClusters.ToArray(), rootNode);
194.             }
195.             clusterTreeView.CollapseAll();
196.         }
197.
198.         /// <summary>
199.         /// Adds the nodes to node
200.         /// </summary>
201.         /// <param name="clusters">The clusters.</param>
202.         /// <param name="node">The node.</param>
203.         private void AddNodes(Cluster[] clusters, TreeNode node)
204.         {
205.             foreach (var cluster in clusters)
206.             {
207.                 var childNode = node.Nodes.Add($"Cluster{cluster.Id}");
208.                 if (cluster.QuantityOfSubClusters > 0)
209.                     AddNodes(cluster.SubClusters.ToArray(), childNode);
210.             }
211.         }
212.
213.         /// <summary>
214.         /// Disables the clustering.
215.         /// </summary>
216.         private void DisableClustering()
217.         {
218.             tabControl.TabPages[1].Enabled = false;
219.             clusterizationToolStripMenuItem.Enabled = false;
220.             buildDendrogramToolStripMenuItem.Enabled = false;
221.             showClusterOverviewToolStripMenuItem.Enabled = false;
222.             exportToolStripMenuItem.Enabled = false;
223.             exportClusterDendrogramToolStripMenuItem.Enabled = false;
224.             exportClusterOverviewToolStripMenuItem.Enabled = false;
225.             exportClusterTableToolStripMenuItem.Enabled = false;
226.         }
227.
228.         /// <summary>
229.         /// Enables the clustering.
230.         /// </summary>
231.         private void EnableClustering()
232.         {
233.             tabControl.TabPages[1].Enabled = true;
234.             clusterizationToolStripMenuItem.Enabled = true;
235.             clusteringToolStrip.Enabled = true;
236.         }
237.
238.         #endregion
239.
240.         #region Events
241.
242.         #region Menu - File
243.         /// <summary>
244.         /// Handles the Click event of the openToolStripMenuItem control.
245.         /// </summary>
246.         /// <param name="sender">The source of the event.</param>
247.         /// <param name="e">The <see cref="EventArgs"/> instance containing the
248.         event data.</param>
249.         /// <exception cref="Clusterizer.CustomException">Проверьте корректность
250.         данных и доступность файла. - Ошибка при открытии входных данных</exception>

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

249.         private void openToolStripMenuItem_Click(object sender, EventArgs e)
250.         {
251.             try
252.             {
253.                 var openFileDialog = new OpenFileDialog {Title = "Открыть файл",
Filter = "CSV File(*.csv)|*.csv"};
254.                 // check if user clicked ok
255.                 if (openFileDialog.ShowDialog() == DialogResult.OK)
256.                 {
257.                     var filePath = openFileDialog.FileName;
258.                     var data = new CSVData(filePath);
259.                     data.CreateDataTable();
260.                     ResetData();
261.                     Tools.Data = data;
262.                     LoadData();
263.                 }
264.             }
265.             catch
266.             {
267.                 throw new CustomException("Проверьте корректность данных и досту
пность файла.", "Ошибка при открытии входных данных");
268.             }
269.         }
270.
271.         /// <summary>
272.         /// Handles the Click event of the saveToolStripMenuItem control.
273.         /// </summary>
274.         /// <param name="sender">The source of the event.</param>
275.         /// <param name="e">The <see cref="EventArgs"/> instance containing the
event data.</param>
276.         /// <exception cref="Clusterizer.CustomException">Произошла ошибка при с
охранении файла. - Ошибка сохранения файла</exception>
277.         private void saveToolStripMenuItem_Click(object sender, EventArgs e)
278.         {
279.             try
280.             {
281.                 if (Tools.Data != null)
282.                 {
283.                     // updates data
284.                     Tools.Data.UpdateRows();
285.                     // saves data
286.                     CSVData.SaveToCsv(Tools.Data, Tools.Data.FilePath);
287.                 }
288.             }
289.             catch
290.             {
291.                 throw new CustomException("Произошла ошибка при сохранении файла
.", "Ошибка сохранения файла");
292.             }
293.         }
294.
295.         /// <summary>
296.         /// Handles the Click event of the saveAsToolStripMenuItem control.
297.         /// </summary>
298.         /// <param name="sender">The source of the event.</param>
299.         /// <param name="e">The <see cref="EventArgs"/> instance containing the
event data.</param>
300.         /// <exception cref="Clusterizer.CustomException">Произошла ошибка при с
охранении файла. - Ошибка сохранения файла</exception>
301.         private void saveAsToolStripMenuItem_Click(object sender, EventArgs e)
302.         {
303.             try
304.             {
305.                 var saveFileDialog = new SaveFileDialog {Title = "Сохранить как.
..", Filter = "CSV File(*.csv)|*.csv"};
306.                 // check if user selected ok

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

307.             if (Tools.Data != null && saveFileDialog.ShowDialog() == DialogResult
result.OK)
308.             {
309.                 var filePath = saveFileDialog.FileName;
310.                 // updates data
311.                 Tools.Data.UpdateRows();
312.                 // saves data
313.                 CSVData.SaveToCsv(Tools.Data, filePath);
314.             }
315.         }
316.         catch
317.         {
318.             throw new CustomException("Произошла ошибка при сохранении файла
.", "Ошибка сохранения файла");
319.         }
320.     }
321.
322.     /// <summary>
323.     /// Handles the Click event of the closeToolStripMenuItem control.
324.     /// </summary>
325.     /// <param name="sender">The source of the event.</param>
326.     /// <param name="e">The <see cref="EventArgs"/> instance containing the
event data.</param>
327.     private void closeToolStripMenuItem_Click(object sender, EventArgs e)
328.     {
329.         ResetData();
330.         ResetComponents();
331.     }
332.
333.
334.
335.     /// <summary>
336.     /// Handles the Click event of the exitToolStripMenuItem control.
337.     /// </summary>
338.     /// <param name="sender">The source of the event.</param>
339.     /// <param name="e">The <see cref="EventArgs"/> instance containing the
event data.</param>
340.     private void exitToolStripMenuItem_Click(object sender, EventArgs e)
341.     {
342.         Close();
343.     }
344.
345.     /// <summary>
346.     /// Handles the Click event of the exportClusterDendogrammToolStripMenuI
tem control.
347.     /// </summary>
348.     /// <param name="sender">The source of the event.</param>
349.     /// <param name="e">The <see cref="EventArgs"/> instance containing the
event data.</param>
350.     private void exportClusterDendogrammToolStripMenuItem_Click(object sende
r, EventArgs e)
351.     {
352.         var saveFileDialog = new SaveFileDialog {Title = "Сохранить как...",
Filter = "PNG File(*.png)|*.png"};
353.         if (saveFileDialog.ShowDialog() == DialogResult.OK) File.Move("tmpDe
ndogramm.png", saveFileDialog.FileName);
354.     }
355.
356.     /// <summary>
357.     /// Handles the Click event of the exportClusterTableToolStripMenuItem c
ontrol.
358.     /// </summary>
359.     /// <param name="sender">The source of the event.</param>
360.     /// <param name="e">The <see cref="EventArgs"/> instance containing the
event data.</param>

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

361.         private void exportClusterTableToolStripMenuItem_Click(object sender, EventArgs e)
362.         {
363.             var saveFileDialog = new SaveFileDialog {Title = "Сохранить как...",
364.             Filter = "CSV File(*.csv)|*.csv"};
365.             if (saveFileDialog.ShowDialog() == DialogResult.OK) File.Move("tmpTable.csv", saveFileDialog.FileName);
366.         }
367.         /// <summary>
368.         /// Handles the Click event of the exportClusterOverviewToolStripMenuItem control.
369.         /// </summary>
370.         /// <param name="sender">The source of the event.</param>
371.         /// <param name="e">The <see cref="EventArgs"> instance containing the event data.</param>
372.         private void exportClusterOverviewToolStripMenuItem_Click(object sender, EventArgs e)
373.         {
374.             var saveFileDialog = new SaveFileDialog {Title = "Сохранить как...",
375.             Filter = "CSV File(*.csv)|*.csv"};
376.             if (saveFileDialog.ShowDialog() == DialogResult.OK) File.Move("tmpOverview.csv", saveFileDialog.FileName);
377.         }
378.         #endregion
379.         #region Menu - Clusterization
380.         /// <summary>
381.         /// Handles the Click event of the clusterizeToolStripMenuItem control.
382.         /// </summary>
383.         /// <param name="sender">The source of the event.</param>
384.         /// <param name="e">The <see cref="EventArgs"> instance containing the event data.</param>
385.         private void clusterizeToolStripMenuItem_Click(object sender, EventArgs e)
386.         {
387.             // shows form
388.             var clusterizeForm = new ClusterizeForm();
389.             clusterizeForm.ShowDialog();
390.             if (Tools.Data != null && clusterizeForm.isParametersSelected)
391.             {
392.                 // updates data
393.                 Tools.Data.UpdateRows();
394.                 // get clusterset
395.                 var clusters = Tools.Data.GetClusterSet(Tools.isChosen);
396.                 Tools.Clusters = clusters;
397.                 // backups clusters before normalization
398.                 _dataPointsBeforeNormalization = new List<DataPoint>();
399.                 for (var i = 0; i < clusters.ClustersList.Count; i++)
400.                 {
401.                     _dataPointsBeforeNormalization.Add(
402.                     new DataPoint(new List<double>(clusters[i].DataPoints[0].Points)));
403.                     _dataPointsBeforeNormalization[i].Id = i;
404.                 }
405.                 // normalize clusters
406.                 clusters.Normalize(clusterizeForm.normalizeMethod);
407.                 var agnes = new Agnes(clusters,
408.                 clusterizeForm.distanceMetric, clusterizeForm.strategy);
409.             }
410.         }
411.

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

415.          // execute clustering
416.          Tools.Clusters = agnes.ExecuteClustering(clusterizeForm.countOfC
clusters);
417.
418.          // builds components
419.          BuildTreeView();
420.          BuildResultTable();
421.          tabControl.SelectTab(1);
422.
423.          // enables functions
424.          buildDendrogramToolStripMenuItem.Enabled = true;
425.          showClusterOverviewToolStripMenuItem.Enabled = true;
426.          exportToolStripMenuItem.Enabled = true;
427.
428.          Task.Factory.StartNew(() =>
429.          {
430.              // shows form
431.              statisticsForm = new StatisticsForm
432.              {
433.                  contentsHeadings = Tools.Data.GetChosenDataPointNames(To
ols.isChosen),
434.                  Clusters = new ClusterSet
435.                  {
436.                      ClustersList = Tools.Clusters.ClustersList.GetRange(
0, Tools.Clusters.Count)
437.                  }
438.              };
439.
440.              // setup form
441.              statisticsForm.Clusters.ClustersList.ForEach(RestoreClustersS
et);
442.              statisticsForm.Setup();
443.
444.              });
445.          }
446.      }
447.
448.
449.      /// <summary>
450.      /// Handles the Click event of the buildDendrogramToolStripMenuItem cont
rol.
451.      /// </summary>
452.      /// <param name="sender">The source of the event.</param>
453.      /// <param name="e">The <see cref="EventArgs"/> instance containing the
event data.</param>
454.      private void buildDendrogramToolStripMenuItem_Click(object sender, Event
Args e)
455.      {
456.          if (Tools.Clusters != null)
457.          {
458.              // show form
459.              var dendrogramFrm = new DendrogramForm();
460.
461.              // setup form
462.              //dendrogramFrm._clusters = Tools.Clusters;
463.              dendrogramFrm.Setup();
464.              dendrogramFrm.ShowDialog();
465.              exportClusterDendrogramToolStripMenuItem.Enabled = true;
466.          }
467.      }
468.
469.      /// <summary>
470.      /// Handles the Click event of the showClusterOverviewToolStripMenuItem
control.
471.      /// </summary>
472.      /// <param name="sender">The source of the event.</param>

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата



```

473.          /// <param name="e">The <see cref="EventArgs"/> instance containing the
event data.</param>
474.      private void showClusterOverviewToolStripMenuItem_Click(object sender, E
ventArgs e)
475.      {
476.
477.          if (Tools.Clusters != null)
478.          {
479.              statisticsForm.ShowDialog();
480.              exportClusterOverviewToolStripMenuItem.Enabled = true;
481.          }
482.      }
483.      #endregion
484.
485.      #region Menu - About
486.      /// <summary>
487.      /// Handles the Click event of the aboutToolStripMenuItem control.
488.      /// </summary>
489.      /// <param name="sender">The source of the event.</param>
490.      /// <param name="e">The <see cref="EventArgs"/> instance containing the
event data.</param>
491.      private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
492.      {
493.          var aboutBox = new AboutBox();
494.          aboutBox.ShowDialog();
495.      }
496.
497.
498.      #endregion
499.
500.      #region DataTableGridView
501.
502.      /// <summary>
503.      /// Handles the DataError event of the dataTableGridView control.
504.      /// </summary>
505.      /// <param name="sender">The source of the event.</param>
506.      /// <param name="e">The <see cref="DataGridViewDataErrorEventArgs"/> ins
tance containing the event data.</param>
507.      private void dataTableGridView_DataError(object sender, DataGridViewData
ErrorEventArgs e)
508.      {
509.          e.ThrowException = false;
510.          e.Cancel = false;
511.          throw new CustomException("Введено не корректное значение.", "Ошибка
при редактировании таблицы");
512.      }
513.
514.      /// <summary>
515.      /// Handles the DefaultValuesNeeded event of the dataTableGridView contr
ol.
516.      /// </summary>
517.      /// <param name="sender">The source of the event.</param>
518.      /// <param name="e">The <see cref="DataGridViewRowEventArgs"/> instance
containing the event data.</param>
519.      private void dataTableGridView_DefaultValuesNeeded(object sender, DataGr
idViewRowEventArgs e)
520.      {
521.          int i;
522.          for (i = 0; i < Tools.Data.StringHeadings.Length; i++)
523.              e.Row.Cells[i].Value = "Text";
524.          for (; i < Tools.Data.FieldsCount; i++)
525.              e.Row.Cells[i].Value = 0.0;
526.      }
527.
528.      /// <summary>
529.      /// Handles the RowPostPaint event of the dataTableGridView control.

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

530.          /// </summary>
531.          /// <param name="sender">The source of the event.</param>
532.          /// <param name="e">The <see cref="DataGridViewRowPostPaintEventArgs"/>
instance containing the event data.</param>
533.      private void dataTableGridView_RowPostPaint(object sender, DataGridViewR
owPostPaintEventArgs e)
534.      {
535.          // add indexes of rows
536.          if (!e.IsLastVisibleRow)
537.          {
538.              using (var b = new SolidBrush(dataTableGridView.RowHeadersDefaul
tCellStyle.ForeColor))
539.              {
540.                  e.Graphics.DrawString((e.RowIndex + 1).ToString(), e.Inherit
edRowStyle.Font, b,
541.                      e.RowBounds.Location.X + 10, e.RowBounds.Location.Y + 4)
;
542.              }
543.          }
544.      }
545.
546.          /// <summary>
547.          /// Handles the DataSourceChanged event of the dataTableGridView control
.
548.          /// </summary>
549.          /// <param name="sender">The source of the event.</param>
550.          /// <param name="e">The <see cref="EventArgs"/> instance containing the
event data.</param>
551.      private void dataTableGridView_DataSourceChanged(object sender, EventArg
s e)
552.      {
553.          for (var i = 0; i < dataTableGridView.ColumnCount; i++) dataTableGri
dView.Columns[i].SortMode = DataGridViewColumnSortMode.NotSortable;
554.      }
555.      #endregion
556.
557.      #region ClusterTableGridView
558.          /// <summary>
559.          /// Handles the DataSourceChanged event of the clusterTableGridView cont
rol.
560.          /// </summary>
561.          /// <param name="sender">The source of the event.</param>
562.          /// <param name="e">The <see cref="EventArgs"/> instance containing the
event data.</param>
563.      private void clusterTableGridView_DataSourceChanged(object sender, Event
Args e)
564.      {
565.          for (var i = 0; i < clusterTableGridView.ColumnCount; i++) clusterTa
bleGridView.Columns[i].SortMode = DataGridViewColumnSortMode.NotSortable;
566.      }
567.
568.          /// <summary>
569.          /// Handles the RowPostPaint event of the clusterTableGridView control.
570.          /// </summary>
571.          /// <param name="sender">The source of the event.</param>
572.          /// <param name="e">The <see cref="DataGridViewRowPostPaintEventArgs"/>
instance containing the event data.</param>
573.      private void clusterTableGridView_RowPostPaint(object sender, DataGridVi
ewRowPostPaintEventArgs e)
574.      {
575.          using (var b = new SolidBrush(clusterTableGridView.RowHeadersDefault
CellStyle.ForeColor))
576.          {
577.              e.Graphics.DrawString((e.RowIndex + 1).ToString(), e.InheritedRo
wStyle.Font, b,

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

578.                e.RowBounds.Location.X + 10, e.RowBounds.Location.Y + 4);
579.            }
580.        }
581.
582.        #endregion
583.
584.        #region ToolStrip
585.        /// <summary>
586.        /// Handles the Click event of the filterButton control.
587.        /// </summary>
588.        /// <param name="sender">The source of the event.</param>
589.        /// <param name="e">The <see cref="EventArgs"/> instance containing the
event data.</param>
590.        private void filterButton_Click(object sender, EventArgs e)
591.        {
592.            if (Tools.Data != null)
593.                Tools.Data.Filter(filterExpressionTextBox.Text, filteredColumnSe
lectComboBox.SelectedIndex,
594.                operandSelectComboBox.SelectedIndex);
595.        }
596.
597.        /// <summary>
598.        /// Handles the Click event of the sortByAZButton control.
599.        /// </summary>
600.        /// <param name="sender">The source of the event.</param>
601.        /// <param name="e">The <see cref="EventArgs"/> instance containing the
event data.</param>
602.        private void sortByAZButton_Click(object sender, EventArgs e)
603.        {
604.            if (Tools.Data != null)
605.                Tools.Data.Sort(sortColumnSelectComboBox.SelectedIndex, true);
606.        }
607.
608.        /// <summary>
609.        /// Handles the Click event of the sortByZAButton control.
610.        /// </summary>
611.        /// <param name="sender">The source of the event.</param>
612.        /// <param name="e">The <see cref="EventArgs"/> instance containing the
event data.</param>
613.        private void sortByZAButton_Click(object sender, EventArgs e)
614.        {
615.            if (Tools.Data != null)
616.                Tools.Data.Sort(sortColumnSelectComboBox.SelectedIndex, false);
617.        }
618.
619.        /// <summary>
620.        /// Handles the Click event of the undoButton control.
621.        /// </summary>
622.        /// <param name="sender">The source of the event.</param>
623.        /// <param name="e">The <see cref="EventArgs"/> instance containing the
event data.</param>
624.        private void undoButton_Click(object sender, EventArgs e)
625.        {
626.            if (Tools.Data != null)
627.                Tools.Data.Undo();
628.        }
629.
630.        /// <summary>
631.        /// Handles the Click event of the redoButton control.
632.        /// </summary>
633.        /// <param name="sender">The source of the event.</param>
634.        /// <param name="e">The <see cref="EventArgs"/> instance containing the
event data.</param>
635.        private void redoButton_Click(object sender, EventArgs e)
636.        {

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

637.         if (Tools.Data != null)
638.             Tools.Data.Redo();
639.     }
640.
641.     /// <summary>
642.     /// Handles the SelectedIndexChanged event of the filteredColumnSelectCo
        mboBox control.
643.     /// </summary>
644.     /// <param name="sender">The source of the event.</param>
645.     /// <param name="e">The <see cref="EventArgs"/> instance containing the
        event data.</param>
646.
647.     private void filteredColumnSelectComboBox_SelectedIndexChanged(object se
        nder, EventArgs e)
648.     {
649.         var selectedIndex = ((ToolStripComboBox)sender).SelectedIndex;
650.         if (selectedIndex >= Tools.StringDataHeadings.Length && Tools.Data !=
            null)
651.         {
652.             operandSelectComboBox.Items.Clear();
653.             operandSelectComboBox.Items.Add("=");
654.             operandSelectComboBox.Items.Add(">=");
655.             operandSelectComboBox.Items.Add("<=");
656.             operandSelectComboBox.Items.Add(">");
657.             operandSelectComboBox.Items.Add("<");
658.             operandSelectComboBox.SelectedIndex = 0;
659.         }
660.         else if (selectedIndex < Tools.StringDataHeadings.Length && Tools.Da
            ta != null)
661.         {
662.             operandSelectComboBox.Items.Clear();
663.             operandSelectComboBox.Items.Add("=");
664.             operandSelectComboBox.SelectedIndex = 0;
665.         }
666.     }
667.
668.
669.     #endregion
670.
671.     #region ClusteringToolStrip
672.     /// <summary>
673.     /// Handles the Click event of the searchButton control.
674.     /// </summary>
675.     /// <param name="sender">The source of the event.</param>
676.     /// <param name="e">The <see cref="EventArgs"/> instance containing the
        event data.</param>
677.     private void searchButton_Click(object sender, EventArgs e)
678.     {
679.         foreach (var row in clusterTableGridView.Rows.Cast<DataGridDataRow>(
            ))
680.         {
681.             row.Cells[0].Style.BackColor = Color.AliceBlue;
682.             if (((string)row.Cells[0].Value).Contains(searchExpressionTextBo
                x.Text))
683.                 row.Cells[0].Style.BackColor = Color.Aqua;
684.         }
685.     }
686.
687.     #endregion
688.
689.     #region MainForm
690.     /// <summary>
691.     /// Handles the FormClosing event of the MainForm control.
692.     /// </summary>
693.     /// <param name="sender">The source of the event.</param>

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

694.          /// <param name="e">The <see cref="FormClosingEventArgs"/> instance containing the event data.</param>
695.          private void MainForm_FormClosing(object sender, FormClosingEventArgs e)
696.          {
697.              DeleteTempFiles();
698.          }
699.          #endregion
700.
701.          #endregion
702.
703.      }
704.  }
705.  }

```

### 1.17. Node.cs

```

1.  using System;
2.  using System.Collections.Generic;
3.  using System.Linq;
4.  using System.Text;
5.  using System.Threading.Tasks;
6.
7.  namespace Clusterizer
8.  {
9.      /// <summary>
10.     /// Data structure of node
11.     /// </summary>
12.     /// <typeparam name="T">Item Type</typeparam>
13.     public class Node<T>
14.     {
15.         /// <summary>
16.         /// Gets or sets the contents.
17.         /// </summary>
18.         /// <value>
19.         /// The contents.
20.         /// </value>
21.         public T Contents { get; set; }
22.
23.         /// <summary>
24.         /// Gets or sets the children nodes.
25.         /// </summary>
26.         /// <value>
27.         /// The children nodes.
28.         /// </value>
29.         public List<Node<T>> ChildrenNodes { get; set; }
30.
31.         /// <summary>
32.         /// Initializes a new instance of the <see cref="Node{T}"/> class.
33.         /// </summary>
34.         /// <param name="contents">The contents.</param>
35.         public Node(T contents)
36.         {
37.             Contents = contents;
38.             ChildrenNodes = new List<Node<T>>();
39.         }
40.
41.         /// <summary>
42.         /// Initializes a new instance of the <see cref="Node{T}"/> class.
43.         /// </summary>
44.         /// <param name="child0">The child0.</param>
45.         /// <param name="child1">The child1.</param>
46.         public Node(Node<T> child0, Node<T> child1)
47.         {

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

48.         Contents = default(T);
49.
50.         ChildrenNodes = new List<Node<T>> {child0, child1};
51.     }
52. }
53. }

```

### 1.18. Program.cs

```

1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4. using System.Threading.Tasks;
5. using System.Windows.Forms;
6.
7. namespace Clusterizer
8. {
9.     static class Program
10.    {
11.        /// <summary>
12.        /// The main entry point for the application.
13.        /// </summary>
14.        [STAThread]
15.        static void Main()
16.        {
17.            Application.EnableVisualStyles();
18.            Application.SetCompatibleTextRenderingDefault(false);
19.            Application.SetUnhandledExceptionMode(UnhandledExceptionMode.CatchException
20.            );
21.            AppDomain.CurrentDomain.UnhandledException += CurrentDomain_UnhandledExcept
22.            ion;
23.            Application.ThreadException += Application_ThreadException;
24.            Application.Run(new MainForm());
25.        }
26.        /// <summary>
27.        /// Handles the ThreadException event of the Application control.
28.        /// </summary>
29.        /// <param name="sender">The source of the event.</param>
30.        /// <param name="e">The <see cref="System.Threading.ThreadExceptionEventArgs"/>
31.        instance containing the event data.</param>
32.        private static void Application_ThreadException(object sender, System.Threading
33.        .ThreadExceptionEventArgs e)
34.        {
35.            if (e.Exception is CustomException)
36.            {
37.                var exception = (CustomException)e.Exception;
38.                MessageBox.Show(exception.Text, exception.Caption, MessageBoxButtons.OK
39.                , MessageBoxIcon.Error);
40.            }
41.            else
42.            {
43.                var exception = e.Exception;
44.                MessageBox.Show(exception.Message, "Ошибка во время выполнения программ
45.                ы", MessageBoxButtons.OK,
46.                MessageBoxIcon.Error);
47.            }
48.        }
49.        /// <summary>
50.        /// Handles the UnhandledException event of the CurrentDomain control.
51.        /// </summary>
52.        /// <param name="sender">The source of the event.</param>

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

49.     /// <param name="e">The <see cref="UnhandledExceptionEventArgs"/> instance containing the event data.</param>
50.     private static void CurrentDomain_UnhandledException(object sender, UnhandledExceptionEventArgs e)
51.     {
52.         if (e.ExceptionObject is CustomException)
53.         {
54.             var exception = (CustomException) e.ExceptionObject;
55.             MessageBox.Show(exception.Text, exception.Caption, MessageBoxButtons.OK, MessageBoxIcon.Error);
56.         }
57.         else
58.         {
59.             var exception = (Exception) e.ExceptionObject;
60.             MessageBox.Show(exception.Message, "Ошибка во время выполнения программы", MessageBoxButtons.OK, MessageBoxIcon.Error);
61.         }
62.     }
63. }
64. }
65. }

```

### 1.19. StatisticsForm.cs

```

1. using System;
2. using System.Collections.Generic;
3. using System.ComponentModel;
4. using System.Data;
5. using System.Diagnostics;
6. using System.Drawing;
7. using System.Globalization;
8. using System.IO;
9. using System.Linq;
10. using System.Text;
11. using System.Threading.Tasks;
12. using System.Windows.Forms;
13.
14. namespace Clusterizer
15. {
16.
17.     /// <summary>
18.     /// Data structure for storing value with its index and groupID
19.     /// </summary>
20.     struct StatisticPoint
21.     {
22.         /// <summary>
23.         /// The group identifier
24.         /// </summary>
25.         public int GroupId;
26.
27.         /// <summary>
28.         /// The value
29.         /// </summary>
30.         public double value;
31.
32.         /// <summary>
33.         /// The index
34.         /// </summary>
35.         public int index;
36.     }
37.
38.     /// <summary>
39.     /// Form for presenting overview of clusters
40.     /// </summary>

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

41.    /// <seealso cref="System.Windows.Forms.Form" />
42.    public partial class StatisticsForm : Form
43.    {
44.        #region Fields
45.        /// <summary>
46.        /// The clusters
47.        /// </summary>
48.        internal ClusterSet Clusters;
49.
50.        /// <summary>
51.        /// The contents headings
52.        /// </summary>
53.        internal string[] contentsHeadings;
54.
55.        /// <summary>
56.        /// The data table
57.        /// </summary>
58.        private DataTable _dataTable;
59.
60.        /// <summary>
61.        /// The contents
62.        /// </summary>
63.        private StatisticPoint[][] _contents;
64.
65.        /// <summary>
66.        /// The default colors
67.        /// </summary>
68.        private Color[] _colors = new Color[]
69.        {
70.            Color.Aquamarine, Color.CornflowerBlue, Color.OrangeRed
71.        };
72.
73.        #endregion
74.
75.        #region Constructor
76.        /// <summary>
77.        /// Initializes a new instance of the <see cref="StatisticsForm"/> class.
78.        /// </summary>
79.        public StatisticsForm()
80.        {
81.            InitializeComponent();
82.        }
83.        #endregion
84.
85.        #region Methods
86.        /// <summary>
87.        /// Setups this instance.
88.        /// </summary>
89.        public void Setup()
90.        {
91.            // creates data tables
92.            _dataTable = new DataTable();
93.            // add columns to datatable
94.            _dataTable.Columns.Add("Название кластера");
95.            for (int i = 0; i < Tools.NumericDataHeadings.Length; i++)
96.                _dataTable.Columns.Add(Tools.NumericDataHeadings[i]);
97.            _dataTable.Columns.Add("Число объектов");
98.            _contents = new StatisticPoint[contentsHeadings.Length][];
99.
100.            // fill data table
101.            for (int i = 0; i < Clusters.Count; i++)
102.            {
103.                var list = Clusters.GetCluster(i).Centroid.Points.Select(x => $"
{x}").ToList();
104.                list.Insert(0, $"Cluster{Clusters.GetCluster(i).Id}");
105.                list.Add($"{{Clusters.GetCluster(i).QuantityOfDataPoints}}");

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата



```

106.         _dataTable.Rows.Add(list.ToArray());
107.     }
108.
109.     // init contents
110.     for (int j = 0; j < contentsHeadings.Length; j++)
111.     {
112.         _contents[j] = new StatisticPoint[Clusters.Count];
113.     }
114.
115.     // gets contents
116.     for (int i = 0; i < Clusters.Count; i++)
117.     {
118.         for (int j = 0; j < contentsHeadings.Length; j++)
119.         {
120.             _contents[j][i] = new StatisticPoint();
121.             _contents[j][i].value = Clusters.GetCluster(i).Centroid[j];
122.
123.             _contents[j][i].GroupId = 0;
124.             _contents[j][i].index = i;
125.         }
126.     }
127.
128.     // group contents
129.     for (int j = 0; j < contentsHeadings.Length; j++)
130.     {
131.         _contents[j] = _contents[j].OrderBy((x) => x.value).ToArray();
132.         var points = _contents[j].Select(x => x.value).ToArray();
133.         var indentifiers = Tools.GroupBy(points);
134.         for (int i = 0; i < _contents[j].Length; i++)
135.             _contents[j][i].GroupId = indentifiers[i];
136.         _contents[j] = _contents[j].OrderBy((x) => x.index).ToArray();
137.     }
138.
139.     // fill data gridview
140.     clustersOverviewGridView.ColumnHeadersVisible = true;
141.     clustersOverviewGridView.AutoSizeColumnsMode = DataGridViewAutoSizeC
142.     olumnsMode.AllCells;
143.     clustersOverviewGridView.DataSource = _dataTable;
144. }
145. #endregion
146.
147. #region Events
148. /// <summary>
149. /// Handles the FormClosing event of the StatisticsForm control.
150. /// </summary>
151. /// <param name="sender">The source of the event.</param>
152. /// <param name="e">The <see cref="FormClosingEventArgs"> instance cont
153. aining the event data.</param>
154. private void StatisticsForm_FormClosing(object sender, FormClosingEventA
155. rgs e)
156. {
157.     using (FileStream fileStream = new FileStream("tmpOverview.csv", Fil
158. eMode.Create))
159.     {
160.         StreamWriter streamWriter = new StreamWriter(fileStream);
161.         for (int i = 0; i < Clusters.Count; i++)
162.         {
163.             var tmpContent = Clusters[i].Centroid.Points.Select(x => $"{
164. x}").ToList();
165.             tmpContent.Insert(0, $"Clusters{Clusters[i].Id}");
166.             tmpContent.Add($"Clusters[i].QuantityOfDataPoints");
167.             streamWriter.WriteLine(string.Join(";", tmpContent));
168.         }
169.         streamWriter.Flush();
170.     }
171. }

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

166.
167.         /// <summary>
168.         /// Handles the Load event of the StatisticsForm control.
169.         /// </summary>
170.         /// <param name="sender">The source of the event.</param>
171.         /// <param name="e">The <see cref="EventArgs"/> instance containing the
event data.</param>
172.         private void StatisticsForm_Load(object sender, EventArgs e)
173.         {
174.             for (int i = 0; i < Clusters.Count; i++)
175.                 for (int j = 0; j < contentsHeadings.Length; j++)
176.                     clustersOverviewGridView.Rows[i].Cells[j + 1].Style.BackColor =
_colors[_contents[j][i].GroupId - 1];
177.         }
178.
179.         /// <summary>
180.         /// Handles the RowPostPaint event of the clustersOverviewGridView contr
ol.
181.         /// </summary>
182.         /// <param name="sender">The source of the event.</param>
183.         /// <param name="e">The <see cref="DataGridViewRowPostPaintEventArgs"/>
instance containing the event data.</param>
184.         private void clustersOverviewGridView_RowPostPaint(object sender, DataGr
idViewRowPostPaintEventArgs e)
185.         {
186.             using (SolidBrush b = new SolidBrush(clustersOverviewGridView.RowHea
dersDefaultCellStyle.ForeColor))
187.             {
188.                 e.Graphics.DrawString((e.RowIndex + 1).ToString(), e.InheritedRo
wStyle.Font, b, e.RowBounds.Location.X + 10, e.RowBounds.Location.Y + 4);
189.             }
190.         }
191.
192.         /// <summary>
193.         /// Handles the DataSourceChanged event of the clustersOverviewGridView
control.
194.         /// </summary>
195.         /// <param name="sender">The source of the event.</param>
196.         /// <param name="e">The <see cref="EventArgs"/> instance containing the
event data.</param>
197.         private void clustersOverviewGridView_DataSourceChanged(object sender, E
ventArgs e)
198.         {
199.             for (int i = 0; i < clustersOverviewGridView.ColumnCount; i++)
200.             {
201.                 clustersOverviewGridView.Columns[i].SortMode = DataGridViewColum
nSortMode.NotSortable;
202.             }
203.         }
204.
205.         #endregion
206.     }
207.

```

## 1.20. Tools.cs

```

1. using System;
2. using System.IO;
3. using System.Linq;
4. using System.Windows.Forms;
5. using System.Xml;
6. using System.Xml.Serialization;
7.
8. namespace Clusterizer

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

9. {
10.    /// <summary>
11.    /// Static class for app wide used settings, methods and data
12.    /// </summary>
13.    public static class Tools
14.    {
15.        #region User-Defined Data
16.
17.        /// <summary>
18.        /// The string data headings
19.        /// </summary>
20.        public static string[] StringDataHeadings;
21.
22.        /// <summary>
23.        /// The numeric data headings
24.        /// </summary>
25.        public static string[] NumericDataHeadings;
26.
27.        /// <summary>
28.        /// The group names
29.        /// </summary>
30.        public static string[] GroupNames;
31.
32.        /// <summary>
33.        /// The group items count
34.        /// </summary>
35.        public static int[] GroupItemsCount;
36.        #endregion
37.
38.        #region App Data
39.        /// <summary>
40.        /// The group items names
41.        /// </summary>
42.        public static string[][] GroupItemsNames;
43.
44.        /// <summary>
45.        /// The group items indexes
46.        /// </summary>
47.        public static int[][] GroupItemsIndexes;
48.
49.        /// <summary>
50.        /// The is chosen
51.        /// </summary>
52.        public static bool[] isChosen;
53.
54.        /// <summary>
55.        /// The data
56.        /// </summary>
57.        public static CSVData Data;
58.
59.        /// <summary>
60.        /// The clusters
61.        /// </summary>
62.        public static ClusterSet Clusters;
63.        #endregion
64.
65.        #region Methods
66.        /// <summary>
67.        /// Minimums-maximum normalization.
68.        /// </summary>
69.        /// <param name="arr">The arr.</param>
70.        public static void MinMaxNormalize(ref double[] arr)
71.        {
72.            var max = arr.Max();
73.            var min = arr.Min();
74.

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

75.         for (var i = 0; i < arr.Length; i++)
76.             arr[i] = (arr[i] - min) / (max - min);
77.     }
78.
79.     /// <summary>
80.     /// Z-Score normalization.
81.     /// </summary>
82.     /// <param name="arr">The arr.</param>
83.     public static void ZScoreNormalize(ref double[] arr)
84.     {
85.         var mean = arr.Sum() / arr.Length;
86.         double bigSum = 0;
87.         foreach (var d in arr) bigSum += Math.Pow(d - mean, 2);
88.
89.         var standartDeviation = Math.Sqrt(bigSum / (arr.Length - 1));
90.
91.         for (var i = 0; i < arr.Length; i++)
92.             arr[i] = (arr[i] - mean) / standartDeviation;
93.     }
94.
95.     /// <summary>
96.     /// Gets the standart deviation.
97.     /// </summary>
98.     /// <param name="points">The points.</param>
99.     /// <returns>Standart deviation of given points</returns>
100.    public static double GetStandartDeviation(double[] points)
101.    {
102.        var max = points.Max();
103.        var min = points.Min();
104.        var avg = (max + min) / 2;
105.        double sum = 0;
106.        foreach (var point in points) sum += Math.Pow(point - avg, 2);
107.
108.        return Math.Sqrt(sum / (points.Length - 1));
109.    }
110.
111.    /// <summary>
112.    /// Groups sequence of points to three subsequences using minimal differ
113.    /// ence in standart deviation
114.    /// </summary>
115.    /// <param name="points">The points.</param>
116.    /// <returns>Grouped indexes</returns>
117.    public static int[] GroupBy(double[] points)
118.    {
119.        // group indexes
120.        var groupIds = new int[points.Length];
121.        var newPoints = points;
122.        MinMaxNormalize(ref newPoints);
123.
124.        var min = double.MaxValue;
125.        int mi = 1, mj = newPoints.Length - 2;
126.
127.        // custom cases
128.        if (points.Length == 1)
129.        {
130.            groupIds[0] = 1;
131.        }
132.        else if (points.Length == 2)
133.        {
134.            groupIds[0] = 1;
135.            groupIds[1] = 2;
136.        }
137.        else if (points.Length == 3)
138.        {
139.            groupIds[0] = 1;
140.            groupIds[1] = 2;

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

140.         groupIds[2] = 3;
141.     }
142.     else
143.     {
144.         // overall case
145.         for (var i = 1; i < newPoints.Length - 2; i++)
146.             for (var j = i + 1; j < newPoints.Length - 1; j++)
147.             {
148.                 // get all possible three subsequences
149.                 var one = newPoints.Slice(0, i);
150.                 var two = newPoints.Slice(i, j);
151.                 var three = newPoints.Slice(j, newPoints.Length);
152.
153.                 // compute sum of standart deviations
154.                 var avg = GetStandartDeviation(one) + GetStandartDeviati
on(two) + GetStandartDeviation(three);
155.                 // get minimum one
156.                 if (avg < min)
157.                 {
158.                     min = avg;
159.                     mi = i;
160.                     mj = j;
161.                 }
162.             }
163.
164.         // setting group ids
165.         for (var i = 0; i < newPoints.Length; i++)
166.             if (i < mi)
167.                 groupIds[i] = 1;
168.             else if (i >= mi && i < mj)
169.                 groupIds[i] = 2;
170.             else
171.                 groupIds[i] = 3;
172.     }
173.
174.     return groupIds;
175. }
176.
177. /// <summary>
178. /// Slices array from the specified index to other index.
179. /// </summary>
180. /// <typeparam name="T"></typeparam>
181. /// <param name="arr">The arr.</param>
182. /// <param name="indexFrom">The index from.</param>
183. /// <param name="indexTo">The index to.</param>
184. /// <returns>Sliced array</returns>
185. public static T[] Slice<T>(this T[] arr, int indexFrom, int indexTo)
186. {
187.     var length = indexTo - indexFrom;
188.     var result = new T[length];
189.     Array.Copy(arr, indexFrom, result, 0, length);
190.
191.     return result;
192. }
193.
194. /// <summary>
195. /// Saves current settings to config file.
196. /// </summary>
197. public static void Save()
198. {
199.     XmlSerializer xmlSerializer = new XmlSerializer(typeof(Configuration
));
200.     Configuration configuration = new Configuration();
201.     configuration.StringHeadings = StringDataHeadings;
202.     configuration.NumericHeadings = NumericDataHeadings;
203.     configuration.GroupNames = GroupNames;

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

204.         configuration.GroupItemsCount = GroupItemsCount;
205.         xmlSerializer.Serialize(new FileStream("dataconfig.xml", FileMode.Cr
eate), configuration);
206.     }
207.
208.         /// <summary>
209.         /// Loads current settings from config file
210.         /// </summary>
211.         /// <exception cref="Clusterizer.CustomException">Конфигурационный файл
dataconfig.xml поврежден. -
        Произошла ошибка при чтении конфигурационного файла.</exception>
212.         public static void Load()
213.         {
214.             try
215.             {
216.                 using (FileStream fileStream = new FileStream("dataconfig.xml",
FileMode.Open))
217.                 {
218.                     XmlSerializer xmlSerializer = new XmlSerializer(typeof(Confi
guration));
219.                     Configuration configuration = (Configuration) xmlSerializer.
Deserialize(fileStream);
220.                     Tools.StringDataHeadings = configuration.StringHeadings;
221.                     Tools.NumericDataHeadings = configuration.NumericHeadings;
222.                     Tools.GroupNames = configuration.GroupNames;
223.                     Tools.GroupItemsCount = configuration.GroupItemsCount;
224.
225.                     // Generates group Items
226.                     var groupCount = GroupNames.Length;
227.                     GroupItemsNames = new string[groupCount][];
228.                     GroupItemsIndexes = new int[groupCount][];
229.
230.                     var ind = 0;
231.                     for (var i = 0; i < groupCount; i++)
232.                     {
233.                         var count = GroupItemsCount[i];
234.                         GroupItemsNames[i] = new string[count];
235.                         GroupItemsIndexes[i] = new int[count];
236.                         for (var j = 0; j < count; j++)
237.                         {
238.                             GroupItemsIndexes[i][j] = ind;
239.                             GroupItemsNames[i][j] = NumericDataHeadings[ind];
240.                             ind++;
241.                         }
242.                     }
243.                 }
244.
245.             }
246.             catch
247.             {
248.                 throw new CustomException("Конфигурационный файл dataconfig.xml
поврежден.", "Произошла ошибка при чтении конфигурационного файла.");
249.             }
250.         }
251.     #endregion
252.
253. }
254. }

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

## 2. СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

- 1) ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
- 2) ГОСТ 19.102-77 Стадии разработки. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
- 3) ГОСТ 19.103-77 Обозначения программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
- 4) ГОСТ 19.104-78 Основные надписи. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
- 5) ГОСТ 19.105-78 Общие требования к программным документам. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
- 6) ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
- 7) ГОСТ 19.401-78 Текст программы. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
- 8) ГОСТ 19.603-78 Общие правила внесения изменений. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
- 9) ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

[illegible]

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13 12				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата